PROJECT MANAGEMENT GAMES USING HIGH LEVEL ARCHITECTURE

Ronald Ekyalimpa[(a)], Simaan Abourizk[(b)], Yasser Mohamed[(c)], Farzaneh Saba[(d)]

[(a)] PhD Candidate, Dept. of Civil and Environmental Engineering, Univ. of Alberta, Edmonton, Alberta, Canada T6G 2G7.
[(b)] Professor, Dept. of Civil and Environmental Engineering, Univ. of Alberta, Edmonton, Alberta, Canada T6G 2G7.
[(c)] Professor, Dept. of Civil and Environmental Engineering, Univ. of Alberta, Edmonton, Alberta, Canada T6G 2G7.
[(a)] PhD Graduate, Dept. of Civil and Environmental Engineering, Univ. of Alberta, Edmonton, Alberta, Canada T6G 2G7.

[(a)]rekyalimpa@ualberta.ca, [(b)] abourizk@ualberta.ca, [(c)] yaly@ualberta.ca, [(d)] fsaba@ualberta.ca

ABSTRACT
Numerous simulation games have been developed since the 60's by researchers within the construction domain. These games were developed for a specific purpose and possess a unique implementation strategy and structure. This paper summarizes some of these games and further discusses a subset of them, particularly those developed within the COnstruction SYnthetic Environment (COSYE). A prototype for a game development framework was proposed based on review of these games. It is proposed to implement this prototype within a distributed simulation environment to ensure consistency, interoperability, and reusability of the components and the game as a whole. COSYE is chosen to provide such an environment because it has the necessary ingredients for developing and maintaining such a prototype. The paper further discusses some concepts of distributed simulation, the features that exist within the COSYE framework, and the standards on which the COSYE framework operates i.e. High Level Architecture (HLA).

Keywords: Simulation games, prototype, COSYE, HLA

1. INTRODUCTION

The pace of the construction industry has traditionally been driven by the needs of its clients and national and global economies in general. This makes it a highly dynamic industry. Lately, projects within this industry have significantly evolved with respect to complexity and size. They are generally larger and more complex, demanding more refined competencies, with respect to cost and time efficiency, safety and quality from the people executing them. As a result, average recent university graduates generally find it challenging to match this required skill set, especially if they are a product of a program that utilizes traditional methods of classroom instruction. Moreover, the industry is highly competitive at an individual and company level.

An effective way of resolving such issues is to use methods complementary to traditional ones. Examples of these methods include site visits, the use of simulation games, the use of case studies and guest lectures from experts in the industry. Simulation games have a lot of potential to develop the desired competencies amongst students because they create a virtual construction site environment with which the students interact as though they were on a real job site. Also, the implementation of these games does not demand significant logistical requirements compared to the other methods, and yet, the players experience sufficient training time and the desired benefits are realized. Therefore, the use of simulation games in construction education needs to be given more emphasis and the tools required to support their use must be up-to-date and easy-to-use.

This paper reviews the structure and development of past simulation games with the objective of deriving common features among them from which a generic game development prototype can be proposed. It is envisaged that this prototype will simplify the development process of future games. Although the prototype can be applied in any suitable environment while developing a game, the authors base their discussion in this paper on an environment that supports the development and execution of distributed simulation systems, COSYE, developed at the University of Alberta by the second and third authors of this paper.

2. CONSTRUCTION MANAGEMENT GAMES

A review of the various simulation-based games that have been implemented in the past within the construction domain reveals a significant degree of diversity with respect to their structure, internal processing algorithms and overall purpose. Nonetheless, they can be placed within two broad categories: process centric games and non-process centric games.

Process centric games are those that model site-level operations using typical process interaction

modeling approaches. They involve a lot of resource manipulations and event scheduling (which replicate the complex logical sequence of project activity execution). The purpose of such games is to teach students how to allocate resources as construction progresses, and how to deal with other uncertain events that are typical of construction projects such as labor strikes, equipment failures, bad weather and other unforeseen poor site conditions. These games develop student skill and knowledge regarding how to effectively keep projects in control amidst such unfavorable circumstances. Examples of games that have been developed in this category include: a foundation excavation game (Au and Parti 1969), CONSTRUCTO (Halpin 1976), a road construction game (Harris et al. 1977), the muck game (Al-Jibouri and Mawdesley 2001) and a tunneling game (Ekyalimpa et al. 2011). CONSTRUCTO was developed by Halpin in 1976 to teach students how to deal with unforeseen site conditions such as labor shortages and unfavorable weather conditions while executing a construction project. It was also aimed at teaching students how to manage resources on their projects from a constrained global resource pool. The foundation game developed by Au and Parti in 1969 was focused on teaching students how to deal with the dynamics and uncertainties associated with building foundation excavation and construction. The muck game was introduced by Al-Jibouri in 2001 and its main purpose was to teach students the concepts involved in earth-moving operations.

