



A simulation as a service methodology with application for crowd modeling, simulation and visualization

Sixuan Wang and Gabriel Wainer

Abstract

Crowd modeling and simulation (M&S) has been used to support the analysis of the behavior of crowds, in order to predict the impact of pedestrian movement and to test design alternatives. In recent years, crowd M&S has become more complex, and new technologies such as CAD (computer-aided design) and BIM (building information modeling) authoring tools are being used to support the process. There are challenges in adopting these technologies due to the lack of automation and integration of these tools for crowd M&S. We propose a method based on a distributed architecture with simulation in the cloud, and composition using workflows. In particular, we adopt a model-driven engineering approach to extract data from CAD/BIM authoring tools, Cell-DEVS theory for crowd modeling, simulation as a service to execute simulation remotely, and three-dimensional visualization. Finally, we present a case study for crowd evacuation, discussing the advantages of the proposed architecture. We show the advantages obtained when using distributed deployment, simulation-based design and collaborative development and we discuss how this facilitates the crowd behavior study and improves reusability in crowd M&S.

Keywords

Crowd modeling and simulation, simulation as a service, cloud computing, model-driven engineering, building information modeling, workflow

1. Introduction

In recent years, the population in public areas and transportation facilities has become much denser. This situation has potentially increased problems in terms of human safety. Numerous incidents and accidents with crowds have been recorded recently;¹ thus, predicting and trying to control the behavior of a crowd is an important issue.

Modeling and simulation (M&S) can support the analysis of such behavior, and it can help designers to evaluate the performance of their designs. Designers can use crowd M&S to find flaws in particular areas of their buildings or urban area designs before the construction has begun. In particular, crowd modeling aims to define the behavior of a crowd in a given area, while crowd simulation focuses on executing one such model in order to predict the impact of crowd movement under different situations.²

Crowd modeling is complex due to numerous factors: physical (position, direction, speed, etc.), psychological (stress, panic, etc.), social (culture, regulations, flocking, etc.) and others.³ These factors are closely related to the area where the crowds are studied (e.g., traffic intersections, shopping malls). To better study the behavior of a

crowd in the design of a building or city section, we need to use a large amount of data (for instance, the layout information of a traffic intersection could be used as input for a crowd model). Nowadays, new technologies and tools are available to manage this kind of data. For instance, building designers use CAD (computer-aided design) tools to improve the quality of building and urban designs, and to enhance communication with stakeholders. BIM (building information modeling) tools are also used with this purpose: a BIM application usually includes CAD facilities enhanced with varied databases for managing the buildings' data.⁴ The data of these tools can be used as crowd model inputs (e.g., the layout of the building or the coordinates of objects in a facility under study),

Department of Systems and Computer Engineering, Centre for Visualization and Simulation (V-Sim), Carleton University, Canada

Corresponding author:

Sixuan Wang, Department of Systems and Computer Engineering, Centre for Visualization and Simulation (V-Sim), Carleton University, Ottawa, ON K1S5B6, Canada.
Email: swang@sce.carleton.ca

and the tools can be used for three-dimensional (3D) visualization, taking advantage of advanced visualization of the crowd simulation results.⁵ By doing this, the designers can efficiently view, analyze and refine the design with different scenarios.⁶

At present, crowd M&S lacks integration with advanced CAD/BIM tools. When a designer wants to evaluate a building design, they need to extract data from the building design, build a crowd model with this data, execute simulation experiments, and study the simulation results; however, there is no method to integrate all the activities. As discussed by Zhou et al.,⁷ this is a complex and important problem to solve. Although there are various tools to support the activities mentioned above, they cannot be integrated: each tool has its own functionality, syntax, semantic and structure, which makes the communication between them hard. For instance, a simulator can record state changes, while a visualization tool cannot animate these changes due to the lack of some information needed for visualization that the simulator does not provide. Most of the recent work in this area has focused on two-part integrations of these activities, and they are neither formalized nor automated. Various efforts have focused on the integration of crowd models and their simulations.⁷⁻⁹ Likewise, crowd simulation and visualization has received some attention.^{10,11} In contrast, the integration of CAD and crowd M&S is still limited^{12,13} and the integration with BIM tools has been rarely investigated.¹⁴

The goal of this research is to provide an architecture to allow a better integration of crowd M&S with CAD/BIM tools and visualization. To do so, we need to deal with the following issues.

- 1) We need to automate the use of CAD/BIM to provide model inputs. Many CAD/BIM tools use standard files based on the industry foundation classes (IFCs). However, this standard is too complex and redundant, and files are difficult to understand.¹⁵ Furthermore, the standard cannot cover all the information needed for specific simulations.¹⁶
- 2) Crowd M&S itself is complex due to not only the factors mentioned earlier, but also in terms of performance. As the model size becomes larger, the simulation execution time becomes unfeasible for doing proper analysis.⁷
- 3) Usually, a crowd model is associated with a specific simulator running on a standalone workstation. Running new simulations needs installation and configuration of the simulator. This requires simulation expertise, which makes it difficult for casual users to execute new crowd models.
- 4) The integration process involves different activities. However, it is complex for building designers to link the solutions of these activities and to reproduce the whole process.

To deal with these issues, we defined an integrated architecture to combine crowd M&S, CAD/BIM tools and 3D visualization. The architecture includes three layers: a component layer (to provide different components focusing on different aspects of the crowd modeling process), a composition layer (to define workflows to integrate those components) and an application layer (to allow users building crowd M&S applications easily).

In order to address the issue of the redundant information found in the IFC files, we adopt a model-driven engineering (MDE) approach. MDE can help users defining domain-specific models (DSMs) for particular simulation purposes, as well as supporting automatic or semi-automatic model transformations. We propose a MDE-based semi-automatic data collection method to query the information needed from IFC files, and to generate the initial files for crowd models.

For modeling crowds, we used various models built using Cell-DEVS theory,¹⁷ a rule-based, event-driven, formal mechanism to define these behaviors. Cell-DEVS helps dealing with modeling complexity.

Simulation as a service (SimaaS) is a method to deploy simulation resources (e.g., simulators, models, input data) in the cloud, using web services (WSs) for users to access these resources. Users can share crowd models, reproduce the execution of crowd simulations and retrieve the simulation results remotely. In order to match the actual computational demands, SimaaS also allows the users to scale up or down the underlying compute and storage capacities according to their needs.

In order to integrate and automate the whole process, we use workflows. A workflow links different efforts mentioned above and reproduces the whole process.

In order to show the uses of the proposed architecture and the methods we defined, we present a case study for evacuation in a shopping mall. We show two ways of collecting data from BIM tools. We also discuss the steps to build a crowd model using Cell-DEVS theory, as well as executing the simulation using SimaaS. Furthermore, we show how to integrate the simulation results with 3D visualization. Finally, we show how to automate the execution of the whole process.

The following sections focus on presenting guidelines and solutions for the integration of crowd M&S and CAD/BIM tools. The proposed architecture has several advantages as follows.

- It supports a distributed way to deploy components (e.g., data collection, 3D visualization). It allows deploying these components in the cloud and using workflows to automate the whole process. Users can deploy their own components and link them as parts of a workflow.
- It supports simulation-based design of buildings or urban areas. The designers can repeat and modify

existing workflows to test different designs, enabling them to compare alternatives, and to find flaws in the design. This can speed up their decision-making before construction has begun.

- It supports collaborative development of crowd M&S. Users from different disciplines (e.g., building designers, modelers, crowd behavior analysts) can work together. Users can focus on the development of their own components and share them with others, reusing existing components.

In the following sections we will focus on these advantages (i.e., distributed deployment, simulation-based design and collaborative development), showing how the proposed architecture is able to facilitate the behavior study of crowds and improve the reusability in crowd M&S.

2. Related work

Crowd modeling aims to define the behavior of a crowd in an area of interest. The kind of behavior is usually related to the movement of individuals. In Hoogendoorn and Bovy,² the authors presented a three-level theory on the behavior of crowds: the *strategic* level, the *tactical* level and the *operational* level. The strategic level reflects the general decisions for the crowd (e.g., opening more doors during rush hours). The tactical level defines the *time* and *space constraints* of the crowd. The time constraints reflect if the people in the crowd are in a hurry, and the space constraints define how much space they need. The operational level defines concrete rules for the crowd behavior (e.g., where to move, how long to wait).

According to Zhou et al.,⁷ the modeling approaches for the above behavior of crowds can be divided into three categories: flow, agent and entity-based. Flow-based modeling focuses more on the higher levels of the behavior of crowds (e.g., strategic and tactical). It treats the crowd as a whole (like a fluid), and the crowd is modeled using differential equations.^{18,19} For instance, in Werberich et al.,²⁰ the authors presented a crowd model based on friction force equations. Agent and entity-based approaches focus on the operational level of the behavior, seeing each individual as an independent agent whose behavior is controlled by specific laws.^{21,22} For instance, in Bae et al.,²³ the authors developed an agent-based model for a crowd evacuation in a city during a bombardment. Entity-based models separate the crowd into homogeneous groups, and the behavior of each group is defined by a set of rules.²⁴ In agent-based models, each individual is governed by rules that only work for that individual. In contrast, in entity-based models, the crowd is separated into groups, and entities in the same group share common rules, which make it easier for management and extension. Agent-based approaches usually require more resources due to the independent processing for each individual.⁷

In recent years, entity-based approaches have gained much attention. In particular, cellular automata (CA) have been widely used for modeling entity-based crowds. CA divide a physical space into a grid of cells. Each cell changes its value based on the values of its neighbors following predefined rules.²⁵ In Hoogendoorn and Bovy,² the authors divided such rules into three categories: moving direction, collision avoidance and speed adjustment. Blue and Adler²⁶ presented a CA for a unidirectional crowd flow over a multilane walkway. In Song et al.,²⁷ a multi-cell crowd model was presented for a simple evacuation case. In Yue et al.,²⁸ the authors presented a CA with a two-step approach that allows handling collisions when two pedestrians want to move in the same cell. In Henein and White,⁹ the authors argued that the forces that impose on cells (e.g., preferred direction) are essential, and they added vector-based information to each cell. Tao and Jun¹⁰ proposed an improved CA model for bi-directional crowd flow. They believed that the position exchange and side-stepping movement in a bi-directional flow were more complicated than the ones in a unidirectional flow.

In this research, we used the Cell-DEVS formalism¹⁷ as an entity-based approach similar to CA. Cell-DEVS is an extension to the DEVS (discrete-event system specification) formalism²⁹ to model cell spaces. In Cell-DEVS, each cell in the space is represented as a DEVS atomic model that changes state according to the values of its neighbors following the rules defined by a local computing function. Similar to entity-based approaches, Cell-DEVS requires fewer resources and is able to build models easily by sharing rules in groups.^{7,30} Cell-DEVS is event-driven and provides asynchronous execution, ignoring unnecessary processing in cells that are inactive; therefore, it is suitable for large-size and long-term crowd simulations. The formalism supports explicit timing constructions that allow a modeler to define complex timing in each cell (which is not allowed by methods like CA, or it can be complex to define). Being derived from DEVS, it provides a formal M&S specification, which allows one to prove properties about the models and the simulation engines running them. It supports fast prototyping and incremental development. Finally, it has been proven that it can be easily interfaced with other components.¹⁷ The CD++ tool¹⁷ provides an environment to execute DEVS and Cell-DEVS models.

In recent years, various Cell-DEVS models have been developed for modeling different applications in the field of architecture, construction and transportation. In Ahmed et al.,³¹ a Cell-DEVS model of diffusion-limited aggregation showed the integration of construction modeling tools. As a case study, a Cell-DEVS model of the growth of mold in building was implemented, and Autodesk Revit (a BIM application) was used for collecting building data and 3ds Max for visualization. In Hammad and Zhang,³²

the authors used Cell-DEVS models for construction sites, trying to deal with conflict analysis in cranes. In a 2012 study,³³ we introduced the use of Cell-DEVS theory for crowd modeling, showing how to build bi-directional pedestrian models. In our 2013 study,³⁴ an advanced crowd model using Cell-DEVS was presented, taking into consideration random movement and varied speed of the crowds.

In crowd modeling, we need information about the studied area (e.g., floor layout, location of doors, etc.). This data is usually available in CAD/BIM tools. CAD tools describe the data in geometry. If designers want to modify a line, they need to change all the data related to the line. In contrast, BIM views everything as objects, so designers can modify the parameters of the object without changing other data. Although CA and Cell-DEVS can model the behavior of crowds, the integration of these models and CAD tools is limited (and even rare with BIM tools). Most CAD tools do not support this kind of data collection, and designers should do this manually, which is time-consuming and error-prone. In Sano et al.,¹² the authors showed a method to collect room information from a CAD system as input data for a crowd model. However, the rooms that they used are of simple shape. STEPS³⁵ is a micro-simulation tool for crowd movement analysis that supports importing CAD geometry. STEPS is an agent-based approach, and each person defined has specific attributes. In Lo et al.,³⁶ the authors proposed a pre-processing engine to transform spatial information of CAD to the inputs of their spatial-grid evacuation model (SGEM). This engine is complex to use due to the uses of a series of different equations.

Most CAD tools support BIM to create, manage and exchange information. However, to date it has been rare to integrate BIM with crowd modeling. For instance, none of the data collection tools above has been integrated with BIM tools. BIM tools usually use the standard IFCs,^{14,15} which are supported also by many CAD tools. The IFC file covers the core project information, such as building elements and geometric and material properties.³⁷ The current standard, IFC2x3, has 653 entities, 317 property sets and 164 enumerations (at the time of writing, IFC2x4 had just been released). Given the popularity of IFC in CAD/BIM, we need a data collection strategy to extract needed data from IFC files. However, the IFC files are too complicated and redundant, which makes it difficult to query them.³⁸ The IFC is complex, as it intends to cover all design domains in one file, which can include a large number of information from many disciplines. Furthermore, the IFC file might not cover all the information needed for a specific simulation.¹⁶ MDE can help with these issues. Jiang et al.¹⁶ used a MDE approach to parse IFC files into a model for saving building energy. MDE can help users to define DSMs for particular purposes, as well as supporting auto/semi-automatic model transformation.³⁹ In MDE,

data exchanges from a source instance to a target instance. MDE supports tools to define mappings between models; using the mappings and a source instance, MDE also supports tools to generate the target instance.

