



SIMULATION

Richard Bridge

TESSELLA SUPPORT SERVICES PLC

Issue V1.R2.M0
June 2003



Introduction

Modern business has to stay competitive by keeping development and training costs and times to a minimum, whilst still keeping high levels of quality for both. Modelling and simulation of systems can provide solutions for product development and personnel training without the costs usually associated with these. In this Technical Supplement the reasons for using software simulation, the types of simulation available and the factors involved in developing a successful simulation are introduced.

What is Simulation?

Simulation is a technique to perform tests using a model written in software. Simulation can be used to test a human's response to situations, the model's response to situations, or to optimise a set of values for a model. Types of simulation are discussed below:

Development Simulations are used to prove a product's viability. Modern microchips and circuits are modelled intensively before production. The inputs and desired outputs are known and the simulation is run to prove that the components will perform correctly before incurring the cost of production.

Theoretical Simulations are used to prove the viability of the model itself. When the theoretical model of a system that contains unknown interactions is proposed, it must be validated. Simulation can be used to compare 'real world' results to those produced by the model. Theoretical simulation for comparison is also known as 'black-box' simulation and is used with System Identification (SID) techniques.

'Man-in-the-Loop' Simulations are usually used to test people. Flight simulation is a common example. The model describes the action of a plane as it moves through the air. The simulation takes the control inputs from the 'pilot' and applies them to the model. The resultant position of the aircraft in relation to other world coordinates is then displayed to the user through simulation.

Optimisation Simulations have seen a rapid growth recently, especially in the financial sector. Known models are used and a series of simulations are run to find a set of inputs that will give the best performance from the model (or the most money). The process can be automated to find the optimal solution within a range of values. Optimisation simulations are really an extension of development

simulations, where the model parameters are reconfigured as additional inputs to the system and the process is automated.

Why use Simulation?

There are several reasons why simulation is used and why it is becoming more common practice to simulate systems using software, but the main reason is cost. Simulation provides a method for developing, analysing, modifying and testing systems instead of physically building them. There is little benefit in spending time creating simulations of how a particular paper aeroplane will fly because paper is cheap and plentiful so the plane can be built physically in little time and at little cost. Building a passenger jet is a much bigger task. Simulating the action of air on the control surfaces, calculating the drag coefficients by simulating turbulence and even simulating the action of impact on the construction has a significant benefit in cost over building a physical construction and performing the tests in the 'real world'. The other benefits are flexibility and time. Making minor changes to the shape of a 20-metre wing is a non-trivial engineering task but could be little more than changing a single number in a simulation. The flexibility that this type of simulation affords allows strong decision support with empirical data. The time saved allows for increased exploration of decisions to reach optimum solutions. The end results are reduced development cost and increased product confidence.

Types of Modelling

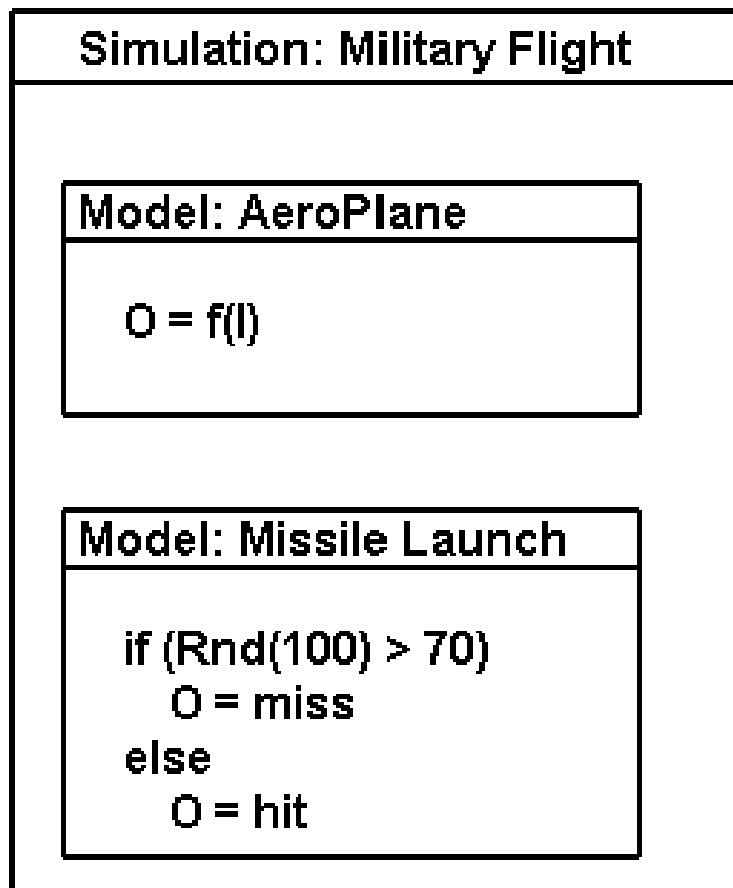
The type of models used varies greatly with application but there are two basic questions that must be answered before developing the model: Should the model be Deterministic or Stochastic? Should the model be Discrete Event or Continuous Time Based?

