



Combining DEVS simulation and ontological modeling for hierarchical analysis of the SARS-CoV-2 replication

Ali Ayadi¹ , Claudia Frydman², Wissame Laddada³,
Isabelle Imbert⁴ , Cecilia Zanni-Merk³ and Lina F Soualmia³

Abstract

This article presents an hybrid and hierarchical model in which two modeling and simulation approaches, discrete event system specification simulation (DEVS) and semantic technologies, were used together in order to help in the analysis of a major healthcare problem, the severe acute respiratory syndrome-coronavirus 2 (SARS-CoV-2). Indeed, the complexity of the SARS-CoV-2 replication process, and the range of hierarchical scales over which it interacts with cellular components (extending from genomic and transcriptomic to proteomic and metabolomic scales), and the intricate way in which they are interwoven, make its understanding very challenging. It is therefore crucial to model the different scales of the replication process, by taking into account all interactions with the infected cell. By combining the advantages of both DEVS simulation and ontological modeling, we propose a hierarchical ontology-based DEVS simulation model of the SARS-CoV-2 viral replication at both the micro-molecular (proteomic and metabolomic) and macro-molecular (genomic and transcriptomic) scales. First, we demonstrate the usefulness of combining DEVS simulation and semantic technologies in a common modeling framework to face the complexity of the SARS-CoV-2 viral replication at different scales. Second, the modeling and simulation of the SARS-CoV-2 replication process on different levels provide valuable information on the different stages of the virus's life cycle and lays the foundation for a system to anticipate future mutations selected by the virus.

Keywords

COVID-19, SARS-CoV-2 replication machinery, virus–host interactions, hierarchical modeling and simulation, DEVS simulation, ontological modeling

1. Introduction

At the end of December 2019, an infectious disease with unknown causes was discovered for the first time in the Chinese town of Wuhan.¹ The World Health Organization (WHO) has quickly declared a global health emergency and pandemic as the infectious disease spreads exponentially worldwide.² The WHO attributed this pandemic to a new coronavirus, called Severe Acute Respiratory Syndrome-Coronavirus-2 (SARS-CoV-2) which causes coronavirus disease-2019 (COVID-19).² Coronaviruses are an important group of single-stranded positive-sense RNA viruses that infect animals and humans.³ Since 2000, three new and highly pathogenic coronaviruses have emerged, SARS-CoV (2003), MERS-CoV (2012), and SARS-CoV-2 (2019).⁴ However, only the SARS-CoV-2 has become a pandemic within a matter of months. It attacks the respiratory system, leading to fatigue, fever, and respiratory

problems.⁵ The Coronavirus genome comprises about 30,000 nucleotides and encodes an exceptionally complex replication/transcription mechanism consisting of 16 viral non-structural proteins (nsps).⁶ By January 27, 2023, SARS-CoV-2 had infected nearly 674,408,760 people, including an estimated 6,755,708 confirmed deaths in at least 219 countries worldwide.¹

¹ICube CNRS UMR7357, University of Strasbourg, Strasbourg, France

²LIS CNRS UMR 7020, Aix Marseille Univ, Université de Toulon, Marseille, France

³Normandie Université, INSA Rouen, LITIS, Rouen, France

⁴LISM CNRS UMR 7255, University of Aix-Marseille, Marseille, France

Corresponding author:

Ali Ayadi, ICube CNRS UMR7357, University of Strasbourg, 300 bd Sébastien Brant, Illkirch, CS 10413, F-67412 Strasbourg, France.
Email: ali.ayadi@unistra.fr

In this scourge, governments around the world are imposing protection measures, i.e., washing hands, staying at home, covering mouths and noses, limiting social gatherings, and so on, to stop the spread of SARS-CoV-2. Meanwhile, health organizations and researchers have focused on this emergency as never before.⁷ In collaboration with biologists and clinicians, mathematical and computing communities have developed new algorithms and mathematical methods to tackle the COVID-19. The majority of the studies have been conducted to understand and characterize the relative risk of SARS-CoV-2 and its spread at a population level. But, there is still much work to learn about its replication (and consequently on the selection of mutations at the genome level) and interaction with host cell.⁸ From a mathematical and computational point of view, only a few studies have investigated the replication mechanism of the SARS-CoV-2, in particular the RNA-dependent RNA polymerase (RdRp), a key protein responsible for viral replication. Besides, the complexity of the SARS-CoV-2 replication machinery makes its design and simulation difficult.⁹

Moreover, the complexity of the SARS-CoV-2 replication process arises due to the range of hierarchical scales over which it interacts with cellular components, extending from genomic and transcriptomic to proteomic and metabolomic scales and the intricate way in which they are interwoven, making its understanding very challenging. Thus, the development of mathematical and computational tools to model the molecular mechanisms, underlying the replication of the SARS-CoV-2 viral genome and its interaction with host cells, warrant urgent investigation.

This work extends the previous study,¹⁰ where the DEVS formalism was combined with ontological models to provide a SARS-CoV-2 replication simulation framework. We extend this work to propose a hierarchical simulation model to represent the dynamics of the SARS-CoV-2 replication cycle at different levels: from the micro-level (the different biochemical reactions, such as the translation and the transcription sub-processes, leading to the production of the viral proteins and modifications that occur in amino acids) to the macro level (the cellular pathways hijacking by the virus to produce new virus particles). Here, our proposed hierarchical model consists of two interconnected modules: the ontological and DEVS simulation modules. The ontological module exploits the *OntoRepliCov* ontology,¹¹ an ontology dedicated to the description of the coronavirus replication process, to provide semantic knowledge about the CoV genome organization, SARS-CoV-2 sequences and the different viral proteins produced. Such micro-level knowledge cannot be provided by the DEVS module. Despite the fact that the DEVS formalism is modular, it is not able to model the micromolecular entities (e.g. at the metabolomic scale) since these entities are very compact and do not have any behavior (biological process) to be simulated and do not

have any inputs or outputs. Furthermore, these micromolecular entities have a simple and basic structure, as they constitute the basic elements of macromolecular entities. This means that they cannot be decomposed into other components and do not have any specific behavior. A way to address this challenge is using an ontology, which can scale down to a finer level of micromolecular modeling through domain and expert biologists' knowledge. While the DEVS module based exploits the Discrete-Event Modeling and Simulation (DEVS) formalism¹² to simulate the dynamic steps of the SARS-CoV-2 replication cycle, starting from its attachment to the host cell to the budding of new virions. Such macro-level simulation cannot be supported by the ontological module.¹³ By leveraging the benefits of both domains, the proposed approach allows understanding, modeling, and simulating, qualitatively and quantitatively, the replication of the SARS-CoV-2. We believe that understanding this complex system, through this proposed hybrid approach, could help biological researchers tackle this outbreak.

The paper is structured as follows. Section 2 introduces the theoretical foundations of this paper: the SARS-CoV-2 replication, the DEVS formalism, and the ontology notions. Section 3 gives a literature review on the existing approaches for modeling and simulating the SARS-CoV-2 replication. Section 4 presents the novel hybrid approach for modeling and simulation of the SARS-CoV-2 viral replication process in the micro- and macro-molecular scales (from genomic and transcriptomic to proteomic and metabolomic ones) by detailing its different modules and how they are connected together. We mainly focus on the mapping between the DEVS and ontological modules. Section 5 illustrates the experimental results by applying the approach to the real SARS-CoV-2 RNA to prove the efficiency of this hybrid approach. Section 6 concludes the paper and gives future perspectives.

2. Overview

2.1. Life cycle of SARS-CoV-2

As illustrated in Figure 1, the SARS-CoV-2 genome consists of a positive-sense single-stranded RNA, about 30 kilobases coated with N protein and covered by a lipid bilayer containing spike S , membrane M , and envelope E proteins.^{14,15} The SARS-CoV-2 life cycle includes four main stages: the virus entry, translation of the viral replicases, genome replication and transcription, structural and accessory proteins' expression, virion assembly and release:¹⁶

- *Virus entry*: the virus enters the target cell either by endocytosis or through viral/plasma membrane fusion. In both cases, this entry is handled by the attachment of the spike glycoprotein (*spike S*)

present on the surface of the virus envelope to the membrane of the host cell through its angiotensin-converting enzyme 2 (*ACE2*) receptors.¹⁷

- **Translation of viral replication machinery & replication:** once released in the host cell cytoplasm, the viral genome is translated by host ribosomes to produce two large replicase polyproteins (*pp1a* and *pp1ab*), which generate through the proteolytic process, sixteen non-structural proteins (*nsp1* to *nsp16*) that have a critical role in viral RNA synthesis, and structural proteins. These 16 non-structural proteins form the viral replication and transcription complex (*RTC*) and the viral RNA-dependent RNA polymerase (*RdRp*) *nsp12* in association with its two viral co-factors *nsp7* and *nsp8* as the catalytic core of RNA synthesis. The function of the *RTC* is to replicate a negative full-length genome and transcribe a set of negative-sense subgenomic (*sg*) RNAs. In turn, these will serve as templates to produce a new positive-sense RNA genome and nine subgenomic messenger ribonucleic acid mRNAs (*sgmRNAs*) encoding structural and accessory proteins.¹⁵
- **Production of the viral structural proteins and virus egress:** at late stages of infection, the structural proteins in spike (*S*), envelope (*E*), membrane (*M*), and nucleocapsid are translated from the *sg* mRNAs by the cellular ribosomes, in the endoplasmic reticulum (*ER*) and retained at the site of budding in the ERGIC. The viral RNA genome coated with N protein ultimately bud into ERGIC membranes, which are decorated with M, E, and S proteins, producing new virions.¹⁵
- **Egress of Virions:** Newly enveloped virions leave the cell by the exocytic pathway or by cell lysis at the ERGIC compartment, they exit the infected host cell by exocytosis through the lysosomal trafficking pathway, budding, or by cell lysis.¹⁵

Accordingly, the SARS-CoV-2 life cycle and its interaction with the human organism can be considered as a complex multicomponent hierarchical dynamical system composed of different scales, the micromolecular scale comprising the proteomic and metabolomic entities, while the macromolecular scale consisting of the genomic and transcriptomic entities. These entities can be transferred from one scale to another one, making their modeling and simulation very complex.

2.2. Modeling and simulation: DEVS formalism

The modeling and simulation theory (M&S) is used to mimic the operation of an existing or proposed system to understand and analyze its dynamic behavior in a simplified replica within a risk-free environment.¹⁸ This theory involves two separate activities. The first modeling activity

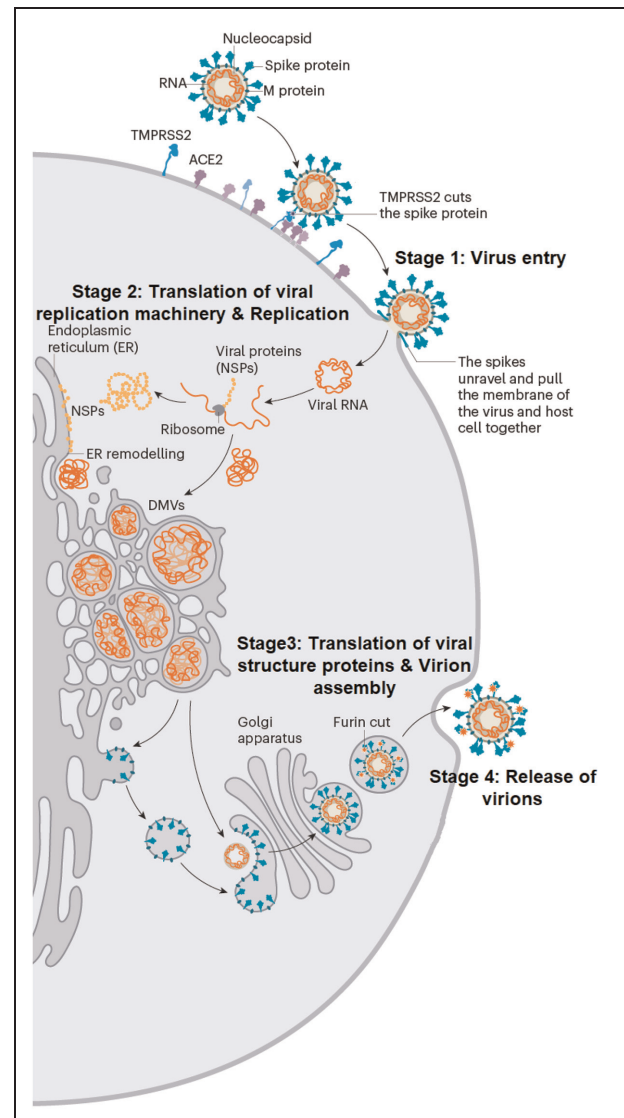


Figure 1. Life cycle of SARS-CoV-2. Inspired from.¹⁶

focuses on making a representation of a system from an observer's point of view. It answers all the questions the observer may have about the structure and function of the system. While the second simulation activity focuses on executing the model to produce its behavior by modifying its inputs and parameters.¹⁸ Various M&S formalisms have been developed, such as the Differential Equation System Specifications (DESS) for traditional differential equation systems or the Discrete Time System Specifications (DTSS) for automata. In our study, we choose to use the Discrete Event System Specification (DEVS) formalism that was initially proposed by Bernard P. Zeigler¹⁹ in the 70s as a discrete-event modeling specification formalism. This formalism supports the modeling and simulation of complex systems by defining them as hierarchical, modular models that can be easily reused. Accordingly, a system

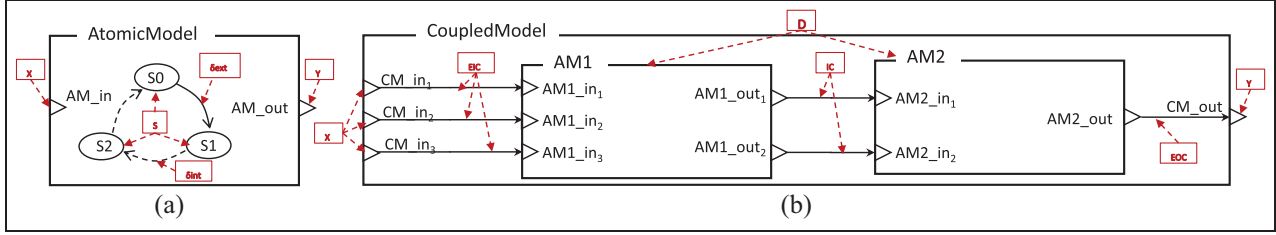


Figure 2. Graphical representation of (a) DEVS atomic model and (b) DEVS coupled model.

is defined as a composite of sub-models, each of them being behavioral (atomic) or structural (coupled), as seen in Figure 2. Coupled models can be embedded in a model hierarchy. Both models consist of a time base, inputs, states, outputs and functions to compute the next states and outputs.

As shown in Figure 2(a), a DEVS atomic model is defined by this seven-tuple equation (1):

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle \quad (1)$$

where X the input events set, Y the output events set, S the state set, δ_{int} the internal transition function ($S \rightarrow S$), δ_{ext} the external transition function ($Q \times X \rightarrow S$), with $Q = \{(s, e), s \in S \text{ and } e \in [0, ta(s)]\}$, λ the output function ($S \rightarrow Y$), and ta the elapsed time function ($S \rightarrow R_0^+ \cup \infty$).

A DEVS coupled model, as illustrated in Figure 2(b), may include several sub-models (atomic and/or coupled models) to define the hierarchical structure of the system. It is defined by equation (2):

$$CM = \langle X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, select \rangle \quad (2)$$

where: X the set of input events, Y the set of output events, D an index for the components of the coupled model, M_d a basic DEVS model $d \in D$ (atomic or coupled) defined by $M_d = \langle X_d, Y_d, S_d, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$. EIC the external input coupling, EOC external output coupling, IC the internal coupling, and $select$ the tie-break selector.

Various extensions of DEVS-based simulators have been proposed,²⁰ including the fwkDEVS²¹ a DEVS/GLE simulation framework designed in java and supporting GDEVs simulations. DEVSJAVA environment²² designed in java supports parallel execution on a uni-processor. This simulator can also be linked with DEVS/HLA and DEVS/CORBA frameworks. VLE (for Virtual Laboratory Environment) is a multi-modeling and simulation platform. It uses C++ to create simulation models and Python to use simulation as a scripting language for data analysis.²³ JDEVs²⁴ is implemented in java and enables discrete event, object-oriented, GIS-connected, and visual simulation model development. PythonPDEVs²⁵ is a Python-based simulation framework for classic and parallel DEVS. ADEVs²⁶ is also distinguished by its support for both discrete event and continuous dynamical systems.