Once the crowd model is created, and the input data for the area under study is available, we need to execute a number of simulation studies. In most cases, the simulation is run on standalone computers, which means that the users need to install and configure them properly. Instead, using web-based simulation and SimaaS, this is not necessary. Web-based simulation tries to invoke simulation services through the web,⁴⁰ allowing the simulation environment to be shared and reused. Existing simulation-related WSs can be categorized into two classes: RESTful⁴¹ and simple object access protocol (SOAP)-based.⁴² SOAP-based WSs are exposed as in remote procedure calls (RPCs). They encapsulate various procedures on the server that can be invoked like local procedures (described in extensible markup language (XML) web services description language (WSDL) documents). At runtime, the client wraps the SOAP messages in hypertext transfer protocol (HTTP), and POSTs the SOAP message to the server. When the server receives the message, it extracts it using a procedure call. The response of the server is similar. For instance, in Madhoun,⁴³ Madhoun et al.⁴⁴ and Wainer et al.,⁴⁵ we define the first distributed DEVS simulation environment over SOAP and user-controlled light paths (using CaNet*4, Geantoptical networks infrastructure). As discussed by Wagh and Thool,⁴⁶ SOAP WSs have shortcomings compared to RESTful WSs. In the SOAP, client-server interaction is tightly coupled, which means any changes on the server result in a complex code change on the client. Also, the SOAP has a heavier payload as compared to representational state transfer (REST), as it consumes more bandwidth in the XML messages, and it exposes details of internal implementation, which makes them harder to develop. These issues were thoroughly investigated and discussed by Al-Zoubi and Wainer.^{47,48} Based on those studies, and even though there are various SOAP DEVS simulators,⁴⁹⁻⁵² we defined the first RESTful distributed DEVS simulator.⁴⁸

Compared to SOAP, RESTful WSs imitate the web interoperability style, using universally accepted standards, resource-orientation, uniform channels and information hiding. RESTful WSs expose all services as resources with uniform channels, and messages are transferred between those resources through the channels. We can access RESTful WSs via uniform resource identifiers (URIs) with XML messages using HTTP methods. We can view these methods as uniform channels: GET (to read the resource or get its status), PUT (to create or update the resource), POST (to append data to the resource) and DELETE (to remove the resource). Our RESTful distributed simulation is the RESTful interoperability simulation environment (RISE). The main objective of RISE is to support

interoperability of distributed simulations regardless of the model formalisms, model languages or simulation engines.

In recent years, new technologies based on WSs have been introduced, including cloud computing.⁵³ Cloud computing can be used for building a simulation environment in a pay-as-you-go fashion, and computing and storage capacities can be increased or decreased according to current usage. It reduces cost in hardware and IT: a cloud provider provides an efficient and affordable computing infrastructure without needing expensive infrastructures. SimaaS can take advantage of the existing technologies in cloud computing – load balancing, fault tolerance and security – while allowing the users to scale up or down the underlying compute and storage capacities according to their needs.⁵⁴ This allows one to develop SimaaS, taking the advantages of the cloud computing services (e.g., infrastructure as a service [IaaS], platform as a service [PaaS] and software as a service [SaaS]).⁵⁵ Although there is no commonly accepted definition for SimaaS, its main focus is on delivering simulation services in the cloud.

SimaaS can be used for crowd M&S, allowing users to execute crowd models and obtain simulation results remotely. Then, using two-dimensional (2D) and 3D visualization can provide an intuitive way to show the simulation results, allowing one to observe the results better and to predict the impact of the crowd movement. At present, due to the limitations in integration of tools, using 3D visualization in CAD/BIM tools for crowd M&S is also limited. In Tao and Jun,¹⁰ a CA-based model was integrated with a 3D virtual reality engine. A similar visualization tool was developed in Castonguay and Wainer¹¹ using Blender. In Ahmed et al.,⁵⁶ we built a prototype interfacing virtual reality software and distributed simulation. However, efforts like these ones used general-purpose tools, and the users needed to adapt them to build their visualizations.⁵⁷

In summary, at present the integration efforts among CAD/BIM tools, crowd modeling, simulation and visualization are limited. In Wang et al.,³³ we presented an integrated framework combining BIM tools and CD++ for a test case focusing on building evacuation simulation. In Wang et al.,⁵⁸ we presented a revised scalable integration framework, taking into consideration the IFC standard and RISE. However, the integration of these components was not formalized or automated, which makes it hard to reproduce the whole process. Some authors proposed to do this by using workflows to link components, by connecting their inputs and outputs. In Rybacki et al.,⁵⁹ the authors adapted the concepts of business process modeling (BPM) and workflows to formalize the M&S process. In Ribault and Wainer,⁶⁰ the authors used Taverna,⁶¹ which provides a compact set of functional modules for building workflows, in order to automate the simulation process. In Ciciirelli et al.,⁶² the authors presented a new workflow approach with timing constraints by using time stream Petri nets (TSPNs). However, none of the above workflows has

considered the integration of crowd M&S with CAD/BIM tools and visualization.

3. An architecture for defining and integrating crowd modeling applications

Assume a construction project that needs to study a crowd scenario in which building designers of a public area (e.g., a railway station) want to evaluate the behavior of the crowd during peak hours. In the area under study, the flow of the crowd is bi-directional; people tend to move forward, but they can change direction if blocked, and can sidestep to avoid collisions.² Let us also assume that the building designers have used BIM tools for their design, and want to integrate the crowd model to improve decision-making. We want to use the BIM floor plan as model inputs, and the BIM tools to visualize simulation results.

As discussed earlier, there are many issues in this process: the redundancy of CAD/BIM files, the complexity of the crowd models, the configuration of the simulator, repeatability, reuse, etc. As discussed in Section 2, there is no well-established method to integrate, formalize and automate all these activities. In order to deal with these issues, we propose the following layered architecture.

The architecture presented in Figure 1 focuses on integrating the different aspects of the crowd M&S with CAD/BIM tools, and it is organized in three layers as follows.

- 1) The *component layer* is responsible for deploying user-developed components in the cloud. We need components for each of the activities required in the crowd M&S process, specifically the following.
 - a) *Cell-DEVS modeling*: it is used for building crowd models using Cell-DEVS theory. Currently various Cell-DEVS crowd models are available (e.g., bi-directional, unidirectional, multi-floor) and they can be expanded and modified easily.
 - b) *Data collection*: it is used to extract data from the designs. This is done automatically using MDE techniques. It can retrieve information from CAD/BIM tools to a DSM instance. Then, this instance can generate inputs for the crowd model.
 - c) *SimaaS*: it is used to run crowd simulation experiments using WSs. We use RESTful WSs for building and executing crowd simulations remotely.
 - d) *3D visualization*: it is responsible for visualizing the results. We defined a mechanism for parsing the simulation results and for visualizing them directly in CAD/BIM tools. This visualization component can be easily customized.

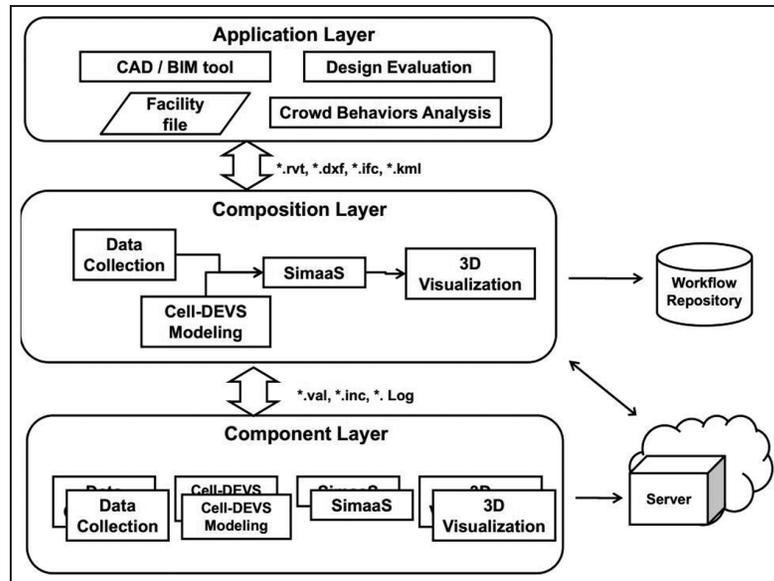


Figure 1. Integrated three-layer architecture for crowd modeling and simulation. CAD: computer-aided design; BIM: building information modeling; DEVS: discrete-event system specification; SimaaS: simulation as a service; 3D: three-dimensional.

These components are independent from each other, and they use well-defined interfaces to interconnect. This layer also helps the users to orchestrate components in the cloud, allowing them to be reusable and accessible. The cloud server supports the necessary hardware dependencies for these components, allowing them to be ready to use and composed.

- 2) The *composition layer* is responsible for defining workflows to formalize the integration of components and automate its execution, as discussed earlier. Users can easily replace components and modify the workflow for their particular purposes. Besides, these workflows can be stored in a workflow repository, allowing users to share and reuse them.
- 3) The *application layer* is responsible for collaboration in a simulation-based design process, allowing designers to build crowd M&S easily. The designers do not need to care about the workflow details: they need to choose a workflow, and to provide a CAD/BIM design file as the workflow input. Then, they wait for the end of the workflow and visualize the results. In addition, they can analyze the crowd simulation results, evaluate their design and restart the process.

This architecture involves three kinds of users as follows.

1. The *multi-domain providers* are specialists from different disciplines who can provide alternative components. These are software engineers building low-level components as SimaaS.

2. The *workflow engineers* are experts for designing and managing the workflows of the components. We have defined varied workflows using Taverna, and other engineers can change them.
3. The *designers* are end users, who start the execution of a workflow and analyze the behaviors of crowds. These are the target users of the architecture.

The layered architecture divides the responsibilities of the integration task into different levels. Each kind of user can focus on their own task, and it makes easy for them to collaborate. The architecture provides the following advantages.

- *Distributed deployment*, which is done by using the proposed component-oriented approach and cloud computing. Components hide their design and implementation details, providing their functionalities using well-defined interfaces. As the architecture deploys these components in the cloud, the users do not need to worry about hardware installation and computational limitations. Section 4 will discuss more details about these components related to crowd M&S.
- *Simulation-based design* using workflows to integrate crowd M&S with CAD/BIM tools. The workflows automate the integration of the crowd M&S components, including data collection, modeling, simulation and 3D visualization. It also makes it easier to add new components as a part of a workflow (e.g., statistics, verification and validation). The designers can repeat and modify existing workflows to test options, not worrying about the implementation. This enables them to compare alternatives, and

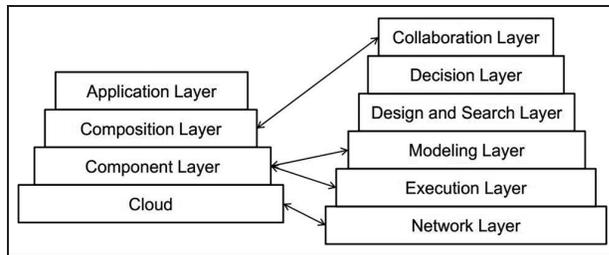


Figure 2. Associating the integrated three-layer architecture with the architecture for modeling and simulation.

improve decision-making before construction has begun. Section 5 will discuss more about the composition of components by using workflows.

- *Collaborative development* of crowd M&S applications. Users from different disciplines (e.g., modelers, builders, architects, workflow engineers, crowd behavior analysts) can work together. Users for the component layer (*multi-domain providers*) can focus on the development of their components and share them. Users for the composition layer (*workflow engineers*) can provide different workflows to combine components by calling their interfaces. Users from the application layer (*designers*) can find and execute workflows to study the behavior of the crowds in a particular area. The case study shown in Section 6 will explain more about this point.

This layered architecture for crowd M&S matches other existing M&S conceptual layers, particularly the architecture for M&S presented by Mittal et al.⁶³ The architecture for M&S divides related M&S aspects into seven layers. Here, we correlate our proposed architecture with it (Figure 2). The *network layer* in the architecture for M&S deals with underlying infrastructure and connecting networks issues. In our proposed architecture, all the components are deployed in the cloud. The cloud supports the underlying infrastructure and reliable network capacities. The *execution layer* in the architecture for M&S deals with the software that executes the models. In our proposed architecture, the execution of models is carried out using the SimaaS services in the *component layer*. The *modeling layer* in the architecture for M&S deals with the development of models. In our proposed architecture, we use Cell-DEVS theory as the modeling formalism to model the behavior of crowds. The Cell-DEVS models work as components and are stored in the cloud. The *collaboration layer* in the architecture for M&S deals with the collaboration and integration to achieve an overall goal. In our proposed architecture, the *composition layer* enables the workflow engineers to define workflows, which can integrate components and automate the

execution of the whole process. In addition, one significant difference of these two architectures is that our proposed architecture can provide applications for the designers. Instead of considering all aspects of the M&S lifecycle, our proposed architecture focuses on the integration of CAD/BIM tools and crowd M&S. It allows users to deploy different components (e.g., data collection and 3D visualization) in the cloud, and compose them using workflows.

4. Cell-DEVS modeling, computer-aided design/building information modeling integration and simulation as a service in the cloud

In this section, we will present more details of each component introduced in Section 3, and we will show how they deal with the issues discussed earlier.

4.1 Crowd modeling using Cell-DEVS

Let us consider the bi-directional crowd scenario we discussed earlier. The area under study is divided into a grid of cells, each covering approximately 0.4×0.4 m² (the typical space in a dense crowd). Each cell can be occupied by a person or an obstacle. Since the crowd is bi-directional, only S/N pedestrians are needed. We consider constant speed (about 1 m/s), so it takes a person 400 ms to cross a cell. Individuals try to move forward avoiding collisions, as seen in Figure 3.

People try to move forward, and change directions if the path is blocked (Figures 3(a)–(e)). Figure 3(b) shows that a person can move to the right or left. To avoid collisions, if more than one person wants to move to the same cell, we decide who can do it. (Figures 3(f)–(h), where two persons want to move to the N of the center. In order to avoid the collision, the person in the center changes direction).

