Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

SIMSPIRE: A Simulator of the Respiratory System

ANDREA BOMBARDA[1], (Member, IEEE), SILVIA BONFANTI[1], (Member, IEEE), ANGELO GARGANTINI[1], (Member, IEEE), and ELVINIA RICCOBENE[2], (Member, IEEE)

[1]Department of Management, Information and Production Engineering, University of Bergamo, Bergamo, Italy (e-mail: {andrea.bombarda, silvia.bonfanti, angelo.gargantini}@unibg.it)

[2]Università degli Studi di Milano, Milan, Italy (e-mail: elvinia.riccobene@unimi.it)

Corresponding author: Elvinia Riccobene (e-mail: elvinia.riccobene@unimi.it).

ABSTRACT This paper introduces SIMSPIRE, a comprehensive simulator of the human respiratory system designed as a versatile tool for simulating various patient conditions. It can be used to model patients with a wide range of physiological traits, such as varying lung compliance and resistance, as well as respiratory conditions, such as chronic obstructive pulmonary disease (COPD) and acute respiratory distress syndrome (ARDS). The versatility of SIMSPIRE makes it suitable for simulating diverse respiratory scenarios and evaluating new algorithms or control strategies for ventilatory support. For this work, we have tested and evaluated our simulator by connecting as input the control software of a ventilator prototype. SIMSPIRE represents a significant advancement in the field of respiratory simulation. It provides a highly accurate and customizable platform for improving the design of ventilation systems and optimizing patient outcomes. By eliminating the need for expensive hardware simulators, SIMSPIRE has the potential to accelerate innovation and contribute to enhanced patient care in critical care settings.

INDEX TERMS Human system modeling, mechanical ventilator testing, medical software certification, medical software testing, respiratory system, simulator, test-driven development.

I. INTRODUCTION

RESPIRATORY failure is a critical condition that requires immediate and accurate treatment to ensure patient survival. Mechanical ventilation is commonly used to support patients with the most critical types of failure, such as chronic obstructive pulmonary disease (COPD) or acute respiratory distress syndrome (ARDS). In recent years, especially owing to the COVID-19 pandemic, considerable effort and resources have been spent on the development of mechanical ventilators. However, the development and testing of mechanical ventilators can be challenging, especially because of the limited availability of accessible lung simulators and the inability to test them on humans. This challenge is even higher when the development is conducted in a distributed fashion and environment, such as reported by the developers involved in the Mechanical Ventilator Milano (MVM) project [1] during the first wave of COVID-19 in Europe. In particular, during that project, considering that all the developers were in lockdown, spread all over the world, and a limited quantity of ventilator prototypes and physical lung simulators were available, only some of them could test the software on the actual ventilator hardware and with a physical lung simulator [2], [3].

Testing medical software, including that embedded in mechanical ventilators, is not only recommended to ensure proper device function, but also mandated to obtain software certification according to the IEC 62304 standard [4]. It is essential to ensure the quality of the devices, as any potential for injury must be ruled out beyond any reasonable doubt. Furthermore, under emergency conditions (such as those reported by MVM developers), using agile practices [5], e.g., test-driven development (TDD) is advisable and can contribute to making the development and certification process faster. Thus, the testing activity for a medical device is paramount important.

However, applying TDD and effectively testing a medical device requires both device hardware and a patient (or an equivalent simulator, such as a manikin) to be available, in addition to the external environment. Indeed, in any medical device, an important part of the behavior may be triggered only by external events (e.g., detachment of the tube for a ventilator or movement of a patient in an X-ray machine). In recent years, researchers have devoted considerable attention to developing various digital twins, models, and simulators for a diverse range of human systems and organs. These efforts (as seen in studies like [6]–[9]) aim to create precise representations that accurately capture the behavior of specific body parts. They are commonly used for several objectives, such as testing medical devices, acting as predictive models for the medical device's adaptation, generating synthetic data to be used for training medical systems or designing personalized drugs. However, as highlighted by MVM developers, during the development of a mechanical ventilator system, no suitable and easily configurable respiratory system simulator is available for use in software testing and development.

In this paper, we present SIMSPIRE (SIMulator of the reSPIRatory systEm), a simulator of the human respiratory system that can be configured to reproduce a range of patients with various physiological characteristics and different diseases, and that can be easily integrated with the software of external devices (e.g., mechanical ventilators). SIMSPIRE only requires to instrument the device source code to include ZeroMQ interfaces substituting sensors reading (e.g., respiratory rate and pressure measurement) and actuators output (e.g., valve openings and closures). We also show the setup we used to test SIMSPIRE with inputs provided by the control software of a prototype of a mechanical ventilator.

Compared to existing simulators (see Table 1), SIMSPIRE has some advantages in several aspects. It offers a domain-specific language (DSL), to specify patients with various health conditions at increasing levels of detail. This makes the simulator user-friendly and highly configurable, while others are complex and require medical knowledge, making them less accessible to non-experts like developers and testers. Thanks to the ZeroMQ interfaces and the bidirectional communication capability, it can be easily integrated with ventilator software. Moreover, the external interface makes it versatile and easily interfaced with other applications, such as those for training healthcare personnel. These advantages can be leveraged by development teams for the following primary purposes:

Reliability: Developers can easily introduce specific patient health conditions and events related to ventilation, and test their ventilation algorithms to gain confidence in their validity and correctness.

Cost reduction: Currently, physical devices are necessary to simulate the various patient conditions for testing mechanical ventilators. The high cost of these devices significantly impacts the final price of the ventilators, which is particularly disadvantageous for poorer countries that cannot afford such high expenses. By introducing software to simulate patient conditions, the cost of testing mechanical ventilators can be reduced, thereby keeping the overall device costs low.

Remote development: Since the 2020 pandemic, remote working has become more prevalent, limiting workers' access to physical devices needed for their tasks. Testing mechanical ventilators typically requires being on-site, which is not always feasible. Software that emulates physical devices allows for remote testing of mechanical ventilators, enabling testing whenever necessary.

To the best of our knowledge, SIMSPIRE is the only available software simulator allowing the connection with ventilator software, thanks to the ZeroMQ interfaces.

