www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

ENABLING REAL TIME SIMULATION OF ARCHITECTURE, ENGINEERING, CONSTRUCTION, AND FACILITY MANAGEMENT (AEC/FM) SYSTEMS: A REVIEW OF FORMALISM, MODEL ARCHITECTURE, AND DATA REPRESENTATION

SUBMITTED: December 2013
REVISED: October 2014
PUBLISHED: January 2015 at http://www.itcon.org/2015/1
EDITOR: Ruikar K.

Amir H. Behzadan, Assistant Professor
Department of Civil, Environmental, and Construction Engineering, University of Central Florida
Email: amir.behzadan@ucf.edu

Carol C. Menassa, Assistant Professor and John L. Tishman Faculty Scholar
Department of Civil and Environmental Engineering, University of Michigan
Email: menassa@umich.edu

Anu R. Pradhan, Assistant Professor
Department of Civil, Architectural, and Environmental Engineering, Drexel University
Email: pradhan@drexel.edu

SUMMARY: *Within the past few years, simulation and modeling have been adopted by the Architecture, Engineering, Construction, and Facility Management (AEC/FM) industry because it provides unique opportunities to transform traditional human-centered decision-making in design, planning, execution, and maintenance practices into simulation-centered decision-making. This allows a decision-maker to understand system behavior, and predict future events without having to test scenarios and interfere with the current operations of an existing system. However, a major challenge in transitioning from human-centered decision-making to simulation-centered decision-making is to establish methods and guidelines that facilitate seamless integration of field data into project planning, monitoring and control, and operation. This integration of field data into a simulation model allows the dynamics and evolving nature of events that may occur in a real system to be integrated into the simulation for effective decision-making. The goal of this paper is to conduct a vigorous review and investigate the underlying challenges associated with aspects such as modular design, data interfaces and translation, and model development and validation. In particular, this paper: 1) presents Discrete Event Simulation Specification (DEVS) as a simulation formalism that facilitates decision-making across a single operation (e.g. earthmoving); 2) reviews High Level Architecture (HLA) as a distributed simulation platform that allows for the coupling of several standalone simulation models and data sources to improve decision-making across complex systems; and 3) describes the hierarchy of multimodal data sources for simulation and different formalisms (i.e. model representations) that facilitate storage and querying of real time data to support simulation and distributed simulation models. The materials presented in this paper are the latest findings of an ongoing collaborative project being conducted by the Simulation Taskforce of the Visualization, Information Modeling, and Simulation (VIMS) Committee of the American Society of Civil Engineers (ASCE), and aim to lay the foundation for future research and implementation efforts in AEC/FM domain-specific real-time simulation systems. In the long term, work in this area will enable researchers to address some of the most fundamental problems in AEC/FM simulation modeling.*

KEYWORDS: *Real time systems, process simulation, formalism, knowledge modeling, decision-making, high level architecture; interoperability.*

REFERENCE: *Amir H. Behzadan, Carol C. Menassa, Anu R. Predhan (2015). Enabling real time simulation of architecture, engineering, construction, and facility management (AEC/FM) systems: a review of formalism, model architecture, and data representation. Journal of Information Technology in Construction (ITcon), Vol. 20, pg. 1-23, http://www.itcon.org/2015/1*

1. INTRODUCTION

Traditionally, project engineers and planners in Architecture, Engineering, Construction, and Facility Management (AEC/FM) have used techniques, such as the Critical Path Method (CPM) and Earned Value Analysis (EVA), to study work progress and identify deviations from project time-cost plans. Most existing scheduling and decision-making approaches however, are unable to model the repetition of operations and resource dynamics (i.e. interactions), and have limited capabilities to support sensitivity (e.g. what-if) analysis. Meanwhile, in most AEC/FM projects (e.g. highway construction, transportation infrastructure, and building systems), activities and events do not take place in fully controlled environments and the presence of ambient factors (e.g. delays, repairs, adverse weather, breakdowns, occupant preferences, and unforeseen conditions) can intensify uncertainty. Within the past several years, the AEC/FM domain has witnessed a rapid pace of information technology (IT) advancement and integration, which has provided unique opportunities to transform traditional human-centered decision-making in design, planning, execution, and maintenance practices into simulation-centered decision-making. Some notable examples of such new approaches include building information modeling (BIM), remote sensing, data mining and fusion, simulation and visualization, and process automation. A review of recent literature reveals several efforts both in academia and industry to employ one or more of these techniques in order to improve the current state of research and practice (Golparvar-Fard and Peña-Mora 2007, Jang and Skibniewski 2009, Abourizk 2010, Akhavian and Behzadan 2013, Bosché et al. 2013). Among these approaches, simulation and modeling offers unique opportunities to understand the system behavior and predict future events without having to test scenarios on a real system or interfere with the current operations of an existing system. The aim of this paper is to investigate the state-of-the-art in AEC/FM simulation modeling, its limitations, and potential future directions.

Most AEC/FM systems, by definition, consist of repetitive operations that involve resources interacting at different levels to transform a set of inputs into a set of outputs within the constraints of time, budget, and required quality. Despite this structured definition, every project is unique as it takes place in an environment that can completely be different from another project. In other words, while the internal mechanisms and logics governing a class of AEC/FM systems (e.g. earthmoving, highway construction, building) may be similar across several projects, the external mechanisms (e.g. location, ground conditions, type of machinery, weather, labor skills, occupants) surrounding each project are exclusive to that project alone. This is in contrast with many manufacturing and industrial systems where internal and external mechanisms are controlled and the effect of unexpected (or undesirable) conditions (i.e. system noise) is kept to a minimum. Discrete event simulation (DES) has thus evolved as a method that allows decision-makers to closely represent an AEC/FM system and to model the effects of internal and external uncertainties on project planning, execution, operation, and maintenance.

From a pure modeling and simulation (M&S) perspective, a system is defined as a collection of entities (or components) that act and interact together toward the accomplishment of one or more logical ends (Maria 1997). Within this context, there are two types of systems: discrete and continuous. In a discrete system, state variables change instantaneously at specific points in time (Cassandras and Lafortune 1999). In building construction, for example, tower cranes operate on a discrete basis since the state variable describing tower crane status (busy or idle) changes only when a new payload arrives at the lifting station. In a continuous system, however, state variables change continuously with respect to time. For instance, in a built facility (e.g. office building or factory plant), if the state variables are defined as instantaneous indoor temperature and humidity, the corresponding system is considered continuous since these variables continuously adjust (not at predefined points in time) to maintain indoor temperature and comfort conditions for building occupants. DES is a paradigm used to model the former case where state changes occur at discrete points in time. A DES model can be either deterministic or stochastic in nature where in the latter changes to a new state are governed by probabilistic conditions.

By simulating the behavior (i.e. state changes) of a system over time, the DES is sought to reproduce the process of how project entities (i.e. materials, equipment, and people) interact with each other, and what output may result from this interaction under different scenarios. Since the late 1970's, several researchers have attempted to introduce DES tools and applications within the AEC/FM domain. Examples include CYCLONE (Halpin and Woodhead 1976), RESQUE (Chang 1986), UM-CYCLONE (Ioannou 1989), Micro CYCLONE (Halpin 1990), COOPS (Liu 1991), CIPROS (Odeh 1992), Stroboscope (Martinez 1996), SIMPHONY (Hajjar and AbouRizk 1999), ABC (Shi 2000), and EZStrobe (Martinez 2001). Despite the advantages of DES in modeling AEC/FM systems, this simulation paradigm still has key limitations that have prevented it from full accreditation and

widespread adoption by the construction industry for large-scale decision-making. In a recent report published by the Visualization, Information Modeling, and Simulation (VIMS) Committee of the American Society of Civil Engineers (ASCE), several grand challenges in simulation for the AEC/FM industry were identified. Among those listed, the existing simulation frameworks are unable to: 1) create realistic operations-level models that can properly model the dynamics of project entities, and 2) adapt to changes in real time (Lee et al. 2013). The common denominator of these two limitations is the lack of systematic strategies that enable a simulation model to communicate with the corresponding real world engineering system through collecting process data and extracting meaningful computer-interpretable knowledge that can help the model update its properties, attributes, and design variables as the real system evolves.

Currently, almost all construction DES models created using existing methodologies are static and built upon rigid structures. For instance, the routing mechanism that guides different entities throughout the system has to be explicitly hard-coded in a model. Similarly, queue disciplines need to be prescribed ahead of time. In reality, the dynamic nature of a real engineering system may require new routing mechanisms to be used or queue entities to be served in a different order depending on new information received from ongoing operations (Akhavian and Behzadan 2014). If data describing such deviations cannot be captured and represented in the simulation model, it is very unlikely that the model can adequately evolve over time or be useful for decision-making beyond its intended application. Given the relatively high cost and unique set of skills needed to create simulation models, this limitation can make the development of simulation models quite uneconomical as they may not be usable beyond the early planning stages of a project when data from the real engineering system is scarce and thus, making engineering assumptions about model parameters can be better justified (Abourizk 2010). The resulting models will be rarely used during the execution phase of the project to help make more informed decisions based on updated data that becomes available over time which better explains the actual system performance and highlight discrepancies from the simulation results. These discrepancies can be investigated to determine issues with the original assumptions, and also to update the model to better predict future performance.

