SIMPHONY: AT THE PINNACLE OF NEXT GENERATION SIMULATION MODELING ENVIRONMENTS FOR THE CONSTRUCTION DOMAIN

Ronald Ekyalimpa[(a)], Stephen Hague[(b)], Simaan AbouRizk[(c)]

[(a)]University of Alberta, 5-047 Markin CNRL Natural Resources Engineering Facility, Edmonton, Alberta, CANADA
[(b)]University of Alberta, 5-048 Markin CNRL Natural Resources Engineering Facility, Edmonton, Alberta, CANADA
[(c)] University of Alberta, 3-015 Markin CNRL Natural Resources Engineering Facility, Edmonton, Alberta, CANADA

[(a)] rekyalimpa@ualberta.ca, [(b)] steve.hague@ualberta.ca, [(c)] abourizk@ualberta.ca

ABSTRACT
This paper presents Simphony simulation system as a tool that is leading the way in the evolution of simulation systems within the construction domain. This discussion is introduced by presenting an overview of simulation, the different simulation methods and the tools that support this method. Simphony is then introduced as an environment that supports discrete event and continuous simulation. Other features such as its extensible API, calendar, data connectivity, special purpose development abilities etc., are also highlighted to show why Simphony is a powerful simulation system. Two practical problems (earth-moving and traffic light) that are solved using Simphony are presented to demonstrate the use of some of these features.

Keywords: Simphony, extensible API, calendars, simulation methods

1. INTRODUCTION

Simulation is a numeric method that has been in use for several years and has been applied in the analysis of complex dynamic systems. The simulation community has three well established methods to apply a simulation-based approach in solving their problems: System Dynamics (SD), Agent-Based Modeling (ABM), and Discrete Event Simulation (DES).

The use of each of these methods depends on the complexity of the system being analyzed and the level to which the modeller would like to abstract the system. System dynamics is famous for its precision in modeling systems that have numerous components that are dynamic, inter-related and interact with a feed-feedback behavior. This method supports a top-bottom approach to that analysis of systems e.g. the evaluation of the impact of different policies or strategies on the behavior of a system. Simulation systems build to support this method implement numeric integration algorithms (e.g. the family of Runga-Kutta equations) in a continuous fashion. It does not involve the flow of tokens but rather tracks rates of change in specified quantities with time using integration. Examples of such systems include AnyLogic, Vensim, PowerSim, STELLA (iThink), Simulink, DYNAMO etc.

Agent-based modeling on the other hand supports a bottom-up approach to the analysis of systems. This approach models a unit or a component within a system as an agent that has intelligent behaviors that are influenced by its peers (other agents) and the environment in which it operates. An agent exhibits different behavior by transitioning through different states. This behavior is controlled by an algorithm embedded within the agents. This algorithm is defined using concepts of state diagrams. Communication between agents and the environment is triggered by the events (in the computing science sense). Examples of simulation systems that support this modeling approach include Repast-Simphony, AnyLogic, A3/AAA, ABLE, Agent Builder, MASON, NetLogo, SimAgent etc.

Discrete event simulation is an approach in which a system is described using entities, resources, activities and other modeling constructs. These constructs interact with each other to define the state of the system and are responsible for its evolution at discrete points in time. In typical DES systems, this change of state is triggered by the flow of entities. Changes in the state of a system typically occur when resources are captured or released, activities are started or finished. DES is best suited for analyzing systems at an operations level. This explains why it has been extensively used in analyzing production systems, supply chain, medical facility operations and construction operations. Various general purpose discrete event simulation software systems have been developed for a wide range of industries: AweSim (Pritsker, 1997) and GPSS/H (Crain, 1997); for construction: Micro-CYCLONE (Halpin, 1973), STROBOSCOPE (Martinez, 1996), and Simphony (Hajjar and AbouRizk, 1999).

In construction, DES has been widely used to model and improve processes such as tunnel construction, earth-moving, fabrication shops, bridge construction, and scheduling.

In 1999, AbouRizk et al. developed a special purpose template in Simphony for analyzing tunnel construction using TBMs. The template was used to evaluate the effect of different site setup configurations at the working shaft through predictions of tunnel advance rates along the tunnel length. Work on this template is on-going and has resulted in it evolving into a more sophisticated yet easy to use simulation tool for analyzing tunnel construction processes. Examples of additions to the template include: (1) ability to model shifts through the use of calendars, (2) features for generating cost estimate reports for the simulated tunnel and (3) the ability of the template to participate in a larger distributed simulation system that is based on the HLA standards, so that it can support other components such as visualization of the tunnel construction. Later on, Zhou et al. (2008) used Simphony to develop a special purpose template for modeling tunnel shaft construction. They refined the way tunnel shafts are simulated so that most site processes and constraints are well represented. They validated their template by implementing a case study (NEST NL1-NL2 tunnel in Edmonton). Modeling constructs developed in this work are used in the current version of the template. Touran and Asai (1987) used CYCLONE simulation system to the advance rate of a TBM during the construction of a long, small-diameter tunnel. Ioannou and Martinez (1996) used STROBOSCOPE simulation system to compare two alternative construction methods for rock tunneling; a conventional verses the New Austrian Tunneling Method (NATM). They demonstrated effective ways of using simulation for comparing alternatives. Al-Bataineh et al. (2013) recently wrote a paper in which they used simulation to project planning and control in tunnel construction.

The earth-moving operation is one that had been extensively analyzed using simulation because of its repetitive nature and simplicity. A small portion of the work done in simulating earth-moving operations is discussed here. In 2002, Marzouk and Moselhi combined simulation and optimization (genetic algorithms) to get optimal cost and durations associated with earth-moving operations. Fu (2012) presented a paper in which he used Global Simulation Platform (GSP), a simulation system developed by Volvo CE, and CYCLONE to simulate and compare three loading scenarios for an earth-moving operation. He compared the options based on fuel cost per unit production. However, logic flaws can be identified in some of the CYCLONE model layouts presented by Fu because they don't explicitly represent the loading of haulers with multiple buckets. In 2011, Cheng et al. proposed a simulation model for virtual simulation of earthmoving operations using petri nets. In 2009, Ahn et al. published a paper in which they presented a simulation-based sustainability analysis of earth-moving operations with respect to emissions. STROBOSCOPE and VITASCOPE were both used as simulation and visualization platforms for estimating omissions and visualizing simulated objects respectively. Rekapalli and Martinez (2011) also presented a recent paper on earth-moving operations.