The paper is organized as follows. Section II presents the MVM case study that inspired our work and background on how the human respiratory system can be modeled using electrical components. Section III outlines our effort in defining a generic respiratory system model, and the Domain-Specific Language (DSL) we developed to express patient models and conditions. In Section IV, we introduce the SIMSPIRE model of the respiratory system, including its architecture and how it communicates with the software of external devices, such as mechanical ventilators. Section V details the process we have followed to test SIMSPIRE with a control software of a mechanical ventilator, while Section VI presents related works on using digital twins, models, and simulators in healthcare and medical software development. Finally, Section VII presents some future work, and Section VIII concludes the paper.

II. BACKGROUND

In this section, we introduce the background of our work. In particular, first, we report some of the details of the MVM case study that inspired our work. Then, we explain the basic principles exploited when modeling a respiratory system, we show the already available models and discuss their limitations.

A. THE MVM CASE STUDY

The Mechanical Ventilation Milano (MVM) [1] was developed during the COVID-19 pandemic as a pressure-regulated ventilator for ICU patients. It has been designed to provide two forms of ventilation, namely Pressure Controlled Ventilation (PCV) and Pressure Support Ventilation (PSV), providing crucial support for patients with varying levels of respiratory function. In situations where the patient is unable to initiate breathing independently, PCV is employed. The physician establishes a consistent respiratory cycle during ventilation, with pressure variation between the desired inspiratory pressure and positive end-expiratory pressure (PEEP). On the other hand, the PSV mode is recommended for patients who can control their breathing to some extent but still require support. This mode utilizes MVM to assist with breathing, with a new breath triggered by a pressure drop and expiration beginning when the patient's inspiratory flow falls below a certain fraction of peak flow. If no breath is detected within the apnea delay set by the physician, the ventilator switches to PCV mode, indicating the patient is unable to breathe independently.

MVM development took place in a distributed fashion, with much of the work conducted remotely due to lockdown restrictions. As reported by researchers participating in the development process, this situation underscored a critical challenge: the need to simulate the human respiratory system and link this simulation with the ventilator software to ensure effective development and testing [2]. This is the rationale behind the work we present in this paper.

Simulator	Reference	Type	Access	Purpose	APIs
Xlung	https://xlung.net/	Web-based	Subscription only	Training	No
VenTrainer	https://www.hamilton-medical.com/en_US/Academy/VenTrainer.html	Local+Mobile app	Free	Training	No
LungSim	https://accuratesolutions.it/en/product-details/lungsim-ventilator-simulator-paired-with-patient-simulator/	Local+Mobile app	Subscription only	Training	No
SimVA	https://www.sim-va.com/	Mobile App	Free	Training	No
VentSim	https://ventsim.cc/	Web-based	Free	Training	No
OPENPediatrics	https://learn.openpediatrics.org/	Web-based	Free subscription	Training	No
SIMSPIRE	https://github.com/foselab/SIMSPIRE	Local+Web	Open source	SW development and testing	Yes

Table 1: Simulators of the human respiratory system

B. HOW TO MODEL THE RESPIRATORY SYSTEM

Over the years, several attempts to describe with a model the human respiratory system and the way it works have been made. In general, as shown in [10], the electrical domain is well suited for translating mechanical models, such as those of the lungs, thanks to the analogies between the current in a circuit and the airflow in the respiratory system. Indeed, the majority of these models are composed of a set of resistances, capacitors, and generators, but a few of them include inductors and switches as well.

Resistors are used to represent the airways and the resistance that air encounters as it moves through the lungs. The resistance of the airways can be affected by many factors, including inflammation and constriction due to diseases such as acute respiratory distress syndrome (ARDS) and chronic obstructive pulmonary disease (COPD). In ARDS, the airways become inflamed and filled with fluid, which increases resistance to airflow. COPD is characterized by a narrowing of the airways due to inflammation and mucus production, which also increases resistance.

Capacitors are used to represent the elasticity of the lung tissue. The lungs are like a balloon that inflates and deflates with each breath, and the elasticity of the lung tissue allows it to stretch and recoil. Diseases such as ARDS and COPD can cause damage to the lung tissue and reduce its elasticity, which can be represented in the model by changing the capacitance.

C. EXISTING LUNG MODELS

For the development of SIMSPIRE, we took into account some time-dependent linear models of which a brief description is given in the following.

One of the simplest and oldest models was proposed by Campbell and Brown [11] in the 60s. The circuit, reported in Figure 1, consists of four elements connected in series representing respectively the endotracheal tube resistance (R_1 in Figure 1), the tracheo-bronchial resistance (R_2 in Figure 1), the lung compliance (C_1 in Figure 1), and the chest wall compliance (C_2 in Figure 1). To complete the circuit a voltage generator is attached to the edges of this segment. These elements are constants and their values were derived from actual patients under treatment at that time. Since these

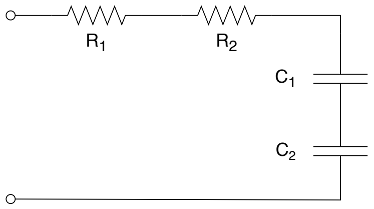


Figure 1: The Campbell and Brown model for the respiratory system

patients had different health conditions like obstructive lung disease or pulmonary fibrosis, the model can be simulated for a range of diseases. A decade later, Jain and Guha adapted the model of Campbell and Brown as a three elements circuit [12] representing respectively the upper airway resistance, the lung tissues resistance, and the compliance of the whole respiratory system. Although the model does not contain data related to specific diseases, it underlines the importance of considering the variability of the values that resistors and capacitors can take in the circuit. Given a set of patients with the same health condition, it is improbable that the elements of the circuit will have the same value. So, they provided a table showing the range in which the different elements of the circuit can vary. At the same time, Baker and Hahn [13] proposed a slightly more complex model representing the lung/thorax system with only linear components. Recent advances in lung modeling have been reported in [14]. Although these electic models are quite simple, they give results which are comparable to real measurements from healthy, asthma, or chronic obstructive pulmonary disease subjects and changes in the morphology [15].

