Special Issue Article

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
2024, Vol. 100(12) 1297–1318
© The Author(s) 2024
DOI: 10.1177/00375497241291871
journals.sagepub.com/home/sim

100 volumes of SIMULATION—20 years of DEVS research

Gabriel Wainer and Sasisekhar Govind

Abstract

The growth of real-time embedded applications has surged in recent years, marked by both an increase in the number and complexity of tasks performed across various industries. Modeling and simulation (M&S) has been used for enhancing product quality while reducing lifecycle costs, primarily through improved testability and maintainability of real-time embedded systems applications, as M&S-based design approach allows for early hardware functionality testing, fosters collaboration between hardware and software teams, and shortens the product development cycle. The discrete-event system specification (DEVS) framework has been used for developing such discrete-event M&S systems. This article starts by highlighting the various DEVS publications in our journal over the past 20 years, reflecting the evolving landscape of simulation methodologies, which we organized into four categories: theory, methodology, tools, and applications. This curated selection reflects the diversity of topics and the evolution of scholarship within the field, encouraging further exploration and innovation. We conclude with recent research in the field, including our own research in real-time embedded systems development using DEVS software for modeling, simulation, and real-time execution of models. This paves the way for future discussions in this important field of research.

Keywords

Applications in science and engineering, embedded systems, BR, theory and methodology, DEVS methodology, discrete systems

1. Introduction

In recent years, the growth of real-time (RT) embedded applications has been remarkable, with not only an increase in the number of systems but in the complexity of tasks they perform. Recent advancements in computing technology have enabled the automation of a wide range of tasks in RT applications to levels once thought impossible, resulting in more complex and sophisticated applications across multiple industries.[1] However, this complexity poses significant challenges in the development of RT software, where critical considerations such as functionality, predictability, and reliability must be addressed to meet operational requirements. RT software often struggles to scale-up effectively, with extensive testing efforts that do not guarantee a bug-free product, which can lead to increased risks of failures in deployed applications. Modeling and Simulation (M&S) techniques have shown to be valuable tools for analyzing such scenarios, providing methods to create products of higher quality while simultaneously reducing lifecycle costs. This is primarily achieved through enhanced testability and maintainability brought about by M&S approaches. Using an M&S-based design approach facilitates early testing of hardware functionality, fostering collaboration between hardware and software teams, reducing the product development cycle and time-to-market period as well as improving the overall efficiency of the development process. This early-stage verification capability allows identifying and addressing potential issues and accelerates innovation by allowing for iterative refinements and optimizations throughout the design phase. M&S methodologies also allow stakeholders to visualize the model before implementation, fostering collaboration and identifying potential flaws early, thus reducing costly errors in later stages.

In particular, the discrete-event system specification (DEVS), introduced by Zeigler,[2] provides a robust

Advanced Real-Time Simulation Laboratory, Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada

Corresponding author:
Gabriel Wainer, Advanced Real-Time Simulation Laboratory, Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S5B6, Canada.
Email: gwainer@sce.carleton.ca

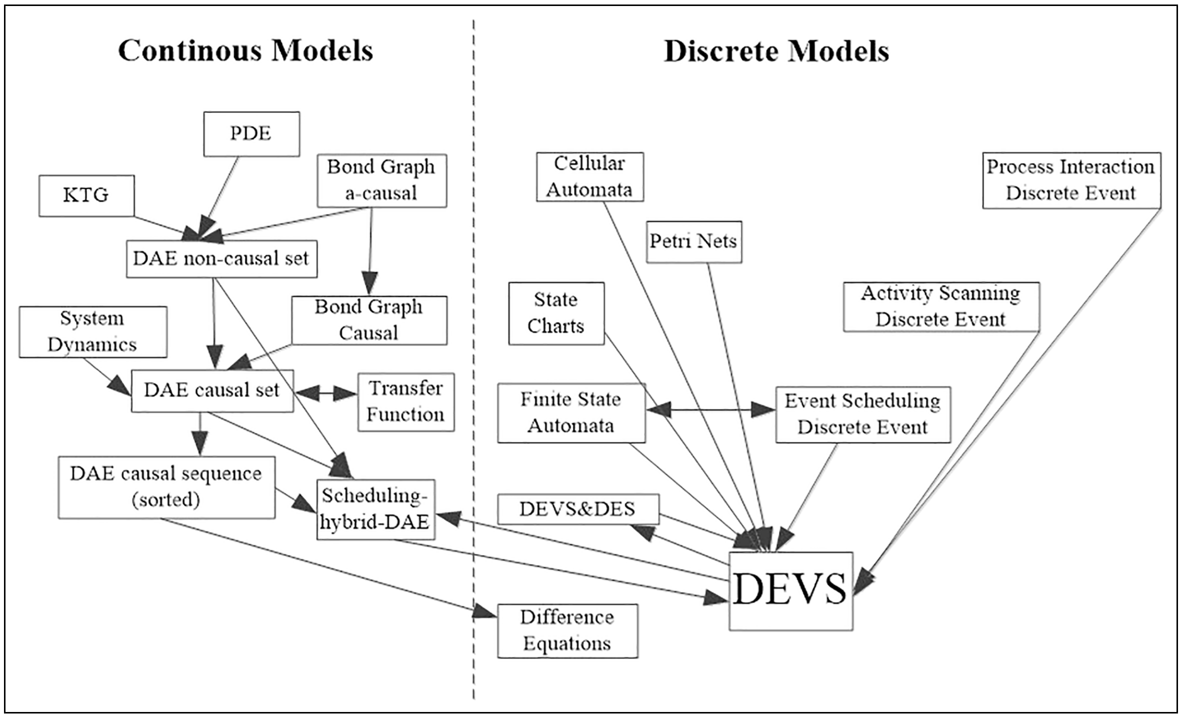

Figure 1. Formalism transformation graph. Adapted from Vangheluwe.[3]

theoretical framework for the development of discrete-event M&S systems. Since its inception, DEVS has shown itself to be useful, versatile, and effective in many applications across different fields. Based on systems theory, DEVS provides a formal method for building complex systems, offering a structured way to create hierarchical modular models that encourage reuse and improve system design modularity.

One of the key strengths of the DEVS formalism is its ability to define hierarchical and modular models, which allows users to construct complex systems in a structured manner. To this end, there are two types of DEVS formal models: behavioral (or atomic) and structural (or coupled). Atomic models encapsulate the behavior of a single system component using basic elements such as its inputs, the outputs that it generates, and state variables. Atomic models include functions to compute the next states and outputs based on the received inputs and the existing state. On the contrary, coupled models consist of multiple interconnected submodels (atomic or coupled). This allows for the representation of complex interactions and dependencies among multiple system components. DEVS supports a layered approach to system design, where individual components can be developed, tested, and validated independently before being integrated into a more extensive system configuration.

Although DEVS is a theoretical M&S technique, it has been applied in practical settings ranging from logistics and manufacturing to telecommunications and health care. Its adaptability to various domains showcases its capability to model dynamic, event-driven systems accurately. The hierarchical structure of DEVS models also promotes reusability, as components can be repurposed across different simulations and modified without impacting the entire system.

Another significant advantage is the separation of model definition, implementation, simulation, and experimentation through independent formal specifications and execution frameworks. This clear separation allows researchers and engineers to focus on the conceptual design of models independently from the technical specifics of execution, enhancing clarity and organization in the development workflow. Furthermore, DEVS supports models across different platforms, which means that models can be implemented in various environments (desktop, RT, parallel, embedded, distributed) without the loss of fidelity in its behavior. This is due to the fact that the DEVS simulation algorithm has been formally verified, ensuring its correctness. This rigorous foundation makes it ideal for safety-critical systems and large engineering projects.

Another important advantage of DEVS, presented by Vangheluwe in 2000,[3] is introduced in Figure 1. DEVS can be seen as a unifying framework that serves as a common denominator for other modeling methodologies and formalisms. Figure 1 shows a formalism transformation graph, showcasing how DEVS can effectively bridge various modeling paradigms. This is advantageous in the context of complex embedded systems where components may be described through different modeling techniques, such as Petri nets, Statecharts, or agent-based models. The ability to compose these models allows for a multi-paradigm representation of the system, capturing interactions across multiple models and formalisms.

In this Volume No. 100 of SIMULATION, we want to highlight the extensive growth of DEVS publications in our journal. The DEVS contributions to this area have been substantial and varied, reflecting the evolving landscape of simulation methodologies. In the past 20 years alone, our journal has served as a prominent platform for a variety of research in the field of DEVS M&S. This period has seen numerous innovative applications, theoretical advancements, and practical implementations of the DEVS formalism that we discuss here. From foundational studies that enhance its theoretical underpinnings to applied research showcasing its utility in diverse fields, the journal has published numerous articles that collectively enrich our understanding of DEVS.

As we celebrate this 100th volume, this paper focuses on the significant contributions of researchers and practitioners who have dedicated their efforts to advancing DEVS methodologies. Their work not only reflects the rigorous academic inquiry associated with the formalism but also showcases its relevance in addressing complex real-world challenges through effective simulation techniques.

The investigation into DEVS research published in the journal SIMULATION has produced valuable insights, which we organized into four categories: theory, methodology, tools, and applications. The focus was particularly on the advances in DEVS through the publications in the journal, with a focus on its application in RT computing. A thorough review was conducted of all SIMULATION publications since 2000, resulting in the selection of 134 articles. From this list, 129 were chosen as representative of significant and relevant research in simulation. This curated selection aims to reflect the diversity of topics and the evolution of research within the field. The selection process sought to consider a broad range of studies and is not meant to diminish any works not included. The goal is to understand recent DEVS simulation research, encouraging further exploration and innovation in the field.

The paper is structured as follows: Section 2 explores recent theoretical advancements in DEVS, emphasizing research articles published in this journal. Section 3 delves into the latest developments in DEVS methodology. In Section 4, we review various articles that focus on DEVS simulation tools, while Section 5 showcases a wide range of applications within the domain of DEVS. Finally, Section 6 highlights current trends in DEVS M&S, including insights from our own research on DEVS applications for RT embedded systems.

2. Advances in DEVS theory

In the last 20 years, SIMULATION has witnessed numerous advances in the field of DEVS theory. These developments improved our theoretical understanding of the field. A historical perspective of the origins of DEVS theory was presented by Ören and Zeigler.[4] This work reviewed the historical development and contributions of A. Wayne Wymore to the field of system-theoretic foundations in M&S. The paper highlights the establishment of the first systems engineering department and formulating the general system theory, particularly through the development of the DEVS formalism, and its significant impact on advancing the field. In this section, we introduce some of these significant advancements published in SIMULATION. Some of the published articles advanced the DEVS formalism itself, as research has enhanced the foundational aspects of DEVS, ranging from continuous to hybrid DEVS models that integrate continuous- and discrete-event simulations (DESs).

For instance, the paper "Generalized Discrete Event Simulation of Bond Graph"[5] presented a GDEVS approach for bond graph modeling, integrating DEVS with bond graphs to represent physical systems' energy and power interactions (including mechanical, electrical, and hydraulic domains, with varied case studies that demonstrate the method's applicability in modeling complex engineering systems). Similarly, Barros[6] defined heterogeneous flow system specification (HFSS), which models combined discrete and continuous systems with dynamic structure. HFSS enables the modular and hierarchical construction of models, accommodating multirate sampling systems and numerical solutions for differential equations with discontinuities. Similarly, Barros[7] Barros proposed a formal representation for hybrid mobile components HFSS, extending HFSS theory by Barros[6] to include mobile components, which are essential for representing systems where entities move between different models. In terms of variable structure models in DEVS which was first introduced, Barros[7,8] introduces a method for incorporating variable structures in DEVS-based component M&S. The approach allows for dynamic changes in model structure during simulation, accommodating the evolving nature of complex systems.[9] Applications include adaptive control systems and reconfigurable manufacturing systems, highlighting the benefits of dynamic structure modeling for flexibility and scalability.

Barros research also included hybrid hierarchical models in πHYFLOW,[10] a modular approach to the process interaction worldview, supporting the definition of hybrid hierarchical models. The formalism is demonstrated through examples such as a DC–DC converter model, showcasing its ability to handle hybrid systems with complex interactions and varying topologies. The paper "Modular representation of asynchronous geometric integrators with support for dynamic topology"[11] presents a framework for the modular representation of asynchronous geometric integrators, enabling support for dynamic topology in system models. This approach leverages the DEVS formalism to handle asynchronous interactions and topological changes, enhancing the ability to model and simulate complex systems with varying structures. Applications in multi-agent systems and adaptive control show this method's effectiveness in maintaining accuracy and efficiency.

