Interfacing the ESL Simulation Language to the Virtual Test Bed

John G Pearce
ISIM International Simulation Limited
161 Claremont Road
Salford, M6 8PA, UK
mailto:johnpearce@isimsimulation.com

Keywords: Virtual Test Bed, VTB, European Simulation Language, ESL, Continuous System Simulation

Abstract

A preliminary investigation into the design of an interface between the ESL Simulation Language and the Virtual Test Bed (VTB) is described. It is proposed to use a standard feature of ESL that allows an ActiveX COM object to be generated from an ESL *embedded segment* structure. The COM object may be accessed form a VTB ESL wrapper entity. An initial limited functionality wrapper has been written and, using a simple single-input single-output model, provides proof of concept. In preparation to test a fully functional interface, an ESL simulation of an autonomous mobile robot simulation has been adapted and validated using a Matlab program. Output from the robot simulation has been successfully used to generate a three dimensional animation of the robot walking over a terrain using the VTB Visualization Extension Engine (VXE).

1. INTRODUCTION

The Virtual Test Bed (VTB) is a software environment for developing simulations of large scale multidisciplinary dynamic systems. It allows alternative designs to be analysed and tested before being committed to manufacture. The main application that is driving the development of the VTB is a need to model advanced power systems for navy ships. In such systems there are many different energy generation and storage devices including nuclear, fuel cells and gas turbines. The distribution networks are also of unconventional design having dc power busses and high numbers of interconnections that can be rapidly reconfigured. Constructing complete coherent simulations of such large scale systems, involving widely differing technologies poses a serious challenge. Each discipline group will use their preferred simulation tool to model their part of the whole system. The VTB aims to satisfy this challenge by providing a common platform in which component models developed by different teams using different tools can be merged. The VTB also includes the Visualization Extension Engine (VXE) for interactive 3D display of simulation data.

ESL [Pearce and Crosbie 2000] is an advanced high-level simulation language for modelling large-scale systems from a variety of disciplines. ESL is an acronym of the *European Simulation Language* (originally European Space Agency Simulation Language) and comprises two components: the language itself and a graphical user interface - the Integrated Simulation Environment (ISE). ESL is a continuous system simulation language and is used for modelling dynamic systems which are usually described by ordinary and partial differential equations. ISE provides the environment from which all stages of the simulation process can be managed. The software was developed mainly through a series of contracts with ESTEC - the European Space Research and Technology Centre - part of ESA with additional support from various industrial simulation consultancy activities.

This paper describes the progress made so far in a project to interface ESL to the VTB so as to allow ESL models to be used within VTB simulations.

2. BACKGROUND TO ISIM AND ESL

The ESL project began in 1979, as a research contract between ESTEC and Salford University Business Services Limited (SUBSL) - the commercial wing of the University of Salford. The objective of this contract was to evaluate simulation algorithms for parallel computer architectures. The main outcome of the contract was the specification of a new simulation language which included a parallel processing capability. The specification also included other novel features such as a hierarchical submodel structure and special statements for describing discontinuous functions. There followed a series of ESTEC contracts to implement and enhance the language, and so ESL was born. Each contract added some new feature or functionality leading to an advanced robust simulation language which was used for a number of years by ESTEC for modelling aspects of satellite and spacecraft systems.

1986 saw the establishment by the late John Hay of ISIM Simulation as a division of SUBSL with the express purpose of marketing ESL and simulation expertise. Several UK and European companies began using ESL.

The company, ISIM International Simulation Limited, was founded in 1992 and development of ESL continued with the addition of a graphical user interface, allowing diagrammatic model input.

In 1996 earlier links between ISIM and Cogsys Limited (a Salford based software engineering company) lead to collaboration on the design of the new graphical interface (ISE) and improved precision. ESL 7.0, was launched in 2000 at the WMC in San Diego [Pearce and Crosbie 2000] and the current version, ESL 8.0 was released in 2004. There is also a limited Linux release of the ESL core language programs but excluding the graphical user interface and plotting.