Simulation has been extensively applied for modeling processes at the different stages of the delivery of industrial projects. They include: structural steel fabrication, pipe spool fabrication, module assembly. In 2008, Liu and Mohamed used an agent-based modeling approach to simulate the dynamics of resource allocation within a module assembly yard for a construction company in Edmonton, Canada. They used Repast-Simphony for their work. Song and AbouRizk (2003) developed Simphony general purpose template models (for a structural steel fabrication shop) which they integrated with CAD drawings to obtain attributes of steel pieces whose fabrication process was to be simulated. Their model used attributes of steel pieces obtained from CAD drawings and embedded artificial neural networks to predict durations for the different fabrication processes (cutting, fitting, welding and painting) (Song and AbouRizk, 2006). Alvanchi et al. (2012) developed a special purpose simulation template in Simphony for modeling the fabrication of structural steel within a shop. The template reads its input of steel pieces to be fabricated from an information management system (database) and simulates the fabrication process for different shop layouts and processing equipment so that the operational efficiency of the shop can be assessed. Sadeghi and Fayek (2008) developed a Visual Basic application that utilizes the Simphony discrete event engine behind the scenes to model operations in a pipe spool fabrication shop. Mohsen et al. (2008) used Simphony general purpose template to simulate the erection of a building that was constructed using a modularized approach. They used their model to determine the utilization of the different resources (crane, rigging crew, welding crew and delivery space) involved in the operation and compared these with those recorded on site. Wang et al. (2009) developed models that simulate the operations in a typical pipe spool fabrication shop. They compared the "traditional batch-and-queue fabrication system" to "the new cell-based work flow fabrication systems" by constructing a simulation model for each system. Cycle time for the fabrication of pipe spools was used as a statistic for comparing the two methods and the new method was found to be more efficient.

In bridge construction, a number of modeling studies have been done. Dulcy and Halpin (1998) pointed out that cable stay bridge construction provides enormous opportunity for the use of computer simulation in the analysis and design of the operations involved. They attributed this to the fact that this type of bridge involves many repetitive cycles of placing concrete segments and supporting cables. They constructed a CYCLONE model for the construction of a cable-stay bridge (Dame Point Bridge in Jacksonville, Florida) and used it to investigate different resource combinations that would result in higher utilizations and shorter construction durations. They came up with an optimum mix of resources for this problem. In 2007, Marzouk et al. also used simulation to model the construction of a bridge in Cairo, Egypt ("The 15th May Bridge") that was constructed using the incremental launching construction technique. Marzouk et al. (2007) stated that concrete bridges can be placed into 6 categories based on the construction method. These include: 1) cast-in-situ on false-work, 2) cantilever carriage, 3) flying shuttering, 4) launching girder, 5) pre-cast balanced cantilever, and 6) incremental launching. They further stated that the incremental launching construction method is the preferred option when the spans being constructed are larger than usual. Marzouk and his colleagues used STROBOSCOPE to develop a special purpose template of the bridge construction process and resource constraints. They ran their model for one scenario similar to that used on site and obtained production results that were very close to actual values on site. They then used this model to experiment with different resource mixes to shorten the project duration. A number of authors have used simulation-based methods to visualize/animate the construction of bridges. Recent examples include: Dori and Borrmann (2011), and Chui-Te et al. (2011). Visualization helps with the verification of constructed simulation models and assists is displaying the evolution of the simulation to those not knowledgeable in simulation in an effective manner. Other studies that have involved the use of simulation for modeling bridge construction processes include work done by Ailland et al. (2010), Liu (2012), and Chuen-Tsai et-al (2013).

Simulation has been used to improve scheduling practices within the construction domain. For example, a state-based simulation approach was used by Hu and Mohammed (2010) to facilitate updates of schedules developed in Microsoft project. They used the Simphony simulation engine. Other studies done on simulation-based scheduling include: Chehayeb and AbouRizk (1998), Zhang et al. (2002) and Lu (2003).

The aforementioned studies demonstrate the vast simulation opportunity that construction processes offer to the simulation community. Simphony has been used significantly in solving construction problems using a simulation-based approach, especially the more complex ones.

2. SIMPHONY SIMULATION SYSTEM

Simphony is a discrete event simulation system that was originally developed by Hajjar and AbouRizk (1999) and is currently being extended and maintained by the 2nd and 3rd authors. Simphony provides an-easy-to-use User Interface (UI), core services (a simulation engine, resources, files, calendars etc.), modeling services and simulation templates.

Simphony is built using the Microsoft .NET framework in a fashion that makes it extensible. Its Application Programming Interface (API) can therefore be utilized within the Simphony UI or any other applications that is compatible with .NET APIs. This is what makes Simphony exceptionally powerful.

Furthermore, Simphony supports the development of custom special purpose templates. These templates provide for an efficient way to abstract complex processes in a manner that makes it easy for domain experts to make use of simulation without having in-depth knowledge of the science behind the method. Examples of special purpose templates previously developed and currently supported in Simphony include the tunneling template, aggregate crushing template, de-watering template, PERT template, earth-moving template, structural steel fabrication template, and range estimating template.

Simphony supports a general purpose template that has elegant graphical modeling elements, and directional arrows which are an essential feature of discrete event simulation modeling languages/environments. Constructing GPT models requires elements to be dragged and dropped onto the modeling surface and connected with directional arrows in a convenient way. Simphony GPT also provides advanced features such as attributes for entities and scenarios, and formula editors into which user written code can be embedded within the models to facilitate solving more sophisticated problems.

Simphony is built to support Monte-Carlo simulation experiments. Figure 1.0 summarizes the manner in which Simphony processes simulation models.

Other features or services that exist within Simphony giving it an edge over other systems include:

- It has discrete event simulation capabilities and supports combined simulation as well (discrete event-continuous simulation).
- It supports calendars.
- It provides for data visualization.
- It supports connectivity to data storage applications e.g. databases.
- It has a neat user interface that provides for model debugging features (a trace window).
- Templates developed in Simphony can easily be integrated into larger distributed simulation systems (developed in line with the HLA).