CD++²⁷ is a general framework for modeling and simulation of classic, parallel and cellular DEVS models. Written in C++, the CD++ toolkit has been widely used to model and simulate a variety of applications.²⁸ Moreover, CD++ was revised and extended several times.^{29,30} A more complete list of DEVS-based simulators is available on the Advanced Real-Time Simulation Laboratory web page: <https://arslab.sce.carleton.ca/index.php/devs-tools/>. As can be seen, there is no better simulation tool, and they cannot be compared. Since each tool uses its own language and can be used for specific purposes. A possible way to compare them is by implementing an appropriate test model in the different simulation frameworks and then comparing the performance of each simulator, but it remains an expensive and time-consuming task. The proposed DEVS models in this study will be implemented in the CD++ simulator.²⁷

2.3. Ontologies and SWRL rules

Over the last few decades, there has been increasing use of ontological modeling in different domains, especially in health and biological data management. Ontologies are historically rooted in philosophy as the study of being, including their main categories and relations. In computer science, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject.³¹ *Concepts* represent explicit classes, used to describe the main entities of a domain of interest. They are usually organized in taxonomies through which inheritance mechanisms can be applied. *Properties* represent a type of association between those concepts. An ontology uses *axioms*, which are assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. It also contains *instances* that are used to represent elements or individuals, the basic objects of the ontology.

Formally, an ontology is defined by this quadruplet³² equation (3):

$$O = \langle C, P, Sub, Ass \rangle \quad (3)$$

where, C is the set of classes describing the concepts of the domain of interest, P is the set of properties describing the

individuals of the classes C , Sub is the subsumption relation function ($C \rightarrow 2^C$) with 2^C the power set of C , and Ass the classes-properties (object and data properties) association function ($C \rightarrow 2^P$).

All of these components are well-defined by the domain expert. The formalization of knowledge representation ontologies can be done using different formalisms or languages, such as first-order logic (FOL), eXtensible Markup Language (XML), Resource Description Framework (RDFS) or the World Wide Web Consortium (W3C) standard Web Ontology Language (OWL).³³ Thus, a formal ontology is computer-readable, allowing the computer to “understand” the relationships, the “formal semantics,” of the ontology. By incorporating expert rules, written as antecedent-consequent pairs, ontologies can generate logical inferences of relevant knowledge.³³ These knowledge rules can be constructed using the Semantic Web Rule Language (SWRL).³⁴ In this study, we use the *OntoRepliCov* ontology,¹¹ an ontology dedicated to the description of the coronavirus replication process, to provide semantic knowledge about the CoV genome organization, SARS-CoV-2 sequences and the different viral proteins produced.

3. Related work

Mathematical modeling and computational simulations play a key role in virology by studying viruses and their interactions with host cells.³⁵ To date, a great number of research efforts have been proposed to understand the SARS-CoV-2 and its COVID-19 disease. In this section, we categorize existing SARS-CoV-2 modeling and simulation approaches into two groups: mathematical approaches and data-driven approaches.

Several mathematical and statistical models have been developed to understand the transmission mechanisms, structures, and features of the SARS-CoV-2.^{36–39} These models were mainly designed for protein structure prediction, virus genomic sequence classification, drug discovery or SARS-CoV-2 transmission. Of these models, the following^{40–43} focused on the molecular dynamics simulation (MDS) for modeling the evolution of the viral protein RNA-dependent RNA polymerase (RdRp) and test its binding affinity to some drugs. Amin et al.⁴⁴ proposed a mathematical modeling based on Monte Carlo optimization for predicting possible SARS-CoV-2 papain-like protease (PLpro) inhibitors. Other mathematical models have been proposed for modeling the dynamics of transmission and spread of SARS-CoV-2. Most of them are based on compartmental models (such as Susceptible Infectious Recovered (SIR) models) using classical ordinary differential equations to aid epidemiologic monitoring.^{45–48} These SIR models are the simplest mathematical framework for describing the spread of diseases where immunity, if

acquired, cannot be lost. These models consider different parameters such as patient improvements, personal protection strategies, regulation implementation, and so on.^{49,50} They have proven their effectiveness in understanding human-to-human transmission and control techniques to stop transmission.

These mathematical models are divided according to the nature and predictability of the events that are modeled, into stochastic models or deterministic models.⁵¹ Deterministic models are based on the use of ordinary differential equations (ODE) and partial differential equations (PDE). While indirect stochastic differential equations (SDEs) deal with random events. Mathematical models are also differentiated into continuous and discrete models. Continuous models are suitable for studying the dynamics of biological phenomena such as the viral infection over time. Their differential equations are equations wherein solutions are not numerical values, they are functions. These equations express relations between these unknown functions and their subsequent derivatives. This makes it difficult to solve differential equation models. For example, in the SIR epidemic prediction model, the parameters of interest correspond to the parameters of susceptible, exposed, infected and recovered individuals. However, all of these settings are considered as a single function of time. Obviously, simulating the behavior of SARS-CoV-2 requires simulating several variables at the same time.^{52,53} Using mathematical models then refers to situations in which the independent variable used in the ODE model represents the most significant factor of our interest. However, we often ignore the possibility of other factors influencing our research.⁵² A high number of parameters is necessary for such mathematical models. Therefore, obtaining a value for each parameter via laboratory experiments or theoretical calculations is difficult, sometimes even impossible.⁵⁴

Other approaches based on machine learning (ML) and deep learning (DL) models have been also proposed to address SARS-CoV-2, its genomic sequence classification making an accurate diagnosis, or discover potential drugs.⁵⁵ Among these studies,^{56–59} proposed approaches using gradient boosting, neural networks, random forest, and support vector machine (SVM) to identify potential RdRp inhibitors. Authors trained their approaches on an RdRp inhibitor data set to perform inference analyses on antiviral drugs. These studies proved that ML models were able to identify new potential molecules (such as remdesivir and baloxavir marboxil) against the SARS-CoV-2 RdRp. Touati et al.⁶⁰ proposed an ML-based classification approach based on six supervised-learning classification models, i.e., linear and subspace discriminant, linear and quadratic SVM, fine and subspace KNN to analyze the virus genomic sequence for timely treatment plans. Authors in Touati et al.⁶⁰ combined different RNA representation and signal processing tools to identify the

SARS-CoV-2 genetic origin. These studies mainly use the Smoothed Discrete Fourier Transform (SDFT) algorithm for analyzing the SARS-CoV-2 genomic signature. Pavlova et al.⁶¹ proposed a machine-learning approach for understanding the dynamic interactions between the receptor-binding domain and human ACE2 receptor. Ghosh et al.⁶² applied a variety of ML algorithms to build several dozen of Quantitative Structure–Activity Relationship models (QSAR models), based on regression and classification algorithms, to find SARS-CoV-2 RdRp inhibitors. QSAR is usually one of the first steps in the drug discovery process, in which large databases of chemical structures are screened through a variety of predictive mathematical models in order to narrow down the number of potential drug candidates for the treatment of SARS-CoV-2. Tang et al.⁶³ proposed an advanced deep Q-learning network with the fragment-based drug design (ADQN-FBDD) for generating potential lead compounds targeting a key enzyme in the life-cycle of coronavirus, the 3C-like main protease (3CLpro). Alakus and Turkoglu⁶⁴ proposed various DL application models, including artificial neural networks (ANN), convolutional neural networks (CNN), long-short term memory (LSTM), recurrent neural networks (RNN), for the detection of SARS-CoV-2 infection.

These ML and DL approaches have shown acceptable results in terms of scalability and performance. However, they require a large amount of annotated data.⁶⁵ As well, data annotation is time-consuming, and requires the expertise of biologists. This leads scientists to train their models on unlabeled data sets, instead of annotated and important data.^{66–68} Furthermore, ML approaches require much data, however, there are not enough.⁶⁹

Besides, the majority of these studies focus on the relative risk of SARS-CoV-2 and its transmission such as investigating molecular dynamics simulations, reproducing the viral infection, measuring the effects of drug treatment, identifying potential drug candidates, spreading of the virus through people, and so forth. They ignore details of the virus life cycle. However, there is still much work to learn about the RNA genome replication and the resulting mutations, and the interaction of the SARS-CoV-2 with its host cell components at different temporal and spatial scales. Only a few studies investigate this issue, such as the work of Grebennikov et al.¹⁵ who propose a deterministic mathematical model for analyzing the SARS-CoV-2 life cycle. But, the paucity of available kinetic data on the intracellular life cycle of SARS-CoV-2 and the difficulty of solving the differential equations of the models lead to a misunderstanding of the viral life cycle. The complexity of the SARS-CoV-2 replication mechanism makes its modeling and simulation difficult.

As reviewed above, while there have been a number of mathematical and computational models and simulations proposed in the literature covering various dimensions to tackle the SARS-CoV-2, there are still constraints and

challenges such as understanding and simulating the replication mechanism of SARS-CoV-2 with the objective of identifying promising SARS-CoV-2 potent viral inhibitors.

4. The proposed hybrid approach

This section gives the core architecture of our proposed hybrid hierarchical approach for the modeling and simulation of the SARS-CoV-2. As a hybrid hierarchical modeling and simulation approach that combines DEVS formalism and semantic technologies, the proposed approach consists of two main modules: (1) the *DEVS module* and (2) the *ontological module*. Figure 3 gives an overview of our approach. In the following section, we describe these two modules and how they are combined.

4.1. The DEVS module

The first module of our proposed approach is based on the DEVS framework to mathematically model and simulate the SARS-CoV-2 replication process. This DEVS module determines the structures, behaviors, and dynamic evolution of the molecular entities involved in the SARS-CoV-2 replication process, and uses atomic and coupled models to model them. This module consists of two submodules (Figure 3), the *DEVS model* and *DEVS simulation engine*. As stated in Section 2.2, the proposed DEVS module is based on the CD++ modeling and simulation toolkit.²⁷ Based on the abstract simulator concepts proposed by, Zeigler et al.¹⁹ CD++ provides an environment to (1) design discrete events conceptual models and (2) simulate them. It consists of main classes *Models* and *Processors*. The first class defines the conceptual DEVS models. Using instance variables, the Model can: identify its associated processor, identify the link to the coupled model containing this model (parent), and specify the interaction of the model with other models (inport and output).⁷⁰ The atomic model is implemented with its own methods namely the internal transition, external transition, output, and time advancement methods. In our case, as the aim of our work is to combine DEVS formalism with ontological modeling, these methods are related to the SWRL rules engine. The aim is that these functions should interact with the inference module of the ontological module and request to execute the rule(s) that define the desired behavior (or process). A coupled model is defined by determining its components, and their coupling relationships. The second Processor class implements the simulation mechanism. We found different types of simulation processors, namely *Simulators*, *Coordinators*, and the *Root-coordinator*. The processors ensure the execution of the abstract simulation procedures through the implementation of the theoretical DEVS concepts. These processors are related to the models: a *simulator* is associated with an *atomic model*, and a *coordinator* with a *coupled model*. In this

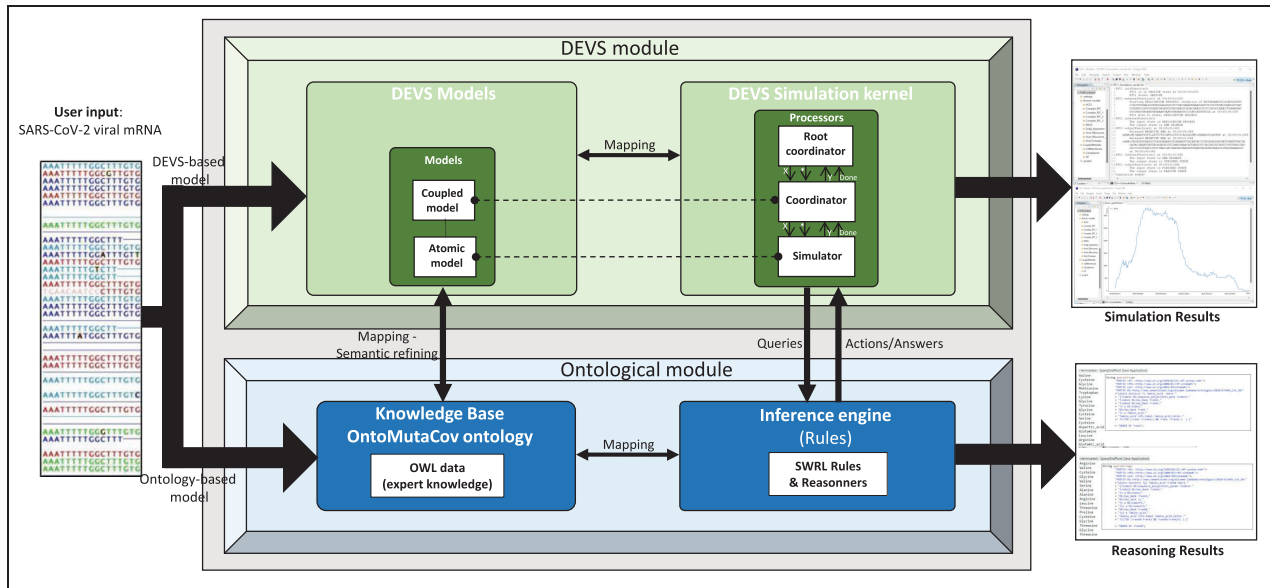


Figure 3. An overview of the proposed approach.

way, both simulators and coordinators ensure the functioning of the atomic and coupled models, respectively. The root processor manages the simulation. Indeed, it ensures the beginning and end of the simulation, the connection of the simulator with the environment in terms of passing external events/outputs from/to the environment, and the advancement of the simulation clock.⁷⁰ More details about these classes and DEVS algorithm simulation can be found in Wainer and colleagues.^{27,70}

The SARS-CoV-2 replication can be seen as a complex system involving a number of cellular and viral components at different scales. We, therefore, propose a hierarchical DEVS model of the SARS-CoV-2 replication process, consisting of a coupled model representing the host cell. This coupled model is itself formed by three coupled models, each one representing a level of the cell, respectively, the cell membrane, the cytoplasm, and the secretory pathways. An overview of the DEVS hierarchical representation of the SARS-CoV-2 replication machinery is displayed in Figure 4. Each of these coupled models is composed of a set of atomic models. The ten atomic models forming the entire SARS-CoV-2 replication system will be detailed in the next section (Section 5).

4.2. The ontological module

This module includes a rule-based system. It consists of an ontology, the OntoRepliCov ontology,¹¹ and a set of expert reasoning rules. The OntoRepliCov ontology has been specially designed and developed to describe the micro-level of the replication machinery. More

specifically, it handles the translation and the transcription sub-processes of the machinery where chains of proteins and codons (amino acids: a combination of three nucleotides) are produced. Modeling these sub-processes requires a description of the virus organization at the micro-level. Therefore, OntoRepliCov provides a hierarchical formalization of elements representing this micro-level organization, such as codons and nucleotides. This formal model is also enriched with rules to infer knowledge (nucleotides, codons, polyproteins) resulting from the different stages of the sub-processes that consider elements from the virus organization. The Inferences are based on symbolic artificial intelligence and developed for semantic web technologies. We exploit these inferences to enrich the DEVS simulation engine with a micro-level description.

As described in Table 1, the OntoRepliCov ontology consists of eighty-four classes. Biologically, these classes can be categorized into two distinct groups: those describing the hierarchical organization of the genome represented by the concept *Genome*, and those describing the replication elements of the virus represented by the concept *Replication_element*.

The first group, describing the viral genome, consists of seven subclasses: *Global_codon*, *Accessory_protein*, *Open_reading_frame*, *Polyprotein*, *Stem_loop*, *Non_structural_protein*, and *Structural_protein*, *Nucleobase*. The second group of classes, represented by the *Replication_element* concept, describes all elements needed for the replication process. This concept is further decomposed in *Subgenomic_RNA*, *Replication_Transcription_Complex*, *m_RNA*, *t_RNA*, *Subgenomic*

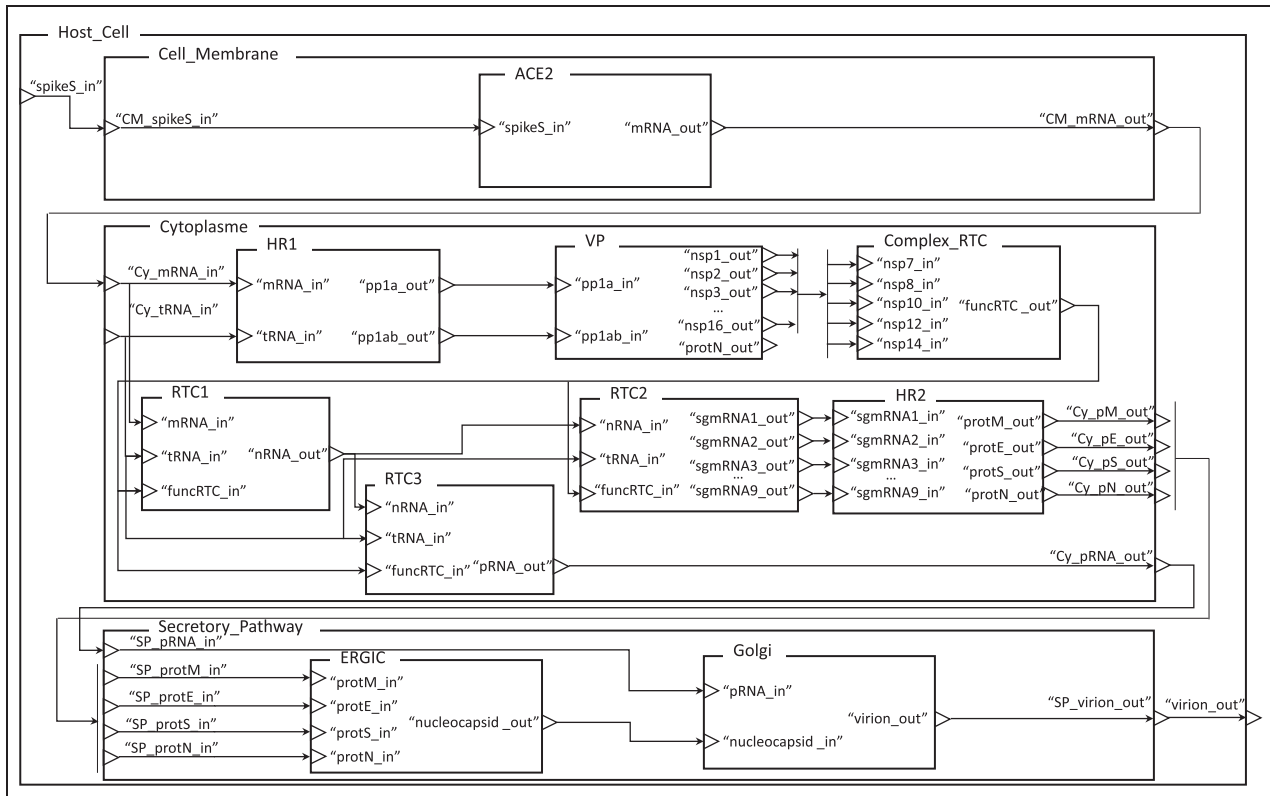


Figure 4. Overall DEVS simulation architecture of the SARS-CoV-2 replication process.