The non-process centric games focus their efforts on teaching students the different concepts of bidding strategies, cost estimation, scheduling and dispute resolution. Examples of games within this category include Construction management game (Au and Parti 1969), SUPERBID (AbouRizk 1993), STRATEGY (McCabe et al. 2000), equipment replacement game (Nasser 2002), virtual construction negotiation (Yaoyuenyoung et al. 2005), Easy Plan (Hegazy 2006), and MERIT (Wall and Ahmed 2006). Different versions of superbid have been developed since the release of the original version by AbouRizk. These have adapted different implementation approaches and have had some additions made to them, such as the virtual player implemented in a version of superbid within COSYE by AbouRizk et al (2010). Details of this version of the game, along with other games implemented within COSYE, will be discussed in this paper.

In their paper, AbouRizk et al (2010) pointed out that with the advances in computing technologies, most of the games discussed above have been rendered obsolete as they can no longer be implemented on today's computers. This could be attributed to the static nature of the methodologies that were used to implement these games. Nonetheless, there are a number of insights to be gained from reviewing the different implementation strategies used for developing the games. These approaches can be summarized as those that were: (1) operated on standalone computers, (2) implemented in a database-server environment, (3) developed in a web-based environment and (4) implemented in a distributed simulation environment like COSYE. For a game setting, a file server, web-based server and distributed simulation approaches offer more implementation flexibility and are acceptable. This issue will be further discussed in the conceptual framework for game development.

3. DISTRIBUTED SIMULATION USING COSYE AND THE HLA

COSYE is an application programming interface which supports the development of large-scale distributed synthetic simulation environments. It is based on the High Level Architecture (IEEE 1516) standard for developing large-scale models (AbouRizk and Hague 2009) and facilitates the creation of separate simulation components (also known as federates) and their integration into a single simulation system (known as a federation) during execution.

The HLA standards are guidelines prescribed by the Institute of Electronic and Electrical Engineers (IEEE) to ensure that distributed simulation systems are developed in an interoperable, reusable and consistent manner. The HLA standard is comprised of three components, namely: the rules, the Object Model Template (OMT) and the Interface specifications.

A typical distributed simulation system will be represented by one federation and a number of federates. A federation can be defined as a virtual space/environment which represents a distributed simulation system. It has different components (federates) that are responsible for its simulation behavior. A federate on the other hand is a piece of software with the capability of participating in a federation execution. Figure 1.0 shows a schematic layout of a distributed simulation system with the different components. Some of these components will be explained in detail. The COSYE framework provides the services required to create federates and integrates them into an executable federation. COSYE has different components which facilitate developers to achieve this. They include: a Run Time Infrastructure (RTI), an OMT editor, a federate host, and a federate form.

3.1. The Run-Time Infrastructure (RTI)

The RTI is software that hosts the virtual distributed simulation environment referred to as a federation. It also manages communications between the different components (federates) within this virtual environment, along with other features such as the object instances and messages within the environment. COSYE has its own RTI which has the ability to support multiple instances of federations. Communication exchanges between the components are not direct, but rather, to the RTI, which delivers these messages in the right order and right time. A message thread sent from the source to a receipt is referred to as a *call* while a response to such messages is referred to as a *callback*.

The exchange of messages between the RTI and federate is accomplished in the HLA and COSYE through the *federateambassador* and *RTIambassador*, which reside within each federate (as shown in Figure 1.0). A federate ambassador and RTI ambassador are classes defined within the *Cosye.Hla.Rti* library. These classes have methods which permit federate developers to make calls and receive call-backs. In COSYE, a federate ambassador is created within a federate by generating a new instance of the federate ambassador class as in any object-oriented programming development (using the *new* keyword). The RTI ambassador, on the other hand, is created when the federate successfully connects to the RTI server. This is accomplished by invoking the connect method of the federate ambassador which takes the location of the RTI (i.e. its URL) as a string parameter. If the call is successful, the method returns a reference to an object of type RTI ambassador that is stored for later use; otherwise an appropriate exception is thrown.

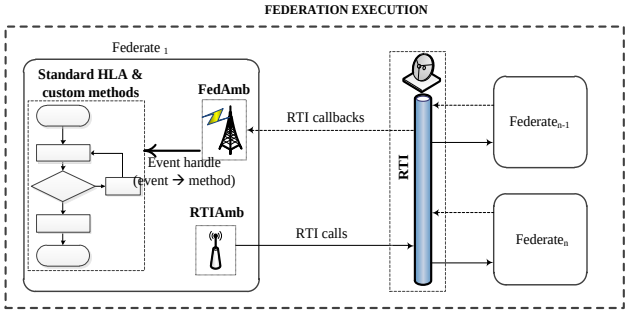

Figure 1: Conceptual Model of a Typical Federation Execution in COSYE

All this is done before the federate joins any federation. Once these exist within the federate, it can send a message requesting to create a federation, join it, execute, resign and destroy the federation. Most communications from the RTI are received by the federate ambassador which in turn translates them into respective methods in which user defined code/algorithms are implemented. It is within these methods that a developer defines the overall behavior of the federate. The RTI ambassador has methods that facilitate a federate to send information/requests to the RTI.

