information

Article

Towards a DEVS Model Management System for Decision-Making Web Applications

Laurent Capocchi [1,*] and Jean François Santucci [2]

SPE UMR CNRS 6134 Laboratory, University of Corsica, 20250 Corte, France
* Correspondence: capocchi@univ-corse.fr

Abstract: The discrete event system specification formalism introduced by Zeigler in the 1970s is ideally associated with new technological advances in the web to offer an almost quasi-automatic mechanism for exporting its simulation models associated with experimental frames into web apps. In this paper, we show how, thanks to the association of certain current web concepts (cloud computing, application virtualization, etc.), discrete event system specification formalism makes it easier to develop web apps that use simulation models to assist decision-making. We propose a simulation model management system used by teams of data scientists, modelers, and developers (engineers) capable of building and deploying web applications from simulation models with minimal web development knowledge.

Keywords: modeling; simulation; discrete event; web; CMS

Citation: Capocchi, L.; Santucci, J.F. Towards a DEVS Model Management System for Decision-Making Web Applications. *Information* **2023**, *14*, 69. https://doi.org/10.3390/info14020069

Academic Editor: Tuncer Ören

Received: 26 November 2022
Revised: 17 January 2023
Accepted: 21 January 2023
Published: 26 January 2023

1. Introduction

Modeling and simulation (M&S) is a discipline oriented first of all toward engineering and research, but has in the last few years been widely used by users and developers of mobile applications through cloud storage and web services [1,2]. Recent developments in the field of cloud computing and service-oriented architecture offer advances in how to better utilize M&S capabilities [3].

If we look at the conceptual and technological advances in languages and protocols for the web that have appeared over the past 20 years (object-oriented languages for the web such as Node.js or Python, or asynchronous communication protocols between objects with WebSockets, for example authentication protocols, etc.), we note that DEVS formalism implements intrinsic computer concepts such as the object-oriented approach, modularity, abstraction, or composition, which, based on technological web advances, allow us to consider it as a collaborative web platform for modeling and simulating discrete event systems. DEVS allows an approach to building its models that can be based on design patterns (a set of DEVS atomic or coupled models defined to model/simulate systems) and building blocks (a well-known notion of directly reusable, coupled models containing a set of interconnected submodels). These two notions (in addition to the management of state duration specific to DEVS) are highlighted in [4] and make DEVS an excellent language for Internet of Things (IoT) systems, for example.

However, we can go further. In fact, these DEVS capabilities can be extended and combined with web concepts to define a collaborative web M&S environment capable of offering a customizable platform accessible by authentication on the web for: (i) collaborative construction of model libraries; (ii) remote simulation of models from web interfaces (applications) or representational state transfer application program interfaces (REST APIs) [5,6]; and (iii) analysis and display on the web of simulation results in real time to improve a decision-making process, for example, a web application dedicated to simulating a medical diagnosis of a patient after receiving, through the web application, a set of answers to a certain number of questions on his state of health. The doctor must make a decision

based on a questionnaire offered on a web interface whose answers are used in a discrete event simulation model. The modeling engineer develops the simulation model (probably stochastic) on the basis of his scientific knowledge of a specific mathematical formalism, but if he has no knowledge of web development, he will have to seek the help of a web developer to build the web application that implements its simulation model under real conditions. It is in this that decision-making systems will benefit from an environment that facilitates the passage from the construction of the simulation model to its use in a web interface by end-users.

This platform can be accessible by user authentication (engineers, analysts, developers, and end-users) accessing shared resources (models, experimental frames, simulation instances, simulation results, etc.) to build, reuse, or analyze DEVS simulation models that must be exposed on the web in order to make them accessible to the greatest number of end-users (specialist or not in the field relating to the exposed simulation model). We are familiar with these types of content managers, which are widely used tools on the web, in particular for the deployment of web applications (or websites) (Wordpress [7], Durpal [8], Joomla [9], etc.) [10], but there are no such tools in the M&S field. However, developers who know the user interface specifications are in charge of developing the final web application, allowing them to play with the simulation models. They rely on knowledge of the simulation model and its execution results to build a web application to interface (configure, simulate, and interpret) with the simulation model remotely. It would be ideal to be able to automatically generate a family of web applications from the implemented simulation model, which could possibly allow developers to deploy the application of their choice over the internet. Thanks to the concept of user profiles, another developer could build a web application from a simulation model already implemented and shared in a collaborative M&S environment.

This paper shows how, thanks to the association of some recent web concepts (cloud computing, application virtualization, etc.) and web techniques (asynchronism, WebSocket communication, etc.), DEVS formalism is able to constitute the basis of a web application manager of DEVS-MMS (DEVS model management system) simulation models built through a back-office interface and directly usable remotely through a front-office web interface generated using a quasi-automatic mechanism.

A content management system (CMS) [11,12] is a tool that allows users to build their websites without having to write everything from scratch. A web CMS has two major components: a front office and a back office. The front office is the front-end user interface for a CMS back end/back office. This is the interface that allows users to log in and create, edit, update, and publish content without having to do any programming or coding. This effectively separates the website or application codebase from content management, so content can be updated without the need for a developer. The back office is the "true" CMS back end that allows content to be actually updated on the website. It contains all the code and logic required to convert front-office content to finally become accessible to end-users on a website. CMS architecture refers to the design and implementation of front-end and back-end processes in CMS systems. The CMS architecture defines the relationship between the tools for publishing and managing posts and pages and the tools for creating and managing them, front end and back end, between the tools used to create and edit them. There are five basic types of CMS architecture: coupled, decoupled, hybrid, headless, and SaaS. In this paper, we are inspired by traditional coupled architecture due to the minimal infrastructure investment and its simple set-up, integration, and deployment. Thanks to its properties of building models by composition, its hierarchy of description and abstraction, and its explicit separation between a model and simulator, which is generated automatically, DEVS allows for M&S of systems with an effective formal and generic approach approved for more than 40 years in several fields of application. As a result, DEVS is ideally associated with new technological advances in the web to offer an almost automatic mechanism for generating web applications or REST APIs to access its simulation models remotely.

In the rest of this paper, after reporting on the work already performed in this area, we present the architecture of the collaborative environment of M&S on the web. Finally, we will give food for thought on the future of the use of this type of environment in the M&S community and the possible evolution of its functionalities in the future in relation to the possible advances of both DEVS formalism and the web.

2. Related Work

As indicated in [13] (which falls in with [14]), M&S needs to be democratized and carried out in a collaborative environment that offers the concepts of composition and reusability of models stored in shared libraries. In this paper, particular attention is paid to DEVS and highlights the difficulty of reusing DEVS models, the lack of tools to visualize and analyze DEVS simulation results, and the lack of tools to assist decision-making. Additionally, DEVS is considered resource-intensive, which can be a problem when modeling and simulating large systems. From the point of view of the web-oriented tools available that partly respond to their problems, the authors mention geographic information systems (ArcGis, Google Maps, MapBox, etc.) and data analytic and visualization tools such as Microsoft PowerBI, Tableau, and D3, for example. From the point of view of the web-oriented tools available and partly responding to their problems, the authors mention that there is no generic web environment (not specific to an application domain) allowing users to build their own models. The authors propose the architecture of a web environment by distinguishing four types of users: modelers, analysts, developers, and decision makers. Each user is limited to his task. For example, modelers feed the databases of models and simulation results (stored on a server). Analysts use and complete the simulation results databases by building data visualization interfaces. Developers use these databases to develop applications for decision-makers. The authors propose a data-centric modeling workflow and a DEVS API as a toolbox for simulation applications. In our presented approach, we propose a quasi-automatic web application generation mechanism from a DEVS simulation model. We offer a DEVS simulation model management system architecture used by teams of data scientists, DEVS modelers, and developers (engineers) capable of generating and deploying web applications for decision-making end-users.