Positive _ RNA, *Subgenomic _ Negative _ RNA*, *Amino _ acid*, and so on. All of these classes and subclasses are shown in Table 1.

Moreover, the OntoRepliCov ontology defines the semantic relations among the previously defined classes through object and data properties. Each property links two concepts, and each data property links a concept with a specific datatype. Both are defined by a binary property. Among these 15 properties, the following can be mentioned:

- *has – first – base (Codon, Nucleobase)*: a property defining the first nucleobase of the codon sequence.
- *has – second – base (Codon, Nucleobase)*: a property defining the second nucleobase of the codon sequence.
- *has – third – base (Codon, Nucleobase)*: a property defining the third nucleobase of the codon sequence.
- *has – next (Nucleobase, Nucleobase)*: a property associating each nucleobase to its next nucleobase.
- *has – next (Codon, Codon)*: a property associating each codon to its next codon.
- *has – beginning – base (Stem-loop, Nucleobase)*: a property specifying the beginning nucleobase of the stem-loop.

- *has – ending – base (Stem-loop, Nucleobase)*: a property specifying the ending nucleobase of the stem-loop.
- *sequence – polyprotein – pp1ab(Codon, CodonFS)*: a property specifying the polyprotein pp1a.
- *sequence – polyprotein – pp1ab(Codon, CodonFS)*: a property specifying the polyprotein pp1ab.
- *has – rank (Nucleobase, xsd: integer)*: a data property assigning the rank of each nucleobase.
- *has – rank (Codon, xsd: integer)*: a data property assigning the rank of each codon.
- *frameshifting (Nucleobase, xsd: boolean)*: a data property indicating the nucleobase where the ribosome slips back in the translation process.
- *is – start(Codon, xsd: boolean)*: a data property specifying the starting codon of the translation process.

In addition to this knowledge base, the ontological module is semantically enhanced with an SWRL rule engine (Table 2) to describe the translation and the transcription sub-processes of the replication machinery. The inferences of these rules represent elements (amino-acid sequences, polyproteins, etc.) resulting from the sub-processes. Each

Table 1. A summary of classes in the OntoRepliCov ontology. The left column presents the major classes and their immediate subclasses. The right column presents their description.

OntoRepliCov classes	Description
Genome	It defines the hierarchical virus organization.
Structural _ protein	It defines the structural proteins of the virus (M, N, S, E).
S	It defines the transmembrane spike (S) glycoprotein.
M	It defines the membrane (M) protein.
E	It defines the envelope (E) protein.
N	It defines the nucleocapsid (N) protein.
Non _ structural _ protein	It defines the non-structural proteins of the virus.
nsp ₁	It describes the first non-structural protein.
nsp...	It describes the ... (between 1 and 16) non-structural protein.
nsp ₁₆	It describes the last (16) non-structural protein.
Accessory _ protein	It defines the s.
Open _ reading _ frame (ORF)	It defines the major six open reading frames that exist in SARS-CoV-2.
ORF1ab	It defines the role of the open reading frame ORF1ab.
ORF3	It defines the role of the open reading frame ORF3.
ORF6	It defines the role of the open reading frame ORF6.
ORF7a	It defines the role of the open reading frame ORF7a.
ORF8	It defines the role of the open reading frame ORF8.
ORF10	It defines the role of the open reading frame ORF10.
Polyprotein	It defines the sequence of amino acids delimited by start and stop codons.
pp1a	It defines the shorter polyprotein pp1a.
pp1ab	It defines the longer polyprotein pp1ab.
Nucleobase	It defines the units forming the genetic code of an Ribonucleic acid (RNA).
Adenine	It defines the adenine nucleic acid defined by the letter A.
Cytosine	It defines the cytosinenucleic acid defined by the letter C.
Guanine	It defines the guaninenucleic acid defined by the letter G.
Uracil	It defines the uracilnucleic acid defined by the letter U.
Global _ codon	It defines the sequence of three nucleobases.
Codon	It defines the codons formed before the frameshifting.
CodonFS	It defines the codons produced after the frameshifting.
Replication _ element	It defines all the elements for the replication process.
Replication _ Transcription _ Complex	It defines the molecule that drives viral genome replication and subgenomic mRNA synthesis.
Protease	It defines an enzyme that cleaves the polyprotein to release individual and active non-structural proteins.
Ribosome	It defines the cellular element able to translate from a message RNA into a protein which is composed of amino acids.
Amino _ acid	It defines the 22 amino acids which are part of the composition of proteins in mammals.
Alanine	It defines the 1st amino acid (Ala).
...	It defines the xth amino acid.
Tyrosine	It defines the 22th amino acid (Tyr).
Stem _ loop	It defines the unit of the structure of single-stranded RNA (stem, double helix, and loop).
m _ RNA	It defines the messenger ribonucleic acid.
t _ RNA	It defines the transfer ribonucleic acid.
Positive _ RNA _ Messenger	It defines the Single Strand Positive RNA produced during the viral replication process.
Negative _ RNA _ Messenger	It defines the Single Strand Negative RNA produced during the viral replication process.
Subgenomic _ RNA	It defines the subgenomic RNA produced during the viral replication process.

sub-process includes different stages. For instance, in the translation sub-process, chains of codons (a combination of three nucleotides) are formed to produce amino acids. The translation terminates when a combination of three codons describes the stop codon. Then, a polyprotein pp1a is synthesized from the beginning of the translation to the Stop codon. But sometimes, the translation slips back one nucleotide and a longer polyprotein pp1ab is produced. In

collaboration with biologists, a set of expert rules has been defined to compute the resulting elements from the sub-processes described with several stages (several rules): the structure of the viral proteins, ribosome activities, discontinuous transcription of subgenomic mRNAs, and continuous genomic RNA replication. Thus, this module contributes to defining and modeling the biological micro-level of the SARS-CoV-2 RNA genome replication

Table 2. The 12 SWRL rules involved in the SARS-CoV-2 replication process.

Rule number	SWRL rule
Rule 1	$Nucleobase(?x) \wedge Nucleobase(?y) \wedge Nucleobase(?z) \wedge hasNext(?x, ?y) \wedge hasNext(?y, ?z) \wedge hasRank(?z, ?t) \wedge swrlb:mod(0, ?t, 3) \rightarrow swrlb:divide(?d, ?t, 3) \rightarrow Codon(?c) \wedge hasRank(?c, ?d) \rightarrow hasFirstBase(?c, ?x) \wedge hasSecondBase(?c, ?y) \wedge hasThirdBase(?c, ?z)$
Rule 2	$Codon(?c1) \wedge hasThirdBase(?c1, ?x) \wedge Codon(?c2) \wedge hasFirstBase(?c2, ?y) \wedge hasNext(?x, ?y) \rightarrow hasNext(?c1, ?c2)$
Rule 3	$StemLoop(?s) \wedge hasBeginningBase(?s, ?b) \wedge hasRank(?b, ?r1) \wedge Codon(?x) \wedge Codon(?y) \wedge Codon(?z) \wedge hasNext(?x, ?y) \wedge hasNext(?y, ?z) \wedge hasThirdBase(?x, ?u1) \wedge Uracil(?u1) \wedge hasFirstBase(?y, ?u2) \wedge Uracil(?u2) \wedge hasSecondBase(?y, ?u3) \wedge Uracil(?u3) \wedge hasThirdBase(?y, ?a1) \wedge Adenine(?a1) \wedge hasFirstBase(?z, ?a2) \wedge Adenine(?a2) \wedge hasSecondBase(?z, ?a3) \wedge Adenine(?a3) \wedge hasThirdBase(?z, ?c) \wedge Cytosine(?c) \wedge hasRank(?u1, ?r2) \wedge swrlb:subtract(?fs, ?r1, ?r2) \wedge swrlb:lessThanOrEqual(?fs, 15) \rightarrow frameshifting(?u1, true)$
Rule 4	$Nucleobase(?u) \wedge frameshifting(?u, true) \wedge hasRank(?u, ?r0) \wedge Nucleobase(?x) \wedge Nucleobase(?y) \wedge Nucleobase(?z) \wedge hasNext(?x, ?y) \wedge hasNext(?y, ?z) \wedge hasRank(?x, ?r1) \wedge hasRank(?z, ?r3) \wedge swrlb:mod(0, ?r1, 3) \wedge swrlb:greaterThanOrEqual(?r1, ?r0) \wedge CodonFS(?c) \wedge hasRank(?c, ?r4) \wedge swrlb:add(?s, ?r3, 1) \wedge swrlb:divide(?d, ?s, 3) \wedge swrlb:equal(?d, ?r4) \rightarrow hasFirstBase(?c, ?x) \wedge hasSecondBase(?c, ?y) \wedge hasThirdBase(?c, ?z)$
Rule 5	$CodonFS(?c1) \wedge hasThirdBase(?c1, ?x) \wedge CodonFS(?c2) \wedge hasFirstBase(?c2, ?y) \wedge hasNext(?x, ?y) \rightarrow hasNext(?c1, ?c2)$
Rule 6	$Codon(?c1) \wedge hasThirdBase(?c1, ?u) \wedge CodonFS(?c2) \wedge hasFirstBase(?c2, ?u) \wedge frameshifting(?u, true) \wedge hasRank(?c1, ?r1) \wedge hasRank(?c2, ?r2) \wedge swrlb:add(?a2, ?r2, 2) \wedge Codon(?x) \wedge CodonFS(?y) \wedge hasRank(?x, ?a1) \wedge hasRank(?y, ?a2) \rightarrow hasNext(?x, ?y)$
Rule 7	$Codon(?x) \wedge isStart(?x, true) \wedge Codon(?y) \wedge Stop(?y) \rightarrow sequencePolyproteinPp1a(?x, ?y)$
Rule 8	$Codon(?x) \wedge isStart(?x, true) \wedge CodonFS(?y) \wedge Stop(?y) \rightarrow sequencePolyproteinPp1ab(?x, ?y)$
Rule 9	$Nucleobase(?n1) \wedge Adenine(?n1) \wedge hasRank(?n1, ?k) \wedge SubgenomicPositive(?sp) \wedge elementFrom(?n1, ?sp) \wedge Nucleobase(?n2) \wedge hasRank(?n2, ?k) \wedge SubgenomicNegative(?sn) \wedge elementFrom(?n2, ?sn) \rightarrow Uracil(?n2)$
Rule 10	$Nucleobase(?n1) \wedge Uracil(?n1) \wedge hasRank(?n1, ?k) \wedge SubgenomicPositive(?sp) \wedge elementFrom(?n1, ?sp) \wedge Nucleobase(?n2) \wedge hasRank(?n2, ?k) \wedge SubgenomicNegative(?sn) \wedge elementFrom(?n2, ?sn) \rightarrow Adenine(?n2)$
Rule 11	$Nucleobase(?n1) \wedge Guanine(?n1) \wedge hasRank(?n1, ?k) \wedge SubgenomicPositive(?sp) \wedge elementFrom(?n1, ?sp) \wedge Nucleobase(?n2) \wedge hasRank(?n2, ?k) \wedge SubgenomicNegative(?sn) \wedge elementFrom(?n2, ?sn) \rightarrow Cytosine(?n2)$
Rule 12	$Nucleobase(?n1) \wedge Cytosine(?n1) \wedge hasRank(?n1, ?k) \wedge SubgenomicPositive(?sp) \wedge elementFrom(?n1, ?sp) \wedge Nucleobase(?n2) \wedge hasRank(?n2, ?k) \wedge SubgenomicNegative(?sn) \wedge elementFrom(?n2, ?sn) \rightarrow Guanine(?n2)$

SWRL: Semantic Web Rule Language; SARS-CoV-2: severe acute respiratory syndrome-coronavirus 2.

process. For sake of space, the set of SWRL rules will be explained in Section 5.

4.3. The mapping between DEVS and ontological modules

In this section, we explain how the DEVS module is connected with the ontological model. The CD++ simulation process is performed through data transfer. Typically, these data include information about the sending component, the time of the related event, and information about the input/output port and a value. In CD++, there are four kinds of data (or data messages) to communicate (Figure 3): (1) the change of state caused by an internal event denoted by the symbol $*$, (2) the arrival of an external event by specifying its value and its port denoted by the symbol X , (3) the output of a model denoted by the symbol Y , and (4) the end of a model's task denoted by the symbol *Done*.⁷⁰

As the aim of our work is to combine DEVS formalism with ontological modeling, we exploit the internal and external transition functions and relate them to the SWRL rules engine. More specifically, the DEVS module requires

some inputs describing elements produced through the sub-processes of the replication machinery at micro-level. These elements are described through OntoRepliCov and represent the inferences resulting from SWRL rules combined with some axioms (concepts definitions); hence, each atomic model's simulator may send queries to the SWRL rule engine to get the elements required to fulfill a specific task in a macro-level of the replication machinery.

As shown in Figure 3, SARS-CoV-2 viral RNA is addressed simultaneously to both the atomic module ACE2, which is the biomolecular component by which the virus will penetrate the host cell as seen in section 2.1, and to the ontological model in which it will be modeled and semantically enriched with expert knowledge. Since the ACE2 atomic model can not identify to which the different groups of amino acids correspond, it appeals to the ontological model which, thanks to the SWRL rules provided by the biology experts, can identify the amino acids—decrypt them—and determine the different role of each amino acid in the viral RNA sequences. Simultaneously, the ontological modeling of the viral RNA will allow enriching it semantically and by means of some expert rules (presented

in Table 2), the ontology will decompose the viral RNA sequences and determine the function of each amino acid, a very detailed analysis (at a micromolecular level) that the DEVS simulation module cannot perform. Next, the ontological module provides the ACE2 atomic component with the identified groups of amino acids and their corresponding biological elements. At this stage, the ACE2 component will have enough knowledge to initiate the simulation at a macromolecular scale, a simulation at a scale that the ontology cannot ensure, and will provide a messenger RNA ready to be replicated in the cytoplasm of the host cell as an output. This is the first step in the life cycle of the virus replication process, corresponding to viral entry as described in section 2.1.

Since the simulation process works through the exchange of messages and data among the different processors (Figure 3). The root processor computes the time of the next external and internal state transition to check which one should occur first. Once the upcoming model is identified and the next event is triggered internally, the root processor sends a *-message, through the middle-level coordinators, to the selected simulator. Then, the atomic model simulator is launched by initializing its current time simulation t and the state s of the atomic model. The simulator requests the inference engine of the ontology module as detailed in Algorithm 4. This one executes the rule(s) that corresponds to the request. Then, SPARQL queries (semantic query language) are exploited to fetch the inferences resulting from the reasoning process. When the response is given on schedule, the simulator performs the internal state transition and repeats. Whether the next event is triggered by an external input, the root processor sends an external X -message to the simulator. This one requests the inference engine, through Algorithm 4, to execute the rule corresponding to the requested task. After completing the appropriate rule(s), the reasoner returns the result to the simulator. Then, the simulator handles the external state transition and repeats. Finally, the simulators return *done*-messages and Y -messages to the (root) coordinator to notify it that the model has completed its task and send it the model's output. Both of these *done*-messages and Y -messages are in turn converted to new *-messages and X -message, respectively. These steps are repeated until the end of the simulation.

