Formal Verification of a Mechanical Ventilator using UPPAAL

Ortiz Vega, James Jerson; ARANDA BUENO, Jesus Alexander; CUARTAS GRANADA, Jaime; Cortés, David; BETANCOURT ARIAS, Joan Sebastian; Garcia, José Isidro; Valencia, Andrés

Published in:
9th ACM International Workshop on Formal Techniques for Safety-Critical Systems

Publication date:
2023

Document Version
Peer reviewed version

Citation for pulished version (HARVARD):
Ortiz Vega, JJ, ARANDA BUENO, JA, CUARTAS GRANADA, J, Cortés, D, BETANCOURT ARIAS, JS, Garcia, JI & Valencia, A 2023, Formal Verification of a Mechanical Ventilator using UPPAAL. in *9th ACM International Workshop on Formal Techniques for Safety-Critical Systems.* International Workshop on Formal Techniques for Safety-Critical Systems, 22/10/23.

Formal Verification of a Mechanical Ventilator using UPPAAL

Jaime Cuartas
jaime.cuartas@correounivalle.edu.co
Universidad del Valle
Colombia

David Cortés
david.cortes@correounivalle.edu.co
Universidad del Valle
Colombia

Joan S. Betancourt
joan.betancourt@correounivalle.edu.co
Universidad del Valle
Colombia

Jesús Aranda
jesus.aranda@correounivalle.edu.co
Universidad del Valle
Colombia

José I. García
jose.i.garcia@correounivalle.edu.co
Universidad del Valle
Colombia

Andrés M. Valencia
andres.v.restrepo@correounivalle.edu.co
Universidad del Valle
Colombia

James Ortiz
james.ortizvega@unamur.be
University of Namur
Belgium

Abstract

Mechanical ventilators are increasingly used for life support of critically ill patients. In this sense, despite recent technological advances, the accurate specification of their properties remains challenging, and the use of formal tools is limited. This work focuses on verifying the properties of the architecture of a mechanical ventilator using UPPAAL as a modeling tool. As a result, the system requirements and specification of a functional prototype were verified and improved using the formal model of a mechanical ventilator. This approach provides a valuable means of ensuring the correctness and reliability of mechanical ventilator systems.

CCS Concepts: • **Computer systems organization** → *Real-time system specification*; Embedded software; • **Computing methodologies** → **Model verification and validation**.

Keywords: Timed Automata, Formal Verification, Mechanical Ventilator

ACM Reference Format:
Jaime Cuartas, David Cortés, Joan S. Betancourt, Jesús Aranda, José I. García, Andrés M. Valencia, and James Ortiz. 2023. Formal Verification of a Mechanical Ventilator using UPPAAL. In *Proceedings of the 9th ACM SIGPLAN International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS '23), October 22, 2023, Cascais, Portugal.* ACM, Cascais, Portugal, 12 pages. https://doi.org/10.1145/3623503.3623536

1 Introduction

Mechanical ventilation is a critical approach to life support in patients who lack the ability to generate a level of ventilation that ensures proper gas exchange, oxygenation, and carbon dioxide elimination in the lungs. Using technological devices called mechanical ventilators, a positive pressure is created to open the airways and expand the lungs [23]. Although mechanical ventilation has proven its effectiveness in critical care scenarios, it has also been responsible for serious adverse effects on certain patients [28, 33], while internal time desynchronizations and settings configuration of mechanical ventilation systems have been linked to possible worse health outcomes [19, 32]. The importance of the respiratory system for the health of patients and the issues related to ventilation systems justify the exploration of formal techniques to ensure the correct specification of mechanical ventilators.

There has been a constant technological evolution of medical systems such as the mechanical ventilator, but the literature analysis revealed limited information related to the integration of formal methods to it. This paper explores the capabilities of Timed Automata (TA) [1] modeling for verifying the properties of these time-critical systems using UPPAAL, a tool for modeling, validation and verification of real-time systems [8], which we believe is a novel and interesting case study of the application of formal methods to the medical industry.

Related Work. Previous research has explored the verification of time- and safety-critical embedded systems [14, 15]. However, these methods have only recently been applied to evaluating and verifying rigorously medical devices. Formalized methods have been developed to improve medical device

FTSCS '23, October 22, 2023, Cascais, Portugal

ACM ISBN 979-8-4007-0398-0/23/10…$15.00
https://doi.org/10.1145/3623503.3623536

protocols [20] and security [35]. In particular, the authors in [3] used extended finite state machines to model check a resuscitation device. Implantable pacemakers, for example, have been extensively studied in the formal verification community; see an attempt at modeling with a synchronous modeling language in SCADE [24], a TA approach with UPPAAL [31], or the PACEMAKER Formal Methods Challenge [30]. In [38], Colored Petri Nets are used to specify an online learning architecture for mechanical ventilation to verify the integration, the coordination of the different resources, and the flow of information generated.

For this present work, other formalisms were also considered. Hybrid Automata (HA) were not used because the verification of relevant properties of the system relies on asking whether the system may reach certain states (i.e. "will the ventilator reach the inspiration state?") and this is generally not decidable for HA [22]. A preliminary model was also sketched in SCADE[1], and we are considering using recent advances in real-time Maude to translate our parametric timed automata and explore the application of rewriting logic for this use case [2].

Document structure. The structural characteristics of the mechanical ventilator considered for the development of this work are described below in section 2. In addition, the analytical expressions of the components of a simplified model of the mechanical ventilator that relate to the variables of flow and pressure are described. In section 3, the formal environment that supports the formal model of the mechanical ventilator in UPPAAL is presented. In section 4, the dynamics of the critical system are formally represented, focusing on the analysis of the properties related mainly to the configuration, the control valves, and the control strategy, which are subsequently verified in section 5. Finally, the results and relevant comments are presented in section 6.

2 Mechanical Ventilator Architecture

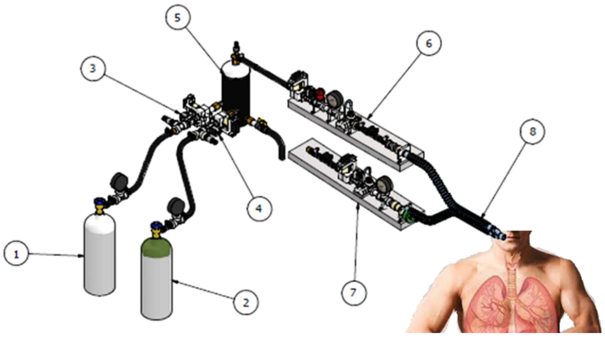

Figure 1. Mechanical Ventilator Architecture.

In Figure 1, we outline the architecture of a mechanical ventilator designed by research groups at the Universidad del Valle [38]. A functional prototype of this ventilator is being used to train medical students [36] and is the subject of this work.

