Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

SIMPHONY – AN INTEGRATED ENVIRONMENT FOR CONSTRUCTION SIMULATION

Simaan AbouRizk
Yasser Mohamed

Department of Civil and Environmental Engineering
220 Civil and Electrical Building
University of Alberta
Edmonton, AB T6G 2G7, CANADA

ABSTRACT

This paper discusses Simphony as an integrated environment for building special purpose simulation tools for modeling construction systems. Simphony provides various services that enable the developer to easily control different behaviors in the developed tool such as simulation behaviors, graphical representation, statistics, and animation. These services allow building flexible and user-friendly tools in a relatively short time. The developed tools (templates) then provide the building blocks that a user (construction engineer) can use to create simulation models for different construction domains without the need for a deep background in simulation techniques. The services available for the developer are discussed and examples of their use are illustrated. Samples of the tools developed with Simphony are also reviewed to highlight some of their features.

1 INTRODUCTION

Simulation is one of the powerful techniques for supporting the decision making process for construction management. An accurate modeling of a construction process can help the development of better alternatives and optimization of the involved resources. However, the use of simulation techniques in the construction industry is believed to be minimal. Complexity of simulation methodologies and lack of deep simulation knowledge among industry personnel are among the main causes of weak utilization of simulation in construction management.

Special Purpose Simulation (SPS) approach was identified as a means for facilitating adoption of simulation by industry. This approach enables a practitioner who is knowledgeable in a given domain, but not necessarily in simulation, to easily model a project within that domain using visual modeling tools that have a high degree of resemblance to the actual construction system. (AbouRizk and Hajjar 1998)

Developing a stand-alone SPS tool requires a relatively large initial investment that would, in some cases, prevent the industry from considering such a tool. To overcome this obstacle, Simphony was developed as a complete SPS tool development and utilization environment.

Tool developers can use Simphony to implement highly flexible simulation tools that support graphical, hierarchical, modular and integrated modeling. It provides them with a comprehensive and complete tool definition, compilation and testing environment.

Tool users can then use the developed tools (templates) to build simulation models in an intuitive and user friendly manner (Hajjar, and AbouRizk 1999).

This paper presents an overview of the tool development process in Simphony and illustrates examples of the available services that facilitate developing easy-to-use simulation tools in a relatively short time. It also presents some of the tools developed with Simphony and their capabilities.

2 BUILDING SIMULATION TOOLS WITH SIMPHONY

Any simulation model built in Simphony is composed of a number of instances of modeling elements that the user creates on screen and links together with relationships. Each of these modeling elements has its own behaviors in response to different events. A collection of these elements that belong to the same construction domain and are designed to work together in a model is referred to as a template.

Creating a new template using Simphony involves two main phases the design phase and the implementation phase.

The design of a template requires a complete understanding of the domain or construction system that will be modeled using this template. Based on that understanding the developer can decide the elements to include in the template and the different behaviors of each element. Several factors have to be considered during this

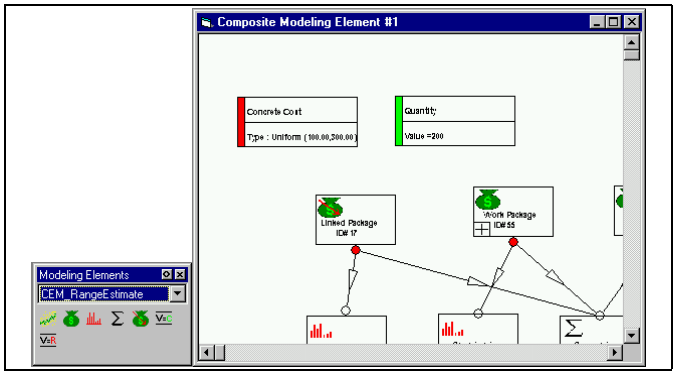

Figure 1: Modeling Elements and Relationships

phase in order to build a flexible and detailed yet user-friendly and easy-to-use template. For example, a large number of elements with small functionality for each and comprehensive interaction between each other would result in a flexible template capable of modeling a wide spectrum of cases within the domain. However, the template user (engineer) may find such template hard to use and may not need such amount of details.