Algorithm 1 describes the operation and function of the Root Coordinator (Root) that is used to drive a Simulator. The simulation process starts by initializing the root coordinator's simulator using the *Initialization()* function. As presented in Algorithm 2, this function initializes the initial states and simulation times of atomic models. Then, it executes its *ComputeStateTransition()* function until the final simulation time is reached. The *ComputeStateTransition()* function computes the next states of the atomic models during a simulation run, as detailed in Algorithm 3. The function starts by selecting the time of

Algorithm 1: Root's simulation process

```

Root Simulation Process
  Input: the total simulation time and the Simulation
  Compute the Initialization() function of the simulator
  while the final simulation time is not reached do
    Compute the ComputeStateTransition()
    function of the simulator
  End
End

```

Algorithm 2: Simulator's Initialization Function

```

Initialization (D)
  input: the list of atomic models  $M_d \in D$ 
  output: a priority queue of pairs  $(d,t)$   $t_i \leftarrow 0$ 
  for each atomic model  $d \in D$  do
     $s_d \leftarrow s_{0,d}$ 
     $t_{n,d} \leftarrow ta_d(s_d)$ 
     $t_{i,d} \leftarrow ta_i$ 
    if  $(t_{n,d} < \infty)$  then
      add  $(d,t_{n,d})$  to the scheduler (priority queue of
      pairs  $(d,t)$ )
    End
  End
  return scheduler
End

```

the first upcoming atomic model in the priority queue consisting of pairs of atomic models and their time (d, t) . This queue is denoted by the *scheduler*. Once selected, the atomic model is removed from the scheduler and added to the *imminentAMset* a set of models in which their next state will be immediately computed.

As described in Algorithm 3, the simulator exploits the *ComputeStateTransition()* function to compute the next states of the atomic models. It starts by checking if the model's inputs x_d are not empty and that the model M_d is scheduled for an internal transition, it computes the confluent transition function $\lambda_{conf,d}(x_d)$. If the model's inputs x_d are empty and that model M_d is scheduled for an internal transition, the simulator computes the internal transition function $\delta_{int,d}(s_d)$ by calling the *InferenceEngineProcess()* function. Otherwise, the elapsed time e_d is found and the model external transition function $\delta_{ext,d}(s_d, e_d, x_d)$ is computed using also the *InferenceEngineProcess()* function. The *ComputeStateTransition()* simulator's function finishes by computing the last transition time, and check if the next transition time $t_{n,d}$ is less than infinity, then the model reference M_d and its next event time $t_{n,d}$ are added to the *scheduler*.

As illustrated in Algorithm 4, the *InferenceEngineProcess()* function cycles through three main sequential steps. In this method, the inference engine starts by finding all SWRL rules (Table 2) that match the given task

Algorithm 3: Simulator's ComputeStateTransition Function

```

ComputeStateTransition ( $M_d$ )
  inputs: an atomic model  $M_d$ 
  output: the next states of the atomic model  $M_d$ 
   $t_n \leftarrow \min\{t_{n,d} \mid d \in A\}$ 
  while  $t_{n,d}(\text{scheduler}) = t_n$  do
    | add  $M_d$  to the imminentAMset
    | remove  $(d, t_{n,d})$  from the scheduler
  End
  foreach atomic model  $M_d \in \text{imminentAMset}$ 
  Do
    | if  $(X_d \neq \emptyset \text{ and } t_{n,d} = t_n)$  then
    | |  $s'_d \leftarrow \lambda_{\text{conf},d}(X_d)$ 
    | else if  $(X_d = \emptyset \text{ and } t_{n,d} = t_n)$  then
    | |  $s'_d \leftarrow \text{InferenceEngineProcess}(\delta_{\text{int},d}(s_d))$ 
    | Else
    | |  $e_d \leftarrow t_n - t_{i,d}$ 
    | |  $s'_d \leftarrow \text{InferenceEngineProcess}(\delta_{\text{ext},d}(s_d, e_d, X_d))$ 
    | End
    |  $t_{i,d} \leftarrow t_n$ 
    | if  $(t_{n,d} < \infty)$  then
    | | insert  $(d, t_{n,d})$  from the scheduler
    | End
  End
  return  $s'_d$ 
End

```

requested by the state transition function (internal or external). It looks for antecedents, and conditions on the left-hand side of the SWRL rule, that satisfy the given state transition function. This first step is ensured by the *Match-rules* function. Then, through the *Select-rules* function, the inference engine prioritizes the different SWRL rules that were matched to determine the order to execute them. Finally, in the third step, the engine executes each matched rule, using the *Execute-rules* function, in the order determined in step two and then iterates back to step one again. This cycle continues until no new rules are matched to the input values given by the state transition function. Finally, the SWRL rule-based inference engine sends the newly computed state of the atomic model to the DEVS simulator.

Therefore, through this simulation process as described by Algorithms 1–4, the two modules can be aligned, and we can benefit from the finer semantic rules provided by the SWRL rules as well as from the rich knowledge of the OntoRepliCov ontology. This mapping between both modules will be further detailed through examples in the next section.

5. Hierarchical model of the SARS-CoV-2 replication implementation

In Section 4, we introduced both core modules of our proposed hybrid hierarchical approach for the modeling and simulation of the SARS-CoV-2, the DEVS and ontological

Algorithm 4: Simulator's InferenceEngineProcess function

```

Inference Engine Processend ( $\delta_{\text{int-ext},d}$ )
  inputs: the given state transition state
  output: the new state transition value
  Compute the Match-rules() function that satisfy the given
  state transition function
  Compute the Select-rules() function to sort the
  corresponding SWRL rules
  Compute the Execute-rules() function to execute the
  rules in order
  return the new state transition value
End

```

modules. In this section, we first describe the development environment and programming tools for the implementation of our proposed approach. Then, we detail how this hierarchical approach has been implemented to model and simulate the SARS-CoV-2 replication.

5.1. Development environment

The different software and tools needed to implement the proposed hybrid approach are:

- *Eclipse*². It hosts the CD++ Builder GUI as a plugin.
- *Boost Software License*³ for knowledge base and ontologies management.
- *CD++Builder*⁴ for discrete event modeling and simulation based on the DEVS and Cell-DEVS formalisms.
- *Cygwin*⁵ to emulate a Unix system under Windows.
- *Protégé*⁶ for developing knowledge-based systems and editing ontologies.
- *owlcpp*⁷ for parsing, querying, and reasoning with OWL ontologies.
- *FaCT++*⁸ for reasoning on OWL DL-based ontologies.
- *Redland Raptor*⁹ to parse and execute Resource Description Framework (RDF) queries.

5.2. Implementation of the proposed hierarchical modeling of the SARS-CoV-2 replication

As described in the previous section (Section 4), to model the structure and behavior of the SARS-CoV-2 replication system, we have used primarily the classic DEVS formalism. Therefore, we have split up the replication process into four coupled models namely, the host cell which itself is divided into three compartments represented by three coupled models: the *cell membrane*, the *cytoplasm* and the *secretory pathways* (Figure 4). The SARS-CoV-2 replication process is ensured by these three coupled models.

Table 3. Correspondence of the cellular and viral components involved in the SARS-CoV-2 replication process with their matching components in the DEVS simulation framework.

Conceptual model elements	DEVS model elements
The infected host cell	<i>Host_Cell</i> DEVS coupled model
The cell membrane	<i>Cell_membrane</i> DEVS coupled model
The ACE2 receptor	<i>ACE2</i> DEVS atomic model
The cell cytoplasm	<i>Cytoplasm</i> DEVS coupled model
The first host ribosome	<i>HR1</i> DEVS atomic model
The viral protease	<i>VP</i> DEVS atomic model
The replication/transcription complex	<i>Complex_RTC</i> DEVS atomic model
The first functional complex RTC	<i>RTC1</i> DEVS atomic model
The second functional complex RTC	<i>RTC2</i> DEVS atomic model
The third functional complex RTC	<i>RTC3</i> DEVS atomic model
The second host ribosome	<i>HR2</i> DEVS atomic model
The secretory pathway	<i>Secretory_Pathway</i> DEVS coupled model
The ER-Golgi intermediate compartment	<i>ERGIC</i> DEVS atomic model
The GOLGI apparatus	<i>Golgi</i> DEVS atomic model
The transfer ribonucleic acid (tRNA)	Input ports
The genomic ribonucleic acid (mRNA)	Input/Output ports, Couplings between ports
The nonstructural proteins (nsp1, . . . , nsp16)	Input/Output ports, Couplings between ports
The subgenomic ribonucleic acid (sgmRNA) transcripts (sgmRNA1, . . . , sgmRNA9)	Input/Output ports, Couplings between ports
The positive-stranded subgenomic RNAs	Input/Output ports, Couplings between ports
The negative-stranded subgenomic RNAs	Input/Output ports, Couplings between ports
The structural proteins (M, S, E, and N)	Input/Output ports, Couplings between ports
The functional RTC	Input/Output Ports, Couplings between ports
The virus nucleocapsid	Input/Output ports
The virions	Output ports

SARS-CoV-2: severe acute respiratory syndrome-coronavirus 2; DEVS: discrete event system specification simulation; ACE: angiotensin-converting enzyme; RTC: replication/transcription complex; ERGIC: *ER-Golgi intermediate compartment*; GOLGI: *Golgi apparatus*.

Each coupled model is composed of a set of atomic models representing its main viral and/or molecular components. The *cell membrane* coupled model consists of only one component to ensure the virus entry step, the *angiotensin-converting enzymes 2* (ACE2) atomic component. The *cytoplasm* coupled model which ensures the translation and the replication/transcription steps of the SARS-CoV-2 RNA genome is composed by, the *first host ribosome* (HR1) atomic component, the *viral protease* (VP) atomic component, the four replication transcription complex (*Complex_RTC*, *RTC1*, *RTC2*, and *RTC3*) atomic components, the *second host ribosome* (HR2) atomic component. Finally, the *secretory pathways* coupled model ensuring virions assembly and their releases outside the host cell consists of the *ER-Golgi intermediate compartment* (ERGIC) atomic component, the *Golgi apparatus* (GOLGI) atomic component. Most of the atomic models rely on the inference engine rule (as presented in the previous section) to compute their state changes and behaviors. The inference engine exploits 12 rules, those described in Table 2. These rules are not only used to compute the atomic components' state changes but also to reach a lower, more microscopic scale, for considering the most basic building blocks and biomolecules (having no

changes of states or behavior), that the DEVS formalism cannot capture.

Table 3 shows the correspondence of the cellular and viral components involved in the SARS-CoV-2 RNA genome replication process, as described in Section 2.1 and Figure 1, with their corresponding components in the DEVS simulation framework. It is important to mention that in our case, we suppose that in our modeling there are four classes of replication/transcription complex (RTC). The first one, denoted by *Complex_RTC* is required to synthesize the genomic and subgenomic RNA transcripts. Consequently, we will give it three different names according to their function. The second class, denoted by *RTC1*, corresponds to the functional complex RTC that allows the synthesis of negative-strand templates from the positive mRNA genome. The third one, denoted by *RTC2*, corresponds to the functional complex RTC responsible for synthesizing subgenomic RNAs using Transcription Regulatory Sequences (TRSs). Then, finally, the fourth class, denoted by *RTC3* corresponds to the functional complex RTC that ensures the synthesis of positive-stranded subgenomic RNAs from the negative-strand templates.

In the virus entry step, the SARS-CoV-2 attaches to the host cell receptor ACE2 through its spike protein S to enter

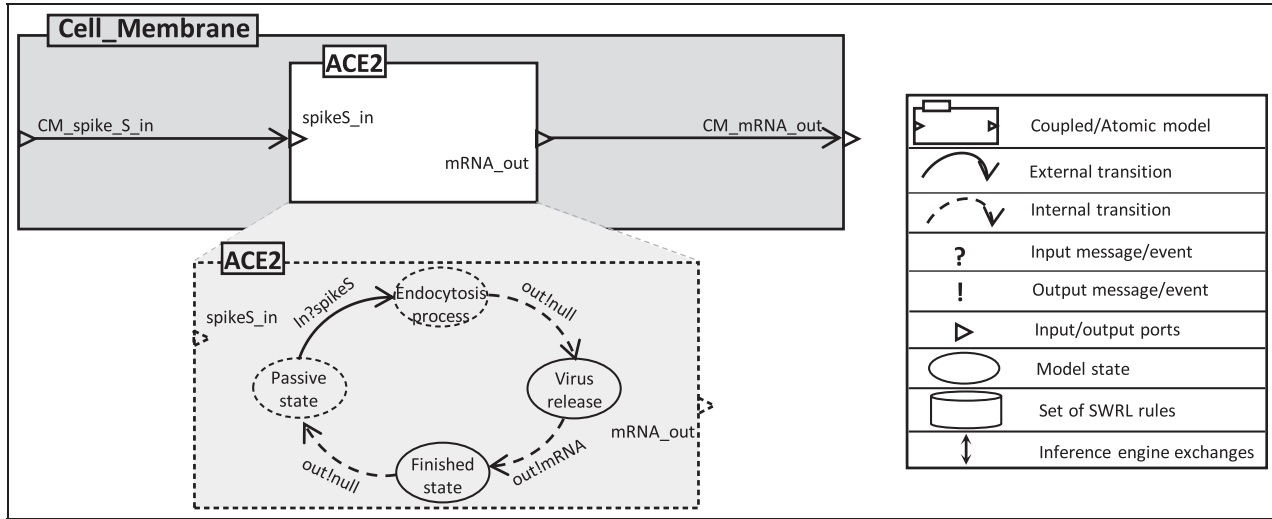


Figure 5. DEVS representation of the cell membrane compartment and its component using the CD++ tool.

Table 4. Formal specification of the cell membrane compartment, and its ACE2 component.

The cell membrane compartment

$$X_{CM} = \{CM_{spikeIn}\}$$

$$Y_{CM} = \{CM_{mRNAOut}\}$$

$$D_{CM} = \{ACE2\}$$

$$EIC_{CM} = \{((CM, "CM_{spikeIn}"), (ACE2, "spikeIn"))\}$$

$$EOC_{CM} = \{((ACE2, "mRNAOut"), (CM, "CM_{mRNAOut}"))\}$$

$$IC_{CM} = \{\emptyset\}$$

The ACE2 atomic component

$$ACE2 = (X_{ACE2}, Y_{ACE2}, S_{ACE2}, \delta_{ext_{ACE2}}, \delta_{int_{ACE2}}, \lambda_{ACE2}, ta_{ACE2})$$

$$S_{ACE2} = \{"Passive", "EndocytosisProcess", "VirusRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{ext_{ACE2}}(phase, \sigma, e, x) = (EndocytosisProcess, endocytosisTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"EndocytosisProcess", "VirusRelease", "Finished"\}$$

$$\delta_{int_{ACE2}}("EndocytosisProcess", endocytosisTime) = ("VirusRelease", virusReleaseTime)$$

$$\delta_{int_{ACE2}}("VirusRelease", virusReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{int_{ACE2}}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{ACE2}("EndocytosisProcess", sigma) = \emptyset$$

$$\lambda_{ACE2}("VirusRelease", \sigma) = (mRNAout, mRNA)$$

$$\lambda_{ACE2}("Finished", \sigma) = \emptyset$$

$$\lambda_{ACE2}("Passive", \sigma) = \emptyset$$

$$ta_{ACE2}("EndocytosisProcess", \sigma) = endocytosisTime$$

$$ta_{ACE2}("VirusRelease", \sigma) = virusReleaseTime$$

$$ta_{ACE2}("Finished", \sigma) = finishedTime = 0$$

$$ta_{ACE2}("Passive", \sigma) = \infty$$

ACE: angiotensin-converting enzyme.

the cell by the endocytic pathway. This first step occurs in the cell membrane compartment through the ACE2 atomic model, as presented by Figure 5. Once the ACE2 component receives the spike S (via its input port), it triggers the fusion of the viral and cellular membranes, allowing the penetration of the virus into the host cell cytoplasm as a genomic RNA (mRNA). Table 4 presents the formal specification of the cell membrane compartment, and its unique ACE2 component, according to Figure 5.

After virus entry, the viral RNA genome is translocated to the cytoplasm compartment in which two main stages of the replication process occur, the (1) translation of viral replication machinery and (2) translation of viral structure proteins (as described in Section 2.1 and Figure 1). Figure 6 illustrates a DEVS representation of the HR1 atomic component, one of the components involved in the cytoplasmic compartment. Since the representation is the same for the different components of the cytoplasm

Table 5. Formal specification of the cytoplasm compartment, and its seven components.

