Discrete Event Modeling and Simulation Approaches for IIoT

GHENA BARAKAT[1], LUCA D'AGATI[1], GIUSEPPE TRICOMI[1,2],
FRANCESCO LONGO[1], ANTONIO PULIAFITO[1], GIOVANNI MERLINO[1]
[1]Università di Messina, Dipartimento di Ingegneria, Messina, ITALY
[2]ICAR-CNR: Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche Italiano, Napoli, ITALY

Abstract: The industry has experienced significant advancements in recent years, primarily focusing on smart manufacturing, culminating in the Industry 4.0 (I4.0) revolution I4.0 emphasizes interconnectivity, real time data capture and transmission among machines, autonomy, and machine learning, providing manufacturing companies numerous growth opportunities. The Industrial Internet of Things (IIoT) is a core component of this revolution, becoming integral to each system and increasing complexity due to the vast number of interconnected devices and diverse physical components. The variety of virtual services distributed across the architectural layers of industrial systems (cloud, fog, edge) and the various connection types between IIoT devices introduce security and privacy challenges, which are critical issues for any system incorporating IIoT. To fully leverage IIoT's potential, addressing these security and privacy concerns is essential. Research and design in this domain are challenging, particularly when creating a simulation environment to study a system's behavior over time. Despite the extensive research in IoT and the significant benefits of simulation based approaches, there remains a challenge in creating detailed representations from the underlying IoT nodes to the application layer in the cloud, along with the underlying networking infrastructure. To assist researchers and practitioners in overcoming these challenges, we propose the Discrete Event System Specification (DEVS) formalism. DEVS provides a mathematical framework for modeling systems, whether discrete or continuous events, allowing for the simulation of these systems within the DEVS environment. Every system, whether real or conceptual, has a time base, inputs, outputs, and functions to determine the next state, as well as outputs that reflect the current state and inputs. Simulating the system within the DEVS environment allows one to study its behavior to predict and optimize performance patterns.

Key-Words: DEVS, Industrial IoT, Industry 4.0, Operational Technology

Received: June 22, 2024. Revised: November 2, 2024. Accepted: November 24, 2024. Published: December 6, 2024.

1 Introduction

The rapid growth in the industrial sector to meet today's diverse and dynamic market demands has sparked a revolution known as Industry 4.0. This revolution focuses on integrating innovative technologies to achieve robust and efficient manufacturing from a cost perspective. A key technology driving this industrial revolution is the Internet of Things (IoT). The widespread adoption of IoT has transformed businesses into digital enterprises by leveraging data from sensors and devices. The significant increase in IoT devices impacts traditional Operational Technologies (OT), enhancing productivity and generating substantial revenue for organizations. However, despite the advantages of Industry 4.0, it encounters various risks, both in terms of security threats and challenges related to implementing complex systems in factories. From a security perspective, Industry 4.0 is vulnerable to threats such as industrial espionage, sabotage, unauthorized access, theft, digital attacks, and malicious disruptions. Ensuring robust cybersecurity measures is essential. On the implementation side, deploying complex systems and monitoring their behavior in real industrial environments may be challenging. Ensuring proper communication and coordination between various components may lead to inefficiencies, increased costs, and delays. Due to such reason, the proposed research work focuses on defining a simulator enabling the designing team of OT systems to customize any aspects of the model in order to focus the study on specific features (or even whole system behavior). In this direction, our simulator offers several facilities that simulate diverse scenarios, allowing for a thorough examination of security vulnerabilities and implementation challenges. The flexible framework offered by the simulator enables users to design the desired system based on their specific needs, focusing either on the entire system's

behavior or individual system components. This approach facilitates a deeper understanding and prediction of potential risks or technical issues that may occur while the system is running. It helps optimize performance and suggests solutions to some issues, enabling system administrators to address and resolve problems effectively. Additionally, it enhances OT systems' resilience against known and emerging threats.

The contribution of this work focuses on addressing challenges may arise when integrating Internet of Things devices into industrial environments, with particular emphasis on implementation complexities and security concerns within Operational Technology. In this research, we introduce a simulator that allows designers to thoroughly examine various aspects of their models, showcasing the flexibility of the simulation environment to create and analyze different scenarios. This enables a detailed assessment of potential risks, especially connection issues that can occur when designing such complex systems in industrial settings. The research also contributes to optimizing system performance by providing solutions to anticipated risks and challenges in real world industrial environments.