Once the design is complete, the developer can then implement it. The implementation involves the creation of new modeling elements in Simphony using the *Template Manager* and the customization of the behaviors for each of these elements. The different behaviors of an element are produced by writing code in the form of event handlers that respond to the various events. Simphony generates these events in response to either user or system actions.

The following table illustrates the different element behaviors that the developer can customize with a brief description for each.

3 SIMPHONY SERVICES

To speed up and facilitate the development process Simphony provides all standard and commonly used SPS structures and routines in the form of libraries (services) which are easily accessible by the developer.

A service is provided through a collection of class properties and methods that the developer can utilize during the implementation of the event handlers. The different services in Simphony are closely tied with the element behaviors shown in Table 1 and enable the customization of these behaviors with great ease.

The following sections discuss examples of the available services in Simphony and their functions and utilization.

3.1 Simulation

The simulation service in Simphony provides support for discrete-event simulation including event scheduling, next-event polling, and queues management. More advanced

Table 1: Customizable Element Behaviors

Behavior	Description
Geometric Attributes	Geometric attributes are mainly used for model layout purpose. They define the two dimensional position of the modeling element in relation to other elements. This information is typically used simply for graphical representation purposes.
User Attributes	User attributes include parameters and outputs. Parameters are what engineers will be manipulating to change the properties of the modeling element. Outputs provide exposed features of the modeling element either for the engineer to examine (Performance Indicators), or to be used as inputs to other element parameters.
Relationships	Relationships between modeling elements is used to define the logic of the simulation model and flow path of entities.
Hierarchy	The concept of a hierarchy is supported mainly through the ability of any element to access its parent's as well as its children's properties.
Simulation	Simulation behavior defines the resources, files, events and entities of a given element and how each simulation event is handled.
Statistics	Statistical collection is defined at the modeling element level. Examples of statistics are resource utilization, queue lengths, and cycle times.
Planning	The planning behavior defines how a given modeling element transforms its simulation results into a project plan that includes a schedule, production and revenue forecast, resource utilization and costs.
Graphical User Interface	This behavior includes graphical representation, which determines what the modeling element looks like to the engineer. The other aspect of this behavior is graphical manipulation, which defines how an engineer can manipulate geometrical attribute, user attribute, and relationship information.
Animation	This behavior defines the element's role in the animation scenario if one is produced after simulation.

methods are also available to support event cancellation and resource preemption.

This service is provided through a class called "CFCSim_ModelingElementInstance". Examples of the

available methods and properties in this class are shown in the following table.

Table 2: Sample Methods and Properties for Simulation Service

Method/Property	Description
AddEvent	Declares a simulation event that can be scheduled during simulation.
AddFile	Adds a simulation file. The file can be declared as *stack*, *queue* or *list.*
ScheduleEvent	Schedule an event for processing after certain simulation-time units.
CloneEntity	Creates a copy of an entity with all the attributes of the original.
RequestResource	Requests a resource and implicitly queues the requesting entity if the resource is not available.

The following example illustrates the use of the "ScheduleEvent" method for scheduling a discrete event. The method requires three input parameters: 1) a reference to the entity that will trigger the event, 2) the name of the simulation event, and 3) the time period after which the event is scheduled starting from the current time. This method will mostly be used in the simulation event handlers which are triggered on initializing a simulation run (*OnSimulationInitializeRun*), or processing an event (*OnSimulationProcessEvent*). The example utilize the "ScheduleEvent" method in the "OnSimulationInitialize-Run" event handler to kick off the simulation events.

```
Public Sub Example_OnSimulationInitializeRun(ob
as CFCSim_ModelingElementInstance, RunNum as
Integer)
   Dim Customer as CFCSim_Entity
   Set Customer = ob.AddEntity
   Ob.ScheduleEvent Customer, "Customer_Arrive",
   sampler.Expntl(10)
End Sub
```

The above code first declares a variable of type CFCSim_Entity to hold on a reference to an entity representing a customer. The second line utilizes the AddEntity method to create a new entity, obtain a reference to it, and assign the reference to the declared variable. The Schedule-Event method is then used to schedule the first event in the simulation run which, in this case, represent a customer arrival. The last parameter of the method is set through a call to another service in Simphony that enables sampling random numbers from different types of distribution by defining the type and parameters of the required distribution. The sampling in the above example is done from an exponential distribution with a mean of 10.