2. RESEARCH OBJECTIVES

Considering the current state of knowledge in AEC/FM simulation, it is perceivable that a major challenge in transitioning from human-centered decision-making to simulation-centered decision-making is to establish methods and guidelines that facilitate the seamless integration of field data into project planning, monitoring and control, and operation. As a result, any data-driven simulation tool designed for this domain must take into account the dynamics and evolving nature of events that may occur in the real system.

This paper reports on the latest collaborative efforts of the Simulation Taskforce of the ASCE VIMS Committee to lay the foundation for future research and implementation of domain-specific real-time simulation systems in AEC/FM. Section 3 provides an overview and definition of real-time simulation. A state-of-the-art review and the latest advancements in the three main areas that facilitate the simulation of AEC/FM systems in real-time are presented in Sections 4, 5, and 6. In particular, Section 4 presents Discrete Event Simulation Specification (DEVS) as a simulation formalism that facilitates decision-making across a single operation (e.g., earthmoving), Section 5 reviews High Level Architecture (HLA) as a distributed simulation platform that allows for the coupling of several standalone simulation models and data sources to improve decision-making across complex systems, and finally Section 6 described the hierarchy of multimodal data sources for simulation and different formalisms (i.e., model representations) that facilitate storage and querying of real time data to support simulation and distributed simulation models. Throughout this paper, different examples from three major areas within AEC/FM are used to facilitate the technical discussions and to demonstrate the applicability of the presented materials about formalism, model architecture, and data representation in a variety of AEC/FM applications. In particular, the examples in this paper are selected from construction (earthmoving), FM (building energy), and engineering (transportation).

A thorough understanding of model formalism, model architecture, and data representation is of utmost importance to the development of domain specific real-time simulation systems for AEC/FM applications. In developing such models, the decision-maker needs to understand how entities function and interact (goal of formalism), how model components are assembled to create complex simulations (goal of model architecture), and how data is interpreted and transferred between different parts of a simulation model (goal of data representation). The objective of this paper is therefore to investigate the underlying challenges associated with

modular design, data interfaces and translation, and simulation model development and validation. Ultimately, the discussions presented in this paper are sought to lay the foundation for future research and implementation efforts in AEC/FM domain-specific real-time simulation systems.

In order to present the most relevant and inclusive information in this review paper, this paper follows a cohesive approach. In particular, the authors adopted a methodology presented in Noguchi (2006) in which papers are grouped into four distinct categories according to their objective:

- *Status quo review*: presentation of the most current research for a given topic or field of research.
- *History review*: development of a field of research over time.
- *Issue review*: Investigation of an issue (i.e. point of disagreement or a question) in a specific field of research.
- *Theory/model review*: Introduction of a new theory or model in a specific field of research.

While this review paper has components from almost all four categories, it mostly fits with the definition of a "theory/model" review since it attempts to introduce researchers and practitioners to the application of new topics such as DEVS and HLA in AEC/FM simulation modeling. The main aspects of the review method adopted in this paper are as follows:

- *Data sources*: The most common engineering research databases such as ASCE Library, Elsevier, IEEE, Taylor and Francis, and Wiley Online were considered.
- *Search terms and strategies*: Main search keywords that were used included "simulation", "real-time", "DEVS", "HLA", "knowledge modeling", "decision-making", and "formalism". However, the authors also relied on their prior work in these areas and included work (in addition to search results) from prominent researchers on a case-by-case basis.
- *Selection criteria*: Since this review paper is intended to focus on AEC/FM domains, articles that had little to offer in terms of application areas in AEC/FM domains were excluded. Also, with a few exceptions (e.g. where classic texts such as well-known dissertations or books had to be cited), articles published before 2000 were excluded. The rationale behind this was that the objective of this work is to introduce the "state-of-the-art" and therefore, more recent publications can better serve the purpose of this study.

3. REAL-TIME SIMULATION SYSTEMS

Shaw (2001) defines a real-time system as a system that "*monitors, responds to, or controls an external environment. This environment is connected to the real-time system through sensors, actuators, and other input/output (I/O) interfaces*". Due to increasing complexity and multidisciplinary nature of many engineering phenomena including AEC/FM projects, such systems often include subcomponents that are distributed over a much larger physical space, while still coordinating and cooperating together to satisfy an objective function. Despite some differences, one key requirement of all real-time systems is timeliness, a property defined as the ability of the system to respond correctly (reliably) to inputs within acceptable time intervals (Laplante 1997). In addition to reliability, real-time systems also benefit from a concurrent event-driven nature, and are characterized by their continuous interaction with an external environment. Therefore, they are sometimes referred to as reactive systems. The combination of temporal requirements (timeliness), limited resources (computing power), concurrent environmental entities (complexity), high reliability requirements, and distributed processing has led to real-time systems being recognized as a distinct discipline with its own body of knowledge and theoretical foundation (Hu 2004).

4. DISCRETE EVENT SIMULATION SPECIFICATION (DEVS) FORMALISM

Formalism refers to the mathematical structure of different building blocks of a simulation model (Antoine-Santoni 2007). Within the context of DES, the discrete event formalism is sought to express changes of variable values (i.e. state variables) at time segments that are piecewise constant. Thus, an event is essentially a change in a variable value that occurs instantaneously. In other words, the goal of formalism is to define how to generate new values for state variables and to identify the time segments these new values take effect. A widely used simulation formalism is the discrete event simulation specification (DEVS) which enables modelers to mathematically define the structures of individual components involved in a single DES model. It also allows individual models to be contextually linked to facilitate the prospect of simulating more complex systems

(Zeigler and Sankait 1993). A variety of DEVS modeling and simulation tools have been implemented in the past several years for domains other than AEC/FM (Li et al. 2011). In DEVS formalism, the time intervals between event occurrences are variable (in contrast to discrete time where the time step is generally a constant number). Using DEVS formalism, one must specify the basic models from which larger ones are built, and how these models are connected together in a hierarchical manner (Posse et al. 2003). The definition of a model in DEVS is slightly different from that defined by the common DES terminology. A DEVS model is a unit that possesses input and output ports through which all interactions with the environment are mediated. In the discrete event case, when external events (from the external environment) are received on input ports, the model description must determine how to respond to them. Also, internal events arising within the model can change its state, as well as resulting events on the output ports. Using DEVS, the behavior, structure, and timeliness of a real-time system can be effectively modeled, evaluated, and ultimately implemented by different simulation methods (Hu 2004). The behavior aspects of a system can be specified by DEVS atomic models. The structure aspects of the system can be specified by DEVS coupled models which allow hierarchical composition of models through coupling input ports and output ports between multiple models. Also, a DEVS model can explicitly specify time using the time-advance function. This makes it possible to model and evaluate real-time requirements in a systematic way. DEVS also has the capability to model dynamic reconfigurations of a real-time system which allows for self-adaptation of a simulation model to the evolving nature of the external environment.

4.1. Basic (Atomic) DEVS Model

An atomic model is an irreducible component in DEVS framework that implements the behavior of a component (Zeigler et al. 1976). It executes the state-machine and interacts with other components using its predefined input and output ports. A basic DEVS atomic model is expressed using Equation (1),

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta > \quad \text{Eq. (1)}$$

In this Equation,

X: set of external input events

S: set of sequential states (including phases and variables)

Y: set of outputs

$\delta_{int}: S \rightarrow S$: internal transition function

$\delta_{ext}: Q \times X_b \rightarrow S$: external transition function

$\delta_{con}: Q \times X_b \rightarrow S$: confluent transition function

$Q = \{ (s, e) \mid s \in S, 0 \leq e \leq ta(s) \}$: set of total states where e represents the elapsed time since last state transition

X_b: set of bags over elements in X

$\lambda: S \rightarrow Y_b$: output function generating external events at the output

$ta: R \rightarrow R^{+}_{0,\infty}$: time-advance function

The interpretation of a DEVS atomic model and relationships between model components are illustrated in Figure 1.