M&S as a service (MSaaS) [15] is a concept based on cloud computing as a service model and combines web services and M&S applications. The MSaaS framework provides a dynamic simulation environment that can be applied and utilized on demand by modelers. Modelers can find new ways to work together and improve their operational effectiveness in M&S. They will be able to communicate with each other directly and will have the ability to connect to the same DEVS simulation models. They can find new opportunities to increase the efficiency of the models and save cost and effort in M&S. In a typical MSaaS platform, a modeler can access M&S features as services using a web browser or smart client. All M&S services are stored in the cloud and available to smart clients that can embed web applications. The approach proposed in this article is the same as any other typical MSaaS platform that claims website access. However, while the number of MSaaS tools is growing [2,15–20], some important features need to be proposed in the field of simulation realization (add/remove models that alter the structure of a model during simulation) and real data acquisition from the sensors embedded in the system of systems (SoS) (like ubiquitous systems) involved in simulations.

In [21], the authors performed a bibliographic search to categorize MSaaS systems (strengths and weaknesses) according to the architectural styles that can affect their functionality. The main results of this study can be summarized in four headings:

- **Architectural Styles**: After the analysis of the M&S applications studied, the layered models are the most adopted to design MSaaS architectures, and therefore future architectures must provide mechanisms to satisfy *interoperability* and *deployability*, which are two important criteria in MSaaS applications before cost, performance, scalability, and configurability.

- **Containerize M&S Applications**: The MSaaS applications of tomorrow must offer an assembly of simulation models running in parallel containers (e.g., Docker containers) isolated and deployed almost automatically. This architecture, in the form of a containerized simulation orchestration, could be appreciated in the field of military applications or M&S applications. It is too often monolithic, and requires significant centralized simulation resources, which could be shared.
- **End Users and Interface**: The proposed review shows that the main aspect, apart from an MSaaS application, is to enable end-users to build and analyze simulation models and results in an effective manner'. Future MSaaS applications must hide the complexity and execution of simulation models, but they must also provide environments for building and configuring simulation web interfaces intended to simplify the use of simulation models by end-users. This can be achieved in part by giving MSaaS applications more flexibility for users to build their own models based on specific needs and benefits from an automatic interface-generation process for simulation models. In the domain of discrete-event M&S, a work set based on a client-side approach to web application allows the creation of user interfaces to access, visualize, and analyze the results of a modeling process executed on servers. This work also offers the ability to run simulations in a browser. The use of web applications for M&S is well described in [22]. The article proposes a client-side approach to improve user interfaces and to perform simulations and visualizations in a browser. The authors describe the use of client-side technologies to create interactive web applications that simulate specific (not generic) models of biochemical oxygen demand and dissolved oxygen in rivers. This approach is very interesting, but is not generic in the sense that no mechanism makes it easier to build web apps from simulation models. Web applications must be implemented from scratch from each simulation model. In our approach, we want to provide a model-driven approach in the construction of web applications of simulation models, and DEVS formalism is advantageous for this, as we will present in the rest of the paper. The authors deal in [23] with a single web application to implement a wide range of models, including the ability to access simulation tools through web applications. The article also describes web processes for sharing simulations and interactively visualizing results. They plan to improve this approach in the future to promote reproducibility, encourage collaboration, and facilitate the creation of simulation models. In [24], the authors propose web applications designed and developed within the MATLAB software suite to propose web applications based on browsers to end-users who are not MATLAB developers. They offer a tool called MATLAB App Designer to implement web application interfaces. However, visualizing the results of simulations through an interface generated using the MATLAB App Designer is quite difficult, as visualization of results requires a different workflow. In our approach, the web formats of the simulation results are deduced from a set of collector models contained in the simulation models. The modeler will facilitate the design phase of the simulation results in the web application based on a parsing system that allows the user to deduce what type of web visualization component (type of graph) the web application will be made of.
- **Architecture for Modern M&S Applications**: In this last point, based on the analysis of the requirements highlighted for a military-type MSaaS application, the need for tools to create and administer parallel simulations is underlined in the study. In fact, the need to run parallel simulation farms and monitor them (suspending, deleting, or updating) would be an added value for future MSaaS applications. This is an aspect that is not directly discussed in this paper, but can be easily integrated as specified in the conclusion.

In this paper, we will show that the proposed approach, which is based on the capacity of DEVS formalism, makes it possible to meet all these needs and necessities.

In [3], a DEVS-based, cloud-deployable framework is introduced in order to speed up the NP-hard problem modeled with DEVS. The framework is used for the optimization of UAV trajectories and sensor strategies in target-search missions. The cloud deployment architecture is facilitated by the xDEVS, and container-based distributed infrastructure is used to execute simulations in an efficient way. This work is interesting because it shows how DEVS is enhanced to make distributed simulations. However, although the results of simulations are stored from each atomic model embedded in a container, there is no mechanism that allows for creation from the simulation model of an interface allowing users to configure a simulation and display the results on the web. Therefore, this work is complementary to that presented here.

From a technical point of view and specifically for the Python language, the Anvil [25] Python framework makes it possible to build web applications (front and back office) with this single language and without prior experience in web development. It is a powerful framework that automates bidirectional communication between the front and back office by implementing callback functions. It is a framework that can be used to implement the approach proposed in this document; however, it would still be necessary to find a way to communicate with the discrete event simulation processes in real time, as will be explained later in this paper. The streamlit [26] Python framework is also a possible implementation solution for the approach proposed in this paper. It allows users to implement a web application in pure Python. There is no graphical user interface to build the front office. Graphical web components are instantiated via coding, and callback functions to communicate between the front and back offices are also implementable in Python. Amazon Honeycode [27,28] is another solution that allows users to implement web and mobile apps without writing code. However, it is not implemented in Python, and it will be more difficult to interface with the M&S DEVS environment proposed in this paper (which is fully implemented in Python). It can be used to develop the front of a web app, which can then communicate bidirectionally with the DEVS simulation process through a REST API instead of communicating directly through socket (as will also be presented in this paper).

3. Generic Proposed Approach

The approach proposed in this paper consists of making possible DEVS M&S on the web in a collaborative environment capable of sharing resources stored in the cloud (models, external data, projects, web components, etc.) between different actors (modelers, analysts, and developers) with the aim of modeling and simulating DEVS systems and proposing web interfaces intended to help end-users make choices in decision-making processes.

Figure 1 presents this approach, in which DEVS-MMS is at the center of the device that generates web interfaces (API or web apps on the right side of the figure) from the databases mentioned above and is manipulated by the three main actors presented on the left side of the figure. End-users are only users of web interfaces generated from a DEVS simulation model, but we will see later that they can also benefit from an interface for composing web interfaces from a simulation not imposed by the developer.