All the behaviors in Figure 3 have been defined as rules in Cell-DEVS. In this case, the states of each cell can be defined as: *empty* (state 0), occupied by an *N pedestrian* (state 1), occupied by an *S pedestrian* (state 2) or an *obstacle* (state 3). In CD++, the behavior of a Cell-DEVS model is defined as a set of rules with format $\langle VALUE \rangle \langle DELAY \rangle \langle PRECONDITION \rangle$ (when the *PRECONDITION* is satisfied, the state of the cell changes to the assigned *VALUE*, which is transmitted after a *DELAY*). Variations in speed can be modeled using different *DELAYS*.³⁴

Let us define the CD++ rules for Figures 3(b) and (c). If the cell to the N is occupied, the N pedestrian will move to a preferred direction (here, if the cell to the E is unoccupied, and no other pedestrian is trying to move to that cell, we move to the E). The corresponding rules are as follows:

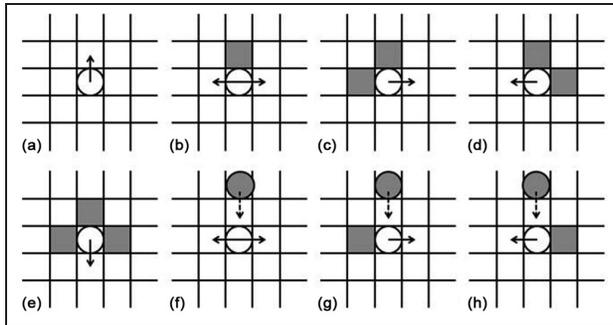


Figure 3. Crowd behaviors: (a) move forward; (b)–(e) change direction; (f)–(h) avoid collision.

Rule : 0 400 { (0,0)=1 and (-1,0)!=0 and (0,1)=0 and (-1,1)!=2 and (1,1)!=1}
 Rule : 1 400 { (0,0)=0 and (0,-1)=1 and (-1,-1)!=0 and (-1,0)!=2 and (1,0)!=1}

The first rule is for an N pedestrian leaving from the current cell (0,0). This rule is executed when cell (-1,0) is not empty, cell (0,1) is empty, and the cells around cell (0,1) do not contain people. The second rule is a symmetric rule for an N pedestrian entering the current cell from cell (0,-1). This rule is executed when the current cell is empty, cell (0,-1) contains an N pedestrian, the cell (-1,-1) is not available, and no other pedestrian is trying to enter the current cell.

Another example can be seen in Figures 3(f) and (g): when an N pedestrian and an S pedestrian are trying to move to the same cell, the N pedestrian will move to the E if the cell to the E is unoccupied, and no other pedestrian is going to take that cell. The corresponding rules are as follows:

Rule: 0 400 { (0,0)=1 and (1,0)=0 and (-2,0)=2 and (0,1)=0 and (-1,1)!=2 and (1,1)!=1}
 Rule: 1 400 { (0,0)=0 and (0,-1)=1 and (-2,-1)=2 and (-1,-1)=0 and (-1,0)!=2 and (1,0)!=1}

The first rule is for a person leaving from the current cell. In this case, a collision could happen if the person in cell (0,0) is an N pedestrian and the person in cell (-2,0) is an S pedestrian, and both of them want to enter cell (-1,0). In order to avoid this collision, the person in cell (0,0) moves to the E. The second rule is for an N pedestrian in cell (0,-1) entering the current cell. This is a rule symmetric to the previous one, and it has similar conditions.

The following figures show some simulation tests based on the rules presented above using CD++. The first example, presented in Figure 4, shows the correct execution of the behaviors (e.g., moving forward and changing direction if blocked).

At the bottom left of Figure 4, an N pedestrian in Cell A faces an S pedestrian in Cell B. However, since another S pedestrian in Cell C wants to move to Cell B, the N

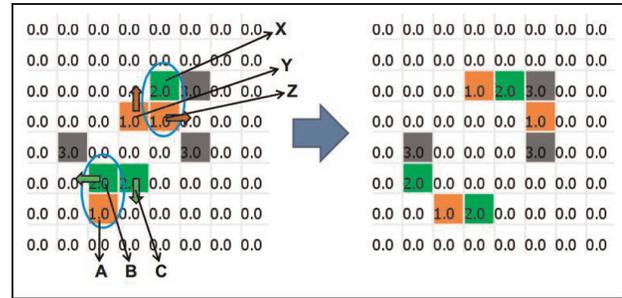


Figure 4. Moving direction scenarios in CD++ of a Cell-DEVS crowd model.

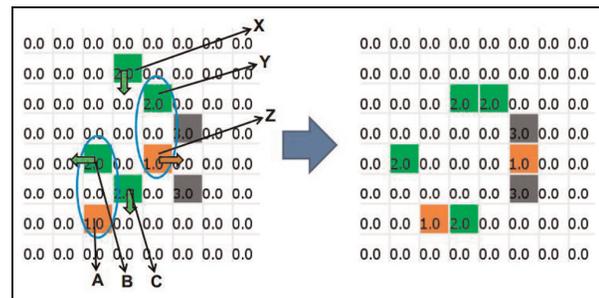


Figure 5. Avoidance collision scenarios in CD++ of a Cell-DEVS crowd model.

pedestrian in Cell A waits. Similarly, in the center of the figure, an S pedestrian in Cell X has space to the W. However, since an N pedestrian in Cell Y wants to take that cell, the S pedestrian in Cell X waits (the rules for moving forward have higher priority than the rules for sidestepping).

The second example presented here shows how collisions are avoided. At the bottom-left of Figure 5, there is a potential collision (the pedestrian in Cell A is going N, and the pedestrian in Cell B is going S). To avoid this collision, they will try to move to their right. For the S pedestrian in Cell B, there is an available cell to the W, so this S pedestrian moves to the W. The N pedestrian in Cell A has an available cell to the E, but there is an S pedestrian in Cell C in the NE. Since Cell C has a higher priority to move forward, the N pedestrian in Cell A waits. A similar situation happens in the upper part of the figure: the S pedestrian in Cell X moves forward to the S, the S pedestrian in Cell Y waits and the N pedestrian in Cell Z moves to the E.

4.2 Data collection from CAD/BIM

In order to start a simulation for the crowd model mentioned above, we need the layout information of the studied area. As we discussed in Section 2, we can collect this kind of information from the IFC files available in most CAD/BIM.

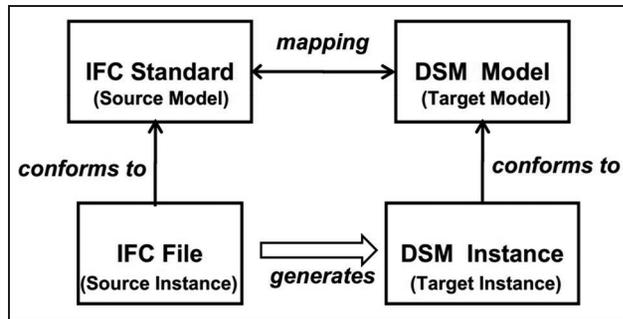


Figure 6. Data exchange among the elements involved in the industry foundation class (IFC) and domain-specific model (DSM).

We adopted a MDE approach to transform information from the IFC file to the DSM for crowd simulation (Figure 6). On one hand, the IFC standard is used as the source model and the IFC file acts as the source instance. On the other hand, a DSM contains the information format for crowd simulations. M&S specialists can design the DSM and the mappings between IFC and DSM. The DSM instance acts as the target instance, which contains the actual information for simulations. The DSM instance can be generated based on the IFC file and the mappings defined.

Based on these ideas, we propose a MDE-based semi-automatic data collection method. The idea is to filter the data needed from the IFC file, use MDE techniques to build a DSM, and finally generate initial files for crowd models (Figure 7). This approach uses the following steps:

- *file collection*: in this step, we query and filter the data needed from the IFC file;
- *building the DSM* for the specific simulation purposes;
- *mapping IFC to DSM*: we define mappings between the IFC standard and the DSM;
- *generating the DSM instance* for specific simulation based on the mappings and the data collected from the IFC file;
- *generating model inputs* for the crowd models using the DSM instance.

This approach resolves some of the IFC file issues of mentioned in Section 2. It deals with the complexity of IFC files by letting users analyze the structure of the IFC and query the elements from an IFC file. In order to handle the mismatches between the IFC standard and DSM, this approach allows users to define the DSMs and specify the mappings between them. The whole process is done semi-automatically: users only need to define the DSM and the mappings (and the rest is automated). In the following, we discuss each step in detail.

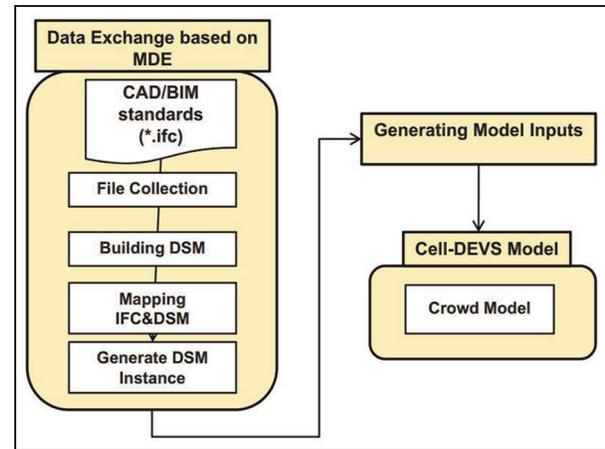


Figure 7. Model-driven engineering (MDE)-based semi-automatic data collection approach. CAD: computer-aided design; BIM: building information modeling; DSM: domain-specific model; IFC: industry foundation class; DEVS: discrete-event system specification.

We first need to query and filter the data needed from a given file. In Wang et al.,⁵⁸ we studied the hierarchical diagram of IFC2x3 (Figure 8). For crowd models, we need to know the layout of the studied area (i.e., walls, doors, obstacles). This kind of information is stored as subclasses of *IfcElement*. Take *IfcWall* as an example: it contains coordinates information in *IfcLocalPlacement* and *IfcProductDefinitionShape*. *IfcLocalPlacement* has the attributes of *IfcAxis2Placement3D* with *IfcCartesianPoint* (coordinates) and *IfcDirection*. *IfcProductDefinitionShape* could contain *IfcPolyline* with the explicit geometric boundary points of its surfaces.

For instance, in Figure 9, *IfcLocalPlacement* represents the starting point, while *IfcProductDefinitionShape* indicates where we can place it (in which direction and how far). Assume a wall of $8'' \times 2'' \times 4''$ starts from (3,2,1) towards direction (-1,1,1); the point diagonally opposite the starting point would then be $(x - \text{width}, y + \text{length}, z + \text{height}) = (1,10,5)$.

There are different tools that allow us to query elements out of the IFC file. We implemented a prototype using *BIMServer.org*,⁶⁴ a model-driven open-source tool to represent the IFC data. For instance, in Figure 9, we can query all the *IfcWall* for each of the floors in the building, obtaining the corresponding (x, y, z) coordinates of the filtered elements. Note that we could query other element types in the same manner (e.g., doors, furniture, windows).

The second step focuses on building the DSM for the specific simulation using MDE techniques. Figure 10 shows an example of such a DSM for a crowd simulation.

All consists of *Variables*, *Configuration* and *Entities*. *Variables* has parameters related to the crowd model, such as *CellUnit* (the size of each cell), while *Configuration*

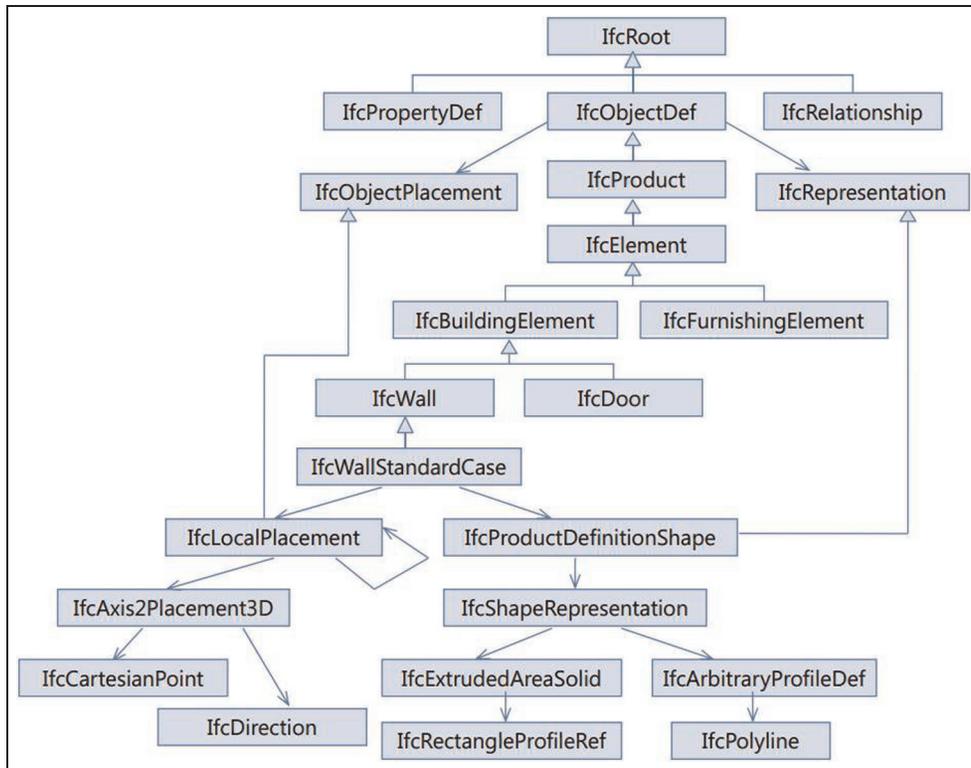


Figure 8. IfcWall hierarchy in industry foundation class standard (adapted from Ahmed et al.⁵⁶).