Various research has focused on the use of DEVS for model verification. For instance, Saadawi and Wainer[12] presented a new extension to the DEVS formalism, called the Rational Time-Advance DEVS (RTA-DEVS), which permits modeling the behavior of RT systems that can be modeled by DEVS, and can be formally verified with standard model-checking algorithms and tools. We introduce a procedure to create timed automata (TA) models that are behaviorally equivalent to the original RTA-DEVS models that enable the use of the available TA tools and theories for formal model checking. Similarly, the research by Cicirelli et al.[13] explored the use of time stream Petri nets (TSPNs) for modeling, analyzing, and enacting workflow processes specified by TSPNs. The authors show that the functional and temporal properties of a TSPN model can be checked using exhaustive verification or a DEVS-based simulation tool and the enactment rests on a decentralized enactment engine based on the service-oriented computing paradigm, which enables execution of workflow processes where the coordinated activities may involve cross-boundary organizations. In the work by Fonseca i Casas,[14] an algorithm and a simulation infrastructure are defined to transform a simulation model represented using the DEVS formalism to the standard language (SDL). The algorithm can be viewed as a mechanism to represent DEVS models graphically. The timed sequential machine (TSM) formalism, defined by Giambiasi,[15] presents a formalism less expressive than the DEVS or TA, but is well adapted to intermediate abstraction levels in the design or analysis processes of discrete systems. The authors also propose methods for the minimization of completely specified finite-state TSMs and for the simplification of incompletely specified finite-state TSMs. Imprecise-DEVS (I-DEVS), introduced by Mello and Wainer,[16] proposed a new method to integrate schedulability analysis to address the design and execution challenges of embedded RT systems under transient overloading conditions. The integrated framework allows for formal verification of high-level models and their execution as RT tasks, incorporating worst-case execution time (WCET) and worst response time (WRT) analysis. The proposed method demonstrates improved predictability and feasibility in scheduling, especially under overload conditions, ensuring that critical tasks meet their deadlines while optional tasks are handled based on system capacity.

Other research focused on the theory of DEVS simulation; for instance, Hong and Kim[17] defined a specification of multi-resolution modeling space (MRMS) for multi-resolution system simulation. The proposed specification is based on concepts of decoupling multi-resolution modeling (MRM) and multi-resolution simulation. This MRMS specification supports MRM in two parts: resolution conversion for dynamically changing simulation model structures and resolution matching interfaces between events in different resolutions. Goldstein et al.[18] introduced a multi-scale approach to representing simulated time, addressing the challenges of time granularity in simulations. By integrating different time scales within a single simulation framework, the method allows for the accurate modeling of processes occurring at varying temporal resolutions. This approach is particularly useful in scenarios such as biological systems and network simulations, where events occur at multiple, interconnected time scales.

Some research extended DEVS theory to include other kinds of formal specifications; for instance, Castro et al.[19] defined an extension to DEVS based on the use of the probability spaces theory, called stochastic DEVS (STDEVS). STDEVS provides a formal framework for M&S of general non-deterministic discrete systems. The main theoretical properties of the STDEVS framework are treated, including a new definition of legitimacy of models in the stochastic context and a proof of STDEVS closure under coupling. Similarly, in the work by Hu and Zeigler,[20] an activity-based framework was introduced; this new extension to DEVS links information and energy using a quantization-based approach for modeling information processing and defines weighted activity to model the energy consumption of information processing. The authors provide a formal description of this framework that enables one to study the interaction between information and energy in energy-aware information processing. Castro and Kofman[21] generalize the concept of activity of continuous-time signals by defining the concept of activity of order n, providing a method to estimate the number of sections of polynomials up to order n which are needed to represent any given signal with a certain accuracy, and deriving a theoretical lower bound for the number of steps performed by quantization-based integration algorithms in the simulation of ordinary differential equations (ODEs).

A fundamental advance in theory of DEVS was presented by E. Kofman and his team at the Universidad National de Rosario; this is discussed by Castro et al.[22] a comprehensive review of the evolution and current state of quantized state systems (QSSs) methods for the DES of continuous-time systems. It discusses the theoretical foundations, key developments, and practical applications of QSS, highlighting its advantages in terms of computational efficiency and accuracy. The review covers various QSS techniques, including first-order and higher-order methods, and their application in engineering and scientific simulations. Kofman[23] introduced a quantization-based approach to simulate differential algebraic equation (DAE) systems, leveraging the DEVS formalism for efficient numerical integration. The quantization method discretizes the continuous state space, reducing computational complexity and improving simulation performance. The article shows various applications in electrical circuit simulation and mechanical systems to demonstrate the effectiveness of the approach. Similarly, Migoni et al.[24] present new classes of numerical ODE solvers based on QSS. The authors present a first-order accurate QSS-based stiff system solver, which exhibits properties that make it a potentially attractive alternative to the classical numerical ODE solvers. A first-order accurate QSS-based solver designed for solving marginally stable systems, called the centered QSS (CQSS), is also outlined. Another advance, presented by Bergero and Kofman,[25] is the introduction of vectorial DEVS (VECDEVS), an extension of the DEVS formalism for large-scale system modeling and parallel simulation. The extension includes an algorithm for automatically partitioning VECDEVS models into submodels for efficient parallel execution. The research by Fernández and Kofman[26] defined a standalone implementation of the QSS integration methods that improve computation times significantly, achieving performance gains of over an order of magnitude compared with their implementation of a DEVS engine. Also, Bergero et al.[27] discuss the effects of replacing time discretization with state quantization in the simulation of a one-dimensional advection–diffusion–reaction (ADR) equation. By discretizing the ADR equation in space using a regular grid, the study compares the performance of traditional time discretization algorithms with QSS methods. The analysis shows that the second-order linearly implicit QSS method significantly outperforms classical algorithms (Dormand-Prince, Radau, and Differential/Algebraic System Solver), achieving more than an order of magnitude improvement in computational efficiency. Also, Di Pietro et al.[28] present enhancements to linearly implicit QSS methods, focusing on improving the stability and efficiency of simulations involving stiff ODEs. The proposed methods adaptively adjust quantization intervals to handle stiffness more effectively, resulting in better numerical stability and reduced computational costs.

3. Advances in DEVS methodology

The theoretical advances in DEVS led to further advances in the methodological aspects of DEVS, with researchers developing sophisticated algorithms and methods that facilitate the simulation of complex models. This includes a variety of new methods and algorithms that improved performance, enabling simulations that were previously computationally prohibitive. In this section, we will explore these advances in greater detail, shedding light on their implications for the future of DEVS methodology through publications in SIMULATION.

This section explores several methodological advancements in simulation research, organizing them into different categories based on their approaches. We first discuss DEVS methodologies in the context of parallel computing, which utilizes multiple processors to execute simulations simultaneously, to improve computational performance. We then discuss DEVS application in distributed systems, where simulations are normally executed across geographically dispersed machines. Another area is the integration of various modeling paradigms (for instance, discrete-event, agent-based, and continuous models) to create more advanced representations of complex systems. Furthermore, we discuss how DEVS methodologies contribute to improved verification and validation processes in simulation software, ensuring that the models represent the intended phenomena and provide credible results. We also discuss spatial modeling, which emphasizes the integration of spatial and geographical factors into simulations. The section concludes a discussion on RT simulations and their application in developing systems that require tight timing constraints. These advancements show the nature of DEVS simulation research and the need of employing diverse methodologies to build complex real-world applications.

Parallel discrete-event simulation (PDES) has allowed executing DESs in parallel environment. In this sense, research by Liu and Wainer[29] shows how to conduct parallel execution of DEVS and Cell-DEVS models using a time warp synchronization layer. Existing simulation algorithms for parallel DEVS and parallel Cell-DEVS models were redesigned for parallel optimistic simulation, simplifying message analysis with *wall clock time slice* and enhancing state-saving with a two-level *user controller state-saving* mechanism, which demonstrated significant speedups. An extension of this work, introduced by Liu and Wainer,[30] introduced multicore acceleration of DEVS, using data and event parallelism through multi-grained parallelism. The idea is to use innovations such as SIMD intrinsics, double-buffered DMA, and SPE-based synchronization to accelerate both memory-bound and compute-bound computational kernels in parallel DEVS simulations.

Other important results in this field include the results presented in[31] which presented a generic model partitioning algorithm to decompose multiscale models into partition blocks using cost modeling and analysis. This approach leverages DEVS to translate domain-specific information into homogeneous cost metrics, enabling optimal partitioning and improved parallelism. Similarly, Cardoen et al.[32] introduced DEVS-Ex-Machina (dxex), a parallel DEVS simulator that implements multiple synchronization protocols. The simulator enhances simulation efficiency by enabling the user to switch protocols at runtime. The authors show that the modularity of dxex does not impede performance and that it can achieve significant speedups under the right conditions. Similarly, the research by Adegoke et al.[33] proposes a conceptual framework to standardize the development of DEVS parallel simulations by defining a taxonomy of key concepts and formal definitions to enable symbolic reasoning and automated code synthesis. The authors offer a systematic approach to constructing DEVS simulators, enhancing the interoperability, scalability, and performance of distributed simulation (DS) systems.

The scaled RT synchronization (SRTS) for parallel DEVS simulation of continuous and hybrid systems introduced by Bergero et al.[34] involves splitting models into submodels that are concurrently simulated on different processors, with local synchronization to a scaled version of physical time. SRTS ensures bounded numerical errors despite synchronization imperfections. Furthermore, the adaptive-SRTS adjusts the time-scaling parameter dynamically based on system workload, improving simulation efficiency. The challenges and risks posed by common errors associated with parallel simulations are discussed by Nutaro and Ozmen.[35] The authors propose two approaches to combat the risks to reproducible parallel simulations, including the Model of Computation approach[36] and DEVS, which is highlighted as a robust approach to M&S that separates model concerns from simulation algorithms.

DS became popular in the 21st century due to several key factors, including advances in networking, cloud computing, virtualization, distributed software, and middleware, which improved DSs by providing scalable resources without significant hardware investment. In addition, the development of standards, such as the High-Level Architecture (HLA), enhanced interoperability and collaboration among researchers. Some of these aspects were summarized by Tolk[37] a comprehensive review of 30 years of simulation interoperability research. This article revisited the levels of conceptual interoperability model (LCIM) and discussed various methods such as message-oriented methods and common object models. The DEVS formalism is highlighted for its role in ensuring that models from different systems align conceptually, promoting accurate and reliable simulations across diverse domains. Other efforts use ontologies for discrete-event M&S,[38] discussing various existing taxonomies and products and an ontology with four subclasses. The concept of DEVS was used to standardize and formalize the representation of simulation models under a process-oriented subclass.

The research team at Arizona Center for Integrative Modeling and Simulation (ACIMS) was a key player in this area. For instance, in the work by Nutaro and Sarjoughian,[39] ACIMS researchers presented a framework for DS based on system-theoretic and logical-process concepts. It outlines a three-part worldview comprising modeling formalisms, abstract simulators, and computational environments, and defines a unified notion of causality for parallel simulations. Later, research introduced by Wutzler and Sarjoughian[40] introduced a shared abstract model approach to enable interoperability among DEVS-based simulations across different programming languages and engines. The approach supports efficient, scalable simulations and flexible integration of independently developed models, making it suitable for heterogeneous simulation environments on parallel and distributed platforms.

Many other researchers contributed in this field. For instance, Zacharewicz et al.[41] showed a workflow environment that leverages DEVS and GDEVS formalisms for DS. The authors proposed a method to transform Workflow specifications into GDEVS models using XML and detailed the development of a HLA-compliant distributed environment. Likewise, Boukerche et al.[42] introduced a formal approach using DEVS to design a RT run-time infrastructure (RTI) for large-scale DSs. DEVS was used to model and predict the performance of RTI designs, facilitating early identification of potential issues and optimizing system performance. DEVS provided a structured, reusable framework that enhances the efficiency and accuracy of RTI system design.

Technological advances, such as the HLA standard and web services, provided further advances in the field. Mittal et al.[43] presented DEVS/SOA, a service-oriented architecture framework for DEVS-based M&S. DEVS ensured consistent model execution across different platforms, supporting net-centric operations and interoperability in DS environments. Similarly, Wang et al.[44] showed a survey of service-oriented simulation frameworks. DEVS showed its pivotal role, providing a structured methodology to translate high-level specifications into executable models that can be tested and validated in a net-centric environment. We introduced the first RESTful interoperability middleware, which was extended by Wang and Wainer[45] leveraging cloud computing and simulation as a service (SimaaS) for scalable and collaborative development.

More recently, researchers in the work by Sun et al.[46] introduced the Universal Service-Oriented Software (USOS) system, which was designed to enhance the development and integration of service-oriented software, emphasizing flexibility, reusability, and efficiency. DEVS is utilized in the context of model debugging and flexible model-driven applications, ensuring accurate simulation and efficient model management.

