Proceedings of the Operational Research Society Simulation Workshop 2010 (SW10)

DEVELOPING A GRID COMPUTING SYSTEM FOR COMMERCIAL-OFF-THE-SHELF SIMULATION PACKAGES

Chris Wood & Shane Kite
Saker Solutions
Upper Courtyard
Ragley Hall, Alcester, B49 5NJ, UK
{chris.wood | shane.kite}@sakersolutions.com

Simon J E Taylor
Distributed Systems Research Group
School of Information Systems, Computing & Mathematics
Brunel University
Uxbridge, Middlesex, UB8 3PH, UK
simon.taylor@brunel.ac.uk

Navonil Mustafee
School of Business and Economics
Swansea University
Singleton Park, Swansea, SA2 8PP, Wales UK
n.mustafee@swansea.ac.uk

ABSTRACT:
Today simulation is becoming an increasingly pervasive technology across major business sectors. Advances in COTS Simulation Packages and Commercial Simulation Software have made it easier for users to build models, often of large complex processes. These two factors combined are to be welcomed and when used correctly can be of great benefit to organisations that make use of the technology. However, it is also the case that users hungry for answers do not always have the time, or possibly the patience, to wait for results from multiple replications and multiple experiments as standard simulation practice would demand. There is therefore a need to support this advance in the use of simulation within today's business with improved computing technology. Grid computing has been put forward as a potential commercial solution to this requirement. To this end, Saker Solutions and the Distributed Systems Research Group at Brunel University have developed a dedicated Grid Computing System (SakerGrid) to support the deployment of simulation models across a desktop grid of PCs. The paper identifies route taken to solve this challenging issue and suggests where the future may lie for this exciting integration of two effective but underused technologies.

KEYWORDS: Simulation Modelling, Grid Computing, Flexsim Simulation Software, COTS Simulation Packages, Witness Simulation Software, Simulation Experimentation

1. INTRODUCTION

In the experience of simulation modelling at Saker Solutions, if there is a typical simulation run time, and the authors are reticent to predict such a thing, one would imagine that a time of 40mins to 80mins per replication would not be untypical. Therefore, even a typical 10 replication run would take 7 to14 hours to run. A replication is a single run of the simulation model using a specific set of random numbers. The author isn't trying to prescribe how many replications are required to support a specific model but simply to identify the time required to run a model with even a modest number of replications. In the era when it would have taken 7 hours to make a simple change to a model then this may have been acceptable. Now changes can often be made in minutes and the user needs to be able to receive a quick response to questions. Given that the authors have seen ("commercial" not "academic") models sometimes take over 14 hours to run *a single replication* then this "need for speed" can be extreme.

It is worth understanding the context of a 'quick response'. This is a comment which many users will quote as a requirement of a simulation, i.e. "*Run Speed*". Indeed it is often quoted as an enhancement request. However, this needs to be put into context. When running a scenario on a single PC, an instant response would be ideal, if not practical. If it takes the time required to get a coffee this would be seen to be instant (no pun intended). Indeed 30 or 40 minutes would seem to be the longest time that someone is prepared to wait, and this would still not be ideal. However conversely if it takes 6-12 hours, then realistically it would be deemed to be an ‘overnight’ experiment. So, to make an impact on speed the requirement is not to double a model’s run speed but rather we need to decrease the time taken to run a scenario by a factor of 10 at least.

The notion of “The Grid” offers an integrated infrastructure providing geographically distributed sites with secure access to computational, data and instrumentation resources (Foster and Kesselman, 1998). It has been suggested as being potentially beneficial to simulation modelling (Robinson, 2005; Taylor and Robinson, 2006). Indeed, previous work has investigated how a Grid based on a network of PCs, a *desktop grid*, can be successfully used to support different simulation applications (Mustafee and Taylor 2008; Mustafee and Taylor, 2009). The question therefore that this paper seeks to address is how far can *“The Grid”* be taken in the world of simulation practice? The paper is structured as follows: section 2 discusses relevant issues in simulation practice using practical experience from Saker Solutions, section 3 reviews Grid computing and desktop grid technology, section 4 presents the system requirements for a Grid computing simulation solution, section 5 outlines the resulting desktop grid system called *SakerGrid* that has been developed and has been successfully deployed by Saker Solutions and the Distributed Systems Research Group of Brunel University and section 6 draws the paper to a close with some concluding remarks and future work.