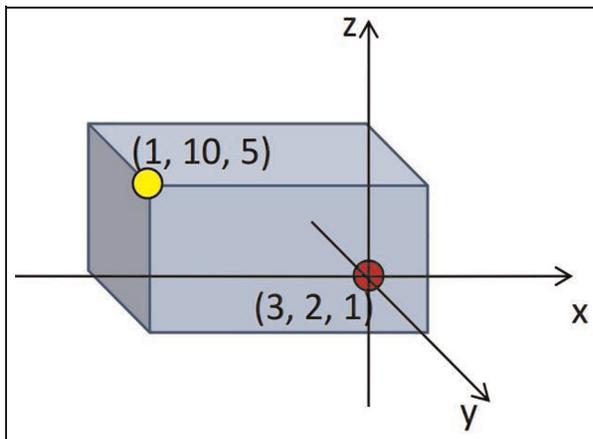


Figure 9. IfcWall example (a wall of 8" × 2" × 4" starts from (3,2,1) towards direction (-1,1,1); the point diagonally opposite the starting point is (x - width, y + length, z + height) = (1,10,5).

has parameters related to the crowd simulation, such as *IFCFilePath* (the location of the IFC file) or *CDppPath* (the location of the simulator). *Entity* is a super-class for all other sub-entities, and it has the following attributes: *id* (sequence number), *type* (e.g., materials, size), references of a direction and a starting point with absolute coordinates. *Entity* has three subclasses: *Wall*, *Door* and *Furniture*. *Door* has subclasses *ExitDoor* and *StairDoor*.

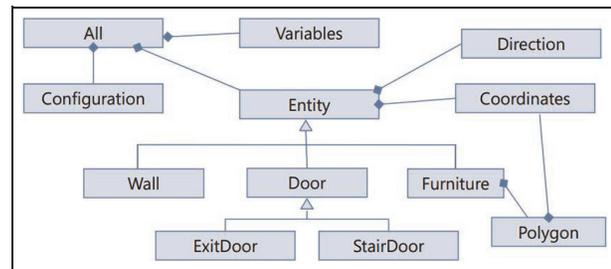


Figure 10. An example of a domain-specific model for simulation (adapted from Wang et al.⁵⁸).

As we can see, the DSM has a different structure to the IFC file (which does not cover all the information for a specific simulation; for example, it does not include information about configuration and variables, as in the DSM). We can define mappings between the IFC standard and the DSM (shown in Table 1). As we can see, class names, structure and attributes look similar but they have differences in syntax and semantics. For example, *Direction* and *Coordinates* in DSM are absolute to a global coordinates system, while the IFC uses relative coordinates.

Such mapping between may be difficult to find. They may have many-to-many relationships or complex mismatches. Our work can be extended using existing model transformation techniques in MDE, such as a

Table I. Mapping between industry foundation class (IFC) and domain-specific model (DSM).

IFC	DSM
<i>IfcElement</i>	Entity
<i>IfcWall, IfcRectangleProfileRef</i>	Wall
<i>IfcDoor, IfcRectangleProfileRef</i>	Door
<i>IfcFurnishingElement, IfcPolyline</i>	Furniture
<i>IfcLocalPlacement, IfcCartesianPoint</i>	Direction
<i>IfcLocalPlacement, IfcDirection</i>	Coordinates

semi-automatic framework to do run-time model transformation, as in Roman et al.,⁶⁵ or a semantic interoperability approach, as in Wang et al.⁶⁶

After this step, the mappings and the IFC instance are ready, and we can generate the DSM instance for a specific simulation. We built a prototype to combine the above steps, in order to generate the DSM instance. This tool can load the extracted data from the IFC file; then, it generates this DSM instance with XMI format (a more general XML file for model transformation).

Finally, we need to generate initialization files for the Cell-DEVS crowd model based on the DSM. For instance, we need the layout of the area under study. Based on the DSM instance, we can calculate the scale of this area by finding $XMIN$ (the horizontal ordinate of the left-most point) and $XMAX$ (the horizontal ordinate of the right-most point). Similarly, we get $YMIN$ and $YMAX$ for the vertical ordinates. Then, knowing the size of each cell, we can compute the number of cells in 3D. After that, we can initialize the state of each cell of the studied area by “paving” the elements according to the DSM instance. Besides, we can also put in other information as input variables, such as the size/number of cells, material feature, etc.

Our MDE-based approach is semi-automatic, requiring additional work to build the DSM and the mappings. Nevertheless, some BIM tools support plug-in functionalities to manage data. Therefore, besides the MDE-based approach, we also developed a tool to extract the data directly from Autodesk Revit architecture, which has a parametric application programming interface (API) and BIM add-in functionality. We used the Revit API to filter parameters, coordinates and other specific information for the studied area. Specifically, this is done in two steps: (1) *filtering elements* and (2) *extracting coordinates*. For the first step, we used the API to filter the element type (e.g., wall, obstacles, doors, stairs), and then to extract the parameter properties needed (e.g., length, width, materials, color). Each element is identified by its element ID, and we select all elements IDs we need. Then, based on the filtered elements ID, we extract their coordinates. After getting the information, we reuse *generating model inputs* (the fifth step discussed above) to initialize the model inputs.

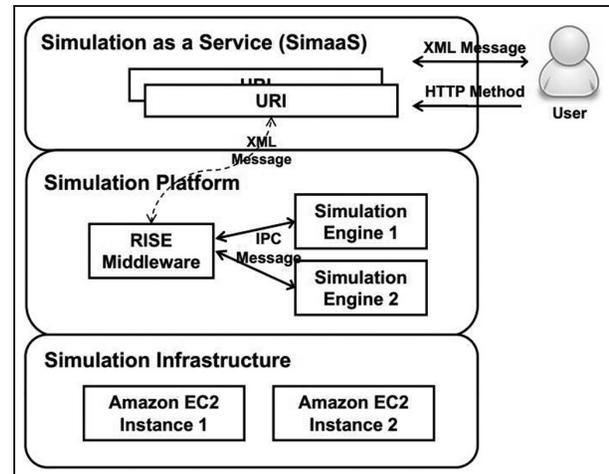


Figure 11. CloudRISE (RISE: RESTful interoperability simulation environment) architecture. XML: extensible markup language; HTTP: hypertext transfer protocol; URI, uniform resource identifier; IPC: inter-process communication; Amazon EC2: Amazon elastic compute cloud.

4.3. Simulation as a service

We have seen how to use Cell-DEVS to build crowd models and how to collect data from CAD/BIM tools. Now, we will discuss the SimaaS component, which focuses on integrating crowd modeling and crowd simulation with the help of cloud computing and SimaaS.

As mentioned in Section 2, although SimaaS has received some attention, it is still at an early stage. We view SimaaS as a special case of SaaS. SimaaS provides simulation services that build on the platform and infrastructure of cloud computing.

Based on these ideas, we have developed CloudRISE, an extended version of RISE running in the cloud, delivering SimaaS for users. Figure 11 shows its three-layer architecture.

- 1) *Simulation infrastructure*: this layer supports the upper layers with the computing and storage infrastructure services available in the cloud. For our experiments, CloudRISE was installed both in Amazon Elastic Compute Cloud (Amazon EC2) and on local servers.
- 2) *Simulation platform*: this layer provides a platform to facilitate the development, deployment and management of simulation services. CloudRISE reuses RISE as middleware between simulation engines and infrastructure.
- 3) *SimaaS*: this layer provides simulation services that build on top of the simulation platform. The simulation services are RESTful WSs. Users can invoke these services through URIs by XML messages using HTTP methods.

In our case, we simulate Cell-DEVS models, which are DEVS models. The ability to execute DEVS models in different operating systems (OSs; e.g., Windows, Linux) and languages (e.g., C++, Java) has already thoroughly discussed in the DEVS literature. For instance, Mittal and Risco-Martin⁶⁷ proposed the “net-centric DEVS Virtual Machine kernel”. The idea is to embed the DEVS kernel in a virtual machine that can be executed in a local, distributed and real-time environment. The proposed CloudRISE is similar to the DEVS kernel at the point of decoupling the model from the simulation platform. The major difference is that the DEVS kernel aims to provide a general DEVS simulator that can be executed anywhere, while CloudRISE is able to deploy any kind of simulator in the cloud, using a middleware to wrap them and expose WSS for them.

4.3.1. Simulation infrastructure. In its current implementation, we use Amazon EC2 for the simulation infrastructure. Amazon is one IaaS provider (this layer can also use other IaaS services, such as Microsoft’s Azure or the Google Application Engine [GAE]). We chose Amazon EC2 because it supports lower level operations and we can completely control the infrastructure and build our own software stack.

We used the Amazon web services management console to manage the Amazon EC2 instances (Figure 12). Amazon EC2 instances are the fundamental infrastructure block for computing needs. They are like virtual servers, where we put our simulation platform and simulation engines. In order to create Amazon EC2 instances, we need to specify the Amazon machine image (AMI) and the instance type (an AMI is a template that contains a software configuration, including an OS). Our current implementation uses the Linux AMI. Instance types have different combinations of central processing unit (CPU)/graphics processing unit (GPU), memory, storage and networking capacity. We can also customize an instance type

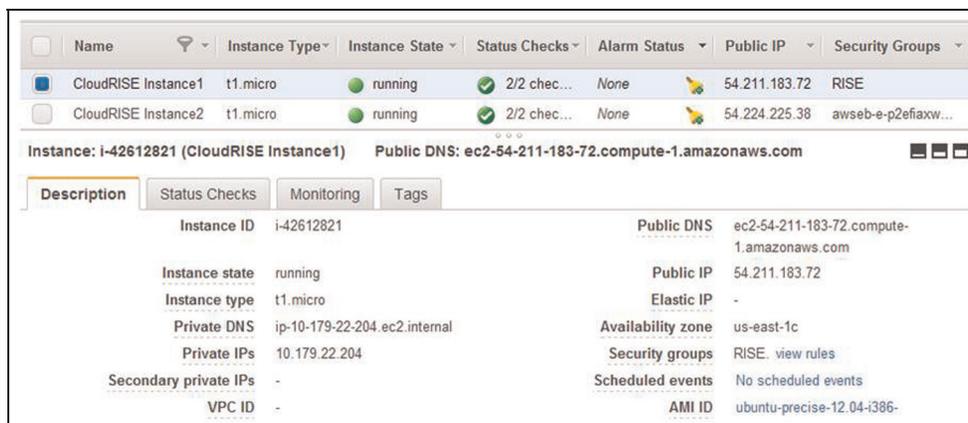
with the appropriate mix of resources. Our current implementation uses micro-instances (they are low cost, providing a small amount of CPU that can be upgraded if needed). We have two instances – *CloudRISE Instance1* and *CloudRISE Instance* – each with its own public internet protocol (IP). We can start/stop/terminate the instance state, check the status and control the network traffic using this console.

4.3.2. Simulation platform. This layer provides a platform to develop simulation services, hiding the complexity underlying the software stack. The basic idea is to use the simulation middleware to link simulation engines, and then to deliver the services to the upper SimaaS layer. Each instance can have one or more simulation middleware (such as RISE), and one or more simulation engines (such as CD++). The simulation service middleware helps the instance to transmit the message between the simulation engines.

In our implementation, we used RISE. We can deploy RISE by uploading it to a Tomcat container on Amazon EC2 instances, or using Elastic Beanstalk to upload its web application archive (WAR) file automatically. We modified RISE, allowing the easy addition of new simulation engines. For instance, in order to add the original CD++ simulator, we upload it to an Amazon EC2 Instance; then, we add a new `./cdppv1` to the URI template `./{ServiceType}` of the RISE URI, and link it to the path of the CD++. Finally, we compile and upload the new RISE middleware.

As seen in Figure 11, when a message comes from the upper SimaaS layer (e.g., a control message to stop the simulation, or an external event at runtime), the execution session is as follows:

- a) RISE gets the XML message from the upper layer; it parses the message to the inter-process communication (IPC) queue on the Amazon EC2 instance;



Name	Instance Type	Instance State	Status Checks	Alarm Status	Public IP	Security Groups
CloudRISE Instance1	t1.micro	running	2/2 chec...	None	54.211.183.72	RISE
CloudRISE Instance2	t1.micro	running	2/2 chec...	None	54.224.225.38	aws-eb-e-p2efaxw...

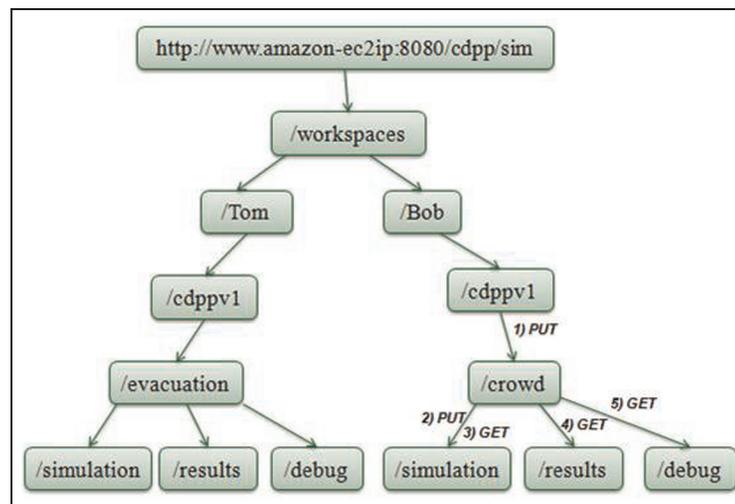
Instance: i-42612821 (CloudRISE Instance1)		Public DNS: ec2-54-211-183-72.compute-1.amazonaws.com	
Description	Status Checks	Monitoring	Tags
Instance ID	i-42612821	Public DNS	ec2-54-211-183-72.compute-1.amazonaws.com
Instance state	running	Public IP	54.211.183.72
Instance type	t1.micro	Elastic IP	-
Private DNS	ip-10-179-22-204.ec2.internal	Availability zone	us-east-1c
Private IPs	10.179.22.204	Security groups	RISE, view rules
Secondary private IPs	-	Scheduled events	No scheduled events
VPC ID	-	AMI ID	ubuntu-precise-12.04-i386-

Figure 12. Amazon EC2 instances for CloudRISE.

Table 2. Simulation as a service application programming interfaces in CloudRISE.

URI	METHOD	DESCRIPTION
<code>../cdppv1</code>	GET	Returns all simulation frameworks for a user
<code>../cdppv1/{framework}</code>	PUT	Creates a simulation framework
<code>../cdppv1/{framework}</code>	POST	Submits files for a simulation framework (e.g., models, initial files)
<code>../cdppv1/{framework}/simulation</code>	PUT	Starts the simulation of a framework
<code>../cdppv1/{framework}/simulation</code>	GET	Returns the execution status of a framework
<code>../cdppv1/{framework}/results</code>	GET	Returns the simulation results
<code>../cdppv1/{framework}/debug</code>	GET	Returns debugging information of a simulation

URI: uniform resource identifier.