The models described so far look at the respiratory system as a single block without going into much detail because of the limited medical knowledge of the time. However, thanks to technological progress over the decades, Albanese et al. [16] were able to design a detailed model of the respiratory system, here reported in Figure 2. The model divides the system into four parts: larynx, trachea, bronchi, and alveoli. Each of these elements is represented by a linear resistance and a linear compliance. More specifically, in Figure 2, R_{ml} and C_l represent the larynx, R_{lt} and C_{tr} represent the trachea,

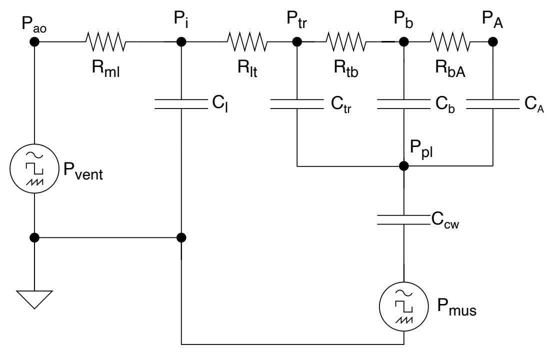


Figure 2: The Albanese model for the respiratory system

R_{tb} and C_b represent the bronchi, and R_{bA} and C_A represent the alveoli. To allow the simulation of both a spontaneous and artificial breathing condition, the model includes two different voltage generators, the combination of which determines the patient's state: P_{vent} takes into account the pressure set by an external mechanical ventilator, while P_{mus} represents the respiratory pressure due to patient's respiratory muscles.

Finally, Al-Naggar proposed a multi-compartment model [17] which focuses on small detail behaviors of respiratory system anatomy. The model divides the airways into two sections, the central and the peripheral, each of which is represented by a resistance. The air that enters the alveoli produces the expansion of the chest-wall cavity whose behavior is given by the connection in series of the lung and chest-wall compliances. Since a small amount of air is shunted away from the alveoli, a capacitor is placed in parallel to the other two compliances. The presence of this element helps to analyze different diseases because this volume of air varies according to the specific patient's condition.

III. DEVISING A GENERIC RESPIRATORY SYSTEM MODEL

In Section II, we have introduced the lung models available in the literature, which can include from a very limited set of features to a wide representation of the respiratory system. However, utilizing these models for simulating the human respiratory system is not always practical, as they rely on fixed values and cannot simulate patients in varying conditions. In fact, when working with medical devices, it is crucial not only to assess their performance with static patients but also to evaluate the safety and effectiveness of the devices as the patient's health status changes over time. Furthermore, in some of the analyzed models, many values are not explained and it is very difficult to identify whether they are constant or can change depending on the patient.

For this reason, the objective of this paper is to propose a respiratory system model that can be easily extended for different patients and reused to simulate disparate possible diseases. In this way, the generic respiratory system model can be personalized depending on the level of accuracy or testing thoroughness needed for the system under test. Like the other models available in the literature, we propose to exploit the electrical analog of the respiratory system and to support every kind of lung model that can be described using a custom-made DSL, as explained in the following.

A. A DSL TO DEFINE LUNG MODELS

In order to describe each model and allow the customization of our simulator's behavior, a Domain Specific Language was defined. This DSL is used to define a set of three different YAML files which represent respectively (1) the circuit model, (2) the archetype, and (3) the demographic patient data. We emphasize that for the sake of brevity, we here report only excerpts of the files containing patients' descriptions, but all of them are available in our replication package at https://github.com/foselab/SIMSPIRE.

Each (1) circuit model must have a **`schemaid`** and an **`elementsList`** reporting all the elements of the circuit. Each element is defined by the following set of attributes:

- **`elementName`**: a description of the element;
- **`id`**: a string that aims to clearly describe and identify the considered element (e.g., with the acronym used in the schematic of the circuit);
- **`associatedFormula`**: the fomula to compute its value, with the following sub attributes;
- **`isTimeDependent`**: to tell if the value of the element is time-dependent;
- **`isExternal`**: to tell if the value of the element is computed or given by an external source. For example, considering a scenario in which an external mechanical ventilator is attached to the patient, the component that represents the ventilator is external because its value is not computed by our tool;
- **`formula`**: the formula used to compute the element's value;
- **`variables`**: a list that specifies all the variables contained in the formula. If the element's value is time-dependent, the temporal variable has to be written as TIME;
- **`type`**: a string identifying the element's class. For instance, resistance will be reported as ResistorElm as described in the documentation available in our replication package (see Section IV);
- **`position`**: the assignment of the coordinates (x and y) is done by placing the circuit on a Cartesian plane and associating to each node an abscissa and an ordinate. These coordinates are used to define the connections between different components.

For example, considering the Campbell and Brown model shown in Figure 1, an excerpt of the YAML file describing the structure of the circuit is reported in Listing 1. In this excerpt, we report two components, a resistor **`R1`** and a capacitor **`C1`**, whose value is defined by a single variable each. Considering that the Campbell and Brown model does not require a patient to dynamically evolve its condition, both components are not time-dependent and are not defined by external components. Finally, given the specified coordinates, **`R1`** and **`C1`** are connected in the point $\langle x = 1, y = 1\rangle$. An additional example of

```
schemaid: 2
elementsList:
- elementName: Lung Resistance (1)
  id: R1
  associatedFormula:
    isTimeDependent: false
    isExternal: false
    formula: resistance1
    variables:
    - resistance1
  type: ResistorElm
  position: {x1: 0, y1: 1, x2: 1, y2: 1}
- elementName: Lung Compliance (1)
  id: C1
  associatedFormula:
    isTimeDependent: false
    isExternal: false
    formula: capacitor1
    variables:
    - capacitor1
  type: CapacitorElm
  position: {x1: 2, y1: 1, x2: 3, y2: 1}
[...]
```

Listing 1: Excerpt of the YAML file describing the structure of a circuit for the Campbell and Brown model

```
schemaid: 3
elementsList:
- elementName:
  id: Rml
  associatedFormula:
    isTimeDependent: true
    isExternal: false
    formula: resistanceMl * (1 + TIME/1000)
    variables:
    - resistanceMl
  type: ResistorElm
  position: {x1: 0, y1: 2, x2: 1, y2: 2}
- elementName:
  id: Pmus
  associatedFormula:
    isTimeDependent: true
    isExternal: false
    formula: muscolarPressure * sin(TIME)
    variables:
    - muscolarPressure
  type: VoltageElm
  position: {x1: 1, y1: 0, x2: 0, y2: 2}
[...]
```

Listing 2: Excerpt of the YAML file describing the structure of a circuit for the Albanese model

YAML file describing a circuit for the Albanese analogue of the respiratory system (see Figure 2) is reported in Listing 2. In this case, the excerpt we report contains two elements depending on the time, namely a resistance and a voltage generator. The former is the resistance R_{ml} which represents the resistance encountered by the air when flowing from mouth to larynx. By making it time-dependent, we model a patient for which the resistance is increasing, i.e., a patient having obstructive diseases. The latter is a voltage generator P_{mus} that represents the muscular pressure that allows spontaneous breathing of the patient. In this example, we set the muscular pressure to behave as a sinusoid, but any other waveforms can be used.