Foundational research in the field was introduced by Mosterman and Vangheluwe[47] where they introduced the concept of Computer Automated Multi-Paradigm Modeling (CAMPaM). This domain-independent framework for multi-paradigm modeling encompasses multi-abstraction, multi-formalism, and meta-modeling. It highlights the central role of model transformations through graph grammar to automate model abstraction and refinement. Illustrated with a power window system example, CAMPaM demonstrates how leveraging different models and simulation techniques facilitates comprehensive design and analysis across various development stages. Further research in this field, presented by Hardebolle and Boulanger,[48] discussed multi-paradigm modeling, emphasizing the integration of various modeling techniques to address complex system behaviors. The DEVS formalism is used to manage discrete-event dynamics within these multi-paradigm frameworks, ensuring a structured approach to modeling system interactions and transformations. More recently, research by Denil et al.[49] constructed a multi-paradigm DEVS simulation model for AUTOSAR-based electronic control units. DEVS formalism was used to support the concurrent simulation of control, plant, and environment models, aiding in the assessment of design choices and ensuring system behavior and performance in automotive applications.

Other research focuses on multi-models and multi-paradigm modeling; for instance, in the work by Risco-Martín et al.,[50] eUDEVS, combined executable UML (eUML) with DEVS to provide rigorous simulation semantics, enabling precise and verifiable execution of eUML models within a DEVS-based simulation framework. In the work by Gianni et al.,[51] SimArch defined a layered architecture designed to simplify the development of DS by converting local simulation systems into DS systems. The authors demonstrate the effectiveness of SimArch through a reference model, showing how it significantly reduces the development effort compared with traditional DS environments. Likewise, in the work by Schmidt et al.,[52] a methodology for fidelity evaluation of complex, modular simulation models was introduced; particularly for cyber-physical systems (CPSs). It employs the system entity structure (SES) ontology to describe a family of models and their configurations, enabling automated test case generation and execution. The approach integrates with MATLAB/Simulink, facilitating a structured and automated fidelity evaluation process, which is important for ensuring the reliability and accuracy of CPS simulations. Similarly, Santucci et al.[53] focused on enhancing the SES framework to incorporate abstraction hierarchies and time granularity into DEVS M&S. The SES framework represents system elements and their relationships hierarchically, and it was extended to manage different levels of detail and temporal granularity in complex systems. DEVS and Modelica were combined describing agent-based models by Sanz and Urquia.[54] Modelica, a robust modeling language for continuous-time systems, lacks support for agent-based models (ABM). The authors proposed using DEVS to formally describe agent behaviors within Modelica, enabling the combination of ABM with Modelica's continuous-time modeling capabilities. Other agent-based simulation multi-paradigm methods were introduced; for instance, Kim and Kim[55] defined a formal framework for modeling and simulating mobile agent systems using the mobile DEVS. A modified abstract simulation algorithm was introduced, and the AgentSim environment was built.

The team led by Lin Uhrmacher at the University of Rostock has used multi-paradigm modeling for biological models; for instance, Peng et al.[56] presented a methodology for reusing experiment specifications by successive composition. The research shows a new methodology for developing complex models by composing existing ones, using a domain-specific language for simulation experiments called Simulation Experiment Specification via a Scala Layer (SESSL) for automated experiment generation. This is demonstrated through the Wnt/β-catenin signaling pathway, to validate composed models' semantic correctness. DEVS is used implicitly for structured model development and validation.

As stated earlier, DEVS includes advanced facilities for verification and validation of simulation software and state-based software. For instance, Yilmaz[57] addressed the challenge of verifying collaborative behavior in component-based DEVS models. It presents formalized local consistency conditions and decidable inference mechanisms to ensure compositional consistency and safe refinement of DEVS models. The proposed method includes interaction policies and mediator consistency analysis to verify that component interactions comply with collaboration constraints. The practical utility of these techniques is demonstrated through a semaphore-based process synchronization case study. Yacoub et al.[58] introduced a formalism combining DEVS and PROMELA for the modeling, verification, and validation of complex software systems like video games. This hybrid approach leverages the strengths of both model-checking and DES to ensure accurate and efficient verification and validation processes. DEVS provides a formal representation of the system's behavior, enabling the combination of qualitative and quantitative analysis, enhancing the robustness and correctness of the software design. Similarly, Samuel et al.[59] showed a framework for the full verification process of complex systems that supports the graphical modeling of both the system of interest and the requirements to be checked, using the High-Level Language for Systems Specification. DEVS formalism is utilized as the semantic domain for simulation, enabling the framework to integrate system behavior modeling and formal verification using tools like UPPAAL.

An important methodological advance included the definition of spatial models. In this sense, the research by Wainer and Giambiasi[60] introduced the Cell-DEVS formalism for spatial modeling. Cell-DEVS combines cellular automata with the DEVS framework to address limitations in traditional cellular automata approaches. This methodology improves precision and performance, allowing for the simulation of complex environmental systems, such as pollution spread, fire dynamics, and watershed formation, in various simulation environments (single-user, RT, distributed/parallel). By leveraging DEVS, Cell-DEVS facilitates model construction, execution, and analysis, enabling subject matter experts without extensive programming knowledge to study and analyze the phenomena in their respective domains. The article described the implementation and performance of Cell-DEVS models, which includes explicit complex timing behaviors by introducing concepts of transport and inertial delays. The research also includes a hierarchical simulator and a flattened one, which obtained a speedup of 2 to 7 times in execution time. Sun and Hu[61] investigated the performance of dynamic structure DEVS for large-scale cellular space models, focusing on models that can dynamically change their structure and couplings. The study demonstrates that we can enhance simulation efficiency by concentrating on active models.

DEVS methodology has shown itself to be a versatile framework for RT computing, CPS, RT co-simulation, and hardware–software co-design. DEVS showed how to effectively handle complexity, enhance system interoperability, and streamline the design process in these areas. DEVS provides a structured approach for modeling systems that require strict timing adherence, and it includes an efficient specification of event-driven behaviors, as well as supporting synchronization and adequate semantic support for modeling. This helps engineers analyze performance and test strategies before real-world deployment. DEVS facilitates the simultaneous interaction of multiple models, aiding comprehensive analysis across different subsystems and improving overall system design. In this area, numerous research has been presented in SIMULATION.

For instance, the research by Song and Kim[62] showed the use of RT DEVS to analyze a discrete-event control system, focusing on untimed controller design and safety analysis with weak synchronization. The proposed reachability analysis algorithm (timed behavior analysis) uses vector time and clock matrices to manage the infinite state space, although it has high space complexity. DEVS allowed unified modeling, analysis, performance evaluation, and virtual prototyping for controller design and implementation. Similarly, Sarjoughian et al.[63] presented a co-design methodology using the DEVS/DOC formalism for distributed mission training systems. The study uses the formalism to characterize the system architecture and predict scalability and performance under different configurations. In the work by Cao[64] a co-simulation of multi-resolution models used the multi-resolution (MR)-agent-DEVS specification within an HLA framework. This framework proposes the structure and formal description of MR-agent-DEVS federate models, emphasizing the communication mechanism and simulation process.

Various M&S tools for CPS were integrated through the MECSYCO middleware,[65] which allows the co-simulation of different pre-existing and heterogeneous tools by using a DEVS-based wrapping strategy. DEVS wrappers are developed (using the Functional Mock-up Interface) for continuous tools, leveraging the DEVS & DESs hybrid formalism and QSS solvers. This approach ensures the rigorous integration of heterogeneous M&S tools, facilitating comprehensive and synchronized co-simulation of CPS.

The team led by Rodrigo Castro at the Universidad de Buenos Aires developed techniques for the modeling of multi-paradigm CPS under a unified M&S platform, leveraging DEVS for the combined specification of QSS solvers and stochastic queueing networks. For instance, a hybrid supervisory controller was developed for unmanned aerial vehicles (UAVs) with sensitive physical-computational couplings.[66] The controller dynamically balances the use of processing resources in a setting where computational (discrete) and physical (continuous) dynamics compete with each other, while operating on an unpredictable environment. The approach is presented as a general end-to-end DEVS-based design blueprint for a broader class of CPS.

DEVSys presented by Kapos et al.[67] introduces a framework that bridges SysML and DEVS to automate the generation of executable simulation code. This framework leverages the model-driven architecture approach, integrating a DEVS SysML profile to enrich SysML models with simulation-specific properties. It also includes a DEVS meta-model and utilizes the QVT (Query/View/Transformation) language to transform SysML models into DEVS models. The Sarjoughian and Gholami[68] extended the DEVS formalism by incorporating actions specified with RT constraints, defined as time windows, which are to be executed within hard RT deadlines. The models are simulated using an abstract simulator protocol that ensures actions are completed within their designated time windows under constrained computational resources.

In summary, this section discussed the numerous advancements in DEVS methodologies, reflecting a dynamic and growing field that continues to adapt to the challenges presented by complex systems. These advancements improve the usability and efficacy of DEVS for academic and industry professionals.

Numerous methodological advances have been focused on *DEVS simulation algorithms*. For instance, Lee et al. [69] describe the design and development of the DEVS/Generic Data Distribution Management (GDDM) environment, a layered simulation framework that supports data distribution management using space-based quantization schemes. DEVS/GDDM aims to reduce data communication in DSs, enhancing performance and scalability. The paper provides empirical results using a ballistic missiles simulation to demonstrate the effectiveness of the DEVS/GDDM environment in reducing local computation demands and achieving favorable communication data reduction. The DEVS BUS research presented by Kim et al.[70] introduced a heterogeneous simulation framework that uses the DEVS BUS to integrate various simulation models and tools. The framework facilitates the interoperability and coordination of different modeling paradigms, enabling comprehensive system simulations. The DEVS BUS acts as a middleware, supporting seamless communication and data exchange between diverse simulation components.

Other research focused on formally defining DEVS-based simulators.[71] The approach involves converting existing factory models, modeled by employing a job-resource relation network (JR-net), into DEVS models to take advantage of DEVS modularity and hierarchical modeling capabilities. The converted DEVS models are used to simulate factory operations, providing detailed insights into production processes and system performance. As well, Nutaro et al. [72] integrate DEVS with continuous system simulation techniques to manage time effectively in hybrid simulations. The split modeling approach proposes the creation of a simulator where the individual subcomponents of a model are simulated using the most appropriate algorithms: numerical integration methods for continuous components and discrete-event algorithms (DEVS + Hybrid Input-Output Automata (HIOA) + event scheduling) for discrete components.

4. DEVS simulation tools

The formal definitions of DEVS enable the straightforward establishment of DEVS simulation environments. Given that the algorithms are rigorously defined and verified, developing software implementations of these algorithms is easy, resulting in a variety of DEVS tools available. Some of these tools have been previously discussed in the SIMULATION journal and will be detailed further in this section. Such advanced DEVS tools enhance the M&S process, including advanced features that facilitate more accurate and efficient simulations. In addition, these tools provide user-friendly platforms for practitioners, promoting the practical application of DEVS methodologies. A critical consideration in the design of simulation software is the performance of these tools. To this end, it is essential to not only support modeling but also evaluate the performance of DEVS simulation models. This evaluation includes identifying bottlenecks to optimize efficiency, ensuring that the tools can operate effectively in various applications.

For instance, Gonzalez et al.[73] presented a novel simulation tool designed specifically for the modeling and control of distributed systems. The tool can handle complex interactions and RT constraints, making it suitable for many applications in distributed computing. DEVS is utilized to provide a robust framework that simplifies the simulation of distributed systems and enhances model accuracy. Research by Cho et al.[74] discuss the RTDEVS/CORBA environment, which integrates DEVS with CORBA middleware to support the simulation-based design of distributed RT systems. The framework ensures model continuity from design to execution, enhancing flexibility and reducing the time and cost of testing. It provides a comprehensive solution for managing software complexity and consistency in large-scale distributed systems. Later, Lee et al.[75] explored the use of DEVS and HLA. The integration enhances interoperability and reusability of models, supporting the development of efficient models by allowing comprehensive testing and analysis. This was later expanded by Lee et al.[76] with an environment that integrates various transportation models to improve traffic management and control, facilitating the development of advanced transportation solutions. The hierarchical approach allows for detailed and scalable modeling of transportation networks. DEVS is used to structure the hierarchical models, to simulate complex transportation systems, to and improve the efficiency of traffic management solutions.

Some of the general-purpose tools for DEVS modeling include, for instance, those introduced in Wainer et al.[77] where various tools developed for the graphical specification and visualization of DEVS models were analyzed. These tools are designed to enhance user interaction with DEVS models by providing intuitive interfaces and advanced visualization capabilities. They aim to make DEVS modeling more accessible and comprehensible, especially for complex systems. This was further expanded by Bonaventura et al.[78] where CD++ Builder defined a graphical modeling tool to simplify the construction and simulation of DEVS models. CD++ Builder uses Eclipse and provides a user-friendly interface for creating and testing DEVS models, making the modeling process more intuitive and efficient.

The PowerDEVS environment, presented by Kofman's team by Bergero and Kofman,[79] is a software tool developed for the modeling and RT simulation of hybrid systems. PowerDEVS integrates DEVS with continuous system simulation using QSS methods to efficiently handle hybrid models. This tool supports the simulation of systems with both continuous and discrete components, enhancing the analysis of complex dynamic systems. QSS was also used by Nutaro[80] which discussed an extension to the OpenModelica compiler that allows the use of Modelica models within DESs (specifically, aDEVS, by James Nutaro). The extension aims to enhance the versatility and applicability of Modelica models by integrating them into DEVS-based simulation environments. This integration supports more comprehensive and detailed system simulations.