2. SIMULATION AT SAKER SOLUTIONS

Discrete-event simulation (henceforth referred to as simulation) has been used to analyse production and logistics problems in many areas such as commerce, defence, health, manufacturing and logistics for many years. The first discrete-event simulation languages appeared in the late 1950s. These evolved during the 1960s and 1970s. With the arrival of the IBM PC, the 1980s saw the rise of visual interactive modelling environments that allowed simulation modellers to visually create and simulate discrete-event models (Ellarby and Kite, 2006). Today, simulation is “practised” using Commercial-Off-The-Shelf (COTS) Simulation Packages (CSPs). They include packages such as AnyLogic™, Flexsim™, Simul8™ and Witness™. Each has a wide range of functionality including visual model building, simulation run support, animation, optimization and virtual reality. Some allow model features to be developed in C++ or Java, some have their own dedicated programming language and all are able to be linked to other COTS software (such as Microsoft Excel). Nearly all CSPs only run under Microsoft Windows™ which dictates that a discussion of the integration of simulation products with other software needs to be focused on a Windows environment. Whilst this may necessarily eliminate some Unix based software, the authors make no apologies for following industry standards.

In any project a certain amount of time is spent running models for both testing and analysis. Our experience has been that as simulations tools have become easier to use and more graphically expressive (Ellarby and Pattison, 2008) it has lead to requests for ever larger models that result in longer running times.

Experience from talking to Saker Solutions clients shows some key pointers to the issues facing simulation practitioners:

- Models are becoming larger in terms of the number of components, event list, physical file size and data files (associated databases and data sources that are accessed during a run). This leads to longer run times and end user frustration.
- Models are longer lived and can exist for the lifetime of the physical system. This allows for a significantly higher investment in simulation than if a model is developed for a one-time problem.
- To make simulation more accessible to the end user, sophisticated front-ends have been developed that “hardwire” good simulation practice. Saker Solutions have developed such frontends which allow the user to demand scenarios quickly and effectively but the user is then left to wait for the response as the simulation works through the replications.

The introduction of *SakerGrid* has meant that not only has the overall duration of a simulation project been reduced, but the effectiveness of the modeller has also been increased because in the time that it takes to analyse the results from one scenario another scenario has been run.

3. DESKTOP GRIDS

A desktop grid is one that aggregates non-dedicated, de-centralised, commodity PCs connected via a network (Mustafee and Taylor, 2009). As “typical” desktop applications do not fully utilise the computing power available on a machine, desktop grids can harvest spare computing resources of desktop PCs (so-called cycle scavenging) (Choi et al., 2004). There are several different desktop grid middleware that can be used to create such an infrastructure. Examples include BOINC (Anderson, 2004), Condor (Litzkow et al., 1988), Platform LSF (Zhou, 1992), Entropia DCGrid (Kondo et al., 2004), United Devices GridMP (United Devices, 2007) and Digipede Network (Digipede Technologies, 2006). Of these, Condor is arguably the most popular as it has a large deployment base, it is relatively easy to use and is available free of cost. However, it is also large (it is general purpose), complex (there are many features) and unsupported (the running of the application using the middleware is the responsibility of the user). Further, like all distributed computing applications, grid desktop middleware uses different communication schemes that need to be matched with local security policies. Some organisations may prefer servers over peer-to-peer processes or vice versa, or indeed only allow communication via web services or through specific ports (for example sharing port 80 – the port that supports the World Wide Web). This can be problematic if the scheme and policy do not match as the user cannot control such things easily. Further, there may well be the need for specific message compression or encoding that the middleware may not support. The fact that many CSPs only work under Microsoft Windows means that some middleware cannot be used because it either does not run on that operating system or runs with limited functionality.

Most desktop grid middleware works on the Manager-Worker principle. The Manager is a job dispatcher that receives and sends jobs out to Workers. Workers work on these jobs and return results. Most middleware also assumes that a "job" consists of the application program and its data. Over and above operating system constraints, the use of a grid to support simulation is more complex and follow different design principles. These design principles (Mustafee and Taylor, 2008) are summarised below.