3.2 Statistical

This service provides support for collecting statistics on both intrinsic and non-intrinsic types of data. Standard statistics include average, standard deviation, minimum and maximum. Statistical collection is done in Simphony through the concept of a "statistic". A statistic must be declared on creating a new instance of an element before any observation values are collected during simulation. To declare a new statistic, the developer can use the "AddStatistic" method as part of the "OnCreate" event handler of the element. The following code shows example of possible use of the statistical services.

```
Public Function Example_OnCreate(ob as
CFCSim_ModelingElementInstance, x as Single, y
as Single) as Boolean
...
ob.AddStatistic "CycleTime", "Truck Cycle
Time",False,True
...
End Function

Public Sub Example_OnSimulationProcessEvent(ob
As CFCSim_ModelingElementInstance, MyEvent As
String, Entity As CFCSim_Entity)
  ...
  Select Case MyEvent
  Case "CollectStat"
    Ob.stat("CycleTime").Collect SimTime-
    entity("StartTime")
  ...
 End Sub
```

The first part of the above code shows use of the AddStatistic method in the OnCreate event handler. The first parameter of the method is a unique string for identifying the statistic. The second is the description that will be used by Simphony when displaying the results to the user. The third parameter is to define if the collected observation should be treated as intrinsic (time of collection will be considered) or non-intrinsic. The last one is to define if Simphony should fully track the observations to produce a graph output for the statistic or not.

The second part of the code shows the collecting of observation for the CycleTime statistic as part of an event in the OnSimulationProcessEvnet event handler. The collection is done by defining the required statistic and calling the collect method for it while passing the value of the observation to the collect method. In this case, the cycle time value is the difference between the current simulation time (*SimTime*) and the time when the entity started its cycle which is held in an entity attribute called "StartTime".

The following figure shows the output for a statistic that is fully tracked.

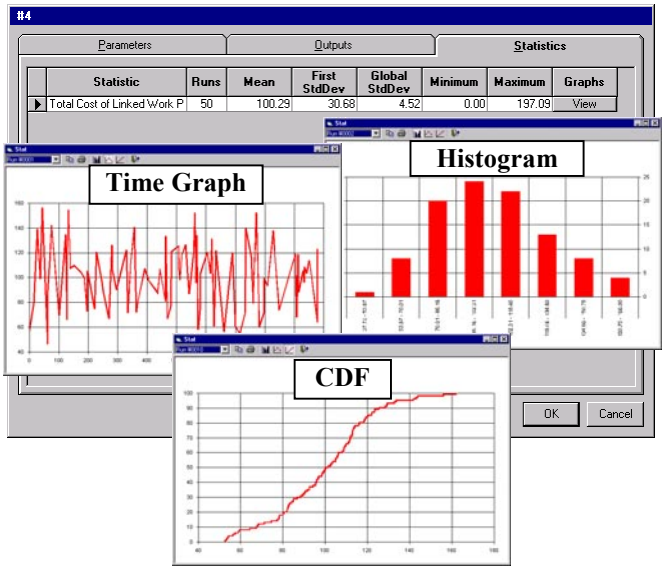

Figure 2: Output of a Fully Tracked Statistic

3.3 Tracing

The tracing service provides a general-purpose mechanism for relaying information to the engineer which could include simulation trace results, error messages, and integrity errors. Simphony also uses this service internally to inform the engineer in case a run-time error in detected in the developer's code.