3.2. The COSYE OMT Editor and the Federate Object Model (FOM)

The HLA standard stipulates that an OMT is comprised of a Federate Object Model (FOM) and a Simulation Object Model (SOM). The FOM documents the object model for the federation while the SOM documents a federate's object model. The COSYE framework provides an OMT editor as a plugin within visual studio. This is added to visual studio after referencing the *Cosye.Hla.OMT* library within visual studio. The OMT editor allows developers to visually create and edit their FOMs or SOMs with ease. Figure 2.0 shows a screen shot of an FOM developed in visual studio.

The FOM is a document that specifies all the objects that will participate in a given federation execution. It represents a structured way for developers to specify the objects and interactions that exist within the federation. This is achieved through the use of the concept of a class in the Dot Net sense. The HLA provides for two types of classes; those that represent *object classes*, and those that represent *interaction classes*. Objects represent instances which persist in the simulation while interactions represent messages and do not persist during simulation execution.

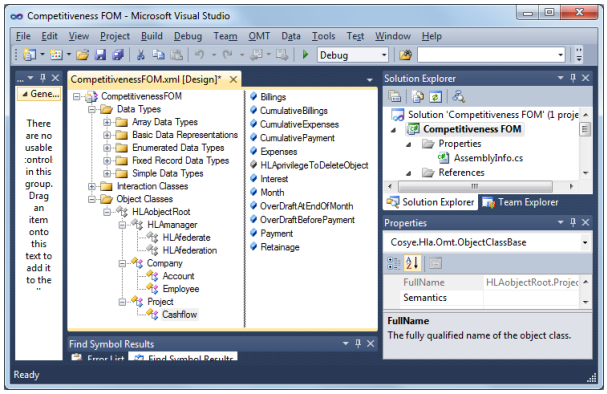

Figure 2: Screen Shot of the COSYE OMT Editor in Visual Studio

The FOM is structured in such a way that allows for the definition and storage of these classes and their associated fields. For the object classes, the FOM documents the name of the class and its attributes. The FOM includes specifications of the interaction class name and its parameters. However, lacking within the FOM are the methods for these classes. In the FOM, the developer can also specify the data type of the attributes and parameters, permission to publish or subscribe to them and the nature of delivery of messages about these attributes during the simulation. Every federation must have one FOM that is documented according to the Object Model Template (OMT), as stipulated in the first rule within the HLA standards (Kuhl et al. 2000).

3.3. The COSYE Federate Form, Federate Host and Test Federate

The federate form is a component that exists within COSYE in the *Cosye.Hla.Framework* namespace. It can be added to a visual studio development project as an inherited form. A sample of a federate windows form within visual studio is displayed in Figure 3.

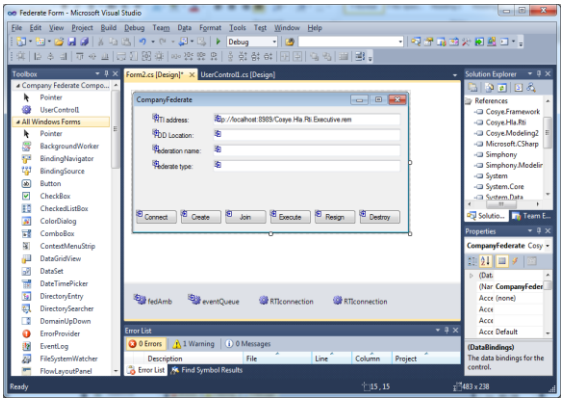

Figure 3: COSYE Federate Form in Visual Studio

This form can be used to develop a federate quickly without the need of several lines of code to implement its participation within a federation execution.

The federate host is a component that can be found within the *Cosye.Hla.Framework* name space in COSYE. This component serves as a container in which different federates participating in a federation execution are put together to provide a simple user interface. Figure 4 shows a screen shot of a sample federate host in COSYE.

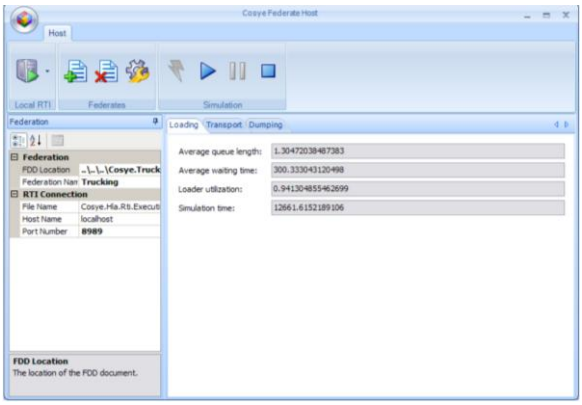

Figure 4: Federate Host (Adapted from AbouRizk 2011)

The federate host can also be used as an interface for (1) starting and shutting down the RTI (2) inputting specifications required for connecting to the RTI (i.e. the RTI host name, the port number and the name of the RTI) and (3) receiving inputs for creation of a federation (i.e. the federation name and the FOM location). The user can add or remove federates from the federate host using the respective buttons. The federation can also be created, executed and terminated within this interface.

