The Mechanical Lung Ventilator Case Study

Silvia Bonfanti(✉) and Angelo Gargantini

University of Bergamo, Bergamo, Italy
{silvia.bonfanti,angelo.gargantini}@unibg.it

Abstract. This paper introduces the ABZ 2024 Case Study: Mechanical Lung Ventilator (MLV), inspired by the Mechanical Ventilator Milano developed during COVID-19. The case study reports the specification of the Mechanical Lung Ventilator used to ventilate patients who are not able to breathe on their own or need ventilation support. Expected contributions to the case study include, among others, modeling, validation and verification, management of temporal behavior, modeling of the graphical user interface or automatically generating executable source code.

Keywords: ABZ · State Based Formal Methods · Mechanical Lung Ventilator · Case Study

1 Introduction

At the beginning of the COVID-19 pandemic, the region around Bergamo, where ABZ 2024 took place, was hit very hard by COVID-19, as suggested by the high number of hospitalizations and deaths. During those days, a group of researchers was involved in the design, development, and certification of an electro-mechanical lung ventilator called MVM (Mechanical Ventilator Milano)[1] [1]. The project started from the idea of the physicist Cristiano Galbiati, who was soon joined by dozens of physicists, engineers, physicians, and computer scientists from 12 countries around the world[2]. The team was able to realize a ventilator that is reliable, easily reproducible on a large scale, available in a short amount of time, and at a limited cost [4,7]. The MVM has obtained the FDA (Food and Drug Administration) Emergency Use Authorization (EUA) followed by authorizations issued by Health Canada and the CE marking.

The specification of the mechanical lung ventilator chosen as case study for ABZ 2024, is inspired by MVM, with some simplifications to make it suitable as a case study:

[1] https://mvm.care/.

[2] At that period, many projects on mechanical ventilator started, but only a few get certified https://github.com/PubInv/covid19-vent-list.

S. Bonfanti et al. (Eds.): ABZ 2024, LNCS 14759, pp. 281–288, 2024.
https://doi.org/10.1007/978-3-031-63790-2_23

– we have removed one component, the supervisor which was responsible for monitoring the controller, the GUI, and the hardware. In the case of errors, it raises alarms if not already raised by the controller or the GUI, ensuring patient safety.
– we use only visual alarms, instead of audio and visual alarms.

The requirement specification for the ABZ 2024 Case Study is available here [3]. That document is NOT intended to be used as software requirements specification of a real ventilator.

Note that this case study is one of those proposed by the ABZ conference, all the case studies can be found here: https://abz-conf.org/case-studies/.

2 Mechanical Lung Ventilator

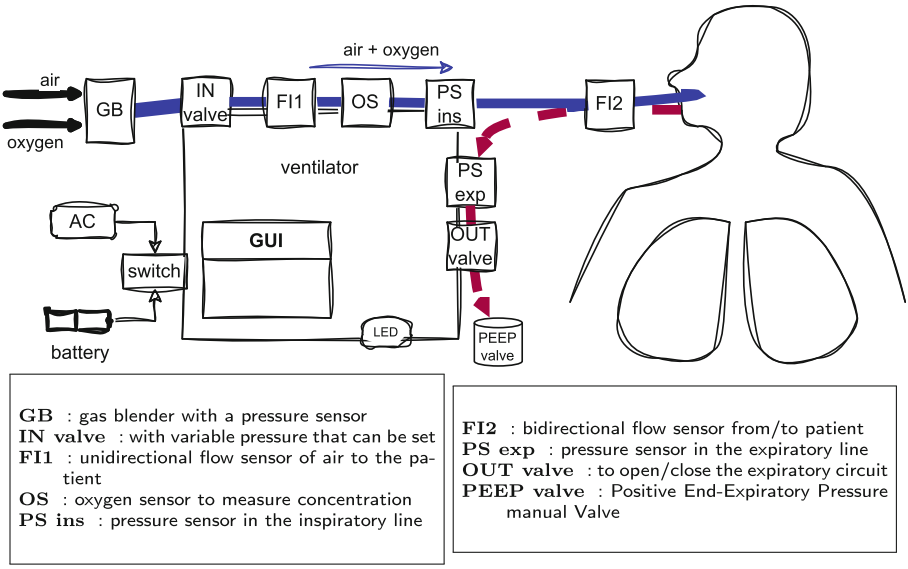

Fig. 1. High level view of ventilator sensors

The mechanical lung ventilator is intended to ventilate patients in intensive therapy who require mechanical ventilation [8]. The ventilator proposed for the ABZ 2024 Case Study works in pressure mode, i.e., the respiratory time cycle of the patient is controlled by pressure, and, therefore, this ventilator requires a source of compressed oxygen and medical air readily available in intensive care units.