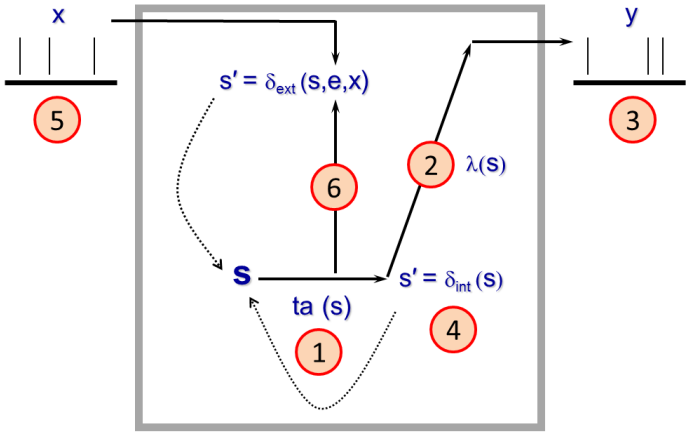

Fig. 1 DEVS atomic model semantics.

This model contains information necessary to explicitly describe the following (Hu 2004):

- The set of input ports through which external events are received.
- The set of output ports through which external events are sent.
- The set of state variables and parameters – two state variables are usually present: S (phase) and σ (sigma). The latter determines how long the system stays in the current phase in the absence of external events.
- The time-advance function which controls the timing of internal transitions.
- The internal transition function which specifies to which next state the system will transit after the time given by the time-advance function has elapsed.
- The external transition function which specifies how the system changes state when an input is received. The effect is to place the system in a new S (phase) and σ (sigma), thus scheduling it for a next internal transition. The next state is computed based on the present state, the input port, value of the external event, and the time that has elapsed in the current state.
- The confluent transition function which is applied when an input is received at the same time that an internal transition is to occur. By default, the internal transition function is applied before applying the external transition function to the resulting state.
- The output function which generates an external output right before an internal transition takes place.

Within AEC/FM and in particular, in many construction systems, entities have to go through different processes. For instance, steel sections arriving on a jobsite on flatbed trucks must be lifted and erected by a crane at the installation location. As a result, modeling such interactions constitute an important part of almost any construction simulation model. The DEVS formalization of a single processor (e.g. crane) that receives different jobs (e.g. lifting schedule) is described by Equation (2). Each job has an associated processing time (either deterministic or probabilistic). Once that time is over, event "ready" is produced.

$$PM = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \qquad \text{Eq. (2)}$$

In this Equation,

$$X = NJ$$

$$S = NJ \cup [\emptyset] \times R^{+}$$

$$Y = PJ$$

$$NJ = \{job_1, job_2, job_3, \dots, job_n\}$$

$$PJ = \{P(job_i) \mid job_i \in S, 1 \leq i \leq n\}$$

$$\delta_{int}(nj, \sigma_{nj}) = (\emptyset, \infty)$$

$$\delta_{ext}(nj, \sigma_{nj}, e, x) = \begin{cases} (x, \sigma_x) & S = \emptyset \\ (nj, \sigma_{nj} - e) & otherwise \end{cases}$$

$$\lambda(nj, \sigma_{nj}) = P(nj)$$

$$ta(nj, \sigma_{nj}) = \sigma_{nj}$$

According to the external transition function δ_{ext} of this atomic model, once a new job (modeled as an external event x) arrives at the processor (i.e. crane's lifting station), the processor state is evaluated. If the processor is already busy with another job (nj), this new event is ignored. In reality, this is rarely the case as newly arrived jobs that cannot be immediately processed are often stored in the order they are received for future processing. To model this behavior, queues are added to the simulation as stand-alone DEVS components that precede high-demand (and/or low-capacity) processors (e.g. crane). As an example, Figure 2 shows a road construction project where a queue is used in the real-world operations to hold multiple haulers to be served by a single excavator, one at a time. The arrival of a hauler in this queue can be described by a set of triples (id, α, p) where id represents the hauler identifier (e.g. license plate), α is a placeholder for time (e.g. how long it takes to load that hauler), and p is the position of the hauler in the queue. Also, assume that the queue has infinite capacity and no exit delay. In other words, as soon as it receives a "ready" signal from the excavator, it releases a hauler through an output event. The DEVS atomic model of this queue can be described by Equation (3),

$$QM = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \qquad \text{Eq. (3)}$$

In this Equation,

$$X = H \cup \{'ready'\}$$

$$S = H \cup [\emptyset] \times \{1\}$$

$$Y = H$$

$$H = \{hauler_1, hauler_2, hauler_3, \dots, hauler_n\}$$

$$\delta_{int}(h, \sigma) = (h, \infty)$$

$$\delta_{ext}(h, \sigma, e, x) = \begin{cases} (x, \infty) & x \in H \\ (S.first(), 0) & x = \{'ready'\} \end{cases}$$

$$\lambda(h, \sigma) = h$$

$$ta(h, \sigma) = \sigma$$

Fig. 2 In a road projects, multiple clients (haulers) form a queue in front of a single processor (excavator).

According to this atomic model, the queue receives two different input types: haulers, and a "ready" signal. Haulers can arrive according to a deterministic or probabilistic pattern. The "ready" signal is transmitted from the excavator as soon as the load area is vacant (i.e. the hauler in service has just left the load area). The output of the queue will be a hauler which is sent to the load area to be served by the excavator. Possible states of the queue include haulers or "empty". When there is no new hauler arriving into the queue, the internal transition function simply replaces the current time dimension (i.e. 1) with ∞ for all existing haulers in the queue. Consequently, the time-advance function advances the internal clock to ∞. This guarantees that these haulers are not revisited unless they are about to be released. As soon as a new input is detected, the external transition function checks if it is of hauler type. If so, then the model state is changed to this new hauler. If the input is a "ready" signal, however, the state changes to the first hauler in the queue as determined by the $S.first()$ function while the time dimension gets a value of 0 to guarantee that the hauler in front of the queue is immeidately released (no exit delay). This change in state causes an immediate internal transition which in turn, replaces the time dimension with the value ∞. However, before this internal transition takes place, the λ function outputs the selected hauler. Note that the implementation of the $S.first()$ function is not included in the above model, but in essense, this function selects the hauler that is located in front of the queue according to the queue descipline. For example, in a First-In-First-Out (FIFO) queue, the hauler with the minimum position in the queue must be selected. Therfore, the implementation will be as follows,

$$S.first() = \{ h_i | h_i.p = \min(h_j.p), h_j \in S\} \quad \text{Eq. (4)}$$

Similiarly, for a Last-In-First-Out (LIFO) queue, the hauler with the maximum position in the queue must be selected. Therfore, the implementation will be as follows,

$$S.first() = \{ h_i | h_i.p = \max(h_j.p), h_j \in S\} \quad \text{Eq. (5)}$$

Definitions such as those described by Equations (4) and (5) are very powerful and flexible as they can be directly linked to hauler data collected from the field. In particular, the triples describing haulers can be created instantaneously using spatiotemporal data transmitted from the haulers. For instance, and depending on the situation, a hauler ID number (id) can be the identifier code of a Global Positioning System (GPS) device mounted on the hauler, or even the hauler's license plate number. Likewise, As soon as a hauler enters the queue, the maximum position number in the queue is incremented by 1 unit and the result is used to update the new hauler's position number (p) in the queue. Additionally, if the collected data from the queue in the real system indicates a change in discipline (e.g. from FIFO to LIFO) while the operation is ongoing, the implementation of the $S.first()$ function can be modified to reflect this change in the real system.

4.2. Coupled DEVS Model

At the operations level, multiple basic DEVS atomic models such as those described in Subsection 4.1 can be combined to form a coupled model that describes more complex operations involving larger numbers of entities or I/O ports. For instance, Figure 3 shows how atomic DEVS models of a queue and a single processor are connected to create a functional client-server system.

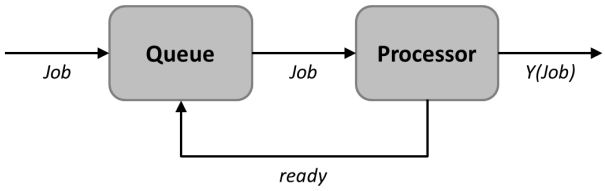

Fig. 3 Coupled model representation of a queue and a processor.

A coupled model specifies how to couple (connect) several component models to form a new model. The resulting model can itself be employed as a component in a larger coupled model, thus allowing modelers to create modular hierarchical simulation structures (Zeigler et al. 1976). A coupled DEVS model is defined by Equation (6),

$$DN = < X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} > \quad \text{Eq. (6)}$$

In this Equation,

X: set of external input events

Y: set of outputs

D: set of component names

M_i: atomic model of component i

I_i: set of influences for component i

$Z_{i,j}$: output translation function from component i to component j

Using this definition, a coupled model is capable of capturing the following information (Hu 2004):

- The set of component models.
- The influence(s) of each component model on the coupled model.
- The set of input ports through which external events are received.
- The set of output ports through which external events are sent.
- The coupling specification that consists of the external input coupling (EIC) that connects the input ports of the coupled model to one or more of the input ports of the component models, the external output coupling (EOC) that connects the output ports of the component models to one or more of the output ports of the coupled model, and the internal coupling (IC) that connects output ports of component models to input ports of other component models.

