# Solutions for Scalability in Building Information Modeling and Simulation-Based Design

**Sixuan Wang[1], Gabriel Wainer[1], Rhys Goldstein[2], and Azam Khan[2]**

[1]Dept. of Systems and Computer Engineering,
Carleton University
1125 Colonel By Dr. Ottawa
ON K1S 5B6, Canada
{swang, gwainer}@sce.carleton.ca

[2]Autodesk Research
210 King St. East, Toronto
ON M5A 1J7, Canada
{firstname.lastname}@autodesk.com

**Keywords:** Scalability, IFC, MDA, DEVS, Remote Simulation

**Abstract**

Simulation-based design can enable a number of advanced architectural and engineering applications, such as energy modeling, occupant behavior prediction, or structural integrity analysis. To help make simulation-based design practical, scalability in terms of data and computation is needed. By using a Model-Driven Architecture (MDA) approach—together with the RISE (RESTful Interoperability Simulation Environment) web interface—a generic scalable simulation design framework is presented. In our system, Building Information Modeling (BIM) data is represented in the Industry Foundation Classes (IFC) open standard from which Domain Specific Models (DSM) may be extracted for particular applications. The open RISE interface to a DEVS (Discrete Event System Specification) simulation provides computational scalability. We present a case study in which our system is applied to an evacuation model of a multi-floor building. We also show a 3D visualization of the simulation results to support further decision making. By enabling designers to extract information automatically from IFC and run simulations remotely, this kind of scalable system makes simulation a viable part of the design process.

## 1. INTRODUCTION

Buildings are multidisciplinary endeavors that cover many specialties, including architecture, engineering, construction, and operations. Therefore, building simulation has also been specialized into a number of areas, such as lighting, thermodynamics, occupancy, advanced control systems, computational fluid dynamics, structural finite element methods, and other modeling domains. Historically, independent efforts have led to simulation solutions appropriate for each individual domain. However, recent research has examined methods for integrating prior work for whole building simulation (Hensen 2002). Several important benefits may result from this systems perspective, such as the handling of interactions between solvers (Wang et al. 2012). For example, an occupancy solver may feed into a simulated building control system that also senses the radiant heating effects from a thermodynamics solver to provide an analysis of occupant comfort.

A key challenge in adopting a systems approach is scalability (Duboc et al. 2012) in terms of the size and complexity of the models that supply building simulations with site-specific information, the computational requirements of simulations involving multiple building domains, and the feasibility of integrating whole building simulation into the design process. Addressing these challenges will require a change in the technologies and techniques currently used to perform simulation-based design. First, the use of a file-based Building Information Model (BIM) usually impedes collaboration between designers due to its complication and redundancy (Won and Lee 2011; Hetherington et al. 2011). Second, current simulation tools do not adequately address the discrepancy between comprehensive BIM standards such as the Industry Foundation Classes (IFC) and the input data that is actually required by simulation models of individual building domains (Hitchcock and Wong 2011; Jiang et al. 2012). Third, running simulations on single-user workstations fails to take advantage of high performance server-based computing, thin clients and web services technology (Ribault and Wainer 2012; Al-Zoubi and Wainer 2010). Fourth, the separation of design tools and analysis tools denies designers the interactivity required to fully benefit from building simulation results (Wang et al. 2012).

In this paper, we demonstrate solutions to each of the four scalability issues listed above. We first propose a generic process framework for integrating all the solutions in the simulation-based design cycle. As an alternative to file-based BIMs, we explore the database approach of BIMserver.org as a tool for simulation-based design. To address the discrepancy between IFC and the domain specific information required by simulation models, we adopt the Model-Driven Architecture (MDA) approach from the field of software engineering. As an alternative to running simulations on single-user workstations, we use web services technology to exploit computing resources on remote servers. Finally, we demonstrate how visualizations of simulation results can be implemented within existing BIM authoring tools. These solutions can enable designers to extract information selectively from a standard IFC file into a specific simulation model, run the simulation remotely, and then visualize the 3D simulation results for making decisions or planning a next design iteration.

A variety of methods can be used to develop Modeling and Simulation (M&S) applications. Among them, we are interested in exploring the use of the Discrete Event System Specification (DEVS) formalism (Zeigler et al. 2000) for building discrete event systems with models composed of behavioral (atomic) and structural (coupled) components. Cell-DEVS (Wainer 2009) extends DEVS by supporting cellular models in a spatial lattice. For remote simulation, we reuse the RISE (RESTful Interoperability Simulation Environment) (Al-Zoubi and Wainer 2010) as a simulation middleware to support RESTful-CD++ web services for remote simulation.