3.1 Middleware Integration Approach

To expand the issue introduced above, jobs sent from the Manager to a Worker typically run in a "sandbox" that is implemented by the middleware. This provides a logically separate and secure execution environment to prevent unauthorized access to a computer. However, to do this, the application software (the CSP) must be integrated with the middleware. A "job" would therefore be the model plus the data needed for a simulation run. This approach is more common for applications such as the Java Virtual Machine, i.e. the application needed to run Java programs. To be successful, this approach would require the CSP Vendor and the middleware supplier to agree to either produce a specific version of the middleware supporting the CSP or to bundle the CSP as part of the middleware. Neither approach is particularly attractive.

3.2 CSP-Runtime Installation Approach

This approach involves the installation of a CSP package at runtime, i.e. just before the simulation experiment is conducted. In this case the CSP is sent to the Workers along with the model and data. This approach allows for flexibility to send jobs to machines which are free irrespective of the configuration of that machine. However this approach may not be feasible for a number of reasons. (1) the size of CSPs frequently exceed 100s of MBs and it may not be feasible to transfer such large amounts of data to multiple Clients over the network, (2) the CSP will first need to be installed on the desktop grid node before the simulation can start, (3) such an installation is normally an interactive process and requires human intervention, (4) an installation normally requires administrative privileges on the Client computers, (5) transferring CSPs may lead to a violation of the software licence agreement that may be in place between the CSP vendor and the organization (if the number of desktop grid nodes executing simulations exceed the number of licences purchased) and (6) time becomes an issue here where the replications are relatively short the time to load the software could be prohibitive.

3.3 CSP-Preinstalled Approach

This involves installing the CSP at the Worker as a normal installation. The jobs sent to the Workers are therefore the model and the data, removing the issues described above. As simulations are created by trusted employees running trusted software within the bounds of a fire-walled network, security in this open access scheme could be argued as being irrelevant. In this environment the sandbox security mechanism described above may be forfeited. This methodology allows for flexibility in allowing jobs to be packaged and sent to machines with the right configuration.

4. SIMULATION REQUIREMENTS

Section 2 outlined relevant issues to simulation practice and Section 3 has outlined desktop grid middleware and CSP support strategies. Let us now discuss this in detail with respect to SakerGrid.

SakerGrid has been configured to support a range of simulation tools. The current implementation is focussed on the Flexsim simulation software. Flexsim is a PC based, object-oriented simulation tool used to model, simulate, and visualise any manufacturing, material handling, logistics or business process. Flexsim has been developed to allow users to build models directly in a 3D environment with the ease of conventional 2D models whilst still allowing more advanced modellers the flexibility to design new objects and if required even embed their own C++ code into the tool, thus allowing Flexsim's modelling objects to be customized to exactly match the processes being studied.

Flexsim requires a hardware dongle, the two most common configurations of which are ‘local’ and ‘network’. A local license allows one instance of Flexsim to run on the machine that it is connected to. A network license allows multiple machines to use a pool of licenses that are available from the license server. Using a network license means that any number of machines can have Flexsim installed, but the number of instances that can be run simultaneously is limited by the number of licenses in the pool. This means that the Grid Manager not only has to limit jobs to the machines that have Flexsim installed but also has to monitor the number of licenses available in the pool.

The input and output data for a Flexsim model is typically stored using one (or more) of three different mechanisms: an internal Flexsim tables, an Excel spreadsheet (either directly, via a DSN or via an intermediate flat text file, such as a CSV) or a database (via a DSN).

Handling unforeseen problems that occur whilst a scenario is being run in a distributed environment is one of the key issues of implementing a desktop grid. Individual machines may crash or be interrupted by a user returning to their machine and network connectivity issues can also cause a machine to leave the grid. In all of these situations it is possible that a job will have been sent to run on a machine, but no results are returned. Replications which are not completed because of a failure outside of the simulation need to be automatically re run. Equally if the failure is related to the simulation then this needs to be logged ad reported to the user.

Whilst models are in development it is possible that a bug will be encountered that prevents a scenario from completing. In this event the grid must ensure that the job is halted allowing other jobs to use the computing resources and communicate as much information as possible to the modeller that submitted the suspended job to aid them in eliminating the cause of the problem.