The mechanical ventilator architecture, see Figure 1, integrates two gas supplies, medical air (1) and oxygen (2), connected to their respective supply lines, which have a pressure sensor and a set of proportional and on-off valves (3) and (4). The operation of these valves makes it possible to control the volume of the gas mixture in a tank (5) according to a fraction of inspired oxygen FiO2 parameter (expressed as the percentage of oxygen-enriched air) set by a medical professional. The oxygen-enriched fluid is then delivered to the patient's lungs through an inspiration tube (6), which includes a set of oxygen, pressure, flow sensors, and a proportional valve. These devices are managed in various modes, spontaneous and assisted. In assist mode, the breaths delivered by the ventilator control either volume or pressure. In volume-controlled ventilation, the target is a specific lung volume during inspiration configured by the physician. These operating modes define the following functions of the ventilator: respiratory rate, regulation of the flow of oxygen-enriched air delivered to the patient, monitoring of the percentage of oxygen added to the mixture, and monitoring of the pressure in the supply line. The carbon dioxide-enriched fluid is then expelled from the patient's lungs into the atmosphere through the expiration line (7), which incorporates pressure and flow sensors and a proportional valve. These devices are also managed according to the various operating parameters configured by the clinician, such as expiratory frequency and end-expiratory pressure. The expiratory line allows the following mechanical ventilator functions to be defined: carbon dioxide enriched fluid regulation, expiratory line monitoring, and expiratory line low-pressure alarm. A disposable line (8), called the patient connection line, is used for gas exchange with a patient. The system under test in this study, described in this section, can be seen in Figure 2.

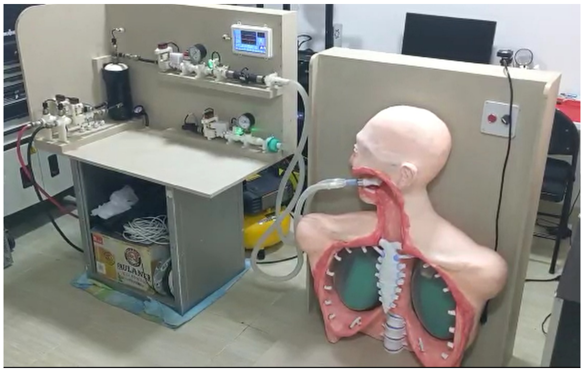

Figure 2. Physical plant of the mechanical ventilator.

[1] SCADE model available here https://github.com/ventynet/ventynet-SCADE

2.1 Simplified Model

According to [18], in a fluid system, such as a mechanical ventilator, we focus on two key variables: pressure (P) and the rate at which air flows (Q). The relationship between these variables is known as hydraulic power. Furthermore, the system consists of three basic components: a capacitor (or accumulator), an injector, and a resistor. The capacitor and injector store energy based on pressure or flow rate, while the resistor dissipates energy.

Assuming a negligible contribution of the inertia of the fluid due to the slow flow in a mechanical ventilator, a simplified model from the supply of gas tanks to the lung is obtained. The model integrates elements of system architecture that ensure a supply of oxygen-enriched air as the power source (1 to 5), two dissipating elements, a pipeline, control valve, representing the hydraulic lines (6 to 8), and a compressible gas accumulator serving as a lung, see Figure 3.

To model the pipeline, it is considered a fluid flow through a narrow pipeline, and the energy dissipation is due to fluid friction. Thus, assuming a laminar flow and a viscous fluid, u, the linear relationship for fluid resistance is described by Equation 1.

$$P = R_f Q \tag{1}$$

Here P is the pressure drop from the power supply to the accumulator ($P_2 - P_1$), Q is the supply flow in the pipeline, related to the supply volume V_1, and R_f is the fluid resistance.

Considering a circular pipeline of diameter, D, and a length, L, the fluid resistance is given by Equation 2.

$$R_f = \frac{u\ L}{\pi D^4} \tag{2}$$

The expression used in this work, called Darcy-Weisbach [25], is described by Equation 3.

$$Q = \frac{(P_2 - P_1) * \pi * D^4}{l * 128 * \mu} \tag{3}$$

In addition, two discrete states were considered to model the control valve: close and open.

A rigid container serves as an accumulator, with a single inlet through which fluid is pumped in at the volume flow rate Q and the pressure inside the container with respect to the outside is P, the linear constitutive equation is Equation 4:

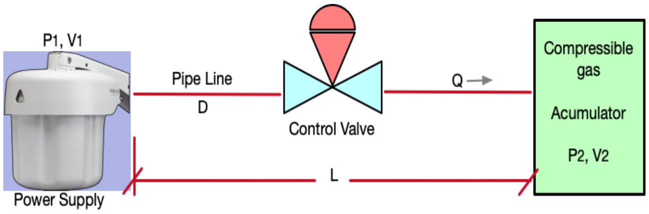

Figure 3. Diagram of simplified ventilator model.

$$C_f \frac{dP}{dt} = Q \tag{4}$$

Where C_f is called fluid capacitance. Assuming oxygen-enriched air as an ideal gas, the gas law is used (Equation 5).

$$PV = nRT \tag{5}$$

Here P is the pressure, V is the volume inside the container, V_2, T is the absolute temperature, n is the number of moles of gas in the volume defined as the mass of gas (kg)/ molecular mass of the gas (kg), and R, the universal gas constant, defined as $8.3145\,\mathrm{J\,K^{-1}\,mol^{-1}}$.

Assuming a slow flow of the gas, a negligible deformation of the container, and a heat transfer that allows a constant temperature of the gas, the following expression is derived from gas law (Equation 6).

$$P\frac{dV}{dt} + V\frac{dP}{dt} = 0 \tag{6}$$

Defining the flow that enters the interior of the tank, Q, as a function of the variation with respect to time of the volume accumulated of air enriched with oxygen as $\frac{-dV}{dt}$ is obtained the following equation (Equation 7).

$$\frac{V}{P}\frac{dP}{dt} = Q = \frac{mR_{sp}}{P^2}\frac{dP}{dt} \tag{7}$$

Here $R_{sp} = \frac{R}{(\text{Molecular mass of the gas})}$, and $C_f = \frac{(mR_{sp})}{P^2}$

Assuming a strategy of control in the mechanical ventilator called constant volume, V, a computational solution for the pressure variation at small-time intervals, t=0.04 s to 0.08 s was estimated using Euler's method [12], (Equation 8).

$$\Delta P = \Delta t * \frac{R * T * \rho * Q}{V} \tag{8}$$

In the programming of mechatronic systems, the waiting time between process cycles called *delay* plays a critical role in ensuring synchrony between the response speed of the actuators and the response speed of the processor. Actuators are responsible for performing physical actions based on signals from the processor, which in turn processes data and makes decisions [13]. If the delay time is excessive, desynchronization between physical actions and processor decisions can occur, resulting in system malfunction and loss of accuracy. On the other hand, if the delay time is too short, the processor may be overloaded. In this sense, finding an optimal balance in the delay time is essential to ensure proper synchrony, allowing the actuators to respond quickly and accurately to the processor's instructions, which guarantees the reliable and efficient operation of the mechatronic system as a whole. Therefore, efficient programming must consider the delay time and adjust it appropriately to achieve perfect synchronization and avoid performance problems.