The integration of building information collection from BIM, simulation with Cell-DEVS, and visualizations of simulation results in BIM authoring tools have been presented in previous work (Wang et.al. 2012). Here, we demonstrate these solutions in conjunction with the MDA approach and RISE remote simulation, which are the primary contributions of this paper. The model-driven architecture (MDA) proposes a development paradigm to handle the increasing complexity of the building models. MDA is based on developing both platform independent and specific models from which executable code can be generated in an automatic/semi-automatic way (Sanchez et al. 2008). MDA uses multilayer meta-models with different abstraction levels for interoperability. In the context of building information modeling, MDA involves the use of a separate Domain Specific Model (DSM) for each building simulation domain. There are also many model transformation techniques which can automatically/semi-automatically generate models between upper models or DSMs (Wang et al. 2011). A comprehensive IFC representation of a building is transformed into each DSM, a process that filters out irrelevant data and resolves inconsistencies between IFC and simulation-specific building representations. An emergency planning example is presented to illustrate this transformation of data, and to highlight the role of all four solutions in a design workflow.

We begin with the literature review. Then we represent the generic scalability BIM&S framework, followed by details of each step, focusing on how to extract information and build DSM simulation model. Section 5 illustrates a case study of evacuation simulation.

## 2. LITERATURE REVIEW

BIM technologies have been widely adopted in the Architecture, Engineering, and Construction (AEC) industry for 3D-rendering, drawing extraction, estimation of cost, and emergency detection (Eastman et al. 2008). At present, there are numerous BIM software applications, including Autodesk Revit Architecture, Autodesk 3Ds Max, Bentley Architecture, and Graphisoft ArchiCAD. Striving for interoperability and standardization between different domains, various BIM standards have been created, such as DSF, STEP, IFC, gbXML, and SAT. IFC, proposed by buildingSMART International, enables different users to build models for lighting, shading, HVAC engineering, and various other AEC domains. The IFC hierarchy covers the core project information such as building elements, geometric and material properties of building products, and so on (Fu et al. 2006). The current popular standard is Ifc2x3 (the next version Ifc2x4 is in development). Ifc2x3 has 653 entities, 317 property sets and 164 enumerations (Hetherington et al. 2011).

Existing simulation processes are not fully integrated into the design life cycle. Various scalability issues have arisen when integrating standard BIM tools and simulation software. Firstly, the standard BIM file is too complicated and too highly redundant to understand and query (Lipman 2010; Hetherington et al. 2011). Take IFC as an example; it intends to connect all design domains into one file, which covers a vast amount of information from various project participants. To make the standard BIM file consistent and reliable, lots of implicit information is added. The size of the IFC file grows and it needs more time to manipulate. Because each individual participant usually requests partial information (Won and Lee 2011), less experienced designers need more time to learn the structure of the IFC file for finding the exact information needed. Though some BIM tools allow users to select, edit or trim the IFC model (e.g. SimpleBIM, Model Checker, or BIMsight), IFC data is often file-based and hard to parse or query (Beetz et al. 2011). We still need an automatic and scalable way to reduce and query the IFC file efficiently.

Furthermore, the integration process lacks explicit support for specific simulation information (Malinowsky et al. 2010; Jiang et al. 2012). Simulations for different scenes are mostly domain-specific, which normally have different syntax, structure, and semantics compared with the standard files. Even when standards are broad, any single file does not cover all the special tasks during the simulation process. Besides, people tend not to support standards that might affect their existing implementation or threaten their dominant market positions (Wainer et al. 2010). In fact, building design and simulation are two independent domains. To solve the first two issues, in (Fu et al. 2006), the authors introduced a concept of “3D to nD Modeling”, encouraging designers using multi-issues (specific domains) of design information to reduce the uncertainties and realize true “what-if” analysis. One solution is to use Model View Definitions (MVD) to facilitate automated querying from the IFC model, which is a subset of IFC (Lipman 2010). However, MVD is dependent on its implementation, and an IFC model may not cover all needed simulation information. Alternatively, in (Jiang et al. 2012; Beetz et al. 2011), the authors suggest using an MDA to parse the IFC file into its own pre-defined data structure DSM based on the Eclipse Model Framework (EMF). We can use XML and meta-models to manage the IFC data using MDA tools, then easily load, query and filter the IFC data.