The rest of the paper is dedicated to demonstrating the use of Simphony for modeling typical simulation problems.

3. MODELING WORK SHIFT DYNAMICS USING CALENDARS IN SIMPHONY.NET

3.1. An Earth-Moving Operation

In order to demonstrate the use of calendars within Simphony's general purpose template, a simple earth-moving operation is described, modeled and experimented with. We shall investigate the effect of using different calendars on the total number of calendar days it will take 5 dump trucks (@ has a capacity of 20cy) to move 10,000 cubic yards of dirt from the source to a placement area. There is one loader at the source responsible for loading dirt onto trucks. The details of the load, haul, dump and return activities are summarized in Table 1.

A simple operation (earth-moving operation) is chosen to demonstrate the concepts of utilizing a calendar within a simulation model. A brief section detailing calendar features within Simphony is introduced and then their applications in modeling the problem at hand are presented.

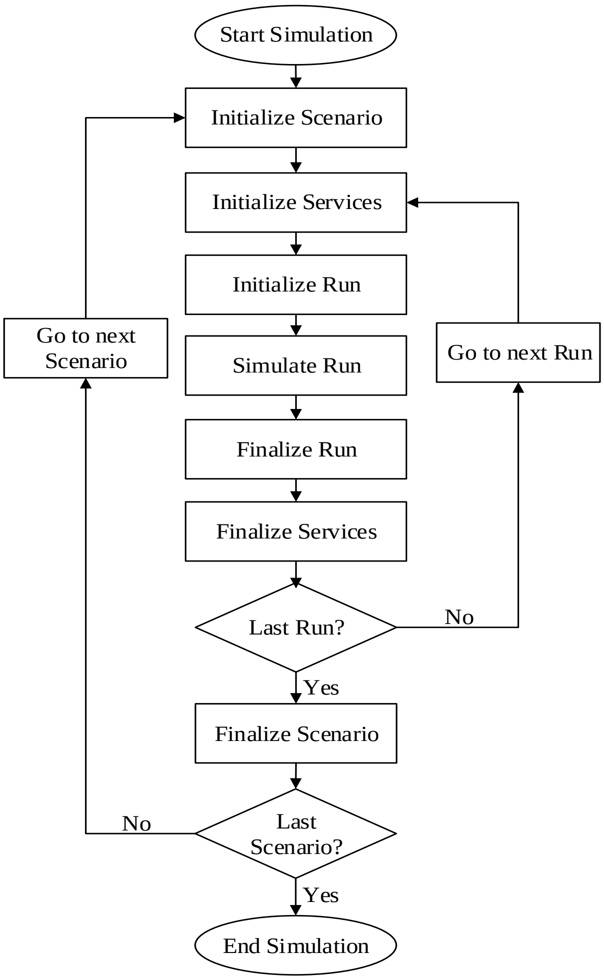

Figure 1: Schematic Layout of Simphony's Simulation Process

Table 1: Earth-Moving Activity Durations

Activity	Duration (Minutes)
Load @ Truck	Constant(8.0)
Haul to dumpsite	Constant(40.0)
Dump Truck load	Constant(5.0)
Return to source	Constant(55.0)

3.2. Calendars within Simphony.NET 4.0

Embedding and using calendars to constrain the execution of simulation models requires a clear understanding of the behavior of calendars from a simulation perspective. When activated, the calendar in Simphony continuously transitions through two states: (1) a working state and (2) a non-working state. The transition between any two calendar states (that are the same or different) gives rise to a calendar event. It is during the processing of a calendar event that a work shift (or the processing or the simulation model) gets turned ON or OFF through the *"SuspendEvent(entity)"* and *"ResumeEvent(entity)"* methods, respectively.

Simphony provides two constructs that facilitate the modeler to model work shifts: (1) a calendar, and (2) a calendar entity. The calendar keeps track of the working time periods, non-working time periods and their lengths. It is responsible for triggering calendar events whenever there is a transition in its working state. The calendar also provides methods that facilitate the modeler to get the total working or non-working times for a particular shift and pay type between specified dates. This is usually useful when the modeler is tracking costs for simulated operations.

The Calendar in Simphony is activated to start raising events as soon as the *"engine.SubscribeCalendar(...)"* method is invoked. The calendar then keeps continuously raising calendar events and will not stop until the simulation is halted by either (1) maximum criteria achieved, (2) maximum count achieved or (3) through an explicit halt invoked within an *"execute element."* The criteria for terminating the simulation through the engine running out of simulation events can never be achieved when the calendar is activated because the calendar keeps looping and raising calendar events infinitely. The calendar can be de-activated by invoking the *"engine.UnsubscribeCalendar(...)"* method. The modeller can obtain a calendar from the calendar list (defined in Simphony core services or in the *"calendar property"* of the scenario).

The calendar entity on the other hand carries with it information about the calendar event that has been triggered (summarized in Table 2). This entity is passed on to the calendar event handler so that this information can be used for implementing computations at the time that the calendar event is being processed. These properties are of a calendar entity are summarized in the schematic layout presented in Figure 2.0.

Table 2: Properties of a Calendar Entity

Calendar Entity Property	Purpose of the Property
Calendar	Gets or sets the calendar with which the entity will be associated
Entities	Avails the entities controlled by the calendar
IsWorking	Determines whether the calendar is currently in a working (true) or non-working (false) state
Time Remaining	Avails the time to the next calendar event (time span)

It is important to carefully track the work state of the calendar because it affects the action taken on the entities controlled by the calendar (suspends their processing, resumes their processing or does nothing). The modeller would like to act on the entities when a calendar event is associated with a change in the work state of the calendar (Figure 3.0) and not do anything when a calendar event is not associated with a change in the work state of the calendar (Figure 3b). An example of a situation where an action is warranted is when there is a transition between a working period and a work break and a work break and work period. This is illustrated in Figure 3a. At calendar event 1, all entities are suspended and at calendar event 2, processing of all entities is resumed. An example where there is no need for action on entities arises when there are two consecutive work periods, especially when there is a shift change without a break – i.e. from shift 1 to shift 2 (Figure 3b).