The tracing service represents a powerful debugging tool for both the developer and the user. During development, the developer can embed trace messages at key points to verify the execution sequence of the code. Once the developed tool is functioning as required, trace messages can then be embedded to enable the user track the flow of simulation events in a model to make sure the model is built properly or to get better understanding of the interaction between the elements in the model during simulation.

The developer can produce trace messages by making a call to the "Trace" method as shown in the following code.

```
Tracer.Trace "The message shown to the user",
"Trace  category1", "Trace category2", "Trace
category3"
```

The following figure shows a typical trace message window after running the simulation.

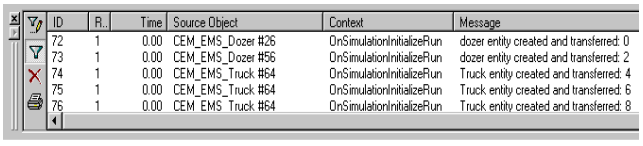

Figure 3: Trace Messages Window

3.4 Animation

Animation service provides the users with extra explanation and insight of how a simulation model works. The developer can associate animation screens with one or more of the modeling elements. After simulation ends, each screen will display a trail of animated objects that maps the sequence of events and changes in the model. The developer has the control of what objects to display, the graphical representation of them, the required movement or modification in response to certain events during simulation. A number of methods and properties are provided through the animation classes for controlling these animation behaviors.

An example of the use of animation is illustrated in the following code. The objective in the example is to move an icon to represent a truck movement over a road.

```
Public Sub Road_OnSimulationInitializeRun(....)
 ...
 ob.Parent("Screen").Reference.addPathLine(ob.I
 D & "Path",ob.ConnectionPoints("c2").x,
 ob.ConnectionPoints("c2").y,
 ob.ConnectionPoints("c2").RelationsTo(1).dstCo
 nnection.x,
 ob.ConnectionPoints("c2").RelationsTo(1).dstCo
 nnection.y)
 ...
 ob.Parent("Screen").Reference.addBitmap(ob.ID
 & "TruckEmpty", GetImage(ob,"Truck"& Direction
 & "Empty.bmp"))
 ob.Parent("Screen").Reference.addBitmap(ob.ID
 & "TruckLoaded", GetImage(ob,"Truck"&
 Direction & "Loaded.bmp"))
 ...
End Sub

Public Sub Road_OnSimulationProcessEvent (....)
...
Case "StartTravel"
...
 ob.Parent("Screen").Reference.pathObj(ob.ID &
 "Path",ob.ID & "TruckEmpty",ob("InstNum"))
 ob.Parent("Screen").Reference.startPath(ob.ID
 & "Path",ob("InstNum"),SimTime)
...
Case "FinishTravel"
 ob.Parent("Screen").Reference.endPath(ob.ID &
 "Path",
 ob("InstAssociation").Collection(CStr(Entity.I
 D)), SimTime)
End Sub
```

The first part of the above code is inserted in the event handler triggered on initializing the simulation (*OnSimulationInitialize*). The code handles the creation of a new path in the animation screen of the parent element. The path is created using the connection point coordinates of the current element in addition to the elements connected to it. Within the same event handler, a new animation object is also created to represent the moving truck. Different bitmap images were chosen to graphically represent the truck in the empty and loaded situations.

The second part of the code is embedded in the event handler responsible for processing the simulation events (*OnSimulationProcessEvent*). When a "StartTravel" event is processed, the path is associated with an instance of the truck animation object and then the "StartPath" method is used to begin moving the object along the path at a given simulation time. In the same way, when a "FinishTravel" event is reached, the "EndPath" method is used to end the movement of the truck along the animation path.

The following figure shows part of the animation screen for the previous example.