ESL was originally developed for the space industry, some examples of space related applications are:

Giotto – simulation of the control system for the antenna de-spin mechanism of the Harley's comet probe.
HST - a study of thermally induced vibrations as the telescope solar panels pass into and out of eclipse.
ERS - a major simulation of satellite based batteries with the objective of on-line optimization of power usage during charge and discharge cycles [Hay 1987].
XMM, *MOLISOVA*, Artemis - an on-going series of uses of ESL to simulate the dynamic parts of test environments for validating on-board satellite software [Bonillo et al 2000], [Holliday 2000].

However, ESL is a general purpose tool and not restricted to the space sector. Some examples of recent non-space applications include:

Off-shore Gas-Rig training Simulator - here ESL provided the underlying dynamic model for a training simulator. The ESL model was embedded in a COGSYS [Davison and Kraft 1990] program which managed the training scenarios and graphics.
Gas Compressor Station Simulation - an on-line real-time simulation for validating the compressor control systems during commissioning [Kraft and Pearce 2000].
Rapid Gravity Water Filter simulation - a customised water treatment works simulation tool for optimising filter bed management [Pearce et al. 2000], [Appleton et al].

3. THE ESL SIMULATION TOOL

The ESL Language itself is a traditional continuous system simulation language (CSSL) and has the following particular features:

Robustness - "a simulation engine that runs forever"
Extensive model "correctness" checking;
Accurate discontinuity treatment;
Vector and matrix arithmetic;
Submodel concept;
Parallel segments (distributed simulation);
Embedded simulation, ActiveX COM interface;
Real-time capability.

An ESL program may be run in an *interpretive* mode or, for faster execution *translated* and compiled, as shown in Figure 1.

The Interactive Simulation Environment (ISE) provides a graphical interface for managing each stage of the simulation activity. This includes a graphical editor for block diagram model descriptions. Textual ESL code may also be used where appropriate – for example, to describe highly non-linear elements. The simulation can then be executed immediately through an interpreter, or further translated, compiled and built into an executable program. In either case, execution is managed by an interactive control panel which provides run-time control of the simulation. All program variables and parameters can be accessed from the control panel when running the simulation. Graphical and tabulated output on the block diagram can be specified through the use of special simulation display elements or alternatively from a versatile display manager window. Run time commands and output specifications can be logged to a driver file and used at a later time to repeat simulation scenarios.

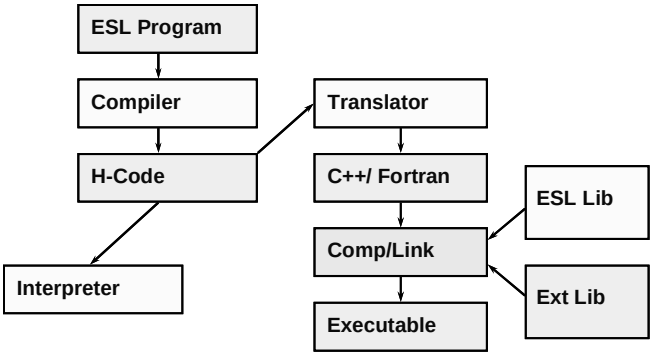

Figure 1 ESL execution routes

4. ESL – VTB INTERFACE

The question to be addressed is how to integrate ESL into the VTB, or, more precisely, how to enable ESL models to be included in a VTB simulation. First it is necessary to understand the different ways in which VTB models (or model entities) are created. Model entities may be *interactive* or *compiled.* Interactive entities are built and executed at the VTB software run-time, while compiled entities are typically created by some other computer languages that require compilation of the entity's code. Interactive entities can be easily modified at run-time but execute much slower than compiled entities. Another consideration is the type of coupling. The VTB supports two types of coupling – *signal* and *natural*. Signal coupling is used in traditional block-diagram system descriptions when connections between elements represent signal flow in one direction, for example, a voltage output from a power supply or a load torque input to an electric motor. Natural coupling enforces physical conservation laws and entities are connected through *ports* which have associated *across*

and *through* variables, e.g. voltage and current in the case of an electrical terminal. Currently ESL supports only signal coupling.

The VTB provides access to external models in a number of ways. These include *wrapper entities* and *user defined devices (UDDs)*. Wrapper entities provide the interface to models created in other simulation software, such as Simulink, Matlab or ACSL. The UDD software, which is part of the VTB development kit, creates C++ source code for the model, which can then be compiled and used as a native VTB entity.