As an example, consider a building heating/cooling system. This real world complex system can be modeled as the coupling of six individual DEVS atomic models, as shown in Figure 4. In this Figure, the "outdoor climate" (OC) model has only one output (i.e. outdoor temperature). There is no input to this model. However, it is perceivable that outdoor temperatures are internally provided to the model, either through direct readings from the environment or by remotely connecting to an online database (e.g. weather service).

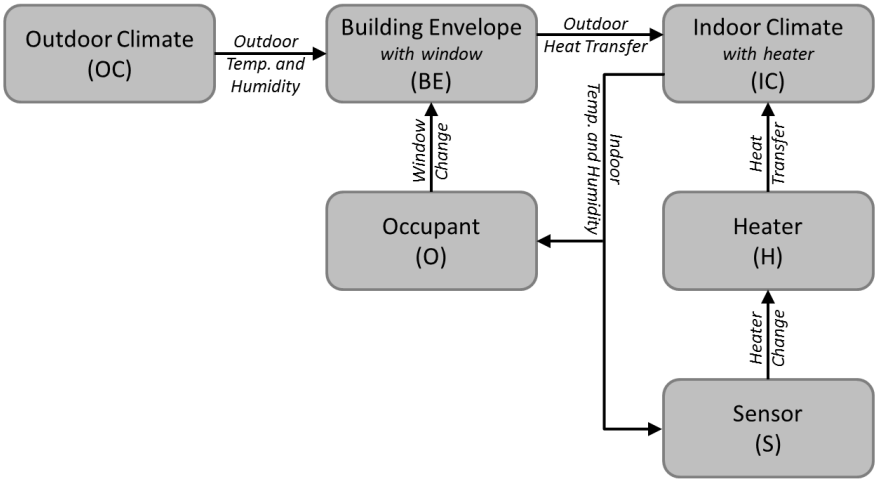

Fig. 4 Coupling of six DEVS atomic models in a building air control system.

This basic model is further enhanced by adding a "building envelope" (BE) model that encloses an indoor space, and an "indoor climate" (IC) model. Next, the heating system is added which can be activated at an indoor temperature of 70°F, and deactivated at 75°F. Using DEVS, this can be achieved by connecting a "sensor" (S) model and a "heater" (H) model. In this case, the output of the IC model becomes the input of the S model, and the output of the H model becomes a second input to the IC model. Finally, an occupant (O) model can be added to the coupled DEVS model. The occupant can (for instance, in response to indoor temperature variations) manipulate a window in the building envelope. This can be implemented in the coupled DEVS model by updating the BE model since opening or closing a window can change the properties of the building envelope.

4.3. Benefits of DEVS Formalism

To the authors' best knowledge, none of the existing AEC/FM simulators has the capability to tune a running simulation in response to changes in the external environment. Due to tight coupling between the network model and the simulation engine in such simulators, the capability to introduce changes in parameter values during execution is limited or non-existent. In all such platforms, design variables and system parameters are often hardcoded prior to simulating the network, and there is no support for changing parameters and component structures during simulation. This shortcoming among others has served as a major motivation behind the presented work. Despite recent advancements in data sensing and knowledge modeling, many construction simulation platforms still lack the ability to embrace such new technologies. As a result, the concept of real-time simulation which is quite commonly used in other domains such as manufacturing and computer science is still an alien concept in construction and several other AEC/FM domains.

In contrast, DEVS allows for dynamic simulation control since it separates model, experimental frame, and simulator. This modularity facilitates the development of a simulation framework supporting run-time simulation tuning. The motivation behind providing "real-time" intervention is to support a rapid feedback cycle that allows experimentation with network parameters and structures. This can result in an effective system configuration. Furthermore, such instantaneous observation and control enable important transient situations to be recognized and considered which enhances the quality and timeliness of project-related decision-making tasks. Since every DEVS model is built upon DEVS formalism that itself is based on mathematical systems theory, the behavior expressed through DEVS can be translated to any other formalism. With the separation of the model from the simulator implementation, the advantage is that it supports formalism interoperability, and enables distributed simulation by allowing model components to be executed on single or multiple distributed platforms.

As stated in Subsection 4.1, a DEVS atomic model has well defined states, state transition functions (triggered by external or internal events), time-advance function, and output function. From the design point of view, an atomic model can be viewed as a timed state machine. The transition from one state to another is triggered by external events (received from the input ports) or internal events (time outs generated internally). The model can generate output and send it out through its output ports. In the following Subsections, the DEVS simulation formalism of a quarry transport operation is presented, and its structure and behavior is discussed in detail.

4.3.1. Quarry Transport Operations: Model Description

In this conceptual scenario adopted from Wainer (2004), a rock quarry operation is modeled using DEVS formalism and both atomic and coupled models. Assume there is a large pile of ore that needs to be transported from a quarry location to a remote processing plant. For this purpose, multiple haulers and a shovel are used. The conceptual model structure is illustrated in Figure 5.

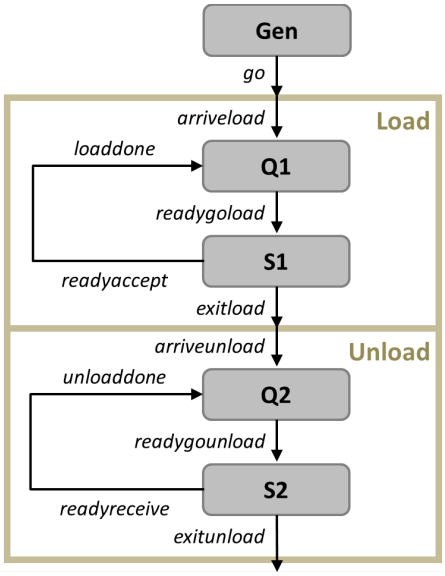

Fig. 5 Conceptual model structure of the quarry transport operation.

As shown in this Figure, a generator (*Gen*) will create haulers. Each hauler will undergo a load service first. Since the load service processor (*S1*) needs certain time to finish the current loading job, before entering into the load service, hauler will have to wait at a designated load service queue (*Q1*). Only when *S1* has finished the current job and sent a "done" feedback message to *Q1*, the first hauler lining up in *Q1* will enter into *S1*. After finishing the load service, haulers will head to the unload service at the processing plant location. Similar to the load service, before entering into the unload service processor (*S2*), each hauler will have to first go through an unload service queue (*Q2*). Only when the current instance of unload service completes and a "done" feedback message to *Q2*, the first hauler lining up in *Q2* can enter into *S2* to receive an unload service. When a hauler unloads its payload, it will leave the system. This quarry transport operations consists of three top-level models: *Gen*, *Load*, and *Unload*. The *Gen* model is an atomic model that generates haulers and assigns a *HaulerID* to each hauler. Both *Load* and *Unload* are coupled models. *Load* is composed of *Q1* (load service queue) and *S1* (load service processor), while *Unload* is made up of *Q2* (unload service queue) and *S2* (unload service processor). In total, there are five atomic models: *Gen*, *Q1*, *S1*, *Q2*, and *S2* that are grouped into three types: *Gen* is a typical generator, *Q1* and *Q2* are of queue model type, and *S1* and *S2* are processor models.

4.3.2. Quarry Transport Operations: Formalism of Atomic Models

The function of *Gen* is to produce haulers. It has no input port and one output port called "go". At the end of each time interval, *Gen* will produce a hauler and output *HaulerID* (= 1,2,…,n) through output port "go". *Gen* will be in "active" state all the time. The formalism of the *Gen* atomic model is as follows. Note that in this formalism, *Gen_time* is the time interval between two consecutive hauler generations.

$$GM = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$$

$$X = \{\ \}$$

$$Y = \{HaulerID\}$$

$$S = \{active\} \times \{HaulerID\}$$

$$\delta_{int}(active, Gen\_time) = \{active\}$$

$$\lambda(active, Gen\_time) = \{HaulerID\}$$

$$ta(active) = Gen\_time$$

Load, on the other hand, is a coupled model that provides load service. This model is used to load each hauler that is generated by *Gen*, and consists of two atomic models: *Loadqueue* (*Q1*) and *Loadservice* (*S1*). When the load service processor receives haulers produced by *Gen*, they must first enter into *Q1* and then enter into *S1* only when *S1* is in the "free" state. *Load* has one input port named *arriveload* and one output port named *exitload*. *Q1* has two input ports and one output port. It maintains and constantly updates the *HaulerID* list corresponding to haulers waiting to be served by *S1*. Once the simulation is launched, *Q1* will be in "passive" state and thus, ready to accept the first *HaulerID* from *Gen*. When it receives the first *HaulerID* through its input port *arriveload*, it will be activated after *Prep_time* and then sends the *HaulerID* to *S1* through output port *readygoload*. Immediately, after, *Q1* will return to "passive" state. Other incoming *HaulerID*s from *arriveload* will not activate *Q1* until it receives a feedback from its second input port *loaddone*, at which point *Q1* will output the first *HaulerID* in its waiting list through its output port *readygoload*. *S1* is used to execute load service. It has one input port called *enterload* and two output ports named *exitload* and *readyaccept*. First, *S1* will be initialized and ready to accept *HaulerID*s from *Q1*. When it receives a *HaulerID* from *Q1* through its input port *enterload*, it will become busy and take certain time to process the job. When the current hauler is loaded, *S1* will send the *HaulerID* out through its output port *exitload*, and immediately alerts *Q1* that the current task has been completed and therefore, a new job can be accepted through the output port *readyaccept*. At this point, the state of *S1* has changed from busy to passive.

