A WEB BASED MODELING AND SIMULATION ENVIRONMENT TO SUPPORT THE DEVS SIMULATION LIFECYCLE

Bruno St-Aubin
Jon Menard
Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON, CANADA
staubin.bruno@gmail.com
jonmenard@cmail.carleton.ca
gwainer@sce.carleton.ca

ABSTRACT

The Discrete Event System Specification (DEVS) is a hierarchical and modular simulation formalism that can represent an unlimited range of real-world systems. Its flexibility comes at a cost: increased complexity when developing simulation models, less model reusability, and lowered adoption rates. In addition, the DEVS software environment is generally fragmented and siloed. Tools have been developed to support model debugging or simulation trace visualization but, they are often closely coupled to simulators and consequently their usefulness is limited. We present a web-based modeling and simulation environment that is meant to alleviate some of the pressure points faced by DEVS users across the simulation lifecycle. The environment provides a library of reusable models meant to favor sharing models and collaboration, a workflow-based approach for modeling large scale systems, front-end resources in the form of an API to develop case-specific simulation-based applications and a set of web resources to facilitate the simulation lifecycle management.

Keywords: Modeling and simulation environment, simulation lifecycle, visualization, and analysis

1 INTRODUCTION

Modeling and simulation (M&S) is widely used to study real world systems and to support decision-making. Through an abstraction of the system, it is possible to conduct experiments that require less resources and involve less risk than studying the system itself. It is a way to evaluate new or upgraded systems without compromising limited resources, interrupting operations, compromising safety, etc. However, conducting accurate simulation experiments is a complex, multi-step process that can be difficult and time-consuming. This is particularly true when new models or simulators must be built. It requires extensive domain knowledge and a deep understanding of the simulation method used. Because of the significant effort involved in understanding a specific domain and building the required models, the tendency in the field is to build single use simulators centered on a narrow domain of application.

Examples of single-use simulators abound. *EnergyPlus,* first released in 2001, is well-established and one of the most used simulation engines for building energy simulation. It provides a variety of features for engineers and architects to simulate heating, cooling, ventilation, lighting, and other aspects of energy usage in buildings (Crawley et al. 2001; EnergyPlus 2021). *DIALux* is a software suite for planning, calculating

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

and visualization of indoor and outdoor lighting, it boasts a user base of over 700,000 users (DIALux 2021). *Radiance* is also meant to analyze and visualize lighting in design, its creator received the Daylight Award for his work on the software suite (Fuller and McNeil 2017). *OpenFlows FLOOD* is a geospatial simulation software developed by *Bentley Systems* that aims at understanding and mitigating flood risks in urban, riverine and coastal systems (Bentley Systems Incorporated 2020). These types of simulators are well-tailored to the application domain they address and generally offer useful tools directly or through integration with other software that are common to the field.

Generic simulation methods, such as the Discrete Event System Specification (DEVS), are not tailored to specific domains of application and therefore are well positioned to break down disciplinary silos. DEVS supports hierarchical and modular model development, and it favors model reuse. It is based on a rigorous formalism to manage inputs and outputs of models. It has been shown that the DEVS formalism can be used as a common denominator for any formal method of modeling, whether discrete or continuous (Vangheluwe 2000). Yet, it sees only limited usage in the industry and outside the dedicated research community due to several barriers to entry. Compounding the issue is the fact that these obstacles are generally not the focus of the research by the community. Thus, the available modeling and simulation environment, particularly for non-expert users, does not adequately support the lifecycle of DEVS based applications. Over the past 10 years, panels of researchers in the field have recurrently raised issues that warrant further study and that contribute to the lack of adoption for generic simulation methodologies:

1. Fujimoto identified the need to lessen the burden on the modelers by reducing the modeling efforts required through model composability. It should be possible to compose existing models of subsystems into an accurate model of a wider system (Fujimoto 2016). Morse had a similar opinion in (Taylor et al. 2015).

2. There is a lack of adequate tools and environments to support collaboration over the entire simulation lifecycle, including experiment management, results analysis, refinement, feedback process, etc. (Taylor et al. 2012; Taylor et al. 2013). Taylor proposed *Web Simulation Science* and *Modeling & Simulation Ecosystems* as fields of study for "creating the theories, methods and technologies needed for the successful realization of large-scale simulations and M&S projects."

3. Modeling and simulation as a service (MSaaS) has been consistently identified as a necessary step towards increased accessibility to simulation in decision-making processes (Taylor et al. 2012; Taylor et al. 2015; Yilmaz et al. 2014). Cloud-based MSaaS has been a way to decrease the technological cost of simulation-based processes, to provide access to high performance, scalable simulation hardware and software.

4. The democratization of M&S was also noted as a challenge to be addressed. In (Taylor et al. 2015; Taylor et al. 2013), it was stated that M&S should capitalize on the fact that technical skills are increasingly prevalent in all research fields. Therefore, researchers require adequate tools to empower them to collaborate across multiple disciplines. Uhrmacher and Page also advocated the development of tools to support the entire lifecycle of simulation as a way to democratize M&S (Taylor et al. 2012).