The second input file is (2) the archetype of the patient. In this file, we define the characteristics of the patient, i.e., we give the value to the variables defined in the file previously discussed. It is defined by a matching **`schemaid`** of the circuit model and the list of all the known **`variables`**' values needed to compute the circuit values. Different input values will lead to the representation of different diseases. An example of the YAML file specifying the archetype for a patient in healthy condition, represented with the Campbell and Brown model (see Figure 1), is shown in Listing 3. Note that Listing 1 and Listing 3 report the same schema **`id`**, since they are referred to the same model.

Finally, the last input file contains some demographic data about the patient as **`gender`**, **`age`**, **`height`**, and **`weight`**. An example of the content of this file, in the case of a 75-year-old female patient who is 1.9m in height, and weighs 100 kg, is reported in Listing 4.

Note that, by using the three YAML files previously explained, our DSL can be used to model patients with a different level of accuracy and complexity (depending on the chosen circuit model) and with possibly different diseases (depending on the archetype file). For example, by setting different values for the resistances and the compliances in the circuit, patients with chronic obstructive pulmonary disease (COPD) or acute respiratory distress syndrome (ARDS) can be easily modeled.

```
patient: healthy patient
schemaid: 2
variables:
    resistance1: 8.0
    resistance2: 2.0
    capacitor1: 0.010
    capacitor2: 0.15
```

Listing 3: Archetype of an healthy patient for the Campbell and Brown model

IV. THE SIMSPIRE TOOL

In this section, we present SIMSPIRE, a configurable simulator of the human respiratory system. SIMSPIRE takes as input the three YAML files described in Section III and simulates the electrical circuit, which is the analogue of the human respiratory system, by assigning element values as requested by the user. It allows for connecting external ventilator software to test its correct functioning. The tool

```
gender: female
age: 75
height: 1.9
weight: 100
```

Listing 4: Demographic data for a patient

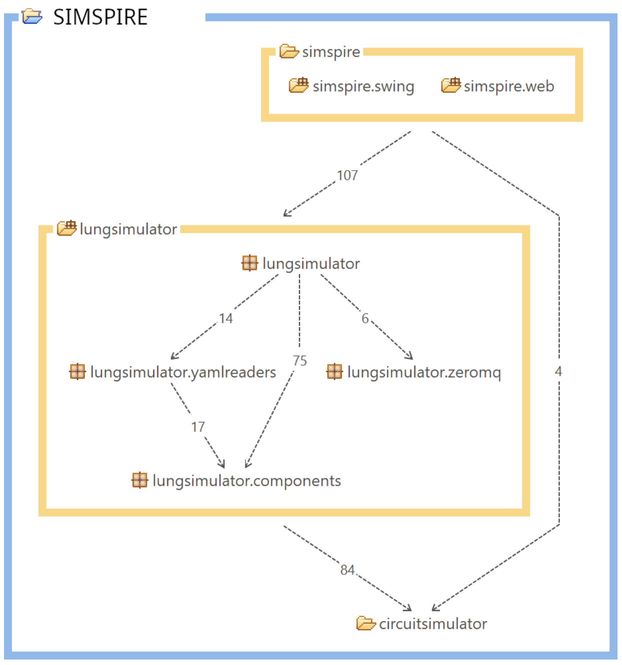


Figure 3: The architecture of the SIMSPIRE tool

is open-source and is available in our replication package https://github.com/foselab/inspire/, together with the relevant documentation, describing its use and configuration.

A. TOOL ARCHITECTURE

The architecture of SIMSPIRE is reported in Figure 3. As described in Section III, the human respiratory system can be modeled using the corresponding electrical analogy. For this reason, an important part of SIMSPIRE is the circuitsimulator based on Paul Falstad's work[1], which has been ported to Java and adapted to remove the parts used for the graphical representation of circuit components.

To accurately simulate a patient's respiratory system, SIMSPIRE may need to interact with external devices, exchanging patient parameters and receiving the settings from these devices to apply to the electrical circuit. For instance, when simulating a patient on mechanical ventilation, SIMSPIRE will send pressure and flow data to the ventilator's software while receiving input pressure from the ventilator. This process is supported by the functionalities offered by the lungsimulator.zeromq package, which use ZeroMQ[2], an open-source universal messaging library allowing the implementation of the most common communication patterns, such as publisher-subscriber and request-reply.

Then, the functioning of the simulator can be checked using two different and alternative GUIs, one working as a normal Swing app (see Figure 4a), i.e., simspire.swing, and one executed inside a web browser and implemented with Vaadin[3] (see Figure 4b), i.e., simspire.web. Both GUIs allow varying the patient parameters, starting or stopping the simulation, and seeing notable curves of the desired measures (typically, pressures or flows in the desired parts of the respiratory system). In particular, with both GUIs, it is possible to decide at which point of the circuit the flow and pressure have to be measured and, thus, vary the curves shown.

The communication between the GUI (whichever version is chosen) and the circuit simulator is made possible thanks to the lungsimulator component. It contains methods to set the component values in the circuit depending on those set by the user in the GUI or to show plots regarding the pressure and the flow in the desired part of the respiratory system.

Finally, SIMSPIRE has been designed to be as general as possible in terms of patient characteristics. Indeed, to allow users to represent different types of patients, under disparate health conditions, SIMSPIRE supports patients' models written using the DSL described in Section III-A. These models are read by the lungsimulator.yamlreaders component, which parses the files and correctly sets the circuit to represent the intended patient.

