Hindawi Publishing Corporation
Journal of Construction Engineering
Volume 2014, Article ID 847430, 12 pages
http://dx.doi.org/10.1155/2014/847430

Research Article

Hybrid Simulation Environment for Construction Projects: Identification of System Design Criteria

Mohamed Moussa,[1] Janaka Y. Ruwanpura,[1] George Jergeas,[1] and Tamer Mohamed[2]

[1] *Centre for Project Management Excellence, Schulich School of Engineering, University of Calgary, Calgary, AB, Canada T2N 1N4*
[2] *Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB, Canada T2N 1N4*

Correspondence should be addressed to Mohamed Moussa; mohmoussa@shaw.ca

Received 27 January 2014; Accepted 28 May 2014; Published 29 June 2014

Academic Editor: Eric Lui

Large construction projects are complex, dynamic, and unpredictable. They are subject to external and uncontrollable events that affect their schedule and financial outcomes. Project managers take decisions along the lifecycle of the projects to align with projects objectives. These decisions are data dependent where data change over time. Simulation-based modeling and experimentation of such dynamic environment are a challenge. Modeling of large projects or multiprojects is difficult and impractical for standalone computers. This paper presents the criteria required in a simulation environment suitable for modeling large and complex systems such as construction projects to support their lifecycle management. Also presented is a platform that encompasses the identified criteria. The objective of the platform is to facilitate and simplify the simulation and modeling process and enable the inclusion of complexity in simulation models.

1. Introduction

Building a computer simulation model requires specialised knowledge in software engineering and modeling. Modifications to models are difficult to implement. Simulation is generally regarded by professionals in the construction industry as an additional layer to the business software environment. Feedback from industry practitioners and simulation researchers shows that minimizing the specialized knowledge and simplifying the modeling process are necessary to increase the appeal to simulation [1]. Simulation modeling methodologies for construction projects are developed around modeling of repetitive/cyclic operations [2]. Such models are small compared to those required to model a complete construction project. Projects are data dependent where data are updated periodically by actualizing the data of completed work and reforecasting future work. Models should be current with latest data. Projects are managed by people who influence the project outcomes by their decisions. Projects are also affected by external and internal events that could change their schedule and financial results. Neglecting the inclusion of decisions and events in the model results in unrealistic project forecast.

Presented in this paper are the criteria used in the development of a simulation environment that is aimed at simplifying the modeling process and facilitating the modeling of construction systems to enable practitioners to use simulation as integrated technology during their project management. We achieve this objective by incorporating agent-based modeling, network modeling, object oriented paradigm, discrete event simulation, Monte Carlo analysis, and distributed calculation in one framework and by integrating simulation models with databases. In this paper, we focus on presenting and reasoning out the design criteria without addressing the implementation or the detailed possible applications.

2. Background

Simulation programs require specialized knowledge, high level of programming ability, and time for building and

analyzing the models [1, 3]. Software engineering remains the obstacle to using simulation [4]. One notable application of simulation in project management is the assessment and quantification of risks ([5], p. 299) and modeling of cyclic operations [2]. Simulation is "highly advantageous for designing construction operations and can be of great help to project manager" [6]. Nevertheless, simulation-based applications in construction project management lack industry support [2], which demonstrates the need to simplify the modeling process and demonstrate the advantage of using simulation in project management. This section provides a background on simulation and suggests areas of improvements to facilitate the utilization of simulation in managing construction projects.

2.1. Computer Simulation. Meredith et al. [7] define simulation as "the process of conducting experiments with a model of the system that is being studied or designed" (p. 228). Pritsker and O'Reilly ([8], p. 6) described computer simulation as "the process of designing a mathematical/logical model of a real world system and experimenting with the model on a computer." Simulation allows for modelling and testing concepts and ideas ([9], pp. 7 and 8). Fujimoto ([10], p. 27) defines computer simulation as a system that emulates another system using computer. Sadowski and Grabau [11] define success of a simulation application as one that delivers "useful information at the appropriate time to support meaningful decisions." Provision of the simulation-based model at the appropriate time requires a simulation environment that enables fast modeling, fast data input, and less human interaction. Data acquisition, input, and analysis take the most time in the modeling lifecycle [12]. Simulation applications for construction projects did not pass the planning and design stage of the projects with lack of applications during project performance [2].

The simulation process is an iterative procedure that may be described based on "inputs and outputs with feedbacks" ([7], p. 228). Inputs define the set of values, events, and conditions that can affect the system in the real world and outputs predict the system responses. By studying the outputs, modelers learn more about the system and may use this knowledge to define new sets of inputs.

2.2. Historical Background. Computer simulation requires knowledge of a special language that allows the programmer to simulate the system ([13], p. 614). Early simulation-based models were written by programming languages such as FORTRAN [14]. Models were manipulated at the same level of the programming languages in which they were created [14]. Programming languages created for simulation were many which made it impossible to decide "which language is best fit for a specific problem" [15]. The programmer would design the model and provide the services to draw and perform the logical and mathematical functions. Languages were procedural or machine languages.

Object oriented programming fits simulation better than the procedural languages where systems are designed as a group of objects. One important outcome is the "library-elements" concept where systems are designed as set of classes grouped in a container/library of icons [1]. The simulation environment is created using a computer language that the user need not be aware of but provides functions that the modeler can use to program the icons. The object oriented paradigm created collaboration between two types of skills: domain experts and simulation engineers. The experts define the problem and data required and the engineers develop the problem logic and specifications into the simulation computer program [16]. By the library-elements concept the developers build libraries of elements/icons that can be reused in modeling systems of the same characteristics. Modelers drag and drop the icons from the library on a drawing window to construct the model (similar to using Microsoft Visio). Libraries are programmed once and users may build many organizations of models using these icons. The "prevailing approach" of modeling construction projects is the library-icons concepts using discrete event processes [6]. However, programming the libraries requires strong programming knowledge and a simulation engineer to develop these libraries [17].

Development of a simulation-based model can be "tedious and expert knowledge on simulation technique is required" [17] and can also be time consuming [18]. In an attempt to simplify the simulation-based modeling process in construction, some methodologies were developed. Halpin [19] introduced CYCLONE (CYClic Operations Network) as a simulation method for planning and analyzing construction operations that are repetitive [20]. CYCLONE was used to build simulation platforms such as INSIGHT [21], RESQUE [22], UM-CYCLONE [23], STROBOSCOPE [24], and DISCO [25]. CYCLONE uses graphical network modeling [20] as a general purpose simulation (GPS) environment. Reference [26] developed Simphony as a new development from CYCLONE for modeling construction operations and the concept was presented as a special purpose simulation (SPS) for domain specific modeling. Programmability of platforms such as STROBOSCOPE and Simphony makes them extendable to be used in a wide range of applications.