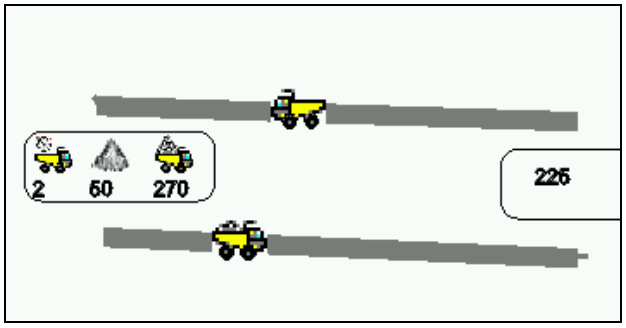

Figure 4: Animation of Truck Travel/Return Process

4 SIMPHONY TEMPLATES

The services provided by Simphony as a development environment allows for developing flexible and user-friendly simulation tools in a relatively short time. These services gives the developer full control over a wide range of element behaviors. Although Simphony is intended for developing SPS tools, its capabilities enabled developing general-purpose tools with full functionality. Samples of the developed tools are introduced in the following sections to highlight some of the features that can be easily accomplished in a tool developed with Simphony.

4.1 General Purpose Simulation Tools

Two templates were successfully developed in Simphony that can be used for general-purpose modeling. The first one is referred to as the "common" template. It provides basic constructs that can be used to model a system using process interaction concepts. The common template features most of the required functions for general purpose modeling that could be found in stand-alone general-purpose simulation software. The use of this template requires the user to have background in simulation techniques. Many of the modeling elements in the template can be used in conjunction with elements from other templates to add certain behaviors to the model.

The template includes elements for handling hierarchical modeling, entity creation and routing, resources, statistics, activities, and tracing. The following table briefly describes the function of some elements in the template.

Table 3: Elements of the Common Template

Element	Description
Composite	The composite element is used to build sub-models inside the main model. The user can create other elements as children inside a composite element and link them to higher-level elements through *InPort* and *OutPort* elements.
Conditional branching	This element enables routing entities into two different branches based on a condition associated with the element.
Probabilistic Branching	This element enables routing entities into a number of different branches based on a probability associated with each branch.
Resource Handling Elements	These are a number of elements for managing resources in the model. They allow declaring resources and waiting files, requesting, and releasing of the declared resource. Multiple or single resource requests in addition to prioritized queuing are allowed.
Create Entities	This element creates new entities with the number, start time and time intervals specified by the user and transfer them out through its output connection point.
Set Entity Attributes	Assigns values for new or existing attributes of entities passing through it.
Consolidate	The consolidate element helps managing the number of entities flowing through it by either consolidating or cloning them.
Execute	This element enables the execution of a user written code during the simulation to perform any function that is not supported by the elements in the template.
Statistics Elements	A number of elements that help declaring and collecting statistics at key points in the model.
Task	This element represents a normal task that requires duration to perform.
Trace	This element enables producing trace messages at selected points to check the integrity of the model.

The second general purpose template enables the user to build models based on the CYCLONE methodology. It supports the standard CYCLONE elements (i.e. queue, normal, combi, generate/consolidate) in addition to elements for supporting hierarchical modeling and probabilistic branching. The hierarchical feature in the templates allows embedding CYCLONE models in other models created by different templates.

The following figure shows the icons of the modeling elements in the two templates from which the user would select to create a model.

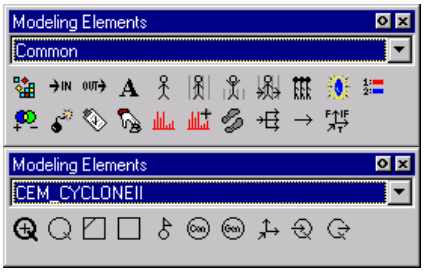

Figure 5: Icons of the Common and Cyclone Elements

Both templates illustrate the capabilities of Simphony for developing highly flexible simulation tools that can accurately model complicated systems. Furthermore, they validate the effectiveness of Simphony as a developing environment that reduces the development time by a remarkable amount.

4.2 Special Purpose Simulation Tools

Special Purpose Simulation is a proven principle that can lead to the effective transfer of simulation knowledge to the construction industry. The development of Simphony was originally motivated by the need for an environment that tailors to the needs of both novice and advanced simulation tool developers and users (Hajjar and AbouRizk 1999).