The overall structure of the ventilator proposed by the case study is depicted in Fig. 1. The pressurized air enters the inspiration circuit (continuous line) and is mixed with the oxygen. Its flux is controlled by a valve (IN) and monitored by

several sensors. The air enters the breathing circuit with a flux monitored (by FI2), then, after being inspired by the lungs, it expires and exits the breathing circuit (dashed arrow) with its pressure monitored by a sensor (PS exp) and controlled by two valves, one (OUT valve) is controlled by the machine while the other (PEEP) is manually set to provide a constant Positive End-Expiratory Pressure.

2.1 Ventilation Modes

By alternating the opening and closing of the two valves, the MLV governs the entering/exiting flux and air pressure in the lungs. More precisely, the ventilator has two operative modes: *Pressure Controlled Ventilation* (PCV) and *Pressure Support Ventilation* (PSV). In the PCV mode, the respiratory cycle is kept constant and the pressure level changes between the target inspiratory pressure (P_{insp}) and the positive end-expiratory pressure (PEEP). New inspiration is initiated either after a breathing cycle is over, or when the patient spontaneously initiates a breath. In the former case, the breathing cycle is controlled by two parameters: the respiratory rate (RR) and the ratio between the inspiratory and expiratory times (I:E). In the latter case, a spontaneous breath is triggered when the ventilator detects a sudden pressure drop within the trigger window during expiration. The PSV mode is unsuitable for patients who are not able to start breathing on their own. The respiratory cycle is controlled by the patient, and the ventilator partially takes over the work of breathing. A new respiratory cycle is initiated with the inspiratory phase, detected by the ventilator when a sudden pressure drop occurs. When the patient's inspiratory flow drops below a set fraction of the peak flow, the ventilator stops the pressure support, thus allowing exhalation. If a new inspiratory phase is not detected within a certain amount of time (apnea lag), the ventilator will automatically switch to the PCV mode because it is assumed that the patient is not able to breathe alone.

The ventilator allows the air to enter/exit through two valves, i.e., an input valve and an output valve. When the ventilator is not running, the valves are set to safe mode: the input valve is closed and the output valve is opened. In this configuration, the ventilator does not prevent breathing thanks to some relief valves.

When the inspiration starts, the input valve is opened and the output valve is closed, while during the expiration the input valve is closed and the output valve is opened. Both in PCV and PSV mode, inspiratory pause, expiratory pause, and recruitment maneuver are allowed by user request. Inspiratory/Expiratory pause consists in closing the input and output valves of the ventilator respectively after the inspiration and expiration phases. The inspiratory pause allows measuring the pressure reached inside the alveoli at the end of the inspiratory cycle, while the expiratory pause allows measuring the residual pressure to check possible obstruction in the exhalation channel. The recruitment maneuver is an emergency procedure required after intubation, and it consists of prolonged lung inflation as necessary to reactivate the alveoli immediately; during this maneuver, the input valve is opened and the output valve is closed.

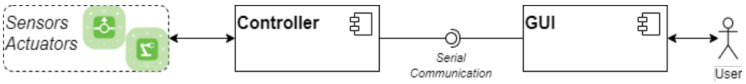

Fig. 2. The high-level software architecture

2.2 Software Architecture

The high-level software architecture, shown in Fig. 2, illustrates the communication among the software components: graphical user interface (GUI) and controller. The GUI is a touchscreen panel that displays the information needed to check the respiratory condition, allows parameter setting, and displays ventilation parameters and alarm settings. When the controller receives operator input from the GUI, it communicates with the valve controllers, serial interfaces, and other subcomponents and sends them commands.

Before starting the ventilation, the ventilator controller passed through three phases. The *start-up* in which the controller is initialized with default parameters, *self-test* which ensures that the hardware is fully functional, and *ventilation off* in which the controller is ready for ventilation when requested. If during ventilation and other phases, the controller detects a severe condition that prevents the ventilator from sustaining the ventilation, the machine is brought to fail-safe mode (in valve closed and out valve open).

3 Structure of the Specification Document

In this section, we explain the structure of the requirement specification document available here [3]. In this article, we do not detail the requirements since there are around 370 requirements.