The test federate is another component within COSYE that is used by both inexperienced and experienced developers. The former use it as a tool that facilitates the fast and easy creation, execution and destruction of a federation in COSYE. It helps them appreciate the manner in which COSYE manages the various services that the HLA provides for, such as management of: a federation, time, objects, ownership and declaration. The latter users apply it for verifying their federation development process. The test federate can also be used for verifying the behavior of the RTI. Figure 5 is a screen shot of a sample test federate within COSYE.

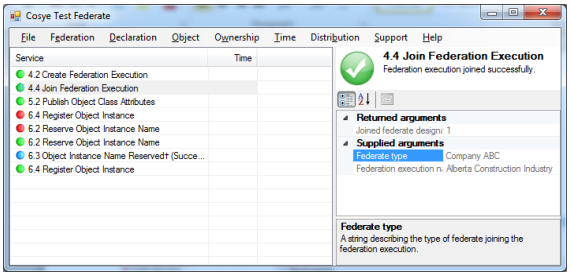

Figure 5: The Test Federate Interface in COSYE

The test federate traces a log of all the calls and callbacks made within the federation execution, along with the exceptions thrown.

4. A REVIEW OF SIMULATION GAMES DEVELOPED IN COSYE

Three games that were previously developed within COSYE are reviewed within this section with the objective of establishing features that are common to them. These features are then used as a basis for proposing the generic game development prototype for the construction domain.

4.1. The Crane Lift Planning Game

A "Mobile crane lift planning game" is the first of the three COYSE games to be discussed in this paper. The objective of this game is to teach students the concepts, knowledge and skills necessary to analyze and plan heavy lift operations on a congested site using mobile cranes.

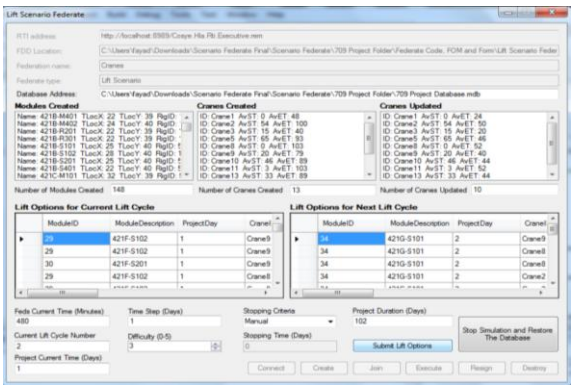

Figure 6: Scenario Set-up Federate in the Crane Lift Planning Game (Adapted from Ekyalimpa and Fayyad 2010)

The game comprised of three core federates, namely: a scenario-setup federate (developed by Ekyalimpa and Fayyad 2010), a player federate (developed by Jangmi et al. 2010) and an operations simulator (developed by Gonzales et al. 2010). Figures 6 and 7 show screen shots of interfaces for the scenario-setup federate and the player federate.

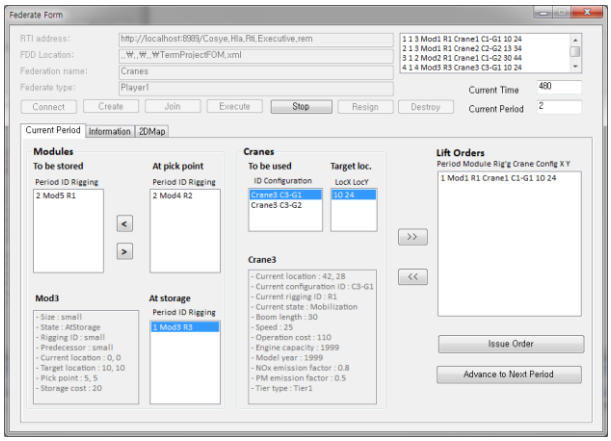

Figure 7: Player Federate in the Crane Lift Planning Game (Adapted from Jangmi et al. 2010)

In this game, an industrial construction site is assumed in which modules have to be lifted into place using mobile cranes. Modules arrive from a hypothetical assembly yard and would either be temporarily put aside while waiting to be lifted into position, or they would be offloaded into storage in case the predecessor modules have not yet been installed. This is because not all the modules arrive to site in the order in which they are to be erected on site. A finite number of mobile cranes with specified lift capacities were to be assigned to lift modules the designed location. In some cases, a mobile crane would have to be moved from one location to another in order to complete a lift. Once a lift plan has been generated (speculating the modules to be lifted, the cranes to lift them and the positions in which the lifts should be executed), this information is passed on to an operations simulator that executes the plan within a simulation environment. From the simulation, we get an indication of the time taken to complete the entire operation, including vital statistics such as the utilization of the crane resources and the time consumed before modules are erected (waiting time). This is done in cycles (module arrivals lift plan generation and lift plan execution).

This game was developed as a term project in an advanced simulation course that makes use of the COSYE environment for simulation implementations. The game is comprised of five components (federates), namely: administrator, player, operations, visualization and emissions. The visualization and emissions federates are not discussed in this paper. A conceptual model for the game implementation (federation) is shown in Figure 8.