A feature of ESL is that models can be specified as *embedded segments* which, when translated into C++ and compiled, can be built into or embedded in an external C++ program. The embedding program is able to call the embedded ESL model to initialise it, then repeatedly to advance the simulation one time-step at a time. Data communication with the embedded model is achieved through C++ classes corresponding to ESL *package* structures. Further, an ActiveX COM object can be created from an ESL embedded segment. In this case the model is initialised and run by invoking appropriate object *methods* and transferring data through its *properties*.

Considering the alternative methods, it appeared that the simplest way of including ESL models in VTB simulations, would be through the use of wrapper entities which would access ESL models in COM object form. This could be achieved either: by writing a *generic* ESL wrapper which would be associated with a specific ESL model at VTB run-time by specifying the model name as a wrapper parameter; or by creating a *specific* ESL wrapper for each ESL model before running VTB. The former approach is described in this paper.

5. THE ESL COM INTERFACE

The ESL COM interface is described in the following tables:

5.1. Methods

The embedded segment COM methods are listed in Table 1.

Table 1 ESL COM properties

Name	***Meaning***
ExStrt	Prepare embedded code for use - must only be used once at program start.
ExInit	Initialise embedded segment for a single simulation run.
ExSim	Advance Simulation by one time-frame (specified by the simulation parameter CINT).
ExFin	Close down simulation - must only be used once at program termination.

5.2. Properties

Table 2 shows the simulation parameters are properties of the top-level embedded segment COM object.

Table 2 ESL simulation parameters

Parameter	***Default value***	***Meaning***
T		Current value of time**.**
Tstart	0.0	Initial value of time (T) at start of run.
Tfin	10.0	Final value of T at end-of-run.
Cint	1.0	Communication interval – must be same as VTB.
Diserr	0.0001	Discontinuity detection error tolerance.
Interr	0.001	Integration error tolerance.
Algo	1 (RK5)	Integration algorithm (8 algorithms are available).
Nstep	1	Minimum number of integration steps in CINT.

The model input and output variables are made accessible to the calling program by defining them in named ESL *package* structures (*Esl_inp* and *Esl_out*). These variables then appear as properties of the COM object. Similarly any model parameters are defined in an ESL package – *Esl_par* and these also appear as COM object properties. Note that the simulation parameters (*Tfin, Cint* etc) are direct properties of the COM object whereas the interface variables and model parameters are parameters of the sub-objects – *Esl_inp, Esl_out* and *Esl_par*.

Thus to run an ESL COM model, ExStrt is first invoked to initialise the software, after which the simulation parameters and model parameters can be changed or set through the COM properties. ExInit is then invoked to initialise the simulation. This must be repeated each time the model is initialised before a new run. ExSim is then invoked repeatedly to advance the simulation at time-frame. Finally ExFin is invoked to terminate the simulation.

6. EXAMPLE

In order to test these ideas, a very simple single-input, single-output (SISO) ESL model was prepared. The ESL source code is presented Figure 2.

This model has I/O variables *in* and *out* and simply introduces a first-order lag between the input and output. It has two model parameters – *out0* (initial value of *out*) and *tau* (the lag time constant). Note the definition of the interface variables and model parameters in the ESL

packages and the starting keyword *embedded* to force creation of an ESL embedded segment, from which the COM object can be subsequently generated. In addition to directly writing source code, ESL allows a graphical block-diagram model specification, in which the packages may be readily defined.

```
embedded
package esl_inp;
 real: in;
end esl_inp;
package esl_out;
 real: out;
end esl_out;
package esl_par;
 real: out0/0.0/,tau/0.6/;
end esl_par;
segment embed;
use esl_par, esl_inp, esl_out;
real: y;
initial
  y:=out0;
dynamic
  y':=-(y-in)/tau;
communication
  out:=y;
  tabulate " ",t,in,out;
  prepare " ",t,in,out;
end embed;
```

Figure 2 ESL embedded segment for VTB

7. TESTING AND RESULTS

The original idea was to write a *generic* ESL wrapper, from which the specific ESL model would be specified as a parameter at the VTB run-time. This furnishes the wrapper code with sufficient information to locate the COM object program identifier from the registry and hence access its methods and properties. However this posed several problems, one being that the number of interface variables is not known until VTB run-time and VTB 2003 does not have the capability of dynamically changing the number of connections to a wrapper icon in a schematic. In the first instance, therefore, a basic wrapper was written that was hard coded to access the specific ESL model. Although this fell short of the ideal, it allowed the mechanisms to be tested and provide "proof of concept". Figure 3 is a screen dump showing the appearance of the ESL wrapper in a VTB schematic together with graphical output generated in the Visual Extension Engine (VXE).