The remainder of the paper is structured as follows: the related work section discusses the existing work focusing the security threats faced by IIoT devices, followed by an overview of the IoTNetSim and "FS-IIoTSim" simulators, highlighting their limitations in section 3. Next, we explain the Discrete Event System Specification (DEVS) simulator and its application for modeling and simulating various industrial systems, comparing it to the previous two simulators. This is followed by a case study that demonstrates the use of the DEVS environment to model a simple system and extends the approach to designing more complex systems in detail in section 4. Section 5 concludes the proposed work and suggesting future research that could validate DEVS's potential for creating realistic industrial models.

2 Related Work

The nature of IoT makes it tough for researchers to bring their ideas from the lab into the real world. They face obstacles when trying to develop and test initial concepts and prototypes. Building these early prototypes with many connected devices is often impractical during the early design and testing phases because of costs, operational challenges, and unproven protocols. Additionally, using real hardware for experiments is costly and requires broad technical expertise. To tackle these issues, researchers turn to simulation environments to test their ideas and theories. However, each simulation environment has its own set of drawbacks. This section will look at different simulation environments used for modeling and simulating systems in IIoT security, focusing on their limitations. This will help us highlight the benefits of using the Discrete Event System specification simulator to address these limitations by exploring different modeling and simulation techniques used to assess IIoT security.

2.1 IIOT Security Threats

The integration of diverse technologies and operational environments in IIoT systems brought a cybersecurity challenges that encompass several threats aiming at exploitation of vulnerabilities across various layers of IIoT by some of attacking methods like[1], [2]:

Remote Code Execution: This type of attack is used to gain remote control over devices by allowing the attackers to exploit the vulnerabilities in the system. The attackers aims at leveraging from this attack in performing malicious activities like orchestrating botnet attacks, data theft, or malware installation.

SQL Injection: Attackers aims at injecting a malicious SQL commands into URL query strings, or web forms. This will cause comprimisation in database integrity by granting administrative access stealing sensitive data.

DNS Poisoning/Spoofing: By applying this type of attacks to the system, attackers will be able to corrupt DNS data and redirect users to malicious websites instead of the requested one in order to steal their personal or payment details or to install malware on their devices.

Web Application: Malware infection on IIoT devices may cause stealing users data, using the infected devices to lunch Denial of Service (DoS) attacks on specific server or even disrupting operations. Furthermore the devices security and functionality may be comprimised by the Brute Force attacks that target IIoT devices by repeatedly guessing passwords, which will also led to using the attacked devices in the same way as the malware infection uses them.

Data Sniffing: The attacker in this type of attacks aims at data interception and capturing, resulting in breaching privacy, and compromising sensitive information. The Reverse Engineering as well exposes sensitive information of the targeted devices as well as their vulnerabilities, allowing to sophisticate attacks or even inject malicious code.

Wireless Network: In this type of attacks the attackers aims at exploiting vulnerabilities in wireless protocols or the encryption or the configuration, aiming at gaining unauthorized access to the network. Due to this type of attack the data may be stolen, or the network resources may be used by unauthorized

users or even there will be disruption for the data being transmitted.

Data Manipulation Due to the lack of integrity checks, or plaintext data transmission or even the checksum vulnerabilities in the industrial networks, attackers try to corrupt data, disrupt operations or even breach system security.

Replay Attacks This type of attack involve capturing data backets being transmited through the network and retransmit it to specific device system or operators, causing to breach system security, misleading data interpretations, or even operational disruptions.

2.2 IoTNetSim

IoTNetSim, [3], [4], is a platform for modeling and simulating IoT services and networks from start to finish. It has a self contained, multilayered architecture enabling the user to model different structures of IoT systems, including the services they offer and the network connections between components. It is primarily known for its detailed modeling of IoT nodes and sensors, considering power sources and mobility, which makes it very accurate for testing algorithms and configurations. This simulation environment supports a wide range of communication protocols like LoRa, ZigBee, WiFi, and cellular, allowing users to evaluate system performance from various angles. IoTNetSim also supports a broad spectrum of IoT components, including gateways and mobile nodes, making it easier to simulate different applications.