Web applications are software accessible over the internet. Also called web apps, they differ from standard websites in particular thanks to their functionality; in other words, they do not just display information, but also allow users to perform various user-centric tasks. Mobile applications can be considered web apps that improve the use of simulation models and make them more available to a large number of non-specialist users. The deployment plan of a model, from creation to use, is simplified using the DEVS-MMS suite (the tool chain) as a generic collaborative framework to export DEVS simulation models that can be simulated from the web. This tool chain cannot be categorized as a full MSaaS platform, but offers attractive MSaaS services related to real data acquisition and dynamic structure modification during simulation. The proposed generic approach (Figure 1) allows users to automatically deploy a DEVS simulation model built by a team of engineers on a mobile

terminal through web services (smartphone, tablet, etc.). When the system is modeled and validated (via simulation), it can be proposed to the end-users to remotely simulate the resulting model of the studied system with data acquired from mobile terminal sensors immersed in a real physical environment, for example. The benefit of the approach and its motivation are mainly highlighted through the ability (i) to perform data acquisition from mobile terminal sensors in order to simulate a system in a real physical environment and (ii) to interact dynamically with the model from the mobile app during simulation. The mobile terminal becomes a source of input data for simulated models and allows the user to feed its simulations with real data. For example, initially the user can select a model depending on its position or the context in which it is located (mobility). Therefore, the selected model is dependent on real data that may be used by the simulation. This innovation is possible due to DEVS formalism, which shares its ability to specify systems with a discrete event approach in a hierarchical and modular way in a well-defined experimental frame. As will be explained in Section 5, DEVS allows for the definition of design patterns based on model interconnection, which facilitates the specification of a web application associated with a given simulation model. The building-block approach (DEVS component) which is made possible through the DEVS formalism facilitates the work of the modeler, who can be guided in the development of a web application corresponding to his simulation model without prior knowledge of web development. Furthermore, communication between the web application and the DEVS simulation process (executed on a web server) is also made possible by the introduction of specific DEVS models that can be directly integrated into the rest of the models that make up the simulation model due to the application of a design pattern intended to export DEVS simulation models to a web server.

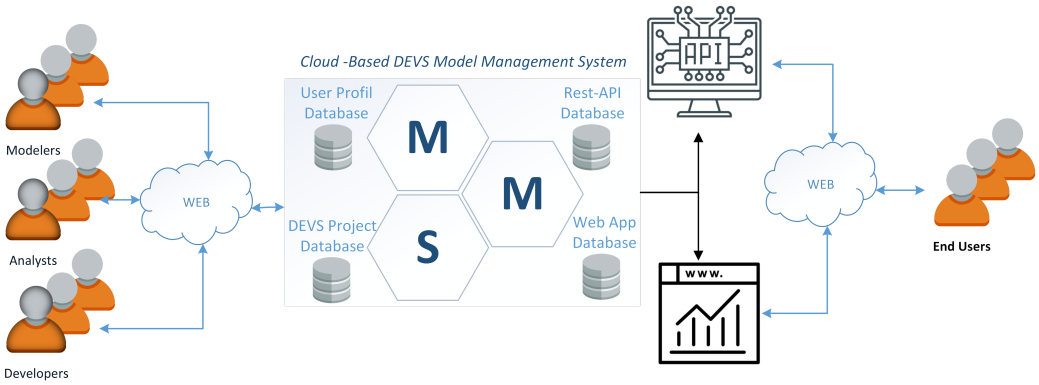

Figure 1. The generic approach proposed for the cloud-based DEVS-MMS is shown in the center of the figure. Modelers, analysts, and developers (on the left), whose profiles are stored in the database, work collaboratively from the database of models and DEVS M&S projects containing, among other things, the results of simulations to specify and implement DEVS simulation models with their web interfaces. DEVS-MMS generates web interfaces (API or web apps), giving end-users (right) the ability to remotely operate DEVS simulation models through the web.

A REST API (or "RESTful") is an application programming interface that uses HTTP requests to obtain, place, post, and delete data. REST technology can be considered the language of the internet. Today, the increasing use of the cloud leads to the appearance of different APIs aimed at presenting web services. REST is a logical choice for developing APIs that allow end-users to connect and interact with cloud services. Many sites use RESTful APIs, including Google, Amazon, Twitter, and LinkedIn.

In the rest of the paper, we detail the architecture of the DEVS-MMS and show how the properties of DEVS formalism simplify the implementation of such a collaborative environment using new web techniques.

4. Architecture And Workflows

Figure 2 shows the architecture of the DEVS M&S manager and the DEVS simulation model web interface, DEVS-MMS. Each user connects to a back office using a profile stored in a user database (user profile database in Figure 1). The modeler builds the simulation models that are validated via the simulations (marked plain circle 1). These simulations are stored in the database and are accessible by the analyst, who uses the simulation results to provide visuals (tables, graphs, etc.) to analyze the behavior of the simulation model (marked plain circle 2). Finally, the simulation model, as well as its simulation results, allows developers to propose web applications or REST APIs (marked plain circle 3) that are intended for an end-user who wants to take advantage of the simulation model without knowing its details or even the way it is simulated (marked plain circle 4).

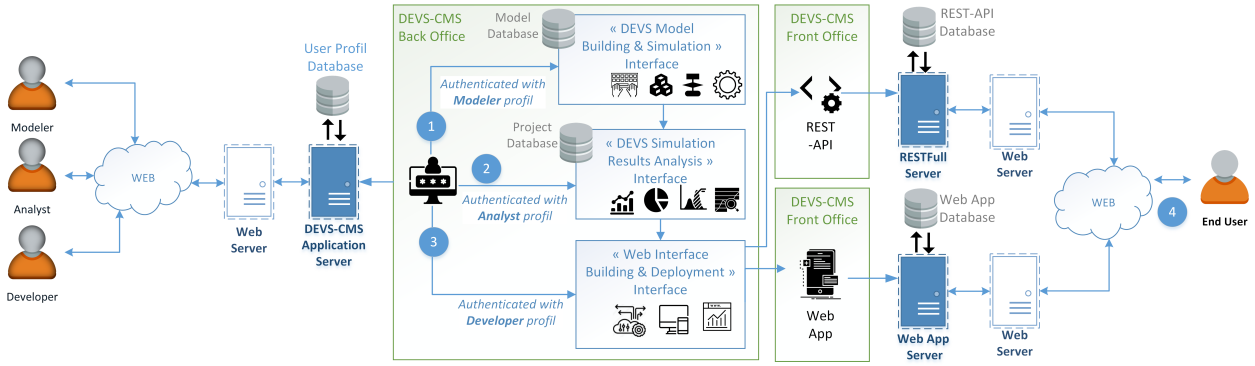

Figure 2. Architecture of the DEVS simulation model's web interface manager, DEVS-MMS. Users (modelers, analysts, and web application developers on the left of the figure) access their respective back office after authentication (marked plain circles 1, 2, and 3) from the profile database. The modeler accesses the interface to build models and simulations that are necessary for the analyst, who produces the data visualization. These analysis elements are used to build web interfaces (REST API or web apps stored in the database), allowing the end-user to interact with simulation models remotely via the web (marked plain circle 4).

Figure 3 shows the process of building DEVS simulation models. The modeler works with any external data that help to configure or calibrate the specified models using the domain natural language [29], for example. The structure of the simulation model (coupling between the atomic or coupled DEVS models) can be built from design patterns previously defined when building simulation models responding to the same type of problem. The simulation model is then validated by running its abstract DEVS simulation tree, which is automatically generated. Here, too, it is due to this DEVS simulation tree, which is composed of a simulator (in charge of executing the atomic models) and a coordinator (in charge of orchestrating the coupled model's execution), that the parallel or distributed execution algorithms can be easily applied on intensive computing web servers (although in [30], Professor Zeigler shows that PDEVS formalism makes it possible, in certain cases, to avoid having recourse to the execution of its parallel or distributed algorithms). If it is not valid, the modeler must review the specification or assembly phase of the DEVS model. Otherwise, the simulation results analysis phase can begin. In parallel with this process, a set of databases is used; the models database contains all the simulation models, and the compiled models database contains the compiled version of the models and is used by the stored simulators in a specific database to obtain the simulation results, which are stored in a project database.