**Figure 13.** CloudRISE uniform resource identifier hierarchy for crowd models.

- b) the CD++ simulator that resides on the instance keeps listening to the IPC queue;
- c) as long as there is a message in the queue, CD++ will get the message and execute it.

4.3.3. SimaaS. This layer delivers the simulation services based on the RISE middleware. The original RISE API includes various services (e.g., `../log` for system log, `../accounts` for user account information, `../framework` for the simulation framework, etc.). For simplicity, in CloudRISE, the SimaaS layer only exposes the API services related to the simulation framework. Table 2 shows these APIs (the URIs start from the endpoint, i.e., `http://www.amazon-ec2ip.com:8080/cdpp/sim/workspaces/Tom`). In this URI, “`http://www.amazon-ec2ip.com`” can be replaced with any public IP of an Amazon EC2 instance. Besides, `/Tom` is a user name; all resources related to this person will be under this URL.

We now need to simulate the crowd models that were previously defined. Figure 13 shows an example of the URI hierarchy in CloudRISE for this. We can see from this figure that each URI starts from the endpoint

`http://www.amazon-ec2ip.com:8080/cdpp/sim` (e.g. `http://ec2-184-73-236-191.compute-1.amazonaws.com:8080/cdpp/sim/workspaces/`). It has two users (*Tom* and *Bob*), each of which can use different simulation engines. At present, the original CD++ is under the `../cdppv1`. Furthermore, the simulation engine can use different frameworks. For example, Tom has the URI `../Tom/cdppv1/evacuation`, which means that Tom has a simulation framework for running an evacuation model. Each simulation framework can further have sub-branches with more operations (e.g., `../simulation`, `../results` and `../debug`). For instance, we can issue a GET HTTP call to `../cdppv1/evacuation/results` to get the simulation results of the evacuation model.

The CloudRISE API, listed in Table 2, shows how we can integrate the crowd M&S. After we get the crowd model and its input files, we can use this API to execute the simulation and get the simulation results. To do so, we follow the steps below:

- 1) name a framework for this simulation, with PUT to `../cdppv1/{framework}`;
- 2) upload crowd models and related files, POST the files to `../cdppv1/{framework}`;

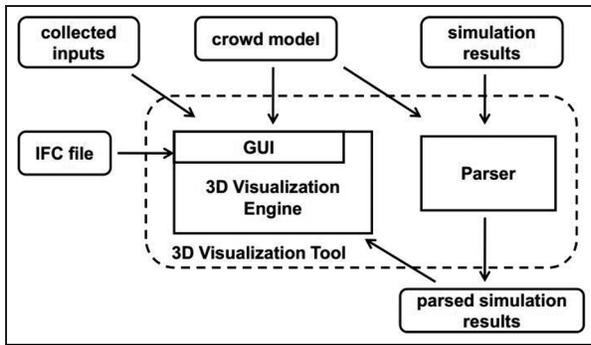


Figure 14. General data flow in the three-dimensional (3D) visualization tool. IFC: industry foundation class; GUI: graphical user interface.

- 3) start crowd simulation, with PUT to `../cdppv1/{framework}/simulation`;
- 4) check simulation status, with GET to `../cdppv1/{framework}/simulation`;
- 5) retrieve simulation results, with GET to `../cdppv1/{framework}/results`.

In our simulation scenario, we have a crowd model available (for instance, the Cell-DEVS bi-directional model introduced in Section 4.1) and the inputs collected from BIM tools (the initial layout for the Cell-DEVS model introduced in Section 4.2). Assume that Bob wants to run this simulation. Bob will follow the steps labeled in Figure 13. First, in order to create a framework, Bob names the framework as *crowd* and issues PUT to `../Bob/cdppv1/crowd`. Then, Bob can upload the files (i.e., the crowd model and the initial layout) via POST to the newly created framework. Now, the simulation is ready to go, and Bob uses PUT `../Bob/cdppv1/crowd/simulation`. Next, Bob can check the simulation execution status with GET `../Bob/cdppv1/crowd/simulation`. The simulation results are obtained by GET to `../Bob/cdppv1/crowd/results`.

4.4 Three-dimensional visualization

In order to complete the cycle, we now introduce how to use 3D visualization (see Figure 14). We use a BIM authoring tool (in order to import and use the IFC files).

The crowd simulation results contain different kinds of information that might not be useful for visualization (including synchronization messages for controlling the simulation phase and time). The *parser* deals with this issue. The *3D visualization engine* provides different animation functions for the crowd movement. The engine uses the *parsed simulation results*, the *crowd model* and the *collected inputs* from the IFC file (Section 4.2). Specifically, the collected data can tell us (X_{min} , Y_{min}) and (X_{max} , Y_{max}), which constitute the bounding box of the building. Besides, the Cell-DEVS crowd model tells us

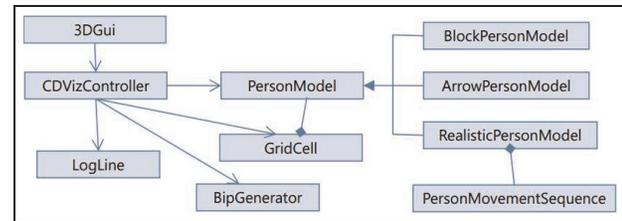


Figure 15. Class diagram of the three-dimensional (3D) visualization engine (adapted from Freire et al.⁵⁷).

the cell dimensions, in order to know how many cells should be in the area. Based on this information, we can establish the exact coordinates of each cell.

The 3D visualization engine has been built using the model-view-controller (MVC) design pattern. Figure 15 shows its class diagram. The controller *CDVizController* is for the key logic. The *LogLine* class keeps records of lines from the parsed log. The simulation grid is represented by an array of *GridCell* objects, which keeps track of the persons in each cell. *PersonModel* is responsible for visualization of each person with different models, allowing a person to be represented in different ways: if the crowd model does not include the direction information, we use a cube (implemented in *BlockPersonModel*); if we have direction information, we can use either cones pointing to the direction, or realistic models (avatars). Cone models are implemented in *ArrowPersonModel*, while avatars are implemented by the class *RealisticPersonModel*.

In our case, we implemented a prototype using 3ds Max, which has key framing ability for person animations, allowing a smooth transition between continuous key frames. For instance, in the realistic person model, we first prepare an animation file for the movements of a person. We use the Motion Mixer API of 3ds Max to mix all the animations from this file. Due to the different coordinate systems of the Cell-DEVS model and 3ds Max, we need a transformation of coordinates. This is done by applying a scaling and coordinate system transformation to the studied area (in our case, it is implemented in the *CDVizController* class). We also provided a graphical user interface (GUI) to improve usability (seen in Figure 16). The user uses the GUI to load the crowd model and inputs files and simulation results. It also allows the user to focus on parts of the studied area (e.g., hiding floors or focusing on a specific person). This function is implemented in the *CDVizController* class, by applying the *hide/unhide* MAXScript functions on the corresponding *PersonModel*.

For instance, Figure 17 shows the visualization results for the crowd model presented in Section 4.1 at two different times. In this case, we chose the block models for representing the crowd.

The N and S pedestrians can be identified with different colors. The tool helps the designers to observe crowd

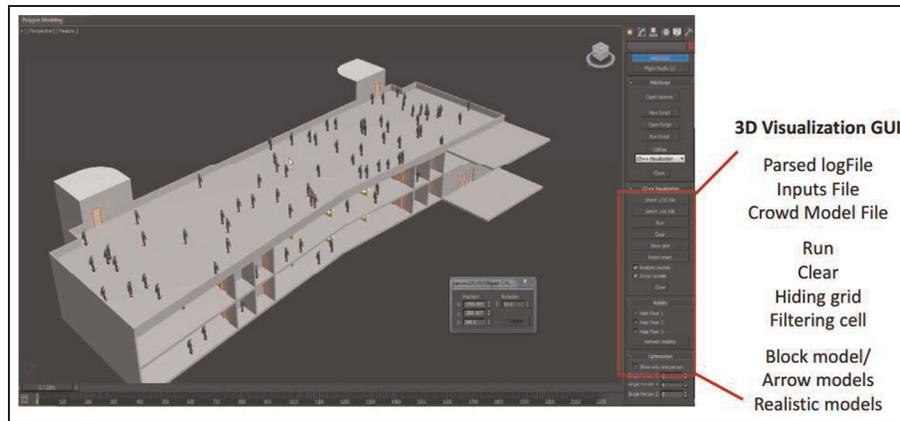


Figure 16. Graphical user interface (GUI) for the 3ds Max visualization tool. 3D: three-dimensional.

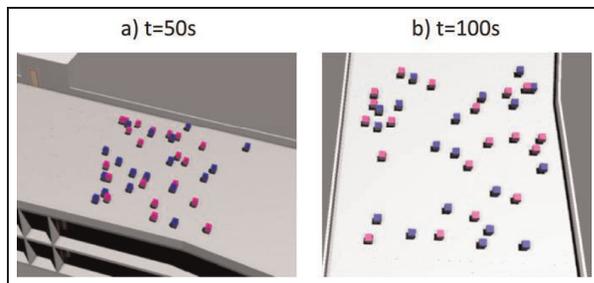


Figure 17. Three-dimensional (3D) simulation results using the 3D visualization tool: (a) $t = 50$ s; (b) $t = 100$ s.

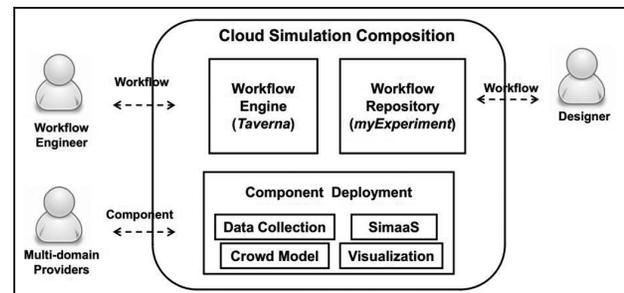


Figure 18. The composition layer in the cloud. SimaaS: simulation as a service.

movements with different perspectives. This visualization also helps us to verify the behavior of the Cell-DEVS model. It can detect unexpected scenarios (e.g., a person missing after a collision). It will display the error messages with the time and location of the person.

5. Composition of components

As discussed in Section 3, the integration process of the components needs to be formalized and automated to make it simple for the designers to reproduce the whole process. To solve this issue, in this section, we discuss the *composition layer* of the proposed architecture.

The composition layer is for deploying all the components related to crowd M&S in the cloud and defining workflows of the integration process. Figure 18 illustrates its overall conceptual structure. It consists of the *component deployment*, the *workflow engine* and the *workflow repository*.

- *Component deployment*: it is responsible for deploying the components in the cloud. A component provides its functionality as an interface for other people (a simulation service, a tool, a script,

etc.). Component providers from different domains have access to the cloud infrastructure. After a component is developed, its provider can upload it to make it reusable.

- *Workflow engine*: it is responsible for defining workflows of the available components. A workflow engineer is in charge of building workflows by linking the components. Here, we have used the Taverna, which supports RESTful WSSs, and we have integrated Taverna and RISE.
- *Workflow repository*: it is responsible for sharing the workflows for reuse by keeping existing workflows in the cloud. Workflow engineers reuse the workflows, and building designers can specify the workflow inputs and execute using a workflow engine in the cloud. In our case, we used *myExperiment*⁶⁸ as the workflow repository.

In order to achieve our goals, at first, we need to consider the crowd M&S components. We divided them into two categories: crowd simulation services (such as SimaaS in CloudRISE) and crowd M&S tools (such as BIM, CD++ and 3ds Max). On one hand, the crowd simulation services are accessible via the web. Therefore, a workflow can invoke these simulation services directly. For example,

SimaaS in CloudRISE uses RESTful WSs, which already encapsulate simulation engines and their environment. Thus, we can use the *TavernaRESTful WS* module to call these services. On the other hand, the crowd M&S tools provide particular functions for crowd M&S. As discussed above, in order to let the workflow invoke the tool, the provider has to make it reusable. In our implementation, we let the provider deploy the tool in the cloud. The provider may have to install the necessary dependencies for this component; for example, the IFC data collection tool can take the inputs of the IFC file and DSM, and then it can generate the outputs of the DSM and the initial input files for the models. After the provider deploys the IFC data collection tool in the cloud, the workflow can invoke it and link its inputs and outputs with other components.

Based on these ideas, we can build a workflow to execute the crowd M&S. Figure 19 shows the overall view of the workflow for the simulation scenarios. In general, it takes the inputs of *crowdModel* (the cell-DEVS crowd model), *DSMmodel* (the DSM including the particular information for the crowd simulation), the *IFCfile* (including the building information of the studied area) and a “*framework*” (a string name to describe the new crowd simulation framework). The output of this workflow is the *parsedLogFile* (the parsed simulation results that can be visualized in 3ds Max). The designer can simply provide these inputs, and run this workflow using the Taverna Engine, which will run the workflow and generate the parsed simulation results. After these are ready, the designer can use the GUI presented in Section 4.4 to visualize them in 3ds Max.

As we can see in Figure 19, in order to start the workflow, the crowd model should be ready (for instance, the bi-directional Cell-DEVS model of Section 4.1). The input parameter *crowdModel* records the model’s location. This location can be either a uniform resource locator (URL) that indicates to a model repository, or a file path in the cloud. In order to integrate the CAD/BIM data with the crowd model, we need to extract information from the studied area as the model’s inputs. *IFC2CDpp* is the crowd M&S tool for this task: it takes the input parameters of *DSMModel* and *IFCfile*, and it then collects input data (Section 4.2). Then, it generates the outputs of the DSM and the initial files (e.g., the layout file and the initial parameters) as inputs for the crowd model. After these are ready, it is combined with the *crowdModel* as inputs for another two crowd M&S tools (*XML Configuration* and *Files2Zip*). *XML Configuration* is a tool for generating the XML configuration file for the simulation, and *Files2Zip* is a tool for compressing the files to be updated to the simulation framework. At this point, we have the crowd model and its inputs ready. We now need to define a new simulation experiment to start the simulation.