Unload is another coupled model that provides unload service to haulers. It consists of two atomic models: *Unloadqueue* (*Q2*) and *Unloadservice* (*S2*). Haulers entering *Unload* are first held in *Q2* and then enter into *S2*. *Unload* has one input port named *arriveunload* and one output port called *exitunload*. *Q2* has two input ports *arriveunload* and *unloaddone*, as well as one output port *readygounload*. Its state transition and job process are

similar to those described above for *Q1*. *S2* has one input port *enterunload* and two output ports *exitunload* and *readyreceive*. Its state transition and job process are similar to those described above for *S1*. The formalism of the *Q1* and *Q2* atomic models are as follows,

$GQ =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$

$//state\ variables$

$Time\ sigma;\ //\ time\ variable$

$ElementList\ elements;\ //\ list\ used\ to\ store\ waiting\ HaulerIDs$

$Int\ pid;\ //\ HaulerID$

$Int\ finish;\ //\ default\ is\ 0.\ When\ the\ service\ done\ it\ changes\ to\ 1\ and\ message\ is\ sent$

$//initialization$

$sigma = Prep\_time;\ //\ when\ the\ queue\ is\ ready\ to\ accept\ the\ next\ hauler$

$Elements.empty()\ //\ initialize\ the\ queue\ as\ empty$

$finish = 0;\ //\ serves\ as\ a\ flag$

$//formal\ specification$

$X = \{arriveload, loaddone\}$

$Y = \{readygoload\}$

$S = \{HaulerID\} \times \{"passive", "active"\}$

$\delta_{int}(active, elements) = \{passive, elements\}$

$\delta_{ext}(arriveload, loaddone, elements, finish, pid)$

{

$if\ input\ is\ received\ through\ arriveload\ //\ new\ hauler\ has\ arrived$

{

$elements.push\_back(msg.value(\ ));\ //\ add\ new\ hauler\ to\ the\ elements\ (FIFO)$

$if\ elements.first(\ )\ AND\ finish == 0\ //\ for\ the\ very\ first\ hauler$

$\{hold(active, Prep\_time);\ //\ process\ the\ hauler\ \}$

$if\ elements.count(\ )\ AND\ finish == 1\ //\ for\ all\ haulers\ after\ the\ first\ one$

$\{hold(active, Prep\_time);\ //\ process\ the\ hauler\ \}$

}

$if\ input\ is\ received\ through\ loaddone\ //\ server\ is\ ready$

{

$if\ elements.count(\ )\ //\ elements\ is\ not\ empty$

$\{hold(active, Prep\_time);\ //\ process\ the\ hauler\ \}$

$if\ !elements.count(\ \ )\ \ //\ elements\ is\ empty$

$\{finish = 1;\}$

$\}$

$\}$

$\lambda(elements, readygoload)$

$\{$

$send(elements.front(\ \ ));\ //\ send\ the\ front\ hauler\ out$

$elements.pop\_front(\ \ );\ //\ move\ forward\ other\ haulers\ in\ the\ queue$

$if\ finish == 1$

$\{finish = 1;\}$

$\}$

$ta("active") = t(output)$

$ta("passive") = t(next\ done\ message) - t(last\ state\ change\ from\ active\ to\ passive)$

The formalism of the *S1* and *S2* atomic models are as follows. In each server, when the task at hand is completed, the state changes from busy to passive.

$GS =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$

$//state\ variables$

$Int\ pid;\ //\ HaulerID$

$//formal\ specification$

$X = \{HaulerID\}$

$Y = \{HaulerID\}$

$S = \{HaulerID\} \times \{"passive", "busy"\}$

$\delta_{ext}(HaulerID) = ("busy", HaulerID)$

$\lambda("busy", HaulerID) = HaulerID$

$ta("busy", HaulerID) = HaulerID.serv\_time\ //\ time\ needed\ to\ service\ a\ hauler$

$ta("passive") = \infty$

4.3.3. Quarry Transport Operations: Formalism of Coupled Models

As discussed in Subsection 4.3.2, *Load* and *Unload*, are both coupled models that provide load and unload services, respectively. Recalling Equation (6), the formalism for these two coupled models can be written as follows,

$LM =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$

$X = \{HaulerID\}$

$Y = \{HaulerID\}$

$D = \{Q1, S1\}$

$M = \{QM, SM\}$

$I(Q1) = \{S1\}$

$I(S1) = \{Q1, S1\}$

$Z(Q1) = \{S1\}$

$Z(S1) = \{Q1\}$

$Z(S1) = \{S1\}$

$UM =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$

$X = \{HaulerID\}$

$Y = \{HaulerID\}$

$D = \{Q2, S2\}$

$M = \{QM, SM\}$

$I(Q2) = \{S2\}$

$I(S2) = \{Q2, S2\}$

$Z(Q2) = \{S2\}$

$Z(S2) = \{Q2\}$

$Z(S2) = \{S2\}$

In addition, as shown in Figure 5, the quarry transport operation itself is a coupled model made of one atomic model (*Gen*) and two coupled models (*Load* and *Unload*). The formalism for the quarry transport operation model can be thus written as follows,

$Qaurry =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$

$X = \{\ \ \}$

$Y = \{HaulerID\}$

$D = \{Gen, Load, Unload\}$

$M = \{GM, LM, UM\}$

$I(GM) = \{LM\}$

$I(LM) = \{UM\}$

$Z(LM) = \{GM\}$

$$Z(UM) = \{LM\}$$

$$Z(UM) = \{UM\}$$

5. MODEL ARCHITECTURE, INTEROPERABILITY, AND SCALIBIILTY

As stated earlier, most simulation and data collection systems for AEC/FM projects are often developed for a specific purpose and phase of a constructed facility. In facility operations and building systems, for example, energy modeling and simulation tools (e.g. eQuest) are typically used during the design phase with no potential to extend their use for decision-making during the building operation phase. Similarly, building automation systems are designed to provide a platform to manage building systems (e.g. HVAC and lighting) during the operation phase of the facility. In contrast to segmented simulation models with limited scope of use, the design and implementation benefits and added values of coupling individual simulation elements at operations level were discussed in Subsection 4.2 through an illustrative example of a building heating/cooling system (Figure 5). The resulting coupled model architecture enabled the creation of a flexible model architecture that consisted of individual entities and the contextual relationships between them (e.g. input and output data dependency, hierarchical relationships) in support of the building heating/cooling operation.

In general, couplings can be used to establish connections between output and input ports to enable interoperability between models. Within this framework, a system that exhibits inter-communication as well as hierarchical relationships between its entities can be conveniently modeled using DEVS coupled modeling. A good example of such systems is prefabrication shops (e.g. steel shape manufacturing) where raw materials go through several workstations according to a fabrication sequence to eventually be transformed into a final product (Sadeghi and Robinson Fayek 2008). Such systems can be essentially modeled as multi-agent coupled systems consisting of two agents: a leader and a follower, that communicate directly with one another. The follower agent itself can be decomposed into a series of internal sensors, controller, and actuators. In case of a steel shape fabrication shop, the sensors can detect temperature or vibrations.

At a broader scheme, and within the context of a project or facility life cycle, it can be inferred that coupling can result in robust model architectures that are built upon multiple models or other coupled models each defining different subsystems, or may have developed in various platforms for different purposes. Each model has its own input, output, and transition functions. Within the FM domain, for instance, researchers have discussed that the capability of using data available through the building automation system in energy simulation models has the potential to improve decision-making and reduce energy use in buildings (Kamat et al. 2013; Menassa et al. 2014; Wetter 2011). As AEC/FM projects become more complex, these individual and decoupled systems will soon turn obsolete and provide little value (if any) to a decision-maker whose goal is to explore solutions across the entire facility life-cycle with a high level of detail and confidence. One solution is to develop very large and complex simulations. In this case, the fidelity of such models will be a major concern mainly because model developers have expertise in only a few specific parts of the domain to be modeled. One approach to address these issues is to allow distributed systems to communicate with each other and exchange required data without altering the core environment in which these systems run. This concept, typically known as distributed and remote component-based computing, has been gaining traction in several areas such as manufacturing, defense, and space mission control (Reid 2000; Steel 2000).