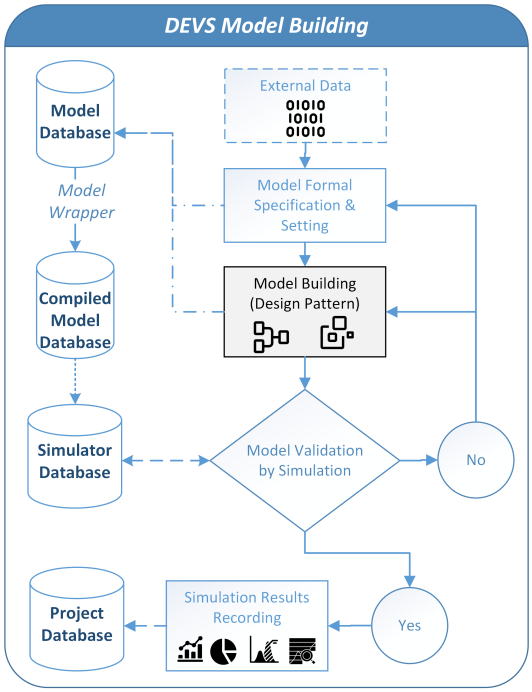

Figure 3. Process of building DEVS simulation models from external data (which may be massive) possibly used in the formal specification of DEVS models. Design patterns can be used in the block construction (black in the center of the figure) of the models. Validation of the models is carried out via simulation to obtain the traces necessary for analysis of the simulation results. The left part of the figure shows the management of models and simulation results in the associated databases.

Figure 4 presents the workflow followed by the analyst, who, from the DEVS simulation results, builds the dashboards composed of graphics and statistical information from the simulations. These data are stored in the database and will be used by the developer to build web access mechanisms for the simulation model (REST API or web app).

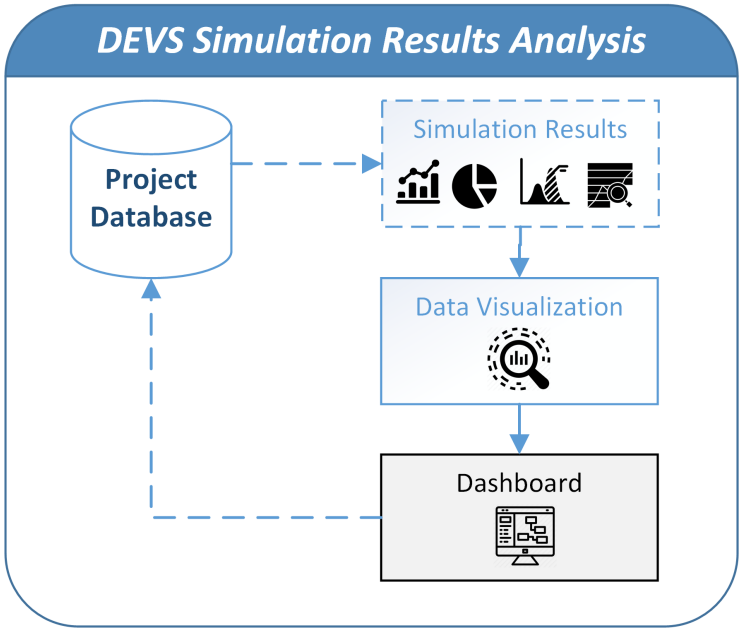

Figure 4. Process of analyzing simulation results stored in the database to generate analysis tables, which then allow the developer to implement web interfaces. The data visualization phase can be performed using artificial intelligence techniques.

Figure 5 shows the process of building the web interface of a simulation model and then deploying it as a container on a cloud web application deployment platform (Heroku [31], AWS [32], or Google App Engine [33], for example). To do this, it is necessary first to create an autonomous version of the simulation model, which is a compressed file consisting of the DEVS simulation model and the sources of the associated DEVS simulator. From this archive, the developer implements the web application (in black in Figure 5) or the REST API. In one case, the web files will need to be coded (classical HTML, CSS, or JS files, for example) to create a progressive [34] or responsive [35] application to configure and run the simulation model through a web application. In the other case, the developer will just have to implement the REST API with the right routes, which will also allow for the simulation model to be run, but from a correctly formed URL. In any case, the developed web interface is stored in a database and will only be deployed on the developer's orders. The deployment is performed by adding a Docker [36] file to the implemented web interface sources. Finally, a cloud-based web application platform, such as Heroku or AWS, can be used to expose the web interface online.

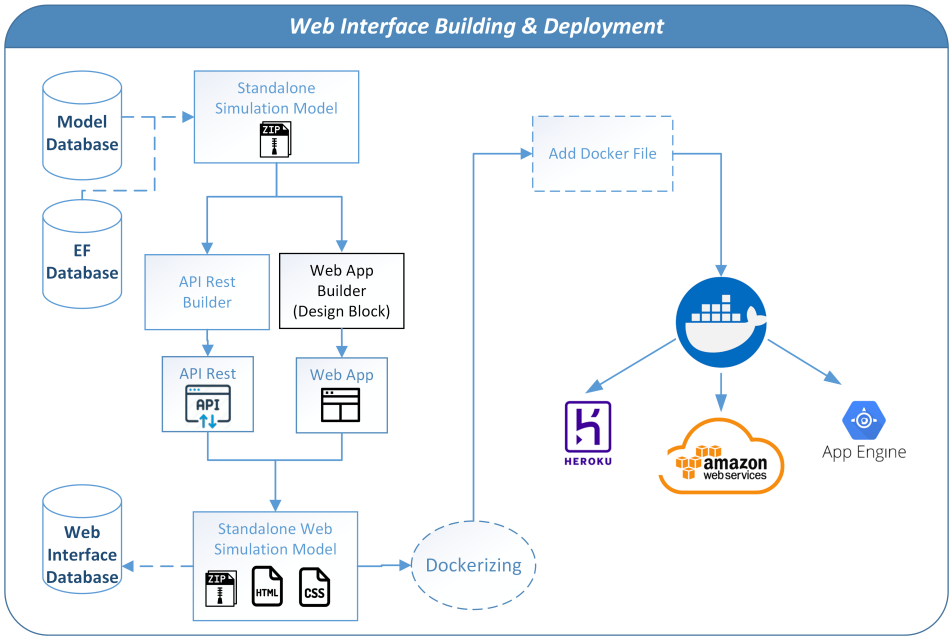

Figure 5. Construction of the web interface of a simulation model from its standalone version, which is a compressed package that includes the DEVS simulation model with its experimental frame (EF) and its simulator. The process of building the web application (in black) that allows the simulation model to interface with a graphical application is detailed below. The web interface (REST API or web app) is stored in a database. It will only be published via Dockerization [36] if the developer wants it. Dockerization is a process that creates an isolated container of the standalone simulation model and its web interface, which can then be easily exposed on a web application service such as Heroku or Google App Engine. This process requires the addition of a Docker file to the source codes of the standalone simulation model.

5. Web-Based Discrete Event System Modeling and Simulation

Why is DEVS advantageous in exposing its simulation models on the web? Zeigler developed DEVS formalism at the end of the 1970s with an object-oriented approach, which, it must be emphasized, was not very widespread in the field of computer development at that time. His vision was correct, and we can consider that his modular and hierarchical approach to system modeling with the principle of interconnecting atomic models within coupled models (which can themselves be contained in other coupled models) is without doubt advantageous in the field that interests us in this paper: M&S of DEVS models through the web.