The process to be developed integrates the application of the physical model describing the system behavior with the computational model developed in UPPAAL to determine the optimal time interval, Δt, associated with the previously mentioned delay time. The main objective of this process is to achieve an adequate synchronization between the response speed of the actuators and the processing capacity of the mechatronic system. Once the optimal value of Δt is obtained, it will be implemented in the development equipment available to carry out tests to validate and verify the effectiveness of this methodology. These experimental tests will be fundamental to verify the optimization of the mechatronic system performance and will provide empirical support to the usefulness and practical relevance of the proposed strategy.

3 Clocks, Clock Constraints and Automata

To model the continuous time domain, we use non-negative real-valued variables, which are known as *clocks*. *Clocks* in Timed Automata (TA) increase synchronously at the same rate. TA are extensively studied formalisms for modeling critical timed systems (CTS) [1]. Several model checkers, such as UPPAAL [17], KRONOS [11], and HYTECH [21], rely on TA for verification purposes. TA are an extension of Finite State Automata (FSA) with the addition of a set of clocks that increase concurrently. Resetting a clock in TA entails updating its value to zero. Transitions in TA can be enabled or disabled based on clock constraints, and transitions are taken if all other conditions are satisfied.

3.1 Timed Automata

The following definitions outline the guidelines for defining constraints and invariants over clocks.

Definition 3.1 (Clock constraints)**.** Let X be a finite set of clock variables ranging over $\mathbb{R}_{\geq 0}$ (non-negative real numbers). Let $\Phi(X)$ be a set of clock constraints over X. A *clock constraint* $\phi \in \Phi(X)$ can be defined by the following grammar:

$$\phi ::= true \mid x \sim c \mid \phi_1 \ \wedge \ \phi_2$$

where $x \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, >, \leq, \geq, =\}$.

Definition 3.2 (Clock Invariants)**.** Let X be a finite set of clock variables ranging over $\mathbb{R}_{\geq 0}$. Let $\Delta(X)$ be a set of clock invariants over X. Clocks invariants are clock constraints of the following form:

$$\delta ::= true \mid x < c \mid x \leq c \mid \phi_1 \ \wedge \ \phi_2$$

where $x \in X$, $c \in \mathbb{N}$.

Clock constraints in the form of *true* or $x \sim c$ are referred to as *non-diagonal constraints*, while those in the form of $x - y \sim c$ are termed *diagonal constraints*. The set of non-diagonal constraints defined over the set of clocks X is denoted as $\Phi(X)$. In this context, we employ *non-diagonal constraints* as described in [5], where the comparison between two clocks is not permitted, as indicated in [10]. It's worth noting that *diagonal constraints* do not contribute any additional expressive power to TA. Consequently, *diagonal constraints* can be eliminated from TA, as discussed in [10]. However, this removal results in an exponential increase in the number of states, which is generally an unavoidable consequence, as established in [9].

Definition 3.3 (Clock valuations)**.** Given a finite set of clocks X, a clock valuation function $v : X \longrightarrow \mathbb{R}_{\geq 0}$ assigns to each clock $x \in X$ a non-negative value $v(x)$. We denote $\mathbb{R}^X_{\geq 0}$ the set of all valuations. For a clock valuation $v \in \mathbb{R}^X_{\geq 0}$ and a time value $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the valuation satisfying $(v + d)(x) = v(x) + d$ for each $x \in X$. Given a clock subset $Y \subseteq X$, we denote $v[Y \longleftarrow 0]$ the valuation defined as follows: $v[Y \longleftarrow 0](x) = 0$ if $x \in Y$ and $v[Y \longleftarrow 0](x) = v(x)$ otherwise.

Definition 3.4. Timed Automaton (TA) [5] A timed automaton is a tuple $(L, l_0, X, \Sigma, E, I)$, where:

- L is a finite set of locations,
- $l_0 \subseteq L$ is an initial location,
- X is a finite set of clocks,
- Σ is a finite set of actions,
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$ is a finite set of edges between locations.
- $I : L \longrightarrow \Delta(X)$ assigns invariants to locations.

For a transition $(l, a, \phi, Y, l') \in E$, we classically write $l \xrightarrow{a,\phi,Y} l'$ and call l and l' the source and target locations, ϕ is the guard, a the action (or alphabet), Y the set of clocks to reset. The semantics of a TA is a Timed Transition System (TTS) where a *state* is a pair $(l, v) \in L \times \mathbb{R}^X_{\geq 0}$, where l denotes the current location with its accompanying clock valuation v, starting at (l_0, v_0) where v_0 maps each clock to 0. The transitions can be of two types: Delay transitions only let time pass without changing location. Discrete transitions occur instead between a source and a target location. The transition can only occur if the current clock values satisfy both the guard of the transition and the invariant of the target location.

Definition 3.5 (Semantics of TA)**.** Let $\mathcal{A} = (L, l_0, X, \Sigma, E, I)$ be a TA. The semantics of TA $\mathcal{A}$ is given by a TTS$(\mathcal{A})$ = $(S, s_0, \Sigma_\Delta, \longrightarrow)$ where:

- $S \subseteq L \times \mathbb{R}^X_{\geq 0}$ is a set of states,
- $s_0 = (l_0, v_0)$ with $v_0(x) = 0$ for all $x \in X$ and $v_0 \models I(l_0)$,
- $\Sigma_\Delta = \Sigma \uplus \mathbb{R}_{\geq 0}$,
- $\longrightarrow \subseteq S \times \Sigma_\Delta \times S$ is a transition relation defined by the following two rules:
  - **Discrete transition:** $(l, v) \xrightarrow{a} (l', v')$, for $a \in \Sigma$ iff $l \xrightarrow{a,\phi,Y} l'$, $v \models \phi$, $v' = v[Y \longleftarrow 0]$ and $v' \models I(l')$ and,
  - **Delay transition:** $(l, v) \xrightarrow{d} (l, v + d)$, for some $d \in \mathbb{R}_{\geq 0}$ iff $v + d \models I(l)$.

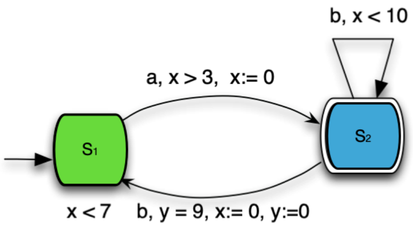

Figure 4. A TA with two clocks x and y.

The interconnection between two TA can be obtained by using synchronization channels. One automaton emits the signal in one transition and is received by one or more automata. With this, a Network of TA (NTA) may be formed.