Based on these ideas, we propose a Modeling and Simulation Environment to support all categories of users, expert and non-expert, across the entire DEVS simulation lifecycle. This architecture seeks to fill the gaps between the steps of the lifecycle by addressing several concrete issues in the modeling, simulation, analysis, and visualization phases. Although some components of this architecture are currently in various stages of implementation, this paper mainly focuses on the concept rather than its implementation.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

2 BACKGROUND & RELATED WORK

2.1 The Discrete Event System Specification (DEVS)

The Discrete Event System Specifications (DEVS) is a simulation methodology derived from systems theory. It is meant to model and simulate discrete-event dynamic systems and was first described by Bernard Zeigler in 1976 (Zeigler, Praehofer, and Kim 2000). It provides a discrete event-based method to abstract systems into models that can be used for experimentation in cases where it is impractical or impossible to experiment on the real system. It supports modular and hierarchical modeling and therefore, favors their reusability and composition into coupled models representing larger and more complex systems. It is a generic modeling and simulation method; any system can be modeled with an infinite set of possible states where the new state, triggered by an event, may depend on the time elapsed in the previous state (Wainer 2009). DEVS can be seen as a common denominator for formal modeling methods, whether they are based on a discrete or continuous representation of time (Vangheluwe 2000; Vangheluwe, Lara, and Mosterman 2002).

The generic nature of DEVS introduces complexity in the modeling process. Indeed, to provide modelers with the flexibility to represent any real-world system, it is mandatory for them to code the behavior functions of a model (internal transition, external transition, output function and time advance) using a programming language. In addition, coding these functions often relies on a simulator specific framework which means that a model prepared for one simulator will not be compatible with another simulator even if both are based on the same programming language. In some cases simulators rely on small languages that help decouple the model from the simulator, this is the case for CD++ Cell-DEVS models for example (Wainer and Giambiasi 2002). This is made possible due to the well-constrained nature of cellular automata models. Otherwise, standardization of models has been attempted only with mitigated success (Wainer et al. 2016). There are also no universally accepted metadata specification for models. Therefore, it remains difficult to assemble a library of properly documented, reusable DEVS models.

In addition to these issues with models themselves, there are also no standards for their outputs. Simulation results are formatted in an *ad hoc* manner that depends on the simulator employed (St-Aubin, Loor, and Wainer 2021). The consequence is that it is more difficult for analysts and decision-makers to extract meaning from simulation results since there are no common tools available that can interpret them. The result of this is a fragmented, heterogeneous Modeling and Simulation Environment for DEVS: there is no toolchain that can support a user through the entire simulation lifecycle, from modeling to decision-making.

2.2 DEVS Tools to Support the Simulation Lifecycle

Researchers often propose tools to support the modeling step to lessen the burden on modelers. One of the more popular way to do this is to guide them through the modeling process using a visual programming language. CoSMoS for example, is one such tool (Sarjoughian and Elamvazhuthi 2010). CoSMoS Users can manually connect boxes representing individual models to compose a larger coupled model. They select logical models from a database, then the system automatically translates the structural elements of the model but, modelers must still code the model behavior in Java. DesignDEVS is a more recent example of a visual graphic user interface meant to support the modeling process (Goldstein, Breslav, and Khan 2018). In a similar manner, this software allows modelers to pick pre-implemented models from a list, configure them through a UI and assemble them into a larger coupled model. It also provides debugging tools such as a timeline of output messages and a summary visualization capacity for cellular automata models. It provides a mechanism to alter the behavior of models using LUA, a simple scripting language. There are many other examples of this, for example CD++ Builder (Bonaventura, Castro, and Wainer 2013), PowerDEVS (Bergero and Kofman 2011), etc.

Although these Modeling and simulation environments effectively support the simulation lifecycle, they are built as monolithic software: they combine modeling, simulation, and visualization into a single application. Users must remain within those applications to conduct the entire simulation experiment. This

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

locks them into a specific simulator, and they have no possibility of customizing visualizations or analyses to cater the needs of a scenario specific decision-making process.

2.3 DEVS Simulation Mashups

Mashups have also been proposed to provide more flexibility in the development of DEVS simulation applications. In (Wainer and Wang 2017; Wang and Wainer 2015), the authors describe MAMSaaS, a Mashup Architecture with Modeling and Simulation as a Service to support the rapid development of modeling and simulation mashup applications. It relies on the RESTful Interoperability Simulation Environment (RISE) platform (Al-Zoubi and Wainer 2015) to offer simulation as a service to users and a visual interface that allows users to build simulation applications using of pre-built GUI components. These components tie in to existing third party web resources such as Google Maps, YouTube, or Wikipedia. MAMSaaS is essentially an application generator: once users have completed the configuration, they can generate an application that is ready for deployment.

Although the mashup solution is a good way for non-simulation experts and non-programmers to rapidly develop web applications, we believe that it is not suited to the modern web experience: it reduces the expressive potential of simulation-based applications. Users building applications with it are restricted to the components that are available within the mashup environment, which limits possibilities. The automatic generation of applications is still worth pursuing, as shown by the increasing popularity of software like Power BI (Microsoft 2021) or the ArcGIS Experience Builder (ESRI 2021) that are respectively used to generate data visualizations and web GIS applications.