Indeed, when modelers build simulation models by interconnecting basic components, they have no idea how their models can be used by other end-users, who are often not specialists in the domain to which the model belongs. For example, in order to model the health pathways of a patient suffering from a pathology, a developer would develop or use DEVS models that interconnect to simulate the behavior of the patient when subjected to medical treatment. The structure of the DEVS simulation model is not built to be used in any other way or by other users. This is where the modular aspect associated with the concept of the discrete event of DEVS makes it possible to create a bridge with the web world directly from the simulation model, without great effort on the part of the modeler, while also facilitating then the task of the developer who will build a web interface. Indeed, the application of a design pattern composed of generator and collector models makes it possible to transform a classic simulation model into a simulation model that communicates with web interfaces.

To help the simulation team deploy a DEVS simulation model on the web, we propose a design pattern (DEVS-TO-WEB Decorator) made up of two types of DEVS atomic models: *Web Generator* and *Web Collector*. The DEVS Web Generator atomic model is responsible for retrieving data from the end-user web application (marked plain circle 1 in Figure 6), and the DEVS Web Collector atomic model is dedicated to collecting simulation results and sending them back to the end-user web application (marked plain circle 2 in Figure 6). The Web Generator embeds a loop process that waits for an event coming from the end-user web app. It transmits this event to the DEVS-coupled model connected to the Web Collector, which will send the simulation results back to the web app.

In the "DEVS M&S" top layer in Figure 6, the engineer designs a DEVS-coupled model of the complex system under study using classic DEVS Generator and Collector models to validate the DEVS-coupled model corresponding to the complex system through simulation. Then, in the DEVS-TO-WEB design layer, the DEVS-TO-WEB Decorator can be applied to replace these classic DEVS models with the new Web Generator and Web Collector models. Moreover, the engineer needs to make some settings related to the network configuration in these two models in order to enable directional communication between the simulation process and the web app (mainly URL network settings, as explained in the next section).

Figure 6 shows how the modeler makes this communication with the web possible by applying a design pattern based on two atomic models: the Web Generator and the Web Collector. Traditionally, a simulation model is embedded in an experimental frame composed of an atomic model that generates events (DEVS Generator in Figure 6), one or more coupled models that process these events (DEVS Coupled Model in Figure 6), and event collection models (DEVS Collector in Figure 6). When the simulation model is validated through simulation (the modeler considers that the behavior is acceptable with respect to the targeted accuracy objectives), he only needs to replace the event generators with the Web Generator model and the event collectors with a Web Collector template. These models are implemented to communicate with the web; the Web Generator has an event loop in its internal transition function, which causes an output upon receipt of a WebSocket, while the Web Collector has a mechanism for sending a WebSocket in its output function. The modeler must still configure these models by assigning socket server URLs as well as the name and type of incoming and outgoing sockets in the simulation model. This mechanism allows an end-user to communicate, for example, by sending a WebSocket with the running simulation model without a time limit. Of course, this mode of communication is in addition to that which consists of executing the simulation remotely by simply executing the autonomous simulation model. In this case, the user remotely requests the execution of the DEVS model and waits for the end of the process without interacting with it. This is also the case when the web interface comes down to a simple interface to configure and execute the model with a simple click of a button. This is also the case for invoking a REST API.

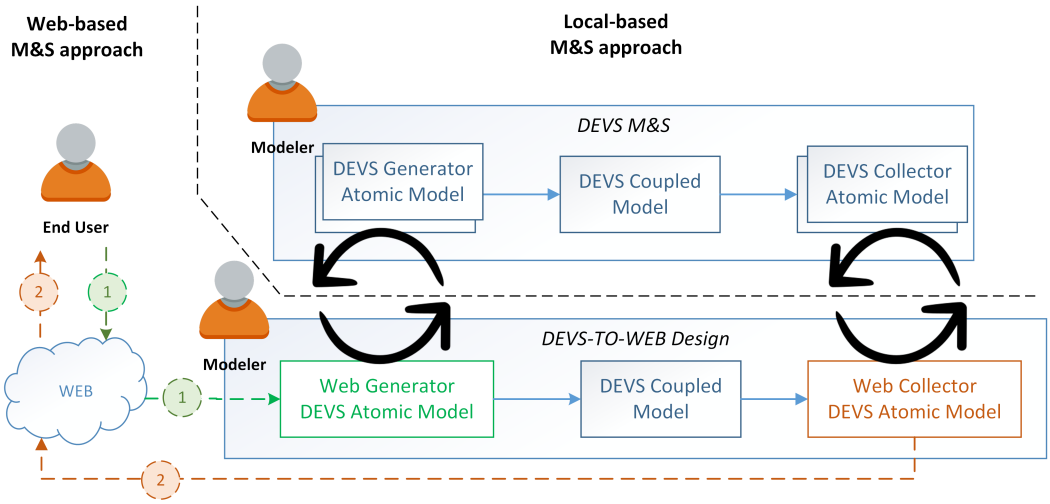

Figure 6. DEVS model-building methodology using DEVS-TO-WEB Decorator involving the DEVS *Web Generator* and *Web Collector* models introduced to enable real-time, bidirectional communication. In the web-based M&S approach, DEVS Generators (resp. Collectors) are replaced by a Web Generator (resp. Collector) DEVS atomic model involved in the local-based M&S approach.

This mechanism is feasible from a technical point of view thanks to the latest advances in the web, and, more particularly, with regard to the integration of sockets in client and server applications. In fact, libraries now allow users to manage sockets using browsers and object-oriented applications.

Figure 7 shows how the bidirectional communication mechanism is implemented using the socket client/server message over the HTTP protocol. When an end-user wants to send a message to a standalone simulation model stored on an asynchronous web server (marked plain circle 1), the web application sends a socket to the web server that transforms this request into a request sent to the Web Generator server embedded in a standalone simulation process. The simulation extracts the data from the received message, and the coupled model is executed to generate a message for the Web Collector model. This model sends a message to a server that sends a socket to the web app to update it dynamically (marked plain circle 2).

Another advantage of the formal aspect of DEVS formalism is that its structure makes it possible to almost automatically build a graphical interface for interaction with its models. In fact, if a correspondence rule is established between atomic models and user interface elements such as forms, output fields, or graphics, it is possible to propose a basic graphical interface by browsing the specifications of the simulation model.

Figure 8 shows how, from the specifications of a simulation model built by a modeler, it is possible to propose a web app. The specification browsing process is based on templates, making it possible to propose basic graphic diagrams composed, for example, of acquisition fields in a number equivalent to the number of attributes (properties) of the Web Generator model. In the same way, the number of text-type output fields is equivalent to the number of input ports of the Web Collector model, etc. Finally, a button to run/stop the simulation is always inserted in the web app at the level of a navbar. The developer then accesses a composition interface of user interface elements to complete the web app. Of course, there is the possibility of skipping the parsing phase and composing the web interface 'from scratch'.