Example 3.6. Let $\mathcal{A}$ be the TA depicted in Figure 4. $\mathcal{A}$ contains two locations: l_0 (initial) and l_1. In particular, l_0 is the only location to define an invariant not trivially true: $I(l_0) = (x < 7)$, forcing the TA to exit l_0 before x becomes 7. Location l_1 has a true invariant (thus not drawn), allowing it to stay in l_1 forever. Suppose the current location is l_1. The transition $l_1 \xrightarrow{b,(y=9),\{x:=0;y:=0\}} l_0$ specifies that when the action b occurs and the guard $y = 9$ holds, this enables the transition, leading to a new current location l_0, while resetting clock variables x and y. Note that using a location invariant (which specifies the time limit to stay in a given location) differs from using a guard (specifying when the transition is enabled).

3.2 UPPAAL Model Checker

The UPPAAL model-checker is built upon the theory of TA [1]. Its modeling language offers additional features, including bounded integer variables. UPPAAL is a comprehensive tool environment designed for the modeling, validation, and verification of real-time systems represented as networks of TA extended with data types [7]. In UPPAAL, a system is typically modeled as a network consisting of multiple TA running in parallel [7]. This allows for the representation of complex systems with diverse behaviors and interactions. The UPPAAL tool provides significant advantages, especially when dealing with systems that exhibit large and complex dynamics and stochastic behaviors. UPPAAL Statistical Model Checking (SMC) is an extension of UPPAAL that provides an alternative approach to address these challenges [16]. The core idea behind UPPAAL SMC is to monitor system simulations and utilize statistical results to assess the satisfaction of properties with a certain level of reliability. The stochastic interpretation in SMC replaces non-deterministic choices between multiple enabled transitions with probabilistic choices, where the choice is randomized. One of the main benefits of UPPAAL SMC is its ability to avoid exhaustive exploration of the state space of the model [16]. It represents systems as networks of TA, allowing for modeling behaviors that depend on stochastic and non-linear dynamical features. Each component of the system is described as an automaton with clocks that evolve at the same rate [7].

The UPPAAL SMC has been successfully applied in a wide range of case studies [6, 27, 29, 39]. It provides users with additional query capabilities related to the stochastic interpretation of TA. Furthermore, users can visualize expression values during simulated runs, gaining insights into the behavior of the system and enabling the exploration of more complex properties using the model-checker. Simulation queries, for example, allow the estimation of quality properties for feasible configurations by running one or more simulations on the configuration model with the corresponding quality attributes [16].

3.3 Query Language

In model checking, it is essential to verify that a model meets the required specifications. To do this, the properties must be expressed in a formal language, such as a simplified version of TCTL (Timed Computational Tree Logic) that is supported by UPPAAL. Various supported path formulae are shown in Table 1.

Where:

$$\rho ::= true \mid ap \mid \rho_1 \wedge \rho_2 \mid \rho_1 \vee \rho_2 \mid \neg\rho$$

is an expression used in symbolic queries, where is ap an atomic proposition [5] and:

$$\varphi ::= ap \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \cup^{x}_{\leq d} \varphi_2$$

is a weighted extension of the temporal logic MITL, ap is a conjunction of predicates over the state of a NTA, $d \in \mathbb{N}$ and x is a clock. [16].

4 The Model in UPPAAL

This section provides a detailed description of the mechanical ventilator model developed in UPPAAL. The model is designed to represent the core functionality and behavior of the mechanical ventilator in [36]. By using the formal modeling capabilities of UPPAAL, we can capture the dynamics and interactions within the ventilator system, enabling rigorous analysis and verification of its temporal and safety aspects. The model is constructed using four TA that together form a NTA. The four components of the network are as follows:

- **Setup**: This component is responsible for setting the initial parameters of the ventilator, including the FiO2 (Fraction of Inspired Oxygen), flow rate, and duty cycle.
- **Control**: The control component synchronizes with the other TA to implement the classic mechanism of a mechanical ventilator. It oversees the configuration, mixing, inspiration, pause, expiration, and pause phases of the ventilation process.

Path formula	UPPAAL	Description
$\exists\Diamond\rho$	`E<>`ρ	There exists a path where ρ eventually holds.
$\exists\Box\rho$	`E[]`ρ	There exists a path where ρ is always satisfied.
$\forall\Box\rho$	`A[]`ρ	For all paths, ρ is always satisfied.
$\forall\Diamond\rho$	`A<>`ρ	For all paths, ρ is eventually satisfied.
$\rho \leadsto \xi$	$\rho \rightarrow \xi$	ρ implies ξ eventually, always.
UPPAAL-SMC		Description
`simulate[<=t;n]`φ		Visualize the value of φ over n runs and time $\leq t$.
`Pr[<=t]`(φ)		Estimates the probability of φ where time $\leq t$.
$\Pr[\leq t](\varphi) \geq p_0$		Probability of φ being $\geq p_0$ during time $\leq t$.
$Pr[\leq t_1](\varphi_1) \geq Pr[\leq t_2](\varphi_2)$		The probability of φ_1 being $\geq$ than the probability of φ_2, with t_1, t_2 as time bounds for each event.

`E[bound; N](min:expr)`
or
`E[bound; N](max:expr)`
		Shows the minimum (resp. maximum) value that `expr` will hold within bound duration of `N` runs.

Table 1. Query language in UPPAAL [7, 17, 37]

- **Injector**: The injector component models a supply line that delivers either oxygen or air into a mixing chamber. The fluids are mixed until the desired FiO2 level is achieved. The resulting mixture is then delivered to the patient.
- **ExpValve**: The expiration valve component represents the valve responsible for safely extracting air from the patient's lungs after the inspiration phase.

The inputs to the model include the desired flow rate, inspiration and expiration times in centiseconds (cs), duration of the inspiration/expiration pauses (cs), and the desired FiO2 level. The following subsections will describe these components in more detail.

Setup

Figure 5 shows the Setup model, which represents the initial configuration phase of the mechanical ventilator. The model begins at the Idle location, waiting to be awakened by the initSetup? action. Next, a succession of four committed locations and five transitions set up the system, the locations need to be committed to ensure there's no interleaving at this point. First, the FiO2 parameter is selected. Next, the appropriate levels of oxygen (O2) and air, as well as the required duty for the oxygen and air valves, are determined based on a set of rules provided by the team that built the physical plant. Finally, the pressure in the mixing chamber is set on the next transition using Equation 8 and the automaton signals the rest of the system that the configuration phase has finished via the `endSetup!` action. The model then re-enters Idle, completing the cycle.