Stochastic models are based on a series of probabilities. This can be demonstrated with a simple example: In a military simulation there may be a 70% chance of a missile hitting a target. Whenever a 'missile' is launched the model must generate a random number between 1 and 100 and if the number is greater than 70 then the missile will miss. In a stochastic model the input does not dictate the output directly so that running the simulation several times with the same inputs will not guarantee receiving the same outputs.

Deterministic models are based on a combination of mathematical formulae. Given any set of inputs the model can be run repeatedly, always returning the

same value. A model describing the movement of an aircraft (as for a flight simulator) will produce identical output when provided with the same input.

It is of course possible to mix model types within a simulation. Deterministic models are particularly good at representing well-understood and provable methods such as the flight path of an aircraft with no disturbance. Stochastic models are more suited to describing actions that cannot or should not be fully described within a simulation but rather have a known statistical probability of occurring, such as the probability of an unknown factor causing a missile to miss the target. So a military flight simulation may combine both types of model in a single simulation, to make the simulation behave more realistically.



Deterministic and Stochastic Models in Simulation.

The flight simulation example used above would be described as a ‘Continuous Time’ model. In Continuous Time models, the model continues to run and change dependant on time, i.e. with no input from the pilot the plane will still move forward through the air due to its momentum. The alternative is Discrete Event modelling where changes in inputs trigger changes in state within the model. However, there is an overlap between the two types of event models due to the use of software. Continuous Time has to be approximated on digital computer systems as the actual calculations and time increments are discrete. Therefore a practical application of a Continuous Time model can become a Discrete Event model with a clock as an input. This consideration is usually ignored during design but can be useful when representing the system for optimisation.

Levels of Simulation

Designing a model is an exercise in compromise. Accuracy and precision are usually in conflict with execution speed and development time. For this reason it is important to decide on the detail level before commencing development. Simulating an integrated circuit, for a radar detection system for example, has no human interaction so execution speed would probably not be a priority, but accuracy and precision are important to give a high confidence in the results. In this case a high level of detail would be included for each component, to give the most realistic model available. Conversely a simulation that requires user interaction, such as the use of a radar detection system in a flight simulation, must keep non-essential calculation to a minimum to enable ‘real-time’ interaction. So the flight simulator does not need the same accuracy of information about the states within the radar detection circuit itself but rather a statistical description of the likelihood of detecting any given object. High-level simulations may require large numbers of assumptions to be made about the details that make up the simulation and become purely stochastic.

Improving Execution Speed

The speed of execution of the simulation can be critical, making a successful model redundant if the processing requirements are unfeasible. In man-in-the-loop simulations, the simulation will be expected to perform in real-time. If the model is simple or purely stochastic this may not be difficult to achieve, but increased complexity and realism come with a high overhead in terms of processing resources. Similarly, optimisation simulations look to be an excellent way of reducing development time and costs by just running the simulation repeatedly with different sets of inputs, systematically searching for the best solution.