The cytoplasm compartment

$$X_{CY} = \{CYmRNAIn, CYtRNAIn\}$$

$$Y_{CY} = \{CYMOut, CYEOut, CYSOOut, CYNOOut, CYpRNAOut\}$$

$$D_{CY} = \{HR1, VP, RTC, RTC1, RTC2, HR2, RTC3\}$$

$$EIC_{CY} = \{(CY, "CYmRNAIn"), (HR1, "mRNAIn"), (CY, "CYtRNAIn"), (HR1, "tRNAIn"), (CY, "CYmRNAIn"), (RTC1, "mRNAIn"), (CY, "CYtRNAIn"), (RTC1, "tRNAIn"), (CY, "CYmRNAIn"), (RTC2, "mRNAIn"), (CY, "CYtRNAIn"), (RTC2, "tRNAIn"), (CY, "CYmRNAIn"), (RTC3, "mRNAIn"), (CY, "CYtRNAIn"), (RTC3, "tRNAIn")\}$$

$$IC_{CY} = \{(HR1, "pp1aOut"), (VP, "pp1aIn"), ((HR1, "pp1aOut"), (VP, "pp1aIn")), ((VP, "Nsp7Out"), (RTC, "Nsp7In")), ((VP, "Nsp8Out"), (RTC, "Nsp8In")), ((VP, "Nsp10Out"), (RTC, "Nsp10In")), ((VP, "Nsp12Out"), (RTC, "Nsp12In")), ((VP, "Nsp14Out"), (RTC, "Nsp14In")), ((RTC, "FrtcOut"), (RTC1, "FrtcIn")), ((RTC, "FrtcOut"), (RTC2, "FrtcIn")), ((RTC, "FrtcOut"), (RTC3, "FrtcIn")), ((RTC1, "nRNAOut"), (RTC2, "nRNAIn"), ((RTC1, "nRNAOut"), (RTC3, "nRNAIn")), ((RTC2, "sgm1Out"), (HR2, "sgm1In")), ((RTC2, "sgm2Out"), (HR2, "sgm2In")), ((RTC2, "sgm3Out"), (HR2, "sgm3In")), ((RTC2, "sgm4Out"), (HR2, "sgm4In")), ((RTC2, "sgm5Out"), (HR2, "sgm5In")), ((RTC2, "sgm6Out"), (HR2, "sgm6In")), ((RTC2, "sgm7Out"), (HR2, "sgm7In")), ((RTC2, "sgm8Out"), (HR2, "sgm8In")), ((RTC2, "sgm9Out"), (HR2, "sgm9In"))\}$$

$$EOC_{CY} = \{((HR2, "MOut"), (CY, "CYMOut")), ((HR2, "EOOut"), (CY, "CYEOut")), ((HR2, "SOOut"), (CY, "CYSOOut")), ((HR2, "NOOut"), (CY, "CYNOOut")), ((RTC3, "pRNAOut"), (CY, "CYpRNAOut"))\}$$

The HR1 atomic component

$$HR1 = (X_{HR1}, Y_{HR1}, S_{HR1}, \delta_{ext_{HR1}}, \delta_{int_{HR1}}, \lambda_{HR1}, ta_{HR1})$$

$$X_{HR1} = \{(HR1In1, (+)RNA), (HR1In2, tRNA), \}$$

$$Y_{HR1} = \{(HR1Out1, pp1a), (HR1Out2, pp1ab)\}$$

$$S_{HR1} = \{"Passive", "TranslationProcess", "ProteinRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{ext_{HR1}}(phase, \sigma, e, x) = (TranslationProcess, translationTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"TranslationProcess", "ProteinRelease", "Finished"\}$$

$$\delta_{int_{HR1}}("TranslationProcess", translationTime) = ("ProteinRelease", virusReleaseTime)$$

$$\delta_{int_{HR1}}("ProteinRelease", proteinReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{int_{HR1}}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{HR1}("TranslationProcess", \sigma) = \emptyset$$

$$\lambda_{HR1}("ProteinRelease", \sigma) = (HR1Out1, pp1a), (HR1Out2, pp1ab)$$

$$\lambda_{HR1}("Finished", \sigma) = \emptyset$$

$$\lambda_{HR1}("Passive", \sigma) = \emptyset$$

$$ta_{HR1}("TranslationProcess", \sigma) = translationTime$$

$$ta_{HR1}("ProteinRelease", \sigma) = proteinReleaseTime$$

$$ta_{HR1}("Finished", \sigma) = finishedTime = 0$$

$$ta_{HR1}("Passive", \sigma) = \infty$$

compartment and for the sake of space, only one of them (the HR1 atomic component) is presented. See Appendix 1 for the entire DEVS representation of cytoplasm compartment (with its seven components), including its formal specification in Table 5 and Appendix Tables 8 and 9. These tables present the formal specification of the seven cytoplasm compartments, according to the Appendix Figure 13. During this phase, the host cell translation machinery is hijacked and dedicated to viral component synthesis. Thus, the genomic mRNA is translated by the host ribosomes (denoted here by HR1) to produce two polypeptides *pp1a* and *pp1ab* from *ORF1a* and *ORF1a*, which are then further processed by proteolytic cleavages (Figure 6). As this process is complex and consists of four steps namely the *initiation*, *elongation*, *termination*, and *recycling*, the host ribosome component uses some specific SWRL rules. Indeed, the DEVS formalism is able to manage the different states, inputs, and outputs of the host

ribosome component, but is unable to capture how these four steps are done. Therefore, the state transition function of the host ribosome component uses the adequate SRWL rules set. This process of rules selection is managed by the Algorithm 4. At this stage of the replication (the translation process through the host ribosome component), and as illustrated in Figure 7, the Algorithm 4 is set up by parameters that specify the *mRNA_in*. Since we are in the translation process the algorithm computes the *Match-rules()* function and selects (*Select-rules()*) those that satisfy this stage of transition. As a result of these two functions, eight rules are selected 1, 2, 3, 4, 5, 6, 7, and 8 (Table 2) to compute the outputs in the right order. Once the rules are selected, the *Execute-rules()* function is considered to execute the rules from 1 to 8 and get the output of the different translation sub-stages. First, rules 1 and 2 are used to ensure the division and definition of the genomic codons. Then, rules 3, 4, and 5 are used for the

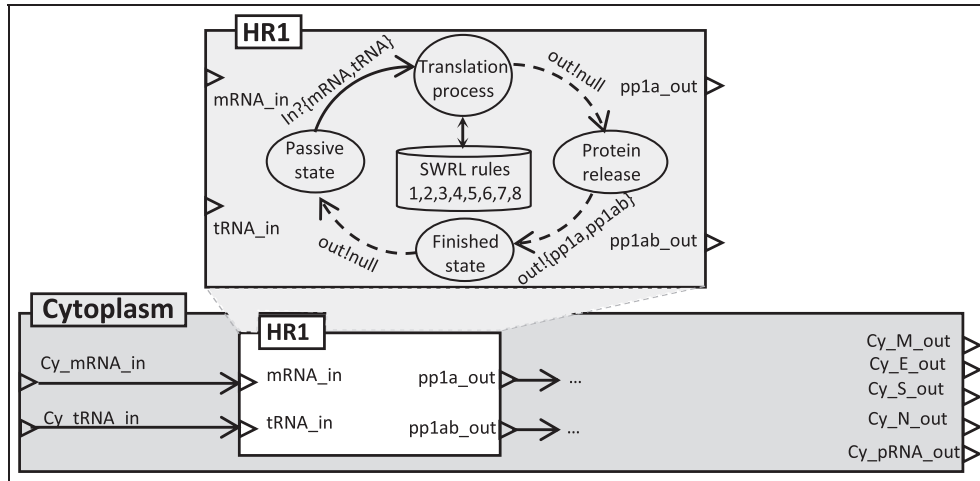


Figure 6. DEVS representation of the host ribosome I atomic component inside the cytoplasm compartment, using the CD++ tool.

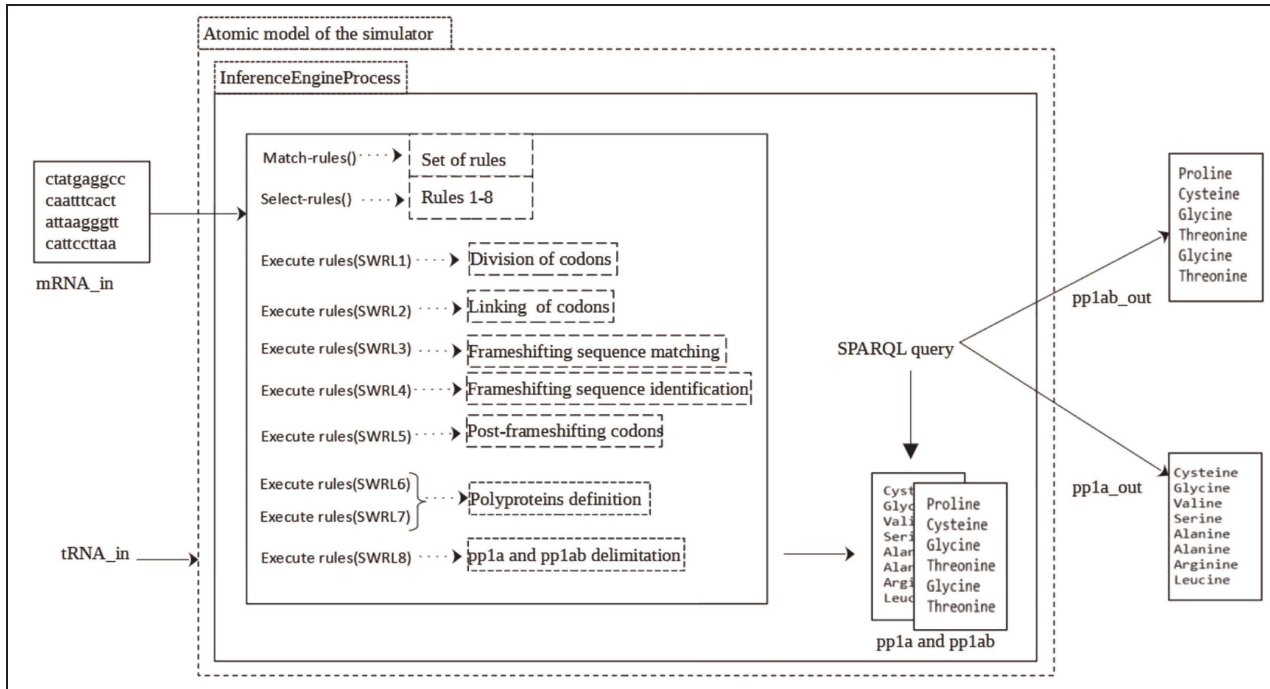


Figure 7. An example of the Algorithm 4 execution for the translation process.

ribosomal frameshifting. As the ribosome may slip back to translate the ORF1b, it uses these three SWRL rules to identify the pattern responsible for the ribosomal frameshifting and at which nucleotide rank this frameshifting occurs. In 30% of cases, the ribosome by-pass the stop codon of the *ORF1a* to translate the *ORF1b*. Thus, the polyprotein 1a (*pp1a*) is extended to produce the *pp1ab*. This process called post-frameshifting codons is ensured by the SWRL rule number 6. Finally, the SWRL rules 7 and 8 delimit the polyproteins *pp1a* and *pp1ab*. These

rules ensure the elongation step in which the encoded polyproteins *pp1a* and *pp1ab* are assembled until the termination occurs when the elongating ribosome meets the stop codon (Table 2). These polyproteins are the results of several incremental inferences and are considered outputs for the Algorithm 4 and H1 and inputs for VP. Therefore, a SPARQL query is required to get the sequence of these two polyproteins and the answer to this query is returned through the Algorithm 4. An example of a SPARQL query is presented in Figure 8.


```

<terminated> SparqlEndPoint [Java Application]
Arginine
Valine
Cysteine
Glycine
Valine
Serine
Alanine
Alanine
Arginine
Leucine
Threonine
Proline
Cysteine
Glycine
Threonine
Glycine
Threonine

String queryString=
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"+
"PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"+
"PREFIX NS: <http://www.semanticweb.org/wissame.laddada/ontologies/2020/9/SARS_CoV_2#>"
+"select distinct ?y1 ?amino_acid ?rank0 where "
+ "{?codon1 NS:sequence_polyprotein_pp1ab ?codon2."
+ "?codon2 NS:has_Rank ?rank2."
+ "?x a NS:Codon;"
+ "NS:has_Rank ?rank3;"
+ "NS:has_next ?y."
+ "?y a NS:CodonFS."
+ "?y1 a NS:CodonFS;"
+ "NS:has_Rank ?rank0."
+ "?y1 a ?amino_acid."
+ "?amino_acid rdfs:label ?amino_acid_letter."
+ "FILTER (?rank0>?rank3 && ?rank0<?rank2+1 ).}"

+ "ORDER BY ?rank0";

```

Figure 8. A SPARQL Query to extract the sequence of the polyprotein pp1ab from OntoRepliCoV through the Algorithm 4.

Both polyproteins *pp1a* and *pp1ab* are cleaved by the viral protein component (VP), through the proteolytic cleavages by two viral proteases, to provide non-structural proteins. The first polyprotein is cleaved into non-structural proteins *nsp1* to *nsp12*, while the second one is cleaved into additional non-structural proteins, *nsp12* to *nsp16*. These nsp proteins can be considered as the first viral proteins to be expressed after virus entry. In this process, the translation of the second polyprotein requires a -1 frameshifting event. As the DEVS formalism cannot ensure this process, the VP atomic component also calls some specific SWRL rules, namely rules 1, 2, 3, 4, and 5 in Table 2, and axioms in charge of the amino acids' inference from codons. The SWRL rules 1 and 2 are used for the division and definition of the codons, and rules 3 and 4 are for ensuring the frameshifting during the cleavage process.

Among the non-structural proteins, the *nsp7*, *nsp8*, *nsp10*, *nsp12*, and *nsp14* are joined together to form the minimal catalytic core of the proofreading RNA polymerase complex. They are modeled in our study by the RTC atomic component. This combination produces a functional RTC that will be used to synthesize the viral genomic and subgenomic RNA transcripts. This functional RTC has a major role in the viral replication process, as it ensures the replication of the RNA genome and the transcription of a set of subgenomic mRNAs. As stated at the beginning of this section, we distinguish in our DEVS modeling three kinds of RTC aside from the functional RTC (the creation of RTC). RTC 1, a functional RTC, that ensures the synthesis of negative-strand templates $-RNA$ from the genomic RNA. RTC 2, a functional RTC, ensures the synthesis of subgenomic RNAs using Transcription Regulatory Sequences (TRSs). And, RTC3, a functional RTC, ensures the synthesis of positive-stranded subgenomic $+RNA$ from the negative-strand templates $-RNA$. All these functional RTC components (RTC1, RTC2, and RTC3) use also some specific SWRL rules (9, 10, 11, and

12) and the axioms of amino acids to ensure the translation (in both directions) and synthesis of subgenomic RNAs. The rules (9, 10, 11, and 12) are used to ensure the transcription process. These rules are used to produce the complementary copy of the viral mRNA (either positive-strand RNA or negative-strand RNA).

The genomic and subgenomic RNAs are translated by the host ribosome HR2 into structural proteins namely the Spike (S), the Envelope (E), the Membrane (M), and the Nucleocapsid (N). The HR2 atomic component relies on the specific SWRL rules designed for the translation process, 1, 2, 9, 10, 11, and 12. Rules 1 and 2 are used for the division and definition of codons as ribosomes translate a codon (formed by three nucleobases) into an amino acid, constituting the proteins. While the rest of the rules (9, 10, 11, and 12) are used to ensure the transcription process and produce the fourth structural proteins as described in Figure 9(b).

At late stages of infection, new virus particles are formed and released into the extracellular environment (as described in Section 2.1 and Figure 1). This step is ensured by the secretory pathway compartment through both atomic components ERGIC and Golgi, as presented by Figure 10. The ERGIC atomic model ensures the virus assembly process by joining the four structural proteins (M, E, S, and N). This process is totally ensured by the DEVS formalism without the need for SWRL rules. While the Golgi atomic component ensures the release process of the assembled virions outside the cell via a process called exocytosis. When the nucleocapsid is created, the Golgi takes as input the viral nucleocapsid and some regions of the genomic $+RNA$, and then releases a mature virion outside the cell. This process is completely ensured by the DEVS formalism without the need for SWRL rules. Once outside the cell, these released virions become extracellular virions and are able to infect other healthy cells. The formal specification of the secretory pathway compartment

Table 6. Formal specification of the secretory pathway compartment, and its two components.