The object oriented programming paradigm resulted in passive models that do not take decisions and lack the intelligence to adapt to changes in its environment. Agent-based modeling solves this problem (refer to Wooldridge [27], pp. 25–27, for the difference between object oriented and agent-based concepts). Agent-based modeling is a philosophy of modeling where decision makers are conceptualized as agents in the modeling environment [28]. There is no universal definition of an agent; however, it is agreed that agents are characterized by autonomous behavior and ability to take decisions [29]. Agent-based modeling is therefore suitable for modeling systems where decisions are made based on changes in the system and where the system is subject to events that may change the system structure and outcomes (see [30] for the use of agent-based modeling in complex dynamic systems). We incorporate in our proposed framework the agent-based modeling in an attempt to emulate construction system more accurately.

2.3. Summary of Construction Projects Modeling Challenges. Abourizk et al. [6] summarize the difficulty in using simulation in three points: (1) identification of the logic of the model, (2) developing the simulation algorithm, and (3) availability of software tools and applications. The first relates to lack of modeling and analysis skills and the second and third address the lack of software skills that [4] is attributed to absence of software training in engineering schools. Centeno and Carrillo [3] also agree that unavailability of skills is one of the obstacles to introducing simulation to industry. It is not surprising that simulation has been an academic tool since the 1960s with no real success in construction industry [20]. A simulation survey presented by Hlupic [31] revealed many simulation platforms limitations, such as unsuitability for complex and nonstandard problems, much expensiveness, difficulty to learn, lack of integration/compatibility with other packages such as database management tools, and lack of flexibility. Similar problems were reported by Mohamed and AbouRizk [32], Centeno and Carrillo [3], and AbouRizk [2]. Also, simulation platforms lack the services necessary to incorporate decisions and events to emulate the construction projects correctly which might have contributed to hindering the ability to demonstrate the usefulness of simulation to industry professionals.

The above discussion leads to two areas of challenges.

(1) Application challenges related to correctly emulating complex and large systems in a unified modeling philosophy.

(2) Technical challenges related to speeding and simplifying the modeling process.

The rest of the paper addresses the above two areas and discusses the criteria incorporated in a simulation platform developed to respond to them.

3. Design Criteria of the Simulation Environment

Construction projects are complex, dynamic, and objective-oriented systems. They consume resources and take time to complete. To achieve their business objectives, organizations running many construction projects evaluate their projects as a portfolio. This section presents the properties of construction projects to deduce the criteria of the simulation environment suitable for modeling construction projects and presents these criteria.

3.1. Characteristics of Construction Projects and Related Simulation Environment. A construction project is planned, organized, and performed in work packages. Work packages are organized in a scheme defined in a top-bottom tree of work breakdown structure (WBS). Work packages are also planned in a dependency network of successor/predecessor networks. Projects are influenced by internal and external factors such as labour productivity, site conditions, weather, and regulatory and political factors. Therefore a simulation environment for construction projects should provide the infrastructure to (1) model the work packages at different levels of details and abstraction levels vertically (organizational) and horizontally (dependency); (2) incorporate the decisions and the decision criteria and permit the response to these decisions; and (3) include the forces and events that affect the project outcomes. Additionally, organizations do not manage projects in isolation of other projects. Projects consume and compete for resources and are subject to uncertainty in planning and execution. To respond to these characteristics, the simulation environment should facilitate modeling of networks in discrete event simulation and resource utilizations in a standalone and distributed computing facility to permit modeling of a portfolio of projects.

Projects progress over time where large projects span many years. At frequent time intervals, the project team measures project performance, actualizes data of completed work, changes plans, redefines resources, and reassesses the external and internal forces affecting the project. Models are built to emulate a project and the environment in which it is built at a point in time. As the project progresses and plans are updated the simulated environment should reflect the project status and the new planning assumptions. Experimenting with a simulation model requires the ability to test scenarios and conduct sensitivity analysis of different options. The modeling/remodeling process should be easy and speedy with minimum manual data input and model restructure efforts. Similarly, programming effort needs to be done with minimum code writing requirements.

The above discussion leads to the following five criteria as requirements in a simulation environment serving construction projects (refer to Kluegl et al. [33] for the difference between simulated and simulation environment) to serve the management of large construction projects.

(1) Modeling architecture: modeling of networks vertically (bottom-up and top-bottom designs) and horizontally (dependency/relational) to represent construction operations and organizations hierarchy.

(2) Modeling concepts: use the library-elements concept, the agent-based philosophy, Monte Carlo analysis, and discrete event simulation (DSE) with resource utilization analysis with a queue and process calculation algorithm (refer to [9], pp. 11–17, for more on DSE).

(3) Programming paradigm: object oriented programming modified to suit agent-based modeling philosophy and automating the generation of simulation libraries and icons.

(4) Modeling life cycle management: importation and exportation capabilities to databases and automation of data input and documentation activities.

(5) User interface: graphical user interface for drawing networks and retrieving simulation results and user-friendly menu system for analyzing the data.

3.2. Overview of the Modeling Components. A simulation model of a system consists of three categories: (1) modeling

elements representing parts of the system; (2) agents representing decision makers; and (3) triggers representing events that may take place during the life of the system and can affect the system. This section presents the three categories.

3.2.1. The Modeling Element. A modeling element represents a system or a component of a system. It has presentation, parameters, connections, and behavioural characteristics that describe it. Modeling elements may be connected by links to form a network that emulate the system components they represent. A modeling element does not take decisions. It can send messages carrying information about its state to other elements. Similarly, it can receive messages sent by other elements. Elements have both input and output properties. Input properties are the set of values that the modeler defines to describe the modeling element before simulation. Output properties represent the modeling elements response to methods, events, decisions, and other changes that take place in the system during the simulation process. Links connect the modeling elements to represent dependencies, flow of information, and/or sequence of processes in a system. A link connects two modeling elements and therefore has two ends with a directional arrow indicating the flow of messages transfer, dependency, and/or sequence. Links function as pipelines to transfer messages from one modeling element connected to the tail side of the link to another connected to the head side. A modeling element can be described by the value of its parameters, the behaviours of its methods, and the relationships with other elements.

(1) Parameters describe the states of the modeling element. They are either user-defined or system-defined. User-defined parameters are the (i) graphical parameters that represent the pictorial characteristics such as the shape and colour of the element and the (ii) data parameters that represent the different states of the element. The data parameters are the input, output, and internal data parameters. Values of input parameters are assigned by the modeler at the steady state of the model before experimenting with it. The program uses these values to calculate the output parameters. Internal parameters are temporary holders of values that the developer uses to perform calculation during the simulation process. Internal parameters are not acceptable or retrievable by the modeler. System-defined parameters are read only parameters assigned by the simulation platform.