However, optimisation simulations need to be run many times to get a sample of the search space. Each variable parameter adds an extra dimension of complexity into the problem space and consequently adds an extra dimension to the total search time. Stochastic simulations are non-repeatable and as such require multiple runs to produce a statistical representation of the output. Stochastic simulations can become particularly time consuming when it is the ‘freak’ or low-frequency occurrence cases causing discontinuities in the problem space that are of real interest. Because of the advances in computing hardware in recent years the practicality of using simulations has increased but hardware is not yet at the point where we can disregard processing limitations in our simulations. New techniques are being developed constantly to speed up forms of simulation; commercially available simulation packages such as VeriLog have optimisation routines included as standard to help reduce the processing overhead. Simplification methods are application specific and include reducing data sets via Level Of Detail (LOD) approximations, approximating motion calculations using predefined splines, employing lookup tables in place of calculations, using Neural Networks to generate simplified model approximations and splitting the model into separate threads.

Parallel/Distributed Processing

Following the ‘Bigger is Better’ approach, Parallel or Distributed Computing attempts circumvents the problem of limited resources by splitting the simulation so it may be run on n multiple machines. The elapsed time (t_e) can be reduced to t_e/n , theoretically, although the overhead of running a distributed computing environment makes this impossible to achieve. The efficiency is therefore reduced on the whole operation and the total processor time required for the simulation increases.

This technique has recently been employed successfully for large scale Monte Carlo Simulations over the Internet. Monte Carlo Simulation is a technique whereby a model is run repeatedly, with inputs generated at random (or sub-random), to map a problem space. The method is very computationally expensive, so some simulations (such as Climate Prediction.com) split the problem into smaller tasks and ask people attached to the Internet to use their computer’s spare processing time to perform the calculations.

Is the model correct?

If the aim of a simulation is to test the model then this question is the reason for

the simulation existing, but it should be asked of every simulation's model. To check the model there are three stages to go through: Validation, Verification and Calibration. These checks should occur at several stages in the model development and require a strong description or set of requirements to be properly effective.

Validation checks that the theoretical model is describing the system that needs to be modelled. This should be performed first, before any software is produced, and readdressed during development in case practical implementations are forcing simplifications or changes on the model.

Verification checks that the software model conforms to the theoretical model. This stage is especially important in a complex problem domain where a software developer may not have the domain knowledge of the person who developed the theoretical model.

Calibration is the last stage of checking the model is correct and usually concentrates on value checks with a series of test cases.

Simulation Software

Traditionally software has been procedural, with calculations and logical statements performed in sequence. Object Oriented (OO) software groups together actions and data to describe individual entities in the system. OO has proved highly beneficial for generating flexible simulation environments. By using an Object's ability to represent methods and data together, individual components of a model can be created separately and then joined together using a system of interfaces known as a Framework. It should be noted that OO methods tend to have a speed overhead that is absent in procedural designs, meaning that they are not always the optimal solution.

Interfaces

With the possible exception of man-in-the-loop simulations, it is often the case that the majority of thought is given to the development of a model, with only minor consideration given to those who will use it. The creation of a well-designed User Interface (UI) can hold significant advantages for a simulation. When a simulation is designed to be used by non-expert users with limited domain knowledge, it is essential to have an interface that allows the user to visualize what the changes they make are doing. Without appropriate feedback, simulations

become time-consuming to use and the non-expert user will quickly become frustrated. If the user cannot see the effects of changes to input parameters it becomes difficult to suggest improvements to parameters, even when there is output data available. Similarly, when data is produced from the simulation, the lack of a suitable format for viewing the data and drawing conclusions is a large handicap. Even a technically perfect model with excellent performance characteristics will not be used effectively if the support structure surrounding it is not in place.

Tessella and Simulation

Tessella has been involved with numerous simulation projects, covering complex problem domains and a variety of techniques and technologies. A few of these projects are included here as examples to highlight points made in this supplement.