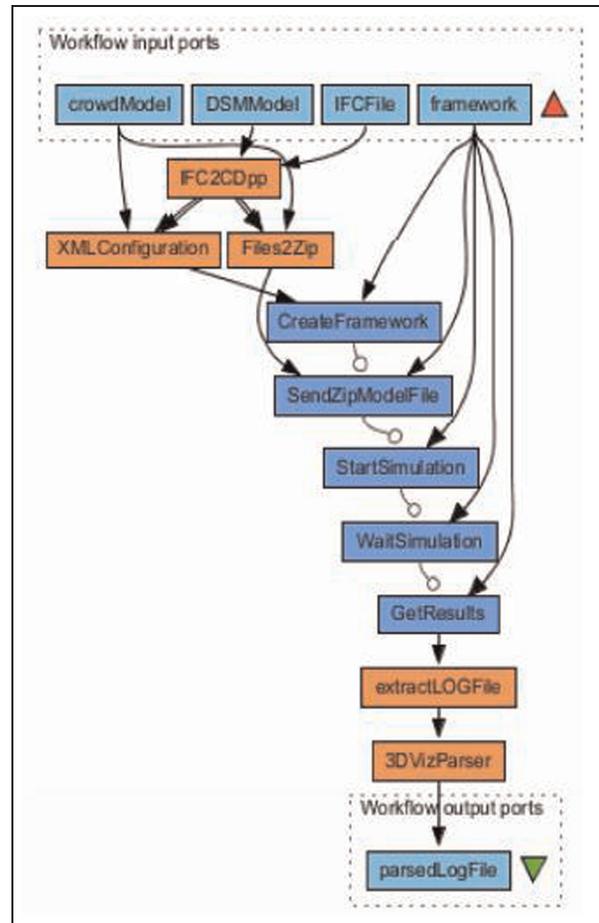


Figure 19. Overall workflow in Taverna.

As described in Section 4.3, we provide SimaaS in CloudRISE. To use the services, we need to invoke them in the workflow (light boxes in the middle of Figure 19). *CreateFramework* helps us create a new framework for the crowd model. It uses *TavernaRESTful WS*, which PUTs the *XML configuration* file to <http://ec2-184-73-236-191.compute-1.amazonaws.com:8080/cdpp/sim/workspaces/Tom/cdppv1/crowdFramework>. In this URI, *crowdFramework* is the value of the input parameter *framework* of the workflow. Next, *SendZipModelFile* POSTs the zip file generated from *Files2Zip*. Now, the simulation is ready to start. *StartSimulation* calls PUT to the *../simulation* URI, and it starts the simulation. Then, *WaitSimulation* checks the simulation execution status using GET on URI *../simulation*. When the simulation finishes, the results are available in a zip file, and we use *GetResults* (sending GET to the URI *../results*). Then, the *extractLOGFile* extracts the simulation results and passes it to *3DVizParser*, which parses the simulation results to be visualized in the 3D visualization.

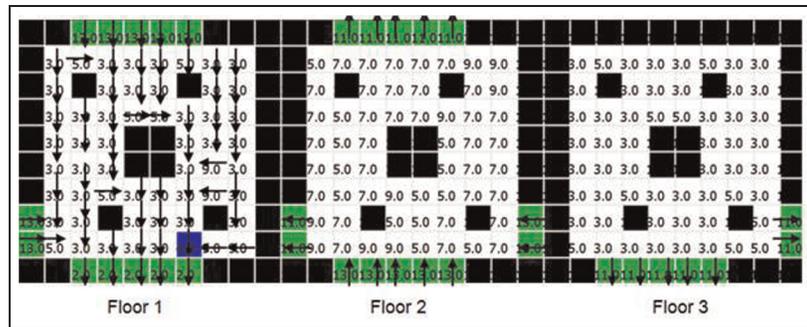


Figure 20. Building with three floors and pathways.

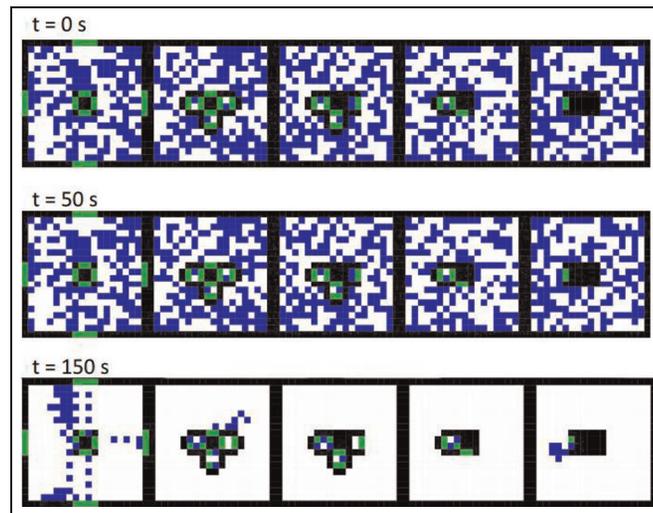


Figure 21. Evacuation simulation result of sample building.

In addition, we can share the workflow on myExperiment. For instance, the workflow in Figure 19 can be found at <http://www.myexperiment.org/workflows/3960.html>. As discussed earlier, a designer can now download this crowd simulation workflow, specify the workflow inputs and execute the workflow in a Taverna workflow engine available in the cloud.

6. Case study: crowd behavior analysis for emergency planning

In this section, we present a complete crowd M&S study using the proposed architecture. We discuss how the designer can reuse the components and the workflow mentioned above. The study presented here focuses on emergency planning. Emergency planning has drawn attention in building design and related applications in terms of human safety and property security.⁶⁹ In this section, we consider a one-directional crowd model in a multi-floor building.^{33,58} The objective is to show how the integrated architecture is able to help the designers to study the

evacuation time of crowds and make decisions between alternatives.

The emergency evacuation model is a Cell-DEVS model built to determine the evacuation time and the occupancy level of a building under emergency conditions. The building under study has multiple floors connected by stairwells, presented in Figure 20. Here, the black cells represent walls; the gray cells represent exits (the four cells at the bottom of Floor 1) or stairs (the rest of the gray cells). There are exit doors on the first floor. In the case of emergencies, people try to evacuate along the pathway on each floor, moving down towards the exits. Each cell contains the information of direction to the exit. The pathway, shown in Floor 1 in Figure 21, is implemented using a Voronoi diagram of the exit route. It starts with the neighbors near the exits/stairs, it assigns each neighbor a direction to the exits/stairs, then it repeats this process in the neighboring cells. Finally, each cell will have a direction that points to the shortest path to the exit/stairs. Each person occupies a cell and the behavior of this person depends

Table 3. Cell states and definitions of the emergency evacuation model.

State	Color	Name	State	Color	Name
1		Wall	8		N Occupied
2		Exit	9		W
3		S	10		W Occupied
4		S Occupied	11		Top of Stairs
5		E	12		Top Occupied
6		E Occupied	13		Bottom of Stairs
7		N	14		Bottom Occupied

Note: Color available in the online version of this article.

on the cell state, presented in Table 3. Besides walls (state 1) and exits (state 2), cells can be either occupied (e.g., states 4, 6, 12) or unoccupied (e.g., states 3, 5, 13).

The simulation rules will be listed in descending order according to their priorities. To avoid collisions we used different priorities to move (e.g., $S > E > N > W$), and only the person with the highest priority can move (note that the order of these priorities is just an example and they can be customized). The rules of this model can be categorized as follows³³:

- someone enters a cell;
- someone moves out of a cell;
- someone moves from a cell to a stairwell;
- someone enters a stairwell;
- someone exits a stairwell.

In the following, we show a few examples of the implementation of these rules.

- *Someone enters a cell*: the rules in this group focus on the empty cells and they are executed when an empty cell is ready to accept a person from a nearby cell (i.e., N, S, W, E) or a stair. The following example shows a subset of the rules used for this purpose:

Rule : 4 100 { (0,0,0) = 3 and ((0,1,0) = 10 or (-1,0,0) = 4 or (0,-1,0) = 6 or (-1,0,0) = 14 or (1,0,0) = 14 or (0,1,0) = 14 or (0,-1,0) = 14) } ...

Rule : 10 100 { (0,0,0) = 9 and ((1,0,0) = 8 or (0,1,0) = 10 or (-1,0,0) = 4 or (-1,0,0) = 14 or (1,0,0) = 14 or (0,1,0) = 14 or (0,-1,0) = 14) }

Each empty cell contains the information of direction, which means that if this cell is occupied, the person in this cell should follow this direction to evacuate. The first rule

says that the current cell is empty, and its direction information is to the S. In this case, if a person wants to enter, the cell becomes occupied. Likewise, the second rule shown as an example has a similar logic; the only difference is that the direction information in the cell is to the E instead of to the S.

- *Someone exits a stairwell*: the following rules are used for the occupied stairs cells; they change to empty if the current cell is occupied and there are empty cells nearby:

Rule : 13 100 { (0,0,0) = 14 and ((1,0,0) = 3 or (1,0,0) = 5 or (1,0,0) = 7 or (1,0,0) = 9 or (-1,0,0) = 3 or (-1,0,0) = 5 or (-1,0,0) = 7 or (-1,0,0) = 9 or (0,1,0) = 3 or (0,1,0) = 5 or (0,1,0) = 7 or (0,1,0) = 9 or (0,-1,0) = 3 or (0,-1,0) = 5 or (0,-1,0) = 7 or (0,-1,0) = 9) }

Rule : 11 100 { (0,0,0) = 12 and (0,0,-1) = 13 }

The first rule says that if the current cell is an occupied stair and there is an empty cell to its N, S, W or E, then the cell turns out to be empty. The second rule says that if the current cell is an occupied stair cell and there is an empty cell to its downstairs, then the person goes down.

When we execute this model, we can obtain varied simulation results. The results can be visualized using traditional tools. In our case, the simulation showed that the crowds evacuate orderly following the behaviors defined. Figure 21 shows a test for a building with five floors (20 × 20 cells on each floor); from left to right shows Floors 1–5. The building design has a single stairwell (in the middle of each floor). People try to evacuate through the stairwell, and leave through the main door in Floor 1. The occupancy level is adjustable so that an optimal occupancy can be determined. We used a 50% occupation rate (730 persons), and it took 205 s to be fully evacuated.

Once a Cell-DEVS model like this one is developed, we can store it in the SimaaS middleware. In order to start a crowd simulation, this model firstly needs the layout information of the studied area from CAD/BIM tools. To do so, we use the data collection component (Section 4.2). In our case, this uses the standardized IFC files to build the DSM instance and then generates the initial files. For instance, Figure 22 shows the IFC file of this case in Revit.

For the DSM, the designer can use the methods and tools presented in Section 4.2. The tool for building the DSM instance requires two inputs: the DSM and its mappings with the IFC standard. The following example shows a part of the results of a DSM instance in XML after this step:

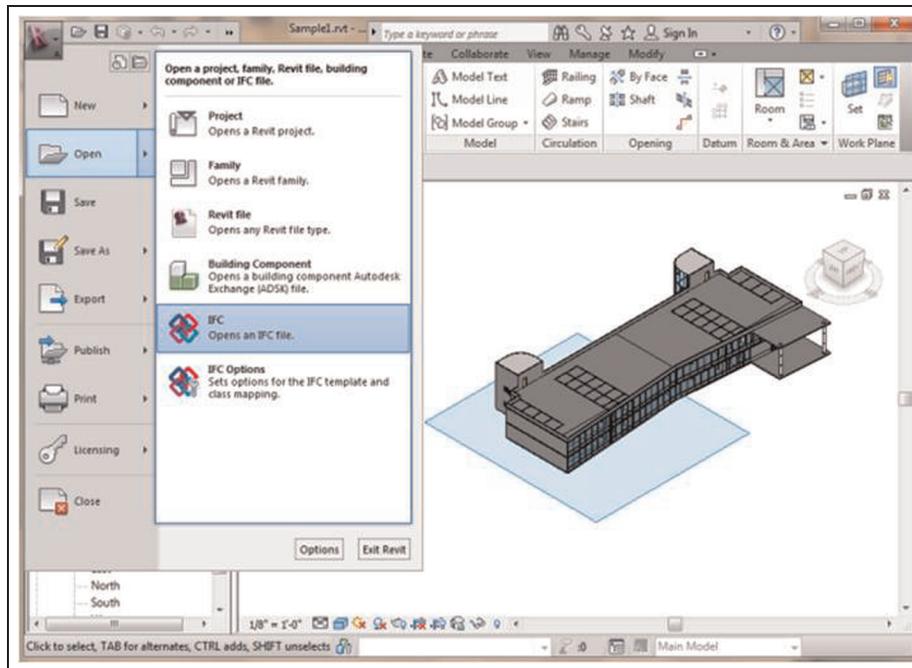


Figure 22. Industry foundation class loading in Revit for the sample building.

```

< ?xml version="1.0" encoding="ASCII" ? >
< EVA:ALLxmlns:xmi="http://www.omg.org/XMI" ... >
< Configuration IFCVersion="Ifc2x3" IFCFile="eva.IFC"
initialFiles="eva.val;eva.inc;eva.ma" / >
< DEVSVariabescell_size="24" cell_x="109" cell_y="43" cell_z="3" / >
< entitySetxsi:type="EVA:Wall" id="195643" type="Basic Wall:Generic - 8":249"
width="0.6666667" length="12.0" >
  < startingPoint X="-48.333332" Y="20.333334" Z="24.0"/ >
  < direction Xdir="-1.0" Ydir="0.0" Zdir="0.0"/ >
< /entitySet >
...
< entitySetxsi:type="EVA:Door" id="181862" type="36" x 84"" width="0.0"
length="3.0" >
  < startingPoint X="4.9112697" Y="-5.993986" Z="12.0"/ >
  < direction Xdir="-1.0" Ydir="0.0" Zdir="0.0"/ >
  < wallBelonged >
    < startingPoint X="4.577936" Y="19.4317" Z="12.0"/ >
    < direction Xdir="0.0" Ydir="-1.0" Zdir="0.0"/ >
  < /wallBelonged >
< /entitySet >
...
< entitySetxsi:type="EVA:Furniture" id="181832" type="96" x 96"" >
  < polygonSet >
    < coordinateSet X="-35.333336" Y="-3.8333333" Z="2.5"/ >
    < coordinateSet X="-33.333336" Y="-3.8333333" Z="2.5"/ >
    < coordinateSet X="-33.333336" Y="-8.833333" Z="2.5"/ >
  < /polygonSet >
...
< /entitySet >
< /EVA:ALL >