(2) Behaviours are methods (i.e., logical processes/programs) grouped in three categories; (i) system provided/system implemented methods that are provided by the simulation platform. The program processes these methods during performing calculation with no interference from the users; (ii) System provided/user implemented methods where the simulation environment requires the developer to write their implementation. The simulation environment provides their signature and the developer writes their code. These methods are invoked during simulation calculation in a predefined sequence; (iii) user-defined/user implemented methods that are created by the developer who also decides when to trigger them from other methods.

(3) Relationships describe elements connections and communication. There are three types of possible relationships among elements: (i) node/link relationship where an element connects to other elements in a network via connection points and directional links. A connection point is either "in" or "out" depending on the direction of message transfer (sending and receiving messages). An element may have multiple in/out connection points as well as many elements connected through one point. A connection point is programmatically an instant of a class. It holds references to the elements connected to it; (ii) parent/child relationship (a "one-to-many" relationship) where an element can have one parent but may have many children in multilevel modeling hierarchy; and (iii) remote relationship where two elements are not physically connected in the model through a link and are not in a parent/child relationship. An element can send messages to another element by address. An element can have remote relationship with an element in the same model, in a different model in the same computer, or in a deferent model in geographically separated computer. Elements communicate by exchanging messages. A message is programmatically an instant of an object that possesses user-defined parameters. The modeling element has a system-defined method that is invoked when a message is received and has a method that facilitates sending messages to elements in any of the three relationships.

3.2.2. Agents. An agent is an element that is knowledgeable of the simulation environment in which it exists. It can take decisions based on predefined criteria when changes take place in the environment. Programmatically agents can be embedded in modeling elements. A modeling element can take decisions by executing methods based on predefined criteria (such as by using an "if statement"). Agents can change the values of the properties of the other modeling elements and enforce behaviours as well as adding, deleting, or reconfiguring elements and relationships. Agents have input properties (normally decisions criteria) but do not have output properties because they are not affected by the environment in which they exist. Decision criteria may be stored in external databases or in the input properties of the modeling elements. They communicate with other modeling elements by sending and receiving messages. A simulation modeling environment should provide means to store decision criteria and allow the agents to access these criteria at the simulation run time.

3.2.3. Events/Triggers. Triggers are events that may happen during the simulation lifecycles. When an event is realized, interested modeling elements and agents need to be

notified by the simulation environment. Triggers are conditions/events that may arise during the system/project life and the interested modeling elements and agents respond to them by invoking actions/methods. Triggers can programmatically be represented in tables embedded in modeling elements or stored in external databases. Triggers can materialize randomly or as a result of other events or conditions.

4. Application of Hybrid Simulation Environment

The simulation environment criteria described in Section 3 were implemented in the design of a simulation platform. The platform consists of three components: (1) modeling element generator; (2) simulation modeling environment; and (3) simulation services. This section describes the characteristics of the three components.

4.1. Overview of the Simulation Environment. The simulation environment developed to incorporate the criteria presented in this paper consists of three components: (1) the elements generator, (2) the simulation modeling environment, and (3) the simulation services. The simulation and modeling process includes two main functions: (i) development (performed by a developer) and (ii) modeling (done by a modeler). Figure 1 shows a conceptual framework describing the relationship among the three components. The development process is automated. A developer creates a library of elements using the element generator. He/she decides the name and type of the input and output properties and the methods of each element. The modeler uses the simulation modeling environment to graphically design a model by dragging the elements from the library and dropping them onto a drawing board. Models can be constructed in up-down hierarchy in layers (multilayers). In each layer, the modeler adds elements and connects them using links to draw a horizontal network of the elements representing the logic, the sequence of operation, and/or the flow of information among the elements. The multilayer is a parent/children hierarchy that permits decomposing elements (parents) to smaller elements (children) in a containment relationship (i.e., the parent element contains the children elements) (see, e.g., Moussa et al. [34]). Each element has input parameters whose values are assigned by the modeler at the modeling time. The simulation platform invokes the methods to calculate the output properties in a predefined order controlled by the simulation services.

A model can be described as a container of elements that can be arranged in successor/predecessor and parent/children relationships. Elements are described by their parameters. The model can be saved and retrieved in a format readable by the simulation environment. The model data—elements, relationships, and the parameters values—can be exported to and imported from a database by using the simulation services. The simulation services control the simulation calculation process, the number of simulation runs, and the simulation time and provide the agents with access to the simulated environment information. The following

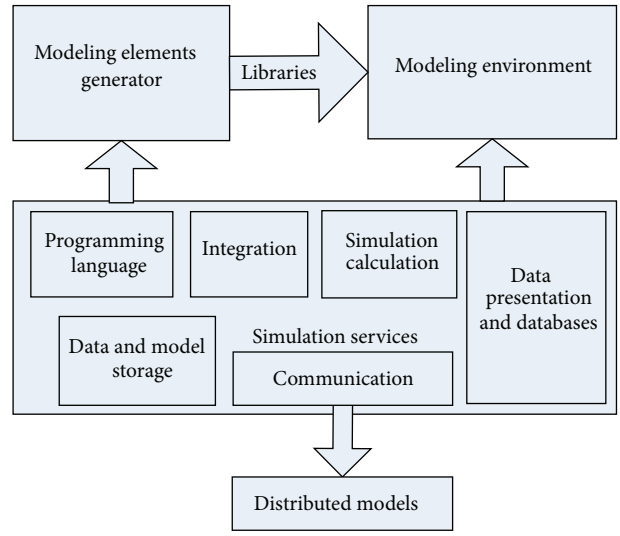

Figure 1: Conceptual framework of the simulation environment.

subsections present the three components as implemented in the simulation platform.

4.2. Modeling Elements Generator (MEG). The modeling element generator helps developers create libraries and modeling elements code free by translating menu selections into code. A library is a collection of modeling elements. The developer creates a library using the menu system and provides the library with a name of his/her choice (e.g., a library that contains modeling elements representing the components of a critical-path method may be named CPM). The developer creates the modeling elements using the menu system and groups them in a library. A modeling element is derived from a program provided class (base-element) using inheritance (refer to any object oriented literature for inheritance explanation). The base-element is used by the simulation environment as the blueprint from which all modeling elements are derived using inheritance. The user has the option to derive the newly generated modeling element from an existing modeling element in the library. By inheritance, the developer is able to use the parameters and methods of an existing modeling element without rewriting code. Using the menus of the MEG the user defines the following.

(1) The name of the element (the user may select a name that represents the modeling element in real life such as "activity" in a "CPM" library or "machine" in a "factory" library).

(2) The parent element (the modeling element from which to inherit; for example, the developer may inherit the modeling element "car" from the base-element and inherit the modeling element "truck" from the element "car" and so on).