Given that even when using a desktop grid it may take a considerable amount of time to run a large number of scenarios of a particular model, it is important that an appropriate prioritisation strategy is employed by the Grid Manager to allow more urgent jobs to supersede this work if the need arises. Indeed the authors note that ultimately this is the key to a successful grid application. Scheduling is a complex topic in any paradigm and in the case of distributing simulations this is no less the case. Machines (resources) have different speeds and capabilities, models (jobs) require different amounts of time from the resources and users (customers) have different priorities.

Given the above, of the three CSP-grid integration approaches discussed in section 3, the CSP-preinstalled approach is considered the most appropriate because (1) it does not require any modification to the CSPs – thus, CSPs that expose package functionality can be grid-enabled, (2) it does not require any modification to the grid middleware, (3) CSPs that are usually installed on the PCs of the simulation practitioners can be utilized for running simulation experiments from other users in the background and (4) it supports the restrictions imposed by the license requiring the presence of a hardware dongle.

In *SakerGrid*, the Worker has been designed to integrate with commercial simulation packages (CSPs) by using the “pre-installation approach” (Mustafee and Taylor, 2008). When the Worker is started it scans the machine of the desktop grid for all of the available simulation packages and registers these with the Grid Manager. In this way SakerGrid supports different products and versions of products transparently to the user. When the Grid Manager is ready to distribute a block of work it will use this information to determine which of the Workers are capable of handling it.

As we have discussed, the key elements to creating a grid based solution is really in the integration and support of the user interface and the CSP. One might therefore take the view that it does not matter which middleware is used. However, our justification for developing our own system is that we wish to supply a *supported* grid solution in an *unknown*, possibly highly restricted, security environment that is *optimised* for simulation and provides for future expansion to allow for distributed simulation and optimisation. We now discuss our solution to this, *SakerGrid*.

5. SAKERGRID

There are three components to *SakerGrid*; the Manager, the Worker and the Client. Each grid consists of one Manager that handles the job queue and dispatches the jobs to Workers in packages of work referred to as 'blocks'. When the block completes on the Worker the results are returned to the Manager where they are combined with the results from the other Workers. At this point the Worker is available to receive the next block of work from the Manager. The Client is used by the analyst to submit jobs to the Manager, monitor the progress of their execution and to download the results when they are complete.

Figure 1 shows how the SakerGrid middleware isolates the CSP from the network and other implementation details of the Grid. This means that the CSP can be used unmodified out of the box. All non-sensitive models and datasets are cached by the Workers to reduce the amount of network traffic and to reduce the start-up time of the Worker when it is issued a job. The modular architecture of the Manager allows the models and data to be stored either locally or on a central server whilst the jobs are pending in the queue before they are dispatched to the appropriate Worker when required.

The reduction in running time that has been achieved can be clearly seen in Figure 3. It gives a comparison of the overall running time of 1, 5, 10, 20, and 40 replication scenarios with a model that runs for approximately 7.5 minutes per replication The model used was a finished client model which although a relatively small project, the model was a 'real' model and typical for a small simulation projects. This model had the advantage of allowing us to experiment with significant numbers of replications. Whereas previously a 40 replication scenario would have been termed an 'overnight' experiment it can now be completed in less than 30 minutes giving almost instant results. Similar reductions in overall running time have also been observed on larger models with some taking up to 14 hours per replication.

Our experience of using SakerGrid to test models during development has been that the overall project duration is reduced by approximately 10%. However, the amount of testing that can now be accomplished in this time is far greater than if the test packs were run sequentially. This enables the modeller to not only test the specific area of the model that has been modified during that phase of the development, but also to run a comprehensive regression test without increasing the duration of the project. This in turn has lead to higher quality, more robust models.

Figure 2 is a screenshot of the Client application showing a number of jobs queued on the Manager. The list at the top shows the progress and state of the jobs and statistics about their running time. It is possible to drill-down into each of the jobs by clicking the plus symbol to see detailed statistics about the individual replications and the Workers that have run them. The chart at the bottom is currently displaying the average running time by Worker of the replications that have completed in the highlighted scenario.