Research by Stolpe et al.[81] introduced a logic-based event controller designed for means-end reasoning within simulation environments. The controller supports complex decision-making processes, enabling simulations to handle intricate event sequences and dependencies more effectively. DEVS provides the framework for the event controller, allowing for the integration of logical reasoning capabilities within DEVS-based simulations and enhancing their ability to manage complex scenarios.

The team led by Hans Vangheluwe introduced by Van Tendeloo and Vangheluwe[82] in which they use computational resource usage models, or "activity" models, to enhance the performance of DEVS simulators. By augmenting DEVS models with domain-specific information, the approach optimizes data structure usage, load balancing, and model allocation. The improvements are validated using the PythonPDEVS simulator, showing significant performance gains. PythonPDEVS was also used by Van Mierlo et al.[83] which presents a visual modeling, simulation, and debugging environment. The authors deconstruct and reconstruct the PythonPDEVS simulator using the Statecharts formalism, which allows the integration of debugging features such as breakpoints, manual state manipulation, and reversible debugging. This environment combines traditional code debugging concepts with simulation-specific operations to provide a robust tool for model debugging. The approach facilitates interactive, visual debugging and enhances the usability and effectiveness of DEVS simulations.

Visualization aspects are crucial in most simulation tools, as discussed by Maleki et al.[84] by a team lead by Azam Khan and Rhys Goldstein at Autodesk Research. The research discusses the design of visual interfaces for DEVS modeling, aimed at making DEVS more accessible to end-user programmers. The interfaces were designed to simplify the modeling process and enhance user experience, enabling users with varying levels of expertise to construct and manipulate DEVS models effectively. Similarly, this team developed DesignDEVS,[85] a simulation environment tailored for practical applications of DEVS modeling. The research focuses on various features and enhancements made to improve usability and simulation efficiency, emphasizing practical aspects such as model management and execution.

This non-comprehensive list shows the variety of DEVS tools and their publication in our journal. In fact, Van Tendeloo and Vangheluwe[86] provided a comprehensive evaluation of various DEVS simulation tools, assessing their performance, usability, and feature sets. The evaluation aims to guide users in selecting the appropriate tool for their specific needs, highlighting the strengths and limitations of each tool. This research made use of an important analysis tool, presented by Wainer et al.[87] the DEVStone benchmark. DEVStone is a synthetic benchmark we defined with the purpose of having a standard mechanism to evaluate the performance of DEVS-based simulators. DEVStone generates a suite of models with varied structures and behaviors to automate performance evaluations. DEVStone was also used by Risco-Martín et al.[88] where it was used to evaluate DEVS-based simulators' performance. The authors introduce new equations to compute the number of triggered events and propose a new model, HOmem, which has similar computational requirements as a complex benchmark but is easier to implement and analyze. The study compares five DEVS simulators—DEVSJAVA, CD++, aDEVS, xDEVS, and PyPDEVS—on two hardware platforms, analyzing execution time and memory usage. The results highlight significant differences in performance, with aDEVS and xDEVS generally performing better in terms of memory usage and execution time, respectively.

As we can see, there is a broad array of DEVS tools available, each of which emphasizes the framework's abilities across various applications. Each tool possesses unique features tailored to specific needs, such as advanced modeling capabilities, user-friendly interfaces for non-programmers, and performance analysis tools for optimizing simulations. Overall, this diversity reflects ongoing innovation in the field, which allows users to choose the right tools for their work when DEVS methodology is needed.

5. Advances in DEVS applications

The exploration of DEVS has expanded into diverse domains such as health care, transportation, and many others, which we will introduce in this section. The interdisciplinary nature of DEVS encourages collaboration among researchers from different fields, enhancing the methodologies used and fosters the development of new models, driving further advancements in DEVS theory. For instance, DEVS has been used for planning and designing complex rail infrastructures,[89] up to simulating networks of spiking neurons.[90] This section will include examples of applications in a wide range of areas of interest.

For instance, in the work by Xie and Verbraeck,[91] a particle filter-based data assimilation framework for DESs using DEVS was presented. This method addresses state retrieval and variable dimension problems with an interpolation operation and particle filters. Validated through a gold mine system case study, the framework improves accuracy and robustness in estimating system states from RT data. A follow-up research in this area by Huang et al.[92] examined the impact of three critical experimental conditions on particle filter-based data assimilation in discrete-event systems. By analyzing time intervals between iterations, particle numbers, and measurement error levels using an M/M/1 queuing system, the study found that shorter intervals and optimal particle counts significantly improve estimation accuracy.

DEVS has been used in numerous projects in the field of development of *artificial systems*. For instance, Ghosh and Giambiasi[93] introduced an innovative approach to modeling and simulating mixed-signal electronic designs using nVHDL. By integrating DEVS with nVHDL, the methodology unifies the temporal resolutions of analog and discrete subsystems, facilitating comprehensive simulations of mixed-signal circuits. This approach enhances the accuracy and efficiency of electronic design simulations, addressing the challenges posed by the differing natures of analog and digital components. Likewise, Hamri et al.[94] highlight the advantages of using the Min–Max DEVS formalism for modeling and simulating digital circuits. The Min–Max DEVS formalism effectively handles the inherent uncertainty in hardware delays by defining delays as temporal windows (intervals), allowing a more realistic and flexible representation of timing behavior compared with fixed or probabilistic delays used in traditional hardware description languages. Research introduced by Sarjoughian et al.[95] defined a DEVS-based simulation environment for MIPS32 processor designs. The simulator models single-cycle, multi-cycle, and pipeline processors at the register-transfer level using DEVS formalism, providing detailed visualization, performance data collection, and animation of processor operations. It enhances the educational experience by allowing students to explore and understand processor architectures through interactive simulations, supporting various levels of abstraction and reusable component-based models. In the work by Migoni et al.,[96] DEVS-based algorithms using QSS were used to simulate switched mode power Supplies. The study demonstrated that linearly implicit QSS models significantly improve simulation speed and accuracy compared with traditional solvers. The method efficiently handles stiffness and discontinuities, and it can achieve 3–200 times faster simulations with enhanced accuracy. DEVS has been used to study network performance,[97] by modeling a wireless sensor networks, including routing management, energy consumption, and CPU activity, as well as CPSs;[98] here, SimConnect and SimTalk facilitate the coordination of multiple simulators, preserving event causality among different models of computation and abstraction levels.

Some of the artificial systems defined had a specific domain in the area of *defense and emergency planning.* For instance, Pang and Mathew[99] proposed a reconfigurable discrete-event control framework for network-centric warfare. Using DEVS, the study extends an existing framework with matrix decomposition to enable reliable ad hoc reconfiguration of tasks and resources during multiple simultaneous missions. The environment allows different platforms to join and leave the network seamlessly, enhancing the flexibility and responsiveness of command and control structures. Extensive simulations involving the Singapore Armed Forces demonstrate the effectiveness of the proposed system in dynamically managing mission tasks and resource allocation without causing operational blockages. In this field, research by Kim et al.[100] presented a framework for battle experiments that integrates mission-level and engagement-level models. The study demonstrated that combining these heterogeneous models provides new insights into military doctrines, particularly in naval air defense scenarios. The interoperation of models reveals optimal decision-making timings for command and control that are not obtainable from standalone models, highlighting the benefits of simulation interoperation for comprehensive doctrine analysis. The paper in the work by Seo et al.[101] presented a DEVS-based modeling method for engagement-level military simulations, focusing on the structural and behavioral delineation of combat entities. The study categorizes combat entities into platform and weapon models and details their core activities using hierarchical and compositional DEVS models. This approach improves model reusability and provides comprehensive insights into engagement scenarios, validated through anti-submarine warfare simulations.

Various models have been used to deal with the current global warming crisis, and various DEVS models exist to study *energy* and *environmental* applications. For instance, Fard and Sarjoughian[102] presented a DEVS-based interaction model for simulating water and energy interactions. Utilizing the Knowledge Interchange Broker approach, the framework integrates WEAP and LEAP models through a RESTful architecture, enhancing flexibility and scalability. Research by Risco-Martín et al.[103] presented DEVS-BLOOM, a novel framework designed for the RT monitoring and management of harmful algal and cyanobacterial blooms. Utilizing the DEVS formalism, the framework integrates Internet of Things (IoT) and digital twin technologies to support high-performance hazard detection and response. The study demonstrated the ability to provide early warning systems, predict harmful formations, and optimize the deployment of resources like sensors and unmanned surface vehicles. Byon et al.[104] highlighted the stochastic nature of wind turbine loading and the complexities of maintenance decision-making. It compared scheduled maintenance and condition-based maintenance (CBM) strategies, finding that CBM significantly improves wind power generation by reducing turbine failure rates. The simulation results showed that CBM offers more efficient maintenance planning and better economic benefits. A further investigation presented by Pérez et al.[105] showed that DEVS-based modeling provides significant insights into optimizing maintenance schedules, improving wind farm performance, and reducing operational costs by effectively managing component deterioration and failure. Similarly, a new extension on the previous research, presented by Pérez,[106] developed a simulation-driven online scheduling algorithm to optimize the maintenance and operation of wind farm systems.

Various research has delved into the realm of social science, and *social simulation* models have been defined. Research presented by Bouanan et al.[107] showed an architecture using DEVS for simulating the dissemination of information within multi-layer social networks. The paper presents a set of models to represent individual behaviors and their interactions within a social network, validated through a military scenario. Likewise, Khalil and Wainer[108] presented a comprehensive framework for modeling, simulating, and predicting indoor carbon dioxide (CO_2) dispersion using Cell-DEVS and deep learning techniques. The framework combined the strengths of Cell-DEVS for detailed spatial-temporal modeling with the predictive power of deep learning to forecast CO_2 levels under various ventilation scenarios. The approach was validated through simulations that demonstrate its effectiveness in predicting CO_2 concentrations, offering valuable insights for improving indoor air quality in classrooms.

Various works in the area of *medical simulation* have been presented. Some of these articles focus on the logistic aspects of patients and clinics. Pérez et al.[109] discuss nuclear medicine patient service management, including scheduling patients, radiopharmaceuticals, and other resources, considering patient preferences and resource constraints. The model, validated using historical data from a real clinic, provides insights into optimizing patient throughput, resource utilization, and patient satisfaction. Various scheduling algorithms were tested, demonstrating the model's effectiveness in improving health care service management. DEVS was used to derive a generic simulation model for nuclear medicine patient service management that takes both patient and management perspectives. In the work by Alvarado et al.,[110] a DEVS-based framework is presented for modeling and simulating the operations of oncology clinics to improve patient flow and resource utilization. The model captures various processes such as patient registration, consultation, treatment, and follow-up, enabling the analysis of clinic performance under different scenarios. The simulation results provide insights into optimizing scheduling and resource allocation to reduce patient wait times and enhance service quality. Djitog et al.[111] introduce a multi-paradigm modeling framework combining DEVS, agent-based, and system dynamics approaches to simulate health care systems. The framework supports holistic analysis by integrating different modeling paradigms to capture various aspects of health care delivery, such as patient behavior, resource allocation, and system dynamics. This approach is demonstrated through case studies on hospital emergency departments and chronic disease management. Likewise, paper proposed by Traoré et al.[112] proposes a M&S framework for value-based health care systems using DEVS and agent-based modeling. The framework aims to optimize health care delivery by evaluating different value-based care strategies and their impact on patient outcomes and system costs. The simulation results offer insights into improving care coordination, resource allocation, and patient satisfaction in value-based health care models.

Various models built in DEVS dealt with the spread of diseases. Özmen et al.[113] investigated how different modeling choices and assumptions impact the outcomes of compartmental epidemiological models, particularly in the context of the 1918 influenza pandemic. By comparing differential equation-based models and agent-based models, the authors highlighted significant differences in disease spread dynamics based on factors such as model trajectories, temporal resolution, and the nature of interactions between agents. The study emphasizes the importance of documenting initial modeling assumptions to ensure the reliability and reproducibility of epidemiological models. DEVS is used to create event-based ABMs. In particular, the COVID-19 pandemic emphasized the need for such models, and in Davidson et al.,[114] we proposed a geographical Cell-DEVS approach to model and simulate the spatial spread of infectious diseases. The methodology incorporates geographical information and mobility patterns to create a realistic representation of disease dynamics. The model was validated through simulations of COVID-19 spread in urban areas, highlighting the importance of geographic factors in disease transmission. Similarly, in the work by Ayadi et al.,[115] a hybrid approach combining DEVS simulation with ontological modeling was defined to analyze the hierarchical spread and control of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The integrated framework enabled detailed modeling of virus transmission dynamics and the evaluation of intervention strategies. Case studies demonstrate the effectiveness of the approach in simulating the impact of social distancing and vaccination on infection rates.