The secretory pathway compartment

$$X_{SP} = \{SPpRNAIn, SPMIn, SPEIn, SPSIn, SPNIn\}$$

$$Y_{SP} = \{SPVirionOut\}$$

$$D_{SP} = \{ERGIC, Golgi\}$$

$$EIC_{SP} = \{((SP, "SPpRNAIn"), (Golgi, "pRNAIn")), ((SP, "SPMIn"), (ERGIC, "MIn")), ((SP, "SPEIn"), (ERGIC, "EIn")), ((SP, "SPSIn"), (ERGIC, "SIn")), ((SP, "SPNIn"), (ERGIC, "NIn"))\}$$

$$IC_{SP} = \{((ERGIC, "nucleocapsidOut"), (Golgi, "nucleocapsidIn"))\}$$

$$EOC_{SP} = \{(Golgi, "virionOut"), (SP, "SPVirionOut")\}$$

The ERGIC atomic component

$$ER = (X_{ER}, Y_{ER}, S_{ER}, \delta_{extER}, \delta_{intER}, \lambda_{ER}, ta_{ER})$$

$$X_{ER} = \{(ERIn1, MIn), (ERIn2, EIn), (ERIn3, SIn), (ERIn4, NIn)\}$$

$$Y_{ER} = \{(EROut, Nucleocapsid)\}$$

$$S_{ER} = \{"Passive", "AssemblyProcess", "NucleocapsidRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{extER}(phase, \sigma, e, x) = (AssemblyProcess, assemblyTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"AssemblyProcess", "NucleocapsidRelease", "Finished"\}$$

$$\delta_{intER}("AssemblyProcess", assemblyTime) = ("NucleocapsidRelease", capsidReleaseTime)$$

$$\delta_{intER}("NucleocapsidRelease", capsidReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{intER}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{ER}("AssemblyProcess", \sigma) = \emptyset$$

$$\lambda_{ER}("NucleocapsidRelease", \sigma) = (EROut, Nucleocapsid)$$

$$\lambda_{ER}("Finished", \sigma) = \emptyset$$

$$\lambda_{ER}("Passive", \sigma) = \emptyset$$

$$ta_{ER}("AssemblyProcess", \sigma) = assemblyTime$$

$$ta_{ER}("NucleocapsidRelease", \sigma) = capsidReleaseTime$$

$$ta_{ER}("Finished", \sigma) = finishedTime = 0$$

$$ta_{ER}("Waiting", \sigma) = \infty$$

The GOLGI atomic component

$$GO = (X_{GO}, Y_{GO}, S_{GO}, \delta_{extGO}, \delta_{intGO}, \lambda_{GO}, ta_{GO})$$

$$X_{GO} = \{(GOIn1, Nucleocapsid), (GOIn2, pRNA)\}$$

$$Y_{GO} = \{(GOOut, Virion)\}$$

$$S_{GO} = \{"Passive", "ExocytosisProcess", "VirionRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{extGO}(phase, \sigma, e, x) = (ExocytosisProcess, exocytosisTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"ExocytosisProcess", "VirionRelease", "Finished"\}$$

$$\delta_{intGO}("ExocytosisProcess", exocytosisTime) = ("VirionRelease", virionReleaseTime)$$

$$\delta_{intGO}("VirionRelease", virionReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{intGO}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{GO}("ExocytosisProcess", \sigma) = \emptyset$$

$$\lambda_{GO}("VirionRelease", \sigma) = (GOOut, Virion)$$

$$\lambda_{GO}("Finished", \sigma) = \emptyset$$

$$\lambda_{GO}("Passive", \sigma) = \emptyset$$

$$ta_{GO}("ExocytosisProcess", \sigma) = exocytosisTime$$

$$ta_{GO}("VirionRelease", \sigma) = virionReleaseTime$$

$$ta_{GO}("Finished", \sigma) = finishedTime = 0$$

$$ta_{GO}("Waiting", \sigma) = \infty$$

ERGIC: ER-Golgi intermediate compartment; GOLGI: Golgi apparatus.

genome, at $t = 30$, its external transition function changes its state in the Translation process. After some time (at $t = 40$), the internal state function changes the state of the host ribosome from the translation process to proteins release as the HR1 completed the translation process. During this state, the output function transfers the products (here 70% of *pp1a* and 30% of *pp1ab* polyproteins) to the next atomic model. After the release state, the internal transition function changes the state of the ribosome to finished (at $t = 45$), then to the passive state (at $t = 46$).

Figure 9(b) presents the dynamic state changes and the output of the RTC1 atomic component responsible for the

transcription process. It shows the synthesis of negative-strand templates from a positive mRNA genome, a process called transcription. At $t = 0$, the complex RTC1 atomic model is created and initialized in the passive state. Once it receives the positive viral genome (at $t = 1:20$), its external transition function changes its state in the replication process. After some time (at $t = 1:60$), the internal state function changes the state of the RTC1 from the replication process to the RNA release as the RTC1 completed the transcription process. During this state, the output function transfers the negative RNA produced to the next atomic models. After the release state, the internal transition function changes the state of RTC1 to finished (at $t = 1:65$),

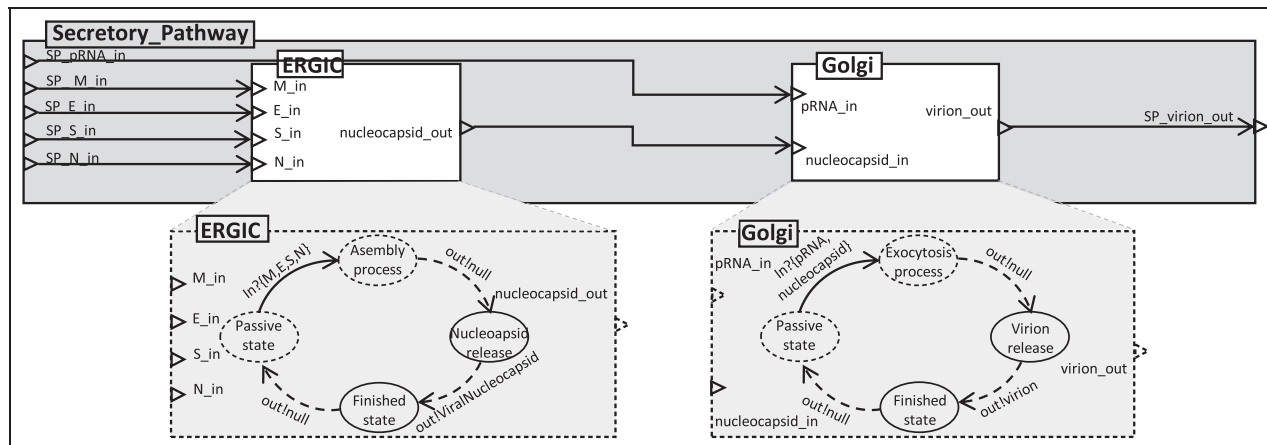


Figure 10. DEVS representation of the secretory pathway compartment and its components using the CD++ tool.

Table 7. Simulation parameters.

	Parameter	Value	Units	Reference
Transcription and Translation	Number of RdRp complex per negative sgmRNA	2	molecules	Kim and Yin ⁷²
	Basal transcription/replication elongation rate	2	kb/min	Kim and Yin ⁷²
	Ribosome density	0.0125	ribosome/aa	Fehr and Perlman ⁷³
	Translation rate	120	aa/min	Fehr and Perlman ⁷³
	Programmed Ribosome frameshift (– 1)	25	%	Fehr and Perlman ⁷³
Assembly and packaging	Packaging ratio for protein E	43.4	–	Lauer et al. ⁷⁴
	Packaging ratio for protein S	1.5	–	Lauer et al. ⁷⁴
	Packaging ratio for protein N	2.1	–	Lauer et al. ⁷⁴
	Packaging ratio for protein M	1.0	–	Lauer et al. ⁷⁴
	Packaging ratio for positive viral RNA	1.4	–	Lauer et al. ⁷⁴

then to the passive state (at $t = 1:66$). Unfortunately, we were unable to compare our results with other approaches, as at present there is no comparable work available with precise data. Furthermore, the purpose of this work was to provide biologists with a tool that would address their need to understand the different phases of SARS-CoV-2 viral replication and the interactions of viral entities with cellular entities at different biological scales. These experiments were carried out with uncertain parameters. At present, biologists face difficulties in obtaining the right values for the parameters of the simulation. Therefore, it must be noted that the simulation parameters used in this simulation are not realistic and are not approved by the biologists. Currently, no study has been done to provide the exact parameters to conduct a precise simulation. The SARS-CoV-2 simulation starts with a single initial virus particle. The reference genome was downloaded from the ^{NCBI}. As the exact parameters of the SARS-CoV-2 replication machinery are still unknown, we conducted a literature search to collect settings that may meet our needs. The collected parameters used in the simulation are presented in Table 7. The translation step is started with a translation

rate of 120 amino acids per minute (aa/min), which depends on the ribosome density per cell set at 0.0125 ribosome per amino acids (ribosome/aa). When a polypeptide is translated, ribosome frameshift is randomly sampled with a probability of 25 %. Once these polypeptides are synthesized, non-structural proteins are randomly produced. According to the literature, we assumed that the RdRp has a constant transcription/replication rate equal to two kb/min without any transcription activation factor. Two RdRp complex are assigned to each negative subgenomic mRNA for transcription. In the absence of precise biological parameters and with the agreement of expert biologists, we assumed that the number of subgenomic mRNAs is equal to 9 (sgmRNA1 to sgmRNA9). In the absence of precise biological parameters and with the agreement of expert biologists, we assume that the packaging ratio for the proteins E, S, N, and M are equals to 20, 1.5, 2.1, and 1.0, respectively. While the packaging ratio for positive viral RNA was fixed to 1.4. The packaging ratios have no units.

This simulation experiment was performed only to illustrate the viral quantification process while taking into



Figure 11. A CD++ screenshot showing the quantitative simulation results of the proposed approach and the quantification of the secreted virions.

account the different replication steps presented by the proposed approach. Figure 11 (a CD++ screenshot) presents a first proof of concept of the viral quantification of an epithelium host-cells with a single SARS-CoV-2 virion, focusing into the different stages of the replication cycle presented in this work. The figure presents the number of SARS-CoV-2 virions leaving host-cell during an infection period of 13.5 days (around 1.160.014 s) over its viral life cycle. As observed, this graphical description reflects the number of virions according to the different phases of the virus life cycle. During the initial stage, a single virus causes the infection by binding and penetrating the cells. Then, we observe in the viral growth curve virions are released from the infected host cells at the same time, creating an exponential growth of produced virions until reaching a peak. It is at this stage that the transcription and translation processes reach their maximum, and therefore triggering a significant assembly and packaging of new virions. Such a process is called a “viral burst.” Finally, when no viable host cells remain, the virus particles begin to slowly decrease until they totally disappear. According to expert biologists, this viral growth curve approximates the real behavior of SARS-CoV-2 infection. It is important to note that our simulation was not conducted on real data. It was only designed to make a viral quantification considering the different stages of the viral replication cycle as detailed in the previous sections. From a biological point of view, the viral evolutionary appearance and structure, as well as its approximate quantification, seems logical—it clearly shows the phases of the SARS-CoV-2 viral replication. Several studies have recently started to investigate these fundamental data of

SARS-CoV-2 infection, on which numerous uncertainties remain. This underlines the importance of our proposed hybrid approach which, despite the lack of certain parameter values, approximates the viral replication cycle. Through the experiment, we clearly observe that both main processes of the viral replication, the *translation* and the *transcription*, are handled by the DEVS simulator through the ontological module through SWRL rules and axioms. For sake of space, we only display the results of both the *translation* and the *transcription* processes.

We conclude that the CD++ simulator is able to provide quantitative information on the different viral molecular components produced during the infectious viral cycle. In other words, we obtained both a qualitative simulation outputting the viral components that are produced during the different stages of the viral cycle (polyprotein cleavages, non-structural proteins production, RNA replication, transcription of the different sg mRNAs, assembly of the structural proteins) and a qualitative simulation outputting the expected number of virions produced as a result of the different stages of the viral cycle. Combining qualitative and quantitative information for simulation-based analysis gives several advantages, which will be discussed in the next section. However, it is important to note that this simulation does not take into account the defenses (innate immunity) that the cell can put in place to fight against the infection.

It is important to notice that, in this work, we are only trying to understand the viral replication and how viral entities interact with the host cell components during the replication. This is why all DEVS atomic components depend on each other and the input values are ignored in the external

transitions. Only the atomic component ACE2 considers the input value, as it is activated exclusively by an external transition, the presence of viral RNA. Such input values will be considered in future work, especially when biologists investigate the impact of drugs on viral replication performance, a question not addressed in this work.

5.3. Results and discussion

The results of the implementation, presented in the previous section, illustrate how the proposed hybrid approach is enhancing the understanding of the SARS-CoV-2 cell infection process at the single-cell level. The hybrid approach provided a thorough analysis and understanding of viral replication biological processes occurring in the infected cell and how they influence each other. It allows tracking the evolution and replication of the SARS-CoV-2 through hierarchical cellular compartments and quantifying its release at the individual cell level. The hierarchical approach can be considered as a suitable framework to model heterogeneous populations and simulate the behavior of complex systems with an intrinsic discrete nature, such as the virus replication process and virus–host interaction.

We have demonstrated that, through its modular structure, the DEVS formalism is suitable to be integrated with an ontological-based model within a hierarchical framework that can link the small biomolecules to the cell levels. The approach effectively embeds the viral replication behavior from the simulation of the (1) dynamics of the individual atomic models (e.g. ribosomal translation of the different viral proteins and RNA replication), (2) interaction among atomic components (e.g. the interaction between viral proteins and cellular ribosomes), and (3) micro and macroscopic scales (e.g. interaction among the different cell's compartments). These simulation results will provide valuable support for new cutting-edge biological approaches to accurately analyze transcriptome dynamics at the single-cell level (Single-Cell RNA-Seq) as well as to quantify viruses released at the single-cell resolution (viro-fluidic method).

With the hybrid nature of the proposed modeling and simulation approach, an interoperable relationship has been established between both DEVS simulation and ontology modules. It takes advantage of the benefits achieved by grouping and combining both domains. As seen in the previous section, both modules of the approach were able to adapt and collaborate together.

Using the DEVS formalism, the hierarchical approach easily captures spatial-related aspects of cell compartments and can integrate viral component behaviors at different scales. The SARS-CoV-2 multiplication process lies on a hierarchical structure with different spatial and time scales: the building biomolecules scale typically spans from

nanoseconds to microseconds to the cell level from seconds to hours. It is also used to describe the temporal variation of viral stage state process variables and capture both their temporal and spatial-related evolution.

The OntoRepliCov domain ontology, rich in viral replication semantic knowledge, enriches the simulation model and improves its efficiency. Moreover, the ontological module provides rich knowledge about the microscopic scale of the viral replication process and computes as well the dynamic behavior of micro-level components. Through this module, the approach offers a natural description of the infected host-cell system through the definition of rules governing the viral and cellular component activities (e.g. protein synthesis, assembly of nucleobase, polyprotein production, transcription, and frameshifting). It provides the DEVS module with semantic rules to compute the dynamic state transitions of DEVS models depicting viral and cellular components at a micromolecular level. Such micro-level cannot be modeled by the DEVS model alone. Thus, one of the limitations of DEVS simulation models is their lack of formal semantics and definition of the basic micro-models. For example, in the modeling and simulation of the SARS-CoV-2, there are some basic components representing the micro-level, such as building molecules, which have no states or inputs and outputs. As the ontology was populated with more than 11 million SARS2 sequences from GISAID repository (<https://gisaid.org/>), it is able to consider the smallest and finest components of the replication process, such as amino acids, nucleobases, etc., which the DEVS model is unable to capture.

Therefore, our proposed hybrid approach offers several advantages, including formalizing a relationship between DEVS simulation components and domain knowledge. This relationship has resulted in enriching the simulation model with semantically rich knowledge that the DEVS simulation alone could not successfully model or simulate. The DEVS formalism is very suitable for modeling and simulating the dynamic behavior of a complex system such as the SARS-CoV-2 viral replication. Nevertheless, it cannot express the semantic interpretation of what is occurring in the executed simulation or in the simulation module. Likewise, it cannot deduce any kind of semantic knowledge about the domain area. Therefore, it is not able to enrich or complete the missing data on viral replication. Moreover, the use of a unique DEVS model can potentially increase the complexity of the simulation model, since the formal semantics of the various parameters used by this DEVS model and their relationship with domain knowledge is often overlooked. This explains the need for an ontological model that would complement the DEVS formalism and semantically enhance it.

Classical approaches for modeling and simulation of viruses and their interaction with host cell components

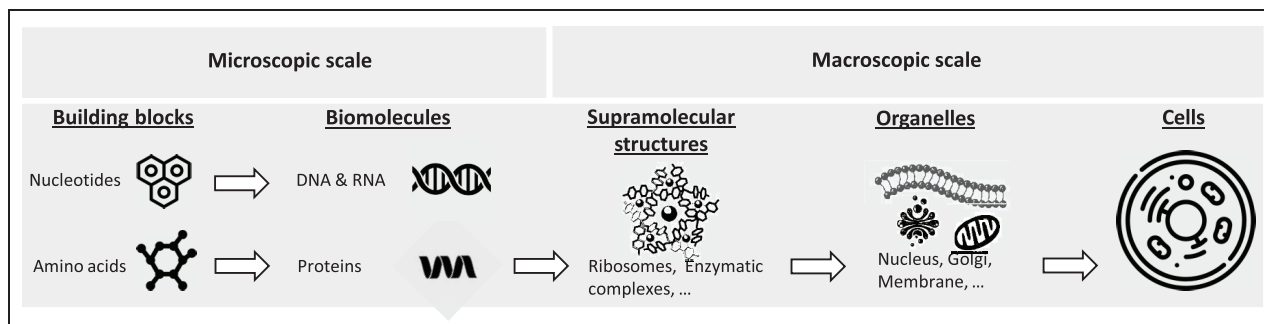


Figure 12. Hierarchy of scales: from the microscopic scale to the macroscopic scale.

are generally focused on single biological scales. The hybrid hierarchical approach proposed in this study meets the objective of integrating biological knowledge within a simulation formalism in which multiple scales are considered at the same time. As presented in Section 2.1, the characteristics of the viral replication process are self-organization and self-reproduction. As described in Figure 12, in this process, both characteristics are extended from the micromolecular scale including small molecules, building blocks, and biomolecules, to the macromolecular scale including supramolecular structures, organelles, and the whole cell.