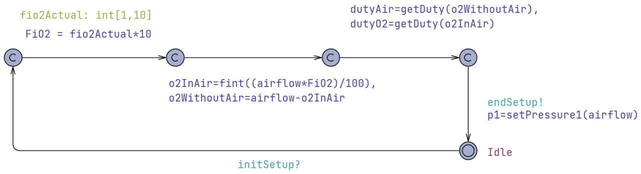

Figure 5. Setup model

Control

The model depicted in Figure 6 represents the mechanics of the inspiration and expiration cycles in a classical mechanical ventilator. The initial location, Idle, is at the top, marked as urgent to prevent waiting before sending the initSetup! signal. After receiving endSetup? we introduce a pause of 50 cs as that is the amount of time the system takes to set up. Following a series of steps, the system branches into two diamond-like loops, representing the expiration cycle on the right side and the inspiration cycle on the left side. The model uses two clocks: a is used to define timing constraints at various locations and transitions, while pause is employed for the timing constraints of pauses through the simulation (inspiratory, expiratory, or from the implementation itself). Initially, the top three locations, Idle, Setup and Start synchronize with the Setup automaton to set the initial parameters. After the configuration step, the model synchronizes with the Injector models to achieve the appropriate mixture of air and oxygen as determined by the FiO2 and desired flow parameters. Subsequently, the inspiration phase commences at the openInsp location with the invariant $pause \leq waitd$, where $waitd = \Delta t * 100$, modeling the time step of the real system in cs, which directly affects when a reading is taken. Once the outgoing transition is taken, the flow is updated based on the fluid resistance relationship described in Equation 3. This relationship is implemented as follows:

```
double getQ(double p_i, double p_j, double l) {
    return (PI * (p_i-p_j) * pow(d1,4)) /
    (l*128*w);
}
```

Thus, the pressure in the container to which the flow is directed is also updated using Equation 8 like this:

```
double getDelta() {
    return (Q*dt*R*rho)/vol2;
}
```

At the Insp location, if the elapsed time is less than the configured inspiration time ($a < tInsp$), the loop starts again; otherwise, it reaches the InsPause location, and the system stays there for the duration of the inspiration pause. At the end of the inspiratory pause, the system enters the openExp location, indicating the expiratory cycle's beginning. At this point, if $a < totalT$, then the expiration loop is repeated; otherwise ($a \geq totalT$), it means that there's no more fluid to extract from the patient's lungs ($Q = 0$), so the expOff! signal is fired, telling the ExpValve automaton to close its valve, and the expiration pause starts at ExpPause for pExp cs. After that, the clock a in the outgoing transition is reset to zero, and the whole cycle starts again.

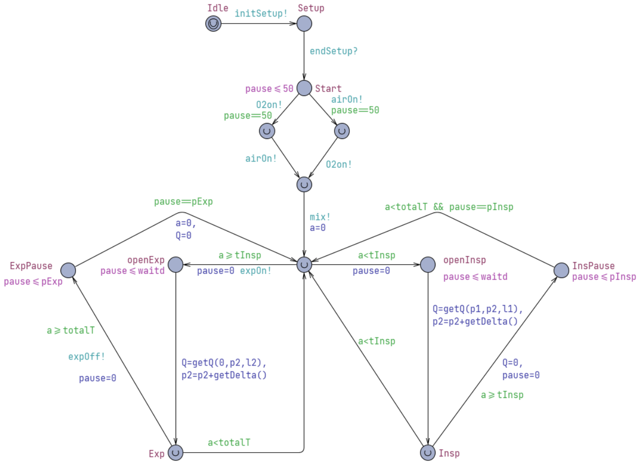

Figure 6. Control automaton

Injector

The Injector model in Figure 7 represents the O2 and air injector on/off valves that control the flow of oxygen and air into the mixing chamber. This model synchronizes with the rest of the system to determine when to turn on and off. Once the valves are turned on, they remain in that state for the entire duration of the inspiration time, denoted with *tInsp*.

ExpValve

The ExpValve model in Figure 8 operates similarly to the Injector model. Once it enters the on state, it remains in that state for a duration of tExp (in centiseconds), representing the operation of the expiration valve.

The next section presents the symbolic and statistical properties to be checked and their meaning.

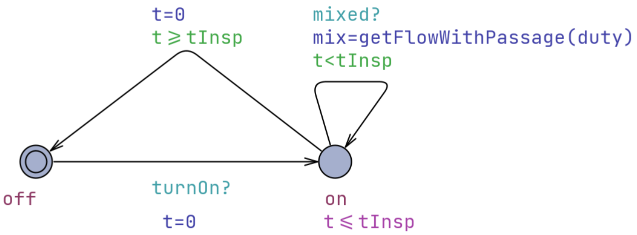

Figure 7. Injector model

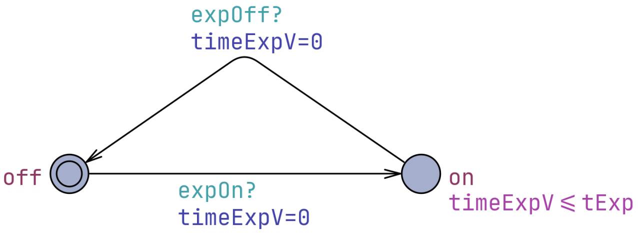

Figure 8. TA for the expiratory valve.

5 Property Verification

The following properties were verified upon the previously presented UPPAAL model:

Property 1

```
A[] not deadlock
```

At every stage, at least one transition will be eventually enabled.

Property 2

```
E<> Control.Insp
```

It is possible to reach the inspiration state.

Property 3

```
E<> Control.Exp
```

It is possible to reach the expiration state.

Property 4

```
A[] (not (InjectorO2.on or InjectorAir.on)
or ExpValve.off)
```

If there is an inspiration injector on, then the expiration valve is closed.

Property 5

```
E<> (InjectorO2.on or InjectorAir.on)
```

It is possible to reach states where injectors are on. This property also verifies that the previous property has a possible antecedent.

Property 6

```
A[] (not ExpValve.on or (InjectorO2.off
and InjectorAir.off))
```

For every path, if the expiration valve is on, both injectors must be off.

Property 7

```
E<> ExpValve.on
```

It is possible to reach a state where the expiration valve is on. This property also verifies that the previous property has a possible antecedent.

Property 8

```
A[] (not ExpValve.on or
not(ExpValve.time > tExp))
```

The expiration valve never lasts more than the configured duration for the expiration phase (tExp) turned on.

Property 9

```
A[] (not (InjectorO2.on and
InjectorAir.on) or not(InjectorO2.time > tInsp))
```

The oxygen and air injectors are never on for more than the configured inspiration time (tInsp).

Property 10

```
Pr[Ventilator.a <= tInsp;500](<> Ventilator.Insp)
```

Tells the probability of reaching the inspiration state too early over 500 runs. This is a safety property to ensure the patient receives air to her lungs at the appropriate time.

Property 11

```
Pr[Ventilator.a <= totalT+pExp]
(<> Ventilator.ExpPause)
```

Computes the probability that the expiratory pause is reached during its allotted timeframe.

Property 12

```
simulate[<=totalT]{q}
```

Displays the values that the flow variable q takes along one respiratory cycle (totalT).