Other biomedical simulations are focused on low-level phenomena. Watanabe et al.,[116] present a model of reproducible disease using the systems biology markup language (SBML) to ensure consistency and transparency in biomedical simulations. The paper emphasizes the need for collaborative efforts to improve model reproducibility and reliability in the biomedical research community. While the primary focus is on SBML, the paper acknowledges the role of DEVS in modeling discrete-event systems and its potential integration with SBML to enhance the reproducibility and scalability of disease models. Ozmen et al.[117] defined a tissue-scale agent-based model to simulate the premalignant progression of Barrett's esophagus toward esophageal adenocarcinoma. The model uses DEVS to handle the discrete-event nature of cellular changes and proliferation within the esophageal tissue. The simulation provides insights into the spatial dynamics of dysplasia and cancer development, aiding in the optimization of screening protocols.

Our team developed a few *spatial models* using the Cell-DEVS formalism; Wainer and Fernández[118] provided an in-depth exploration of the application of Cell-DEVS for modeling and simulating complex cellular automata. We showed how Cell-DEVS enhances the modeling process by introducing timing delays and simplifying the definition of complex timing behaviors. The paper highlighted the use of the CD++ tool to create executable models of cellular automata, illustrating the methodology with examples such as a three-state two-color Turing machine and a self-reproducing cellular automaton. Similarly, Wainer and Giambiasi[119] showed how to use Cell-DEVS to build asynchronous cell spaces with explicit timing delays. The paper focused on modeling the electrical behavior of heart tissue and traditionally analyzed using differential equations and cellular automata. The integration of GDEVS with Cell-DEVS provided precise results at a fraction of the computational cost, offering a more efficient and scalable approach for complex system analysis. Also, Wainer[120] explored the use of Cell-DEVS for environmental modeling. The paper highlights the application of Cell-DEVS in modeling different environmental scenarios, demonstrating the advantages of this formalism in simplifying the paradigm shift from discrete-time to discrete-event-based modeling. Kazi and Wainer[121] introduced an integrated framework for modeling and simulating ecosystems using the Cell-DEVS formalism, coupled with web-based simulation and geographic information systems (GIS). This framework automates data extraction from GIS, employs the CD++ cellular modeling tool for simulation, and integrates results with Google Earth for visualization.

Two particular types of spatial models have drawn attraction from the DEVS community. One of them is *crowd modeling and pedestrian flow*. A method that enables pedestrians to make movement decisions based on significant changes in their spatial position rather than at fixed time intervals was defined by Qiu and Hu.[122] The research shows that this approach improves simulation efficiency. By applying this approach, the study demonstrates more efficient simulations, particularly for heterogeneous pedestrian crowds, compared with conventional time-based models. The results show that the spatial activity-based model can effectively capture crowd dynamics while reducing computational costs. The paper in the work by Bae et al.[123] employed a multi-agent system to model individual behaviors and interactions, considering factors such as road networks, shelters, and transportation modes. The results highlight the importance of coordinated evacuation plans, the impact of different evacuation strategies on traffic congestion, and the critical role of RT information dissemination in improving evacuation efficiency and reducing casualties. The authors make use of the large-scale, distributed, extensible, and flexible (LDEF) formalism, an extension of DEVS that supports incremental and flexible modeling of multi-agent systems. Al-Habashna and Wainer[124] introduced a method for M&S of pedestrian behavior using Cell-DEVS formalism. It presents various examples of entity-based crowd models in one, two, and three dimensions, extending to complex scenarios such as multi-level building evacuations. The models are validated through simulations that demonstrate their efficiency and accuracy in predicting pedestrian movement patterns, particularly in emergency evacuation scenarios. Similarly, Jafer and Lawler[125] built a framework for modeling and simulating emergency crowd evacuation scenarios using the Cell-DEVS formalism. The authors developed 12 egress models representing various human behaviors and authority activities during emergencies. The framework aims to provide a risk-free, economical method for practicing evacuation strategies, training first responders, and aiding decision-making in RT emergency situations.

The second type of spatial model that achieved popularity is those related to the M&S of forest fires, an important application these days. Ntaimo et al.,[126] modeled forest fire spread and suppression using a spatial DEVS approach, modeling Rothermel's stationary model. The model leverages DEVSJAVA to simulate dynamic conditions such as changing weather patterns and firefighting efforts. The simulation helps to predict forest fire behavior, enabling tactical decision-making for fire control and suppression. Lately, Muzy et al.[127] built and validated a DES model for fire spread using DEVS and Cell-DEVS. The research addresses the complexity of accurately modeling fire behavior, and the computational challenges involved in RT simulation. DEVS and Cell-DEVS were utilized to create modular, hierarchical models that support complex timing behaviors and efficient simulation. The study involved comparing simulation results with controlled laboratory experiments, demonstrating the effectiveness of these formalisms in improving model accuracy and computational efficiency for fire spread analysis. The paper in the work by Ntaimo et al.[128] focuses on the definition of DEVS-FIRE, a DEVS-based integrated simulation model for predicting wildfire spread and containment. The model uses a cellular space approach combined with agent-based models for fire suppression, incorporating real spatial data for fuels, terrain, and weather. The dynamic structure implementation of DEVS-FIRE improves simulation performance and allows for RT decision support in wildfire management. Preliminary experiments validate the model's effectiveness in simulating wildfire behavior using high-resolution GIS data. This was expanded by Hu et al.[129] where a cellular space model for fire spread and agent-based models for fire suppression were presented. The study introduces a new ignition event scheduling method, which significantly improved simulation performance compared with the original method. The model is validated using real and artificially generated data, demonstrating its utility and efficiency in wildfire simulation and suppression planning. In the work by Filippi et al.,[130] a new method, formalism, and software for the simulation of interface dynamics were defined. The method (called

Vector-DEVS) focuses on front-tracking without splitting the space into discrete nodes, allowing high-resolution simulations on large scales. By coupling DEVS with a physical fire rate of spread model, the study achieves efficient simulation of fire dynamics, demonstrating the method's capability to perform large-scale simulations with high resolution on standard personal computers in a short time. Gu[131] introduced a novel method for data assimilation in wildfire spread simulations using a localized recursive spatial-temporal state quantification (LRSSQ) approach. The LRSSQ method improves the state inference of wildfire simulations by measuring particle convergence at runtime and adaptively adjusting the particle filter to enhance accuracy and computational efficiency. The LRSSQ method leverages the DEVS-FIRE model to assimilate RT observation data, enhancing the accuracy and performance of wildfire simulations.

The section emphasized the wide-ranging applications of DEVS across various domains, ranging from patient to networks and hardware design, including social models and urban planning. This shows the interdisciplinary nature of DEVS and the potential collaboration among researchers.

6. Current trends

Besides more traditional applications, DEVS is now widely applied across several technological domains, notably in machine learning, the IoT, and digital twins. As discussed in the following paragraphs, DEVS also offers a structured framework for modeling complex systems, enabling to create detailed simulations, generating synthetic data vital for enhancing predictive analytics and decision-making using machine learning. This framework allows for the exploration of various scenarios without the limitations of real-world data collection, producing customized data sets that improve algorithm performance and support thorough model validation. In the case of IoT, DEVS can be used to model the interactions between numerous connected devices, identifying performance bottlenecks, which is crucial for optimizing network architecture and functionality. DEVS has also been used to develop digital twins creating dynamic models for various scenarios, facilitating predictive maintenance and improved decision-making.

In particular, Choi and Kim[132] showed how to identify an unknown discrete-event system by recognizing characteristic functions of a DEVS model. The identification process consists of two major steps: behavior learning using a compound recurrent neural network (CRNN) and extraction of a DEVS model from the trained network. The CRNN network is specifically designed to learn the input/output behavior of an unknown DES and then extract the DEVS model that accurately represents this behavior. The method is validated through experiments with different types of DESs, demonstrating the effectiveness of this model extraction approach. Kim et al.[133] identified the limitations of data M&S modeling approaches and proposed a complementary cooperation modeling method that combines their strengths. The authors apply this method to a real-world greenhouse, using data modeling for the controller and simulation modeling for the plant and actuator. The results demonstrate improved control performance and stability, highlighting the effectiveness of the combined modeling approach in managing complex systems with big data. Kang et al.[134] proposed a method to reduce simulation time for analyzing combat effectiveness in network-centric warfare. The authors define a discrete-event dynamic surrogate model for communication systems, integrated with a command and control model. This model uses machine learning to identify unknown functions and reflects communication effects accurately while significantly reducing simulation execution time. DEVS is used to describe the dynamic surrogate model, ensuring the discrete-event nature of the simulation.

A formal framework for enhancing Digital Twin technology with self-updating capabilities, presented by Diakité and Traoré,[135] was used to organize various inference capabilities into a structured model and validated through a smart mobility system case study. The framework extends the DEVS formalism, adding the concepts of phase, semantic domain, activity. Niyonkuru and Wainer[136] presented the concept of Digital Quadruplets, which extends the Digital Twin model to include a formal discrete-event model (Digital Triplet) and a physical model (Digital Quadruplet) for RT embedded systems. This approach leverages the DEVS formalism to ensure model continuity and consistency across simulation, visualization, and RT execution. The paper focuses on using a DEVS kernel running on bare-metal hardware, avoiding the need for an operating system, thereby improving performance and predictability in embedded system design and implementation. In the next sections, we will discuss the current advances in our research related to this field: the use of DEVS for embedded RT applications.

6.1. Using RT cadmium for IoT and sensor fusion

As DEVS continues to showcase its versatility across emerging technological domains, its relevance within the IoT and embedded systems grows significantly. The modularity and hierarchical nature of the DEVS formalism make it especially adept at modeling the intricate interactions that characterize these systems, where physical processes and computational control are closely intertwined. A prime example of this can be seen in the definition of a methodology for Discrete-Event Modeling of Embedded Systems,[137] an M&S-based approach for software development in embedded RT systems. This expands upon

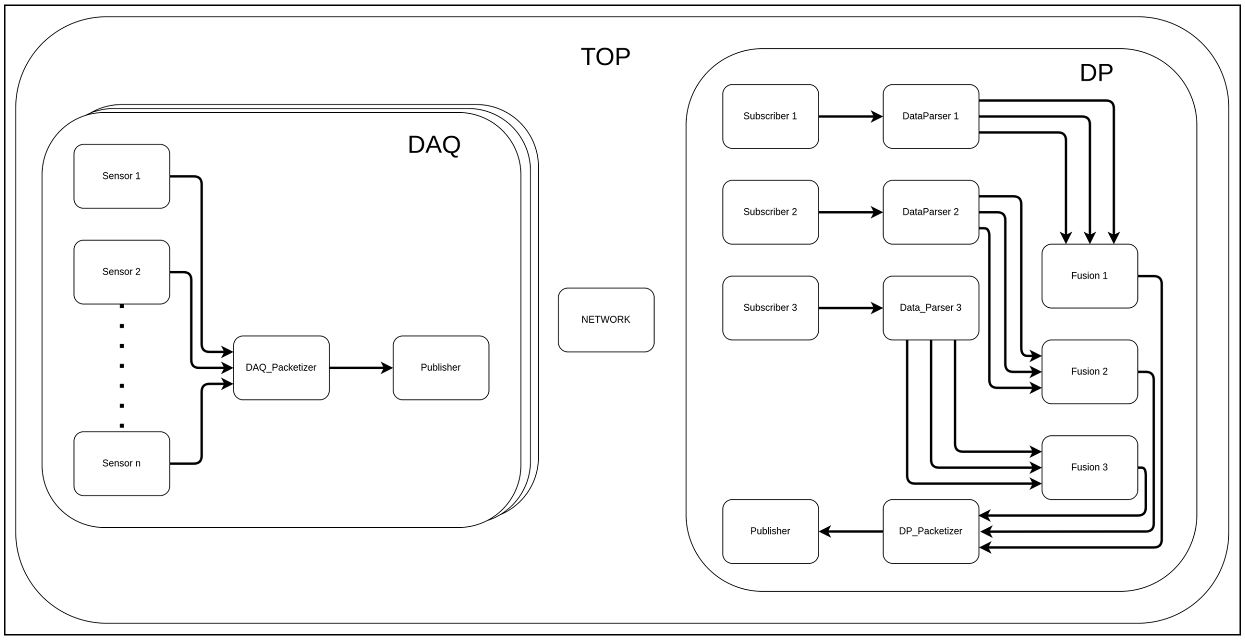

Figure 2. System Architecture Diagram.

DEVS to establish a cohesive modeling framework that spans the entire development cycle of embedded systems—from simulation through to RT implementation.