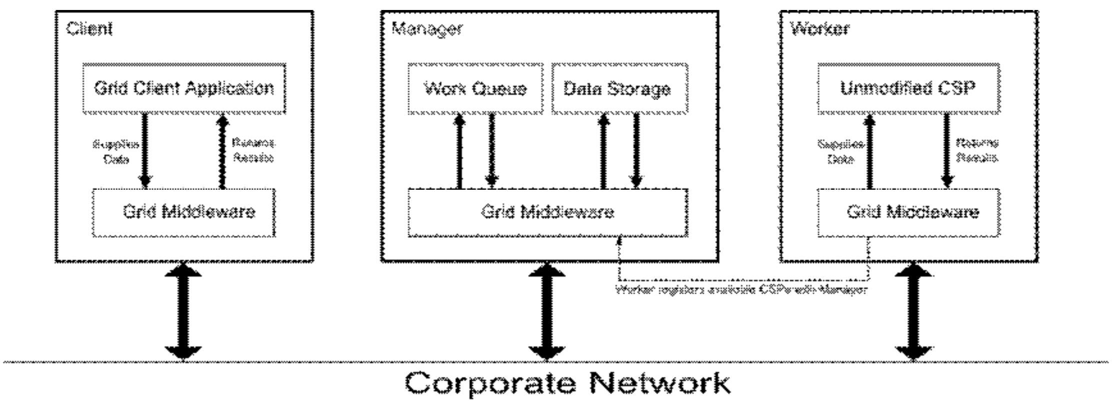

Figure 1 *SakerGrid Architecture*

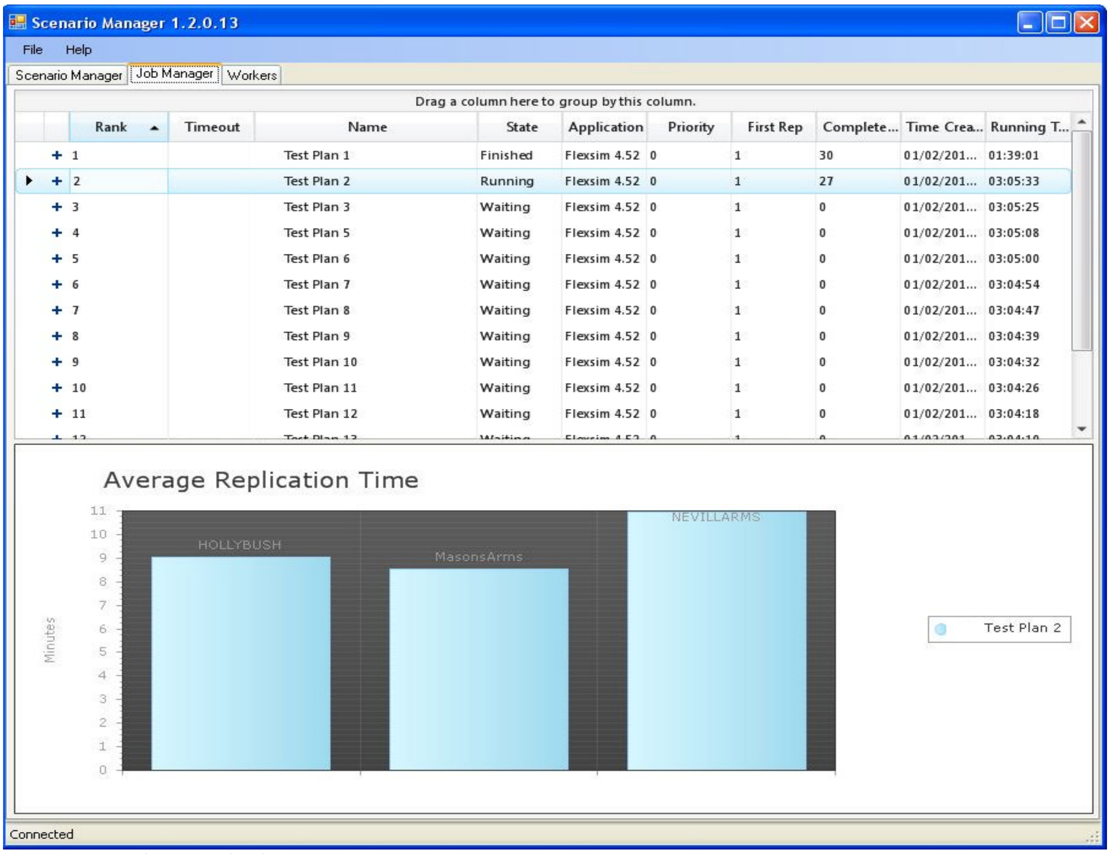

Figure 2 *Saker Grid Client GUI*

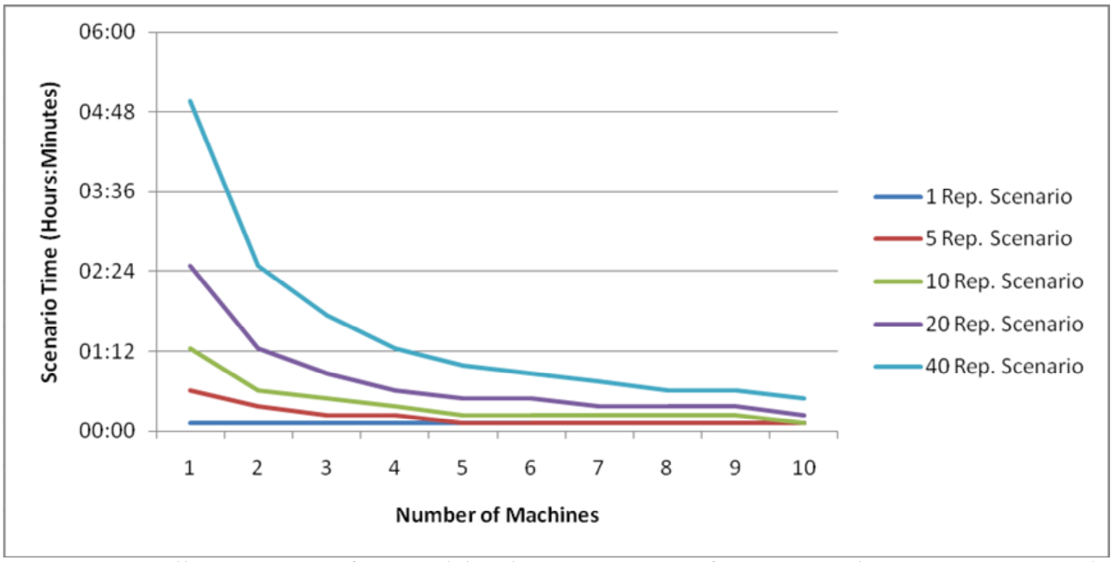

Figure 3 *Overall scenario time for a model with a running time of approximately 7.5 minutes per replication*

.

6. CONCLUSIONS AND FUTURE WORK

This paper has reported on a successful industrial-academic collaboration between Saker Solutions and the Distributed Systems Research Group of Brunel University that has resulted in SakerGrid. This novel system enables users to obtain a step change in performance. An important advantage is that while other incarnations of Grid-enabled simulation have required users to be expert in computational technology, SakerGrid is a packaged solution that allows users to focus on building models, experimenting (quickly) and analysing the results where ultimately the true benefit of simulation lies. SakerGrid, whilst taking significant development effort, has already reaped benefits and has now been in constant use for many months. Saker Solutions are now in the process of launching the software into the market.

SakerGrid still has limitations. In order to achieve this step change in performance it requires a significant expenditure in licences to support multiple running of replications. Saker Solutions is already working on the development of a virtual grid which will allow users to 'issue' models to the Virtual SakerGrid where replications will be distributed across numerous nodes giving a performance ahead of what even the largest simulation users can hope to achieve. This is only part of the answer; Saker Solutions are also working with Software vendors to look at new licensing options to allow simulation models to take a licence with them as they are allocated across SakerGrid. Effectively allowing SakerGrid to distribute temporary Grid run time licences, thereby significantly reducing the investment cost and allowing the users to define for any scenario the balance between response time and cost. Additionally, the authors perceive that as use of SakerGrid grows so will the need for advanced scheduling options. Future development is therefore being targeted to increase the functionality of the product particularly in enabling more sophisticated scheduling rules to facilitate multiple users with diverse model run times and resource requirements. In addition SakerGrid will shortly be enabled to support other simulation software with Witness and AnyLogic already under development and more will then follow.

REFERENCES

D.P. Anderson (2004). "BOINC: a system for public-resource computing and storage". In *Proceedings of the 5th International Workshop on Grid Computing*, pp.4-10. IEEE Computer Society, Washington, DC, USA.

S. Choi, M. Baik, C. Hwang, J. Gil, and H. Yu (2004). "Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment". In *Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications*, pp. 366-371. IEEE Computer Society, Washington, DC, USA.