Property 13

```
E[<=totalT;500](max:q)
```

This SMC query checks the maximum value attained during the total duration (totalT) of the ventilation over 500 runs. This property is useful to validate how much the actual flow in the system deviates from the desired flow.

Property 14

```
E[<=totalT;500](min:q)
```

Checks the minimum value attained during the total duration (totalT) of the ventilation over 500 runs. This property is useful to validate how much the actual flow in the system deviates from the desired flow.

Query	CPU Time (ms)	States explored	Result
1	60	2590	Satisfied
2	50	17396	Satisfied
4	50	17396	Satisfied
5	30	6772	Satisfied
6	10	4356	Satisfied
7	25	17546	Satisfied
8	30	20546	Satisfied
9	35	20861	Satisfied

Table 2. Summarized results of verification.

6 Verification Results

This section presents the results after executing the properties described in section 5 against multiple input parameters. The model is available at https://github.com/ventynet/ventynet/blob/master/ventilator.xml. It is important to mention that we verify symbolic and stochastic properties using the same model. In our opinion, this kind of mixed symbolic and stochastic verification offers (1) improved reliability because symbolic verification guarantees that a given formula holds (or not) for all possible states, and (2) additional insights via stochastic queries to survey the numerical behavior of a system such as this one, where not only time constraints must be met, but also constraints relative to numeric variables. Finally, to not have state doubles, we used integer variables scaled up to avoid a significant loss of precision. We will check border and middle values for the input parameters according to their specified ranges:

- Flow [10,50] L/min,
- FiO2 [10,100] %,
- Inspiration and expiration times [100,300] cs,
- Inspiration and expiration pauses [0,30] cs.

Table 2 displays the average values obtained from running the model and verifying its symbolic properties between 1 and 9, using various combinations of boundary values for the input parameters. Based on these results, we can confidently conclude that the model adheres to its specification since all safety and liveness properties were successfully satisfied.

Additionally, Table 3 shows the statistical queries result (i.e., properties 10 and 11) with confidence of 95% in all cases, for example in the case of Property 11, which is a safety property (i.e. something bad never happens) has a probability of less than 0.07% which says that it will most likely never happen.

Query	CPU Time (ms)	Probability
10	61	< 0.07%
11	99	≥ 91%

Table 3. Summarized results of SMC verification.

Query	CPU Time (ms)	Value
13	219	≈ 29.2622
14	218	≈ -32.4605

Table 4. Result of the last two SMC properties for a desired flow of 30 L min^{-1}.

Comparison of results with a physical mechanical ventilator

In this section, we compare the flow rate values obtained from the model using Property 12 against the experimental readings from the physical plant. The goal is twofold: (1) to identify potential areas for improvement revealed by the model and (2) to help establish the ideal error in the system, which in this case is defined as the relative error with respect to the delivered volume (V) as in [26]:

$$\delta = 100\% * \left| \frac{\mathrm{V} - \mathrm{V}_{observed}}{\mathrm{V}} \right| \tag{9}$$

In this context, V represents the desired volume, and $\mathrm{V}_{observed}$ is the volume observed from various simulations of the model. Since the model is a theoretical abstraction of the actual system, δ indicates the relative error that would exist in an ideal system under perfect conditions.

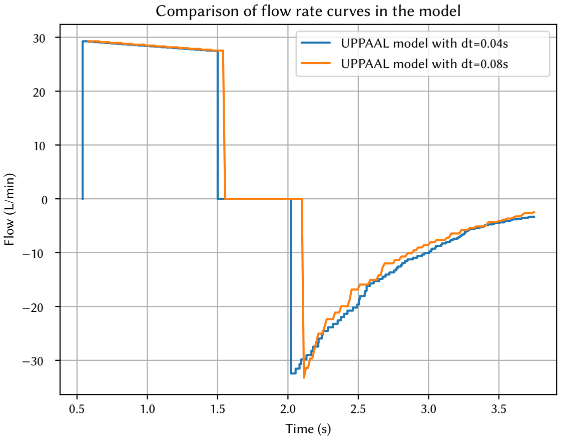

Figure 9. Flow rate over time in the model with different values for Δt.

Firstly, Figure 9 shows two smooth line plots from the UPPAAL using different delay times (Δt). Using a bigger Δt samples data more sparsely, and to fill in the gaps, we perform interpolation to estimate the missing values and ensure a seamless comparison between the two sets. The resulting plot showcases the trends and variations in the data; however, it is easy to see that the two are very similar, especially in the inspiration phase, i.e., where *time* <1.2 s. Furthermore, the reproduced plot goes in line with the classic curves of flow rate vs. time in a volume-controlled mechanical ventilator found in the literature [40].

Observed volume with Δt = 0.08 s 435.340 322 5 mL	δ 3.26%
Observed volume with Δt = 0.04 s $453.6795\overline{3}$ mL	δ 0.82%
Expected volume 450 mL	

Table 5. Relative error (Equation 9) with respect to delivered volume after varying Δt in the model.

Moreover, we calculated the relative error present in those simulations and presented it in Table 5. One notable finding was that altering the time step (Δt) had a considerable impact on the system's relative error (δ), reducing it from 5% to about $\approx$ 3%. This reduction is significant, as it translates to a difference of around 10 mL of oxygen-enriched air delivered to the patient by simply taking samples 0.04 s earlier. In critical situations or with specific patient conditions, even such small changes in ventilation can be relevant and have implications for patient care [34].

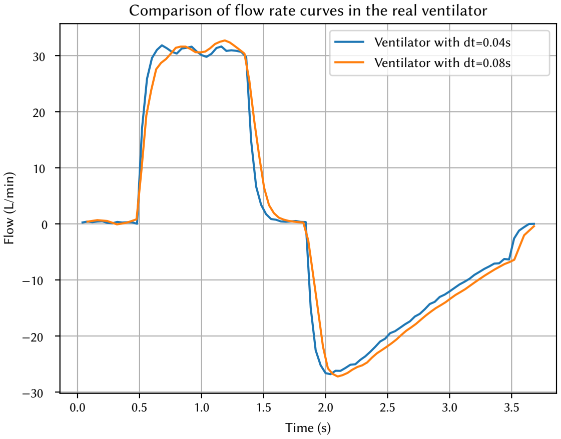

Figure 10. Flow rate over time in the ventilator with different time step duration Δt.

Observed volume with Δt =0.08 s 472.324 887 3 mL	δ 5.0%
Observed volume with Δt = 0.08 s 461.641 811 7 mL	δ 2.6%
Expected volume 450 mL	

Table 6. Relative error (Equation 9) with respect to delivered volume after varying Δt in the mechanical ventilator.

After seeing the results of Table 5 we decided to test the physical plant under similar conditions and see if reducing the delay time also decreased relative error, Table 6 shows the experimental results after doing so, and Figure 10 shows the generated curves from the readings of the IRL ventilator. Indeed, we got a similar behavior in the system for the same amount of volume to inject (450 mL): we were able to reduce the relative error from 3.26% down to 0.86% by using Δt =0.04 s. Since 0.04 s is the lowest the physical ventilator can go regarding delay time, we conclude that the ideal relative error in the system is 0.86%.