2.4 Web Based Environments in Other Fields that Support Complex Business Processes

Loosely coupled web-based environments have had a major impact on technology adoption by the public especially with complex subject matter that can be difficult for non-expert users to understand. Geographic information systems (GIS) are a good example of this. The mid 1990s saw the apparition of the first web GIS applications, and the 2000s saw a proliferation of web-based GIS applications and a refinement of the environments underlying them, enough so that GIS is now “in the home of millions and in the hands of billions” (Fu and Sun 2010). Modern web GIS applications are developed by leveraging a variety of web-enabled components. They are typically built using a client-server architecture where the more computationally expensive operations are offloaded to the server. In a web GIS application, the server is often tasked with distributing large volumes of data to the front-end application in a manner that optimizes the user experience. In some cases, the server will send vector data when the volume allows it but, it can also pre render maps before sending them to the client as images, for cases where the volumes of data are too high. Through the server, applications can also conduct complex spatial analysis operations such as determining the shortest path to a destination or spatially intersecting layers of data. The environment generally also provides a front-end API that abstracts the complexity of communication with the backend. These environments have reached a level of maturity where it is even possible to interchange data formats, front-end APIs, GIS servers or the underlying database technology.

Even more remarkable is the fact that multiple GIS application generators have been made available to the public. Users can employ an intuitive graphic user interfaces to build complex applications without entering a single line of code. They can, for example, create story maps and dashboards to support high-level decision making processes (ESRI 2021). These more advanced tools are made possible by the robust and flexible technologies that support them. This phenomenon also exists outside the world of GIS. The field of data analytics and visualization for example, has recently seen similar advancements. Tools like Microsoft PowerBI allow users to easily build web based applications in a similar manner (Microsoft 2021). In this case, the environment provides a way for analysts to publish and host statistical datasets, retrieve their data through a web interface and to present the analyses through their end-users through a browser GUI, also built using an application generator.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

In the following section, we propose a concept that follows the design of these web-based environments. It consists of a set of loosely coupled, server-side and client-side, components that can be used independently or as a whole. The intent is that a lower-level approach will provide more flexibility than monolithic simulation software based on visual programming interfaces or mashup applications. Such an environment is a necessary preliminary step before a more useful simulation application generator approach can be successfully used. Through a rich web-based set of tools, developers will be able to quickly craft solutions that are tailored to specific decision-making scenarios. In turn, end users of those application will have a better experience with DEVS models and simulations. A complete toolchain to support the DEVS simulation lifecycle will contribute to increased adoption of the formalism as a simulation methodology.

3 A WEB-BASED MODELING AND SIMULATION ENVIRONMENT TO SUPPORT THE ENTIRE DEVS SIMULATION LIFECYCLE

In this section, we describe an environment meant to support users throughout the entire DEVS lifecycle, from modelling the system under study for simulation to the development of simulation-based applications for decision-making. Throughout the discussion that follows, the expression "simulation-based application" is employed to mean any type of application that requires the modelling and simulation of a system, analysis of simulation results and the visualization of simulation models and results. An example of this would be a web-based platform that supports users in the development of models related to energy policy planning at a municipality scale, that provides an easy way to simulate them, and the analytical capability for comprehensive decision-making.

3.1 Categories of Users

In most use case scenarios, several users of variable technical and analytical proficiency will interact with the environment in one way or another. To contextualize the environment, we propose categories of users based on their role within a project and their level of knowledge with regards to simulation, software development and data analysis. Of course, not every user can be neatly assigned to one specific category. Depending on their background or their role within an organization, a user may fit more than one category. We propose the four following categories of users:

1. **Model developers** are focused on the development of simulation models. They are familiar with the simulation formalism but lack knowledge in software development and data analysis. They require features to speed up model development, debug models, generate model code from domain-specific data, organize and document models, reproduce experiments, etc.

2. **Data analysts** aim to analyze simulation results and extract patterns and meaning. They are knowledgeable of data analysis techniques but unfamiliar with simulation formalisms and software development. They require tools to clean up log files, identify patterns and build analytical products for subsequent decision-making.

3. **Software developers** build simulation-based software to support an organization's operations within a specific domain of application. They are proficient in software development but lack knowledge in simulation formalisms and data analysis. They require features that will support them in development tasks by abstracting the complexity inherent to simulation and data analysis.

4. **Decision-Makers** are the end users of simulation-based software. They lack knowledge in simulation formalisms, software development and data analysis. They are concerned with efficiently making accurate decisions based on well-presented simulation results. Therefore, they require that simulation results be presented in an intuitive, expressive, and comprehensive manner, through clear reports, data analytics, infographics, and interactive simulation visualization.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

3.2 Overview of the Integrated DEVS Architecture

The goal of our architecture environment is to eliminate some of the pressure points in the DEVS simulation lifecycle and ultimately, improve adoption rates for DEVS as a simulation formalism. A high-level overview is presented in Figure 1, where we show how the categories of users contribute to the simulation lifecycle through the environment.