Digipede Technologies (2009). "The Digipede Network". Accessible online *http://www.digipede.net/products/digipede-network.html*. Last accessed 17 November 2009.

M. Ellarby, and S. Kite (2006). "Are Rich Visual Environments a Gimmick or a Real Aid to the Understanding and Acceptance of Results?" In *Proceedings of the 2006 OR Society Simulation Workshop*, Leamington Spa, UK, March 28-29, 2006, pp. 295-299.

M. Ellarby, and G. Patisson (2008). "Are State of the Art Simulators Moving the Modelling Goalposts?" In *Proceedings of the 2008 OR Society Simulation Workshop*, Worcestershire, England, April 1-2, 2008, pp. 291-296.

I. Foster, and C. Kesselman (1998). The grid: blueprint for a new computing infrastructure. *San Francisco, CA: Morgan Kaufmann*.

D. Kondo, A. Chien, and H. Casanova (2004). "Resource management for rapid application turnaround on enterprise desktop grids". In *Proceedings of the 2004 Conference on Supercomputing (SC'04)*, paper 17. IEEE Computer Society, Washington, DC, USA.

M. Litzkow, M. Livny, and M. Mutka (1988). "Condor - a hunter of idle workstations". In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pp.104-111. IEEE Computer Society, Washington, DC, USA.

N. Mustafee, and S.J.E. Taylor (2008). "Investigating Grid Computing Technologies for Use with Commercial Simulation Packages". In *Proceedings of the 2008 UK Operational Research Society Simulation Workshop*, Birmingham, UK, 297-307.

N. Mustafee, and S.J.E. Taylor (2009). "Speeding Up Simulation Applications Using WinGrid". *Concurrency and Computation: Practice and Experience*, 21(11): 1504-1523.

S. Robinson (2005). "Discrete-event simulation: from the pioneers to the present, what next?" *Journal of the Operational Research Society*, 56 (6): 619-629.

S.J.E. Taylor, and S. Robinson (2006). "So where to next? A survey of the future for discrete-event simulation". *Journal of Simulation*, 1(1): 1-6.

United Devices. (2007). "Grid MP: The technology for enterprise application virtualization". Accessible online *http://www.ud.com/products/gridmp.php*. Last accessed 18 March 2007.

S. Zhou (1992). "LSF: Load sharing in large-scale heterogeneous distributed systems". In *Proceedings of the 1992 Workshop on Cluster Computing*. Supercomputing Computations Research Institute, Florida State University, Florida, USA.

AUTHOR BIOGRAPHIES

CHRIS WOOD is a consultant at Saker Solutions. Since completing his degree in Computer Systems Engineering at Warwick in 2006 he has worked on a number of both software engineering and simulation projects. His email address is <chris.wood@sakersolutions.com>

SHANE KITE has been involved in the Simulation industry for over 25 years. With a background in Manufacturing Engineering at Ford, Shane developed early applications of graphical simulation in the automotive industry, using the Fortran based 'See Why' product amongst others. Since then, Shane has had a successful career in simulation. Prior to becoming Managing Director of Saker Solutions he was President of Lanner Inc. and a board member and founder shareholder of Lanner Group, the developers of the Witness Simulation product. Shane is a member of the Informs College on Simulation and the Society of Computer Simulation as well as the UK Operational Research society. His email address is <shane.kite@sakersolutions.com>

SIMON J. E. TAYLOR is the co-founding Editor-in-Chief of the UK Operational Research Society's (ORS) *Journal of Simulation* and the Simulation Workshop series. He has served as the Chair of the ORS Simulation Study Group from 1996 to 2006 and was appointed Chair of ACM's Special Interest Group on Simulation (SIGSIM) in 2005. He is also the Founder and Chair of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. He is a Reader in the Distributed Systems Research Group in the School of Information Systems, Computing and Mathematics at Brunel. His email address is <simon.taylor@brunel.ac.uk>.

NAVONIL MUSTAFEE is a lecturer in the School of Business and Economics at Swansea University. Prior to this, he was a Research Fellow in Brunel University and Warwick Business School. His research interests are in e-Infrastructures and Grid Computing, Information Systems, Operations Research and Healthcare Simulation. He completed his PhD in Information Systems and Computing from Brunel University in 2007. He is a member of the drafting group of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. His email address is <n.mustafee@swansea.ac.uk>