(3) The elements parameters (input, internal, and output parameters) and their data types (i.e., integer, double, string, tables, etc.).

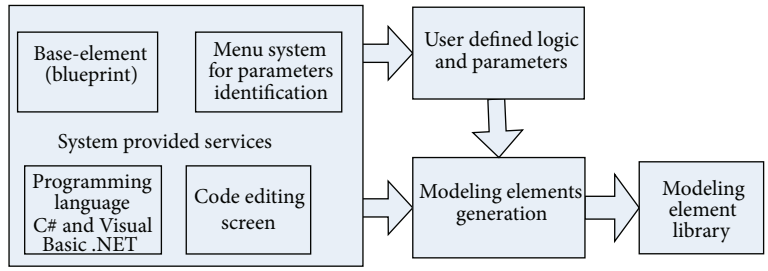

Figure 2: Conceptual framework—code generation architecture.

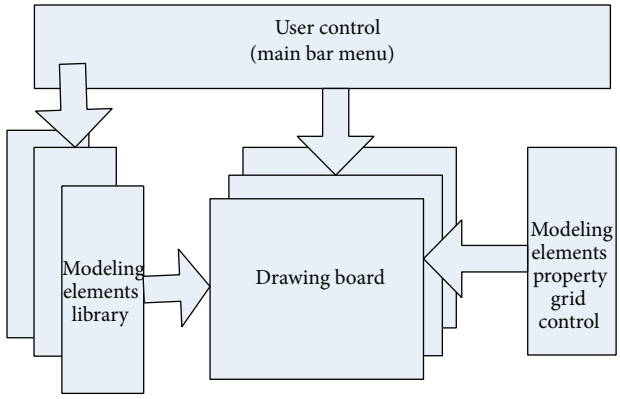

Figure 3: Modeling environment.

(4) The in/out connection points.

(5) The simulation discrete events and their time schedule.

(6) The code of the system-defined/user implemented methods.

(7) The user-defined methods.

The MEG uses the reflection capability of the .NET technology to create the modeling elements at run time (refer to .NET documentation for more information on reflection). Figure 2 demonstrates the relationship between the MEG and the simulation environment. The user defines the parameters, events, and methods using the simulation services (i.e., the code editor, the menu systems, the data input tables, the programming language compiler, and the base-element class). The MEG translates the user inputs into C# or Visual Basic .NET code using reflection. The developer designs the shape of the element using a graphical user interface of drawing design capabilities. The developer is not required to write the code to create the graphical presentation of the element, the parameters, the connection points, or the signature of the methods. The developer, however, needs to write the code of the model logic specific to the element. The MEG has a drawing board that has predefined basic shapes that can be used to design the shape of the element. Users may add pictures or icons to the element and use its property grid to define the pictorial properties of the element (refer to .NET documentation for information about the property grid).

4.3. Simulation Modeling Environment (SME). As shown in Figure 3, the SME consists of

(1) the modeling elements library;

(2) the drawing board;

(3) the menu system;

(4) the property grid.

To construct a model, users drag and drop an element from the library on the drawing board and connect the elements in a network. The menu system provides the means to access the simulation services such as importing from and exporting to Microsoft Access database. Modelers use the property grid to input values to the input parameters and retrieve the output properties. A modeler accesses the child window of an element by double clicking on the element. In the child window, the modeler can add and draw networks. An element has system provided properties that hold the value of its parent element, children elements, and predecessor/successor elements. The ME uses the simulation services to save and retrieve a model as well as export and import models to databases.

4.4. Simulation Services (SS). The simulation services (SS) provide the following functions to create and experiment with a model.

(1) Programming language compiler: users can use C# .NET or Visual Basic .NET.

(2) Database integration: users can export input parameters, output parameters, or the model relationships to databases. Users can also import the values of the input parameters and the model from a database. Database exportation allows analyzing and storing the model outside the simulation environment. Database importation speeds up and automates the modeling and updating process.

(3) Simulation type: models can be discrete event/process-based simulation in a standalone/distributed model(s). The discrete event simulation performs scheduling and management of events where the simulation time proceeds based on a clock controlled as a global variable managed by the simulation environment. The platform provides the means to create and manage resources and measurement of

the utilization of the resources. Modelers can choose to perform a standalone modeling or distributed modeling depending on the number of models involved in the simulated environment.

(4) Calculation control: the user decides the number of times he/she wants the simulation to run "n" and the simulation time "t." The simulation calculation process starts with iterating through the modeling elements to initiate the simulation initialization method of the elements. The simulation initialization method performs functions required before running calculation. It includes processes such as validating the model input values and/or the relationships among the elements. Afterwards, the simulation engine iterates "n" times though the elements. During each run, the simulation clock advances from simulation time zero to "t." Each simulation iteration starts by initializing the input parameters to their start-up values and performs functions written in the initialization method. At this stage, density function parameters are assigned a value based on the type of their density function and input values. Events are scheduled and the next procedure/event is invoked. As the simulation time advances events are triggered at their scheduled time and users-defined procedures are called per the program logic. At the end of the run, the postrun simulation function is called. When the program performs all the "n" iterations the function "postsimulation" is invoked. Figure 4 summarizes the simulation calculation steps described above.

(5) Random number generation of continues functions.

(6) Data storage: the simulation output results are stored within the element. The output results can also be stored in a database. The modeler defines the name of the database and the name of the table to store the results. The modeler can run the simulation using different input parameters and can store the simulation results in different tables inside the same database.

(7) Output presentation: output parameters hold the simulation results where there is a value for each simulation time/run. At the end of calculation, the values of the output parameters are displayed in tabular format and can be exported to a spreadsheet or a database table. The presentation module displays the cumulative density function (CDF) graph, the probability distribution function (PDF) graph, and the statistical results such as averages, means, and standard deviation for each variable collected.

(8) Distributed simulation (DS): DS allows modeling for several models located at different machines simultaneously using .NET Windows Communication Foundation technology. The modeling structure of the DS consists of identifying a master machine that controls the synchronization among the participating models, a server machine (could be any of the machines in the distributed models) that controls receiving/sending protocol of the messages, and the modeling machine that hosts the models. The SS provide the methods that allow models to send messages (objects that carry information) to other models in the network (based on address/name of the machine) and events that are triggered when messages are received. Synchronization among the models is currently to the simulation run number (i.e., the master machine controls the simulation runs so that all machines are running the same simulation run number at any time). Synchronizing the simulation time is part of the next development upgrade of the SS. With the current DS architecture of the SS,