It emphasizes realistic simulation of network and battery failures. It can model data flow in addition to its ability to allow the researchers to implement their algorithms, which in some cases aim at keeping the Security and privacy of the data being transformed between different nodes and components of the designed system as well as processing across different layers of the system. Designed to be compatible with numerous tools and programming languages, IoTNetSim is well suited for IoT research and development. However, it has some limitations. One major challenge is modeling complex sensor mobility, which can be difficult. Indeed, the management of huge complex systems can also be challenging, and the current simulator architecture may need such efforts when integrating new protocols with a wide range of tools. This is a great platform limitation, making the simulation of different IoT systems difficult.

2.3 "FS-IIoTSim"

"FS-IIoTSim", [5], [6], is a simulation tool designed for evaluating the performance of Industrial Internet of Things systems under various conditions. It addresses the unique challenges of industrial applications, including predictive maintenance, real time monitoring, and factory automation. "FS-IIoTSim" supports several existing and new communication protocols and can handle large scale IIoT networks. Its flexibility and scalability allow users to add or update components, making it suitable for testing and validating IIoT solutions in real world industrial settings. One of its key strengths is its ability to provide detailed modeling of IIoT devices, considering communication capabilities, power consumption, and data generation rates. This ensures accurate performance assessments for the models being tested.

Additionally, "FS-IIoTSim" can simulate device malfunctions, power outages, and network failures, which is crucial for evaluating the resilience and reliability of IIoT systems in harsh industrial environments. The tool also allows for real time data processing during transfers between devices. Additionally, it can simulate security threats within the IIoT environment, enabling users to model various attacks to evaluate the robustness of specific protocols and analyze network performance under attack conditions. This helps designers develop recovery plans to reduce the impact of such threats on real systems and improve the performance and effectiveness of the proposed protocols. Despite its strengths, "FS-IIoTSim" has some limitations. The complexity of simulating certain systems can make it difficult and time consuming to set up and achieve desired results. Custom simulation scenarios may be needed for specific research purposes, but the tool might not come with a comprehensive library of prebuilt scenarios. Integrating "FS-IIoTSim" with other development and analysis tools can require additional effort and may only be successful sometimes. While "FS-IIoTSim" has a user friendly interface, mastering its full range of capabilities can take time, especially for complex systems. The tool is structured into three integral components: scenario modeling, performance evaluation, and the user interface. Scenario modeling facilitates the creation of detailed models and generates trace files containing device operation logs. Performance evaluation analyzes these trace files, focusing on latency and energy consumption metrics. The user interface ties these components together, providing a comprehensive simulation experience.

Despite the capabilities of the current simulators, Industry 4.0 needs to improve its tools to study IIoT systems from a security perspective. Most existing simulators predominantly focus on power consumption and data flow within IoT systems. Our research aims to address this gap by emphasizing

the significance of using DEVS for modeling IIoT systems through a security lens. We will provide an overview of the DEVS framework, followed by a detailed discussion of its applications within the industrial sector.

As shown in Table 1, DEVS excels in several key areas, such as scalability, flexibility, and security modeling. This makes it a highly versatile tool for modeling and simulating complex industrial systems, particularly those involving IIoT, where real time behavior and security are critical factors.

3 Descret Event System Specification

In 1976, Bernard P. Zeigler introduced the Discrete Event System Specification (DEVS) formalism, which offers a mathematical framework for modeling both discrete and continuous event based systems. Every real or conceptual system operates with a time base, inputs, outputs, and functions that determine the system's next state and output, reflecting the current state based on its inputs. Utilizing DEVS for simulation allows us to analyze system behavior, enabling performance prediction and optimization, [7, 8].

Figure 1 illustrates the conceptual framework that underpins the DEVS formalism, focusing on three fundamental objects involved in the modeling and simulation process:

- Source system, whether existing (real world) or proposed, can be viewed as a source of information or a set of input/output data over time segments.
- Behavior Database is a reference that holds records of the source system behavior in different scenarios to confirm and validate the model's accuracy and its simulation results to match it with real life behavior.
- The Experimental Frame outlines the scope and limits of the source system behavior by detailing the data collection, testing of the system, and evaluation of the behavior after simulation.
- Model, a set of instructions designed to generate data comparable to what can be observed in a real system. The model's structure consists of these instructions, while its behavior encompasses all potential input/output data that can be produced by accurately executing these instructions.
- Simulator executes the model's directives to produce its behavior.

