ABSTRACT

Developers of simulation software have responded to the increasing commercial demand for customised solutions by adding new features and tools to their existing simulation packages. This has led to huge, monolithic applications with functionalities that are constantly extended by addition of wizards, templates and add-ons in a 'generalising-customising-generalising' development cycle. This approach has been successful so far, but customising much of the contemporary simulation software is increasingly difficult.

An alternative approach is to compose, simulation packages from prefabricated components that the users may select, modify and assemble so to acquire the functionality that suits each simulation model. This strategy requires component-based paradigms and integration mechanisms that support the straightforward composition of components regardless of their development and deployment contexts.

This research exploits the Microsoft's .NET integration philosophy to investigate how discrete event simulation (DES) software could pursue a component-based approach that integrates components sourced within distinct contexts. The thesis describes the DotNetSim project that explores the composition of DES applications from components developed within different Microsoft packages in different programming languages. This is done by prototyping DES software across the entire requirements of a simulation application package.

The DotNetSim prototype consists of a Visio-based DES modelling environment which integrates with a .NET simulation engine which, in turn, integrates with an Excel-based output analysis environment. The graphical modelling environment emulates

Schruben's Event Graph methodology for simulation modelling. Visio™ is extended by a number of VBA programs to link together different Microsoft applications in order to capture the models' application logic and dynamics. The simulation engine consists of a number of C# and VB.Net components that implement an event-based simulation executive. It reads the model's logic and dynamics by instantiating the graphical modelling environment, runs the event-based simulation and returns the simulation results to Excel for analysis. The output analysis environment is a template that illustrates the specialisation of the generic data analysis and reporting capabilities of Excel™ to serve the simulation analysis. The components interact directly by instantiating one another's objects.

These three coarse-grained components could be substituted by others that deliver the same functionality, though with different internal operations. With further work, these components could be deployed as web services to which the model's logic is remotely input.