Discussion

The application of a UPPAAL-based model has proven to be an effective approach for evaluating and improving the mechanical ventilation system. By reducing the delay time (Δt) from 0.08 s to 0.04 s, a significant enhancement in the performance of the mechanical ventilator was achieved, resulting in a reduction of the relative error associated with the volume of air delivered to the set air volume from 5% to 2.6%.

These results demonstrate the potential for improvement in the system, highlighting that a faster and more synchronized response between the actuators and the processor can have a significant impact on the accuracy and efficiency of the mechanical ventilator. This improvement holds great importance for mechanical ventilators for several key reasons:

1. Accuracy in Air Volume Delivery: Ensuring accurate and safe ventilation for patients requiring ventilatory support is crucial. Even small variations in air delivery can have significant consequences on the patient's well-being, affecting oxygenation and increasing the risk of respiratory complications.
2. Efficiency Optimization: Reducing the relative error in air delivery leads to greater efficiency in the mechanical ventilator's operation. By minimizing discrepancies between the set and delivered volume, wastage is reduced, and resource consumption is optimized, resulting in a more sustainable and cost-effective operation of the equipment.
3. Confidence and Reliability: A lower ventilation error rate instills greater confidence in healthcare professionals regarding the mechanical ventilator's accuracy and reliability in providing respiratory support. This confidence becomes especially critical in critical situations, such as in intensive care units, where precise ventilation can significantly impact patient recovery.

It is crucial to emphasize that the direct modeling of a mechanical ventilator using UPPAAL has certain practical drawbacks. First, UPPAAL-SMC uses the Euler method for solving differential equations [4], which is known to be less accurate and causes a large performance overhead compared to an analytical method, especially if we want to reason about long simulation runs. UPPAAL-SMC is not optimized for running long simulations. Second, because UPPAAL-SMC uses numerical methods to solve the flow equations, simulating a dynamic model requires significantly more computational resources than simulating a non-dynamic model. Based on the above characteristics and on the formal modeling of mechanical ventilators using UPPAAL, the authors propose integrating a supervisory system to detect and isolate failures in real-time for a mechanical ventilator. The results of this implementation will be shared with the scientific community in a future article. Additionally, based on the verified properties presented, the authors propose scaling the mechanical ventilator's functionality to allow online configuration of operating parameters and monitoring of critical variables. These future results are expected to provide valuable insights into further enhancing the mechanical ventilator system.

7 Conclusions and Future Work

In this paper, we present a formal model of a mechanical ventilator using UPPAAL to enhance its reliability, identify potential areas for improvement, and provide a basis for rigorous verification and validation of such devices. Our model captures essential safety and liveness properties, allowing us to reason about its completeness and closeness to the real device behavior.

The primary contribution of this work lies in the development and analysis of the formal model of a mechanical ventilator, which, to the best of our knowledge, has not been done before in UPPAAL. In addition, we provide valuable insights into the selection of an optimal delay time (Δt) supported by the model. Our approach sets the stage for future modeling and verification of these devices without compromising expressiveness.

Furthermore, our study demonstrates the effectiveness and benefits of using formal methods in medical device design. The systematic approach provided by formal modeling and verification can significantly increase confidence in the correctness and reliability of the device, contributing to its successful implementation in real-world healthcare environments.

While our formal model has shown promising results, there are still areas for future exploration. One avenue for further research could be to refine the model to include more complex device characteristics or environmental influences. In addition, the model can be further validated and enhanced by incorporating additional real-world data and test cases, improving its accuracy and practical applicability.

Acknowledgments

Special thanks to Universidad del Valle, which funded this project via the 1898 and 1900 consecutive, and to the multilateral agreement between the University of Namur and Universidad del Valle.

References

[1] Rajeev Alur and David L. Dill. 1994. A Theory of Timed Automata. *Theoretical Computer Science* 126 (1994), 183–235.

[2] Jaime Arias, Kyungmin Bae, Carlos Olarte, Peter Ölveczky, Laure Petrucci, and Fredrik Rømming. 2022. Rewriting Logic Semantics and Symbolic Analysis for Parametric Timed Automata. In *8th International Workshop on Formal Techniques for Safety-Critical Systems, FTSCS 2022, Auckland, New Zealand, December 7, 2022 (FTSCS 2022)*, Cyrille Artho and Peter Csaba Ölveczky (Eds.). ACM, 3–15. https://doi.org/10.1145/3563822.3569923

[3] David Arney, Raoul Jetley, Paul Jones, Insup Lee, and Oleg Sokolsky. 2007. Formal Methods Based Development of a PCA Infusion Pump Reference Model: Generic Infusion Pump (GIP) Project. In *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*. 23–33. https://doi.org/10.1109/HCMDSS-MDPnP.2007.36

[4] Kendall E. Atkinson. 1989. *An Introduction to Numerical Analysis* (second ed.). John Wiley & Sons, New York. http://www.worldcat.org/isbn/0471500232

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

[6] Davide Basile, Maurice H. ter Beek, and Vincenzo Ciancia. 2018. Statistical Model Checking of a Moving Block Railway Signalling Scenario with Uppaal SMC. In *Leveraging Applications of Formal Methods, Verification and Validation. Verification*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 372–391.

[7] Gerd Behrmann, Alexandre David, and Kim G. Larsen. 2004. *A Tutorial on Uppaal*. Springer Berlin Heidelberg, Berlin, Heidelberg, 200–236. https://doi.org/10.1007/978-3-540-30080-9_7

[8] Gerd Behrmann, Alexandre David, and Kim G Larsen. 2006. A tutorial on Uppaal 4.0. *Department of computer science, Aalborg University* (2006).

[9] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. 1998. Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundam. Informaticae* 36, 2-3 (1998), 145–182. https://doi.org/10.3233/FI-1998-36233

[10] Patricia Bouyer, François Laroussinie, and Pierre-Alain Reynier. 2005. Diagonal Constraints in Timed Automata: Forward Analysis of Timed Systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*. https://api.semanticscholar.org/CorpusID:14842124

[11] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. 1998. Kronos: a model-checking tool for real-time systems. In *Computer Aided Verification 10th International Conference, CAV'98 (Lecture Notes in Computer Science, Vol. 1427)*, Hu, Alan J.; Vardi, and Moshe Y. (Eds.). Vancouver, BC, Canada, 546–549. https://doi.org/10.1007/BFb0028779

[12] Steven C Chapra, Raymond P Canale, Reyna Susana García Ruiz, Víctor Hugo Ibarra Mercado, Enrique Muñoz Díaz, and Guillermo Evangelista Benites. 2011. *Métodos numéricos para ingenieros*. Vol. 5. McGraw-Hill New York, NY, USA.

