*Proceedings of the 2010 Winter Simulation Conference*
*B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.*

# IMPROVING SIMULATION THROUGH ADVANCED COMPUTING TECHNIQUES: GRID COMPUTING AND SIMULATION INTEROPERABILITY

Simon J. E. Taylor

Brunel University
ICT Innovation Group
School of Information Systems,
Computing and Mathematics
Uxbridge, UB8 3PH, UK

Navonil Mustafee

Swansea University
School of Business and Economics
Singleton Park
Swansea, SA28PP, Wales, UK

Shane Kite
Chris Wood

Saker Solutions
Upper Courtyard
Ragley Hall
Alcester, B49 5NJ UK

Stephen J. Turner

Nanyang Technological University
Parallel & Distributed Computing Centre
School of Computer Engineering
Singapore 639798, SINGAPORE

Steffen Straßburger

Ilmenau University of Technology
School of Economic Sciences
Helmholtzplatz 3
98693 Ilmenau, GERMANY

## ABSTRACT

Today, due to exciting developments in advanced computing techniques and technologies, many scientists can now make use of dedicated high speed networks and high performance computing. This so-called 'e-Science' is enabling scientists across many fields to work together in global virtual research communities. What do these advancements mean for modeling and simulation? This advanced tutorial instigates two key areas that are affecting the way M&S projects are being developed and deployed. Grid Computing addresses the use of many computers to speed up applications. Simulation Interoperability deals with linking together remote simulations and/or speeding up the execution of a single run. Through the use of case studies we hope to show that both these areas are making a major impact on the practice of M&S in both industry and science, as well as in turn supporting the future capabilities of e-Science.

## 1 INTRODUCTION

'e-Science' is a term that is appearing more and more across scientific communities studying a wide variety of disciplines such as astronomy, biology, physics and medicine. The 'e' comes from the enhanced support that high performance communication networks and high performance computing give to a scien-

tist's day-to-day activities. Combining these facilities with advanced distributed computing techniques, novel software applications are being written that allow scientists to work together across the world more effectively by sharing and processing huge data sets quickly, collaborating remotely in real time and accessing remote instrumentation that they were previously not able to.

What does e-Science mean for modeling and simulation (M&S)? So far no real 'e-Science version' of M&S exists as most work has been focused on specific scientific fields. However, work is gradually starting on *in-silico* virtual laboratories that will use many techniques that are familiar to the M&S community. This drive will possibly become a new area of research for M&S specialists. In terms of e-Science benefits for M&S, over the past decade there has been a major drive to create generalized high performance computing techniques to speed up applications. This area, called Grid Computing, can potentially have a major impact on M&S applications. In parallel with this have been developments in Simulation Interoperability. These focus on methods to run simulations together over a network to either speed up a simulation or link together geographically remote simulations. Both these areas can be highly beneficial to M&S. So far these have 'evolved' almost separately.

The purpose of this advanced tutorial is to raise awareness of the benefits of these distributed computing areas. To investigate these areas we present two case studies of Grid Computing and an overview of Simulation Interoperability.

## 2 GRID COMPUTING

Grid Computing was introduced over ten years ago as an approach supporting hardware and software infrastructures that could provide access to dependable, consistent, pervasive and inexpensive high-end computational resources (Foster and Kesselman 1998). The idea was that organizations would share these resources in collaborative working environments to work on a specific problem for a specific time. These organizations are called Virtual Organizations. On this basis Grid Computing enables coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. A broader definition of Grid Computing includes the use of computing resources within an organization for running organization-specific applications. In practice, the supporting organizations tend to be academic and/or government in nature and the resources tend to be high performance computing clusters.