As discussed above, the proposed hierarchical modeling and simulation approach was specifically designed to be modular and extensible to support continuous updating and parallel development. Considering the modularity of the formalism DEVS, the proposed approach can be easily scalable and adapted to a change of scale in the viral life cycle (or the viral cell interaction process) and support updating in response to rapid advances in the understanding of infection mechanisms. For example, if we intend to add other cellular or viral components involved in the viral replication cycle in light of scientific discovery (as cell immunity), it would just be enough to add the atomic or coupled components and their SWRL rules describing their behavior. Such changes in the hierarchical model will not impact the original model, which can maintain its functionalities even with the new changes. As such, the proposed approach is sufficiently flexible and generic to be applied to the replication of other viruses belonging to (+)RNA virus families with an entirely cytoplasmic life cycle (e.g. Dengue, Zika, Chikungunya, and Hepatitis E viruses).

6. Conclusion and future work

In this work, we mix both DEVS and ontological domains in a common hybrid approach to gain a micro–macro modeling and simulation approach for understanding the SARS-CoV-2 replication process and its interaction with

the host cell. The need to integrate the multiplicity of knowledge and scales of description for modeling complex systems, such as the SARS-CoV-2 replication process and its interaction with the host cell, calls for combining knowledge representation and simulation methods. Therefore, we propose a hybrid hierarchical modeling and simulation approach combining DEVS formalism and ontology domains. Accordingly, the proposed approach is based on two modules: (1) the *DEVS module* to model the hierarchical structure of viral replication (including viral and cellular components), their states during the replication process, the virus–cell interaction at different levels, and simulate its multiplication. And (2) the *ontological module* to provide knowledge about the viral replication process, and SWRL rules to compute the dynamic state transitions of DEVS models depicting viral and cellular components at a micromolecular level.

An immediate and important impact of this work will be a real support to an emerging method named “viro–fluidic” which combines microfluidic and virology at single-cell and single-virus resolutions. Indeed, real-time visualization and quantification of viruses released by a cell are crucial to further decipher infection processes. Kinetics studies at the single-cell level will circumvent the limitations of bulk assays with asynchronous virus replication. Thus, simulation results will be of crucial help in this new cutting-edge approach.

This novel hybrid approach offers several advantages such as modeling, simulating and understanding the virus–host interaction, the different mechanisms involved in the replication process, the role and function of each cellular or viral component in this process, as well as the dynamic behavior of the virus in the different cellular compartment. This approach contributes to studying the SARS-CoV-2 life cycle, from the micro to the macro level, and therefore, meets a primary challenge: understanding virus–cell interaction and virion formation in order to develop efficient weapons against the COVID-19 pandemic. Thus, we hope that the proposed hierarchical modeling and simulation approach might be

complementary tools to understand for understanding virus replication processes and optimizing their therapies.

The contributions proposed in this paper may induce potential future research. The first future work concerns the biological levels covered by the proposed approach. We are interested in integrating additional levels, including cell-cell interaction. The intention is to investigate the viral spread and propagation within cellular tissues and eventually in the whole organism.

Since the proposed DEVS approach has been developed in the CD++ tool which is fully able to run DEVS and Cell-DEVS models, we plan to develop advanced cellular simulation models based on the Cell-DEVS formalism. As the used CD++ tool simulates both DEVS and Cell-DEVS models, this integration is expected to be straightforward. Using these advanced Cell-DEVS simulation models, which has proven to be very effective to study the dynamics of complex models, we can reinforce

our hierarchical approach to address the host immune response and cellular tissue damage in time and space. Furthermore, an important extension of this perspective would be to use the distributed simulation, such as the distributed simulation engine D-CD++.⁷⁰ This will adapt our current work to address the future mutations of the SARS-CoV-2 and facilitate the inclusion of additional levels of DEVS models. Thus, exposing the functionality of our approach to remote users, increasing its reusability, and minimizing the time required for expert biologists to conduct their experiments.⁷⁵

We also plan to extend this work toward the development of a DEVS plugin, namely a CD++ library, able to combine both formal knowledge models and the DEVS simulation models. That could provide a global framework for the modeling and simulation of hierarchical complex systems and be applied to a wide range of domains.

Appendix I

DEVS representation of the cytoplasm compartment, including its formal specification

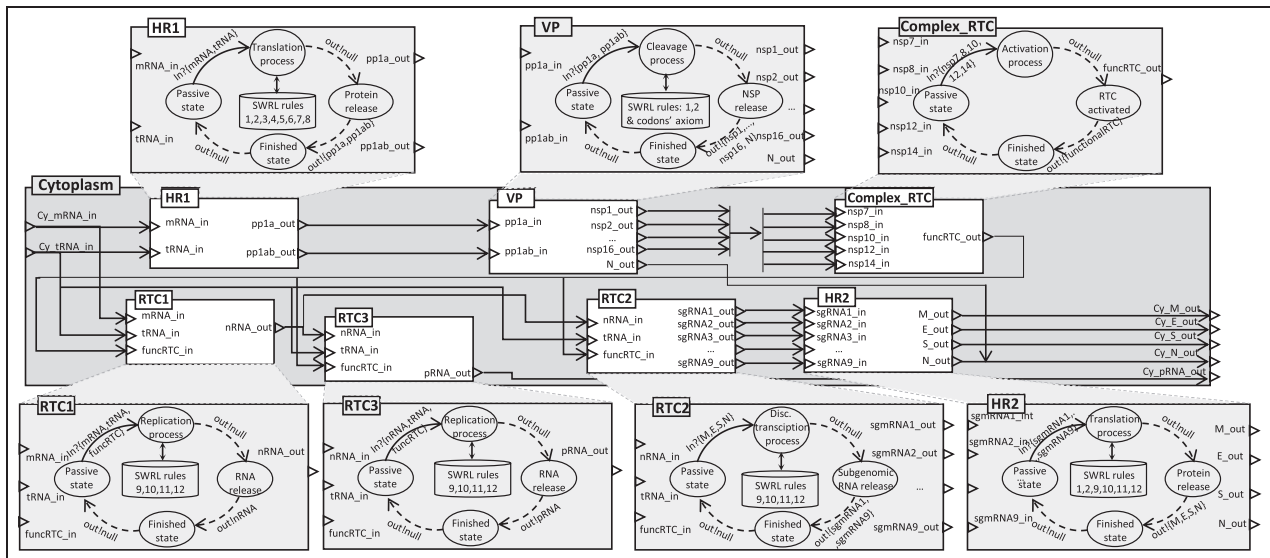


Figure 13. DEVS representation of the cytoplasm compartment, its components, and their exchanges with the ontological inference engine.

Table 8. Continuation of the formal specification of the cytoplasm compartment, and its seven components.

The cytoplasm compartment

The Complex RTC atomic component

$$RTC = (X_{RTC}, Y_{RTC}, S_{RTC}, \delta_{extRTC}, \delta_{intRTC}, \lambda_{RTC}, \lambda_{VP}, ta_{RTC})$$

$$X_{RTC} = \{(RTCIn1, nsp7), (RTCIn2, nsp8), (RTCIn3, nsp10), (RTCIn4, nsp12), (RTCIn5, nsp14)\}$$

$$Y_{RTC} = \{(RTCOut, FuncRTC)\}$$

$$S_{RTC} = \{"Passive", "ActivationProcess", "RTCActivated", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{extRTC}(phase, \sigma, e, x) = (ActivationProcess, activationTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"ActivationProcess", "RTCActivated", "Finished"\}$$

$$\delta_{intRTC}("ActivationProcess", activationTime) = ("RTCActivated", RTCActivatedTime)$$

$$\delta_{intRTC}("RTCActivated", RTCActivatedTime) = ("Finished", finishedTime)$$

$$\delta_{intRTC}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{RTC}("ActivationProcess", \sigma) = \emptyset$$

$$\lambda_{RTC}("RTCActivated", \sigma) = (RTCOut, FuncRTC)$$

$$\lambda_{RTC}("Finished", \sigma) = \emptyset$$

$$\lambda_{RTC}("Passive", \sigma) = \emptyset$$

$$ta_{RTC}("ActivationProcess", \sigma) = activationTime$$

$$ta_{RTC}("RTCActivated", \sigma) = RTCActivatedTime$$

$$ta_{RTC}("Finished", \sigma) = finishedTime = 0$$

$$ta_{RTC}("Waiting", \sigma) = \infty$$

The RTC1 atomic component

$$RTC1 = (X_{RTC1}, Y_{RTC1}, S_{RTC1}, \delta_{extRTC1}, \delta_{intRTC1}, \lambda_{RTC1}, ta_{RTC1})$$

$$X_{RTC1} = \{(RTC1In1, functRTC), (RTC1In2, mRNA), (RTC1In3, tRNA)\}$$

$$Y_{RTC1} = \{(RTC1Out1, nRNA)\}$$

$$S_{RTC1} = \{"Passive", "ReplicationProcess", "RNARelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{extRTC1}(phase, \sigma, e, x) = (ReplicationProcess, replicationTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"ReplicationProcess", "RNARelease", "Finished"\}$$

$$\delta_{intRTC1}("ReplicationProcess", replicationTime) = ("RNARelease", rnaReleaseTime)$$

$$\delta_{intRTC1}("RNARelease", rnaReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{intRTC1}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{RTC1}("ReplicationProcess", \sigma) = \emptyset$$

$$\lambda_{RTC1}("RNARelease", \sigma) = (RTC1Out, nRNA)$$

$$\lambda_{RTC1}("Finished", \sigma) = \emptyset$$

$$\lambda_{RTC1}("Passive", \sigma) = \emptyset$$

$$ta_{RTC1}("ReplicationProcess", \sigma) = replicationTime$$

$$ta_{RTC1}("RNARelease", \sigma) = rnaReleaseTime$$

$$ta_{RTC1}("Finished", \sigma) = finishedTime = 0$$

$$ta_{RTC1}("Passive", \sigma) = \infty$$

The RTC2 atomic component

$$RTC2 = (X_{RTC2}, Y_{RTC2}, S_{RTC2}, \delta_{extRTC2}, \delta_{intRTC2}, \lambda_{RTC2}, ta_{RTC2})$$

$$X_{RTC2} = \{(RTC2In1, nRNA), (RTC2In1, tRNA)\}$$

$$Y_{RTC2} = \{(RTC2Out1, sgRNA1), (RTC2Out2, sgRNA2), (RTC2Out3, sgRNA3), (RTC2Out4, sgRNA4),$$

$$(RTC2Out5, sgRNA5), (RTC2Out6, sgRNA6), (RTC2Out7, sgRNA7),$$

$$(RTC2Out8, sgRNA8), (RTC2Out9, sgRNA9)\}$$

$$S_{RTC2} = \{"Passive", "DisTranscriptionProcess", "SGRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{extRTC2}(phase, \sigma, e, x) = (DisTranscriptionProcess, disTransProcessTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"DisTranscriptionProcess", "SGRelease", "Finished"\}$$

$$\delta_{intRTC2}("DisTranscriptionProcess", disTransProcessTime) = ("SGRelease", sgReleaseTime)$$

$$\delta_{intRTC2}("SGRelease", sgReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{intRTC2}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{RTC2}("DisTranscriptionProcess", \sigma) = \emptyset$$

$$\lambda_{RTC2}("SGRelease", \sigma) = (RTC2Out1, sgRNA1) \wedge (RTC2Out2, sgRNA2) \wedge (RTC2Out3, sgRNA3) \wedge (RTC2Out4, sgRNA4)$$

$$\wedge (RTC2Out5, sgRNA5) \wedge (RTC2Out6, sgRNA6) \wedge (RTC2Out7, sgRNA7) \wedge (RTC2Out8, sgRNA8) \wedge (RTC2Out9, sgRNA9)$$

$$\lambda_{RTC2}("Finished", \sigma) = \emptyset$$

$$\lambda_{RTC2}("Passive", \sigma) = \emptyset$$

$$ta_{RTC2}("DisTranscriptionProcess", \sigma) = disTransProcessTime$$

$$ta_{RTC2}("SGRelease", \sigma) = sgReleaseTime$$

$$ta_{RTC2}("Finished", \sigma) = finishedTime = 0$$

$$ta_{RTC2}("Passive", \sigma) = \infty$$

Table 9. Continuation of the formal specification of the cytoplasm compartment, and its seven components.

The cytoplasm compartment

The RTC3 atomic component

$$RTC3 = (X_{RTC3}, Y_{RTC3}, S_{RTC3}, \delta_{ext_{RTC3}}, \delta_{int_{RTC3}}, \lambda_{RTC3}, ta_{RTC3})$$

$$X_{RTC3} = \{(RTC3In1, nRNA), (RTC3In2, tRNA)\}$$

$$Y_{RTC3} = \{(RTC3Out1, pRNA)\}$$

$$S_{RTC3} = \{"Passive", "ReplicationProcess", "RNARelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{ext_{RTC3}}(phase, \sigma, e, x) = ("ReplicationProcess", replicationTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"ReplicationProcess", "RNARelease", "Finished"\}$$

$$\delta_{int_{RTC3}}("ReplicationProcess", replicationTime) = ("RNARelease", rnaReleaseTime)$$

$$\delta_{int_{RTC3}}("RNARelease", rnaReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{int_{RTC3}}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{RTC3}("ReplicationProcess", \sigma) = \emptyset$$

$$\lambda_{RTC3}("RNARelease", \sigma) = (RTC3Out, pRNA)$$

$$\lambda_{RTC3}("Finished", \sigma) = \emptyset$$

$$\lambda_{RTC3}("Passive", \sigma) = \emptyset$$

$$ta_{RTC3}("ReplicationProcess", \sigma) = replicationTime$$

$$ta_{RTC3}("RNARelease", \sigma) = rnaReleaseTime$$

$$ta_{RTC3}("Finished", \sigma) = finishedTime = 0$$

$$ta_{RTC3}("Passive", \sigma) = \infty$$

The HR2 atomic component

$$HR2 = (X_{HR2}, Y_{HR2}, S_{HR2}, \delta_{ext_{HR2}}, \delta_{int_{HR2}}, \lambda_{HR2}, ta_{HR2})$$

$$X_{HR2} = \{(HR2In1, sgRNA1), (HR2In2, sgRNA2), (HR2In3, sgRNA3), (HR2In4, sgRNA4), (HR2In5, sgRNA5), (HR2In6, sgRNA6), (HR2In7, sgRNA7), (HR2In8, sgRNA8), (HR2In9, sgRNA9), (HR2In10, tRNA)\}$$

$$Y_{HR2} = \{(HR2Out1, M), (HR2Out2, E), (HR2Out3, S), (HR2Out4, N)\}$$

$$S_{HR2} = \{"Passive", "TranslationProcess", "ProteinRelease", "Finished"\} \times \mathbb{R}_0^+$$

$$\delta_{ext_{HR2}}(phase, \sigma, e, x) = (TranslationProcess, translationTime) \text{ if } phase = "Passive"$$

$$(phase, \sigma - e) \text{ if } phase \in \{"TranslationProcess", "ProteinRelease", "Finished"\}$$

$$\delta_{int_{HR2}}("TranslationProcess", translationTime) = ("ProteinRelease", proteinReleaseTime)$$

$$\delta_{int_{HR2}}("ProteinRelease", proteinReleaseTime) = ("Finished", finishedTime)$$

$$\delta_{int_{HR2}}("Finished", finishedTime) = ("Passive", \infty)$$

$$\lambda_{HR2}("TranslationProcess", \sigma) = \emptyset$$

$$\lambda_{HR2}("ProteinRelease", \sigma) = (HR2Out1, M) \wedge (HR2Out2, E) \wedge (HR2Out3, S) \wedge (HR2Out4, N)$$

$$\lambda_{HR2}("Finished", \sigma) = \emptyset$$

$$\lambda_{HR2}("Passive", \sigma) = \emptyset$$

$$ta_{HR2}("TranslationProcess", \sigma) = translationTime$$

$$ta_{HR2}("ProteinRelease", \sigma) = proteinReleaseTime$$

$$ta_{HR2}("Finished", \sigma) = finishedTime = 0$$

$$ta_{HR2}("Passive", \sigma) = \infty$$

Acknowledgements


The authors acknowledge the use of CD++ toolkit services, and are grateful to Prof. Gabriel Wainer from the Department of Systems and Computer Engineering, Carleton University (Ottawa, ON, Canada) for his helpful discussions on the use of CD++ toolkit services.

Funding

This work was supported by the REACTing COVID-19 initiative (Research and ACTION targeting emerging infectious diseases) and by the French National Research Agency under reference "ANR-20-COVI-0006-01."

ORCID iDs

Ali Ayadi  <https://orcid.org/0000-0003-1660-4100>

Isabelle Imbert  <https://orcid.org/0000-0001-5630-9945>

Notes

1. <https://www.worldometers.info/coronavirus/>
2. <https://www.eclipse.org/>
3. <https://www.boost.org/>
4. <http://www.sce.carleton.ca/esg/CDppBuilder/>
5. <https://cygwin.com>
6. <https://protege.stanford.edu/>
7. <http://owl-cpp.sourceforge.net/>
8. <http://owl.cs.manchester.ac.uk/tools/fact/>
9. <https://librdf.org/>

References

1. Xu H, Liu L, Zhao L, et al. Psychological impact and compliance with staying at home of the public to covid-19 outbreak during Chinese spring festival. *Int J Environ Res Public Health* 2022; 19: 916.