B. COMMUNICATION WITH EXTERNAL DEVICES

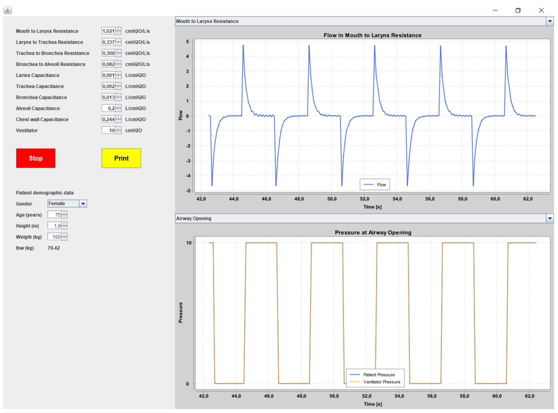
As previously introduced in Section IV-A, SIMSPIRE includes a ZeroMQ interface which is used for establishing the communication between the simulator and an external device, such as a mechanical ventilator. This communication channel is bidirectional since the external device can use it when values normally measured in the patient are needed, while SIMSPIRE must ask the counterpart which inspiratory pressure is set at every execution cycle. In particular, SIMSPIRE supports the following input messages, sent by the external device:

- `getFlow`: the external device asks for the current flow in the patient's airways;
- `getFlowPeak`: the external device asks for the maximum flow reached during the last respiratory cycle;
- `getPressure`: the external device asks for the current pressure (normally measured at mouth level) experienced by the patient;
- `getRespiratoryRate`: the external device asks for the current respiratory rate;
- `getFlatTopPressure`: the external device asks for the maximum pressure (normally measured at mouth level) experienced by the patient during the last respiratory cycle;
- `getPeep`: the external device asks for the PEEP (see Section II for further details).

When SIMSPIRE receives one of these messages, it replies with the requested measured value.

On the other hand, SIMSPIRE only needs to know which pressure is applied to the patient's mouth at each simulation cycle. For this purpose, it may use the `getPressure` message. When the external device receives this message, it shall respond with the current set pressure, which is used by SIMSPIRE to correctly set the generator in its electric circuit and simulate for the next simulation cycle the patient's status.

[1]https://github.com/pfalstad/circuitjs1
[2]https://zeromq.org
[3]https://vaadin.com



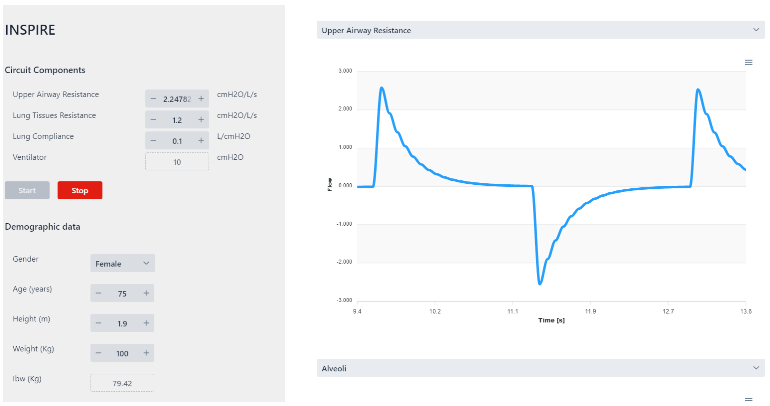
Mouth to Larynx Resistance	1,021	cmH2O/L/s
Larynx to Trachea Resistance	0,337	cmH2O/L/s
Trachea to Bronchea Resistance	0,306	cmH2O/L/s
Bronchea to Alveoli Resistance	0,082	cmH2O/L/s
Larinx Capacitance	0,001	L/cmH2O
Trachea Capacitance	0,002	L/cmH2O
Bronchea Capacitance	0,013	L/cmH2O
Alveoli Capacitance	0,2	L/cmH2O
Chest wall Capacitance	0,244	L/cmH2O
Ventilator	10	cmH2O

Patient demographic data

Gender	Female
Age (years)	75
Height (m)	1,9
Weigth (kg)	100
Ibw (kg)	79.42

Stop
Print
Mouth to Larynx Resistance
Flow in Mouth to Larynx Resistance
Flow
Time [s]
Airway Opening
Pressure at Airway Opening
Pressure
Patient Pressure
Ventilator Pressure

(a) Java Swing



INSPIRE

Circuit Components

Upper Airway Resistance	2.2478	cmH2O/L/s
Lung Tissues Resistance	1.2	cmH2O/L/s
Lung Compliance	0.1	L/cmH2O
Ventilator	10	cmH2O

Demographic data

Gender	Female
Age (years)	75
Height (m)	1.9
Weight (Kg)	100
Ibw (Kg)	79.42

(b) Vaadin

Figure 4: SIMSPIRE GUIs

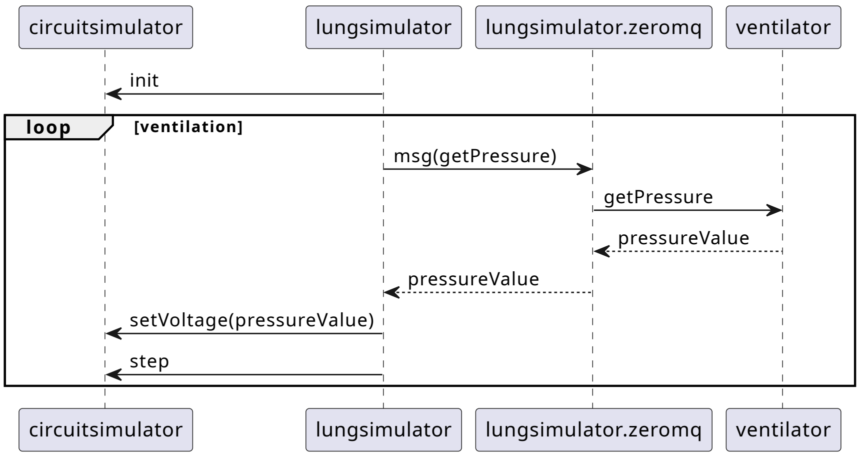


Figure 5: Communication flow between SIMSPIRE and a ventilator

An example of a simple communication flow between a ventilator and SIMSPIRE is reported in Figure 5. The communication is started by the lungsimulator component, which initializes the circuitsimulator. Then, the ventilation process can start. To simulate a patient under ventilation, the respiratory system simulator needs to know the pressure set by the external ventilator. For this reason, by exploiting the functionalities of the lungsimulator.zeromq component, it sends the `getPressure` message to the ventilator, that replies with the set pressure value. It is used by the lungsimulator to set the value of a voltage source in the circuitsimulator. Then, an execution `step` is performed and the loop is repeated until the ventilation is stopped.