The chapter 2. System Requirements contains the general high-level specification of the ventilator: functional requirements (2.1), measured and displayed parameters (2.2), values and ranges of parameters (2.3), interfaces between components (2.4), and alarm requirements (2.5).

The chapter 3 GUI Requirements consists of the specifications of the GUI, which is responsible for receiving information from the user and displaying information to the user. It details each mode of operation: Start-up Mode (3.1), Start Mode (3.2), Menu Mode (3.3), Self-Test Mode (3.4), Ventilation Mode (3.5), Show Real Time Data Mode (3.6), Settings Mode (3.7), Frozen Mode (3.8), and Alarm settings Mode (3.9).

The chapter 4 Controller Requirements comprehends the specification of the controller, which is responsible for controlling the phase of the respiratory cycle (inhalation, pause, exhalation) by operating on the valves and receiving information from sensors and commands from the GUI. It details each mode of operation: Start-up Mode (4.1), Self Test (4.2), PCV Mode (4.3), PSV Mode (4.4), and requirements common to multiple modes like inspiratory/expiratory pauses (4.5).

The chapter 5 Alarms presents the specification of the alarm system responsible for raising alarms.

Requirement Numbering Convention

The specification is divided into requirements, and the format of each requirement ID is: XXX.n.y, where: XXX is a three-letter code indicating a requirement type, n is the requirement number, and y is the sub-requirement number (it is empty if a parent). The following table reports the connection between the three-letter code and the type of requirement.

Three-letter code	Description
AL	Alarm requirements
CONT	Controller requirements
FUN	Functional (general) requirements
GUI	GUI requirements
INT	Interfaces requirements
PER	Values and ranges requirements
SAV	Safety requirements

Table 1 reports an example of a GUI requirement:

Table 1. Example of GUI requirement

ID	Requirement
GUI.7	The transition from Menu to Ventilation shall occur when the Self-test is passed if required (new patient connected), and the clinician wants to proceed with the ventilation

Scenario

Besides the requirement specification document, we have released the scenarios for the initialization. Each scenario reports the state of the controller and the GUI and the events that lead to the state change. For example, the scenario in Table 2 refers to the new patient connection: when a new patient is connected, the doctor selects a new patient and runs the self-test. Once the self-test is passed, the ventilation can start.

4 Suggested Outcomes

During the development of the MVM software, no formal method has been applied, mainly because of a lack of developers' skills with any formal method.

Table 2. Scenario example

	event	*powerOn*	*start-up ended*	–	*selfTestPassed*
Controller	**state**	Start-Up	SelfTest	SelfTest	VentilationOff
	event	*powerOn*	*start-up ended*	*new Patient*	*selfTestPassed*
GUI	**state**	Start-Up	Start	SelfTest	Menu

However, we want to propose this case study to demonstrate the feasibility of developing the ventilator by using a formal method-based approach. Mechanical lung ventilators, as well as other medical devices which incorporate software, must be certified before their use. Several standards for the validation of medical devices have been proposed - as ISO 13485, ISO 14971, IEC 60601-1, EU Directive 2007/47/EC -, but they mainly consider hardware aspects of the physical components of a device, and do not mention the software component. The only reference concerning the regulation of medical software is the standard IEC (International Electrotechnical Commission) 62304. This standard provides a very general description of common life cycle activities of the software development, without giving any indication regarding process models, or methods and techniques to assure safety and reliability.

With this case study, we aim to study the applicability of formal methods in the software development of medical devices to satisfy the standards, in this case, the IEC 62304.

We have envisioned several aspects of the ventilator that could be the object of research activities. In the following, we give a non-exhaustive list of possible outcomes.

- A classical approach consists of modeling the system or part of it and applying the classical V&V activities (as partially done [2]), like formal verification of the correctness or validation of scenarios. One could check that the system behavior is correct, like in case of some errors, the system goes into a fail-safe mode.
- A critical aspect of the system is its temporal behavior. Many properties and constraints have explicit temporal requirements (like after 10 s ...). One could model these aspects and make a temporal analysis of the system.
- After the good experience with ABZ2023, we decided to include the GUI. Research activities could model this critical component and analyze the human-computer interaction.
- Generation of executable source code and implement a prototype of the ventilator on a simple electronic board like Arduino (or part of it).

5 Contributions to ABZ 2024

In this section, we resume (in random order) the contributions accepted for publication in the ABZ 2024 proceedings. All the papers use different formal modeling techniques and contribute to different outcomes.