Most modeling and simulation methods run on single-user workstations (Ribault and Wainer 2012), which normally require too much time for installation and configuration of all the software and dependencies needed for simulation. It is better to remotely access simulation resources with web services (Jiang et al. 2012), improving scalability (e.g., accessibility, availability and interoperability). In this way, users can reuse and share simulation resources on site without worrying about the capability of their local machine CPU or memory. Plus, with distributed simulation technologies, simulations can be executed on distributed computers via communication networks, which can further speed up the execution time (Al-Zoubi and Wainer 2010). SOAP-based Web Services use the Remote Procedure Call (RPC) to allow two distributed heterogeneous systems to call the functions of each other; but this exposes the implementation details and lacks scalability (Al-Zoubi and Wainer 2010). Another choice, RESTful Web Services, can solve these issues. Access to RESTful Web Services is through Web resources (URIs) and XML messages using uniform HTTP operations (GET, PUT, POST and DELETE). Based on these ideas, in (Al-Zoubi and Wainer 2010), the authors presented the RESTful Interoperability Simulation Environment (RISE) middleware. The main objective of RISE is to support interoperability and mash-ups of distributed simulations regardless of the modeling formalism, programming language, or simulation engine.

## 3. GENERIC SCALABLE BIM&S FRAMEWORK

The design phase has several iterative cycles through alternative simulations. In order to increase scalability, we propose the Generic Scalable BIM&S Framework, which is shown in Figure 1. The IFC file acts as a core information repository, whilst DSMs act as a bridge between the standard IFC model and the specific simulation demands.

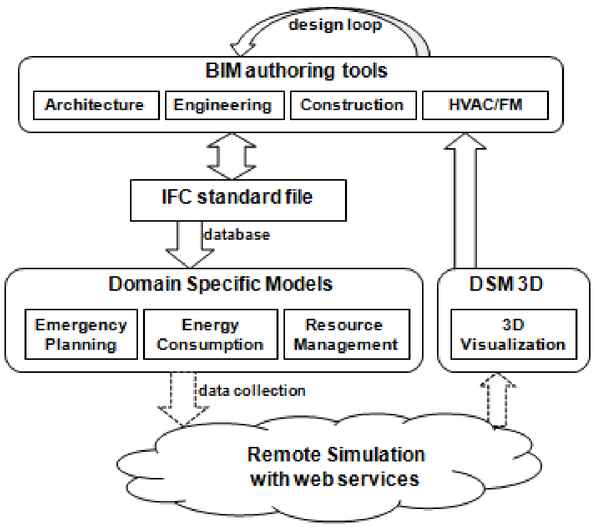


**Figure 1.** Generic Scalable BIM&S Framework

Generally, the process is as follows: 1) At the beginning of each cycle, various design teams collaborate and share multidisciplinary design information. They use a number of BIM authoring tools for CAD drawings, which are object-oriented, attribute-driven with 3D geometric information. 2) These BIM authoring tools support the use of an IFC standard file that acts as a core information repository for collaborating and communicating. Nowadays, several BIM tools support this step for sharing and collaboration. BIMserver.org is a model-driven open-source server that adopts EMF to represent the IFC data. In this paper, we use it to manage and query the IFC file. 3) The IFC standard file then is parsed by EMF using the MDA approach and stored in a relational database. EMF includes a meta-model (Ecore) for describing models and runtime support (e.g., XMI serialization, efficient query API, etc.). This characteristic of manipulating EMF objects generically allows us to build DSMs for specific simulation demands. A DSM can contain different syntax, structure, and semantics than the corresponding IFC file. 4) After collecting data from the DSM, all the necessary inputs are uploaded to remote simulation resources with web services. Then users can retrieve the results after simulation completion. Here we use RISE (RESTful Interoperability Simulation Environment middleware). 5) Then, the simulation results are parsed to a DSM model for 3D visualization, with special visualization features developed in the BIM authoring tools.

## 4. IFC & DEVS REMOTE SIMULATION

Figure 2 shows the concrete steps for integrating IFC and DEVS remote simulation by employing the proposed Generic Scalable BIM&S Framework.

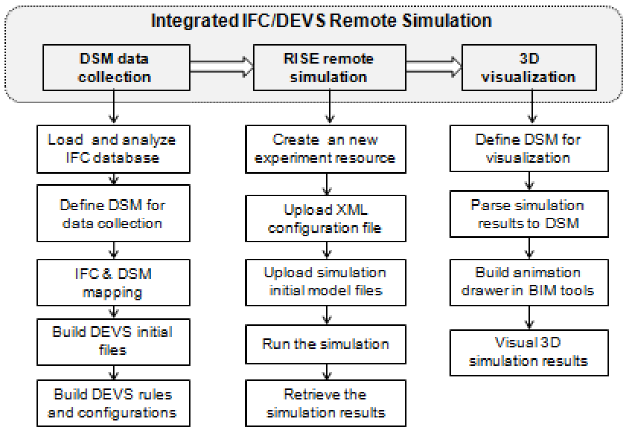

Integrated IFC/DEVS Remote Simulation
DSM data collection
RISE remote simulation
3D visualization
Load and analyze IFC database
Define DSM for data collection
IFC & DSM mapping
Build DEVS initial files
Build DEVS rules and configurations
Create an new experiment resource
Upload XML configuration file
Upload simulation initial model files
Run the simulation
Retrieve the simulation results
Define DSM for visualization
Parse simulation results to DSM
Build animation drawer in BIM tools
Visual 3D simulation results