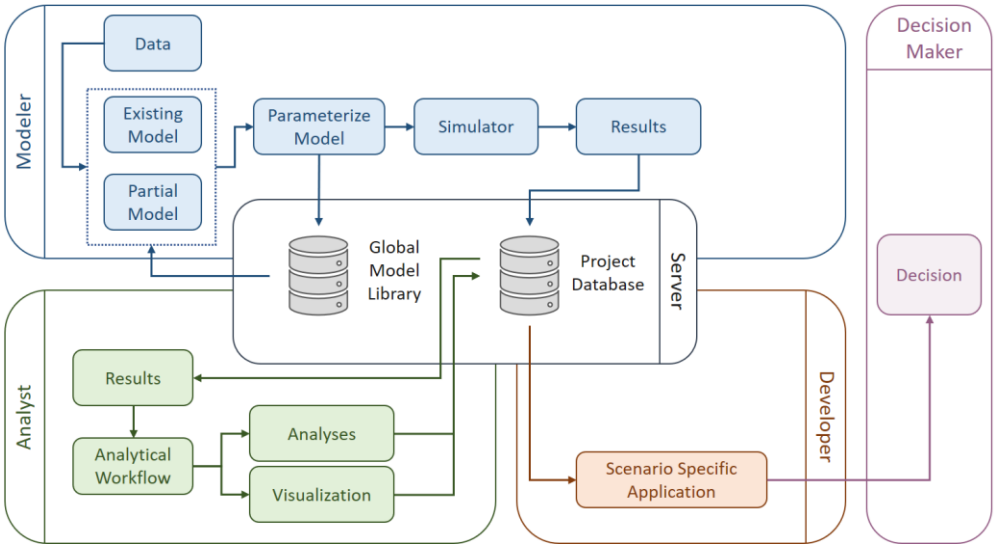

Figure 1. A high-level overview of the DEVS environment proposed.

The modeler's task is centered on developing a formal model of a real-world system and implementing it for simulation. The environment assumes a component-based, data-centric approach to model development: existing models are mapped onto data. There are three key principles that must be followed. First, models must be discoverable, retrievable, and modifiable. Second, modified versions of models must return to the library of models so that the library grows over time. Third, models must be developed in a way that they can be easily parameterized. Parametric models in a component-based modeling approach are a reliable way of quickly and automatically generating simulation models. Once a model is completed, documented, and stored in the library of models, it will be possible to run a simulation through a simulation as a service. In this manner, the barrier of entry will be lowered. Generated results are then stored in a project level database and analysts can extract them for further processing and analysis. Raw simulation results tend to be difficult to interpret for many reasons. They are often verbose, contain extraneous data that obfuscates patterns, are stored in inconvenient formats, etc. To solve this issue, the architecture will provide data-centric analytical workflows to process stored simulation results, to convert them into more convenient formats and to extract patterns. They will also be able to construct visualizations from results. Again, workflows, analysis results and visualization configurations will be stored in the project level database for collaborative work or future retrieval. Finally, developers will have access to a front-end API to build simulation-based applications. The goal is to provide tools to tailor applications to the specific needs of a decision-making process. Using this API, developers will be able to leverage the work accomplished by modelers and analysts by presenting the simulation artefacts (models, results, analysis, workflows, visualizations, etc.) in an intuitive and comprehensive manner to decision-makers.

3.3 A Library of Reusable Models

DEVS boasts high reusability; however, for various reasons, it is uncommon for models to be reused. One reason is that research teams lack a common platform to document and share models. We therefore propose an implementable conceptual data model for thorough documentation of models and experiments. A better documentation process and a more structured organization of DEVS models within a library of models (LoM) would foster collaboration and encourage model reusability. It is also a way to increase model

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

discoverability through metadata: users will be able to search for models, simulation projects they were used in, simulation artefacts that were generated or, any associated documentation using search parameters such as domain of application, author, or tags. The LoM adopts crowd-sourcing principles. Modelers will be able to retrieve models, modify them according to the requirements of their simulation then, submit them back for others to use subsequently. There are two goals to this component. The first is that over time, the library becomes able to support a fully automated component-based modeling approach through modeling workflows. The second is to help establish a continuous, iterative model improvement process.