Another practical case in simulation coupling is from the transportation domain where modelers have been leveraging large-scale complex simulation programs, such as CORridor SIMulation (CORSIM) and TRansportation ANalysis SIMulation System (TRANSIMS), to design highway and traffic signal systems, and conduct regional transportation system analyses (Owen et al. 2000; Nourzad and Pradhan 2013). The majority of the contemporary traffic simulation systems (e.g. TRANSIMS) are based on stochastic cellular automata approach. Such simulation programs typically model the movement of vehicles based on geometric conditions of a road network, control conditions (e.g. traffic signals), and behavior of drivers on road. The cellular automata approach divides each road segment into a finite number of cells, and incorporates particle-hopping theory where a vehicle moves from one cell to another cell based on a set of transition rules. An example of a transition rule is shown in Figure 6 where a vehicle can change lane if another vehicle is not next to it in its adjacent lane. In this Figure, the center vehicle hops to the next lane based on particle-hopping theory and transition rule. The inputs to such programs usually include road network data, spatial distribution of households in a given region, and

control conditions captured by sensors. As will be discussed in Section 5, one of the challenges associated with performing large-scale traffic simulations has been the lack of interoperability among input data sources. For example, traffic network data are available in different proprietary and open-standard formats, and sensor data are typically stored in vendor-specific proprietary format.

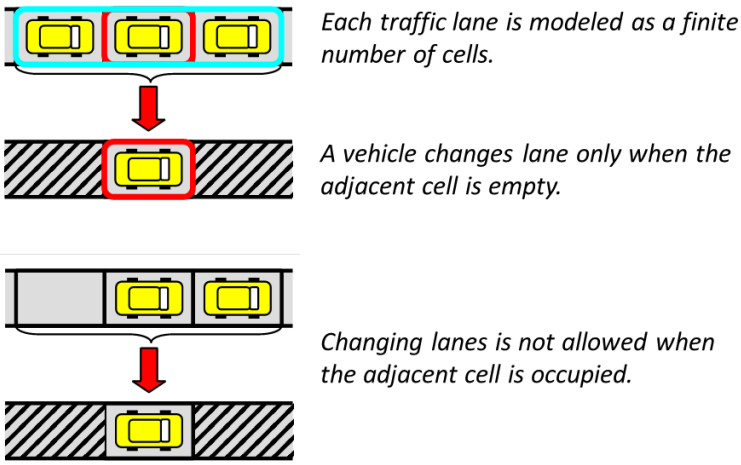

Fig. 6 Example of cellular automata to model vehicle movements based on particle-hopping theory and transition rules.

Several methods provide frameworks, sets of functional rules, and common interfaces for integrating separate and disparate simulation models and systems into larger more inclusive simulations for decision-making. Examples include Distributed Interactive Simulation (DIS), Ptolemy II, and High Level Architecture (HLA). A common characteristic of these and similar systems is that they provide a unified methodology to facilitate and synchronize data exchange across multiple separate simulation models and data acquisition systems. In the following Subsections, HLA is introduced as an example of such simulation coupling formalisms.

5.1. High Level Architecture (HLA)

In this Subsection, a detailed account of the HLA as a potential framework for use in AEC/FM simulation applications is provided. HLA has gained a significant interest in AEC/FM mainly because its underlying architecture simplifies the system development process and allows different components of the system to be easily reused in other applications. This is mainly due to the widespread knowledge and clear understanding of how the Run Time Infrastructure (RTI), the software that implements the HLA interface specification, works (Steel 2000). Recently, some researchers presented examples of how HLA applications can support decision-making in AEC/FM environments (e.g. Kamat et al. 2013; Menassa et al. 2014). The main goal of using HLA is to enable scalability, extensibility, interoperability, and shared model development for complex multidisciplinary problems, and to allow reuse and assembly of multiple (existing) models as part of a larger, interdependent, and complex simulation, while reducing software development and implementation costs (Kuhl et al. 1999; Reid 2000).

HLA provides a framework, a set of functional rules, and common interfaces that guide the integration of several simulators, also known as federates into one large simulation. This approach is especially suited for situations where game-like situations arise in which numerous "actors" need to be part of the simulation and interact with other actors, while maintaining the flexibility to enter and exit the simulation. Recalling the road construction project shown in Figure 3, a large simulation can model the interplay between several excavators, loaders, haulers, and traffic information to coordinate the digging, loading, hauling, and dumping processes involved. In Menassa et al. (2014), HLA was used to model the interplay between energy models (in this case DOE2) and an agent-based model designed to model occupant energy use behavior. In this work, the benefits of using an agent-based model for modeling individual occupant energy use behavior was further extended to include that behavior into the energy model which would otherwise assume uniform occupant energy use characteristics. General rules governing the development of complex, interoperable simulations capable of running across multiple computers

in a distributed network environment using HLA, is described in the IEEE 1516 standard. The most important terms used in an HLA compliant model are defined below (Reid 2000):

1) *Federate* is a representation of an individual simulation model or data collection system which is a standalone component. HLA integrates these federates into a larger collaborative simulation.
2) *Federation* consists of two or more integrated federates. A session in which a federation is running is referred to as *federation execution*.
3) *Federation Object Model (FOM)* is a common object model that describes the data shared between federates in a federation. It specifies every class of object and interaction known to the federation. Every object is defined by a list of all attributes it includes while every interaction is defined by parameters. An object in this sense is a storage device for data created by the different federates during federation execution, and typically available for the duration of the simulation or deliberately destroyed prior to exiting the simulation.
4) *Simulation Object Model (SOM)* is the object model that describes the data that an individual federate shares with the federation and other interaction information. FOM defines all classes of an object, and any SOM that wishes to publish or subscribe to an object must define that object.
5) *Runtime Infrastructure (RTI)* is the software that runs the federation execution, and allows for the data sharing and interactions between all federates in the federation. The RTI software must conform to the HLA specifications and provide simulation facilitation services (e.g. time and data synchronization).

There are three primary components to the HLA specification: the HLA rules (IEEE 1516), the HLA interface specification (IEEE 1516.1), and the Object Model Template (OMT) (IEEE 1516.2). An HLA compliant model enforces HLA rules when defining a federate (i.e. single simulation model, live participant, or incoming data stream) or federation (i.e. collection of multiple running and interacting models, participants, or several data streams). The HLA interface specification defines the functional modes of interaction between multiple federates and the framework's RTI. The OMT prescribes standards for defining HLA object modeling information, which includes the data to be handled by the RTI when a simulation federate executes, the objects and interactions responsible for managing the federation execution, and the documentation describing functionality for each simulation federate. The HLA thus provides standard specifications and rules to build a large-scale distributed, scalable, extensible, and interoperable federation by combining smaller federates. Examples of complex AEC/FM processes that can be modeled under this HLA mindset are numerous. For instance, Kamat et al. (2013) and Menassa et al. (2014) illustrated that it is possible to conceive a building energy simulation federation where a set of models developed for their own purpose (e.g. occupant behavior analysis, HVAC system analysis) can be composed into a set of interacting, coupled simulations for a global objective.

Similar examples from construction phase operations (i.e. excavation, pile driving, vertical construction, and highway construction) can benefit from the HLA capabilities to improve decision-making. For example, Taghaddos et al. (2008) used HLA to couple several agent-based simulation models that represented various aspects of a project in an effort to provide a resource-driven process of simulation and optimization for large-scale projects. Azimi et al. (2011) used HLA to develop an automated and integrated project monitoring and control framework project managers to take immediate corrective measures and avoid potential problems due to deviations encountered on steel fabrication projects. Finally, AbouRizk et al. (2011) discussed the Construction Synthetic Environment (COSYE) as a new approach based on HLA framework developed for modeling and simulating construction operations.

5.2. Developing High Level Architecture (HLA) Simulations for AEC/FM

Implementing and customizing an HLA-based simulation federation for AEC/FM applications requires a carefully designed approach since HLA does not exist as readily consumable software that can be used in a plug-and-play environment. Thus, a simulation model developer should: 1) develop a conceptual understanding of the problem to be modeled, its different components (i.e. federates), and the expected data exchanges and interactions between these federates during federation execution; 2) investigate the requirements for creating an HLA-based simulation framework; and 3) establish the necessary computing infrastructure. As far as the specific design of the software infrastructure is concerned, several implementations of the HLA RTI are available that can be evaluated and compared by the model developer for their scalability, platform-independence, development state and continuity, and cost (Liu et al. 2006). Among all services provided by an RTI implementation, communication between processes is the most fundamental capability for distributed

simulation. Within a whole distributed simulation system (i.e. federation), each sub-simulation process (i.e. federate) interacts locally with an RTI Ambassador process (RTIA). The RTIA will always "listen" to both the federate and the RTI Gateway (RTIG), which is the centralization point in the architecture that manages the creation and destruction of the federation execution, as well as the publication and subscription of data. When a message is received from a given RTIA, the RTIG delivers it to the interested RTIAs. Overcoming the challenge of making all constituent simulation models HLA-compliant will thus make it conceivable to construct a larger coupled simulation system using the constituent federates as building blocks. There are still significant challenges in HLA simulation. For example, Liu et al. (2006) found it useful to store the specifications of all the HLA objects in a database and then allow users to control their mappings to the federates before the start of a simulation. This allows the same model to map to simulations with just one federate on a single computer to a simulation with multiple federates. However, in practice, there can be constraints that limit the mappings, because of data privacy concerns.