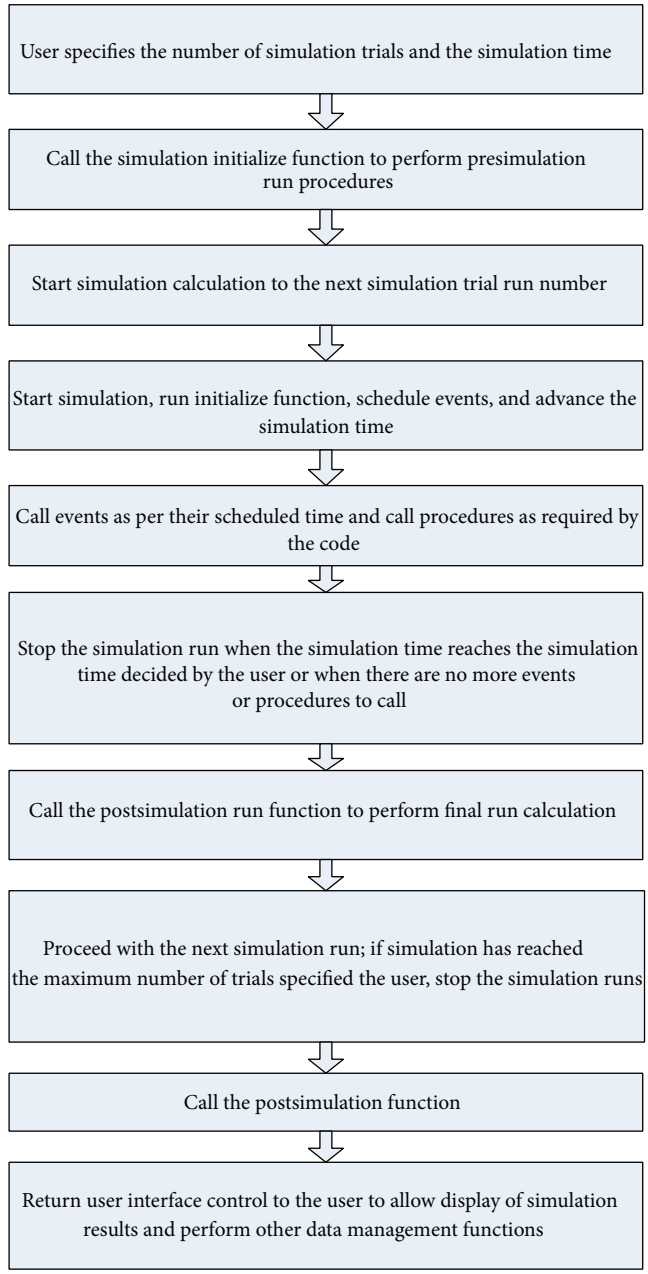

Figure 4: Simulation calculation procedure.

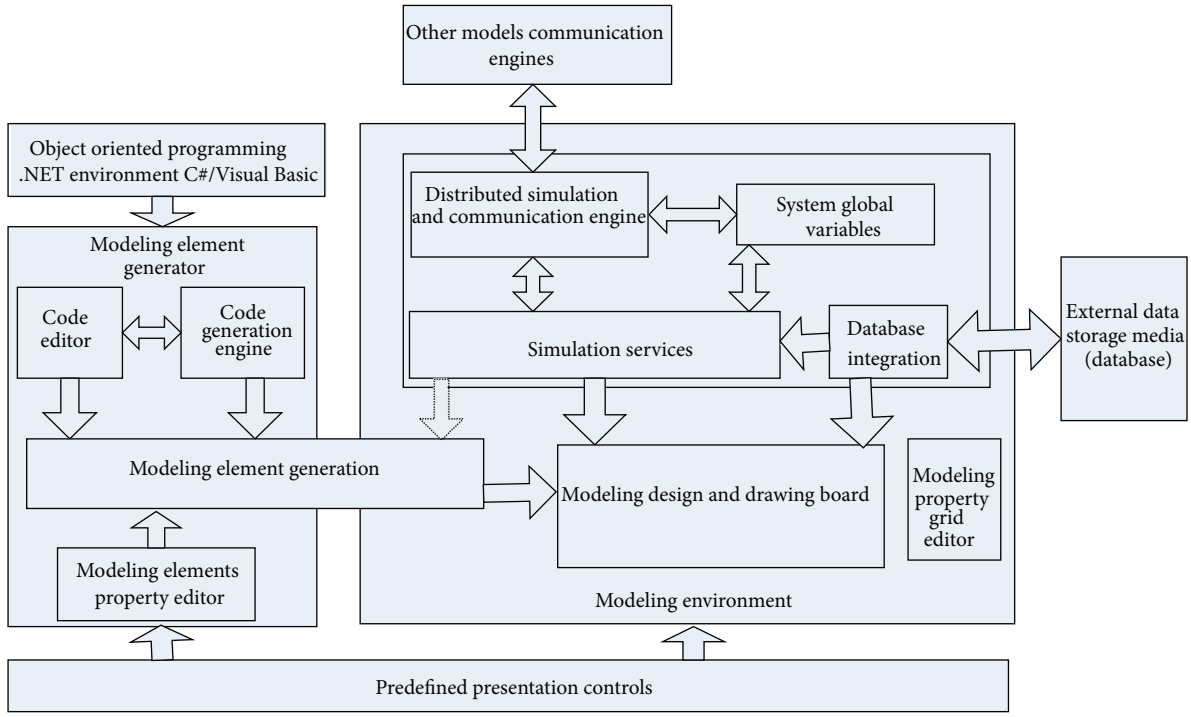

Figure 5: Simulation platform components.

several construction projects were modeled and the model located in the master machine collected the results of the models (e.g., cost and schedule values). Models sent messages to the master machine at the end of each simulation run carrying information about the model and the master machine collected this information to calculate the portfolio status as well as the status of each model in the DS.

(9) Agent-based/network/object oriented modeling: the SS provide services that allow for the agent-based concept such as the ability to expose the simulation environment parameters to agents and the ability to interact with databases at run time. Decision criteria and actions may be stored in tables that the agents can access and iterate through to invoke methods based on the conditions and information stored in the tables. The SS provide the means to draw networks consisting of nodes and links.

4.5. Integration among MEG, SME, and SS. The integration between the MEG, the SME, and the SS provides a unified environment for simulation modeling. Agent-based modeling allows for the inclusion of decision points and events. The SS utilize the reflection capability of the .NET framework to generate modeling elements at the run time reducing the need for specialized programming knowledge and provide integration with databases to speed up the modeling and remodeling (updating) process. Included in the unified environment is the ability to perform distributed simulation using the same modeling structure. Models can be modeled separately or as part of distributed model.

Figure 5 shows the three components of the platform and illustrates their role during the simulation process. A developer creates a library of the elements and agents required to model a system using the modeling elements code generator. The modeler drags and drops the elements from the library to the simulation modeling environment or may use data stored in a database to import the model or update the model input parameters using the SS. As summarized in Figure 6, the simulation modeling process comes in three steps.