[13] Cheng Chen and James M Ricles. 2009. Analysis of actuator delay compensation methods for real-time testing. *Engineering Structures* 31, 11 (2009), 2643–2655.

[14] Edmund M. Clarke and Jeannette M. Wing. 1996. Formal Methods: State of the Art and Future Directions. *ACM Comput. Surv.* 28, 4 (dec 1996), 626–643. https://doi.org/10.1145/242223.242257

[15] Jaime Cuartas, Jesús Aranda, Maxime Cordy, James Ortiz, Gilles Perrouin, and Pierre-Yves Schobbens. 2023. MUPPAAL: Reducing and Removing Equivalent and Duplicate Mutants in UPPAAL. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 52–61. https://doi.org/10.1109/ICSTW58534.2023.00021

[16] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. 2015. Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer* 17, 4 (2015), 397–415.

[17] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikuăionis, and Danny Bogsted Poulsen. 2015. Uppaal SMC Tutorial. *Int. J. Softw. Tools Technol. Transf.* 17, 4 (Aug. 2015), 397–415.

[18] Clarence W De Silva. 2017. *Modeling of dynamic systems with engineering applications*. CRC Press.

[19] Ewan C Goligher, Niall D Ferguson, and Laurent J Brochard. 2016. Clinical challenges in mechanical ventilation. *The Lancet* 387, 10030 (April 2016), 1856–1866. https://doi.org/10.1016/s0140-6736(16)30176-3

[20] Elsa L. Gunter, Insup Lee, Jaime Lee, Wonhong Nam, Frederick Pearce, Steve Van Albert, Jiaxiang Zhou, Rajeev Alur, and David Arney. 2004. Formal specifications and analysis of the computer-assisted resuscitation algorithm (CARA) Infusion Pump Control System. *International Journal on Software Tools for Technology Transfer (STTT)* 5, 4 (May 2004), 308–319. https://doi.org/10.1007/s10009-003-0132-7

[21] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. 1997. HyTech: A Model Checker for Hybrid Systems. *Software Tools for Technology Transfer* 1 (1997), 460–463.

[22] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. 1995. What's decidable about hybrid automata? (1995), 373–382. https://doi.org/10.1145/225058.225162

[23] Dean R Hess and Robert M Kacmarek. 2019. *Essentials of mechanical ventilation*. McGraw Hill Education.

[24] Michaela Huhn and Sara Bessling. 2013. Enhancing Product Line Development by Safety Requirements and Verification. In *Foundations of Health Information Engineering and Systems*, Jens Weber and Isabelle Perseil (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 37–54.

[25] Frank P Incropera and David P DeWitt. 2002. Heat and mass transfer. *Fifth John Wiley and Sons* (2002), 533–593.

[26] David Ronald Kincaid and Elliott Ward Cheney. 2009. *Numerical analysis: mathematics of scientific computing*. Vol. 2. American Mathematical Soc.

[27] Verena Klös, Thomas Göthel, and Sabine Glesner. 2016. Formal models for analysing dynamic adaptation behaviour in real-time systems. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 106–111.

[28] Jan Willem Kuiper, A B. Johan Groeneveld, Arthur S. Slutsky, and Frans B. Plötz. 2005. Mechanical ventilation and acute renal failure. *Critical Care Medicine* 33, 6 (June 2005), 1408–1415. https://doi.org/10.1097/01.ccm.0000165808.30416.ef

[29] Kim G Larsen. 2014. Verification and performance analysis of embedded and cyber-physical systems using UPPAAL. In *2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. IEEE, IS–11.

[30] Dominique Méry, Bernhard Schätz, and Alan Wassyng. 2014. The Pacemaker Challenge: Developing Certifiable Medical Devices (Dagstuhl Seminar 14062). *Dagstuhl Reports* 4, 2 (2014), 17–37. https://doi.org/10.4230/DagRep.4.2.17

[31] Miroslav Pajic, Zhihao Jiang, Insup Lee, Oleg Sokolsky, and Rahul Mangharam. 2012. From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study. In *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*. 173–184. https://doi.org/10.1109/RTAS.2012.25

[32] V Marco Ranieri, Rocco Giuliani, Gilda Cinnella, Caterina Pesce, Niki Brienza, EUSTACHIO L Ippolito, Vincenzo Pomo, Tommaso Fiore, Stewart B Gottfried, Antonio Brienza, et al. 1993. Physiologic effects of positive end-expiratory pressure in patients with chronic obstructive pulmonary disease during acute ventilatory failure and controlled mechanical ventilation. *American Review of Respiratory Disease* 147, 1 (1993), 5–13.

[33] Sunil K Sinha and Steven M Donn. 2000. Weaning from assisted ventilation: art or science? *Archives of Disease in Childhood - Fetal and Neonatal Edition* 83, 1 (2000), F64–F70. https://doi.org/10.1136/fn.83.1.F64 arXiv:https://fn.bmj.com/content/83/1/F64.full.pdf

[34] Arthur S. Slutsky. 1999. Lung Injury Caused by Mechanical Ventilation. *Chest* 116 (July 1999), 9S–15S. https://doi.org/10.1378/chest.116.suppl_1.9s-a

[35] Annette ten Teije, Mar Marcos, Michel Balser, Joyce van Croonenborg, Christoph Duelli, Frank van Harmelen, Peter Lucas, Silvia Miksch, Wolfgang Reif, Kitty Rosenbrand, and Andreas Seyfang. 2006. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine* 36, 3 (March 2006), 193–209. https://doi.org/10.1016/j.artmed.2005.10.006

[36] Gloria Toro Córdoba, Fanny Gómez, and José Melo. 2023. Artículo original producto de la investigación Design and development of a virtual learning environment in mechanical ventilation with teleoperated practice -VENTYLAB. (01 2023). https://doi.org/110.22490/24629448.6925

[37] Qurat Ul Ain and Osman Hasan. 2019. Formal Timing Analysis of Digital Circuits. In *Formal Techniques for Safety-Critical Systems*, Cyrille Artho and Peter Csaba Ölveczky (Eds.). Springer International Publishing, Cham, 84–100.

[38] Andres M Valencia, Jesus Caratar, and Jose Garcia. 2022. Modeling of an online learning architecture for mechanic ventilation integrating teleoperated equipment using colored Petri nets. *Material Science and Engineering International Journal* 6, 4 (Dec. 2022), 158–165. https://doi.org/10.15406/mseij.2022.06.00194

[39] Danny Weyns and Usman Iftikhar. 2016. Model-based simulation at runtime for self-adaptive systems. *Proceeding Models at Runtime, Würzburg 2016* (2016), 1–9.

[40] Fikret Yalcinkaya, Mustafa Emrah Yildirim, and Hamza Ünsal. 2015. Pressure -Volume Controlled Mechanical Ventilator: Modeling and Simulation.

Received 2023-07-21; accepted 2023-08-27