**Figure 2.** Integrated IFC and DEVS remote simulation

It is composed of three phases: DSM data collection, RISE remote simulation, and DSM visualization. These phases constitute a subset of the design cycle, with detailed operations shown in Figure 1. In the DSM data collection phase, we load and analyze the IFC database, define our DSM for a certain specific simulation, then find the mapping between IFC and the DSM and build initial files for the remote simulation. Then we follow the API of RISE, creating resources, uploading files, executing the simulation, and retrieving results. Next, we parse the simulation results and visualize them back in certain BIM authoring tools. Here we will elaborate on the most important operations in this process.

### 4.1. Industry Foundation Classes (IFC)

The IFC hierarchy covers the core project information, such as building elements, the geometric and material properties of building products, and so on (Fu et al. 2006). The most abstract entity in the IFC model is *IfcRoot*, which is in the kernel and provides attributes for the identification, ownership, and self-description of all sub-entities. Figure 3 is a hierarchical diagram of a part of the Ifc2x3 EXPRESS standard under *IfcRoot*, which is also similar to Ifc2x4.

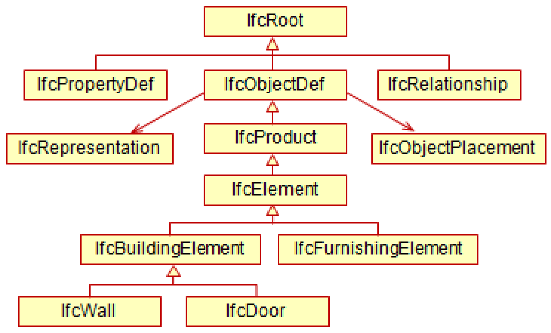


**Figure 3.** Part of the IFC hierarchy under IfcRoot

As we can see, *IfcRoot* is the parent for three subtypes, *IfcObjectDef* (generalizing any object or process), *IFcPropertyDef* (defining characteristics to entities), and *IfcRelationship* (describing relationship between entities). All these elements are identified from the super-class *IfcObjectDef*. The building elements category has fifteen groups of elements (e.g., walls, doors, stairs, beams, columns, etc.). Each element inherits attributes from all of its parent entities, such as geometric representation information (*IfcObjectPlacement* and *IfcRepresentation*). *IfcObjectPlacement* stores the placement information of an object. It contains the coordinates (x,y,z) which could be absolute (relative to the global coordinate system), relative (relative to the *IfcObjectPlacement* of another object), or constrained (relative to the grid axes). *IfcRepresentation* stores the shape representations of an object. There are two geometric shape representations: 1) the explicit geometric representation of the shape (i.e., points, curves, surfaces, and solid boundaries; for example, a wall could be defined with 8 points, 12 edges and 6 surfaces.); and 2) the attribute-driven representation of the shape (i.e., extruded profiles, directions, length, and width; for example, a wall could have length, width and height).

Each specific element could have particular attributes inherited from *IfcObjectPlacement* and *IfcRepresentation*, such as *IfcWall* presented in Figure 4.

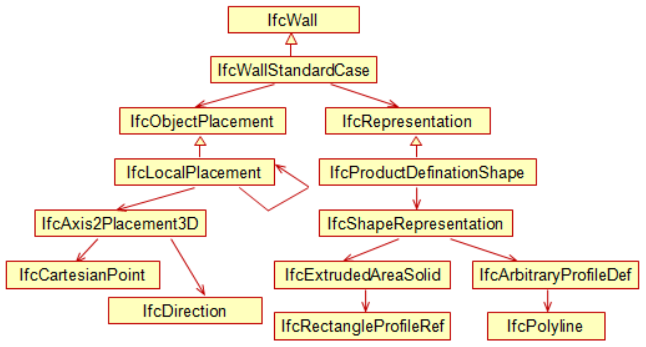


**Figure 4.** Building element example IfcWall

Inherited from *IfcWall*, *IfcWallStandardCase* contains the standard information of each wall. *IfcWallStandardCase* contains *IfcLocalPlacement* and *IfcProductDefinationShape* (the subclass of *IfcRepresentation*). *IfcLocalPlacement* could have not only the attribute of *IfcAxis2Placement3D* with *IfcCartesianPoint* (coordinates) and *IfcDirection* (the trend of spreading) that are relative to the global coordinate system, but also could have the reference of another *IfcLocalPlacement* with relative coordinates. *IfcProductDefinationShape* contains *IfcRectangleProfileRef*, with length, width and height, and *IfcPolyline*, with boundary points. In other words, *IfcLocalPlacement* represents the starting point, while *IfcProductDefinationShape* indicates where we can place it (in which way and how far). Assume a wall with length, width and height of (8"x2"x4") starts from (3, 2, 1) towards the direction of the Y axis with its length, the X axis with its width and the Z axis with its height. The point diagonally opposite the starting point would then be (x-width, y+length, z+height) = (1, 10, 5).