8. ESL – VTB DEMONSTARTION APPLICATION

In order to evaluate an ESL-VTB interface more fully and provide a measure of performance, a more substantial application was required. To this end, an existing ESL

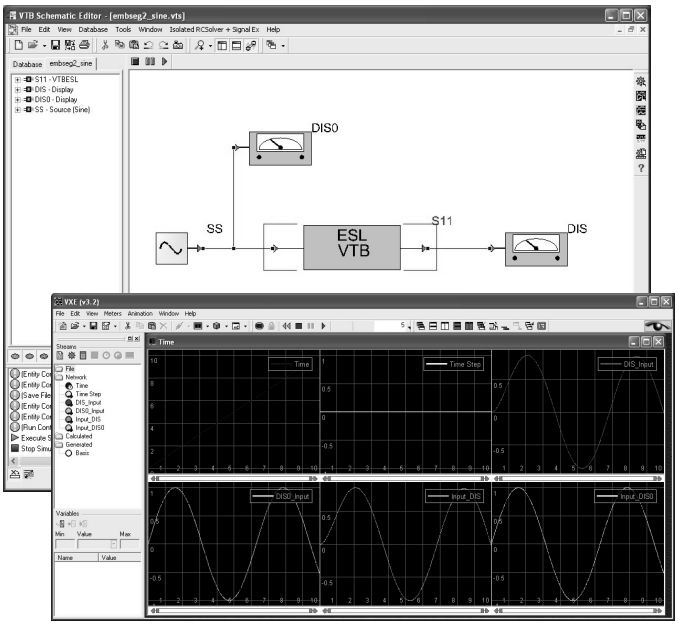

Figure 3 ESL wrapper object in VTB

simulation of a six legged walking robot – "Genghis" [Hay et al 1994] – was modified to be run under the VTB.

The simulation was originally undertaken under an ESA contract to demonstrate ESL's *distributed* and *real-time* simulation capability. The simulation comprised a kinematic model of the autonomous robot (which included edge avoidance, mutual avoidance and tracking algorithms); an interactive graphical control panel, from which the robot's speed and direction could be controlled; and a main module which coordinated the other components and generated graphical data for a 3D visualization. For the ESA contract the simulation ran on Sun workstations – two instances of the robot model plus two corresponding control panels were supported, each (in principle) running on different networked computers. A separate Silicon Graphics computer graphically rendered the model using VISTA software written by EADS-CASA of Spain. The robots had the capability of simultaneously negotiating an arbitrarily defined uneven terrain while tracking a moving target, or each other. Figures 4 and 5 show the original architecture and the appearance of the robot and control panel as rendered by the Silicon Graphics computer.