(1) The developer generates the library (also called toolbox) and the modeling elements. A modeling element may be inherited from the base-element or from an existing element in the library.

(2) The modeler creates a model using the elements in the toolbox/library. The modeler may store and retrieve a model from a database or update the model with parameters stored in an external database.

(3) The modeler experiments with the model and if there is more than one model, the modeler may experiment with a collection of distributed models.

Developers require knowledge of C# .NET or Visual Basic .NET. Modelers do not need programming knowledge but should have a basic computer understanding to use the menus and be knowledgeable of the problems being modeled.

4.6. Applications. The criteria described in this paper are intended to facilitate modeling of large systems that are

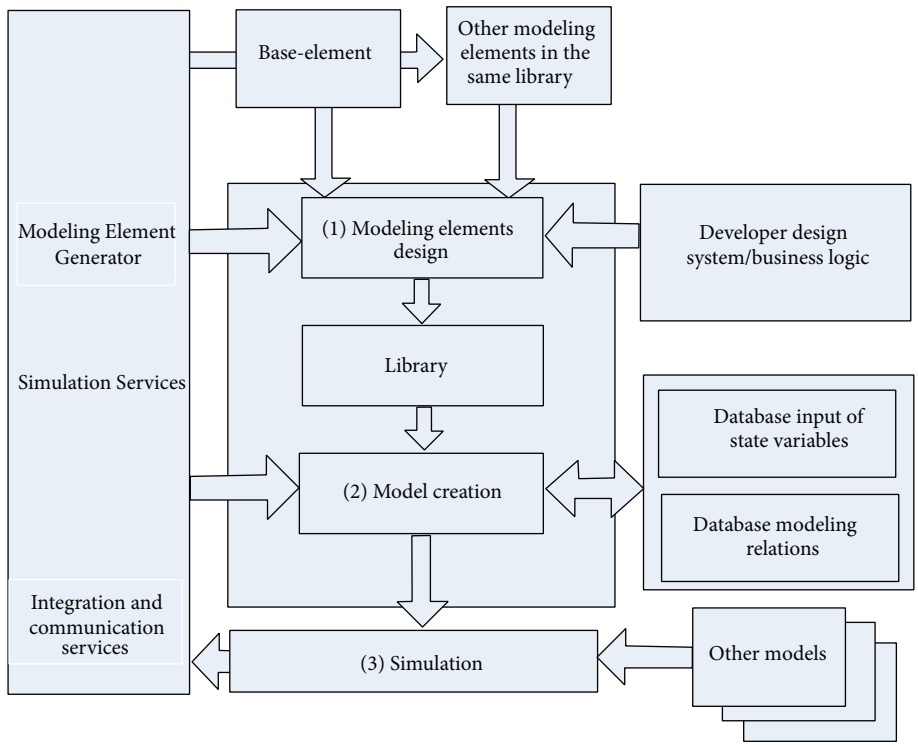

Figure 6: Simulation modeling process.

dynamic (change over time) and require simulation models to be built to study them in different time spans. The main advantage is the possibility to store the system data in external databases, update the database, and incorporate the updated databases in the model. It allows experimenting with different scenarios stored in databases. Modelers can store the different criteria and structures of these scenarios in external databases and then run the simulation model of these scenarios without the need to rebuild the model for each scenario. Modelers need only to import the database reflecting the new information, to run the model, and to store the outputs in a new database. Modelers can compare the outputs of these databases for analysis. Moussa [35] shows an application of the platform in developing a CPM stochastic model where the intent of the model is to study the impact of risks stored in an external database on the project schedule. The application also demonstrated the advantage of the distributed simulation in modeling a portfolio of projects that share a risk register but at the same time have their own local risk register.

4.7. Highlight on Validation and Verification. The simulation platform was used to build three libraries: (1) a simple discrete event library that can be used to model cyclic truck/batch plant operation, (2) a risk assessment library that reads and writes to a database of risk events and use of the critical path method (CPM) to assess project risks, and (3) a CPM risk assessment library to model distributed projects and measure the risks for a portfolio of projects. The risk assessment library was used in a graduate course work in the University of Calgary in the academic year 2012. Many tests were conducted on the simulation process to test the batch data processing and the automation of the modeling element generation and the modeling construction. More applications are planned to be designed to increase the scope of use and test the platform. Validation and verification as well as applications will be addressed in future papers.

5. Summary

This paper presented the criteria required in a simulation environment suitable for modeling construction projects. Also presented is a simulation platform that incorporates the identified criteria. The objective of the platform is to increase the appeal to simulation and enable modeling of large complex systems. The simulation methodology uses concepts from network modeling, Monte Carlo simulation, discrete event simulation, agent-based modeling, and object oriented programming.

The platform consists of three components: the modeling element generator that aims at reducing the coding/programming requirements; the simulation modeling environment that provides an easy-to-use graphical user interface resembling that of other Windows-based graphical interfaces; and the simulation services that allow integration with external data storage media and facilitation of modeling distributed systems. The platform offers a unified simulation environment in which models are programmed, built, and analyzed.

The appendix demonstrates different screens from the simulation platform that was developed to encompass the

criteria described in this paper. The purpose of the demonstration is to illustrate the criteria detailing the different user interfaces or the implementation of the platform. The appendix should help the reader visualize how the criteria are reflected in an end user program.

Appendix

A. Selected Simulation Platform Screens

This appendix shows 11 screenshots from the simulation platform developed as per the criteria described in this paper. The appendix does not provide all the available output/input screens; however, it shows miscellaneous screens to demonstrate the criteria described in this paper. Not all options and features of the simulation platform are demonstrated in this appendix; only those features related to the criteria described in this paper are presented.

A.1. Modeling Element Generator Screens. The left-hand side window of Figure 7 shows the nine element generator steps. When a developer selects any of the shown nine selections, a related input screen is displayed at the right-hand side. For example, selecting the "Element Name" displays a screen where the user can provide the name of the new element and select from a list of available elements that the new element can inherit from. Similarly the "Ports" option allows the developer to provide the ports/connection points information. By selecting "Ports," the user adds, deletes, or modifies a port. Figure 7 shows the input screen of the input properties screen where the developer can add/modify/delete the input properties, specify their type, and provide their default values.

Figure 8 shows the methods input window. The user can add a method (user-defined/user implemented methods) or select from the existing list of methods (system provided/system implemented methods) and write necessary code required. The windows allow users to add, modify, or delete methods.

The user can review the generated code by selecting "Source Code" (see Figure 7). Figure 9 shows an example of a compiled source code screen. The user draws the shape of the elements and adds pictures if needed using a drawing user interface similar to that provided by other drawing programs.

A.2. Simulation Modeling Environment. The simulation modeling environment allows the user to draw a model, input values to the input properties, run the simulation calculation, and retrieve the simulation results. Figure 10 shows the drawings screen after a model was generated.