Several SPS templates have been developed in Simphony for modeling different construction processes. Some of these templates are introduced in the next sections.

4.2.1 Dewatering Template

The dewatering template tackles one of the common problems in construction work. The need to handle subsurface water is always encountered during and after construction. During construction, the removal of water from working areas is necessary to provide workers and equipment with better working conditions. Construction dewatering is not an easy task to achieve, especially when excavations extend more than a few feet below groundwater level. In these cases, open ditches are not a practical solution, and well-point systems or deep wells are normally used.

One of the main problems associated with construction dewatering using deep wells and well-point systems is defining the best possible well configurations that result in the least pumping effort, and therefore the lowest construction costs.

The dewatering template was developed to allow for the graphical modeling and analysis of such operations. Building the dewatering model requires the following information:

- site coordinates,
- pump wells layouts
- excavation area dimensions and depth,
- original water table level, and
- aquifer properties (i.e. confined or unconfined aquifers, layers, permeabilities)

Most of the information is entered in a graphical form that is later on translated into input parameter to the simulation engine.

After running the simulation, the predicted water table level can be observed through user defined observation points, cross-sectional views or 3D graphs.

This template efficiently utilizes the graphical services provided by Simphony and gives a good example of their capabilities. Figure 6 shows the model layout and graphical output of the simulation.

4.2.2 Earth Moving Template

The earth moving template is a special purpose simulation tool for the design and analysis of earthmoving operations. It is aimed at providing a flexible and cost effective method for construction managers interested in optimizing their earthmoving production.

The template models loading and spreading operations, plus complex hauling and interfering traffic patterns. It provides a flexible tool for experimenting with various alternatives. The planner specifies pertinent information, such as the road conditions, the amount of earth to be moved, traffic delays, and the equipment and its properties to be used. Different scenarios can be built and examined by using different combinations of trucks, dozers, and excavators in addition to different work cycles and hauling roads grades in order to optimize the utilization of the equipment fleet and the total project time.

This template illustrates some of the features that could be helpful for the developer and the user. The hierarchical modeling services are well utilized in the template to allow modeling the operations in the source and placement areas through a lower hierarchical level inside the source and placement elements. A separate model is constructed at that level to map the preparation/ loading or dumping/spreading operations. A third hierarchical level is also created to define the preparation and spreading operations.

Integration between different templates in the same simulation model is a useful feature that is also presented in this template. This advanced feature can greatly expand the modeling capabilities of the templates if handled with