## 4.2. Domain Specific Models (DSM)

From the above analysis, we can see the generality and complexity of IFC, which might not cover all the requirements of a specific simulation. More importantly, there may be inconsistencies of syntax, structure and semantics. Hence, we propose the use of the MDA approach to represent the model for specific simulations and to extract/map/build DSMs from IFC data. Figure 5 shows an example of a DSM for this purpose.

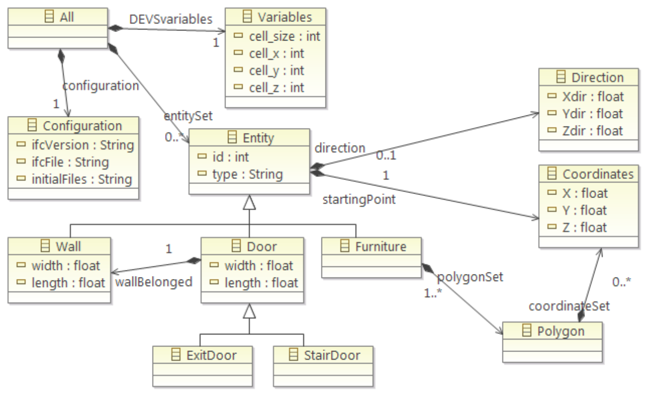


**Figure 5.** An example of DSM for simulation

This particular DSM is meant for tracking simulated occupants as they move through a building. Using a Cell-DEVS model, the simulated occupants move along a lattice of cells that facilitates pathfinding and collision avoidance. The class *All* consists of *Configuration*, *Variables* and *Entities*. *Entity* is the super-class for all other sub-entities, consisting of attributes of ID (sequence number), type (e.g., materials, size) and references to a direction and a starting point with absolute coordinates. *Entity* can generalize *Wall*, *Door* and *Furniture*. Apart from inherited attributes, both *Wall* and *Door* have width and length attributes. *Door* has two subclasses of *ExitDoor* and *StairDoor*. Due to the irregularity of *Furniture*, it can have a set of *Polygons* with the coordinates of turning points. *Configuration* and *Variables* are specific information for DEVS/Cell-DEVS simulations, like ifcVersion, ifcFile, initialFiles, numbers and units of lattice cells, etc.

In order to collect the necessary data, we need to find the mappings of elements from IFC to the entities of the DSM, shown in Table 1. As we can see, they look similar but they have differences of syntax and semantics. The DSM is more compact and smaller. The class names are similar but not exactly the same (*Wall* vs. *IfcWall*), and attributes of these classes are not identical; for example the Wall in the DSM is a subset of the combination of *IfcWall* (ID and type) and *IfcRectangleProfileRef* (width and length). One of the biggest differences is between *Direction* and *Coordinates*. In DSM all attributes are absolute to a global coordinates system, while in IFC there are different levels of relative coordinates, which require converting from relative coordinates to absolute coordinates.

| DSM | IFC |
|---|---|
| Entity | IfcElement |
| Wall | IfcWall, IfcRectangleProfileRef |
| Door | IfcDoor, IfcRectangleProfileRef |
| Furniture | IfcFurnishingElement, IfcPolyline |
| Direction | IfcLocalPlacement, IfcCartesianPoint |
| Coordinates | IfcLocalPlacement, IfcDirection |

**Table 1:** Mappings examples between DSM and IFC

In current implementation, Figure 6 illustrates the main classes of this DSM Data Collection tool (for clarity, the data model classes are not presented, i.e., the IFC elements in Figure 3 or DSM entities in Figure 5).