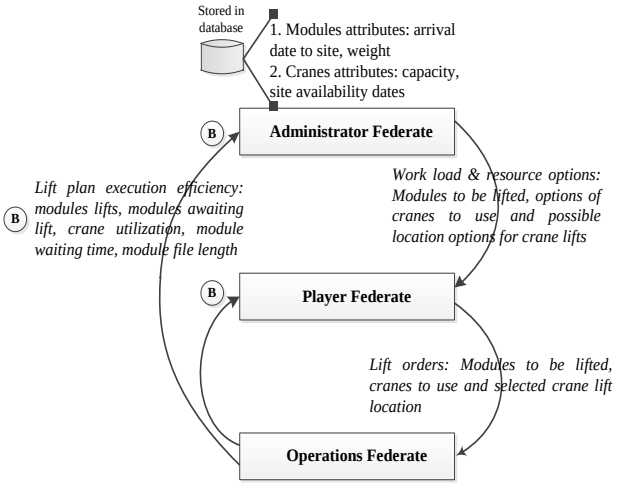

Figure 8: Interaction between Federates in the Crane Lift Planning Game

4.2. The Bidding Game (with a Virtual Player)

Different versions of the bidding game have been developed since the release of the original version, SUPERBID, by AbouRizk in 1993. However, the version discussed in this section is that developed within the COSYE framework by AbouRizk et al (2010). The primary purpose of the bidding game was to teach students bidding strategies. This version of the game comprised of six federates (shown in Figure 9). At the beginning of the game, a federation is created by the administrator federate, into which it subsequently joins. If there is a need for a virtual player, the administrator federate starts up and enables an instance of it to join the federation. Player federates then join the federation with each player representing a unique general contractor. As each player federate joins the execution, an instance of each is created. A bank account is created for each player with an initial amount of money in it, which is randomly sampled from a statistical distribution. The market federate joins the federation execution creating an instance of a market in which projects and sub-contractors will exist for the general contractor to pick from as the game advances. The player makes a decision regarding which projects to bid on, secures a bond for the bid of interest, selects subcontractors, and then submits its bid, which includes their profit margin. As the game advances, the project is awarded to the contractor that submitted the lowest bid. The winning contractor is the one who creates the most value (has the most money in their account) at the end of the game. The performance of the player in each period is dependent on the quality of the subcontractors that the contractor chooses to use, their past experience in building similar projects and the location of these projects relative to the contractor's location.

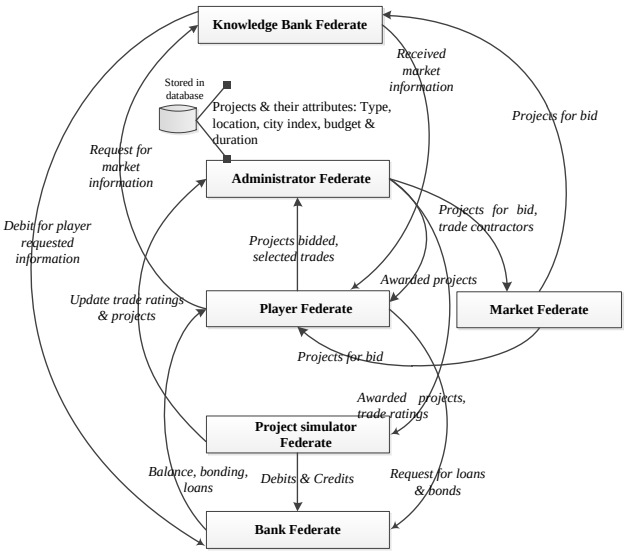

Figure 9: Interaction between Federates in the Bidding Game

4.3. The Tunneling Game

The tunneling game was built off of an existing tunneling federation. This federation was initially developed to support the planning and analysis of a tunnel to be built, or one already in construction. Incorporating gaming features into the federation was possible because the HLA and COSYE developments facilitate extensibility while maintaining their inter-operability and reusability characteristics. One component (federate) was developed from scratch to host a number of gaming facilities, namely: the user interface, the reporting facilities and the scenario generator. Figure 10 presents a screen shot of this federate.

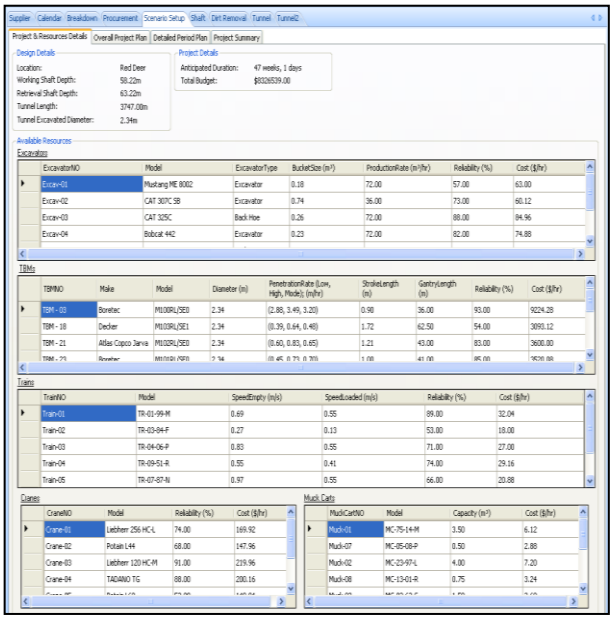

Figure 10: User Interface of the Tunneling Game (Adapted from Ekyalimpa et al. 2010)