Concerning IoT, the integration of the open-source IBM-developed Node-RED[138] into IoT projects is a natural fit due to its ability to manage and orchestrate communication between diverse devices and services, making it an indispensable tool for creating complex, interconnected systems. Node-RED's capability to streamline the development process by abstracting the underlying complexity of IoT protocols and device communication allows developers to focus on building functional workflows without getting lost in the technical details. This ease of integration is crucial in the IoT landscape, where systems often involve multiple hardware platforms, sensors, and actuators that need to work together seamlessly. By leveraging Node-RED, developers can quickly prototype, test, and deploy IoT applications, ensuring that data flow efficiently between all components of the system.

Building on this foundation, the integration of DEVS with the Node-RED platform marks a leap forward in the field of IoT and embedded systems. By merging the modularity and timing precision of DEVS with the flexibility and user-friendliness of Node-RED, we have developed a method that enables interaction between real-world devices and DEVS-based simulations. This integration accelerates the development and deployment of IoT applications, creating a dynamic environment where DEVS models can engage with physical devices through Node-RED's intuitive interface. As a result, the applicability of DEVS in real-world scenarios is greatly enhanced. In addition, the project introduced a distributed data acquisition (DAQ) and processing system built on DEVS, specifically designed to simulate and evaluate system behavior under demanding conditions, thereby ensuring the system's robustness and reliability.

Figure 2 shows a block diagram of the software architecture proposed, which consists of three primary components: DAQ units, a network, and data processing (DP) units. The DAQ section includes multiple sensors, a DAQ_Packetizer, and a Publisher, all collaborating to gather data from the physical environment and publish it to the network. The network model facilitates the seamless transfer of data between the DAQ units and the DP units. Meanwhile, the DP unit—comprising Subscribers, DataParsers, Fusion blocks, a DP_Packetizer, and an additional Publisher—handles the processing of data collected by the various DAQ units. This organized structure ensures efficient data flow and processing throughout the system.

One of the widely adopted protocols in IoT is called Message Queueing Telemetry Transport (MQTT) protocol.[139] The development of the DEVS-MQTT framework was defined to enhance the capabilities of DEVS in IoT environments. By building on the foundational principles of DEVS, this integration facilitates efficient communication among the distributed components of a CPS. The framework plays a critical role in managing data flow and synchronization across a network of devices, which substantially improves the scalability of DEVS-based systems. The focus of this initiative was on ensuring secure and efficient data transmission within distributed networks, enabling DEVS models to operate effectively in RT IoT applications with minimal latency and overhead.

The distributed architecture defined in this research effectively segregates DAQ and processing across multiple microcontroller units (MCUs), utilizing the DEVS formalism to ensure synchronized operation. By distributing the responsibilities of DAQ and processing, the system can manage larger volumes of data and implement more complex processing algorithms without overloading any single component. This approach not only enhances the scalability and flexibility of the system but also allows for more efficient utilization of computational resources, ensuring that each component can operate independently while remaining aligned with the overarching system objectives.

While the integration of DEVS with Node-RED and MQTT brought significant benefits in terms of flexibility and scalability, the project also faced challenges, particularly in managing the synchronization and timing of distributed components. These challenges highlighted the need for further enhancements, leading to the development of an improved RT-clock mechanism, as explored by Govind et al.[140] In addition, the need for a robust communication protocol was addressed by Govind and Wainer[141] which introduced enhancements to the DEVS-based communication framework. Finally, the issues related to handling interrupts in RT simulations were tackled by Govind and Wainer[142] which proposed a novel approach to managing asynchronous events within the DEVS framework. These advancements collectively contributed to the creation of a more resilient and reliable DEVS-based system for RT IoT applications.

6.2. Managing clocks for RT applications

RT systems depend on precise timing mechanisms, making the selection of an appropriate clock crucial for optimal performance. Typically, simulations of DEVS models are conducted in virtual or simulation time,[143] where events are processed based on their scheduled timestamps rather than on real (wall-clock) time. This approach is highly effective for simulating complex systems, as it prioritizes the analysis of model performance over strict adherence to RT. However, when deploying DEVS models in RT platforms such as embedded systems or IoT environments, it becomes essential that the system aligns with RT requirements.

The RT-clock in cadmium enables RT synchronization between simulation events and hardware interactions, contrasting with the simulation clock that skips over time intervals. In DEVS, the wait time is defined by the *ta(s)* of the system's models. To help embedded systems with limited resources meet strict deadlines, the RT-clock features a "scheduler slip" mechanism that allows for slight adjustments in timing constraints. By specifying a value, modelers can accommodate hardware limitations without compromising system performance. This RT-clock uses standard C++ libraries, like *chrono*, to manage timing constraints across different platforms.[140] However, variations in library implementations have caused performance inconsistencies between simulation and execution. While effective for rapid testing, the generic RT-clock falls short in optimizing performance, prompting the development of platform-specific RT-clocks. The ESP32 was chosen for its IoT applicability. Previous designs that employed a hardware abstraction layer (HAL) resulted in performance degradation and increased binary sizes. To overcome these challenges, the authors created an improved RT-clock using the ESP32's native framework, enhancing overall performance.

The results by Govind et al.[140] show the performance improvements a platform-specific RT-clock would introduce. It significantly reduces scheduler slip by using dedicated hardware timers, which operate with much finer granularity than the software-based timers in the HAL implementation. The new clock mechanism also allows for more reliable timekeeping and deadline management, ensuring that the DEVS models remain in sync with the real-world hardware processes even under load. These enhancements are particularly valuable in applications like sensor fusion and RT DP, where timing precision directly impacts the system's performance and accuracy.

6.3. Defining communication protocols: DEVS-Inter Atomic Communication

In distributed IoT systems, reliable and predictable communication protocols are essential for swift responses to RT stimuli. While working on the DAQ-DP research, we faced challenges with the erratic performance of the communication protocol, which hindered scalability and resulted in inconsistent message delivery and synchronization issues among devices with varying processing abilities. As more devices were introduced, these problems worsened, leading to data loss and increased latency. Understanding the root causes of these issues, particularly in protocols like MQTT, is complex due to external factors such as Wi-Fi signal strength. Developing the capability to model and simulate the communication protocol end-to-end could help predict system behavior across different network configurations and facilitate the resolution of potential issues before deployment.

In this context, DEVS offers a comprehensive framework for modeling and simulating communication protocols by breaking them into modular atomic models that represent various components of the communication stack. This approach allows for the isolated testing and optimization of individual components and their interactions, tailored to specific system requirements. The DEVS-inter atomic communication (DEVS-IAC) protocol was developed using this method, enabling detailed simulations of distributed systems like the DAQ-DP.[141] This facilitated analysis of data management during network congestion and synchronization across geographically dispersed devices. The ability to model both normal and stressed conditions proved crucial for refining the protocol to meet the demands of RT distributed IoT systems.

We used the Open Systems Interconnection (OSI) model to create various layers of a communication protocol tailored for hard RT distributed systems. While traditional protocols like Fieldbus and CAN have been effective,[141] the rise of Ethernet as a cost-effective standard for businesses and educational institutions offers an opportunity to move away from outdated protocols like RS485.[144] However, since Ethernet was not initially designed for hard RT communication, modifications to its protocol stack are necessary. Pedreiras et al.[145] examined strategies to adapt Ethernet by altering its layers, enabling RT communication among distributed system nodes. By leveraging the Ethernet stack,[141] they developed the DEVS-IAC protocol, resulting in a reliable communication solution that meets the demands of IoT applications in varying network conditions with minimal delay and data loss.

Using DEVS for simulating a communication protocol allowed for a detailed comparison between predicted theoretical performance and actual real-world performance. The system was tested under various network conditions, including additive white Gaussian noise, to replicate real environmental disruptions. A case study, such as a traffic light system, was used to evaluate the protocol's performance in both RT and simulation contexts. The findings showed a significant alignment between DEVS models and physical implementations, with RT performance closely matching simulation predictions, even under stress and noise conditions. This correspondence underscores DEVS's effectiveness in modeling complex RT communication systems, paving the way for more robust and scalable IoT and embedded applications.

6.4. Adding interrupt-based external events in Cadmium

The implementation of the DEVS-IAC protocol on hardware needed specific configurations tailored to the platform for both data transmission and reception. Initially, the receiver node employed polling to capture incoming messages. While polling is a common method for monitoring inputs in RT systems, it has notable drawbacks, especially when handling asynchronous events and high data throughput. In the case of the DEVS-IAC protocol, the receiver's reliance on continuous polling for new data resulted in increased latency and inefficient resource usage. This approach required the system to continuously query the input status, which consumes processing power even during periods of inactivity. As the system scaled up and the frequency of events increased, polling became increasingly impractical, leading to greater delays and diminished overall system responsiveness. These challenges underscored the necessity for a more efficient, event-driven solution to effectively manage RT inputs. To deal with these limitations, we introduced an interrupt component (IC) designed to efficiently handle asynchronous inputs.[142] Unlike polling, which requires the system to continuously check for new data, interrupts allow the system to remain in a passive state until an external event occurs, at which point it is "interrupted" to process the input. This significantly reduces computational overhead, as the system is not burdened by constant checks. The IC in RT-cadmium effectively bridges the gap between real-world signals and the DEVS framework by converting physical inputs into DEVS-compatible messages, ensuring that external events are captured and processed with optimal efficiency. Polling involves continuous checking of an input, which can drain system resources and lead to delays, particularly when dealing with high-frequency inputs. Interrupts, however, allow the system to only respond when necessary, minimizing unnecessary processing. DEVS inherently supports interrupt-driven behavior through its event-based architecture, making it a natural fit for integrating interrupts within the simulation environment. By utilizing interrupts, systems can prioritize critical tasks without sacrificing performance, especially under high load conditions, which is essential for maintaining the responsiveness and timing constraints in RT systems.

As mentioned earlier, DEVS is well-suited for the efficient handling of asynchronous inputs. In RT-cadmium, the IC was developed to enhance the abstract simulation algorithm, facilitating the seamless integration of external interrupts into the simulation framework. The IC functions by intercepting physical signals, converting them into DEVS-compatible messages, and forwarding them to the root coordinator of the DEVS simulation. This ensures that external events are addressed in RT without disrupting the internal logic of the DEVS models. The case study on the video surveillance system illustrates how the IC enabled prompt responses to environmental stimuli while preserving the integrity of the DEVS formalism throughout both the simulation and execution phases.

The transition from a polling-based system to an interrupt-driven approach greatly improved the performance of the DEVS-IAC protocol. Initially, polling caused delays in input detection and processing, particularly under high system loads, leading to resource wastage and performance bottlenecks in distributed systems managing high-frequency asynchronous events. The integration of the IC allowed the receiver to respond instantly to incoming signals, ensuring that DEVS models stayed synchronized with RT inputs, even under stress. This shift reduced latency, optimized resource use, and significantly enhanced system scalability, as demonstrated in a case study involving efficient handling of asynchronous video capture and transmission in a high-frequency environment.

These advancements position DEVS as a critical tool for the development and optimization of IoT and CPS, where precise M&S are essential for ensuring system reliability and performance.

7. Conclusion

Modern hard RT distributed systems consist of decentralized, interconnected compute nodes that execute time-sensitive processes, which can be independent or interdependent. The performance of these systems is often hampered by the communication network, making the development of a robust, deterministic, and reliable communication protocol essential for meeting deadlines. Furthermore, traditional embedded system design typically involves independent hardware and software development, leading to disconnects and delays. Software teams often await hardware prototypes, while hardware developers rely on software for verification, which hinders collaboration and extends the development timeline. The conventional practice in the early stages of embedded system design often involves the independent development of hardware and software components.