Commercial M&S tends to be performed by companies that do not really fit into the Virtual Organization model for many reasons. Further, typical commercial M&S software cannot practically run on high performance computing clusters. An alternative to this is Desktop Grid Computing, or Desktop Grids. A Desktop Grid is one that aggregates non-dedicated, de-centralized, commodity PCs connected via a network (Mustafee and Taylor 2009). As "typical" desktop applications do not fully utilize the computing power available on a machine (especially multi-core systems), 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, CONDOR, Platform LSF, Entropia DCGrid and Digipede Network. Of these, CONDOR (Litzkow, Livny, and Mutka 1988) 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 organizations may prefer servers over peer-to-peer processes or vice versa, or indeed only allow communication via web services or through specific ports in a computer (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 COTS Simulation Packages (CSPs) used in industry 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 Desktop Grid to support simulation is more complex and follows different design principles. These design principles (Mustafee and Taylor 2008) are summarized below.

### 2.1 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. A job is the 'package' of the simulation with input data. This is the most common 'form' of Grid Computing. However, if we are using a CSP, the CSP software must be integrated with the middleware. A "job" would therefore be the simulation software, the model and 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.

Within the context of this tutorial, the next two principles are specific to CSP software.

### 2.2 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 Workers over a 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 Worker computers, and (5) transferring CSPs may lead to a violation of the software license 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 licenses purchased).

### 2.3 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.

To illustrate the benefits of Grid Computing we now present two case studies. The first is a scientific case from Systems Biology that describes how the SIMAP Utility developed at Brunel University was 'Grid-enabled' with the grid middleware CONDOR using the 'middleware integration approach' (Wang et al. 2009). The second is a case study from Saker Solutions, a simulation consultancy in the UK, that used the 'CSP-Preinstalled Approach' with in-house grid computing software called SAKERGRID to speed up simulation projects (Wood et al. 2010).

## 3 SYSTEMS BIOLOGY

In terms of Systems Biology, a biological system is a set of complex interactions (network structure) rather than many individual molecular components. This can be anything from a simple biological process, such as a biochemical reaction cycle, a gene regulatory network or a signaling pathway in a cell, tissue, an entire organism, or even an ecological web. Simulation can involve, for example, the process of simulating an abstract model of a biological system to test hypotheses with *in-silico* experiments or to provide predictions to be tested by *in-vitro* and *in-vivo* studies. In order to achieve the goal of answering biological questions, models have to reliably depict a biological system and be able to predict its behavior.

The SIMAP Utility is a platform-independent software environment for biomodel engineering developed at Brunel University (Wang et al. 2009). This supports the modeling of biochemical networks, and also the simulation and analysis of the dynamic behavior of biochemical models. It uses a modular architecture that allows other developers to easily plug-in various components and to update them without reinstalling the whole tool. The system has been developed using Java technology and can be run on many platforms that support JRE (Java Runtime Environment 1.5.x or higher). The tool can compute changes of species concentrations over time with particular parameter values by simulating a Systems Biology Markup Language (SBML) model numerically with SOSlib. The simulation results can be presented in two ways: plots and report text files. The SIMAP Utility consists of several plug-in modules which include the Data Management module, Simulation and Analysis Tools module, and a Grid Access Point. The Simulation and Analysis Tools module includes a set of computational modules for simulating and analyzing biochemical models. These are an Ordinary Differential Equations-based simulator, a sensitivity analyzer, a parameter scanner, a model fitting module, a gene knockdown analyzer, and a model logic checker.

### 3.1 CONDOR

For this research we used CONDOR as it has a large deployment base, it is relatively easy to use and is available free of cost. CONDOR is an opportunistic job scheduling system that is designed to maximize the utilization of workstations through identification of idle resources and scheduling background jobs on them (Chapman et al. 2005). A collection of such workstations is referred to as a CONDOR pool. Dual-core PCs are increasingly available in workplaces, and CONDOR exploits these multiple cores transparently.