Figure 1 also shows that the main components are connected through two types of relations.

- Modeling relation, which defines how well the model represents the system or entity being modeled and where it links between the real system and model.

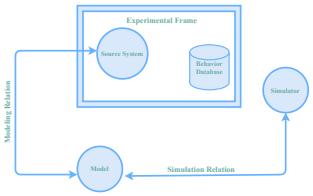

Fig. 1: Basic Entities and Relations in DEVS Environment

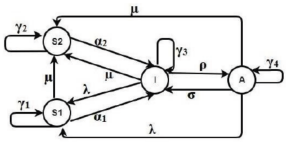

Fig. 2: Model state diagram for constant population

- Simulation relation, which represents how faithfully the simulator can carry out the instructions of the model where it links between the model and simulator.

This formalism specifies the system structure through interconnected models and describes the system behavior using atomic models, including input and output events, states, and how the system will evolve in response to external inputs [7, 9, 10].

3.1 Using DEVS to Model and Simulate IIoT Systems

In [11], we employed the DEVS simulator to model and simulate an IoT Botnet to examine how the botnet spreads through a network when targeting IoT devices. We categorized the total population of IoT devices into four primary states, and the transition behavior of these devices from one state to another is illustrated in Figure 2. The flow between the four states can be explained as follows: A node in the Susceptible state (S_1) can become infected and move to state I at a rate of α_1. Alternatively, it might switch to another Susceptible state (S_2) at a rate of μ or remain in S_1 at a rate of γ_1. Nodes in S_2, which have a lower risk of infection, can either stay in S_2 at a rate of γ_2 or become infected and transition to state I at a rate of α_2. Once a node is in the Infected state (I), it can move to the Active state (A) (where it can attack the target) at a rate of ρ, remain infected in I at a rate of γ_3, return to the susceptible S_1 state at a rate of λ, or move to the lower risk S_2 state at a rate of μ. If a node reaches the Active state (A), it can either go back to I at a rate of σ, remain in A at a rate of γ_4, or transition to either the S_1 or S_2 state at rates of λ and μ, respectively.

Due to the rapid technological advancements, cybersecurity attracted the researchers focus especially in the industrial systems. The complixity and operational importance for these systems requires

Table 1. Comparison of DEVS, IoTNetSim, and "FS-IIoTSim" Simulators

Feature	DEVS	IoTNetSim	"FS-IIoTSim"
Scalability	High: Handles large, complex systems with modular decomposition	Medium: Struggles with very large systems, especially complex sensors	High: Can scale to large IIoT networks but setup is time consuming
Flexibility	Very flexible: Easily integrates new protocols and components	Limited: Architecture makes integrating new protocols difficult	Flexible: Can add or update components but requires additional effort
real time Simulation	Detailed real time behavior and interactions between components	Supports real time simulation, but less robust than DEVS	Supports real time data processing but requires more effort
Security Modeling	Strong focus on security, allows detailed simulation of attack scenarios and failure conditions	Limited security simulation, focuses more on power consumption and data flow	Simulates security threats but not as extensively as DEVS
Ease of Use	Medium: Requires initial setup but components can be reused for larger systems	High: User friendly for small to medium systems, challenging for complex scenarios	Medium: User friendly interface but mastering complex systems takes time
Integration with other Tools	High: Standardized formalism allows easy integration with other tools	Limited: Challenges in integrating new tools due to system architecture	Limited: Integration with other tools can require additional effort

robust security measures, aiming at preventing any financial losses or damages that may affect its performance. The industrial IoT systems requires heightened awareness of potential security threats. This can be linked several factors like: the wide range of the technological and manufacturing devices and services, as well as the continuous communication and data exchange between systems and their associated networks. This research uses the Discrete Event System Specification simulator to understand and evaluate how systems behave. It offers practical recommendations to reduce and prevent different types of attacks, helping to safeguard systems from serious damage and losses.

By comparing between IoTNetSim and DEVS simulator, we found out that IoTNetSim Unlike DEVS experiance diffeculties in dealing with complex systems especially if these systems contains advanced sensors. While the performance of DEVS in simulating complex systems seems to be better due to its capabilities in breaking down these systems into smaller sub systems (Atomic Models). Furthermore, implementing new protocols which considered important for communication in such industrial IoT systems does not requier major architectural changes in DEVS. Unlike IoTNetSim that find it challenging process to implement the new protocols. Additionally, the real time simulation capabilities for behaviour and the interaction between the system components in DEVS is higher than in IoTNetSim. Where despite the strong modeling for nodes and sensors, IoTNetSim still faces inherent limitations in its simulation capabilities making it less robust.