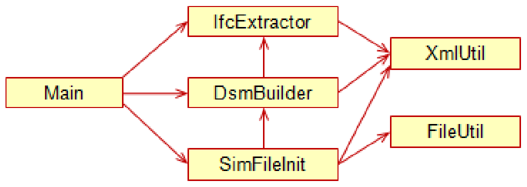


**Figure 6.** Part of Class Diagram of DSM Data Collection

*Main* takes control of the data collection process. As mentioned before, MDA technologies can parse the IFC file into a database with a UML class hierarchy based on EMF, from which we can query and filter more easily. *IfcExtractor* uses related APIs in the open source BIMServer.org. It loads the IFC data model into a database for extracting necessary information (e.g., wall, doors, furniture). *DsmBuilder* is responsible for constructing the defined DSM instance based on the extracted information and mappings list in Table 1. For example, to construct the *Wall* in DSM, we query ID and type from *IfcWall* and width and length from *IfcRectangleProfileRef*. Then, *DsmBuilder* serializes this DSM with the XMI (or, more general, XML) format for scalability and simplicity by calling *XmlUtil*. *SimFileInit* enables us to build the initial files for the specific simulation automatically. The macro variables could be calculated with size/number of cells and coordinate boundaries. The basic idea of initializing the state of each cell is to include all needed elements according to their type priorities (e.g. wall, furniture, stair, door). For each type, we find all elements belonging to it; for each element, we calculate all passing coordinates according to its starting coordinates and attributes (e.g. directions, length, width). Finally, we assign the state value to the cells that hold these coordinates.

In reality, IFC and other DSMs could have lots of many-to-many relationships or semantic inconsistencies that make it difficult to find mappings. To accommodate a DSM that represents the real world in a fundamentally different way than IFC, this work could be expanded using existing model transformation techniques in MDA. In (Roman et al. 2010), a semi-automatic framework is presented to build mappings of two meta-models and perform run-time model transformation automatically. In (Wang et al. 2011), a platform-independent approach towards semantic interoperability is proposed to analyze various kinds of semantic inconsistencies (type, scale, precision, synonym, homonym, granularity, and coverage), and address many-to-many mappings using heuristic similarity analysis.

## 4.3. Restful Interoperability Simulation Middleware

As discussed in previous sections, RESTful Web Services can further increase the scalability in terms of the plug-and-play interoperability and availability. In (Al-Zoubi and Wainer 2010), the authors presented the first existing RESTful Interoperability Simulation Environment (RISE) middleware. RISE is accessed through Web resources (URIs) and XML messages using HTTP methods. Implementations can be hidden in resources, which are represented only via URIs. Users can run multiple instances as needed, which are persistent and repeatable by specific URIs. The HTTP methods are typical four types: GET (to read a resource), PUT (to create or update a resource), POST (to append data to a resource), and DELETE (to remove a resource). An interface between RISE and CD++ allows running DEVS/Cell-DEVS distributed simulations. RISE API likes a classic URL http://www.example.com/cdpp/sim/workspaces, attached with the following services:

- **../{userworkspace}/{servicetype}/{framework}** allows interacting with a framework (including the simulation's initial files, configuration and src code).
- **../{userworkspace}/{servicetype}{framework}/simulation** interacts with the simulator execution.
- **../{ userworkspace}/{servicetype}/{framework}/results** contains the simulation outputs.

For a specific simulation, we can realize the remote simulation by using these URIs with HTTP methods. For example, after getting initial model and configuration files for simulation, we use PUT to create a framework with the configuration file and POST to upload these initial model files to this framework. Then, this newly created simulation environment can be executed by using PUT to *{framework}/simulation*, then we wait for the simulation to finish and GET the simulation results files from *{framework}/results*.

### 4.4. 3D Visualization

3D visualization provides a way to obtain visual simulation results in BIM authoring tools, enabling the designers to check the performance of the current design, find the flaws, and redesign for a next iteration cycle. The main reason for using 3D rather than 2D is that it is more intuitive and attractive. Most BIM authoring tools support full-featured 3D visualizations of buildings. Among them, Autodesk 3ds Max is a powerful BIM tool for its animation and rendering ability for 3D visualization. The IFC file can be easily imported into 3ds Max, what we want is to view the simulation results superimposed on the building model. To do this, we continue using MDA technologies, by parsing the results into a DSM that contains necessary but minimal information about changes to cells locations and values. Then we build a GUI written in MAXScript, enabling the user to load the parsed file and Cell-DEVS model. Finally we visualize it in 3ds Max using our advanced plug-in with different animation models.

## 5. CASE STUDY: EVACUATION PLANNING

In this section, we show a case study of evacuation planning by employing the proposed framework. We adapted and updated our previous work (Wang et al. 2012). In recent years, studying emergency evacuation has become an important design step for successful construction projects, which aims to analyze bottlenecks in order to minimize evacuation time. To do so, we applied the integration method proposed in this paper. It uses a Cell-DEVS model of the evacuation process of a multi-floor building. The current implementation uses the CD++ toolkit as the Cell-DEVS modeling environment, RISE as the remote simulation middleware, Autodesk Revit Architecture and Autodesk 3ds Max for the BIM tools, and BimServer.org for IFC data management and MDA. When an iteration cycle starts, designers from different domains could use BimServer.org to share their collaboratively developed IFC model. They could find the newest version of the IFC and load it into Revit, as shown in the left side of Figure 7.