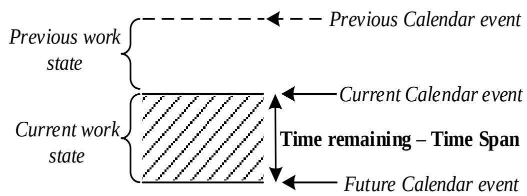

Figure 2: Schematic Layout for Occurrence of Simphony Calendar Events

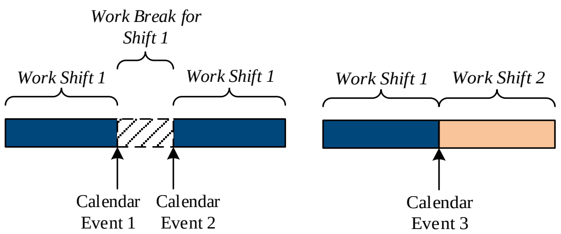

Figure 3: Different Transitions between Work Periods (3a: Work-to-Non Work-to-Work; 3b: Work-to-Work

Details of the working states associated with each calendar event are summarized in Table 3.0. Based on the explanations provided, it becomes necessary to keep track of the current work state associated with each calendar event (provided by the calendar entity) and the previous working state (tracked by the modeller).

Table 3: Shift Details Associated with Figure 3.0

Calendar Event	Current State	Previous State	Shift
1	Working	Non-working	1
2	Non-working	Working	2
3	Working	Working	1→2

At present, the intrinsic statistics reported in Simphony (such as resource utilizations and file length) are not reliable when calendars are enabled because they don't distinguish between working time and non-working time in the course of the simulation.

3.3. Simphony Model Layout, Discussion and Results

At the start of simulation, the model subscribes to a calendar when the initialize run method is invoked on the *"Subscribe to a Calendar"* execute element. Within this element, C# code snippet is written to achieve the subscription to the calendar. A method associated with the calendar event handle is written within the partial formulas class for this execute element. Figure 4.0.

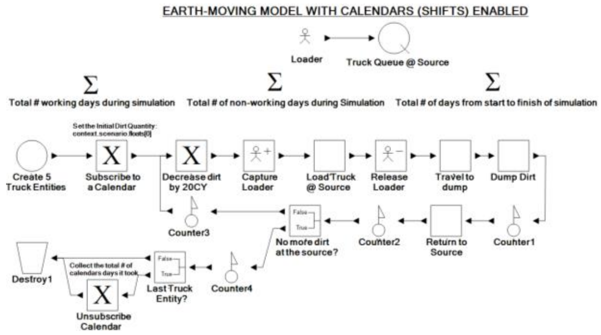

Figure 4: Earth-moving Model Layout in Simphony

Five truck entities are created at the start of the simulation which represent dump trucks. These entities flow through the model (Figure 4.0) emulating the movement of dirt from source to placement. When there is no more dirt to move, the simulation is terminated.
At the end of the simulation, the last truck entity flows through the *"Unsubscribe Calendar"* execute element where the the subscription to the calendar is undone and working days and non-working days retrieved from the calendar and saved in the respective statistics nodes. The C# code written within the execute element formula to achieve this is presented in Figure 5.0.

Three different calendars were experimented with in this model; a standard calendar, 24 Hour calendar, Night Shift and a calendar that was created with custom settings. The custom calendar used in this experiment was defined using the Simphony calendar editor (accessed through the *"Calendars"* property of the scenario – see Figure 7.0). The calendar was setup such that the work (or non-work) periods presented in Table 4.0 are utilized. For simplicity, all work periods were considered as regular time since the simulation did not model dynamics of work performance changes with work time or costs incurred due to different pay types.

This custom calendar considers Sunday as a non-working day, Saturday as a working day with one-eight-hour work shift and all other week days as working days with two-eight-hour work shifts.

Simulation results (see Table 5.0) indicate that the work scope can be completed earliest with the *"24 Hour"* calendar, followed by the *"custom"* calendar defined. Given that the *"24 Hour"* option is not a calendar per se, the *"custom"* calendar would be the most efficient option to complete the work in the shortest time. The modeller can experiment with different resource and shift configurations to obtain results that can be compared to pick a work plan that best suits their needs.

This section demonstrates that the Simphony simulation system fully supports the integration of calendars in simulation models.

```
public static partial class Formulas
{
  //Private fields for the methods to use in their
  calculations
  private static System.Boolean WasWorking = false;
  public static Simphony.Simulation.DiscreteEventEngine
  SimulationEngine;
  public static System.Boolean
  Formula(Simphony.General.Execute context)
   {
    //Set the Initial quantity of dirt to 10,000 CY
    context.Scenario.Floats[0] = 10000.0;
    //Get a Calendar to subscribe to from the scenario
    Simphony.Simulation.Calendar MyCalendar =
    context.Scenario.Calendars["My Custom Calendar"];
    //Create an instance of a calendar entity
    CalendarEntity MyCalendarEntity = new
    CalendarEntity();
//Make a global attribute point to this calendar
entity
    context.Scenario.Objects[0] = MyCalendarEntity;
    context.Scenario.Objects[1] = MyCalendar;
    //Get access to a calendar
context.Engine.SubscribeCalendar(MyCalendarEntity,MyCalend
ar,CalendarEvent);
    //Make reference to the simulation engine
    SimulationEngine = context.Engine;
    return true;
   }
  //Call the method that is related to the calendar events
  public                    static                    void
CalendarEvent(Simphony.Simulation.CalendarEntity e)
   {
    //Check if there is a change in the work state of the
calendar ---> If there is a change,Then do something
    if(e.IsWorking != WasWorking)
     {
      //It is a change from a non-work period to a work
period ---> Resume processing of events
      if(e.IsWorking)
      {
        System.Diagnostics.Trace.WriteLine("A work period
is      begining      now      at      time:      "      +
SimulationEngine.DateNow.DayOfWeek           +",         "+
SimulationEngine.DateNow + " and will end in "
        + e.TimeRemaining.TotalHours + " hours");
        foreach(var entity in e.Entities)
        {
         //Resume only suspended entities
         if(entity.IsSuspended)
         {
          SimulationEngine.ResumeEvent(entity);
          System.Diagnostics.Trace.WriteLine("Work     on
truck #" + entity.Id +" has just been resumed on " +
SimulationEngine.DateNow.DayOfWeek+","+SimulationEngine.Da
teNow);
          }
        }
      }
      else
      {  System.Diagnostics.Trace.WriteLine("A work break
is begining now at time: " +
         SimulationEngine.DateNow.DayOfWeek      +",     "+
SimulationEngine.DateNow + " and will end in "
         + e.TimeRemaining.TotalHours + " hours");
         foreach(var entity in e.Entities)
         {
     //Suspend all entities controlled by this calendar
          SimulationEngine.SuspendEvent(entity);
          System.Diagnostics.Trace.WriteLine("Work      on
truck #" + entity.Id.ToString() +" has
           just      been      suspended      on      "      +
SimulationEngine.DateNow.DayOfWeek + ", " +
           SimulationEngine.DateNow);
          }
        }
      }
      WasWorking = e.IsWorking;}
```