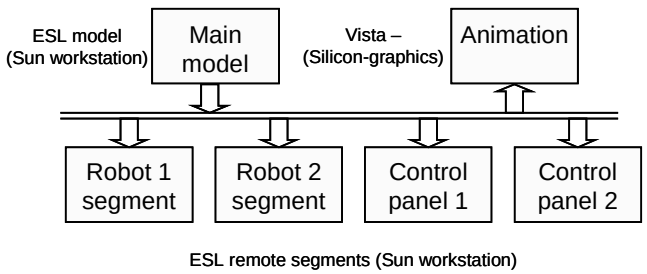

Figure 4 Original Genghis architecture

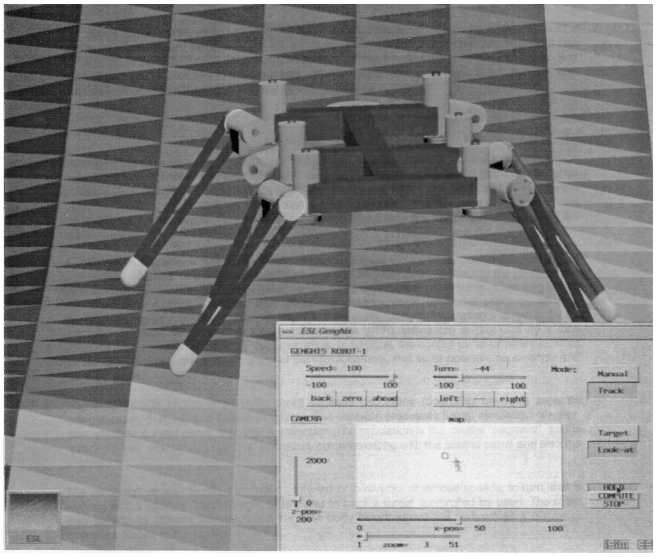

Figure 5 Genghis on Silicon Graphics display

Two architectures were considered for the demonstration -

- A single VTB model with one robot as a remote segment and one control panel as a remote segment.
- Three VTB models comprising a main model which generates graphical data, one robot model and one control panel model.

The first option simply calls a single entity from the VTB (the main embedded segment) and would not, as such, result in a very interesting VTB schematic (there would be no interconnections between models). However, it retains the ESL distributed simulation capability (the control panel and robot are run as remote segments, potentially on different computers).

The second option would entail three ESL entities in the VTB schematic with multiple connections and therefore provide a better test of the ESL-VTB interface.

These architectures are shown in Figures 6 and 7.

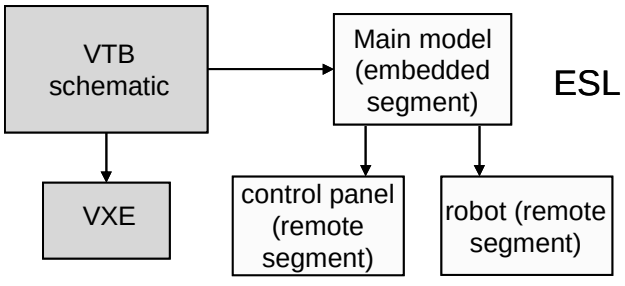

Figure 6 Single VTB model – two remote segments

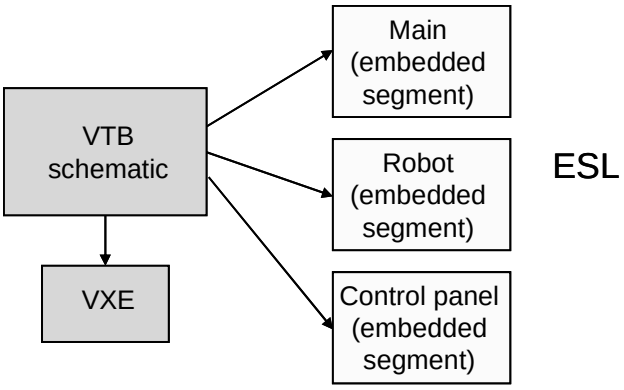

Figure 7 Three VTB models

Because a fully functional ESL wrapper was not available at this time, both architectures were validated using a Matlab program in place of the VTB schematic. The Matlab program calls the ESL segments exactly as they would be called from VTB schematic. For second architecture the Matlab program performs transfer of input and output variable values between ESL embedded segments, replicating the interconnections that would appear on the VTB schematic. The Matlab program generates text file streams of position and orientation data for VXE, allowing an off-line visualization of the robot to be presented. The appearance of the walking robot is shown as a screen dump in Figure 8. As can be seen, at this stage in the project, the rendering of the robot components has been greatly simplified for this exercise.

Figure 8 Genghis walks in the VTB

The panel on the left of the window shows the data streams generated from the Matlab program which are used to drive the animation. The panel on the right shows the hierarchical relationships between the robot components.

9. DISCUSSION

ESL is a powerful simulation tool with a proven track record in a number of major applications. Its particular strengths lie in the rigorous simulation language that underlies graphical or textual models; its accurate treatment of multiple discontinuities and its parallel processing capability provided by the segment structures. With the *emulated* parallel mode, multi-rate simulations can be undertaken thus maximizing computer resources.