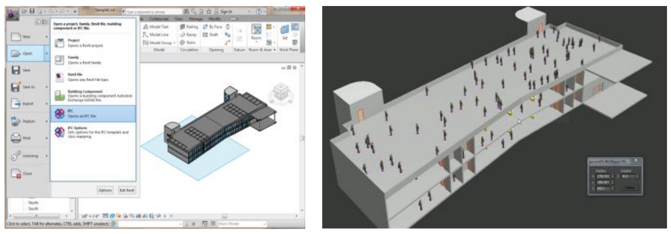
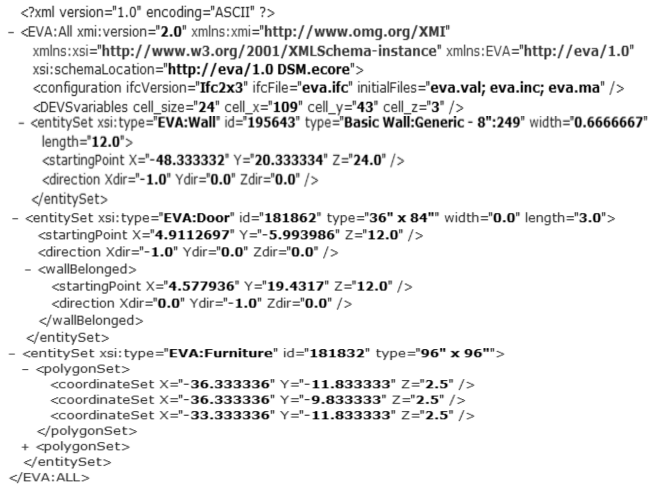


**Figure 7.** IFC loading in Revit (left) and 3D in 3ds Max (right)

Then analysts load the IFC file into an EMF database and extract data from it into a DSM model using our developed tool, which is currently specific to multi-floor evacuation. Figure 8 shows a part of the results of a DSM XMI file (an instance of the DSM presented in Figure 5) after the data extracting from IFC. It consists of all needed simulation information from the building. As we can see, it records the information for configuration (e.g., *ifcVersion*, *ifcFile* and *initialFiles* names), DEVS variables (e.g., *cell_size* with 24, the number of cells in dimensions of 109x43x3), and all needed entity information (e.g. *Wall*, *Door*, *Furniture*). For example, the Wall shown in Figure 8 has *id* (19564), *width* (0.67), *length* (12.0), *startingPoint* (-48.3, 20.3, 24.0) and *direction* (-1.0, 0.0, 0.0, which means its length is oriented towards the negative X axis).



```
  <?xml version="1.0" encoding="ASCII" ?>
- <EVA:All xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:EVA="http://eva/1.0"
    xsi:schemaLocation="http://eva/1.0 DSM.ecore">
    <configuration ifcVersion="Ifc2x3" ifcFile="eva.ifc" initialFiles="eva.val; eva.inc; eva.ma" />
    <DEVSvariables cell_size="24" cell_x="109" cell_y="43" cell_z="3" />
  - <entitySet xsi: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>
  - <entitySet xsi: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>
- <entitySet xsi:type="EVA:Furniture" id="181832" type="96" x 96"">
  - <polygonSet>
      <coordinateSet X="-36.333336" Y="-11.833333" Z="2.5" />
      <coordinateSet X="-36.333336" Y="-9.833333" Z="2.5" />
      <coordinateSet X="-33.333336" Y="-11.833333" Z="2.5" />
    </polygonSet>
  + <polygonSet>
  </entitySet>
</EVA:ALL>
```

**Figure 8.** DSM instances of evacuation simulation after IFC extracting

Based on the DSM XMI file, the tool can initialize the files for the remote simulation, including *.val (initial values of each cell) and *.inc (macro variables). With the RISE server, we can use the PUT method to create a framework for the evacuation simulation, and the POST method to upload the initial model files. We can run this simulation by using PUT to *evacuation/simulation*. We wait until the simulation finishes, then GET simulation results files from *evacuation/results*. Finally, after parsing the simulation log file into a DSM for the specific 3D visualization, we can load the same version of the IFC file in 3ds MAX and visualize the simulation animations with the help of its GUI. The right part of Figure 7 shows how the 3D visualization would look. According to the performance observed from 3D visualization, all steps are scalable and easily repeated for multiple design iterations.

## 6. CONCLUSION

To solve the scalability issues that arise when integrating BIM and simulation, we have proposed a Generic Scalable BIM&S Framework using the IFC standard, DSMs generated with a MDA approach, RISE middleware, and 3D visualization. We developed a tool for extracting information from IFC, building DSMs, and constructing the initial simulation files. Our case study uses a Cell-DEVS model of the evacuation process of a multi-floor building, using Autodesk Revit, BimServer.org, CD++, RISE, and Autodesk 3ds Max. The study shows that this kind of system can enable designers to extract information automatically from IFC into a specific simulation model, and run the simulation remotely through web services. Finally, designers can visualize the 3D simulation results in a BIM authoring tool to make further decisions or plan the next design iteration. Some future directions for this work are to use MDA model transformation techniques to automatically find mappings to provide a one-stop workflow to automate the entire proposed process.