Figure 5: C# Code Snippet for Subscribing to the Calendar in Execute Shown in Figure 4.0

Table 4: Work Periods Used in the Custom Calendar

Day of Week	Work Times
Sunday	-
Monday - Friday	6:00 – 10:00
	11:00 – 15:00
	15:00 – 19:00
	20:00 – 00:00
Saturday	6:00 – 10:00
	11:00 – 15:00

```
public static partial class Formulas
{
  public              static               System.Boolean
Formula(Simphony.General.Execute context)
   {
    //Get the statistics node from the scenario for the
total work days
    Simphony.General.Statistic           S1           =
context.Scenario.GetElement<Simphony.General.Statistic>(
"Total # working days during
    simulation");
    //Get the statistics node from the scenario for the
total non-work days
    Simphony.General.Statistic S2 =
context.Scenario.GetElement<Simphony.General.Statistic>(
"Total # of non-working days
    during Simulation");
    //Get the statistics node from the scenario for the
total # of days from start to
    end of simulation
    Simphony.General.Statistic S3 =
context.Scenario.GetElement<Simphony.General.Statistic>(
"Total # of days from start
    to finish of simulation");
    //Get the calendar from the global attribute --->
So that we can get the working and
    non-working time
    Simphony.Simulation.Calendar   MyCalendar   =      \
(Simphony.Simulation.Calendar)(context.Scenario.Objects[
1]);
     //Collect   the   total   #   of   work   days
S1.Collect(MyCalendar.GetWorkingTime(context.Scenario.St
artDate,
     context.Engine.DateNow).Days);
     //Collect   the   total   #   of   non-work   days
S2.Collect(MyCalendar.GetNonWorkingTime(context.Scenario
.StartDate,
     context.Engine.DateNow).Days);
     //Collect the total # days from start to finish of
simulation
     System.TimeSpan  TS  =  context.Engine.DateNow  -
context.Scenario.StartDate;
     S3.Collect(TS.Days);
     //Trace the finish date of Simulation
     System.Diagnostics.Trace.WriteLine("The      earth-
moving operations has been completed
     on date:" + context.Engine.DateNow);
     //Get the Calendar entity --> So that we can
unsubscribe calendar
     Simphony.Simulation.CalendarEntity
MyCalendarEntity =
(Simphony.Simulation.CalendarEntity)(context.Scenario.Ob
jects[0]);
     //Unsubscribe            the            calendar
context.Engine.UnsubscribeCalendar(MyCalendarEntity);
     return true;
    }
 }
```

Figure 6: C# Code Snippet for Unsubscribing to the Calendar and Computing Work and Non-working Days

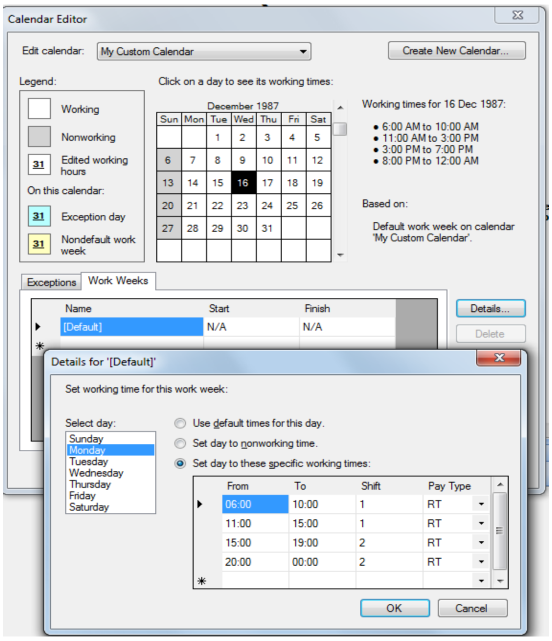

Figure 7: Dialogue for Creating or Editing Calendars in Simphony

Table 5: Experimental Results from Simulation Using Different Calendars

Calendar	Start Date	Finish Date	Total work days	Total non-work days
Standard	July 15, 2013	Aug. 13, 2013	22	7
24 Hours	**July 15, 2013**	**July 22, 2013**	**7**	**0**
Night	July 15, 2013	Aug.10, 2013	6	19
Custom	**July 15, 2013**	**July 27, 2013**	**7**	**5**

4. MODELING TRAFFIC LIGHT CONTROLS IN SIMPHONY

4.1. Problem Description

Simphony is an easy-to-use simulation system for modeling typical discrete event simulation problems regardless of whether they are within the construction domain or not. A sample problem identified from two popular simulation text books by Halpin (1992) and Pritsker (1997) is described here and used for purposes of demonstrating the modeling abilities of the Simphony environment.

One lane of a 500 m section of road is closed off for major repair work. The road comprises two lanes with traffic flowing in opposite directions (east bound traffic and west bound traffic). For this section of road, lights allow traffic to flow for a specified time interval from only one direction. This arrangement is depicted in Figure 8.0.