Figure 11 shows an example of the property windows used to input values for an element's property.

Running a standalone simulation, the user selects Simulate Diagram as shown in Figure 12.

The screen in Figure 13 shows the simulation options menu screen. Users define the number of times they want to run the simulation calculation, the simulation time of each run (ticks) in discrete event simulation models, whether they want the results to be saved in a database, and whether they want to see the simulation progress during the calculation.

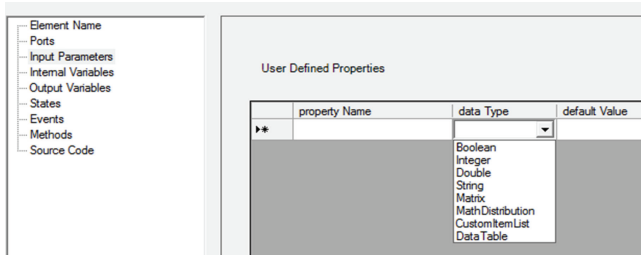

Figure 7: Properties definition screen.

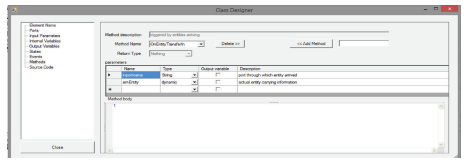

Figure 8: Methods properties definition screen.

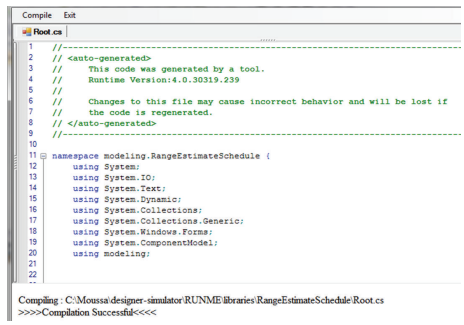

Figure 9: Source code review and compiling screen.

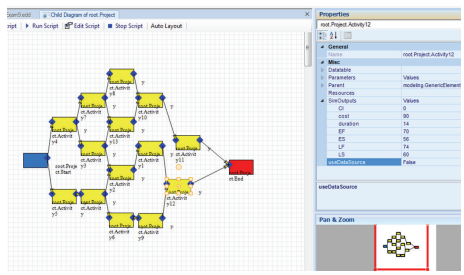

Figure 10: Simulation modeling environment drawing screen.

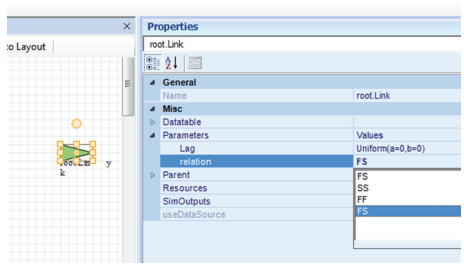

Figure 11: Example of inputting value to a property using a drop down menu.

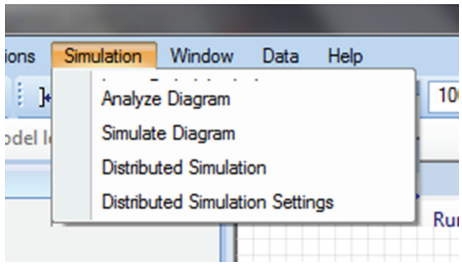

Figure 12: Simulation menu.

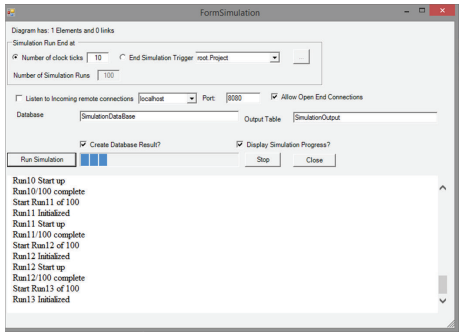

Figure 13: Simulation window.

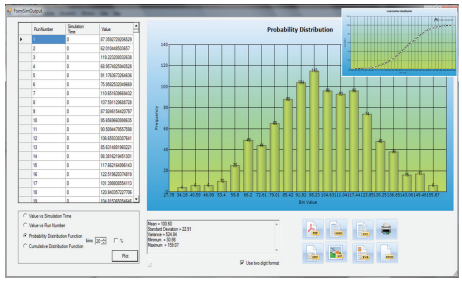

Figure 14: Simulation result of a property graph and data table output.

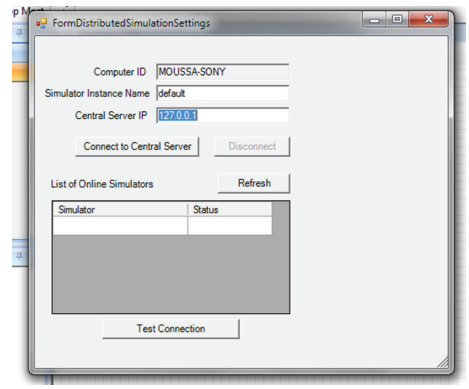

Figure 15: Distributed simulation settings.

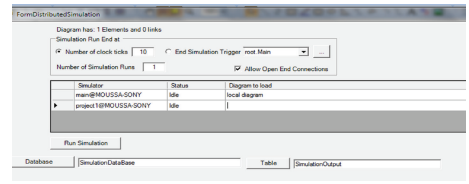

Figure 16: Distributed simulation settings selection.

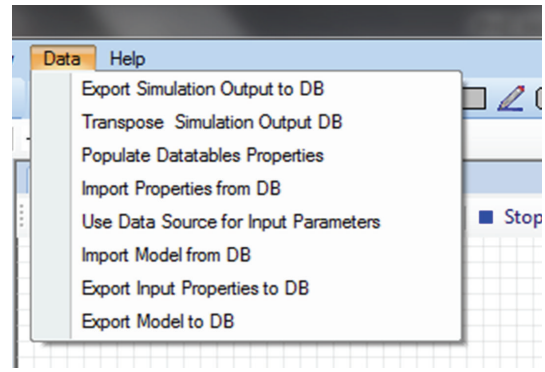

Figure 17: Distributed simulation settings selection.

When the simulation calculations are complete, users can retrieve the results (Figure 14) as a graph or a table and in a summary showing the average, minimum, maximum, and standard deviation of the values.

If the user is running a distributed simulation model, first the user should establish the distributed simulation settings (Figure 15) by providing the IP address of the computers involved in the session. The user then performs the simulation for the participating machines using the screen in Figure 16.