6. INPUT-OUTPUT DATA REPRESENTATION

The available multimodal data sources for simulation consist of a hierarchy of knowledge, information, and data contained in their digital representations. As shown in Figure 7, different formalisms (i.e. model representations) are currently used to store and query such multimodal data sources.

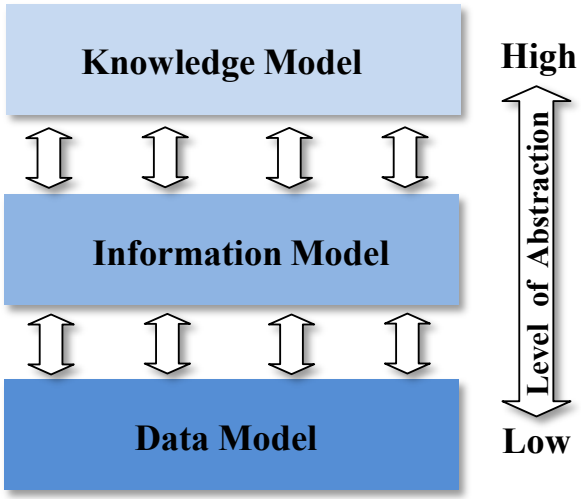

Fig. 7 Level of abstraction of different models

According to the National Institute of Standards and Technology (NIST), an **information model** is defined as "*a representation of concepts, relationships, rules, constraints, and operations to specify data semantics for a chosen domain of disclosure*" (IAI 2011; Eastman et al. 2008). In general, the terms information model and data model are used interchangeably; however there is a difference between them. An information model provides formalism without constraining how that description is mapped to an actual implementation in software. Examples of information model are building and bridge information models. There may be many mappings of the information model. Such mapping is called a **data model**. A data model details how to take real world objects and make them interpretable to a computer system. Data models focus more on data structure, units, format, and the mechanisms for indexing and storing the data sets. On the other hand, **knowledge model** is aimed at storing and capturing heuristic knowledge in a formal way to simulate intelligence. Examples of knowledge models are semantic network model and mental reference model.

Several researchers within the AEC domain have emphasized the importance of fusing different types of knowledge, information, and data models for the development of simulation frameworks (Akhavian and Behzadan 2014, Pradhan and Akinci 2012). The experts view that future data fusion frameworks need to incorporate "deep" problem domain knowledge in order to achieve the problem solving proficiency of human analysts, and for this purpose, knowledge, information, and data models will play an increasingly important role towards the development of successful, robust, and context-sensitive algorithms and frameworks (Akinci et al. 2008). The fusion of different knowledge, information, and data models brings about the issue of

interoperability which refers to the capability to communicate and transfer knowledge, information, and data among different models in a manner that does not require the user to necessarily have much knowledge about the unique characteristics of these models. The following Subsections will describe the differences in knowledge, information, and data models, and the recent interoperability standardization efforts in three different domains of knowledge modeling, information modeling, and data modeling.

6.1. Knowledge Modeling

Realizing interoperability as being an important issue, several international standards organizations such as the World Wide Web Consortium (W3C) has initiated efforts to develop standards on knowledge representation languages and frameworks. Different standards such as Resource Description Framework (RDF), Web Ontology Language (WOL), and DARPA Agent Markup Language (DAML) supported by the U.S. Defense Advanced Research Projects Agency (DARPA), and DAML+OIL have been developed and currently extended for knowledge modeling (Fensel et al. 2001; Fensel 2002; W3C 2010). These standards are eXtensible Markup Language (XML) based knowledge representation languages. Some standards, such as DAML+OIL, are an extension of two standards (i.e. DAML and OIL). The purpose of developing these standards is to allow machines to automatically understand the meaning (or semantics) of knowledge available on the World Wide Web (WWW). Similar efforts are carried out by a variety of consortiums in information modeling domain as discussed in the following Subsection.

6.2. Information Modeling

A variety of AEC and geospatial consortiums have focused on developing information modeling standards to enable seamless information transfer and interoperability among software systems within each domain. For example, the International Alliance for Interoperability (IAI) is specifying standards such as BIM and Bridge Information Modeling (BrIM) for the AEC community (IAI 2011; Eastman et al. 2008). Similarly, Open Geospatial Consortium (OGC) has been carrying out similar standardization efforts such as the Geography Markup Language (GML) and Sensor Markup Language (SensorML) for the geospatial community (OGC 2010). Recently, the interest in inter-domain interoperability between AEC and geospatial domains has spurred new standardization efforts such as the Industry Foundation Class (IFC) 2x3G specification (IAI 2011) which is aimed at integrating BIM and Geographic Information system (GIS) information models (Akinci et al. 2008). While these efforts have focused on enabling information exchange between various information modeling platforms, they have not yet addressed issues related to differences in data modeling capabilities between them.

6.3. Data Modeling

The OGC has also been working on a number of data model standardization efforts such as Geography Imagery Encoding specification, Symbology Encoding Implementation specification, and Filter Encoding specification (OGC 2010). Similarly, the National Institute of Standards and Technology (NIST) has been leading multiple data modeling standardization efforts related to exchange file formats such as the STEP format (Eastman et al. 2008). Such standards have solely focused on data modeling domain.

6.4. Interoperability Challenges

Although data, information, and knowledge models have different roles, the mutual dependencies of data, information, and knowledge need to be taken into consideration for performing data-driven simulations. It is important to understand how data, information, and knowledge should be characterized so that their differences and other relationships that may influence the performance of the resulting simulation model can be properly identified and unified. For instance, in order to generate realistic traffic simulations for better understanding of traffic congestion, different types of knowledge, information, and data models are required. For example, real-time positions of different types of vehicles could be obtained from GPS devices carried by drivers. The transmitted positional information, instead of being stored in proprietary formats, can use open standard formats, such as sensorML. Similarly, the road network structure can be encoded using GML instead of proprietary vendor-specific geospatial formats. In fact, at present one of the main obstacles to data-driven simulation is not the shortage of raw data; rather it is the lack of interoperability among heterogeneous data models that yields unreliable and/or inadequate amount of quality data that can be readily used inside simulations. The main

advantage of leveraging existing open standard specifications for different models is to minimize such interoperability problems.

7. SUMMARY AND CONCLUSIONS

This paper reported on the latest findings of an ongoing collaborative project being conducted by the Simulation Taskforce of the Visualization, Information Modeling, and Simulation (VIMS) Committee of the American Society of Civil Engineers (ASCE), and aim to lay the foundation for future research and implementation efforts in AEC/FM domain-specific real-time simulation systems. It is widely accepted that simulation allows a decision-maker to make informed decisions related to design, planning, execution, and maintenance practices of AEC/FM systems. The conventional human-centered decision-making process has critical limitations in terms of performing what-if scenario analyses due to the inability of properly incorporating real time data which can provide valuable nuggets of information. The goal of this paper was to investigate the underlying challenges associated with modular design, data interfaces and translation, and simulation model development and validation. Several examples from different AEC/FM domains including construction, building energy, and transportation were used throughout the paper to highlight the role and importance of the presented discussions.

In particular, this paper presented details about Discrete Event Simulation Specification (DEVS) as a powerful simulation formalism that can streamline the mathematical formulations describing input-output relationships in a real system, high-level architecture (HLA), as well as data-information-knowledge representation and interoperability challenges associated with simulating AEC/FM systems. First, the process of creating atomic (single) and coupled models using DEVS formalism was described. Next, the HLA compliant model was presented through discussions about concepts such as federate, federation, federation object model, simulation object model, and real-time infrastructure. Finally, the input-output representation section provided the concept of hierarchy of knowledge, information, and data contained in multi-modal data sources that could be used in simulations, as well as the interoperability challenges associated with such data sources.

The materials presented in this paper provided fundamental knowledge about three important aspects of simulation modeling that would otherwise take much longer to comprehend from the sparse literature. This will serve as basis to encourage future research and investigation of domain specific real-time simulation systems for AEC/FM. A thorough understanding of these three aspects (i.e. model formalism, model architecture, and data representation) is of utmost importance since in simulation modeling, one needs to know how entities function and interact (goal of formalism), how model components are assembled to create complex simulations (goal of model architecture), and how data is interpreted and transferred between different parts of a simulation model (goal of data representation). Ultimately, the discussions presented in this paper are sought to lay the foundation for future research and implementation efforts in AEC/FM domain-specific real-time simulation systems.