Making use of the existing federate simulating an actual tunnel construction, the game created an instance of a tunnel which the students would be expected to construct. Attributes of the tunnel as such the length, depth, soil conditions, diameter, budget and schedule, would all be made available to the player. The tunnel instances are created from an access database that contains a list of different tunnel scenarios. This database also contains a list of different resources that would be required to execute the project. In this game, resource options would be made available to the players, such as different sizes of muck carts and different quality of TBMs (with respect to excavation rate and failure rate). Each of these would have a different cost associated with them. At the beginning of the game, players would be expected to develop a plan in which they decide the rate at which they would like to perform the work (meters advanced/day) and the resources they would like to assign to achieve that work (size and number of muck carts, type of TBM and number of crew members). For each play period, the simulator would take this plan as its input and generate results (money spent, actual time taken and liner distance advanced) at the end of the period which would be available for the player to review. If the player were not content with their performance in the last period, they could change their plan in order to improve. At the end of the game session, players would be ranked based on their performance using earned value methodology. This game teaches students how to plan for the construction operations, especially within the domain of tunneling. A concept design for this game is presented in Figure 11.

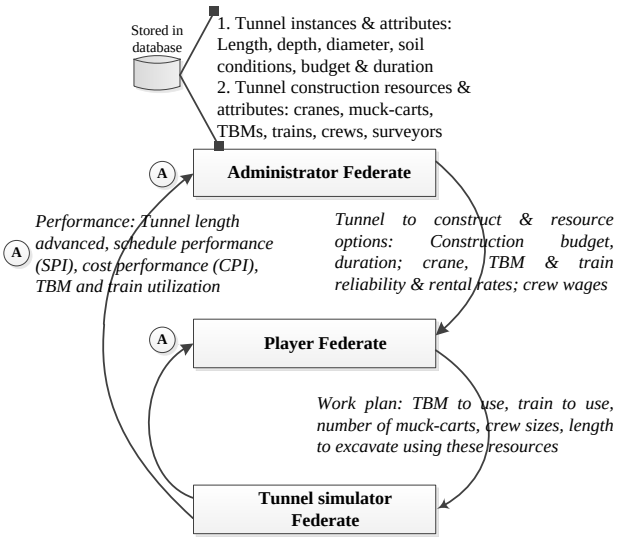

Figure 11: Interaction between Federates in the Tunneling Game

5. A PROTOTYPE FOR GAME DEVELOPEMENT

Based on the review of the games developed in COSYE, a prototype for game creation was proposed which is intended to simplify the development process. The prototype was summarized from a perspective of

component software and then from an object model perspective. These are discussed in the following sub-sections.

Given that there is a diverse range of environments in which construction management games have been implemented in the past, it should be mentioned that the generic game development prototype can also be used in any of these i.e. in a database-server setting or a web-based environment. However, for the sake of discussion in this paper, a distributed simulation environment like COSYE is proposed for use.

5.1. Software Structure of the Prototype

The structure of the prototype represents the software pieces that need to be created to obtain a fully functional game. They include: an administrator, a process simulator, player and virtual player (shown in Figure 12). This section discusses the features expected in each piece, including their anticipated simulation behavior.

The game administrator is a component which will always exist within any simulation-based game because it controls the creation and destruction of the game. Besides this, the administrator regulates a number of features during the game session such as:

- Managing the creation of possible game scenarios to be simulated.
- Managing time (the length of time between decision windows and the length of decision window).
- Varying the difficulty level of the game.
- Tracking the performance of each player.
- Enabling or disabling the virtual player from participating in the simulation.

In order to accomplish its time management responsibilities, this federate needs to be implemented as a time-stepped simulation. It also needs to be linked to a storage medium, like a database, from which it reads all the possible scenarios to be populated in the game. The administrator of the game, who in most cases will be a tutor of the instructed course, will require an interface for this federate which displays reports of player performance and other occurrences in the game during execution. This federate should have a criterion for rating the performances of each player. To guarantee that this federation will destroy the federation, it should be the last federate to achieve the "ReadyToTerminate" synchronization point, usually after a pre-determined criteria is achieved.

The game simulator represents a component that uses the inputs from the player to processes the work that needs to be done. It generates performance measures based on the actual progress made in executing this work. It is also responsible for modeling the uncertainty that surrounds the execution of this work. In order to do this, the simulator federate needs to be implemented using a time management scheme.

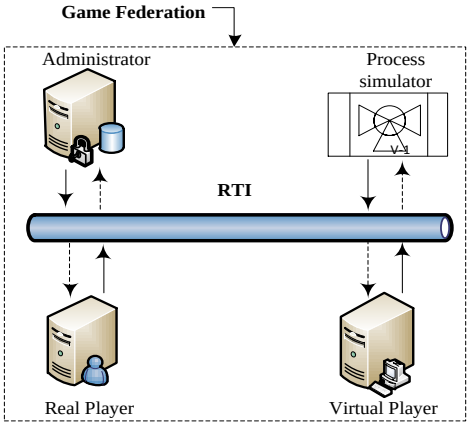

Figure 12: Conceptual Model for the Game Development Prototype

This federate may be designed as a next-event simulation or a time-stepped simulation depending on how the processing of the operation is implemented. It is mandatory for all simulator federated implemented as next-event simulation to make use of a discrete event simulation engine in its implementation. This is not the case with a time stepped simulation.