A.3. Simulation Services. Many of the simulation services have been integrated in the simulation modeling environment. Figure 17 shows the options available to integrate with an external database. Selecting an option from the list shown in the figure displays a screen that allows the user to map the modeling fields to a database. The options allow modelers to export and import the model data to/from a database.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. Hajjar and S. M. AbouRizk, "Unified modeling methodology for construction simulation," *Journal of Construction Engineering and Management*, vol. 128, no. 2, pp. 174–185, 2002.

[2] S. AbouRizk, "Role of simulation in construction engineering and management," *Journal of Construction Engineering and Management*, vol. 136, no. 10, pp. 1140–1153, 2010.

[3] M. A. Centeno and M. Carrillo, "Challenges of introducing simulation as a decision making tool," in *Proceedings of the 2011 Winter Simulation Conference*, pp. 17–21, Arlington, Va, USA, 2011.

[4] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: review and development recommendations," *Simulation*, vol. 82, no. 9, pp. 609–623, 2006.

[5] Project Management Institute (PMI), *A Guide to the Project Management Body of Knowledge (PMBOK' Guide)*, Project Management Institute (PMI), 2008.

[6] S. Abourizk, D. Halpin, Y. Mohamed, and U. Hermann, "Research in modeling and simulation for improving construction engineering operations," *Journal of Construction Engineering and Management*, vol. 137, no. 10, pp. 843–852, 2011.

[7] D. D. Meredith, K. W. Wong, R. W. Woodhead, and R. H. Wortman, *Design and Planning of Engineering Systems*, Prentice-hall, New Jersey, NJ, USA, 1973.

[8] A. A. B. Pritsker and J. J. O'Reilly, *Simulation with Visual SLAM and AweSim*, John Wiley & Sons, New York, NY, USA, 1999.

[9] S. Ghosh and T. S. Lee, *Modelling and Asynchronous Distributed Simulation: Analyzing Complex Systems*, IEEE Press, New York, NY, USA, 2000.

[10] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*, John Wiley & Sons, New York, NY, USA, 2000.

[11] D. A. Sadowski and M. R. Grabau, "Tips for successful practice of simulation," in *Proceedings of the 2000 Winter Simulation Conference*, vol. 1, pp. 26–31, Orlando, Fla, USA, December 2000.

[12] T. Andel, "Get it right before it's real," *Material Handling Engineering*, vol. 54, no. 7, pp. 55–61, 1999.

[13] P. A. Jensen and J. F. Bard, *Operations Research: Models and Methods*, John Wiley & Sons, Hoboken, NJ, USA, 2003.

[14] D. Hajjar and S. M. AbouRizk, "Application framework for development of simulation tools," *Journal of Computing in Civil Engineering*, vol. 14, no. 3, pp. 160–167, 2000.

[15] R. E. Shannon, "Introduction to simulation languages," in *Proceedings of the 9th Conference on Winter Simulation*, pp. 14–20, Gaithersburg, Md, USA, 1977.

[16] M. A. Hossain and D. K. H. Chua, "Autogeneration of simulation network of the design process," *Journal of Computing in Civil Engineering*, vol. 24, no. 5, pp. 452–461, 2010.

[17] D. K. H. Chua and G. M. Li, "RISim: resource-interacted simulation modeling in construction," *Journal of Construction Engineering and Management*, vol. 128, no. 3, pp. 195–202, 2002.

[18] Y. Yuan, C. A. Dogan, and G. L. Viegelahn, "A flexible simulation model generator," *Computers and Industrial Engineering*, vol. 24, no. 2, pp. 165–175, 1993.

[19] D. W. Halpin, "Cyclone: method for modeling of job site processes," *Journal of the Construction Division. American Scoitey of Civil Engineers*, vol. 103, no. 3, pp. 489–499, 1977.

[20] D. W. Halpin and L. Martinez, "Real world applications of construction process simulation," in *Proceedings of the 1999 Winter Simulation Conference*, vol. 2, pp. 956–962, Phoenix, Ariz, USA, December 1999.

[21] B. C. Paulson Jr., "Interactive graphics for simulating construction operations," *Journal of the Construction Division*, vol. 104, no. 1, pp. 69–76, 1978.

[22] D. Y. Chang and R. I. Carr, "RESQUE: a resource oriented simulation system for multiple resource constrained processes," in *Proceedings of the 1987 Project Management Institute Seminar & Symposium*, Milwaukee, Wis, USA, 1987.

[23] P. G. Ioannou, *UM-CYCLONE User's Guide*, Department of Civil Engineering, The University of Michigan, Ann Arbor, Mich, USA, 1989.

[24] J. C. Martinez and P. G. Ioannou, "General purpose simulation with stroboscope," in *Proceedings of the 1994 Winter Simulation Conference*, pp. 1159–1166, December 1994.

[25] R. Huang, A. M. Grigoriadis, and D. W. Halpin, "Simulation of cable-stayed bridges using DISCO," in *Proceedings of the 1994 Winter Simulation Conference*, pp. 1130–1136, December 1994.

[26] D. Hajjar, *A unified modelling methodology for simulation-based planning of construction projects [Doctoral dissertation]*, Department of Civil and Environmental Engineering: University of Alberta, Edmonton, Canada, 1999.

[27] M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2002.

[28] F. Klügl and A. L. C. Bazzan, "Agent-based modeling and simulation," *The AI Magazine*, vol. 33, no. 3, pp. 29–40, 2012.

[29] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proceedings of the 37th Conference on Winter Simulation*, pp. 2–15, Orlando, Fla, USA, December 2005.

[30] M. A. Niazi and A. Hussain, *Cognitive Agent-Based Computing-I. A Unified Framework for Modeling Complex Adaptive Systems Using Agent-Based & Complex Network-Based Methods*, Springer, London, UK, 2013.

[31] V. Hlupic, "Discrete-event simulation software: what the users want," *Simulation*, vol. 73, no. 6, pp. 362–370, 1999.

[32] Y. Mohamed and S. M. AbouRizk, "A hybrid approach for developing special purpose simulation tools," *Canadian Journal of Civil Engineering*, vol. 33, no. 12, pp. 1505–1515, 2006.

[33] F. Kluegl, M. Fehler, and R. Herrler, "About the role of the environment in multi-agent simulations," in *Environments for Multi-Agent Systems*, D. Weyns, H. V. V. Parunak, and F. Michel, Eds., vol. 3374 of *Lecture Notes in Computer Science*, pp. 127–149, Springer, Berlin, Germany, 2005.

[34] M. Moussa, J. Ruwanpura, and G. Jergeas, "CTAN for risk assessments using multilevel stochastic networks," *Journal of Construction Engineering and Management*, vol. 133, no. 1, pp. 96–101, 2007.

[35] M. Moussa, *Unified simulation methodology and project risk assessment framework [Doctoral dissertation]*, Faculty of Civil Engineering. Schulich School of Engineering. University of Calgary, Alberta, Canada, 2013.