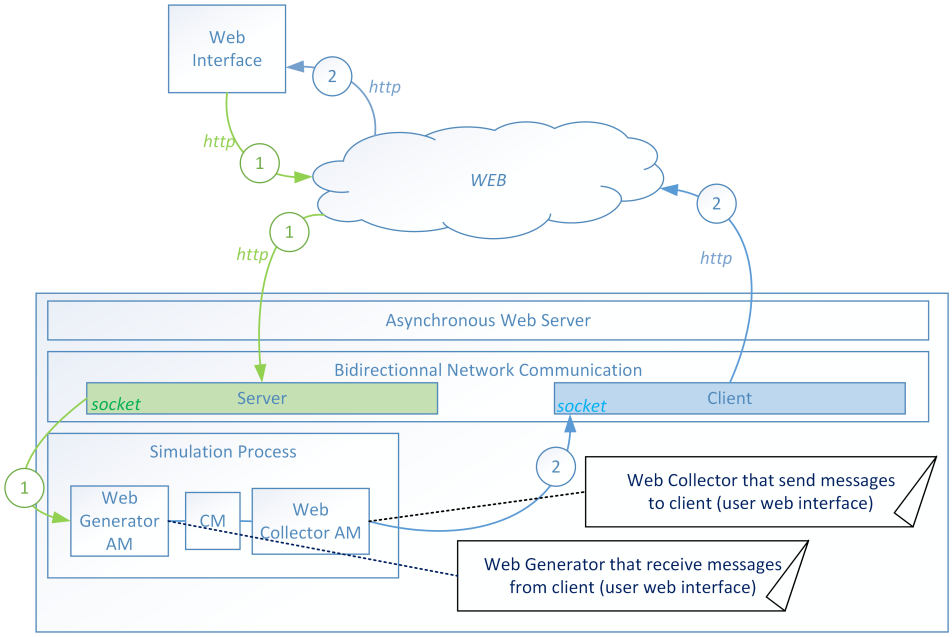

Figure 7. Bidirectional communication between the end-user web app and the simulation process based on the Web Generator and Web Collector DEVS atomic models. (1) Message passing of the input data from the end-user web app to the DEVS coupled model (CM) and (2) the backward simulation results from the Web Collector to the end-user web app.

A web app-builder algorithm is used to generate a web application using design HTML blocks (HTML tags) organized in a smart way by parsing a specification of the simulation model. First, the developer creates an initial web app and has to: (i) give a title (of the simulation, for example) and (ii) give the file corresponding to the DEVS simulation model. Then the web app-builder algorithm generates a first version of the web app according to the content of the DEVS specification file. For example, the web app-builder algorithm transforms all attributes and input messages of the DEVS model Web Generator into HTML input tags (if the Web Generator receives an integer message, the corresponding HTML tags are <input type="number">). Of course, the web app-builder will offer an interface to add, delete, or update these inputs as a result of the parsing.

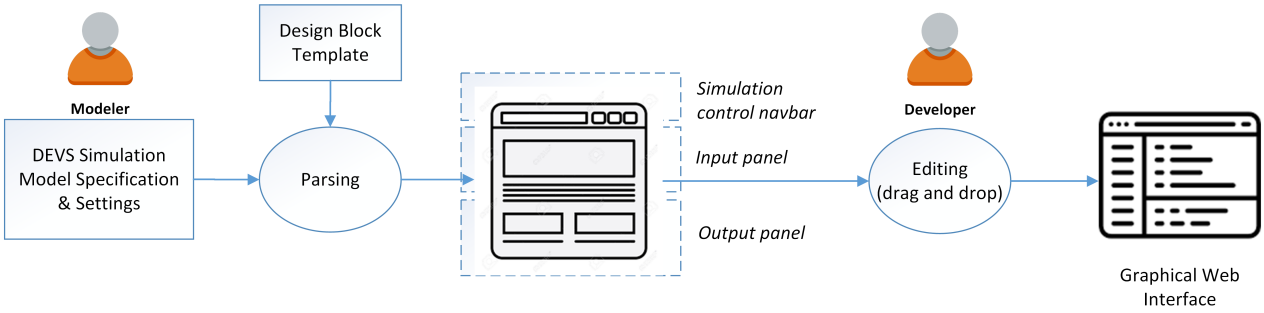

Figure 8. Web interface-building from the DEVS simulation model to graphical web interface (web app). The parsing process uses design block templates to generate the web app in a quasi-automatic way. The developer can refine the web app interface using a back-office editor that allows, among other things, the drag and drop of HTML components. The socket-based JS engine in charge of bidirectional communication does not require adjustment.

Figure 8 gives an overview of the process that enables us to help design an end-user web app. The first step consists of a parsing process that results in a first draft of an end-user web app. The parsing process takes as input both the DEVS simulation model specification file corresponding to the DEVS simulation model created in the previous step and a set

of design block templates defined to guide the user to build a web app associated with a given DEVS simulation model.

Three types of templates are proposed: (i) the simulation control navigation bar composed of internal URL links to navigate inside the web interface; (ii) the input panel with all of the input fields used to get all values embedded in the message dedicated to the Web Generator DEVS model; and (iii) the output panel, with all output fields used to display the values from Web Collector DEVS model inputs (simulation results). The designer can then modify the proposed web application using drag-and-drop elements offered through a web app-builder GUI or edit the code of the web app to customize it more precisely.

6. Discussion

The previous sections point out generic concepts that describe how DEVS formalism could be used to develop the web-based discrete event system M&S. However, these generic concepts have already been instantiated in the Python language through two software implementations: the M&S DEVSimPy environment [37] and the DEVSimPy-mob mobile application [38] as a software suite for the simulation of discrete events based on the web from a mobile application. Due to these implementations, the DEVS models defined and validated through simulation using the DEVSimPy framework are remotely accessible via the DEVSimPy-mob mobile app. DEVSimPy is not a web-oriented environment, as DEVS-MMS would be, but it allows for generating a standalone version of a simulation model (as presented in Figure 5) that can be used to build a web app. The DEVSimPy-mob mobile application of this simulation model is an example of such a web app. Furthermore, the standalone version of the simulation model presented in Figure 5 could be made up of DEVS-coupled models from DEVS model databases populated using DEVS frameworks such as MS4Me [39] or DEVS suite simulator [40].

In [38], the generic concepts presented above are validated using the following:

- The use of web services that allow a user to calibrate models before and during their simulation;
- The DEVSimPy-mob mobile application that allows us to simulate DEVSimPy models defined in a DEVS experimental framework from real data that can come, for example, from platforms embedding sensors;
- The following functionalities associated with the DEVSimPy-mob app are defined: (1) the possibility of sending events to DEVS models in order to interact directly with these models through their functions of external transition; (2) the opportunity to interact graphically with DEVS models; and (3) the possibility to suspend simulations, modify the simulated models, and resume the simulation process.

Future directions should be associated with the development of the DEVS-MMS collaborative web environment to build simulation models of cyber-physical systems such as IoT systems and generate the corresponding web applications as proposed in [41], where a library of DEVSimPy components has been proposed that allows users to model and simulate the behavior of connected objects. Since DEVSimPy models are associated with real objects via the IP address, we can manage connected objects through DEVS simulations. Furthermore, DEVS models are accessible over the internet using a DEVSimPy web server. We have also developed a generic mobile application that allows us to: (1) select a DEVS model to be simulated; (2) perform the simulation; and (3) visualize the results obtained. Interest in the use of simulation models of IoT systems accessible through web services is not recent. In [42], the authors question the impact that the use of these services can have in relation to the modeling methodology used. They conclude by noting that the combination of the web and simulation will surely lead to a change in the way we approach the modeling of complex systems. On the other hand, in [43] the authors emphasize the importance of M&S based on the use of web services, but also the arrival of ubiquitous simulations and the challenges of the interaction of simulation tools at the user level. Today, it seems obvious to offer tools to the scientific community for modeling and simulating ubiquitous systems

through web services. This approach then makes it possible to integrate the simulation as a service accessible using mobile devices (smartphones) or to integrate mobile devices (or sensors and embedded components) as a source of data for the simulation [44]. In recent years, tools have appeared, including M&S environments of ubiquitous systems or networks of connected objects [45].