On the other hand, by comparing "FS-IIoTSim" with DEVS simulator, we found that creating a custom scenario may present some challenges for the users due to the need of pre built scenarios and systems complexities. Despite that, we can integrate DEVS with other tools for some scenarios if required by leveraging its standardized formalism. Unlike "FS-IIoTSim", where this may requires extra effort to realize a successful integration with other tools in which it may obstruct system evaluation

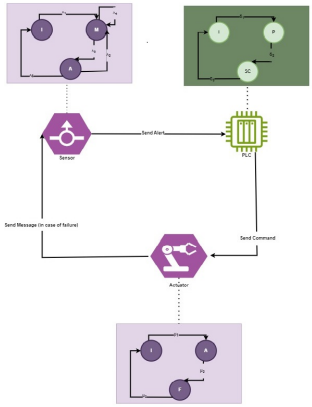

Fig. 3: Smart Factory Sub System

in some cases. Additionally, DEVS outperforms "FS-IIoTSim" in terms of simulation and evaluation from a resilience and reliability perspective. DEVS can accurately model discrete events and failure conditions, including their impacts. In contrast, "FS-IIoTSim" often requires more time and effort to achieve similar results and may reach a different level of accuracy.

4 Case Study Application

In this section, we will provide a detailed explanation of using the DEVS simulator for modeling and simulating complex systems. We will begin by outlining the step by step process of building a simple system and then demonstrate how to expand it to represent a more complex system for a factory.

4.1 Simple System Representation on DEVS

As illustrated in Figure 3, we assume that the basic factory system consists of three primary components: the Sensor, the PLC, and the Actuator. Each component can be considered a separate model with distinct behaviors, referred to as an Atomic Model. We develop each component individually, defining it as a separate model that includes all potential transitions within each model.

Starting with the Sensor Atomic Model, we identify three main states: the first state is Idle,

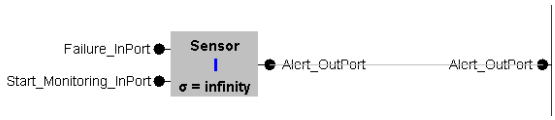

Fig. 4: Sensor Model

the second is Monitoring, and the third is Active. The sensor remains in the Idle state (inactive) until it receives a signal to begin sensing, triggering an external transition that changes its state from Idle (I) to Monitoring (M). After sensing is completed, it generates a message and undergoes an internal transition to change the state from Monitoring (M) to Active (A). In the active state (A), the generated message containing the sensed data is sent to the PLC. It then internally transitions back to the Idle state (I), awaiting another signal to start sensing again. The following steps are needed to model the described behavior:

- Identify and define all the input ports, output ports, and states that represent the system's behavior.
- Initialize the constructor for the Sensor class and add the previously defined ports.
- Create an initialization function to set the sensor's initial state to I (Idle).
- Develop an internal transition function to determine the current state and transition to the next state accordingly.
- Implement the external transition function to change the state based on the received input, which will transition from state I (Idle) to M (Monitoring).
- Design the output function to send the generated message (sensed data) to the next model.

After completing these steps, the model will be generated in the simulator, as illustrated in Figure 4.

Now, considering the PLC Model as illustrated in Figure 3, it also comprises three fundamental states that define its behavior: the first is the Idle state, indicating the PLC is inactive; the second is the Processing state, where the data is analyzed; and the third is the Sending Command state. The model undergoes an external transition when it receives data from the sensor model, moving from the Idle state to the Processing state to analyze the data. Once the analysis is complete, an internal transition occurs, changing the state from Processing (P) to Sending Command (SC). In this state, the PLC sends the command generated during the processing stage to the next model and subsequently returns to the Idle state, awaiting further data from the sensor. The following steps are needed to model the described behavior:

- Define the input and output ports, and states representing the model's behavior.