Regardless of the implementation approach used, simulator federates go through two cyclic windows during the game execution, namely, a simulation window and a decision window. The simulation window represents the phase in which the computation of algorithms is done to give rise to results which are then published. The simulator federate should be designed so that this simulation window occurs during the time granting state of the federate. The decision window, on the other hand, should be implemented in the time advancing state. This is shown in the protocol diagram in Figure 13. Federates involved in time management can have two possible states during federation execution, a time advancing state and a time granting state. A federate enters a time advancing state as soon as a call is made to the RTI to advance its time. In COSYE, this may be through a *NextMessageRequest* or a *TimeAdvanceRequest*. A federate enters a time granting state when it receives a time advance grant from the RTI.

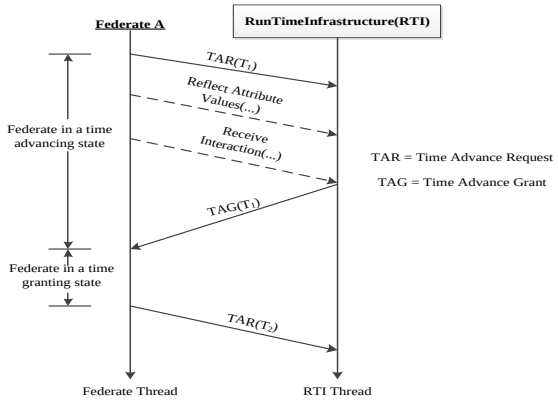

Figure 13: Schematic of a Protocol Diagram showing the States in time-stepped federate

For event-driven federates, the simulation window involves invoking the *DiscreteEventSimulationEngine.Simulate (theGrantedTime)* and *RTIAmb.NextMessageRequest (DiscreteEventSimulationEngine.TimeNext)* methods sequentially. These two methods are implemented within the time advance grant call-back of the simulator federate (when the federate is in a time granting state). This represents the processing of simulation events (algorithms) which were scheduled within the *ReflectAttributeValues* and *ReceiveInteraction* call-back methods of this federate during its time advancing state. The decision window represents a phase when the simulation engine is not doing anything. During this window, results are viewed in the player federate. Also, choices submitted by each player are received by the simulator federate, which translates them into corresponding simulation events that are scheduled to take place at the appropriate time.

In time-stepped federates, the simulation window is also within the time granting state of the federate. Simulation algorithms are processed within the time advance grant call-back method. The last statement within this method is a request for time advancement which changes the state of the federate into a time advancing state. During the time advancing state, the *ReflectAttributeValues* and *ReceiveInteraction* call-back methods of this federate are invoked as messages arrive at the federate.

In case a process discrete event simulation model (such as a Simphony special purpose template model) already exists for the domain being modeled, this model can be federated so that it participates within the respective game federations as an event-driven simulator component. This concept was successfully applied in the development of the tunneling game.

The player component is the portion of the game which participants directly interact with. The most important aspect of the player is its interface design. This should be made so that it is easy for the player to: (1) view the different game options available for upcoming play periods, (2) make a choice and submit it and (3) view their cumulative performance results from previous play periods. The game should have the capability of creating a unique instance of the federate for each participant in the game, with the exception of the virtual player which should be created by the administrator.

A virtual player is an optional, although very useful component, which is included in a game execution in situations where players engaged within a game session are generally inexperienced. The virtual player represents an experienced participant from whom the other players can learn as a result of the challenges that it poses.

5.2. Generic Federation Object Model for the Game Prototype

One other aspect investigated when comparing the three simulation games developed in COSYE was the federation object model. These were compared to establish the existence or non-existence of common features amongst them. It was established that there were some aspects of commonality amongst them which formed a basis for the standard federation object model for future game development efforts. Table 1 summarizes the object classes and parameter classes that should be present within an FOM of a game to be developed within COSYE. The associated attributes and parameters to these classes are also detailed.

Table 1: Specifications of an FOM for the Game Development Prototype

Class Type	Proposed Class Name	Attribute/Parameter
Object class	Player	Name, performance parameters, performance rating
	Scenario - projects	TBD
	Scenario - resources	TBD
Interaction class	Simulation window	Session number, time span
	Decision window	Window number, time span
	Game play options	TBD
	Player decisions	TBD
	Play period performance	TBD

Details of the data types and send order (time stamped or receive order) are left to the developer to determine. For some of the classes, attributes or parameters are highly dependent on the nature of the game being developed, and hence, are left at the discretion of the developer to determine. These have been tagged as "TBD."