The CONDOR architecture defines resource providers and resource consumers over a manager-worker architecture. The resource providers make their resources available to CONDOR for the processing of jobs that originate from the resource consumers. The jobs to be processed may have dependencies with regard to the operating system and the physical machines on which the job is to be processed, the memory and disk space required, the available software libraries that are needed and so forth. On the other hand, the resource providers may have certain constraints (e.g. only Java jobs can be run) and preferences (e.g. jobs originating from resource consumer "x" is given priority) based on which access to their resource is granted. CONDOR allows resource consumers and resource providers to advertise these requirements, conditions and preferences by providing a language called *classified advertisements* (*ClassAds*) that provide a flexible and expressive framework for matching jobs originating from the former with resource offers from the latter. The *ClassAds* are scanned by a CONDOR *matchmaker agent* running on only one computer in a CONDOR Pool to find a match between the requirements advertised by the resource consumers and the resources advertised by the resource providers. Once a match has been found by the matchmaker agent, it notifies both the resource consumer and the resource providers. Upon receiving this notification, the resource consumer claims the resource advertised by the resource provider through a claiming protocol. The job is executed by the resource provider and the results of the computation are returned back to the resource consumer.

CONDOR allows end-users to *submit* jobs and to *query* job status using two alternative mechanisms: (a) through use of a submit description file, and (b) through use of programming APIs that are exposed by

CONDOR as Web Services (Litzkow, Livny, and Mutka 1988). The latter approach can integrate CONDOR capabilities into existing software, and this is the approach used by us to integrate the SIMAP Utility with CONDOR (through a Java-based job manager utilizing CONDOR Web Services).

### 3.2 SIMAP Utility Requirements for Grid Computing

In order to integrate Grid middleware with the SIMAP Utility, the key questions were how to minimize work in porting and how to make the use of such technologies straightforward to support and configure. Ideally, the SIMAP Utility would just need some simple information, such as the Grid server name and an account, so that jobs can be simply submitted to the Grid. When jobs complete, the Utility should be able to retrieve output files either through notifications or periodically via queries. Notification means that the Utility would need to open ports on the machine to "listen" to these notifications. However, this solution is not allowed in many cases because of security policies or firewall settings (a trend that is getting more and more severe each year). Owing to these security considerations, it was decided that a query-based approach with a request/reply protocol should be used. This has been implemented in the 'Grid Access Point'. The core of the Grid Access Point is a user friendly UI interface integrating with the client side APIs of the CONDOR based job manager, which is a Java-based high level facade (API packages) for submitting/querying/retrieving jobs between the SIMAP Utility and the CONDOR pool. This is now described.

### 3.3 CONDOR-Based Job Manager

The integration of the SIMAP Utility with the CONDOR-based job manager through client side APIs is illustrated in Figure 1. The Utility generates the required input file with a set of scanning parameters for each simulation job, and then wraps the job command (shared by all jobs of a simulation) and the corresponding input file into a job request and sends it to the CONDOR Pool resource consumer agent. The Utility will query the job status after submission of job requests, and if any job completes correctly, it will retrieve the relevant output files. During implementation testing of this approach two problems were identified: *job submission* and *output file retrieval*. When the number of job requests is large, for example thousands, the submission procedure itself will take a lot of time (half hour or more). In order to obtain output files the CONDOR Pool's resource consumer agent was queried. If the query frequency is not appropriate, the useless workload may be put to the resource consumer agent by sending too many queries or a bottleneck of many output files waiting for retrieval may accumulate. This is further complicated by the small runtime of a typical Systems Biology simulation (less than a minute).

With regard to submission time, the relationship between submission time and job execution time was also investigated. In the first version of the job manager, all job requests were submitted in one transaction. This means that the CONDOR resource consumer agent could submit jobs to other machines to execute until all job requests are received. This led to delays as the transaction had to be completed prior to commencing results collection. To improve this, job requests were submitted in batches. This allowed the batched jobs to be distributed and to commence their execution before more jobs were received by the CONDOR Pool.