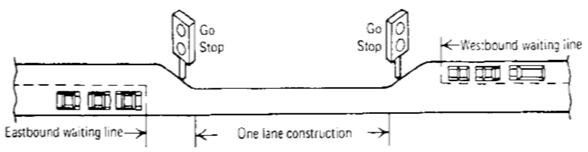

Figure 8: Schematic Layout of Site in the Traffic Light Problem (Halpin and Riggs, 1992)

When a light turns green, the waiting cars start and pass the light every 3 seconds. If a car arrives at the green light when there are no waiting cars, it passes through the light without delay. The car arrival pattern is such that there is an average of 10 seconds between cars from the east direction and 9 seconds between cars from the west direction. A light cycle consists of green for east bound traffic, both red, green for west bound traffic, both red, and then the cycle is repeated. Both lights remain red for 50 seconds to allow cars in transit to leave the repair section before traffic from the other direction can be initiated. The objective is to obtain green times for traffic lights that minimize waiting times for east and west-bound traffic.

4.2. Simphony Models, Discussion and Results

A traffic light cycle is perceived as involving a sequential process in which lights transition through different states (signals) represented by different light colors (*Green* → *All-Red* → *Red* → *Green*). Each traffic light is modelled as a resource so as to provide a convenient link between the traffic light cycle and the traffic flow (permit flows at right time, halt flows and track waiting statistics). An entity is used to loop through the cycle triggering the start and finish of each state. State change is triggered by capture or release of a traffic light resource. Higher priorities are given to the entity flowing within the *"traffic light control loop"* (for the capture of traffic light resources) compared to vehicle entities flowing in the *"traffic flow"* sub-models. The time that the system stays within a given traffic light state is modeled by task elements. *"All-Red"* time is set to 50 seconds and we are to experiment to determine an optimal value of the *"Green"* light times that minimize the waiting time of traffic flowing in both directions.

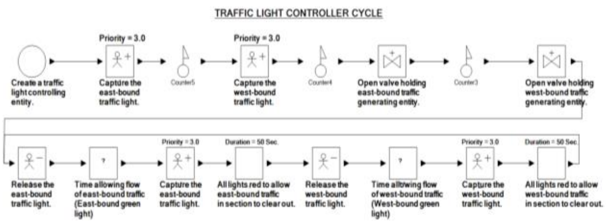

Figure 9: Model Layout of Traffic Light Controller Cycle

A discrete event model was developed in Simphony.NET 4.0 for the traffic system. The constructed model is comprised of 3 sub-models: (1) a traffic light control cycle, (2) an east-bound traffic flow model and (3) a west-bound traffic flow model. Each of these sub-models is discussed in detail. In these sub-models, the traffic lights for the *"east-bound"* and *"west-bound"* traffic lights are modelled explicitly as resources within the Simphony general purpose template. The entities in this model include: the traffic flowing in the east direction, west direction and a flow unit that triggers the traffic light signals (ON/OFF).

Figure 9.0 shows a layout of the sub-model that emulates a typical traffic light cycle. One entity (*"traffic light controller entity"*) is created at the start of simulation which captures the east and west bound traffic lights. Thereafter, the entity triggers opening of valves that were retaining entities created to generate east and west bound traffic, respectively. The traffic controller entity then releases the east-bound traffic resource so that east-bound entities arriving capture this resource and flow through the section. The resource is freed for a specific duration that emulates the time that the traffic light is green after which the traffic light is captured once again by the traffic controller entity (representing the east-bound traffic light turning red). At this point, the traffic controller entity has both traffic light resources in its possession (signaling "all-red" on traffic lights) and is transferred into a task element that holds it for 50 seconds. This 50 second delay mimics the time required by east-bound traffic (caught in the construction zone when the east-bound traffic light turns red) to clear out of this section. The traffic controller entity then flows into an element that releases the west-bound resource and then subsequently into a task element that delays it for a duration equivalent to that for which the west-bound traffic light is green. The west-bound traffic light resource is made available to west-bound traffic entities that were queued or are just arriving, hence, allowing them to flow through the construction zero. Thereafter, the "traffic controller entity" requests for the "west-bound traffic light resource" with a high priority (3.0). It is granted this resource after the current "west-bound traffic flow entity" utilizing it releases it. The "traffic controller entity" will once again have both traffic light resources in its possession and is transferred into an *"all-red"* task element for 50 seconds during which west-bound traffic currently flowing in the construction zero section is expected to clear out. The *"traffic light controller entity"* is then looped back to the start of the traffic light cycle where it resumes with the release of the *"east-bound traffic light resource."*

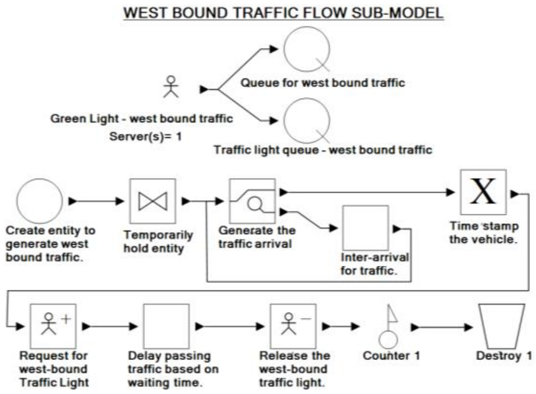

Figure 10: Model Layout for West-Bound Traffic Flow

The east-bound (EB) and west-bound (WB) traffic flow sub-models represent the arrival, queuing and flow of traffic in the east and west directions, respectively. The model layouts (Figures 10 and 11) are identical but involve different resources *("East-Bound Traffic Resource"* and *"West-Bound Traffic Resource"*), waiting files (*"Queue for East-Bound Traffic", "Queue for West-Bound Traffic", "Traffic Light Queue-East Bound Traffic" and "Traffic Light Queue-West Bound Traffic")*, valves, tasks, capture and release elements. In these sub-models, the waiting files for traffic entities are separated from those of the *"traffic light controller entity"* so that the statistics on queued traffic are not distorted.

One entity is created in each sub-model at time zero and held behind a valve control until the "*traffic light controller entity*" has captured the *"East-Bound Traffic Resource"* and *"West-Bound Traffic Resource"* and triggered the valves to open. This entity in each sub-model serves as a *"traffic generating entity."* It is transferred into a *"generate element"* which clones it.