The embedded segment-COM interface is the key to integrating ESL models into the VTB. This preliminary study has demonstrated the practicality of the approach and also raised a number of issues: With the approach used, the number of I/O connections to be shown on an ESL element in a VTB schematic is not known until the model name is specified. It also proves problematic for the wrapper software to dynamically access a different COM server for each ESL model. A single generic ESL COM server might be the answer (as with the Matlab interface). Array I/O ports could be used to multiplex multiple connections. An alternative approach might be to generate a specific wrapper entity for each ESL COM model, with the required I/O, prior to loading it into the VTB.

The Genghis robot simulation promises to be a suitable demonstrator for an ESL-VTB interface once the remaining problems have been addressed. Tests have shown that it generates an interesting 3D visualization which will be enhanced once user interaction has been added.

10. CONCLUSIONS

Significant progress has been made towards interfacing the ESL simulation language to the Virtual Test Bed. A wrapper approach has been investigated and, using a simple SISO model, proof of concept has been demonstrated. The work has highlighted a number of issues for further consideration and helped identify alternative strategies. A substantial application (a simulation of an autonomous walking robot – Genghis) has been developed as a demonstrator in readiness to validate a full ESL-VTB interface.

Acknowledgements

The author wishes the thank Antonello Monti and Aalhad Deshmukh of the Department of Electrical Engineering at the University of South Carolina for their helpful advice and for writing the ESL wrapper software.

References

Appleton, A., Davison, S. and Pearce, J.G. 2001. “Rapid Gravity Filter Modelling”. In Proceedings of the 2001 International Conference on Advances in Rapid Granular Filtration in Water Treatment, (London, UK, April 4-6). CIWEM, London, UK, 23-32.

Bonillo, C., Vega, E. and Mejnertsen, S. 2000. "Flight Dynamic System Modelling using ESL". In Proceedings of the 2000 International Conference on Simulation and Multimedia in Engineering Education, (San Diego, CA, Jan 23-27). SCS, San Diego, CA, 127-132.

Crosbie, R.E., Hay, J.L. and Pearce, J.G. 1981.. "Simulation Studies with Modern Computer Structures". Final Report, (ESTEC Contract 4155/79), ESTEC, Noordwijk, The Netherlands.

Davison, S. and Kraft, R.J. 1990. “COGSYS: Real-time Decision Support for Process Control”. In Proceedings of the 10th International Workshop on Expert Systems and their Applications”. (Avignon, France, May).

Hay, J.L. 1987. "ESL Simulation of Spacecraft Battery Cells". In the 1987 Proceedings of the UKSC Conference on Computer Simulation, (Bangor, UK, Sept 9-11). UKSC/SCS, Ghent, Belgium, 92-97.

Hay, J.L., Pearce, J.G., Crosbie, R.E. and Pallett, S. 1994. “ESL Simulation Tool”. Final Report, (ESTEC Contract 10011/92/NL/JG Work Order No. 2), ESTEC, Noordwijk, The Netherlands.

Holliday, P. 2000. "XMM ACC Environment Dynamics Simulation for SVF". In Proceedings of the 2000 International Conference on Simulation and Multimedia in Engineering Education, (San Diego, CA, Jan 23-27). SCS, San Diego, CA, 133-138.

Kraft, R.J. and Pearce, J.G. 2000. "Using ESL in an Integrated Real-Time Compressor Simulation Application". In Proceedings of the 2000 International Conference on Simulation and Multimedia in Engineering Education, (San Diego, CA, Jan 23-27). SCS, San Diego, CA, 121-126.

Pearce, J.G. and Crosbie, R.E. 2000. "ESL-ISE - A Simulation Tool Developed for the Space Industry". In Proceedings of the 2000 International Conference on Simulation and Multimedia in Engineering Education, (San Diego, CA, Jan 23-27). SCS, San Diego, CA, 115-120.

Pearce, J.G., Davison, S. and Appleton, A. 2000. “Rapid Gravity Filter Modelling in the Water Industry”. In Proceedings of 2000 European Simulation Symposium (Hamburg, Germany, Sept 28-30). SCS, San Diego, CA, 499-503.

ISIM International Simulation Limited
http://isimsimulation.com

The Virtual Test Bed – http://vtb.engr.sc.edu/