## References

Ahmed, A., Wainer, G., and Mahmoud, S. 2010. Integrating Building Information Modeling & Cell-DEVS Simulation. In Proceedings of SimAUD 2010. Orlando, FL

Al-Zoubi, K., and Wainer, G. 2010. Distributed Simulation using RESTful Interoperability Simulation Environment (RISE) Middleware. Intelligence-Based Systems Engineering, pages 129–157.

Beetz, J., Berlo, L.V., Laat, R.D., and Bonsma, P. 2011. Advances in the Development and Application of an Open Source Model Server for Building Information. In Proceedings of the CIP W78 conference, Nice.

Duboc, L., Leiter, E., and Rosenblum, D. 2012. Systematic Elaboration of Scalability Requirements through Goal-Obstacle Analysis. IEEE transactions on software engineering

Eastman, C., Teicholz, P., Sacks, R., and Liston, K. 2008. BIM Handbook. Wiley & Sons.

Fu, C., Aouad, G., Lee, A., M-Ponting, A., and Wu., S. 2006. IFC model viewer to support nD model application. 2006 Automation in Construction, 15 (6), pp. 178-185.

Hensen, J. L. M. 2002. Simulation for performance based building and systems design: some issues and solution directions. In Proc. of 6th Int. Conf. on Design and Decision Support Systems in Architecture and Urban Planning, TUE, NL.

Hetherington, R., Laney, R., Peake, S., and Oldham, D. 2011. Integrated Building Design, Information And Simulation Modelling: The Need For A New Hierarchy. In: Building Simulation 2011, Sydney, Australia. November 2011.

Hitchcock, R. J., and Wong, J. 2011. Transforming Ifc Architectural View Bims for Energy Simulation: 2011. Proceedings of Building Simulation.

Jeong, Y.S., Eastman, C.M., Sacks, R., and Kaner, I. 2009. Benchmark tests for BIM data exchanges of precast concrete. 2009 Automation in Construction, 18 (4), pp. 469-484.

Jiang, Y., Ming, J., Wu, D., Yen, J., Mitra, P., Messner, J. I., and Leicht, R. 2012. Bim Server Requirements to Support the Energy Efficient Building Lifecycle. In 2012 ASCE in civil engineering.

Lipman, R.R. 2010. Developing Coverage Analysis For Ifc Files. Proceedings of the CIB W78 2010: 27th International Conference.

Malinowsky, B., and Kastner, W. 2010. Integrating Process Communication in Building Information Models with IFC and LON. 8th IEEE International Workshop on Factory Communication Systems.

Ribault, J., and Wainer, G. 2012. Simulation process In the Cloud For Emergency Planning. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

Roman, D., Morin, B., Wang, S. and Berre, A. J. 2010. A Model-driven Approach to Interoperability in B2B Data Exchange. Advanced results in MDI/SOA innovation Workshop, MDI.

Sanchez, P., Barreda, J., & Ocon, J. 2008. Integration of domain-specific models into a MDA framework for time-critical embedded systems. In Intelligent Solutions in Embedded Systems, pp. 1-15. IEEE.

Wainer., G. 2009. Discrete-event Modeling and Simulation: a Practitioner's Approach. CRC/Taylor & Francis.

Wainer, G., Al-Zoubi, K., Dalle, O., Mittal, S., Martín, J., Sarjoughian, H., Zeigler, B.P. 2010. Standardizing DEVS Simulation Middleware. Ch. 18, Discrete-Event Modeling and Simulation: Theory and Applications. Eds G. Wainer, P. Mosterman. CRC Press. Taylor & Francis.

Wang, S., Morin, B., Roman, D., and Berre, A.J. 2011. A Semi-automatic Model Transformation Approach for Semantic Interoperability. Proceedings of NATO Symposium & Workshop on Semantic & Domain Based Interoperability. Norway.

Wang, S., Schyndel, M.V., Wainer, G., Subashini, V., and Woodbury, R. 2012. DEVS-based Building Information Modeling And Simulation for Emergency Evacuation. Proceedings of the WSC. Germany. IEEE.

Won, J., and Lee, G. 2011. Algorithm for Efficiently Extracting IFC Building Elements from an IFC Building Model. In Proc. 2011 ASCE workshop on computing in civil engineering.

Zeigler, B.P., Praehofer, H., and Kim, T.G. 2000. Theory of Modeling and Simulation. Academic Press.