Furthermore, one has to envision using DEVS formalism in order to manage discrete event simulations from web or mobile applications using DEVS models associated with connected objects such as board computers, sensors, controllers, or actuators using IA approaches. The result will be the ability to manage connected objects (sensors, computer boards, actuators, and controllers) using DEVS while providing intelligent decisions based on simulations that integrate IA features such as machine learning [46] and, more specifically, reinforcement learning [47].

7. Conclusions

This paper presents how the generic concepts involved in DEVS formalism combined with recent web concepts and techniques such as cloud computing, application virtualization, asynchronism, and WebSocket communication have been used to propose the main lines for web application managers of DEVS simulation models. The proposed approach, called DEVS-MMS, allows modelers, analysts, and developers to collaborate around a web-based DEVS M&S model management system with the goal of generating a set of web applications corresponding to a given DEVS simulation model and dedicated to end-users.

A set of DEVS-based web building blocks is provided in order to allow an end-user to start a simulation of a given DEVS simulation model from a web app, to populate the model through the web app, and to display the results on the web app. Furthermore, a set of basic design elements of a web application have been defined in order to guide the developer to automatically generate a web application associated with a DEVS simulation model. Bidirectional communication between an end-user and the simulation model is proposed and is based on the use of (i) DEVS design block/pattern concepts and implementation, (ii) an asynchronous HTTP client/server, and (iii) object–network communication.

A set of implemented software modules stemming from existing M&S software (the DEVSimPy environment and the DEVSimPy-mob mobile application) can be involved to achieve the goal defined by the DEVS-MMS approach. The current work concerns the development of new web-oriented software modules based on the previous ones. Furthermore, IoT modules could be integrated into the DEVS-MMS platform with the aim of facilitating the design and deployment of IoT systems through mobile apps. Future work should cover handling complex cyber-physical systems involving interconnections of connected objects such as sensors, actuators, and board computers, including features of artificial intelligence such as neural networks, fuzzy inductive modeling, machine learning, and so on. DEVS-MMS can integrate such IA components, since DEVS formalism is able to easily embed such IA features using a building-block approach and design pattern capabilities.

Author Contributions: Conceptualization, L.C. and J.F.S.; methodology, J.F.S.; software, L.C.; writing—original draft, L.C. and J.F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
App	Application
CMS	Content Management System
DEVS	Discrete Event system Specification
DEVS-MMS	DEVS Model Management System
EF	Experimental Frame
IoT	Internet of Things
MMS	Model Management System
M&S	Modeling and Simulation
MSaaS	Modeling and Simulation as a Service
REST	Representational State Transfer
SoS	System of Systems

References

1. Wang, S.; Wainer, G. Modeling and Simulation as a Service Architecture for Deploying Resources in the Cloud. *Int. J. Model. Simulation, Sci. Comput.* **2016**, *7*, 1641002. https://doi.org/10.1142/S1793962316410026.
2. St-Aubin, B.; Yammine, E.; Nayef, M.; Wainer, G. Analytics and Visualization of Spatial Models as a Service. In Proceedings of the 2019 Summer Simulation Conference; Society for Computer Simulation International: San Diego, CA, USA, 22-24 July 2019; SummerSim'19; pp. 39:1–39:12.
3. Bordón-Ruiz, J.; Besada-Portas, E.; López-Orozco, J.A. Cloud DEVS-based computation of UAVs trajectories for search and rescue missions. *J. Simul.* **2022**, *16*, 572–588, https://doi.org/10.1080/17477778.2022.2053311.
4. Zeigler, B. DEVS-Based Building Blocks and Architectural Patterns for Intelligent Hybrid Cyberphysical System Design. *Information* **2021**, *12*, 531. https://doi.org/10.3390/info12120531.
5. Belkhir, A.; Abdellatif, M.; Tighilt, R.; Moha, N.; Guéhéneuc, Y.; Beaudry, E. An Observational Study on the State of REST API Uses in Android Mobile Applications. In Proceedings of the IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems, Montreal, QC, Canada, 25 May 2019; pp. 66–75. https://doi.org/10.1109/MOBILESoft.2019.00020.
6. Segura, S.; Parejo, J.A.; Troya, J.; Ruiz-Cortés, A. Metamorphic Testing of RESTful Web APIs. In Proceedings of the IEEE/ACM 40th International Conference on Software Engineering, Gothenburg, Sweden, 27 May 27–3 June 2018; pp. 882–882. https://doi.org/10.1145/3180155.3182528.
7. Kumar, A.; Kumar, A.; Hashmi, H.; Khan, S.A. WordPress: A Multi-Functional Content Management System. In Proceedings of the 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 10th–11th December, 2021; pp. 158–161. https://doi.org/10.1109/SMART52563.2021.9675311.
8. Andahi, A. *Drupal: Content Management Framework*; CreateSpace Independent Publishing Platform: North Charleston, SC, USA, 2018.
9. Marriott, J.; Waring, E. *The Official Joomla!* 2nd ed.; Addison-Wesley Professional,: Boston, USA, 2013.
10. Patel, S.K.; Rathod, V.; Parikh, S. Joomla, Drupal and WordPress - a statistical comparison of open source CMS. In Proceedings of the 3rd International Conference on Trendz in Information Sciences & Computing (TISC2011), Chennai, India, 8th-9th December, 2011; pp. 182–187. https://doi.org/10.1109/TISC.2011.6169111.
11. Shivakumar, S.K., Basics of Content Management Systems. In *Enterprise Content and Search Management for Building Digital Platforms*; London: Wiley-IEEE Press, 2017; pp. 82–103. https://doi.org/10.1002/9781119206842.ch3.
12. Shivakumar, S.K., Content Management System Architecture. In *Enterprise Content and Search Management for Building Digital Platforms*; London: Wiley-IEEE Press, 2017; pp. 104–153. https://doi.org/10.1002/9781119206842.ch4.
13. St-Aubin, B.; Menard, J.; Wainer, G. A Web Based Modeling and Simulation Environment to Support the DEVS Simulation Lifecycle. In Proceedings of the 2021 Annual Modeling and Simulation Conference (ANNSIM), Virtual, 19–22 July 2021; pp. 1–12. https://doi.org/10.23919/ANNSIM52504.2021.9552123.
14. Wainer, G.; Wang, S. MAMS: Mashup architecture with modeling and simulation as a service. *J. Comput. Sci.* **2017**, *21*, 113–131. https://doi.org/10.1016/j.jocs.2017.05.022.
15. Procházka, D.; Hodický, J. Modelling and Simulation as a Service and Concept Development and Experimentation. In Proceedings of the International Conference on Military Technologies, Brno, Czech Republic, 31 May–2 June2017; pp. 721–727. https://doi.org/10.1109/MILTECHS.2017.7988851.
16. Cayirci, E. Modeling and Simulation as a Cloud Service: A Survey. In Proceedings of the Winter Simulations Conference, Washington, DC, USA, 8–11 December 2013; pp. 389–400. https://doi.org/10.1109/WSC.2013.6721436.
17. Zehe, D.; Knoll, A.; Cai, W.; Aydt, H. SEMSim Cloud Service: Large-scale urban systems simulation in the cloud. *Simul. Model. Pract. Theory* **2015**, *58*, 157–171. https://doi.org/10.1016/j.simpat.2015.05.005.
18. Cayirci, E.; Karapinar, H.; Ozcakir, L. Joint military space operations simulation as a service. In Proceedings of the 2017 Winter Simulation Conference, Las Vegas, NA, USA, 3–6 December 2017; pp. 4129–4140. https://doi.org/10.1109/WSC.2017.8248121.