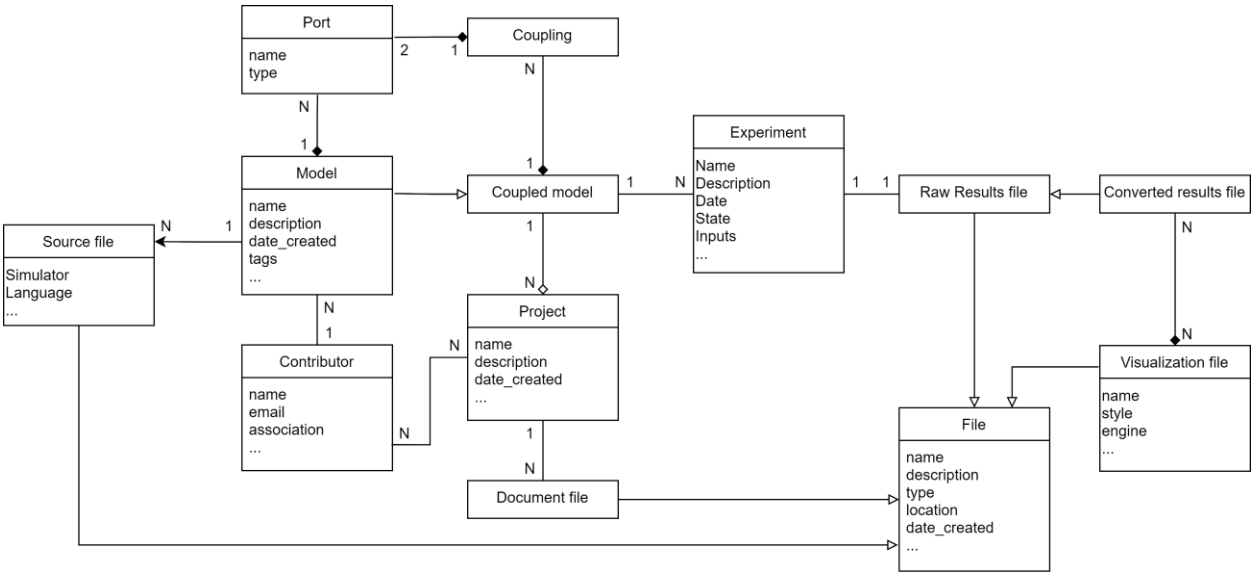

Figure 2. Prototype data model for the LoM.

An abridged conceptual model of the LoM is presented in Figure 2 (with some elements omitted for clarity). The core object in the conceptual data model is a simulation model which can be either atomic or coupled. Models have ports, and coupled models are linked to other models through couplings between model ports. This part of the data model allows the system to reconstruct the structural elements of a model. The main interest of the LoM though, is managing the simulation artefacts related to a model, project, or experiment. A model for example, can have many source files associated to it. Since source files vary depending on the simulator employed, it is important to capture the simulator and language used so that users can retrieve the correct implementation of a model. Experiments are run on a single model and lead to a set of results artefacts for which files are also stored. An experiment can be run on multiple models and one model may be used in multiple experiments. Since the format of raw simulation results are dependent on the simulators used to run the model, the LoM provides a way to convert them to a common specification and store them alongside the raw results. The converted results can then be referenced in a visualization configuration file that will ultimately feed into a front-end visualization platform built using the resources described in section 3.5. The LoM also provides a way to track authorship of models and project participants.

We have undertaken an implementation of the LoM concept using the PostgreSQL relational database management system. Through web services, users can contribute to the LoM; namely, they can set up projects, assign contributors to them, create models and upload sources files. Users can also submit raw results files, convert them to a well-specified format and configure visualizations that use them. The visualization files are compatible with the DEVS WebViewer, a lightweight simulation visualization platform built using the resources described in section 3.5. To provide model discoverability, a front-end application was built using Java Spring and Thymeleaf. It allows users to explore the models within the LoM and their metadata as well as some of their simulation traces. It also allows users to retrieve any of the files associated to models and projects.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

3.4 Workflows for Assisted Modelling

One of the main obstacles for DEVS modelers is the complexity involved in the development of simulation models. Although mechanisms can be employed to automatically generate structural elements of a DEVS model (atomic model instances, coupled models, their ports, and couplings), their behavior functions (internal and external transition, time advance and output function) are another matter. Indeed, solutions that try to simplify the definition of these functions typically fall short one way or another. For example, A modeler building a Cell-DEVS model in CD++ can use a non-programming language to define cell behavior (Wainer 2002) but, this is only applicable to cellular models. In DesignDEVS, modelers can define behavioral functions using LUA (Goldstein, Breslav, and Khan 2018), a simpler programming language but, it still requires that modelers be skilled programmers. Component-based modeling where parametric models are used to represent components of a real-world system, offer an attractive, automatable way of building large-scale, complex models.

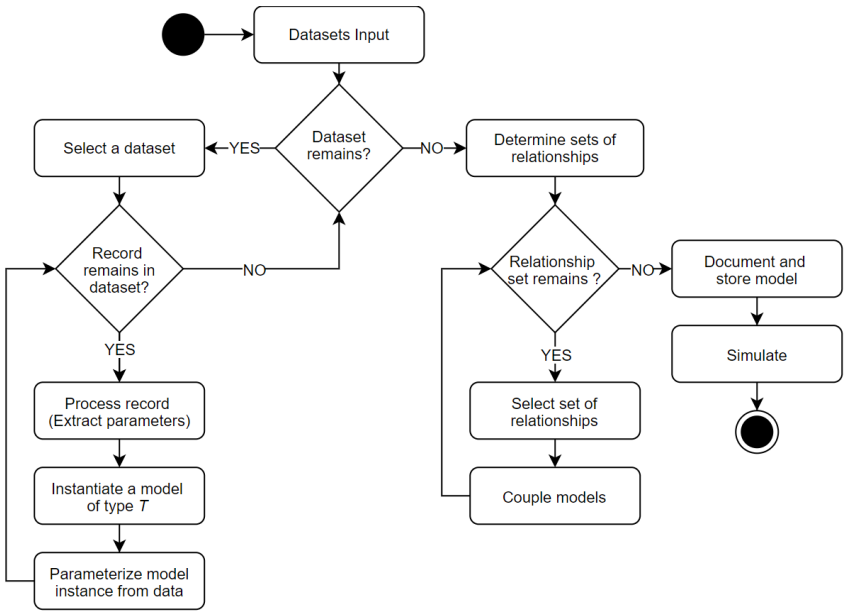

Figure 3. Conceptual high-level representation of a modeling workflow.