ClimatePrediction.net (CPDN) is using monte carlo simulation to help predict climate change at a level of detail not previously attempted. The aim of CPDN is to quantify the uncertainties intrinsic in climate prediction, so that conclusions about the long-term effects of environmental changes can be drawn from a strong probabilistic footing. Tessella has been working with CPDN to develop a distributed computing solution to enable the large number of calculations involved to be performed. Internet users can download models and data to be run on their home computers using the spare processor cycles, much like SETI@Home. The purpose is to validate and calibrate the climate prediction models being used, a task so large that supercomputers alone would be unable to handle it.

For over 5 years Tessella has been developing AutoCad extensions to allow rapid model development and export to simulation environments. The extensions now form the centrepiece of an integrated design and evaluation system. By extending an existing application, users are given a faster way of developing models. The system is well used and realises considerable benefits in time to manufacture.

Using Object Oriented techniques, Tessella has been involved with building a Graphic User Interface driven probabilistic network simulation tool. The tool is being used to model the possible environmental impact of the disposal of radioactive waste in underground repositories. By employing advanced network solution techniques, the application allows component models to be grouped into super models for simulation. The ease of use of the Interface means that expert and non-expert users alike can use it effectively. Object Oriented techniques give

expandability and flexibility required from this type of simulation environment.

SISTM (Simulation of Strategies for Traffic on Motorways) is a microscopic model of traffic flow on networks originally developed by Transport Research Laboratories. Tessella worked on updating the application to improve the simulation engine, increase the precision of the calculations and enhance the outputs gained from the system. Tessella were also asked to perform performance comparisons. Although the models could be validated, neither the verification nor the calibration phases could be attempted because there was not enough suitable data available. This project highlights the requirement for empirical data or well-defined test cases for model evaluation.

Tessella Support Services plc
Creating Software for Science and Engineering

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software
Data Capture
Simulation Software
Advanced Graphics
Systems Support
Database Applications

Other Technical Supplements available include:

- | | |
|---|--|
| <input type="checkbox"/> Archiving of Electronic Info | <input type="checkbox"/> Object Oriented Programming |
| <input type="checkbox"/> Active Server Pages | <input type="checkbox"/> Pocket PC |
| <input type="checkbox"/> Automated GUI Testing | <input type="checkbox"/> Portable GUI Development |
| <input type="checkbox"/> Bayesian Statistics | <input type="checkbox"/> Printer Technology Guide |
| <input type="checkbox"/> Beowulf Clusters | <input type="checkbox"/> Real Time Systems |
| <input type="checkbox"/> C++ | <input type="checkbox"/> Regression Testing |
| <input type="checkbox"/> Client-Server Technology | <input type="checkbox"/> Simulation |
| <input type="checkbox"/> COM | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Computational Fluid Dynamics | <input type="checkbox"/> Software Development Cycle |
| <input type="checkbox"/> Computer Image Processing | <input type="checkbox"/> Software Documentation |
| <input type="checkbox"/> Electronic Data Capture | <input type="checkbox"/> Software Portability |
| <input type="checkbox"/> Electronic Lab Notebooks | <input type="checkbox"/> Software Re-engineering |
| <input type="checkbox"/> Excel | <input type="checkbox"/> Software Specification |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> SQL |
| <input type="checkbox"/> Federal Drug Administration | <input type="checkbox"/> UNIX Inter-Process Comms |
| <input type="checkbox"/> Firewalls | <input type="checkbox"/> UNIX Systems Performance |
| <input type="checkbox"/> FORTRAN 90 | <input type="checkbox"/> UNIX Workstations |
| <input type="checkbox"/> Instrumentation | <input type="checkbox"/> Visual Basic 6 |
| <input type="checkbox"/> J2EE | <input type="checkbox"/> WAP |
| <input type="checkbox"/> Java | <input type="checkbox"/> Web Services |
| <input type="checkbox"/> Lims | <input type="checkbox"/> Windows 2000 Services |
| <input type="checkbox"/> Linux | <input type="checkbox"/> XML |
| <input type="checkbox"/> Microsoft Net | <input type="checkbox"/> X Windows |



INVESTOR IN PEOPLE

Tessella Support Services plc

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: info@tessella.com Web Address: <http://www.tessella.com>