```

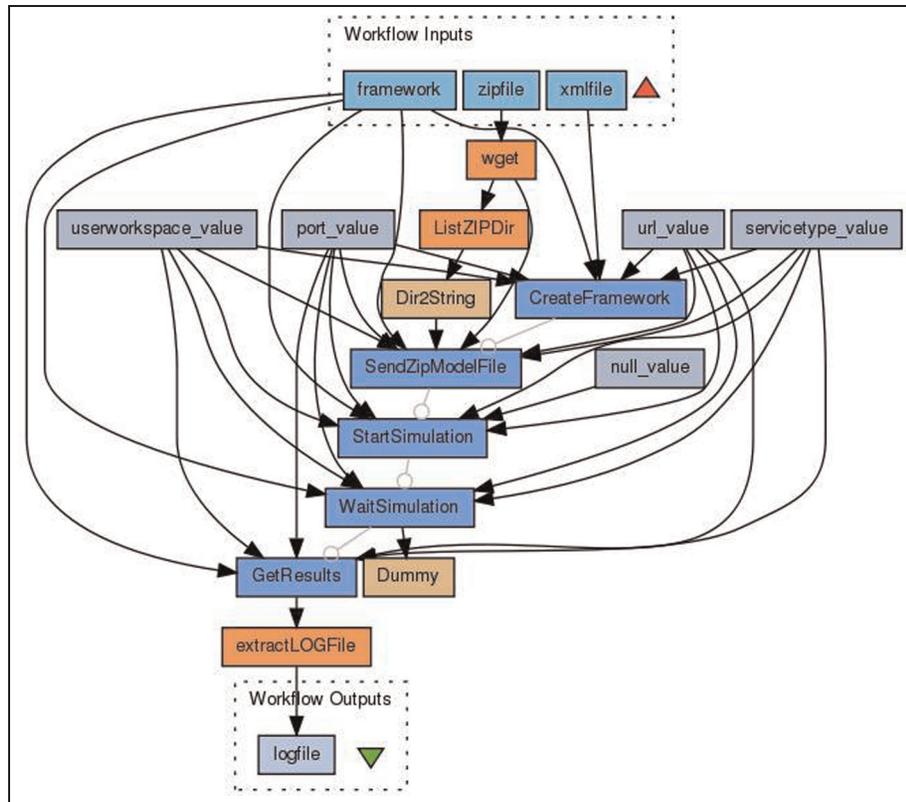


Figure 23. Workflow defined in Taverna for automating web-based simulation using simulation as a service services.

The XML file consists of all the simulation information needed from the building. As we can see, it records configuration information (e.g., *IFCVersion*, *IFCFile* and *initialFiles* names), variables (e.g., *cell_size*, number of cells) and entity information (e.g., *EVA:Wall*, *EVA:Door*, *EVA:Furniture*). For example, the *Wall* shown uses *id* (19,564), *width* (0.67), *length* (12.0), *startingPoint* (−48.3, 20.3, 24.0) and *direction* (−1.0, 0.0, 0.0, which means it points to the negative *x*-axis).

Based on the DSM XML instance file, the designer can reuse the tools in Section 4.2 to generate the initialization files. For instance, some variables generated after this step are *XMIN* (the horizontal ordinate of the left-most point) and *XMAX* (the horizontal ordinate of the right-most point), *Cell_SIZE* (the size of each cell), and *Cell_X*, *Cell_Y*, *Cell_Z* (the number of cells in each dimension). This step can also generate initial values for each cell. For instance, $(22, 26, 0) = 0$ means that the cell with the number 22nd in the *x*-coordinate and 26th in the *y*-coordinate on Floor 1 is Empty (0).

Once the data extracted from the CAD/BIM tool is ready, the designer can reuse the SimaaS services of CloudRISE (Section 4.3) to run the simulation. However, using SimaaS services still needs several steps. For example, the designer needs to choose the right URL and HTTP methods to start an experiment, upload the initialized files,

execute the simulation and get the simulation results. To automate this process, we provide a workflow defined in Taverna (Figure 23). Using this workflow, the designer only needs three inputs: the emergency model file and the initial files generated, an XML to configure the SimaaS experiment and a new experiment name (for instance, *evacuation*). Then, the designer can run the workflow using the Taverna Engine in the cloud. The workflow can run the simulation and generate the output log file after it finishes. In our case, the workflow is available at <http://www.myexperiment.org/workflows/2873.html>.

After the simulation is done, the designer can access the simulation framework through the hypertext markup language (HTML). Figure 24 shows the HTML representation of this experiment *./evacuation*. The execution status and the links of its results can be seen.

After this, the designer can use the advanced 3D Visualization component (Section 4.4) to see the simulation results in Autodesk 3ds Max. The designer can reuse the simulation results parser to get the information (e.g., time, position, direction and occupation of each cell). Then, the designer can load these results in 3ds Max. Figure 25 shows how the results look from different perspectives: for example, (a) a realistic visualization in the main exit; (b) an arrow visualization on the top of building; (c) crowd movement in a stairwell; and (d) crowd

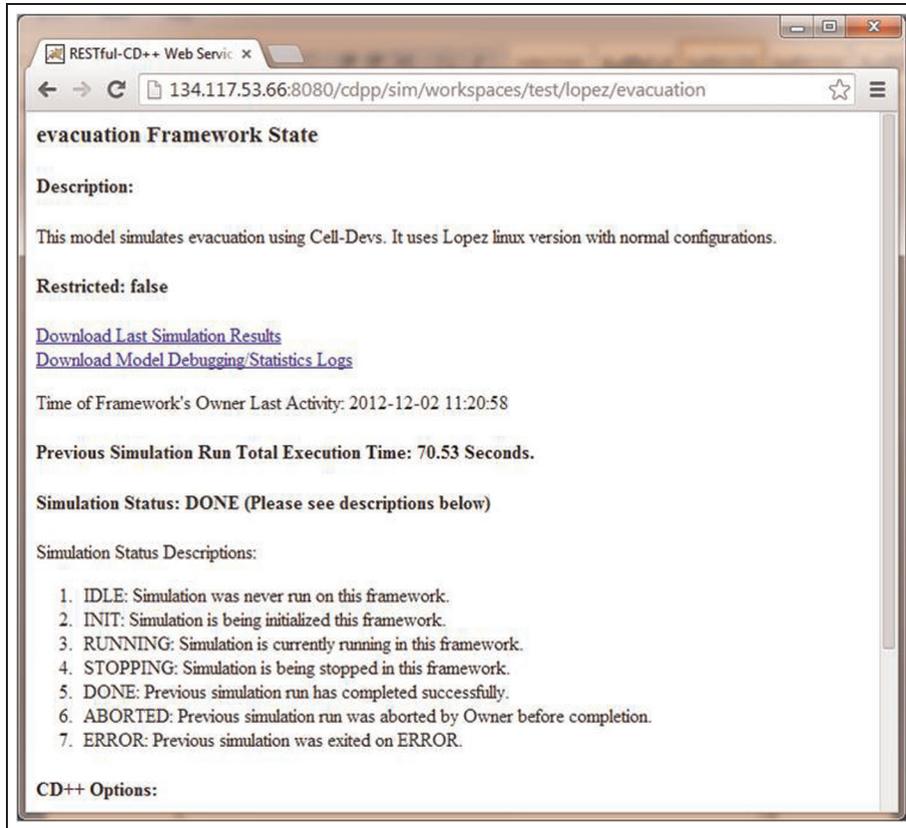


Figure 24. Simulation as a service simulation framework hypertext markup language representation for the experiment/evacuation.

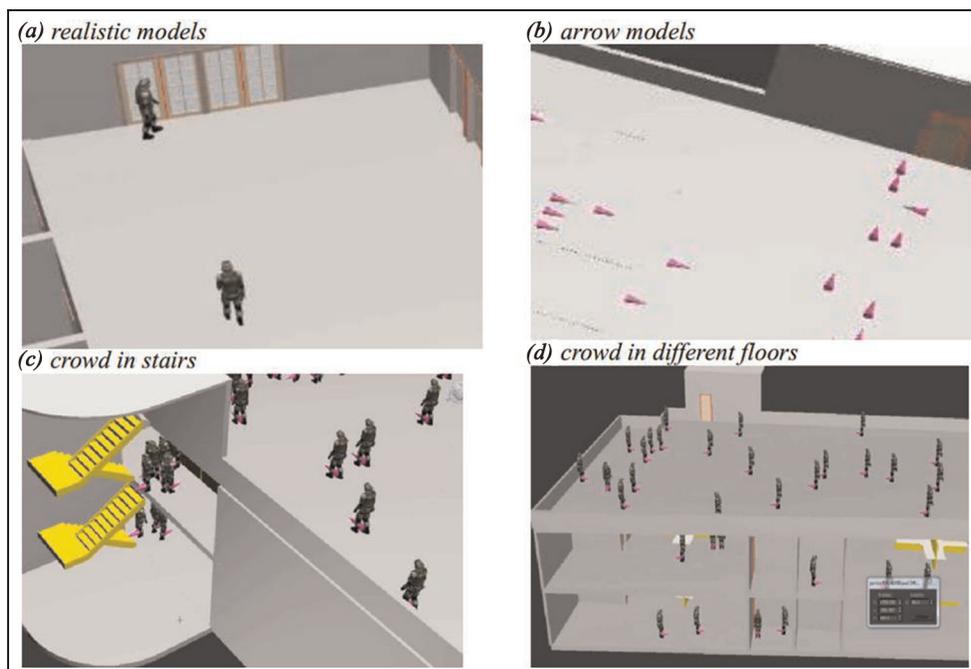


Figure 25. Three-dimensional visualization results in different perspectives: (a) a realistic visualization in the main exit; (b) an arrow visualization on the top of building; (c) crowd movement in a stairwell; and (d) crowd movement in different floors.

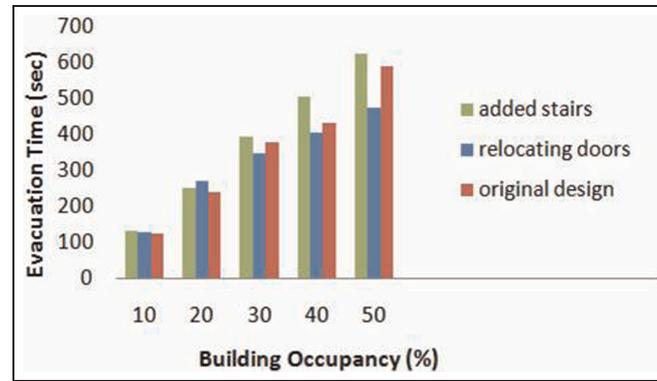


Figure 26. Building occupancy versus evacuation time for design alternatives (original design, relocating the doors or adding more stairs). Adapted from Wang et al.³³

movement in different floors. The GUI also allows the designer to filter specific floors and focus on individuals for better tractability and visibility. The demo of this case study can be found at <http://www.youtube.com/watch?v=u5idq-PDLck>.

At the composition layer of the proposed architecture, workflow engineers can link the components mentioned above to increase the automation of the whole process. They can find and repurpose related workflows in the repository. For instance, in this case, they can reuse the workflow described in Figure 19. The workflow takes the inputs of *crowdModel* (the evacuation model presented above), *DSMmodel* (the DSM with information for the crowd simulation), *IFCfile* (the IFC file of the building design) and *framework* (the name of the new experiment). The output of this workflow is the *parsedLogFile* (the parsed simulation results that can be visualized). To use the workflow, a designer only needs to specify the inputs. In this case, a designer can pass specific inputs (for instance, this emergency model, the same DSM used in Section 4.2, the IFC file of the studied building and the experiment name *./evacuation*). Then, the designer can run this workflow using the Taverna Engine in the cloud. The workflow helps the designer collecting the information, running the simulation and parsing the results. After the simulation results are ready, the designer can use the GUI to visualize them in Autodesk 3ds Max (as shown in Figure 25).

Now, a complete application based on the proposed architecture is available for studying crowds under emergency scenarios. The designers can run this workflow multiple times to evaluate alternatives in order to improve their final design. For instance, a designer might want to test the evacuation time with different occupancy levels for their designs. To do so, they can export the IFC files from their designs and run the workflows of their interest. Figure 26 shows an example. Designers obtained different results for design alternatives (original design, relocating the doors, or adding more stairs). The idea was to reproduce the workflow to extract floor data, run the simulation

in the cloud, and visualize the results obtained. For all designs shown in Figure 26, when the occupancy level increases, the corresponding evacuation needs more time with a roughly linear manner. However, the relocating design shows a least increasing rate when the occupancy level increases. Therefore, this kind of application can tell the designers that the relocating design is the best alternative in terms of reducing the evacuation time in an emergency case.

7. Conclusion

We presented an architecture and a general integration methodology to combine all the components needed for crowd simulation (CAD/BIM data collection, crowd modeling, crowd simulation and 3D visualization) in the cloud. We showed how to use a MDE approach to collect data from CAD/BIM tools, building Cell-DEVS crowd models and using SimaaS middleware to run crowd simulations, and 3D visualization of simulation results. We defined workflows to automate the process and used a repository to share the workflow. We illustrate the use of the architecture with a case study in emergency planning, which illustrates the advantages (distributed deployment, simulation-based design and collaborative development) of the approach.

The main contributions include the following.

- 1) A unified approach for combining loosely coupled components (data collection, modeling, simulation and visualization) into a whole system. This component-oriented approach not only provides interfaces to reuse each component, but also promotes the collaborative development among users from different disciplines.
- 2) Different Cell-DEVS models for crowd behaviors, showing the advantages of Cell-DEVS theory for modeling complex behavior of crowds.

- 3) Different solutions for CAD/BIM data collection (directly or using MDE), SimaaS services for crowd simulation and 3D visualization with realistic crowd entities. Besides, we deployed these solutions in the cloud to increase their reusability and accessibility.
- 4) Workflows to formalize and automate the integration process of the proposed components. Users can easily replace components and modify the workflow for their particular purposes. Besides, these workflows can be stored in a workflow repository for reuse.

This work promotes collaborative development among people from different disciplines (e.g., modelers, builders, architects, building designers, workflow engineers, data analysts). Each person can focus on a specific activity in crowd M&S and provide a component for others. People can provide new components or reuse the existing components in the cloud. Besides, each component can be composed as part of a workflow, allowing the automation of the crowd M&S process. This kind of workflow hides the technical details inside the components. Building designers can test their designs by executing workflows. Therefore, it can lead to faster decision-making between designs and improve building design before construction has begun. This kind of application can increase the reusability of crowd M&S activities and improve productivity.