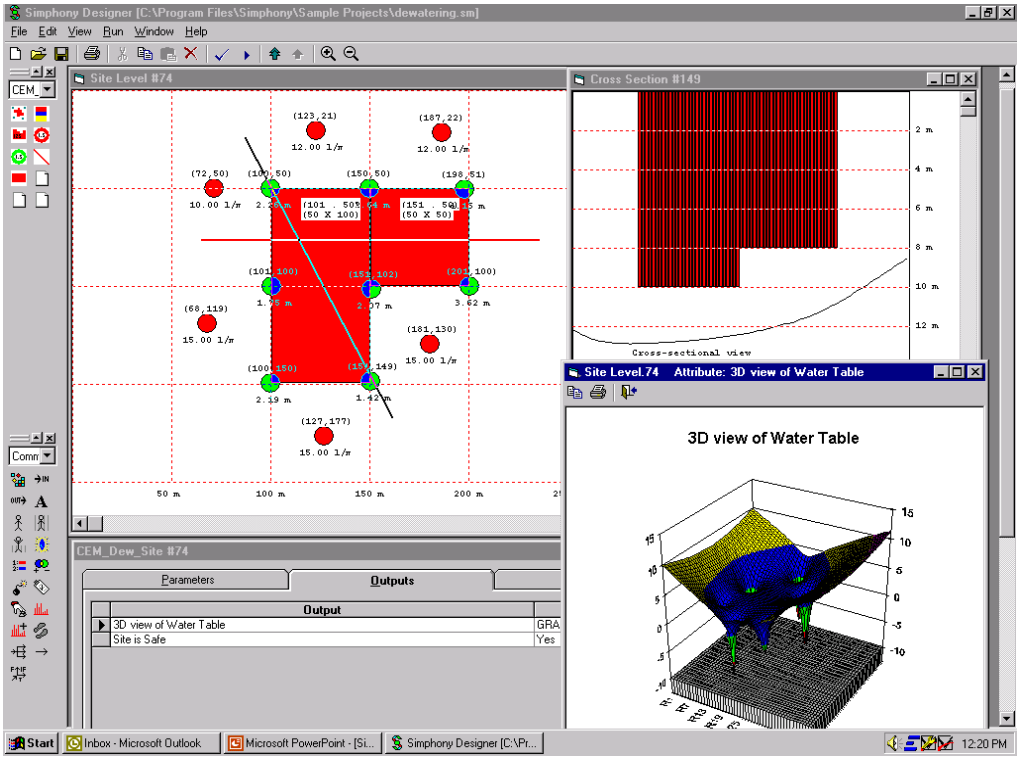

Figure 6: Model Layout and Graphical Output of the Dewatering Template

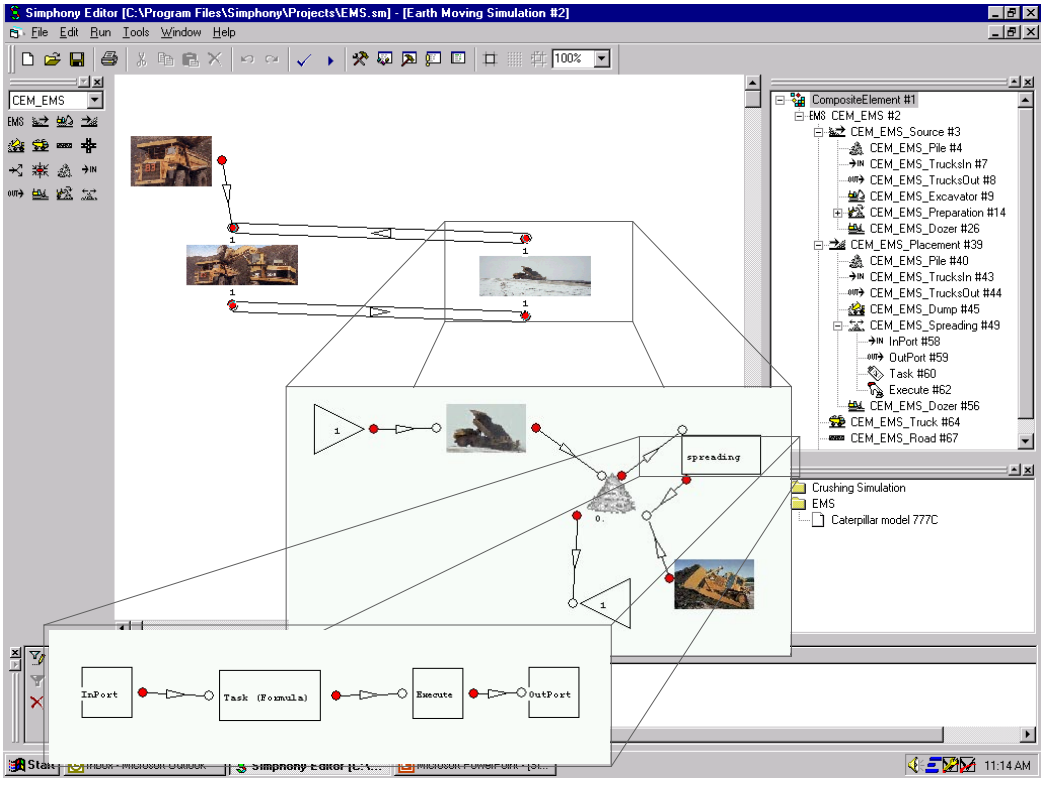

Figure 7: Hierarchical Modeling and Integration Features in the Earth Moving Template

care. In the earth moving template, the preparation and spreading operations are modeled using elements from the common template to account for the expected delay in these processes.