8. REFERENCES

AbouRizk S., Halpin D., Mohamed Y., and Hermann U. (2011). Research in Modeling and Simulation for Improving Construction Engineering Operations, *Construction Engineering and Management*, American Society of Civil Engineers (ASCE), 137(10), 843-852.

AbouRizk S. (2010). Role of Simulation in Construction Engineering and Management, *Construction Engineering and Management*, American Society of Civil Engineers (ASCE), 136(10), 1140–1153.

Akhavian R., Behzadan A.H. (2014). Evaluation of Queuing Systems for Knowledge-Based Simulation of Construction Processes, *Automation in Construction*, Elsevier, 47, 37-49.

Akhavian R., Behzadan A.H. (2013). Knowledge-Based Simulation Modeling of Construction Fleet Operations Using Multimodal-Process Data Mining, *Construction Engineering and Management*, American Society of Civil Engineers (ASCE), 139(11), 04013021-1-11.

Akinci B., Karimi H., Pradhan A., Wu C.-C., Fichtl G. (2008). CAD and GIS Interoperability through Semantic Web Services, *Information Technology in Construction*, 13, 39-55.

Antoine-Santoni T., Santucci J.F., De Gentili E., Costa B. (2007). Modelling and Simulation Oriented Components of Wireless Sensor Network Using DEVS Formalism, *Proceedings of the Spring Simulation Multi-Conference*, Society for Computer Simulation International, Norfolk, VA.

Azimi R., Lee, S., AbouRizk, S.M., Alvanchi, A. (2011). A Framework for an Automated and Integrated Project Monitoring and Control System for Steel Fabrication Projects, *Automation in Construction*, Elsevier, 20(1), 88-97.

Bosché F., Guillemet A., Turkan Y., Haas C., Haas, R. (2013). Tracking the Built Status of MEP Works: Assessing the Value of a Scan-vs-BIM System, *Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), 10.1061/(ASCE)CP.1943-5487.0000343.

Cassandras C.G., Lafortune S. (1999). *Introduction to Discrete Event Systems*, Springer, New York, NY.

Chang D.Y. (1986). *RESQUE: A Resource Based Simulation System for Construction Process Planning*, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Eastman C., Teicholz P., Sacks R., Liston K. (2008). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, Wiley, Hoboken, NJ.

Fensel D. (2002). Ontology-Based Knowledge Management, *Computer*, Institute of Electrical and Electronics Engineers (IEEE), 35(11), 56-59.

Fensel D., Van Harmelen F., Horrocks I., McGuinness D.L., Patel-Schneider P.F. (2001). Oil: An Ontology Infrastructure for the Semantic Web, *Intelligent Systems*, Institute of Electrical and Electronics Engineers (IEEE), 16(2), 38-45.

Golparvar-Fard M., Peña-Mora F. (2007). Application of Visualization Techniques for Construction Progress Monitoring, *Proceedings of the International Workshop on Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), Pittsburgh, PA.

Hajjar D., AbouRizk S.M. (1999). SIMPHONY: An Environment for Building Special Purpose Construction Simulation Tools, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Phoenix, AZ.

Halpin D.W. (1990). *Micro-CYCLONE User's Manual*, Department of Civil Engineering, Purdue University, West Lafayette, IN.

Halpin D.W., Woodhead R. (1976). *Design of Construction and Process Operations*, Wiley, New York, NY.

Hu X. (2004). *A Simulation-Based Software Development Methodology for Distributed Real-Time Systems*, Ph.D. Dissertation, University of Arizona, Tucson, AZ.

IAI (2011). International Alliance for Interoperability, <http://www.buildingsmart.org/> Accessed August 2011.

Ioannou P.G. (1989). *UM-CYCLONE Reference Manual*, Technical Report UMCE-89-11, Department of Civil Engineering, University of Michigan, Ann Arbor, MI.

Jang W., Skibniewski M. (2009). Embedded System for Construction Asset Tracking Combining Radio and Ultrasound Signals, *Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), 23(4), 221–229.

Kamat V.R., Menassa C., Lee S. (2013). On-Line Simulation of Building Energy Processes: Need and Research Requirements, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Washington, DC.

Kuhl F., Weatherly R., Dahmann J. (1999). *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall, Upper Saddle River, NJ.

Laplante P.A. (1997). *Real-Time Systems: Design and Analysis (2nd Edition)*, Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ.

Lee S., Behzadan A.H., Mohamed Y., Kandil A. (2013). Grand Challenges in Simulation for the Architecture, Engineering, Construction and Facility Management Industry, *Proceedings of the International Workshop on Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), Los Angeles, CA.

Li X., Vangheluwe H., Lei Y., Song H., Wang W. (2011). A Testing Framework for DEVS Formalism Implementations, *Proceedings of the Symposium on Theory of Modeling and Simulation: DEVS Integrative M&S Symposium*, Society for Computer Simulation International, Boston, MA.

Liu L.Y. (1991). *COOPS: Construction Object-Oriented Simulation System*, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Liu B., Yao Y., Tao J., Wang H. (2006). Development of a Runtime Infrastructure for Large-Scale Distributed Simulations, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Monterey, CA.

Maria A. (1997). Introduction to Modeling and Simulation, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Orlando, FL.

Martinez J.C. (2001). EZStrobe – General-Purpose Simulation System Based on Activity Cycle Diagrams, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Arlington, VA.

Martinez J.C. (1996). *Stroboscope: State and Resource Based Simulation of Construction Operations*, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Menassa C., Kamat V.R., Lee S., Azar E., Feng C., Anderson K. (2014). A Conceptual Framework to Optimize Building Energy Consumption by Coupling Distributed Energy Simulation and Occupancy Models, *Computing in Civil Engineering –Special Issue on Computational Approaches to Understand and Reduce Energy Consumption in the Built Environment*, American Society of Civil Engineers (ASCE), 28(1), 50–62.

Noguchi J. (2006). The Science Review Article – An Opportune Genre in the Construction of Science, Linguistics Insights, 17, Peter Lang, Bern, Switzerland.

Nourzad H., Pradhan A. (2013). Network-Wide Assessment of Transportation Systems Using an Infection Spreading Methodology, *Proceedings of the International Workshop on Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), Los Angeles, CA.

Odeh A.M. (1992). *Construction Integrated Planning and Simulation Model*, Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

OGC (2011). Open Geospatial Consortium, Inc., <http://www.opengeospatial.org/> Accessed August 2011.

Owen L.E., Zhang Y., Rao L., McHale G. (2000). Street and Traffic Simulation: Traffic Flow Simulation Using CORSIM, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Orlando, FL.

Posse E., Bolduc J.-S., Vangheluwe H. (2003). Generation of DEVS Modelling and Simulation Environments, *Proceedings of the Summer Computer Simulation Conference*, Montreal, Canada.

Pradhan A., Akinci B. (2012). A Taxonomy of Reasoning Mechanisms and Data Synchronization Framework for Road Excavation Productivity Monitoring, *Advanced Engineering Informatics*, Elsevier, 26(3), 563–573.

Reid M.R. (2000). An Evaluation of the High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation, *Proceedings of the 25th NASA Software Engineering Workshop*, Goddard Space Flight Center, Greenbelt, MD.

Sadeghi N., Robinson Fyek A. (2008). A Framework for Simulating Industrial Construction Processes, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Miami, FL.

Shaw S.C. (2001). *Real-time Systems and Software*, Wiley, New York, NY.

Shi J.J. (2000). Object-Oriented Technology for Enhancing Activity-Based Modeling Functionality, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Orlando, FL.

Steel J.S. (2000). The Use of DIS and HLA for Real-Time, Virtual Simulation – A Discussion, *Proceedings of RTO NMSG Conference on the Second NATO Modelling and Simulation Conference*, Shrivenham, UK.

Taghaddoes H., AbouRizk S.M., Mohamed Y., Ourdev I. (2008). Distributed Agent-Based Simulation of Construction Projects with HLA, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Miami, FL.

W3C. (2011). Technical Reports, <http://www.w3.org/> Accessed August 2011.

Wainer G.A. (2004). *SYSC 5104: Methodologies for Discrete Event Modeling and Simulation*, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada.

Wetter M. (2011). Co-Simulation of Building Energy and Control Systems with the Building Controls Virtual Test Bed, *Building Performance Simulation*, Taylor & Francis, 4(3), 185-203.

Zeigler B.P., Praehofer H., Kim T.G. (1976). *Theory of Modeling and Simulation*, Wiley, New York, NY.

Zeigler B.P., Sankait V. (1993). DEVS Formalism and Methodology: Unity of Conception/Diversity of Application, *Proceedings of the Winter Simulation Conference*, Institute of Electrical and Electronics Engineers (IEEE), Los Angeles, CA.