6. CONCLUSIONS

The idea of using construction management simulation games as a teaching aide in construction education is relevant, especially in today's industry where there are growing expectations for construction graduates to perform well on the job, particularly in the early years of their careers. Simulation games adequately resolve the challenges that such expectations pose by serving as tools that are used to develop the skills, knowledge and experience (while still in school) required to deal with the problems that graduates face. The advances in computer technologies provide an opportunity to extend the process of developing more relevant and exciting games to effectively achieve this goal. A conceptual model that structures, guides and simplifies such developments has been presented in this paper.

The COSYE framework is proposed as an appropriate environment in which such a prototype can be effectively implemented, although other

environments, such as file server or web server systems can be used with a few modifications. The main features within COSYE that are required for developing a game federation have also been discussed.

REFERENCES

AbouRizk, S., 2010. Enhancing the competitiveness of the construction industry in Alberta. *IRC Renewal Application, Term 4.*

AbouRizk, S. and Hague, S., 2009. An overview of the COSYE environment for construction simulation. *Proceedings of the 2009 Winter Simulation Conference*, pp. 2624-2634. December 13-16, Austin (Texas, USA).

AbouRizk, S. M., 1993. Stochastic simulation of construction bidding and project management. *Microcomputers in Civil Engineering*, 8 (2), 343-353.

Al-Jibouri, S. and Mawdesley, M., 2001. Design and experience with a computer game for teaching construction project planning control. *Engineering Construction and Architectural Management*, 8 (5), 418-427.

Au, T., Bostleman, R., and Parti, E., 1969. Construction management game – deterministic model. *Journal of Construction Division*, 95 (CO1), 25-38.

Au, T., and Parti, E. W., 1969. Project planning game for foundation excavation. *Journal of the Construction Division*, 95 (CO1), 11-21.

Ekyalimpa, R., Al-Jibouri, S., Yasser, M., and AbouRizk, S., 2011. Design of a tunnel simulation game for teaching project control in construction. *Proceedings of Construction Research Congress Conference*, pp. 2118-2128, June 14-17, Ottawa (Ontario, Canada).

Ekyalimpa, R., and Fayyad, S., 2010. Administrator federate module in the Crane Lift Planning Game, internal report for Advanced Simulation course (Advanced topics in Construction Engineering and Management), University of Alberta.

Gonzalez, C., Hu, D., and Mogadam, M., 2010. Operations federate module in the Crane Lift Planning Game, internal report for Advanced Simulation course (Advanced topics in Construction Engineering and Management), University of Alberta.

Halpin, D., 1976. CONSTRUCTO – an interactive gaming environment. *Journal of the Construction Division*, 102 (CO1), 145-156.

Harris, F. C., and Evans, J. B., 1977. Road construction – simulation game for site managers. *Journal of the Construction Division*, 103 (CO3), 405-414.

Hegazy, T., 2006. Easy plan: computer game for simplified project management training. *Proceedings of the first International Construction Specialty Conference (CSCE), n.p.* May 23-26, Calgary, (Alberta, Canada).

Jangmi, H., Zhang, H., and Farzaneh, S., 2010. Player federate module in the Crane Lift Planning Game, internal report for Advanced Simulation course (Advanced topics in Construction Engineering and Management), University of Alberta.

Kuhl, F., Weatherly, R., and Dahmann, J., 2000. *Creating computer simulation systems: an introduction to the high level architecture.* Prentice-Hall International PTR, One Lake Street, Upper Saddle River, NJ 07458.

McCabe, B., Ching, K. S., and Rodriguez, S., 2000. STRATEGY: a construction simulation environment. *Proceedings of Construction Congress*, pp. 115-120. Orlando (Florida, USA).

Nassar, K., 2002. Simulation gaming in construction: ER - the equipment replacement game. *Journal of Construction Education*, 7 (1), 16-30.

Scott, D. and Culling-ford, G., 1973. Scheduling game for construction industry. *Journal of the American Society of Civil Engineers*, 99 (C01), 81-92.

Yaoyuenyong, C., Hadikusumo, B. H. W., Ogunlana, S. O., and Siengthai, S., 2005. Virtual construction negotiation game – an interactive learning tool for project management negotiation skill training. *International Journal of Business and Management Education*, 13 (2), 21-36.

RONALD EKYALIMPA

Ronald is a Ph.D. student at the Hole School of Construction Engineering in the Department of Civil and Environmental Engineering at the University of Alberta. His research focus is in the area of construction simulation.

SIMAAN M. ABOURIZK

Simaan holds an NSERC Senior Industrial Research Chair in Construction Engineering and Management at the Department of Civil and Environmental Engineering, University of Alberta, where he is a Professor in the Hole School of Construction Engineering. He received the ASCE Peurifoy Construction Research Award in 2008.

YASSER MOHAMED

Yasser is an Associate Professor in Construction Engineering and Management in the Department of Civil and Environmental Engineering, at the University of Alberta. His research focuses on simulation modelling of construction processes to support project planning and control.

FARZANEH SABA

Farzaneh graduated with her Ph.D. from the Hole School of Construction Engineering in the Department of Civil and Environmental Engineering at the University of Alberta. She now works for AECOM as a Structural Engineer.