We emphasize that the previously reported messages are those we needed during the development of SIMSPIRE and its testing with a prototype version of the control software of a mechanical ventilator, but further measures may be needed by different ventilators or other devices. For example, if SIMSPIRE is used to interact with a mechanical ventilator adopting volume-control ventilation, the `getVolume` message sent by the tool to the ventilator software should be added. In this case, SIMSPIRE can be extended to support additional communication flows and messages.

SIMSPIRE may be integrated more extensively with available commercial medical lung ventilators by using the ZeroMQ interface presented in Section IV-A and the commands presented in Section IV-B. This would require modification of ventilator control software: instead of reading sensors and activating actuators, it would have to send commands and read the data to/from SIMSPIRE on the proper ZeroMQ channels.

C. SCALABILITY AND GENERALIZABILITY THREATS

SIMSPIRE works as a simulator of the human respiratory system. Therefore, we designed it to be at least as responsive as the human body. In particular, the typical breathing frequency for humans is between 0.16 and 0.33 Hz [18], while ZeroMQ is designed to work with hundreds of thousands of messages per second[4] and the simulation loop is executed every 0.1 seconds, i.e., with a frequency of 10 Hz.

[4]https://github.com/mikehadlow/Mike.MQShootout

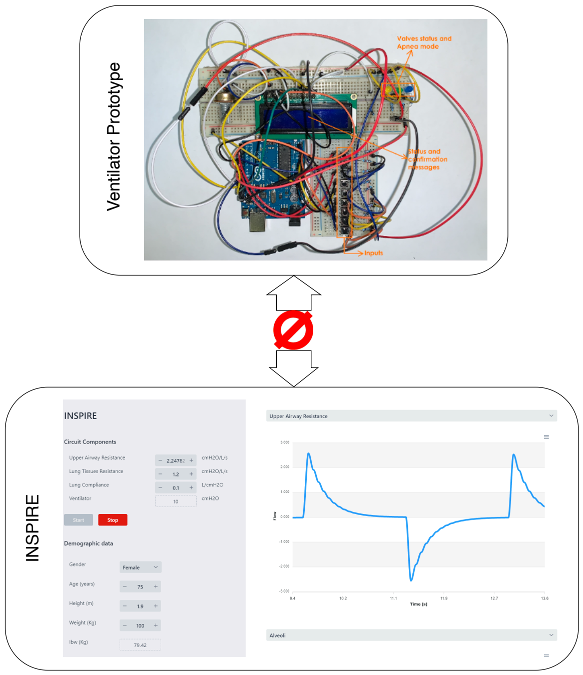


Figure 6: Setup for testing SIMSPIRE with a ventilator prototype

Regarding the spectrum of pathologies that SIMSPIRE could handle, as explained in Section III-A, our DSL allows for modeling disparate patients' conditions, as long as they can be represented by varying circuit parameters. Then, SIMSPIRE can simulate those circuits, without significant overhead and still working at human-compatible frequencies.

V. TESTING SIMSPIRE WITH A VENTILATOR PROTOTYPE

In this section, we describe how we have tested SIMSPIRE using the control software of a prototype of a mechanical ventilator. More specifically, by configuring the system as shown in Figure 6, we have been able to provide input (i.e., the inspiratory pressure) to the respiratory system. In particular, the ventilator software has been instrumented and extended to include a ZeroMQ interface. In this way, all the readings from sensors have been substituted with ZeroMQ requests, while valve actuation has been substituted with ZeroMQ messages sent from the ventilator software to the patient's simulator (as explained in Section IV-B). Then, as done in other works [19]–[21], we deployed the ventilator software on simple hardware, based on an Arduino Uno board. Then, we connected SIMSPIRE to the ventilator prototype using a serial port, which can be used by ZeroMQ for sending and receiving messages. In this way, with feedback from a doctor, we were able to assess whether SIMSPIRE 's performance accurately mirrored that of a patient undergoing mechanical ventilation.

VI. RELATED WORK

The use of digital twins, models, and simulators in the development of medical software has gained significant attention in recent years, especially for what concerns the simulation of human organs and physiological systems. In particular, digital twins have been used for personal and precision healthcare [22], [23], since they enable learning and discovering new knowledge through experiments ‘‘in silico". Indeed, digital twins of patients are created by transferring the patient's physical characteristics and changes in the body to the digital environment. This enables correctly diagnosing and following the treatment processes suitable for the patient, which is one of the most important principles of medicine [24]. Digital twins are frequently utilized in projects where they are employed alongside real medical systems. One instance of this is seen in [25], where a digital twin is implemented within an organ preservation device to enhance transplantation results through active monitoring and management of the organ's functioning during transportation. This allows for the prevention of biological degradation.

In the literature, simulators of several human organs, systems, and their conditions are available, e.g., the digital twin of human hearth [26], of the evolution of a cancer [27], of a stroke [28], or of the brain [6]. In this paper, the SIMSPIRE simulator for the respiratory system has been presented. Other simulators and digital twins of the human respiratory system exist [29], [30], but they have a high level of complexity which does not make them usable for software developers and testers interested in assessing the correct behavior of the software of medical devices, without having extensive knowledge in the medical domain. The SIMSPIRE respiratory system simulator presented in this paper has potential applications beyond simple testing. For instance, in [31], it has been utilized to simulate patient breathing and to support physician training on correctly setting mechanical ventilators.