2. Halvorsen GS, Simonsen L and Sneppen K. Spatial model of ebola outbreaks contained by behavior change. *PLoS ONE* 2022; 17: e0264425.
3. Ravi V, Saxena S and Panda PS. Basic virology of SARS-CoV 2. *Ind J Med Microbiol* 2022; 40: 182–186.
4. Shehata AA, Attia YA, Rahman MT, et al. Diversity of coronaviruses with particular attention to the interspecies transmission of SARS-CoV-2. *Animals* 2022; 12: 378.
5. Nabeela Sultan M, Mohammed Shoaib M, Shagufta Aleem M, et al. Global academic journal of pharmacy and drug research. *Proteins* 2022; 4: 2.
6. Hernández-Morales R, Becerra A, Campillo-Balderas J, et al. Structural biology of the SARS-CoV-2 replisome: evolutionary and therapeutic implications. In: Rosales-Mendoza S, Comas-Garcia M and Gonzalez-Ortega O (eds) *Biomedical Innovations to Combat COVID-19*. New York: Elsevier, 2022, pp. 65–82.
7. Smith EM, Rakestraw C and Farroni JS. Research integrity during the covid-19 pandemic: perspectives of health science researchers. *Account Res*. Epub ahead of print 6 February 2022. DOI: 10.1080/08989621.2022.2029704.
8. Jung C, Kmiec D, Koepke L, et al. Omicron: what makes the latest SARS-CoV-2 variant of concern so concerning? *J Virol* 2022; 96: e0207721.
9. Mostafavi E, Dubey AK, Teodori L, et al. SARS-CoV-2 omicron variant: a next phase of the covid-19 pandemic and a call to arms for system sciences and precision medicine. *Medcomm* 2022; 3: e119.
10. Ayadi A, Frydman C, Laddada W, et al. Combining DEVS and semantic technologies for modeling the sars-cov-2 replication machinery. In: *2021 Annual Modeling and Simulation Conference (ANNSIM)*, Fairfax, VA, 19–22 July 2021, pp. 1–12. New York: IEEE.
11. Laddada W, Soualmia LF, Zanni-Merk C, et al. Ontoreplicov: an ontology-based approach for modeling the sars-cov-2 replication process. *Proced Comput Sci* 2021; 192: 487–496.
12. Zeigler BP, Muzy A and Kofman E. *Theory of modeling and simulation: discrete event and iterative system computational foundations*. Cambridge, MA: Academic Press 2018.
13. Martins M, Ferreira S Jr and Vilela M. Multiscale models for biological systems. *Curr Opin Colloid Inter Sci* 2010; 15: 18–23.
14. Romano M, Ruggiero A, Squeglia F, et al. A structural view of SARS-CoV-2 RNA replication machinery: RNA synthesis, proofreading and final capping. *Cells* 2020; 9: 1267.
15. Grebennikov D, Kholodareva E, Sazonov I, et al. Intracellular life cycle kinetics of SARS-CoV-2 predicted using mathematical modelling. *Viruses* 2021; 13: 1735.
16. Scudellari M. How the coronavirus infects cells and why delta is so dangerous. *Nature* 2021; 595: 640–644.
17. Uraki R and Kawaoka Y. Host glycolipids in SARS-CoV-2 entry. *Nature Chem Biol* 2022; 18: 6–7.
18. Zeigler B, Traoré MK, Zacharewicz G, et al. *Value-based learning healthcare systems: Integrative modeling and simulation*. London: Institution of Engineering and Technology, 2019.
19. Zeigler BP, Kim TG and Praehofer H. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Cambridge, MA: Academic Press, 2000.
20. Zeigler BP and Muzy A. From discrete event simulation to discrete event specified systems (DEVS). *IFAC-PapersOnLine* 2017; 50: 3039–3044.
21. Hamri A. FwkDEVS: a DEVS/GDEVS modeling and simulation framework. *JDF*, 2016. <https://docplayer.fr/85508806-Fwkdevs-simulateur-devs-gdevs-amine-hamri-laboratoire-des-sciences-de-l-information-et-des-systemes-lsis-umr-7296-aix-marseille-university-france.html>
22. Sarjoughian HS and Zeigler B. DEVSJAVA: basis for a DEVS-based collaborative M&S environment. *Simul Ser* 1998; 30: 29–36.
23. Quesnel G, Duboz R and RamatÉ. The virtual laboratory environment—an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simul Model Pract Theory* 2009; 17: 641–653.
24. Filippi JB and Bisgambiglia P. JDEVS: an implementation of a DEVS based formal framework for environmental modelling. *Environ Model Softw* 2004; 19: 261–274.
25. Bolduc JS and Vangheluwe H. *pythondevs: a modeling and simulation package for classical hierarchal DEVS*. Technical report, MSDL, Université De McGill, Quebec, QC, Canada, 2001.
26. Nutaro J. *Adevs (a discrete event system simulator)*. Arizona Center for Integrative Modeling & Simulation (ACIMS), University of Arizona, Tucson, 1999, <http://www.ece.arizona.edu/nutaro/index.php>
27. Wainer GA. CD++: a toolkit to develop DEVS models. *Softw Pract Exp* 2002; 32: 1261–1306.
28. Wainer G, Liu Q, Dalle O, et al. Applying cellular automata and DEVS methodologies to digital games: a survey. *Simul Gaming* 2010; 41: 796–823.
29. Liu Q and Wainer G. Parallel environment for DEVS and Cell-DEVS models. *Simulation* 2007; 83: 449–471.
30. Wainer GA. *Discrete-event modeling and simulation: a practitioner's approach*. Boca Raton, FL: CRC press, 2017.
31. Rubin DL, Lewis SE, Mungall CJ, et al. National center for biomedical ontology: advancing biomedicine through structured organization of scientific knowledge. *OMICS* 2006; 10: 185–198.
32. Cao Q, Zanni-Merk C, Samet A, et al. KSPMI: a knowledge-based system for predictive maintenance in industry 4.0. *Robot Comput-Integrat Manuf* 2022; 74: 102281.
33. Liaw ST, Rahimi A, Ray P, et al. Towards an ontology for data quality in integrated chronic disease management: a realist review of the literature. *Int J Med Inform* 2013; 82: 10–24.
34. Mora M, Wang F, Gómez JM, et al. Development methodologies for ontology-based knowledge management systems: a review. *Expert Syst* 2022; 39: e12851.
35. Jefferys EE and Sansom MS. Computational virology: molecular simulations of virus dynamics and interactions. *Adv Exp Med Biol* 2019; 1215: 201–233.
36. Ng KY and Gui MM. COVID-19: development of a robust mathematical model and simulation package with consideration for ageing population and time delay for control action and resusceptibility. *Phys D: Nonlin Phenomena* 2020; 411: 132599.

37. Overton CE, Stage HB, Ahmad S, et al. Using statistics and mathematical modelling to understand infectious disease outbreaks: COVID-19 as an example. *Infect Dis Model* 2020; 5: 409–441.
38. Ivorra B, Ferrández MR, Vela-Pérez M, et al. Mathematical modeling of the spread of the coronavirus disease 2019 (COVID-19) taking into account the undetected infections. The case of China. *Commun Nonlin Sci Numer Simul* 2020; 88: 105303.
39. Machado JT, Rocha-Neves JM and Andrade JP. Computational analysis of the SARS-CoV-2 and other viruses based on the Kolmogorov's complexity and Shannon's information theories. *Nonlin Dynam* 2020; 101: 1731–1750.
40. Ahmad J, Ikram S, Ahmad F, et al. SARS-CoV-2 RNA dependent RNA polymerase (RdRp)–a drug repurposing study. *Heliyon* 2020; 6: e04502.
41. Elfiky AA. SARS-CoV-2 RNA dependent RNA polymerase (RdRp) targeting: an in silico perspective. *J Biomol Struct Dyn* 2021; 39: 3204–3212.
42. Elmezayen AD, Al-Obaidi A, Şahin AT, et al. Drug repurposing for coronavirus (COVID-19): in silico screening of known drugs against coronavirus 3CL hydrolase and protease enzymes. *J Biomol Struct Dynam*, 2020, https://www.researchgate.net/profile/Anas-Alobaidi/publication/340778261_Drug_repurposing_for_coronavirus_COVID-19_in_silico_screening_of_known_drugs_against_coronavirus_3CL_hydrolase_and_protease_enzymes/links/5e9f8e25a6fdcc20bb35fb17/Drug-repurposing-for-coronavirus-COVID-19-in-silico-screening-of-known-drugs-against-coronavirus-3CL-hydrolase-and-protease-enzymes.pdf
43. Arantes PR, Saha A and Palermo G. Fighting COVID-19 using molecular dynamics simulations. *ACS Centi Sci* 2020; 6: 1654–1656.
44. Amin SA, Ghosh K, Gayen S, et al. Chemical-informatics approach to COVID-19 drug discovery: Monte Carlo based QSAR, virtual screening and molecular docking study of some *in-house* molecules as papain-like protease (PLpro) inhibitors. *J Biomol Struct Dynam* 2021; 39: 4764–4773.
45. Lalmuanawma S, Hussain J and Chhakchuak L. Applications of machine learning and artificial intelligence for covid-19 (sars-cov-2) pandemic: a review. *Chaos Soliton Fract* 2020; 139: 110059.
46. Wang W and Ruan S. Simulating the SARS outbreak in Beijing with limited data. *J Theor Biol* 2004; 227: 369–379.
47. Bai Y and Jin Z. Prediction of SARS epidemic by BP neural networks with online prediction strategy. *Chaos Soliton Fract* 2005; 26: 559–569.
48. Teles P. A time-dependent SEIR model to analyse the evolution of the SARS-CoV-2 epidemic outbreak in Portugal. *arXiv preprint*, 2004, <https://arxiv.org/abs/2004.04735>
49. Jiang S, Li Q, Li C, et al. Mathematical models for devising the optimal SARS-CoV-2 strategy for eradication in China, South Korea, and Italy. *J Trans Med* 2020; 18: 1–11.
50. Wu JT, Leung K and Leung GM. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. *The Lancet* 2020; 395: 689–697.
51. Altmann PM, Liu LL and Michor F. The mathematics of cancer: integrating quantitative models. *Nature Rev Cancer* 2015; 15: 730–745.
52. Chowdhury S, Forkan M, Ahmed SF, et al. Modeling the SARS-CoV-2 parallel transmission dynamics: asymptomatic and symptomatic pathways. *Comput Biol Med* 2022; 143: 105264.
53. Velten K. *Mathematical modeling and simulation: introduction for scientists and engineers*. Hoboken, NJ: John Wiley & Sons, 2009.
54. de Oliveira LP, Hudebine D, Guillaume D, et al. A review of kinetic modeling methodologies for complex processes. *Oil Gas Sci Tech* 2016; 71: 45.
55. Bagabir S, Ibrahim NK, Bagabir H, et al. Covid-19 and artificial intelligence: genome sequencing, drug development and vaccine discovery. *J Infect Public Health* 2022; 15: 289–296.
56. Cozac R, Medzhidov N and Yuki S. Predicting inhibitors for SARS-CoV-2 RNA-dependent RNA polymerase using machine learning and virtual screening. *arXiv preprint*, 2006, <https://arxiv.org/abs/2006.06523>
57. Ivanov J, Polshakov D, Kato-Weinstein J, et al. Quantitative structure–activity relationship machine learning models and their applications for identifying viral 3CLpro-and RdRp-targeting compounds as potential therapeutics for COVID-19 and related viral infections. *ACS Omega* 2020; 5: 27344–27358.
58. Randhawa GS, Soltysiak MP, El Roz H, et al. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: Covid-19 case study. *PLoS ONE* 2020; 15: e0232391.
59. Mahapatra S, Nath P, Chatterjee M, et al. Repurposing therapeutics for covid-19: rapid prediction of commercially available drugs through machine learning and docking. *PLoS ONE* 2020; 15: e0241543.
60. Touati R, Haddad-Boubaker S, Ferchichi I, et al. Comparative genomic signature representations of the emerging COVID-19 coronavirus and other coronaviruses: high identity and possible recombination between bat and pangolin coronaviruses. *Genomics* 2020; 112: 4189–4202.
61. Pavlova A, Zhang Z, Acharya A, et al. Machine learning reveals the critical interactions for SARS-CoV-2 spike protein binding to ACE2. *J Phys Chem Lett* 2021; 12: 5494–5502.
62. Ghosh K, Amin SA, Gayen S, et al. Chemical-informatics approach to COVID-19 drug discovery: exploration of important fragments and data mining based prediction of some hits from natural origins as main protease (Mpro) inhibitors. *J Mol Struct* 2020; 1224: 129026.
63. Tang B, He F, Liu D, et al. Ai-aided design of novel targeted covalent inhibitors against SARS-CoV-2. *Biomolecules* 2020; 12: 746.
64. Alakus TB and Turkoglu I. Comparison of deep learning approaches to predict covid-19 infection. *Chaos Soliton Fract* 2020; 140: 110120.
65. Benameur N, Mahmoudi R, Zaid S, et al. SARS-CoV-2 diagnosis using medical imaging techniques and artificial intelligence: a review. *Clin Imag* 2021; 76: 6–14.

66. Yuille AL and Liu C. Limitations of deep learning for vision, and how we might fix them. *The Gradient*, 2019, <https://thegradient.pub/the-limitations-of-visual-deep-learning-and-how-we-might-fix-them/#:~:text=Here%2C%20we%20identify%20three%20main,of%20tasks%20that%20are%20important.>
67. Rasmussen CB, Kirk K and Moeslund TB. The challenge of data annotation in deep learning—a case study on whole plant corn silage. *Sensors* 2022; 22: 1596.
68. Li M, Fang Y, Tang Z, et al. Explainable COVID-19 infections identification and delineation using calibrated pseudo labels. *IEEE Trans Emerg Topic Comput Intell* 2023; 7: 26–35.
69. Rajan S, McKee M, Hernández-Quevedo C, et al. What have European countries done to prevent the spread of COVID-19? Lessons from the COVID-19 health system response monitor. *Health Policy* 2022; 126: 355–361.
70. Wainer GA, Madhoun R and Al-Zoubi K. Distributed simulation of DEVS and cell-DEVS models in CD++ using web-services. *Simul Model Pract Theory* 2008; 16: 1266–1292.
71. Snijder EJ, Bredenbeek PJ, Dobbe JC, et al. Unique and conserved features of genome and proteome of SARS-coronavirus, an early split-off from the coronavirus group 2 lineage. *J Mol Biol* 2003; 331: 991–1004.
72. Kim H and Yin J. Robust growth of human immunodeficiency virus type 1 (HIV-1). *Biophysical J* 2005; 89: 2210–2221.
73. Fehr AR and Perlman S. Coronaviruses: an overview of their replication and pathogenesis. *Coronaviruses* 2015; 1282: 1–23.
74. Lauer SA, Grantz KH, Bi Q, et al. The incubation period of coronavirus disease 2019 (Covid-19) from publicly reported confirmed cases: estimation and application. *Ann Intern Med* 2020; 172: 577–582.
75. Zacharewicz G, Frydman C and Giambiasi N. Lookahead computation in G-DEVS/HLA environment. *SNE Simul News Europe* 2006; 16: 15–24.
- ICUBE laboratory at Université de Strasbourg. He is working on conceptual representation and discrete event simulation of biological complex systems. His email address is ali.ayadi@unistra.fr.
- Claudia Frydman** is a full professor at Aix-Marseille Université. She is also a member of the Laboratoire d’Informatique et des Systèmes (LIS), and she has been a referee for several scientific journals and a member of the program committee in various international conferences. She is working on knowledge based simulation, especially discrete Event System Specification modeling and simulation. Her email address is claudia.frydman@lis-ab.fr.
- Wissame Laddada** is a post-doctoral researcher at Aix Marseille Université in collaboration with INSA Rouen Normandie. She is working on knowledge based systems. Her email address is wissame.laddada@univ-amu.fr.
- Isabelle Imbert** is a full professor in Biology at Aix-Marseille Université and member of the Architecture and Function of Biological Macromolecules Laboratory (AFMB—CNRS/Aix-Marseille University). She is considered as one of the French experts on coronavirus. Her email address is isabelle.imbert@univ-amu.fr.
- Cecilia Zanni-Merk** is a full professor in Computer Science at INSA Rouen Normandie and the head of the MIND team at the LITIS laboratory. Her research focuses on conceptual representation and inference processes applied to problem-solving. Her email address is cecilia.zanni-merk@insa-rouen.fr.
- Lina F Soualmia** is an associate professor in Computer Science at Université de Rouen and member of the TIBS team at the LITIS Laboratory. She is working in the domain of Artificial Intelligence applied to Health. Her email address is lina.soualmia@litislab.fr.

Author biographies

Ali Ayadi is an associate professor at the CSTB research team (Complex Systems and Translational Biology),