We propose a process where modelers define a case specific workflow that relies on component-based modeling to construct a model without writing code. By assembling a series of analytical steps, models are instantiated, parameterized and couplings are determined. A general representation of the process is shown in Figure 3. Modelers first provide datasets; then, individual records in each dataset are processed. For each record, a model type is determined from configuration provided by the modeler and then instantiated and parameterized from the data. The second part consists of establishing relationships between models from the data, and then coupling them accordingly. To do this, the modeler provides a configurable analytical workflow tailored to the data being processed. There are different ways to implement this while minimizing the burden on the modeler. In the data science field for example, visual programming workflow tools such as KNIME (KNIME Inc. 2021) or PowerBI (Microsoft 2021) are commonplace. The final part is to document and store the newly made model in the LoM such that it can be shared and reused more easily. Effectively, modelers can build large and complex models by assembling a series of common data analysis steps. This requires that parametric models be available in the LoM and that a data analysis and manipulation functions be available. As such, this process is better suited for application domains where rich data sets and well-established analytical tools are available.

A good example of rich data that is naturally conducive to this approach is GIS data. Indeed, each record of a data layer can typically be associated to an instance of a model: a building, a road segment, or a river for example. Model parameters can be established from attribute data and spatial relationships can be used

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

to derive the links between models. Distances between models, topological relationships, geostatistics and other types of spatial analyses can be considered to determine model couplings. As a preliminary proof of concept, we used a subdivision of space, dissemination areas (DA), as a data source to be mapped onto a purposely built toy model that considers population counts to communicate with its 4 closest neighbors. A DA is the smallest standard geographic area for which all census data (population count, number of dwellings, etc.) are disseminated (Statistics Canada 2018). Each DA was coupled to its 4 closest neighbors and a weight value was assigned considering the Euclidean distance between polygon centroids. This is a very simple scenario with obvious fallacies regarding spatial analysis but, it served our initial purpose of evaluating the methodology. Once the workflow was designed, it was easy to extend it to different units of space such as provinces for example. An overview of the workflow is presented in Figure 4 and a visual representation of the simulation is shown in section 3.5 (Figure 5).

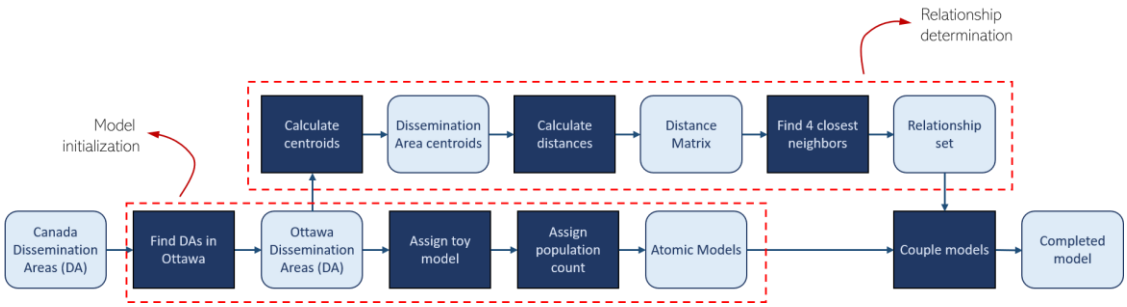

Figure 4. Case specific analytical workflow to initialize models and determine relationships.

3.5 Front-end Resources for Visualization and Analysis

To support the development of applications tailored to specific decision-making processes, we propose a front-end API to interact with the various resources of our DEVS *modeling and simulation environment*. Contrary to approaches such as mashups or application generators which lock application builders in a constrained framework, this approach grants greater expressive potential to developers at the expense of requiring them to be skilled in web development. The rationale for this is two-fold. One, we aim to support specialized decision-making processes which may have requirements that cannot be fulfilled without high flexibility, high expressiveness solutions. Two, a well-made front-end API is a lower-level steppingstone towards higher-level solutions such as mashups and application generators as have been discussed in section 2.4. The API is a natural evolution of (St-Aubin et al. 2019) where we presented the architecture of the DEVS WebViewer visualization platform.

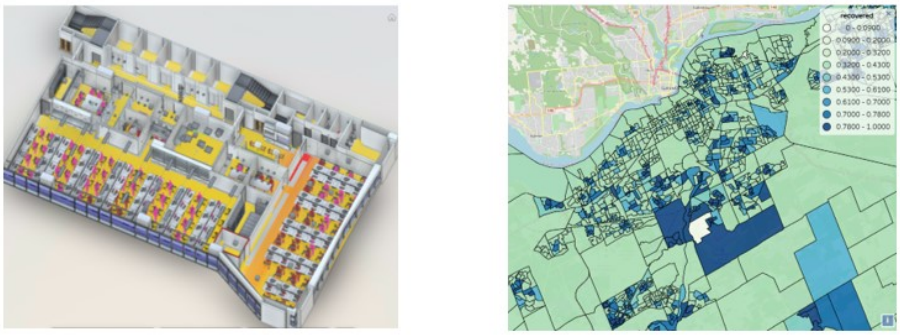

Figure 5. Visualizations based on a Building Information Model and GIS.