VII. LIMITATIONS AND FUTURE WORK

In this paper, we have presented a set of models that can represent, with different levels of detail, the human respiratory system. The current models used in SIMSPIRE are limited in complexity since they employ only simple electronic components. As future work, we are planning to extend SIMSPIRE to support more complex models (e.g., including nonlinear components or behavior), to make the simulation even more realistic. While SIMSPIRE focuses on the respiratory system, it does not currently integrate with other physiological systems that can impact respiratory function, such as the cardiovascular system. To this end, we may explore more comprehensive solutions that enable us to incorporate not only parameters related to the respiratory system into the models but also other factors that may impact its proper functioning. For example, by extending SIMSPIRE in this way, we would be able to include in the simulation external systems, such as the cardiovascular one, which do not appartain to the respiratory system but can affect the human breath, though. Moreover, the tool needs to handle a wide range of patient conditions and diseases, which may require further enhancements to ensure it remains effective across diverse scenarios. Finally, in this paper, we have used only one mechanical ventilator control software to provide inputs to SIMSPIRE. However, the tool may need to be tested with a broader range of devices and under various conditions to fully validate its effectiveness and reliability. In future work, we may use our simulator to design a framework for testing mechanical ventilators. Besides testing mechanical ventilators, SIMSPIRE can be extended to train physicians in using ventilators in simulated scenarios by integrating it in an educational platform.

VIII. CONCLUSIONS

As safety-critical systems, medical devices need to be certified to be marketed and used in practice. During the certification process, testing covers a very important role, since testers need to ensure that no bugs are present that may lead to injuries to patients. However, the functioning of a device that interacts with the environment, including humans, directly depends on the environment itself. For this reason, during the last phases of testing, a medical device needs to be tested as a whole in the environment in which it is supposed to work.

In this paper, we have presented SIMSPIRE, a flexible and fully configurable simulator for the human respiratory system. Thanks to its flexibility, SIMSPIRE can be used to simulate patients with different health conditions. For example, it can be a support during the development or testing of a mechanical ventilator. It allows testers and developers to execute the mechanical ventilator software against different types of patients, modeled as electrical circuits, and simulate the variation of patients' conditions or the onset of diseases influencing how the patients breathe.

SIMSPIRE has been tested by using inputs provided by the control software of a mechanical ventilator, to check whether the simulator's response was corresponding to that expected by doctors when ventilating patients, and it has shown to be effective for testing the device.

ACKNOWLEDGMENT

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.1, Call for tender No. 104 published on 2.2.2022 by the Italian Ministry of University and Research (MUR), funded by the European Union – NextGenerationEU – SAFEST: Trust assurance of Digital Twins for medical cyber-physical systems – CUP F53D23004230006 - Grant Assignment Decree n. 861 on 16-6-2023 by the Italian Ministry of Ministry of University and Research (MUR). This work has been partially funded by PNRR - ANTHEM (AdvaNced Technologies for Human-centrEd Medicine) - Grant PNC0000003 – CUP: B53C22006700001 - Spoke 1 - Pilot 1.4.

References

[1] A. Abba, C. Accorsi *et al.*, "The novel mechanical ventilator milano for the covid-19 pandemic," *Physics of Fluids*, vol. 33, no. 3, p. 037122, 03 2021. [Online]. Available: https://doi.org/10.1063/5.0044445

[2] A. Bombarda, S. Bonfanti, C. Galbiati, A. Gargantini, P. Pelliccione, E. Riccobene, and M. Wada, "Lessons Learned from the Development of a Mechanical Ventilator for COVID-19," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 24–35.

[3] A. Dexter *et al.*, "Validating Lung Models Using the ASL 5000 Breathing Simulator," *Simulation in Healthcare: The Journal of the Society for Simulation in Healthcare*, vol. 13, no. 2, p. 117–123, Apr. 2018. [Online]. Available: http://dx.doi.org/10.1097/SIH.0000000000000277

[4] International Electrotechnical Commission, *IEC 62304:2006 Medical device software — Software life cycle processes*, International Electrotechnical Commission Std., 2015.

[5] A. Bombarda *et al.*, "Guidelines for the development of a critical software under emergency," *Information and Software Technology*, vol. 152, p. 107061, 2022.

[6] W. Lu *et al.*, "Digital Twin Brain: a simulation and assimilation platform for whole human brain," *ArXiv*, vol. abs/2308.01241, 2023.

[7] O. Bjelland *et al.*, "Toward a Digital Twin for Arthroscopic Knee Surgery: A Systematic Review," *IEEE Access*, vol. 10, p. 45029–45052, 2022. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2022.3170108

[8] R. Laubenbacher, J. P. Sluka, and J. A. Glazier, "Using Digital Twins in Viral Infection," *Science*, vol. 371, no. 6534, p. 1105–1106, Mar. 2021. [Online]. Available: http://dx.doi.org/10.1126/science.abf3370

[9] B. Baillargeon *et al.*, "Human Cardiac Function Simulator for the Optimal Design of a Novel Annuloplasty Ring with a Sub-valvular Element for Correction of Ischemic Mitral Regurgitation," *Cardiovascular Engineering and Technology*, vol. 6, no. 2, p. 105–116, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1007/s13239-015-0216-z

[10] A. van Diepen *et al.*, "A model-based approach to generating annotated pressure support waveforms," *Journal of Clinical Monitoring and Computing*, Feb. 2021. [Online]. Available: https://doi.org/10.1007/s10877-022-00822-4

[11] D. Campbell and J. Brown, "The Electrical Analogue Of Lung," *British Journal of Anaesthesia*, vol. 35, no. 11, pp. 684–692, Nov. 1963. [Online]. Available: https://doi.org/10.1093/bja/35.11.684

[12] V. K. Jain and S. K. Guha, "A study of intermittent positive pressure ventilation," *Medical and Biological Engineering*, vol. 8, no. 6, pp. 575–583, Nov. 1970. [Online]. Available: https://doi.org/10.1007/bf02478231

[13] A. Baker and C. Hahn, "An analogue study of controlled ventilation," *Respiration Physiology*, vol. 22, no. 3, pp. 227–239, Dec. 1974. [Online]. Available: https://doi.org/10.1016/0034-5687(74)90074-7

[14] H. S. Ghafarian P, Jamaati H, "A review on human respiratory modeling," *Tanaffos*, no. 2, pp. 61–69, 2016.

[15] I. Muntean, C. Ionescu, and I. Nascu, "Models for the respiratory system using morphology-based electrical analogy," in *2008 First International Conference on Complexity and Intelligence of the Artificial and Natural Complex Systems. Medical Applications of the Complex Systems. Biomedical Computing*, 2008, pp. 243–249.