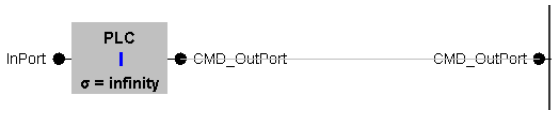

Fig. 5: PLC Model

- Initialize the constructor for the PLC class and incorporate the previously defined ports.
- Create an initialization function to set the PLC's initial state to I (Idle).
- Develop an internal transition function to assess the current state and determine the appropriate transition to the next state.
- Implement the external transition function to change the state based on incoming input, which, in this scenario, will be a transition from state I (Idle) to P (Processing).
- Design the output function to transmit the generated message (command) to the next model.

Upon completing these steps, the model will be created in the simulator, as depicted in Figure 5.

The final model in this scenario is the Actuator Model, which also comprises three states: the Idle, Active, and Failure. When the model receives an input representing the command prepared by the PLC model, it triggers an external transition, changing the system state from Idle (I) to Active (A). If the system is in any other state, its behavior will change internally. The Failure state occurs when an issue with the received command prevents the actuator from processing it. In this case, the actuator sends a signal to the Sensor model to initiate sensing again and repeat the process. To model the described behavior, follow these steps:

- Define the input ports, output ports, and states that capture the behavior of the model.
- Initialize the constructor for the Actuator class and add the defined ports.
- Create an initialization function to set the initial state of the Actuator, starting with state I (Idle).
- Develop an internal transition function to evaluate the current state and decide on the appropriate transition to the next state.
- Implement the external transition function to change the state based on the received input, which in this case will involve transitioning from state I (Idle) to A (Active).
- Design the output function to send a signal back to the Sensor model in case of an error in the command received from the PLC.

After completing these steps, the model will be generated in the simulator, as shown in Figure 6. Furthermore, when the three models are combined, the resulting system is illustrated in Figure 7 below.

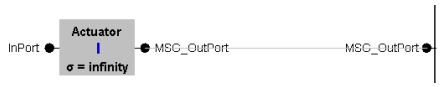

Fig. 6: Actuator Model

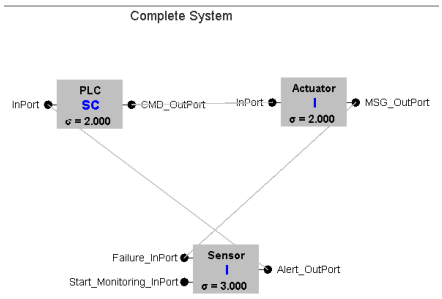

Fig. 7: Complete Simple System

4.2 Complex System Representation on DEVS

When dividing a complex system into smaller parts, we will have a simple part, where building these parts and combining them together will lead to having a complete system, which will be more complex. In our scenario, we discussed a simple system consisting of one Actuator, sensor, and PLC. We discussed the behavior of each system and the way of communication between the three models to explain the idea of using DEVS. In this part of the paper, we will discuss a more complex system consisting of 10 sensors, one PLC, and 10 actuators; note that we can scale up the system as we need to have hundreds of devices based on the factory's needs. However, the whole Idea explains how to convert a simple system to a complex one in our case. We built the three basic atomic models from the previous part: the sensor, PLC, and actuator. In this part, we will use the same models with slight changes to create the model shown in Figure 8. To replicate the complex system illustrated, follow these steps:

- Modify the sensor's code to include the sensor number in the message it sends.
- For the PLC, add output ports equal to the total number of actuators in the system; 10 output ports in this case.

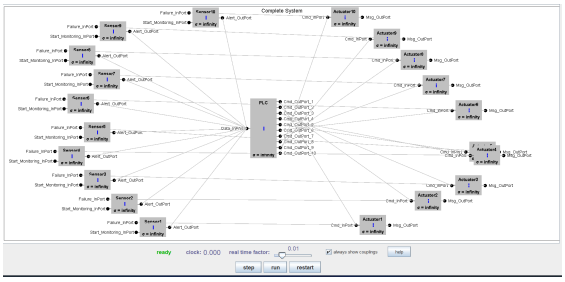

Fig. 8: Complex System

- Implement a queue in the PLC class to store incoming messages from all sensors in the correct order.
- Update the Message out function to extract the sensor number from each message stored in the queue.
- Develop an algorithm according to the system's requirements to forward the command to the appropriate actuator. In this case, the command is sent to the actuator corresponding to the sensor that transmitted the data.
- In the system class that integrates all components, ensure that each model is correctly connected to the appropriate models for sending and receiving data.