In its current form, the API revolves around a data structure acting as a container to hold a simulation trace. It requires that results be provided following a well-defined specification that is meant to capture all the elements of a simulation that are required for its visualization. The specification is flexible enough to consider simulation results originating from DEVS, Cell-DEVS and even, from other simulation

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

formalisms such as Petri-Nets. A full description of the specification can be found in (St-Aubin, Loor and Wainer 2021). In addition to handling simulation results, it also provides various types of visualization capabilities (grid, graph, 3D model, map based), user interaction with the results and, tools to query the server-side simulation lifecycle management web API. Several ready-made UI components are available such as a playback bar, a visualization settings controller, a component to load simulation results, and others. Developers can import these components in their applications, extend and modify them according to their needs, combine them with other libraries or APIs, etc. This set of front-end resources is being used successfully to build simulation applications tailored to different application domains as shown in Figure 5.

4 CONCLUSION

We introduced a concept for a web based, DEVS modeling and simulation environment that seeks to address some of the major grand simulation challenges identified in the past 10 years: model development burden, lack of adequate tools, modeling, and simulation as a service and, democratization of the discipline. The environment supports modelers, analysts, software developers and decision-makers across the complete DEVS simulation lifecycle, it contains the following components:

- A library of reusable models meant to organize and thoroughly document simulation models and their components with the goal of increasing reusability and collaboration in simulation projects.
- Data-centric workflows to allow modelers to quickly develop large-scale simulation models from data sources relevant to their application domain.
- Front-end resources in the form of an API that contains data structures, tools, and UI components to allow software developers to build applications tailored to a decision-making process.

This environment is partially developed. Proofs of concepts and prototypes have been built for data-centric workflows, for the library of model (LoM) and for lifecycle management web resources. Front-end resources for application development are in an advanced stage of development, they have been used to develop multiple visualization and analysis products. Some of the future work will focus on further implementation and integration into a system that guides the user through the whole lifecycle as intended by the environment. We also plan to evaluate the possibility of using an existing metadata specification as a model documentation strategy. For example, the Discovery Metadata Specification for M&S Resources currently being developed by the Simulation Interoperability Standards Organization (SISO 2021).

DEVS, as a simulation formalism, is in a good position to break down the disciplinary silos that tend to form in the field of simulation. Due to its generic, modular, and hierarchical nature, it has the flexibility required to model an unlimited scope of real-world models. Flexibility and genericity have a cost: developing models involves a level of complexity that is often prohibitive for non-expert users. In other disciplines, GIS and data science for example, well-integrated environments have been key to their adoption and spread and therefore, their industrial success. Such an environment does not yet exist in the field of DEVS based simulation and would undoubtedly contribute to a higher adoption rate for the methodology.

ACKNOWLEDGMENTS

The authors would like to acknowledge Compute Canada for providing access to their Arbutus cloud-based infrastructure that allowed us to develop and test the implementation of the work we presented. This research has been partially supported by NSERC. Some of the implementation work was done by Jon Jacob Laboissoniere, Paul Roode and Chang Qiu.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

Al-Zoubi, K., and G. Wainer. 2015. "Distributed Simulation of DEVS and Cell-DEVS Models Using the RISE Middleware". *Simulation Modelling Practice and Theory* vol. 55, pp. 27–45.

Bentley Systems Incorporated. "OpenFlows FLOOD". https://www.bentley.com/en/products/product-line/hydraulics-and-hydrology-software/openflows-flood. Accessed Mar. 21, 2021.

Bergero, F., and E. Kofman. 2011. "PowerDEVS: A Tool for Hybrid System Modeling and Real-Time Simulation". *Simulation* vol. 87 (1–2), pp. 113–32.

Bonaventura, M., R. Castro, and G. Wainer. 2013. "Graphical Modeling and Simulation of Discrete-Event Systems with CD++Builder". *Simulation* vol. 89 (1), pp. 4–27.

Crawley, D. B., L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesend, D. E. Fishere, M. J. Wittef, and J. Glazer. 2001. "EnergyPlus: Creating a New-Generation Building Energy Simulation Program". *Energy and Buildings* vol. 33 (4), pp. 319–31.

DIALux. "DIALux - DIAL". https://www.dial.de/en/dialux/. Accessed Mar. 21, 2021.

EnergyPlus. "EnergyPlus | EnergyPlus". https://energyplus.net/. Accessed Mar. 21, 2021.

ESRI. "Experience Builder System". https://www.esri.com/en-us/arcgis/products/arcgis-experience-builder/overview. Accessed Mar. 21, 2021.

Fu, P., and J. Sun. 2010. *Web GIS: Principles and Applications*. Edited by ESRI Press.

Fujimoto, R. M. 2016. "Research Challenges in Parallel and Distributed Simulation". *ACM Transactions on Modeling and Computer Simulation* vol. 26 (4), pp. 1–29.