Funding

This research has been partially funded by NSERC.

Acknowledgements

The authors thank Michael Van Schyndel (the crowd model in multiple floors), Victor Freire (the 3D visualization tool), Vinu Subashini and Robert Woodbury (the building sample for the case study), and Rhys Goldstein and Azam Khan (valuable insights and support for Autodesk tools).

References

1. Ma J, Lo SM, Song WG, et al. Modeling pedestrian space in complex building for efficient pedestrian traffic simulation. *Autom Constr* 2013; 30: 25–36.
2. Hoogendoorn SP and Bovy PHL. Pedestrian route-choice and activity scheduling theory and models. *Transp Res B* 2004; 38: 169–190.
3. Duives DC, Daamen W and Hoogendoorn SP. State-of-the-art crowd motion simulation models. *Transp Res C* 2013; 37: 193–209.
4. Ham NH, Min KM, Kim JH, et al. A study on application of BIM to pre-design in construction project. In: *3rd international conference on convergence and hybrid information technology*, Busan, Korea, 2008.
5. Al-Hussein M, AtharNiaz M, Yu H, et al. Integrating 3D visualization and simulation for tower crane operations on construction sites. *Autom Constr* 2006; 15: 554–562.
6. Wurzer G, Ausserer M, Hinneberg H, et al. Sensitivity visualization of circulation under congestion and blockage. In: Peacock RD, Kuligowski ED and Averill JD (eds) *Pedestrian and evacuation dynamics*. New York, NY: Springer, 2011, pp.899–902.
7. Zhou S, Chen D, Cai W, et al. Crowd modeling and simulation technologies. *ACM Trans Model Comput Simulat* 2010; 20: 20.
8. Ji X, Zhou X and Ran B. A cell-based study on pedestrian acceleration and overtaking in a transfer station corridor. *Physica A* 2013; 392: 1828–1839.
9. Henein CM and White T. Macroscopic effects of microscopic forces between agents in crowd models. *Physica A* 2007; 373: 694–712.
10. Tao W and Jun C. An improved cellular automaton model for urban walkway bi-directional pedestrian flow. In: *international conference on measuring technology and mechatronics automation*, Zhangjiajie, China, 2009.
11. Castonguay P and Wainer G. Aircraft evacuation DEVS implementation visualization. In: *proceedings of the 2009 spring simulation multiconference*, San Diego, CA, 2009.
12. Sano T, Yoshida Y, Takeichi N, et al. Experimental study of crowd flow passing through simple-shaped room and validation for an evacuation simulator. In: Peacock RD, Kuligowski ED and Averill JD (eds) *Pedestrian and evacuation dynamics*. New York, NY: Springer, 2011, pp.587–599.
13. Castle CJE, Waterson NP, Pellissier E, et al. A comparison of grid-based and continuous space pedestrian modelling software: analysis of two UK train stations. In: Peacock RD, Kuligowski ED and Averill JD (eds) *Pedestrian and evacuation dynamics*. New York, NY: Springer, 2011, pp.433–446.
14. Lu SR, Wu IC and Hsiung BCB. Applying building information modelling in environmental impact assessment for urban deep excavation. In: *proceedings of the ISARC*, Eindhoven. The Netherlands, 2012.
15. Hetherington R, Laney R, Peake S, et al. Integrated building design, information and simulation modelling: the need for a new hierarchy. In: *proceedings of 2011 building simulation conference*. Sydney, Australia, 2011.
16. Jiang Y, Ming J, Wu D, et al. BIM server requirements to support the energy efficient building lifecycle. In: *proceedings of 2012 ASCE international conference on computing in civil engineering*, Clearwater Beach, FL, 2012.
17. Wainer G. *Discrete-event modeling and simulation: a practitioner's approach*. Boca Raton, FL: CRC/Taylor Francis, 2009.
18. Xia Y, Wong SC and Shu CW. Dynamic continuum pedestrian flow model with memory effect. *Phys Rev E* 2009; 79: 066113.
19. Dogbe C. On the Cauchy problem for macroscopic model of pedestrian flows. *J Math Anal Appl* 2010; 372: 77–85.
20. Werberich BR, Pretto CO and Cybis HBB. Pedestrian route choice model based on friction forces. *Simulation* 2014; 90: 1177–1187.

21. Salzarulo L. A continuous opinion dynamics model based on the principle of meta-contrast. *J Artif Societies Soc Simulat* 2006; 9: 5–24.
22. Deffuant G. Extremism propagation patterns in continuous opinion models. *J Artif Societies Soc Simulat* 2006; 9: 1–8.
23. Bae JW, Lee S, Hong JH, et al. Simulation-based analyses of an evacuation from a metropolis during a bombardment. *Simulation* 2014; 90: 1244–1267.
24. Helbing D, Farkas I and Vicsek T. Simulating dynamical features of escape panic. *Nature* 2000; 407: 487–490.
25. Neumann JV and Burks AW. *Theory of self-reproducing automata*. Champaign, IL: University of Illinois Press, 1966.
26. Blue VJ and Adler JL. Emergent fundamental pedestrian flows from cellular automata micro simulation. *Transp Res Rec* 1998; 1644: 29–36.
27. Song W, Xu X, Wang B, et al. Simulation of evacuation processes using a multi-grid model for pedestrian dynamics. *Physica A*. 2006; 363: 492–500.
28. Yue H, Guan H, Zhang J, et al. Study on bi-direction pedestrian flow using cellular automata simulation. *Physica A* 2010; 389: 527–539.
29. Zeigler BP, Praehofer H and Kim TG. *Theory of modeling and simulation*. Waltham, MA: Academic Press, 2000.
30. Farrell R, Moallemi M, Wang S, et al. Modeling and simulation of crowd using cellular discrete event systems theory. In: *proceedings of the 2013 ACM SIGSIM PADS*, Montreal, Canada, 2013.
31. Ahmed A, Wainer G and Mahmoud S. Integrating building information modeling Cell-DEVS simulation. In: *symposium on simulation for architecture and urban design*, Orlando, FL, 2010.
32. Hammad A and Zhang C. Towards real-time simulation of construction activities considering spatio-temporal resolution requirements for improving safety and productivity. In: *proceedings of 2011 winter simulation conference*, Phoenix, AZ, 2011.
33. Wang S, Schyndel MV, Wainer G, et al. DEVS-based building information modeling and simulation for emergency evacuation. In: *proceedings of the winter simulation conference*, Berlin, Germany, 2012.
34. Wang S, Wainer G, Rajus VS, et al. Occupancy analysis using building information modeling and Cell-DEVS simulation. In: *symposium on theory of modeling and simulation*, San Diego, CA, 2013.
35. MacDonald M. *STEPS simulation of transient evacuation and pedestrian movements user manual*. Unpublished work, 2003.
36. Lo SM, Fang Z, Lin P, et al. An evacuation model: the SGEM package. *Fire Saf J* 2004; 39: 169–190.
37. Fu C, Aouad G, Lee A, et al. IFC model viewer to support nDModel application. *Autom Constr* 2006; 15: 178–185.
38. Lipman RR. Developing coverage analysis for IFC files. In: *proceedings of the CIB W78 2010: 27th international conference*, Haifa, Israel, 2010.
39. Sanchez P, Barreda J and Ocon J. Integration of domain-specific models into a MDE framework for time-critical embedded systems. In: *international workshop on intelligent solutions in embedded systems*, 2008.
40. Byrne J, Heavey C and Byrne PJ. A review of web-based simulation and supporting tools. *Simul Modell Pract Theory* 2010; 18: 253–276.
41. Fielding RT. *Architectural styles and the design of network-based software architectures*. Doctoral Dissertation, University of California. Oakland, CA, 2000.
42. Papazoglou M. *Web services: principles and technology*. Upper Saddle River, NJ: Prentice Hall, 2007.
43. Madhoun R. *Web-services definition of discrete-event simulation services*. Master's Thesis, Systems and Computer Engineering, Carleton University, 2006.
44. Madhoun R, Feng B and Wainer G. On the creation of distributed simulation web-service-based distributed CD++. In: *proceedings of artificial intelligence, simulation and planning*, Buenos Aires, Argentina, 2007.
45. Wainer G, Madhoun R and Al-Zoubi K. Distributed simulation of DEVS and Cell-DEVS models in CD++ using web-services. *J Simul Modell Pract Theory* 2008; 16: 1266–1292.
46. Wagh K and Thool R. A comparative study of soap vs rest web services provisioning techniques for mobile host. *J Inf Eng Appl* 2012; 2: 12–16.
47. Al-Zoubi K and Wainer G. Using REST web services architecture for distributed simulation. In: *proceedings of principles of advanced and distributed simulation*, Lake Placid, New York, 2009.
48. Al-Zoubi K and Wainer G. RISE: a general simulation interoperability middleware container. *J Parallel Distrib Comput* 2013; 73: 580–594.
49. Mittal S, Risco-Martín JL and Zeigler BP. DEVSML: automating DEVS execution over SOA towards transparent simulators. In: *proceedings of the 2007 spring simulation multiconference*, Norfolk, VA, 2007.
50. Seo C and Zeigler BP. Interoperability between DEVS simulators using service oriented architecture and DEVS namespace. In: *proceedings of the 2009 spring simulation multiconference*, San Diego, CA, 2009.
51. Muqsith MA and Sarjoughian HS. A simulator for service-based software system co-design. In: *proceedings of the 3rd international ICST conference on simulation tools and techniques*, Malaga, Spain, 2010.
52. Mittal S and Risco-Martin JL. Model-driven systems engineering in a netcentric environment with DEVS unified process. In: *proceedings of the winter simulation conference*, Washington, DC, 2013.
53. Cayirci E. Modeling and simulation as a cloud service: a survey. In: *proceedings of the winter simulation conference*, Washington, DC, 2013.
54. Liu X, He Q, Qiu X, et al. Cloud-based computer simulation: towards planting existing simulation software into the cloud. *J Simul Modell Pract Theory* 2012; 26: 135–150.
55. Tsai WT, Li W, Sarjoughian H, et al. SimSaaS: simulation software-as-a-service. In: *proceedings of the 44th annual simulation symposium*, Boston, MA, 2011.
56. Ahmed A, Moallemi M, Wainer G, et al. VCELL: a 3D real-time visual simulation in support of combat. In: *proceedings of the 2011 summer computer simulation conference*, The Hague, Netherlands, 2011.
57. Freire V, Wang S and Wainer G. Visualization in 3ds Max for Cell-DEVS models based on moving entities. In:

- symposium on simulation for architecture and urban design*, San Diego, CA, 2013.
58. Wang S, Wainer G, Goldstein R, et al. Solutions for scalability in building information modeling and simulation-based design. In: *symposium on simulation for architecture and urban design*, San Diego, CA, 2013.
 59. Rybacki S, Himmelspach J, Haack F, et al. Worms—a framework to support workflows in MS. In: *proceedings of the winter simulation conference*, Phoenix, AZ, 2011.
 60. Ribault J and Wainer G. Simulation processes in the cloud for emergency planning. In: *proceedings of the 12th IEEE/ACM international symposium on cluster, cloud and grid computing*, Ottawa, Canada, 2012.
 61. Oinn T, Addis M, Ferris J, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 2004; 20: 3045–3054.
 62. Cicirelli F, Furfaro A and Nigro L. Using time stream petri nets for workflow modelling analysis and enactment. *Simulation* 2012; 89: 68–86.
 63. Mittal S, Zeigler BP and Risco-Martin JL. Implementation of formal standard for interoperability in M&S/system of systems integration with DEVS/SOA. *Int Command Contr C2 J* 2008; 3: 1–57.
 64. Beetz J, VanBerlo L, DeLaat R and VanDen Helm P. BIMserver.org — an open source IFC model server. In: *proceedings of the CIP W78 conference*. Cairo, Egypt, 2010.
 65. Roman D, Morin B, Wang S, et al. A model-driven approach to interoperability in B2B data exchange. In: *advanced results in MDI/SOA innovation workshop*, IWEL, Stockholm, Sweden, 2010.
 66. Wang S, Morin B, Roman D, et al. A semi-automatic model transformation approach for semantic interoperability. In: *proceedings of the NATO symposium on semantic & domain based interoperability*, Oslo, Norway, 2011.
 67. Mittal S and Risco-Martin JL. *Netcentric system of systems engineering with DEVS unified process: a book in system of systems engineering*. Boca Raton, FL: CRC/Taylor Francis, 2013.
 68. Goble CA and De Roure DC. myExperiment: social networking for workflow-using e-scientists. In: *proceedings of the 2nd workshop on workflows in support of large-scale science*. Monterey, CA, 2007.
 69. Boukerche A, Zhang M and Pazzi R. An adaptive virtual simulation and real time emergency response system. In: *international conference on virtual environments, HCI and measurement systems*, Hong Kong, China, 2009.

Author biographies

Sixuan Wang is a PhD candidate in electrical and computer engineering with the Department of Systems and Computer Engineering at Carleton University. He received double master's degrees from Harbin Institute of Technology, China, and the University of Bordeaux 1, France. His academic experience spreads over a wide range of areas, such as M&S, software engineering, distributed systems and cloud computing. He has participated in many scientific and industrial projects, including the NSERC Engage project with Autodesk Research (2012, Canada) and the European REMICS project within the SINTEF Institute (2011, Norway). He has several publications and is a frequent reviewer of scientific papers. His current research interests are M&S as a service, semantic web, model composition and cloud computing.

Gabriel A Wainer (SMSCS, SMIEEEE) is a full professor at the Department of Systems and Computer Engineering at Carleton University. He is the author of three books and over 260 research articles; he has edited four other books and helped organize over 120 conferences, including being one of the founders of SIMUTools and SimAUD. He is the principal investigator of different research projects. He is the vice-president of conferences. He is special issues editor of *Simulation*. He is the head of the Advanced Real-Time Simulation Lab, located at Carleton University's Centre for advanced Simulation and Visualization (V-Sim). He has been the recipient of various awards, including the IBM Eclipse Innovation Award, SCS Leadership Award and various Best Paper awards. His current research interests are related to modeling methodologies and tools, parallel/distributed simulation and real-time systems.