19. Bocciarelli, P.; D'Ambrogio, A.; Giglio, A.; Paglia, E. Model Transformation Services for MSaaS Platforms. In Proceedings of the Model-Driven Approaches for Simulation Engineering Symposium; Society for Computer Simulation International: San Diego, CA, USA, Baltimore Maryland, April 15-18,2018; pp. 12:1–12:12.
20. Bocciarelli, P.; D'Ambrogio, A.; Mastromattei, A.; Giglio, A. Automated Development of Web-Based Modeling Services for MSaaS Platforms. In Proceedings of the Symposium on Model-Driven Approaches for Simulation Engineering, Virginia, VA, USA, 23–26 April 2017; Society for Computer Simulation International: San Diego, CA, USA, 2017; Mod4Sim'17.
21. Shahin, M.; Babar, M.A.; Chauhan, M.A. Architectural Design Space for Modelling and Simulation as a Service: A Review. *J. Syst. Softw.* **2020**, *170*, 110752. https://doi.org/https://doi.org/10.1016/j.jss.2020.110752.
22. Byrne, J.; Heavey, C.; Byrne, P. A Review of Web-based Simulation and Supporting Tools. *Simul. Model. Pract. Theory* **2010**, *18*, 253–276. https://doi.org/https://doi.org/10.1016/j.simpat.2009.09.013.
23. Shaikh, B.; Marupilla, G.; Wilson, M.; Blinov, M.L.; Moraru, I.I.; Karr, J.R. RunBioSimulations: An extensible web application that simulates a wide range of computational modeling frameworks, algorithms, and formats. *Nucleic Acids Res.* **2021**, *49*, 597–602, https://doi.org/10.1093/nar/gkab411.
24. Uran, S.; Jezernik, K. MATLAB Web Server and M-file Application. In Proceedings of the 12th International Power Electronics and Motion Control Conference, Portorož, Slovenia, 30 August–1 September 2006; pp. 2088–2092.
25. Anvil Development Team. Anvil—Build Web Apps with Nothing But Python. Available online: https://anvil.works/ (accessed on 15 January 2023).
26. Streamlit Development Team. Streamlit—A Faster Way to Build and Share Data Apps. Available online: https://streamlit.io/ (accessed on 15 January 2023).
27. Amazon Honeycode Development Team. Amazon Honeycode—Build Web & Mobile Apps Without Writing Code. Available online: https://www.honeycode.aws/ (accessed on 15 January 2023).
28. ElBatanony, A.; Succi, G. Towards the No-Code Era: A Vision and Plan for the Future of Software Development. In Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code; Association for Computing Machinery: New York, NY, USA; Chicago, Illinois, USA, 17-22 October, 2021,pp. 29–35. https://doi.org/10.1145/3486949.3486965.
29. Zeigler, B.P.; Sarjoughian, H.S.; Duboz, R.; Souli, J.C. *Guide to Modeling and Simulation of Systems of Systems*, 2nd ed.; Springer Publishing Company, Inc.: New York, NY, United States, 2017.
30. Bernard P., Z.; Capocchi, L.; Santucci, J.F. PDEVS Protocol Performance Prediction using Activity Patterns with Finite Probabilistic DEVS. In Proceedings of the Spring Simulation Multi-Conference (SpringSim'16), Pasadena, CA, USA, 3–6 April 2016.
31. Middleton, N.; Schneeman, R. *Heroku: Up and Running*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
32. Bankar, S. Cloud Computing Using Amazon Web Services. *Int. J. Trend Sci. Res. Dev.* **2018**, *2*, 2156–2157.
33. Gupta, A.; Goswami, P.; Chaudhary, N.; Bansal, R. Deploying an Application using Google Cloud Platform. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications, Bangalore, India, 5–7 March 2020; Institute of Electrical and Electronics Engineers, Inc.: New York, USA, 2020; pp. 236–239. https://doi.org/10.1109/ICIMIA48430.2020.9074911.
34. Ater, T. *Building progressive web apps: Bringing the power of native to the browser*; O'Reilly Media, Inc.: Sebastopol, California, USA, 2017.
35. Carver, M. *The Responsive Web*, 1st ed.; Manning Publications Co.: Shelter Island, NY, USA, 2014.
36. Negus, C. *Docker Containers (Includes Content Update Program): Build and Deploy with Kubernetes, Flannel, Cockpit, and Atomic*, 1st ed.; Prentice Hall Press: Englewood Cliffs, NJ, USA, 2015.
37. Capocchi, L.; Santucci, J.F.; Poggi, B.; Nicolai, C. DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems. In Proceedings of 20th IEEE International Workshops on Enabling Technologies, Paris, France, 27–29 June 2011; Reddy, S.; Tata, S., Eds.; Institute of Electrical and Electronics Engineers, Inc.: Los Alamitos, CA, USA, 2011; pp. 170–175. https://doi.org/10.1109/WETICE.2011.31.
38. DEVSimPy-mob. Available online: https://github.com/capocchi/DEVSimPy_mob (accessed 10 October 2019).
39. Chungman Seo, Bernard Zeigler, R.C.; Kim, D. DEVS Modeling and Simulation Methodology with MS4Me Software. In Proceedings of the Theory of Modeling & Simulation Symposium, SpringSim Multi-Conference, San Diego CA, USA, 7–10 April 2013; SpringSim '13.
40. Fard, M.D.; Sarjoughian, H.S. A Restful Persistent Devs-Based Interaction Model For The Componentized Weap and Leap Restful Frameworks. In Proceedings of the 2021 Winter Simulation Conference, Phoenix, AZ, USA, 12–15 December 2021; Kim, S.; Feng, B.; Smith, K.; Masoud, S.; Zheng, Z.; Szabo, C.; Loper, M., Eds.; Institute of Electrical and Electronics Engineers, Inc.: Piscataway, NJ, USA, 2021; pp. 1–12. https://doi.org/10.1109/WSC52266.2021.9715348.
41. Capocchi, L.; Kessler, C.; Santucci, J.F. Discrete-event Modeling and Simulation of Ubiquitous Systems with DEVSimPy Environement and DEVSimPy-mob Mobile Application. In Proceedings of the Winter Simulation Conference, Crystal Gateway Marriott, Arlington State: (USA) Virginia, 11-14 December 2016; IEEE Press: Piscataway, NJ, USA, 2016; pp. 3716–3717; WSC;16.
42. Page, E.H.; Buss, A.; Fishwick, P.A.; Healy, K.J.; Nance, R.E.; Paul, R.J. Web-Based Simulation: Revolution or Evolution? *ACM Trans. Model. Comput. Simul.* **2000**, *10*, 3–17. https://doi.org/10.1145/353735.353736.
43. Taylor, S.J.E.; Khan, A.; Morse, K.L.; Tolk, A.; Yilmaz, L.; Zander, J. Grand Challenges on the Theory of Modeling and Simulation. In Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium; San Diego, California, April 7-10, 2013 Society for Computer Simulation International: San Diego, CA, USA, 2013; DEVS 13.

44. Campillo-Sanchez, P.; Serrano, E.; Botía, J.A. Testing Context-Aware Services Based on Smartphones by Agent Based Social Simulation. *J. Ambient Intell. Smart Environ.* **2013**, *5*, 311–330.
45. Lavirotte, S.; Tigli, J.Y.; Rocher, G.; El Beze, L.; Palma, A. A Dynamic Visual Simulation Environment for Internet of Things, 2015. Available online: https://hal.science/hal-01187315/ (accessed on 24 January 2023).
46. Alpaydin, E. *Machine Learning: The New AI*; The MIT Press: Cambridge, Massachusetts, USA, 2016.
47. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; A Bradford Book: Cambridge, MA, USA, 2018.