Paper [9] illustrates the correct-by-construction approach and introduces an `Event-B` formal model of the MLV controller and part of alarms. Validation and verification techniques are applied using the `ProB` model checker to validate and verify the specification.

In paper [10] the authors model the controller of the MLV using Timed Algebraic State-Transition Diagrams (`TASTD`). Then the specification is validated using `cASTD` compiler to translate the specification into `C++` code.

Paper [6] models controller and general requirements using the Formal Requirements Elicitation Tool (`FRET`) to provide traceability from natural-language requirements to a formal design model. The authors explore the link between formal requirements in `FRET` and formal specifications in `Event-B`, presenting how techniques like `FRET` can be used to guide the development of a formal model in a large case study in a state-based technique (`Event-B` in this case).

The MLV is modeled in the process algebra mCRL2 in [5]. The functional requirements of the MLV are formalized in the modal μ-calculus, and the model checker is used to analyze whether these requirements hold true in the model. Each scenario provided with the case study has been can be captured in a modal μ-calculus formula and verified that the model satisfies those formulas. Their formalization helped us in revealing a few subtle incomplete or not completely clear requirements in the informal document and we have used the feedback to improve the original specification.

The Clock Constraint Specification Language (CCSL) has been applied to the case-study and the authors report their experience in [11]. CCSL captures the causal and temporal behavior of a system by specifying constraints on logical clocks. Logical clocks are integer counters where the occurrence of an event, a tick, advances the counter and marks the advance in time. The paper introduced some new real-time constructs to directly encode phenomena like clock drift, skew and jitter and these constructs are applied to the case study. Earlier versions of the paper, allowed us to clarify some temporal constraints (e.g., the different phases in the PCV mode).

Competing Interests. The author(s) has no competing interests to declare that are relevant to the content of this manuscript.

References

1. Abba, A., et al.: The novel mechanical ventilator Milano for the COVID-19 pandemic. Phys. Fluids **33**(3), 037122 (2021). https://doi.org/10.1063/5.0044445
2. Bombarda, A., Bonfanti, S., Gargantini, A., Riccobene, E.: Developing a prototype of a mechanical ventilator controller from requirements to code with ASMETA. Electron. Proc. Theor. Comput. Sci. **349**, 13–29 (2021)
3. Bonfanti, S., Gargantini, A.: Mechanical Lung Ventilator Requirements Specification, December 2024. https://github.com/foselab/abz2024_casestudy_MLV
4. Bonivento, W., Gargantini, A., Krücken, R., Razeto, A.: The mechanical ventilator Milano. Nucl. Phys. News **31**(3), 30–33 (2021)

5. van Dortmont, D., Keiren, J.J., Willemse, T.A.: Modelling and analysing a mechanical lung ventilator in mCRL2. In: Rigorous State-Based Methods 10th International Conference, ABZ 2024, Bergamo, Italy, 25–28 June 2024, Proceedings, LNCS, vol. 14759. Springer (2024)
6. Farrell, M., Luckcuck, M., Monahan, R., Reynolds, C., Sheridan, O.: FRETting and formal modelling: a mechanical lung ventilator. In: Rigorous State-Based Methods 10th International Conference, ABZ 2024, Bergamo, Italy, 25–28 June 2024, Proceedings, LNCS, vol. 14759. Springer (2024)
7. Guardo, M.C.D., et al.: When nothing is certain, anything is possible: open innovation and lean approach at MVM. R&D Manag. (2021). https://doi.org/10.1111/radm.12453
8. Lei, Y.: Medical Ventilator System Basics: A Clinical Guide. Oxford University Press, Oxford (2017). https://doi.org/10.1093/med/9780198784975.001.0001
9. Mammar, A.: An Event-B model of a mechanical lung ventilator. In: Rigorous State-Based Methods 10th International Conference, ABZ 2024, Bergamo, Italy, 25–28 June 2024, Proceedings, LNCS, vol. 14759. Springer (2024)
10. Ndouna1, A.R., Frappier, M.: Modelling the mechanical lung ventilation system using TASTD. In: Rigorous State-Based Methods 10th International Conference, ABZ 2024, Bergamo, Italy, 25–28 June 2024, Proceedings, LNCS, vol. 14759. Springer (2024)
11. Tokariev, P., Mallet, F.: Real-Time CCSL: application to the mechanical lung ventilator. In: Rigorous State-Based Methods 10th International Conference, ABZ 2024, Bergamo, Italy, 25–28 June 2024, Proceedings, LNCS, vol. 14759. Springer (2024)