The retrieval of output files is dependent on the decision as to when to query job status. In the case of thousands of jobs, all files cannot be retrieved at the time when all jobs complete because the size of thousands of output files can reach several Gigabytes. A simple solution to this is to check the status of each job in a loop and retrieve its output file at once if it completes until all jobs complete. The disadvantage of this approach is that many useless queries regarding status of unfinished jobs may be made. In order to improve the query efficiency, a feedback control-based approach was developed. Here the feedback used for each query is the jobs completion rate $R$. This is the number of completed jobs divided by the number of queried jobs. For example, consider the situation for 100 jobs in total, where a query checks the status of the first submitted 20 jobs. If 15 jobs complete, then the jobs completion rate for this query is 15/20. If $R$ is smaller than a threshold $T$, it indicates that the current query frequency is faster compared with the

job completion speed. In this case, the query interval is increased by making the query process sleep longer before launching the next query and decrease the number of queried jobs in the next time. On the other hand, if *R* is larger than *T*, it indicates that the query frequency is slower compared with job completion speed so the query process should increase the rate of querying. Therefore, the interval between two queries relies on the jobs completion rate. The effect of this feedback control-based approach is to throttle job query frequencies by reacting to the job completion rate and thus balancing output vs processing.

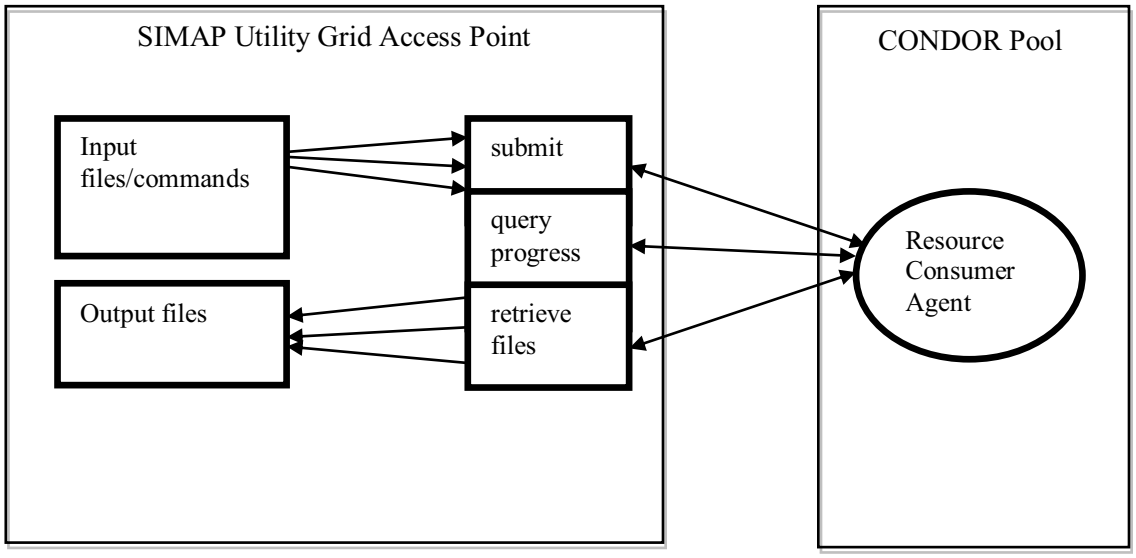


Figure 1: The SIMAP Utility Grid Access Point and CONDOR Pool Integration: Grid Access Point Components and Interaction

### 3.4 EXPERIMENTS AND RESULTS

In order to investigate the performance of the SIMAP Utility with a CONDOR Pool using the modifications described above, a set of experiments were performed on a representative case study, the *mammalian ErbB signaling pathway* (Chen et al. 2009). Briefly, in this pathway the ERbB1-4 receptor tyrosine kinases (RTKs) and the signaling pathways they activate govern most core cellular processes such as cell division, motility and survival and are strongly linked to cancer when they malfunction due to mutations, etc. An ODE-based mass action ErbB model has been constructed and analyzed in order to determine what roles each protein plays and to ascertain how sets of proteins coordinate with each other to perform distinct physiological functions. The model comprises 499 species (molecules), 201 parameters and 828 reactions. The model implements compartments for plasma, endosomal membranes, cytosol, nucleoplasm and lysosomal lumen, as well as clathrin-mediated endocytosis.