[16] A. Albanese *et al.*, "An Integrated Mathematical Model of the Human Cardiopulmonary System: Model Development," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 310, no. 7, pp. H899–H921, Apr. 2016. [Online]. Available: https://doi.org/10.1152/ajpheart.00230.2014

[17] N. Q. Al-Naggar, "Modelling and Simulation of Pressure Controlled Mechanical Ventilation System," *Journal of Biomedical Science and Engineering*, vol. 08, no. 10, pp. 707–716, 2015. [Online]. Available: https://doi.org/10.4236/jbise.2015.810068

[18] N. J. Silvestri and C. H. Gibbons, "5 - Autonomic Dysfunction in Neuromuscular Disorders," in *Neuromuscular Disorders: Treatment and Management*, T. E. Bertorini, Ed. Saint Louis: W.B. Saunders, 2011, pp. 61–77. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781437703726000050

[19] E. Malaekah *et al.*, "Designing Hybrid Mechanical Ventilator System Based on Arduino and Raspberry Pi 4," *Journal of Medical Devices*, vol. 16, no. 1, Mar. 2022. [Online]. Available: http://dx.doi.org/10.1115/1.4054036

[20] A. Bombarda *et al.*, "Developing a Prototype of a Mechanical Ventilator Controller from Requirements to Code with ASMETA," *Electronic Proceedings in Theoretical Computer Science*, vol. 349, pp. 13–29, nov 2021. [Online]. Available: https://doi.org/10.4204%2Feptcs.349.2

[21] M. S. G. Tsuzuki *et al.*, "Mechanical Ventilator VENT19," *Polytechnica*, vol. 4, no. 1, p. 33–46, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1007/s41050-021-00031-z

[22] M. N. K. Boulos and P. Zhang, "Digital Twins: From Personalised Medicine to Precision Public Health," *Journal of Personalized Medicine*, vol. 11, no. 8, p. 745, Jul. 2021. [Online]. Available: https://doi.org/10.3390/jpm11080745

[23] T. Sun, X. He, and Z. Li, "Digital twin in healthcare: Recent updates and challenges," *DIGITAL HEALTH*, vol. 9, p. 205520762211496, Jan. 2023. [Online]. Available: https://doi.org/10.1177/20552076221149651

[24] T. Erol, A. F. Mendi, and D. Doğan, "The Digital Twin Revolution in Healthcare," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020, pp. 1–7.

[25] A. J. Buhagiar, L. Freitas, W. E. S. III, and P. G. Larsen, "Digital Twins for Organ Preservation Devices," in *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 22–36. [Online]. Available: https://doi.org/10.1007/978-3-031-19762-8_3

[26] H. Elayan, M. Aloqaily, and M. Guizani, "Digital Twin for Intelligent Context-Aware IoT Healthcare Systems," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 749–16 757, 2021.

[27] J. Zhang *et al.*, "Cyber Resilience in Healthcare Digital Twin on Lung Cancer," *IEEE Access*, vol. 8, pp. 201 900–201 913, 2020.

[28] I. Hussain, M. A. Hossain, and S.-J. Park, "A Healthcare Digital Twin for Diagnosis of Stroke," in *2021 IEEE International Conference on Biomedical Engineering, Computer and Information Technology for Health (BECITHCON)*, 2021, pp. 18–21.

[29] Y. Feng, X. Chen, and J. Zhao, "Create the individualized digital twin for noninvasive precise pulmonary healthcare," *Significances Bioengineering & Biosciences*, vol. 1, no. 2, pp. 10–31 031, 2018.

[30] C. Patte, M. Genet, and D. Chapelle, "A quasi-static poromechanical model of the lungs," *Biomechanics and Modeling in Mechanobiology*, vol. 21, no. 2, pp. 527–551, Jan. 2022. [Online]. Available: https://doi.org/10.1007/s10237-021-01547-0

[31] A. Bombarda *et al.*, "An Android App for Training New Doctors in Mechanical Ventilation," in *Proceedings of the 17th International Joint Conference on Biomedical Engineering Systems and Technologies*. SCITEPRESS - Science and Technology Publications, 2024, p. 362–369. [Online]. Available: http://dx.doi.org/10.5220/0012323900003657

ANDREA BOMBARDA is a Research Associate at the Department of Management, Information, and Production Engineering at the University of Bergamo. He received his M.E. and Ph.D. degrees from the University of Bergamo in 2018 and 2022, respectively. His research interests are in the areas of model-based testing and combinatorial testing applied to safety-critical systems, mainly in the medical field. He serves as a member of the program committee for several conferences and reviewer for journals in software engineering and software testing. More information is available at https://cs.unibg.it/bombarda/.

SILVIA BONFANTI is a Research Associate at the University of Bergamo (Italy) and she is part of FOSELab (Formal Methods and Software Engineering Laboratory). She received her PhD in Engineering and Applied Science from the University of Bergamo, in collaboration with SCCH (Software Competence Center Hagenberg). Her research interests include software engineering, software testing, formal methods, and medical software certification. She is involved in the organization of international conferences. She serves as a member of the program committee for several conferences and reviewer for journals in software engineering and software testing. More information is available at https://cs.unibg.it/bonfanti/.

ANGELO GARGANTINI is a full professor in Computer Science and Engineering at the University of Bergamo (Italy), and he is the director of the FOSELab (Formal Methods and Software Engineering Laboratory). He received his PhD in computer engineering from the Politecnico of Milan. Before joining the University of Bergamo, he has worked for the Politecnico of Milan, the Naval Research Laboratory in Washington DC, and the University of Catania. His research focuses on automated testing techniques, model-based testing, mutation testing, and the application of formal methods in software validation and verification. More information is available at https://cs.unibg.it/gargantini/.

ELVINIA RICCOBENE is full professor in Computer Science at the Computer Science Department of the University of Milan (Italy), and she is the director of the FALSE Lab (Formal Methods and Software Engineering Laboratory). She received laurea and PhD degree in Mathematics. Her research interests include formal methods, with particular expertise in the Abstract State Machines, integration between formal modeling and model-driven engineering, model analyses techniques for software systems. More information is available at https://homes.di.unimi.it/riccobene/.

• • •