Figure 7 shows the hierarchical modeling and integration between the earth moving and common template.
A third feature of the earth moving template is the link to external databases to retrieve data for some modeling elements. The parameters of the hauling trucks (e.g. travel and return speeds) in the template are retrieved from a database that maintains the data of the truck fleet of the contractor. The user has the option to choose a model from the ones available in the database to define the type for a truck element. Figure 8 shows the list box that is generated based on a database query.

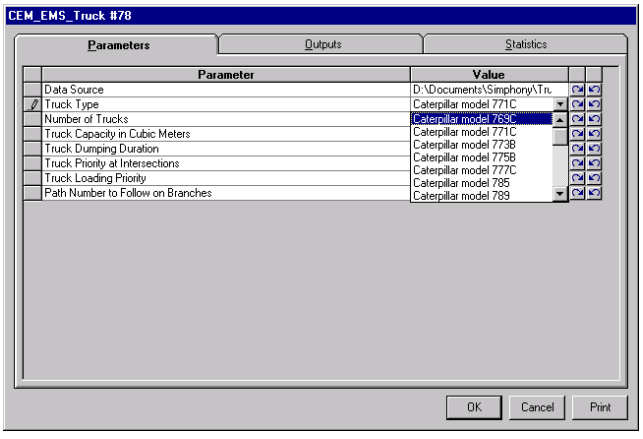

Figure 8: Selecting Truck Type from a Database

5 CONCLUSIONS

This paper discussed Simphony as an integrated environment for developing Special Purpose Simulation Tools. The services provided by Simphony allow the developer to easily control a wide range of modeling elements behaviors which results in building flexible and user-friendly tools in a relatively short time.

The tool development process is overviewed and examples of the services available for the developer for customizing different element behaviors are illustrated. Samples of the templates developed using Simphony are also presented to highlight some of the powerful features that a developer can easily include in a template.

ACKNOWLEDGMENTS

This work was funded by a number of construction companies in Alberta and the Natural Science and Engineering Research Council of Canada under grant number IRC-195558/96.

REFERENCES

AbouRizk, S., and D. Hajjar. 1998. A framework for applying simulation in construction. *Canadian Journal of Civil Engineering*. 25(3): 604-617.

Hajjar, D., and S. AbouRizk. 1999. Simphony: An Environment for Building Special Purpose Construction Simulation Tools. In *Proceedings of the 1999 Winter Simulation Conference*, ed. Phillip A. Farrington, Harriet Black Nembhard, David T. Sturrock, and Gerald W. Evans. 998-1006. Phoenix, Arizona.

Hajjar, D., Y. Mohamed, and S. AbouRizk. 2000. Creating Special Purpose Simulation Tools with Simphony. In *Proceedings of Construction Congress VI,* ed., Kenneth D. Walsh, 87-96. Orlando, Florida.

Simphony Documentation. 2000. Internal Report. NSERC/Alberta Construction Industry Research Chair. University of Alberta.

AUTHOR BIOGRAPHIES

SIMAAN ABOURIZK is a Professor in the Department of Civil Engineering at the University of Alberta. He received his BSCE and MSCE in Civil Engineering from Georgia Institute of Technology in 1984 and 1985, respectively. He received his Ph.D. degree from Purdue University in 1990. His research interests focus on the application of computer methods and simulation techniques to the management of construction projects. His e-mail address is <abourizk@civil.ualberta.ca>.

YASSER MOHAMED is a Ph.D. candidate in the department of Civil Engineering at the University of Alberta. He received his BSCE and MSCE in Civil Engineering form Zagazig University, Egypt, in 1990 and 1996. His research interests are focused on decision support of construction management using simulation techniques. His e-mail address is <yaly@ualberta.ca>.