Fuller, D., and A. Mcneil. 2017. "Radiance — Radsite". https://www.radiance-online.org/, Accessed Mar. 21, 2021.

Goldstein, R., S. Breslav, and A. Khan. 2018. "Practical Aspects of the DesignDEVS Simulation Environment" . *Simulation* vol. 94 (4), pp. 301–26.

KNIME Inc. "KNIME | Open for Innovation". https://www.knime.com/. Accessed May 19, 2021.

Microsoft. "Data Visualization | Microsoft Power BI". https://powerbi.microsoft.com/en-us/. Accessed Mar. 21, 2021.

Sarjoughian, H. S., and V. Elamvazhuthi. 2010. "CoSMoS: A Visual Environment for Component-Based Modeling, Experimental Design, and Simulation". In *2nd International ICST Conference on Simulation Tools and Techniques*, pp. 1–9.

SISO – Simulation Interoperability Standards Organization. "DMS-MSR PDG". https://www.sisostds.org/. Accessed May 18, 2021.

St-Aubin, B., E. Yammine, M. Nayef, and G. Wainer. 2019. "Analytics and Visualization of Spatial Models as a Service". In *Proceedings of the 2019 Summer Simulation Conference*, pp. 1–12.

St-Aubin, B., F. Loor, and G. Wainer. 2021. "A specification for multi-simulator, multi-formalism visualization and analytics on the web." *Simulation Practice and Theory* (Submitted).

Statistics Canada. "Dissemination Area: Detailed Definition". https://www150.statcan.gc.ca/n1/pub/92-195-x/2011001/geo/da-ad/def-eng.htm. Accessed Mar. 21, 2021.

Taylor, S. J. E., A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, and J. Zander. 2013. "Grand Challenges on the Theory of Modeling and Simulation". *Proceedings of the Symposium on Theory of Modeling &*

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.

Simulation, pp 1–8.

Taylor, S. J. E., R. Fujimoto, E. H. Page, P. A. Fishwick, A. M. Uhrmacher, and G. Wainer. 2012. "Panel on Grand Challenges for Modeling and Simulation". *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, pp. 1–15. Berlin, Germany.

Taylor, S. J. E., A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, J. Zander, and P. J. Mosterman. 2015. "Grand Challenges for Modeling and Simulation: Simulation Everywhere—from Cyberinfrastructure to Clouds to Citizens". *Simulation vol.* 91 (7), pp. 648–665.

Vangheluwe, H. L. M. 2000. "DEVS as a Common Denominator for Multi-Formalism Hybrid Systems Modelling". In *IEEE International Symposium on Computer-Aided Control System Design*, pp. 129–34.

Vangheluwe, H. L. M., J. D. Lara, and P. J. Mosterman. 2002. "An Introduction to Multiparadigm Modelling and Simulation". In *AI, Simulation and Planning in High Autonomy Systems,* pp. 9–20.

Wainer, G. 2002. "CD++: A Toolkit to Develop DEVS Models". *Software - Practice and Experience* vol. 32 (13), pp. 1261–1306.

Wainer, G. 2009. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. Edited by Pieter J. Mosterman. Taylor & Francis.

Wainer, G., K. Al-Zoubi, D. R. C. Hill, S. Mittal, J. L. R. Martín, H. Sarjoughian, L. Touraille, M. K. Traoré, and B. P. Zeigler. 2016. "An Introduction to DEVS Standardization". *Discrete-Event Modeling and Simulation: Theory and Applications*, pp. 393–425.

Wainer, G., and N. Giambiasi. 2002. "N-Dimensional Cell-DEVS Models". *Discrete Event Dynamic Systems* vol. 12, pp. 135–57.

Wainer, G., and S. Wang. 2017. "MAMS: Mashup Architecture with Modeling and Simulation as a Service". *Journal of Computational Science* vol. 21, pp. 113–31.

Wang, S., and G. Wainer. 2015. "A Simulation as a Service Methodology with Application for Crowd Modeling, Simulation and Visualization". *Simulation* vol. 91 (1), pp. 71–95.

Yilmaz, L., S. J. E. Taylor, R. M. Fujimoto, and F. Darema. 2014. "Panel: The Future of Research in Modeling & Simulation". *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, pp. 2797–2811. Savannah, GA.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2nd ed. Academic Press, San Diego, CA.

AUTHOR BIOGRAPHIES

BRUNO ST-AUBIN is a Ph.D. student at the Department of Systems and Compute Engineering at Carleton University in Ottawa. He holds a Master's degree in Geomatics Sciences from Laval University in Quebec. His research interests lie in simulation lifecycle management and simulation visualization, particularly for large scale, spatial scenarios. His email address is staubin.bruno@gmail.com.

JON MENARD is a B.Eng. student at the Department of Systems and Compute Engineering at Carleton University in Ottawa. His email address is jonmenard@cmail.carleton.ca.

GABRIEL WAINER is a Professor at the Department of Systems and Computer Engineering at Carleton University. He is a Fellow of the Society for Modeling and Simulation International (SCS). His email address is gwainer@sce.carleton.ca.

Authorized licensed use limited to: Carleton University. Downloaded on April 14,2022 at 15:19:16 UTC from IEEE Xplore. Restrictions apply.