M&S has been widely used to enhance the quality of RT applications while reducing lifecycle costs, primarily through improved testability and maintainability, through early hardware functionality testing, fosters collaboration between hardware and software. This early verification helps identify issues and accelerates innovation through iterative refinements. In particular, the DEVS formalism has been widely used in this area. DEVS provides a robust foundation for developing discrete-event M&S systems. It defines both atomic models, which encapsulate individual system component behavior, and coupled models, which represent complex interactions among multiple components. DEVS fosters a structured, hierarchical design approach that promotes component reuse and independent validation before integration. Its versatility has made it applicable in diverse fields like logistics, telecommunications, and health care, showcasing its capability to model dynamic systems accurately. DEVS also benefits from a clear separation between model definition and execution, allowing for conceptual design without technical execution complexities. Models can be implemented across different platforms, maintaining fidelity and correctness due to the formally verified DEVS simulation algorithm. This makes DEVS particularly suitable for safety-critical systems and large engineering projects. In this paper, we have shown the significant growth of publications on DEVS in the SIMULATION journal over the past two decades. Our journal has become a key platform for a diverse range of research in DEVS M&S, showcasing its effectiveness in addressing complex challenges in various fields such as engineering, computer science, health care, and environmental modeling. We have witnessed a substantial expansion in the theoretical and practical applications of DEVS, leading to improved modeling capabilities. The numerous publications in the field and the recent advances in RT embedded applications using DEVS show the critical role in advancing simulation methodologies and paving the way for future exploration. We look forward to seeing what the next 100 volumes will bring to this area of research.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ORCID iD

Gabriel Wainer https://orcid.org/0000-0003-3366-9184

References

1. Preuveneers D and Ilie-Zudor E. The intelligent industry of the future: a survey on emerging trends, research challenges and opportunities in Industry 4.0. *J Ambient Intel Smart Environ* 2017; 9: 287–298.
2. Zeigler BP. *Theory of modelling and simulation*. New York: John Wiley & Sons, 1976.
3. Vangheluwe H. DEVS as a common denominator formulti-formalism hybrid systems modelling. In: *IEEE international symposium on computer-aided control system design* (ed Varga A), Anchorage, AK, 25–27 September 2000, pp. 129–134. New York: IEEE.
4. Ören TI and Zeigler BP. System theoretic foundations of modeling and simulation: a historic perspective and the legacy of A Wayne Wymore. *SIMULATION* 2012; 88: 1033–1046.
5. Naamane A, Giambiasi N and Damiba A. Generalized discrete event simulation of bond graph. *SIMULATION* 2001; 77: 4–22.
6. Barros FJ. Modeling and simulation of dynamic structure heterogeneous flow systems. *SIMULATION* 2002; 78: 18–27.
7. Barros FJ. A formal representation of hybrid mobile components. *SIMULATION* 2005; 81: 381–393.
8. Barros FJ. Dynamic structure discrete event system specification formalism. *Trans Soc Comput Simul* 1996; 13: 35–46.
9. Hu X, Hu X, Zeigler BP, et al. Variable structure in DEVS component-based modeling and simulation. *SIMULATION* 2005; 81: 91–102.
10. Barros FJ. Defining hybrid hierarchical models in πHYFLOW. *SIMULATION* 2024; 100: 643–655.
11. Barros FJ. Modular representation of asynchronous geometric integrators with support for dynamic topology. *SIMULATION* 2018; 94: 259–274.

12. Saadawi H and Wainer G. Principles of discrete event system specification model verification. *SIMULATION* 2013; 89: 41–67.
13. Cicirelli F, Furfaro A and Nigro L. Using time stream Petri nets for workflow modelling analysis and enactment. *SIMULATION* 2013; 89: 68–86.
14. Fonseca i and Casas P. Transforming classic discrete event system specification models to specification and description language. *SIMULATION* 2015; 91: 249–264.
15. Giambiasi N. An introduction to timed sequential machines. *SIMULATION* 2014; 90: 337–352.
16. Mello BA and Wainer G. Integrating I-DEVS and schedulability methods for analyzing real-time systems constraints. *SIMULATION* 2022; 98: 1143–1159.
17. Hong S-Y and Kim TG. Specification of multi-resolution modeling space for multi-resolution system simulation. *SIMULATION* 2013; 89: 28–40.
18. Goldstein R, Khan A, Dalle O, et al. Multiscale representation of simulated time. *SIMULATION* 2018; 94: 519–558.
19. Castro R, Kofman E and Wainer G. A formal framework for stochastic discrete event system specification modeling and simulation. *SIMULATION* 2010; 86: 587–611.
20. Hu X and Zeigler BP. Linking information and energy—activity-based energy-aware information processing. *SIMULATION* 2013; 89: 435–450.
21. Castro R and Kofman E. Activity of order n in continuous systems. *SIMULATION* 2014; 91: 337–348.
22. Castro R, Bergonzi M, Pecker-Marcosig E, et al. Discrete-event simulation of continuous-time systems: evolution and state of the art of quantized state system methods. *SIMULATION* 2024; 100: 613–638.
23. Kofman E. Quantization-based simulation of differential algebraic equation systems. *SIMULATION* 2003; 79: 363–376.
24. Migoni G, Kofman E and Cellier F. Quantization-based new integration methods for stiff ordinary differential equations. *SIMULATION* 2012; 88: 387–407.
25. Bergero F and Kofman E. A vectorial DEVS extension for large scale system modeling and parallel simulation. *SIMULATION* 2014; 90: 522–546.
26. Fernández J and Kofman E. A stand-alone quantized state system solver for continuous system simulation. *SIMULATION* 2014; 90: 782–799.
27. Bergero F, Fernández J, Kofman E, et al. Time discretization versus state quantization in the simulation of a one-dimensional advection–diffusion–reaction equation. *SIMULATION* 2016; 92: 47–61.
28. Di Pietro F, Migoni G and Kofman E. Improving linearly implicit quantized state system methods. *SIMULATION* 2019; 95: 127–144.
29. Liu Q and Wainer G. Parallel environment for DEVS and Cell-DEVS Models. *SIMULATION* 2007; 83: 449–471.
30. Liu Q and Wainer G. Multicore acceleration of discrete event system specification systems. *SIMULATION* 2012; 88: 801–831.
31. Park S, Hunt CA and Zeigler BP. Cost-based partitioning for distributed and parallel simulation of decomposable multiscale constructive models. *SIMULATION* 2006; 82: 809–826.
32. Cardoen B, Manhaeve S, Van Tendeloo Y, et al. A PDEVS simulator supporting multiple synchronization protocols: implementation and performance analysis. *SIMULATION* 2018; 94: 281–300.
33. Adegoke A, Togo H and Traoré MK. A unifying framework for specifying DEVS parallel and distributed simulation architectures. *SIMULATION* 2013; 89: 1293–1309.
34. Bergero F, Kofman E and Cellier F. A novel parallelization technique for DEVS simulation of continuous and hybrid systems. *SIMULATION* 2013; 89: 663–683.
35. Nutaro J and Ozmen O. Race conditions and data partitioning: risks posed by common errors to reproducible parallel simulations. *SIMULATION* 2023; 99: 417–427.
36. Lee EA and John II. Overview of the ptolemy project, 1999, https://ptolemy.berkeley.edu/index.htm (accessed October 2024).
37. Tolk A. Conceptual alignment for simulation interoperability: lessons learned from 30 years of interoperability research. *SIMULATION* 2024; 100: 709–726.
38. Silver GA, Miller JA, Hybinette M, et al. DeMO: an ontology for discrete-event modeling and simulation. *SIMULATION* 2011; 87: 747–773.
39. Nutaro J and Sarjoughian H. Design of distributed simulation environments: a unified system-theoretic and logical processes approach. *SIMULATION* 2004; 80: 577–589.
40. Wutzler T and Sarjoughian HS. Interoperability among parallel DEVS simulators and models implemented in multiple programming languages. *SIMULATION* 2007; 83: 473–490.
41. Zacharewicz G, Frydman C and Giambiasi N. G-DEVS/HLA environment for distributed simulations of workflows. *SIMULATION* 2008; 84: 197–213.
42. Boukerche A, Zhang M and Shadid A. DEVS approach to real-time RTI design for large-scale distributed simulation systems. *SIMULATION* 2008; 84: 231–238.
43. Mittal S, Risco-Martín JL and Zeigler BP. DEVS/SOA: a cross-platform framework for net-centric modeling and simulation in DEVS unified process. *SIMULATION* 2009; 85: 419–450.
44. Wang W, Wang W, Zhu Y, et al. Service-oriented simulation framework: an overview and unifying methodology. *SIMULATION* 2011; 87: 221–252.
45. Wang S and Wainer G. A simulation as a service methodology with application for crowd modeling, simulation and visualization. *SIMULATION* 2015; 91: 71–95.
46. Sun F, Zhou J, Guo S, et al. Flexible model specification and application for service-oriented software. *SIMULATION* 2019; 95: 363–381.
47. Mosterman PJ and Vangheluwe H. Computer automated multi-paradigm modeling: an introduction. *SIMULATION* 2004; 80: 433–450.
48. Hardebolle C and Boulanger F. Exploring multi-paradigm modeling techniques. *SIMULATION* 2009; 85: 688–708.
49. Denil J, De Meulenaere P, Demeyer S, et al. DEVS for AUTOSAR-based system deployment modeling and simulation. *SIMULATION* 2017; 93: 489–513.
50. Risco-Martín JL, de la, Cruz JM, Mittal S, et al. EUDEVS: executable UML with DEVS theory of modeling and simulation. *SIMULATION* 2009; 85: 750–777.
51. Gianni D, D'Ambrogio A and Iazeolla G. A software architecture to ease the development of distributed simulation systems. *SIMULATION* 2011; 87: 819–836.

52. Schmidt A, Durak U and Pawletta T. Model-based testing methodology using system entity structures for MATLAB/Simulink models. *SIMULATION* 2016; 92: 729–746.
53. Santucci JF, Capocchi L and Zeigler BP. System entity structure extension to integrate abstraction hierarchies and time granularity into DEVS modeling and simulation. *SIMULATION* 2016; 92: 747–769.
54. Sanz V and Urquia A. Combining PDEVS and Modelica for describing agent-based models. *SIMULATION* 2023; 99: 455–474.
55. Kim J-H and Kim TG. DEVS-based framework for modeling/simulation of mobile agent systems. *SIMULATION* 2001; 76: 345–357.
56. Peng D, Warnke T, Haack F, et al. Reusing simulation experiment specifications in developing models by successive composition — a case study of the Wnt/β-catenin signaling pathway. *SIMULATION* 2017; 93: 659–677.
57. Yilmaz L. Verifying collaborative behavior in component-based DEVS models. *SIMULATION* 2004; 80: 399–415.
58. Yacoub A, Hamri MEA and Frydman C. DEv-PROMELA: modeling, verification, and validation of a video game by combining model-checking and simulation. *SIMULATION* 2020; 96: 881–910.
59. Samuel KG, Bouare N-DM, Maïga O, et al. A DEVS-based pivotal modeling formalism and its verification and validation framework. *SIMULATION* 2020; 96: 969–992.
60. Wainer G and Giambiasi N. Application of the cell-DEVS paradigm for cell spaces modelling and simulation. *SIMULATION* 2001; 76: 22–39.
61. Sun Y and Hu X. Performance measurement of dynamic structure DEVS for large-scale cellular space models. *SIMULATION* 2009; 85: 335–351.
62. Song HS and Kim TG. Application of real-time DEVS to analysis of safety-critical embedded control systems: railroad crossing control example. *SIMULATION* 2005; 81: 119–136.
63. Sarjoughian HS, Hild DR, Hu X, et al. Simulation-based SW/HW architectural design configurations for distributed mission training systems. *SIMULATION* 2001; 77: 23–38.
64. Cao Q. Research on co-simulation of multi-resolution models based on HLA. *SIMULATION* 2023; 99: 515–535.
65. Camus B, Paris T, Vaubourg J, et al. Co-simulation of cyber-physical systems using a DEVS wrapping strategy in the MECSYCO middleware. *SIMULATION* 2018; 94: 1099–1127.
66. Pecker-Marcosig E, Giribet JI and Castro R. Hybrid resource allocation control in cyber-physical systems: a novel simulation-driven methodology with applications to UAVs. *SIMULATION* (Submitted).
67. Kapos G-D, Dalakas V, Nikolaidou M, et al. An integrated framework for automated simulation of SysML models using DEVS. *SIMULATION* 2014; 90: 717–744.
68. Sarjoughian HS and Gholami S. Action-level real-time DEVS modeling and simulation. *SIMULATION* 2015; 91: 869–887.
69. Lee JS, Zeigler BP and Venkatesan SM. Design and development of data distribution management environment. *SIMULATION* 2001; 77: 39–52.
70. Kim YJ, Kim JH and Kim TG. Heterogeneous simulation framework using DEVS BUS. *SIMULATION* 2003; 79: 3–18.
71. Choi B-K, Park B-C and Park J-H. A formal model conversion approach to developing a DEVS-based factory simulator. *SIMULATION* 2003; 79: 440–461.
72. Nutaro J, Kuruganti PT, Protopopescu V, et al. The split system approach to managing time in simulations of hybrid systems having continuous and discrete event components. *SIMULATION* 2012; 88: 281–298.
73. Gonzalez FG and Davis WJ. A new simulation tool for the modeling and control of distributed systems. *SIMULATION* 2002; 78: 552–567.
74. Cho YK, Hu X and Zeigler BP. The RTDEVS/CORBA environment for simulation-based design of distributed real-time systems. *SIMULATION* 2003; 79: 197–210.
75. Lee J-K, Lee M-W and Chi S-D. DEVS/HLA-based modeling and simulation for intelligent transportation systems. *SIMULATION* 2003; 79: 423–439.
76. Lee J-K, Lim Y-H and Chi S-D. Hierarchical modeling and simulation environment for intelligent transportation systems. *SIMULATION* 2004; 80: 61–76.
77. Wainer G and Liu Q. Tools for graphical specification and visualization of DEVS models. *SIMULATION* 2009; 85: 131–158.
78. Bonaventura M, Wainer GA and Castro R. Graphical modeling and simulation of discrete-event systems with CD++Builder. *SIMULATION* 2013; 89: 4–27.
79. Bergero F and Kofman E. PowerDEVS: a tool for hybrid system modeling and real-time simulation. *SIMULATION* 2011; 87: 113–132.
80. Nutaro J. An extension of the OpenModelica compiler for using Modelica models in a discrete event simulation. *SIMULATION* 2014; 90: 1328–1345.
81. Stolpe A, Rummelhoff I and Hannay JE. A logic-based event controller for means-end reasoning in simulation environments. *SIMULATION* 2023; 99: 831–858.
82. Van Tendeloo Y and Vangheluwe H. Increasing the performance of a Discrete Event System Specification simulator by means of computational resource usage "activity" models. *SIMULATION* 2017; 93: 1045–1061.
83. Van Mierlo S, Van Tendeloo Y and Vangheluwe H. Debugging parallel DEVS. *SIMULATION* 2017; 93: 285–306.
84. Maleki M, Woodbury R, Goldstein R, et al. Designing DEVS visual interfaces for end-user programmers. *SIMULATION* 2015; 91: 715–734.
85. Goldstein R, Breslav S and Khan A. Practical aspects of the DesignDEVS simulation environment. *SIMULATION* 2018; 94: 301–326.
86. Van Tendeloo Y and Vangheluwe H. An evaluation of DEVS simulation tools. *SIMULATION* 2017; 93: 103–121.
87. Wainer G, Glinsky E and Gutierrez-Alcaraz M. Studying performance of DEVS modeling and simulation environments using the DEVStone benchmark. *SIMULATION* 2011; 87: 555–580.
88. Risco-Martín JL, Mittal S, Jiménez JCF, et al. Reconsidering the performance of DEVS modeling and simulation environments using the DEVStone benchmark. *SIMULATION* 2017; 93: 459–476.
89. Huang Y, Seck MD and Verbraeck A. Component-based light-rail modeling in discrete event systems specification (DEVS). *SIMULATION* 2015; 91: 1027–1051.