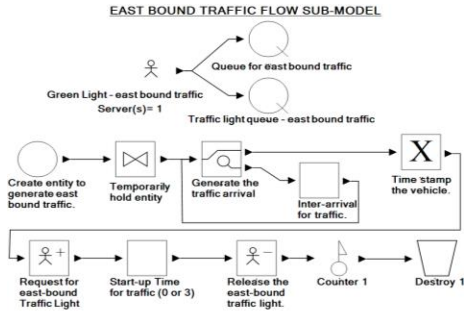

Figure 11: Model Layout for East-Bound Traffic Flow

The entity flowing out of the top point of the *"generate element"* represents an arrival of a vehicle and is routed into an *"execute element."* The cloned entity is transferred out of the bottom output point of the *"generate element"* into a *"task element"* where it is delayed for the inter-arrival duration before being re-routed into the *"generate element"* to release another entity that represents another vehicle arrival. This cyclic process keeps going until the simulation is terminated.

Arriving traffic entities flow through the *"execute element"* where they are time-stamped with the time at which they arrive at the construction zone. Arriving traffic entities then proceed to a capture element where they request their respective traffic light resource. If the traffic light resource is available, the traffic entity proceeds on its journey without delay; otherwise, it is queued until the traffic light resource becomes available. Traffic entities that were queued and are allowed to travel through the construction zone when the light turns green are delayed by 3 seconds as they pass-by the traffic light. These 3 seconds represents start-up time for vehicles moving from a complete stop. This logic is modelled with the *"task element"* using the VB code snippet shown in Figure 12. This was inserted into the formula editor of the duration property for the *"task element."*

After the traffic entity passes by the traffic light, it releases the traffic light resource to the next entity. It then flows through the counter element where the traffic count is registered and then into a "destroy element" where it is removed from the simulation. The flow of traffic entities is halted when the green time is used up (and the *"traffic light controller entity"* captures the traffic light resource).

```
Public Partial Class Formulas
 Public     Shared     Function     Formula(ByVal     context     As
Simphony.Modeling.Task(Of
  Simphony.Simulation.GeneralEntity)) As System.Double
    If context.Engine.TimeNow - context.CurrentEntity.Floats(0)=
     0.0 Then
     Return 0.0
     Else Return 3.0
    End If
 End Function
End Class
```

Figure 13: VB Code for Generating a Delay for a Vehicle Passing a Traffic Light

Simulation settings used are summarized in Table 6.0. These were used because of the stochastic inputs used e.g. the inter-arrivals of traffic. Also, a seed was fixed to ensure that the same sequence of random deviates is used for scenarios that are compared.

To determine the green times to allocate to the east-bound and west-bound traffic, equal arbitrary values were used. This phase of the experiment was used to get the local minimum (waiting time for traffic flowing in each direction). These were different because traffic inter-arrivals in each direction were different. Results from this phase are summarized in Table 7.0 and are plotted in Figure 14.0. The acronyms EB-GT, WB-GT, EB-WT and WB-WT represent east-bound green time, west-bound green time, east-bound waiting time and west-bound waiting time in seconds respectively.

Table 6: Simulation Setting Used for the Traffic Problem

Simulation Setting	Parameter Value
Seed	5,000
Run Count	100
Time Unit	Seconds
Maximum Time	86,400 Seconds (1 day)

Table 7: Phase I Results from Experimenting with the Traffic Model

EB-GT	WB-GT	EB-WT	WB-WT
50	50	6,979.03	10,505.42
60	60	4,089.52	7,890.69
80	80	225.0058	3,186.14
100	100	115.173	344.3368
110	110	110.3978	192.3788
120	120	**109.3351**	154.9445
140	140	111.7418	130.1684
150	150	113.9442	129.5323
160	**160**	116.5949	**128.2344**
180	180	122.5078	132.7623
200	200	128.9449	137.8971
220	220	135.7282	143.9771
240	240	142.6467	150.9405

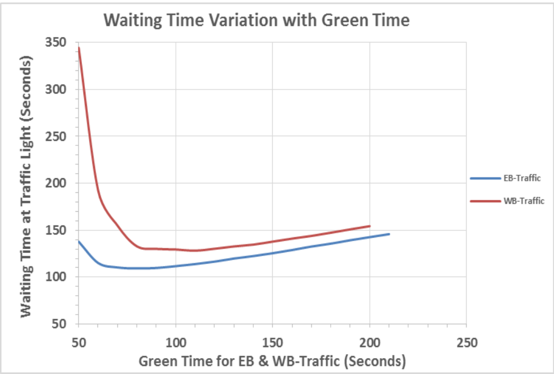

Figure 14: Waiting Time Variation with Green Time

The values obtained from phase I (highlighted in bold in Table 7.0) are used to guide phase II of the experimentation which involves determining the global minimum waiting time for all traffic. Results from this phase are summarized in Table 8.0. Optimal green times were found to be 130 and 140 seconds for east-bound and west-bound traffic respectively.

Table 8: Phase II Results from Experimenting with the Traffic Model

EB-GT	WB-GT	EB-WT	WB-WT
120	160	194.40	97.19
120	150	155.46	101.47
125	160	155.20	100.62
130	160	139.64	104.34
130	155	133.05	106.57
130	150	127.47	109.70
130	**140**	**117.84**	**117.45**

5. CONCLUSIONS

The paper presents a concise overview of simulation, the existing simulation methods, different simulation systems and studies in which simulation has been previously applied within the construction domain.

Simphony is introduced as an example of typical simulation system currently in use, its features discussed and reasons why it remains relevant in the process of defining next generation simulation tools/systems highlighted.

Two practical problems (an earth-moving problem involving shift dynamics and a traffic light problem) modeled in Simphony and experimented with to generate results that can be used to support decision making processes are described to showcase capabilities and features that exist within Simphony.

REFERENCES

AbouRizk, S. M., Ruwanpura, J. Y., Er, K. C., and Fernando, S., 1999. Special purpose simulation template for utility tunnel construction. *Proceedings of Winter Simulation Conference*, 2: 948-955. Dec. 5-8; Phoenix, AZ.

Ahn, C., Rekapalli, P. V., Martinez, J. C., and F. A. Pena-Mora., 2009. Sustainability analysis of earthmoving operations. *Proceedings of Winter Simulation Conference*, pp. 2605-2609. Dec. 13-16, Austin, TX.

Ailland, K., Bargstadt, H., and Hollermann, S., 2010. Construction process simulation in bridge building based on significant day-to-day data. *Proceedings of Winter Simulation Conference*, pp. 3250-3261. Dec. 5-8, Baltimore, MD.