5 Conclusion

In this work, we presented a comprehensive method for utilizing DEVS to model and simulate complex industrial systems. We showed that building a whole system and reflecting it's overall behavior requires breaking it down to its core components, model each of these components separately, and coupled the components to represent the system. In Order to simulate a complex interactions between the atomic models by customizing them. A detailed explanation on constructing a complex model using the previously developed Atomic models to create more sophisticated system were discussed earlier in this research. This shows the flexibility of the DEVS simulator, Gaining clearer insights for the system behavior and possible risks may occur in the industrial environments. The discussed approach for modeling aims at detecting possible risks in OT systems aiming at optimizing and enhancing the system performance. As well as providing valuable insights into the system's resilience, and communication efficiency. Following the modular design principles, future enhancements and modifications can be seamlessly integrated into the system. The potential of the DEVS simulator extends beyond this point. The proposed work opens a scope to focus on incorporating more advanced features such as real time monitoring, scalability for larger systems, and integrating cybersecurity scenarios to better evaluate and mitigate risks in Industry 4.0 environments. This simulator is a versatile tool that is valuable for both practical applications and academic research, enabling the design and optimization of modern industrial systems.

Disclosure Instructions

During the preparation of this work the authors used Grammarly for language editing. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] A. C. Panchal, V. M. Khadse, and P. N. Mahalle, "Security issues in iiot: A comprehensive survey of attacks on iiot and its countermeasures," in *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, pp. 124–130, IEEE, 2018.

[2] Y. Shah and S. Sengupta, "A survey on classification of cyber-attacks on iot and iiot devices," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0406–0413, IEEE, 2020.

[3] M. Salama, Y. Elkhatib, and G. Blair, "Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 251–261, 2019.

[4] K. O. Chee, M. Ge, G. Bai, and D. D. Kim, "Iotsecsim: A framework for modelling and simulation of security in internet of things," *Computers & Security*, vol. 136, p. 103534, 2024.

[5] H.-H. Lee, J.-H. Kwon, and E.-J. Kim, "Fs-iiotsim: a flexible and scalable simulation framework for performance evaluation of industrial internet of things systems," *The Journal of Supercomputing*, vol. 74, pp. 4385–4402, 2018.

[6] R. Almutairi, G. Bergami, and G. Morgan, "Advancements and challenges in iot simulators: A comprehensive review," *Sensors*, vol. 24, no. 5, p. 1511, 2024.

[7] M. Jarrah, "Modeling and simulation of renewable energy sources in smart grid using devs formalism," *Procedia Computer Science*, vol. 83, pp. 642–647, 2016.

[8] B. P. Zeigler, "Devs today: Recent advances in discrete event-based information technology," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pp. 148–161, IEEE, 2003.

[9] M. Albataineh and M. Jarrah, "Devs-iot: performance evaluation of smart home devices network," *Multimedia Tools and Applications*, pp. 1–29, 2020.

[10] N. Akhtar, M. Niazi, F. Mustafa, and A. Hussain, "A discrete event system specification (devs)-based model of consanguinity," *Journal of theoretical biology*, vol. 285, pp. 103–12, 06 2011.

[11] G. Barakat, B. Al-Duwairi, M. Jarrah, and M. Jaradat, "Modeling and simulation of iot botnet behaviors using devs," in *2022 13th International Conference on Information and Communication Systems (ICICS)*, pp. 42–47, 2022.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Author Contributions:

Ghena Barakat: Investigation, Writing original draft, Formal analysis, Methodology, Conceptualization, Visualization, Software.
Luca D'Agati: Methodology, Writing, review & editing, Validation.
Giuseppe Tricomi: Methodology, Writing, review & editing, Validation.
Francesco Longo: Supervision.
Antonio Puliafito: Supervision.
Giovanni Merlino: Supervision, Project administration, Methodology.

Sources of Funding for the Research

This work is partially supported by "JOULE" receiving funds from the Italian Ministry of University and Research PRIN project "JOint ResoUrce Management in ReconfigurabLE I4.0 Factories (JOULE)" D.D. n. 104 del 2/2/2022. Moreover, this work was partially supported also: by project SERICS (PE00000014) under the PNRR MUR program funded by the EU - NGEU, and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE0000001 – program "RESTART")

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.