90. Grinblat GL, Ahumada H and Kofman E. Quantized state simulation of spiking neural networks. *SIMULATION* 2012; 88: 299–313.
91. Xie X and Verbraeck A. A particle filter-based data assimilation framework for discrete event simulations. *SIMULATION* 2019; 95: 1027–1053.
92. Huang Y, Xie X, Cho Y, et al. Particle filter–based data assimilation in dynamic data-driven simulation: sensitivity analysis of three critical experimental conditions. *SIMULATION* 2023; 99: 403–415.
93. Ghosh S and Giambiasi N. Breakthrough in modeling and simulation of mixed-signal electronic designs in nVHDL. *SIMULATION* 2001; 76: 279–281.
94. Hamri M, Naamane A, Frydman C, et al. Why we should use Min Max DEVS for modeling and simulation of digital circuits. *SIMULATION* 2022; 98: 519–532.
95. Yu Chen Sarjoughian HS. A component-based simulator for MIPS32 processors. *SIMULATION* 2010; 86: 271–290.
96. Migoni G, Kofman E, Bergero F, et al. Quantization-based simulation of switched mode power supplies. *SIMULATION* 2015; 91: 320–336.
97. Antoine-Santoni T, Santucci JF, De Gentili E, et al. Discrete event modeling and simulation of wireless sensor network performance. *SIMULATION* 2008; 84: 103–121.
98. Pfeifer D, Valvano J and Gerstlauer A. SimConnect and SimTalk for distributed cyber-physical system simulation. *SIMULATION* 2013; 89: 1254–1271.
99. Pang CK and Mathew J. Dynamically reconfigurable command and control structure for network-centric warfare. *SIMULATION* 2015; 91: 417–431.
100. Kim J, Moon I-C and Kim TG. New insight into doctrine via simulation interoperation of heterogeneous levels of models in battle experimentation. *SIMULATION* 2012; 88: 649–667.
101. Seo K-M, Choi C, Kim TG, et al. DEVS-based combat modeling for engagement-level simulation. *SIMULATION* 2014; 90: 759–781.
102. Fard MD and Sarjoughian HS. A knowledge interchange broker composition modeling framework for simulating water, energy, and water-energy nexus systems. *SIMULATION* 2024; 100: 657–682.
103. Risco-Martín JL, Esteban S, Chacón J, et al. Simulation-driven engineering for the management of harmful algal and cyanobacterial blooms. *SIMULATION* 2023; 99: 1041–1055.
104. Byon E, Pérez E, Ding Y, et al. Simulation of wind farm operations and maintenance using discrete event system specification. *SIMULATION* 2011; 87: 1093–1117.
105. Pérez E, Ntaimo L and Ding Y. Multi-component wind turbine modeling and simulation for wind farm operations and maintenance. *SIMULATION* 2015; 91: 360–382.
106. Pérez E. A simulation-driven online scheduling algorithm for the maintenance and operation of wind farm systems. *SIMULATION* 2022; 98: 47–61.
107. Bouanan Y, Zacharewicz G, Ribault J, et al. Discrete event system specification-based framework for modeling and simulation of propagation phenomena in social networks: application to the information spreading in a multi-layer social network. *SIMULATION* 2019; 95: 411–427.
108. Khalil H and Wainer G. A framework for modeling, generating, simulating, and predicting carbon dioxide dispersion indoors using cell-DEVS and deep learning. *SIMULATION* 2024; 100: 421–434.
109. Pérez E, Ntaimo L, Bailey C, et al. Modeling and simulation of nuclear medicine patient service management in DEVS. *SIMULATION* 2010; 86: 481–501.
110. Alvarado MM, Cotton TG, Ntaimo L, et al. Modeling and simulation of oncology clinic operations in discrete event system specification. *SIMULATION* 2018; 94: 105–121.
111. Djitog I, Aliyu HO and Traoré MK. A model-driven framework for multi-paradigm modeling and holistic simulation of healthcare systems. *SIMULATION* 2018; 94: 235–257.
112. Traoré MK, Zacharewicz G, Duboz R, et al. Modeling and simulation framework for value-based healthcare systems. *SIMULATION* 2019; 95: 481–497.
113. Özmen Ö, Nutaro JJ, Pullum LL, et al. Analyzing the impact of modeling choices and assumptions in compartmental epidemiological models. *SIMULATION* 2016; 92: 459–472.
114. Davidson G, Fahlman A, Mereu E, et al. A methodological approach for modeling the spread of disease using geographical discrete-event spatial models. *SIMULATION* 2024; 100: 39–70.
115. Ayadi A, Frydman C, Laddada W, et al. Combining DEVS simulation and ontological modeling for hierarchical analysis of the SARS-CoV-2 replication. *SIMULATION* 2023; 99: 1011–1039.
116. Watanabe L, Barhak J and Myers C. Toward reproducible disease models using the systems biology markup language. *SIMULATION* 2019; 95: 895–930.
117. Ozmen O, Nutaro J, Ostvar S, et al. Tissue scale agent-based simulation of premalignant progressions in Barrett's esophagus. *SIMULATION* 2022; 98: 275–284.
118. Wainer G and Fernández J. Modelling and simulation of complex cellular models using Cell-DEVS. *SIMULATION* 2016; 92: 101–115.
119. Wainer G and Giambiasi N. Cell-DEVS/GDEVS for complex continuous systems. *SIMULATION* 2005; 81: 137–151.
120. Wainer G. Applying cell-DEVS methodology for modeling the environment. *SIMULATION* 2006; 82: 635–660.
121. Kazi BU and Wainer G. Integrated cellular framework for modeling ecosystems: theory and applications. *SIMULATION* 2018; 94: 213–233.
122. Qiu F and Hu X. Spatial activity-based modeling for pedestrian crowd simulation. *SIMULATION* 2013; 89: 451–465.
123. Bae JW, Lee S, Hong JH, et al. Simulation-based analyses of an evacuation from a metropolis during a bombardment. *SIMULATION* 2014; 90: 1244–1267.
124. Al-Habashna A and Wainer G. Modeling pedestrian behavior with Cell-DEVS: theory and applications. *SIMULATION* 2016; 92: 117–139.
125. Jafer S and Lawler R. Emergency crowd evacuation modeling and simulation framework with cellular discrete event systems. *SIMULATION* 2016; 92: 795–817.
126. Ntaimo L, Zeigler BP, Vasconcelos MJ, et al. Forest fire spread and suppression in DEVS. *SIMULATION* 2004; 80: 479–500.

127. Muzy A, Innocenti E, Aiello A, et al. Specification of discrete event models for fire spreading. *SIMULATION* 2005; 81: 103–117.
128. Ntaimo L, Hu X and Sun Y. DEVS-FIRE: towards an integrated simulation environment for surface wildfire spread and containment. *SIMULATION* 2008; 84: 137–155.
129. Hu X, Sun Y and Ntaimo L. DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models. *SIMULATION* 2012; 88: 259–279.
130. Filippi J-B, Morandini F, Balbi JH, et al. Discrete event front-tracking simulation of a physical fire-spread model. *SIMULATION* 2010; 86: 629–646.
131. Gu F. Localized recursive spatial-temporal state quantification method for data assimilation of wildfire spread simulation. *SIMULATION* 2017; 93: 343–360.
132. Choi SJ and Kim TG. Identification of discrete event systems using the compound recurrent neural network: extracting DEVS from trained network. *SIMULATION* 2002; 78: 90–104.
133. Kim BS, Kang BG, Choi SH, et al. Data modeling versus simulation modeling in the big data era: case study of a greenhouse control system. *SIMULATION* 2017; 93: 579–594.
134. Kang BG, Seo K-M and Kim TG. Machine learning-based discrete event dynamic surrogate model of communication systems for simulating the command, control, and communication system of systems. *SIMULATION* 2019; 95: 673–691.
135. Diakité M and Traoré MK. Formalizing a framework of inference capabilities for digital twin engineering. *SIMULATION* 2024; 100: 887–902.
136. Niyonkuru D and Wainer G. A DEVS-based engine for building digital quadruplets. *SIMULATION* 2021; 97: 485–506.
137. Wainer G. Applying modelling and simulation for development embedded systems. In: *2013 2nd Mediterranean conference on embedded computing (MECO)*, Budva, 15–20 June 2013.
138. Ferencz K and Domokos J. Using node-RED platform in an industrial environment. In: *XXXV Jubileumi Kandó Konferencia proceedings*, Budapest, 14–15 November 2019, pp. 52–63.
139. Soni D and Makwana A. A survey on MQTT: a protocol of internet of things (IoT). In: *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, Chennai, India, 6–8 April 2017, Vol. 20, pp. 173–177. Hubei, China: Aconf.org.
140. Govind SM, Alex JS and Wainer G. Adapting the DEVS kernel "RT-CADMIUM" to the ESP32 embedded platform. *arXiv*, 2023, https://arxiv.org/abs/2304.07961
141. Govind S and Wainer G. DEVS based robust communication protocol for inter-simulation communication in Cadmium. In: *ANNSIM 2024 proceedings*, Washington, DC, 20–23 May.
142. Govind S and Wainer G. Handling asynchronous inputs in DEVS based real-time kernels. In: *Proceedings of the 2024 Winter simulation conference*, Orlando, FL, 15–18 December.
143. Chow AC, Zeigler BP and Kim DH. Abstract simulator for the parallel DEVS formalism. In: *Fifth annual conference on AI, and planning in high autonomy systems*, Gainesville, FL, 13–15 December 1994, pp. 157–163.
144. Neumann P. Communication in industrial automation—what is going on? *Control Eng Pract* 2007; 15: 1332–1347.
145. Pedreiras P, Almeida L and Fonseca JA. The quest for real-time behavior in Ethernet. In: Zurawski R (ed.) *The Industrial Information Technology Handbook*. Boca Raton, FL: CRC Press, 2005, pp. 1–14.

Author biographies

Gabriel Wainer received his PhD degree (highest honors) from Université d'Aix-Marseille III, Marseille, France. He is currently a Full Professor with Carleton University, Ottawa, ON, Canada, where he is also the Head of the Advanced Real-Time Simulation Laboratory, Centre for advanced Simulation and Visualization (V-Sim). He held visiting positions at the University of Arizona; LSIS (CNRS), Université Paul Cézanne, University of Nice, INRIA Sophia-Antipolis, Université de Bordeaux (France); UCM, UPC (Spain), University of Buenos Aires, National University of Rosario (Argentina), and others. He is editor in chief of SIMULATION, Transactions of the SCS. He is also a member of the Editorial Board of the IEEE Computing in Science & Engineering, Wireless Networks (Elsevier), and The Journal of Defense Modeling and Simulation (SCS). He is a member of the Board of Directors of the SCS. He was a recipient of various awards, including various best papers, the SCS McLeod Founder Award for distinguished service to the profession, the ACM Recognition of Service Award, and the IEEE Outstanding Engineering Award (Ottawa Section). He is a fellow of SCS. His email address is gwainer@sce.carleton.ca

Sasisekhar Govind is a PhD candidate at Carleton University under the supervision of Dr. Gabriel Wainer. He holds a bachelor's degree in Electronics and Communications Engineering from VIT, India. His research interests lie in embedded systems and distributed simulations. His email address is sasisekharmangalamgo@cunet.carleton.ca