Al-Bataineh, AbouRizk, S. M., and Parkis, H., 2013. Using simulation to plan tunnel construction. *Journal of Construction Engineering and Management*, 139(5): 564–571.

Alvanchi, A., Nguyen, A., and AbouRizk, S. M., 2012. Structural steel fabrication special purpose simulation. *Construction Research Congress*, pp. 1391-1399. May, 19-23, Purdue, IN.

Chehayeb, N. N. and AbouRizk S. M., 1998. Simulation-based scheduling with continuous activity relationships. *Journal of Construction Engineering & Management,* 124(2): 107-115.

Cheng, F. F., Wang, Y. W., Ling, X. Z., and Bai, Y., 2011. A Petri net simulation model for virtual construction of earthmoving operations. *Journal of Automation in Construction,* 20: 181-188.

Chui-Te, C., Tseng-Hsing, H., Ming-The, W., and Hsien-Yen, C., 2011. Simulation for steel bridge erection by using BIM tools. *Proceedings of 28th ISARC,* pp. 560-563. June 29, Seoul, Korea.

Crain, R. C., 1997. Simulation using GPSS/H. *Proceedings of Winter Simulation Conference,* pp. 567–573. Dec. 7-10, Atlanta, GA.

Dori, G., and Borrmann, A., 2011. Automatic generation of complex bridge construction animation sections by coupling constraint-based discrete-event simulation with game engines. *Proceedings of International Conference on Construction Applications of Virtual Reality*. Nov. 3-4, Weimar, Germany.

Dulcy, M. A., and Halpin, D. W., 1998. Simulation of the construction of cable-stayed bridges. *Canadian Journal of Civil Engineering,* 25: 490-499.

Fu, J., 2012. A microscopic simulation model for earthmoving operations. *World Academy of Science, Engineering and Technology*, 67: 218-223.

Hajjar, D., and AbouRizk, S. M., 1999. Simphony: An environment for building special purpose construction simulation tools. *Proceedings of Winter Simulation Conference,* 2: 998-1006. Dec. 5-8, Phoenix, AZ.

Halpin, D. W., 1973. *An investigation of the use of simulation networks for modeling construction operations.* PhD. Thesis, University of Illinois, Urbana-Champaign, III.

Halpin, D.W., and Riggs, L.S., 1992. *Planning and Analysis of Construction Operations.* John Wiley & Sons, Inc.

Hu, D., and Mohammed, Y., 2010. State-based simulation mechanism for facilitating project schedule updating. *Construction Research Congress,* pp. 369-378. May 8-10, Banff, AB.

Ioannou, P. G., and Martinez, J. C., 1996. Comparison of construction alternatives using matched simulation experiments. *Journal of Construction Engineering and Management*, 122(3): 231-241.

Liu, Y. and Mohamed, Y., 2008. Multi-agent resource allocation (Mara) for modeling construction processes. *Proceedings of Winter Simulation Conference*, pp. 2361-2369. Dec. 7-10, Miami, FL.

Liu, H., Siu, M. F., Hollerman, S., Ekyalimpa, R., Lu, M., AbouRizk, S., and Bargstadt, H., 2012. Simulation of mobile falsework utilization methods in bridge construction. *Proceedings of Winter Simulation Conference, pp. 1-13.* Dec. 9-12, Berlin, Germany.

Lu, M., 2003. Simplified discrete-event simulation approach for construction simulation. *Journal of Construction Engineering and Management*, 129(5): 537-546.

Martinez, J. C., 1996. *STROBOSCOPE – State and resource based simulation of construction processes*. PhD thesis, University of Michigan.

Marzouk, M., El-Dein, H. Z., and El-Said, M., 2007. Application of computer simulation to construction of incremental launching bridges. *Journal of Civil Engineering and Management*, 13(1): 27-36.

Marzouk, M., and Moselhi, O., 2002. Simulation optimization for earthmoving operations using genetic algorithms. *Journal of Construction Management and Economics*, 20(6): 535-543.

Mohsen, O. M., Knytl, P. J., Abdulaal, B., Olearczyk, J., and Al-Hussein, M., 2008. Simulation of modular building construction. *Proceedings of Winter Simulation Conference*, pp. 2471-2478. Dec. 7-10, Miami, FL.

Pritsker, A. A. B, O'reilly, J. J., and Laval, D. K. , 1997. *Simulation with visual SLAM and AweSim*. New York, NY: John Wiley & Sons, Inc.

Rekapalli, P. and Martinez, J., 2011. Discrete-event simulation-based virtual reality environments for construction operations: Technology introduction. *Journal of Construction Engineering and Management,* 137(3): 214–224.

Sadeghi, N. and Fayek. A. R., 2008. A framework for simulating industrial construction processes. *Proceedings of Winter Simulation Conference,* pp. 2396-2401. Dec. 7-10, Miami, FL.

Song, L., and AbouRizk, S. M., 2003. Building a virtual shop model for steel fabrication. *Proceedings of Winter Simulation Conference,* pp. 1510-1517. Dec. 7-10, New Orleans, LA.

Song, L., and AbouRizk, S. M., 2006. Virtual shop model for experimental planning of steel fabrication projects. *Journal of Computing in Civil Engineering,* 20(5): 308-316.

Sun, C.-T., Wang, D.-Y., & Chang, Y.-Y., 2013. Effects of thinking style on design strategies: Using bridge construction simulation programs. *Educational Technology & Society,* 16 (1): 309–320.

Touran, A. and Asai, T., 1987. Simulation of tunneling operations. *Journal of Construction Engineering and Management*, 113(4): 554–568.

Wang, P., Mohamed, Y., Abourizk, S., and Rawa, A., 2009. Flow production of pipe spool fabrication: Simulation to support implementation of lean technique. *Journal of Construction Engineering and Management*, 135(10): 1027–1038.

Zhang, H., Tam, G. M., and Shi, J. J., 2002. "Simulation-based methodology for project scheduling." *Journal of Construction Management and Economics,* 20(8): 667-678.

Zhou, F., AbouRizk, S. M., and Fernando, S., 2008. A simulation template for modeling tunnel shaft construction. *Proceedings of Winter Simulation Conference*, pp. 2455-2461. Dec. 7-10, Miami, FL.