The testbed comprised of a CONDOR Pool with 32 Desktop PCs (64 cores). Each machine was configured with dual 2.1GHz cores and 2 Gigabytes RAM. A further PC configured with the same specification hosted the resource consumer agent. All the machines were connected to the network at 100Mbps. On a single PC of this specification using a single core a single simulation in our case study has an approximate run time of 20 seconds. The job size was around 1 MB as were the output results files. To investigate the performance 32 to 4096 simulation jobs were ran in steps. The corresponding speedup graph is shown in Figure 2. This shows that the CONDOR implementation achieved a speedup of around 12 using 16 machines (32 cores). When the number of jobs is small, such as 32 or 64, fewer machines are effectively used in the CONDOR Pool giving a smaller speedup of around 6. Another conclusion from the results is that the speedup does not increase beyond 16 machines (i.e. 32 cores). This indicates that the system has entered a full loaded status with this kind of job granularity.

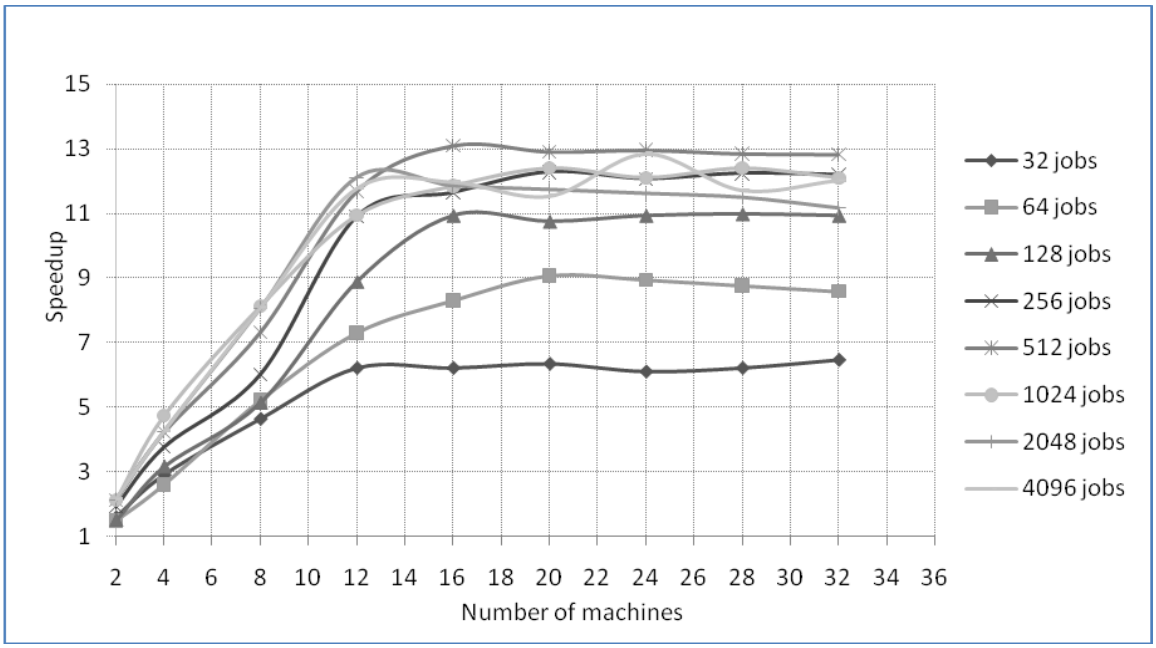


Figure 2: Speedup of jobs on different numbers of machines

To determine if better performance could be achieved the role of the CONDOR resource consumer agent was studied. The small model size and the relatively large data file might mean that this agent was 'overloaded' and was not given the chance to properly distribute the work. In order to confirm this assumption, a test was performed which investigated the effect of an additional resource consumer agent. 512 original jobs were submitted (20s computation time) over 32 machines through one resource consumer agent and then through two resource consumer agents. The performance improvement by using 2 resource consumer agents is shown in Figure 3 and demonstrates that using two resource consumer agents achieves a better speedup than just one.

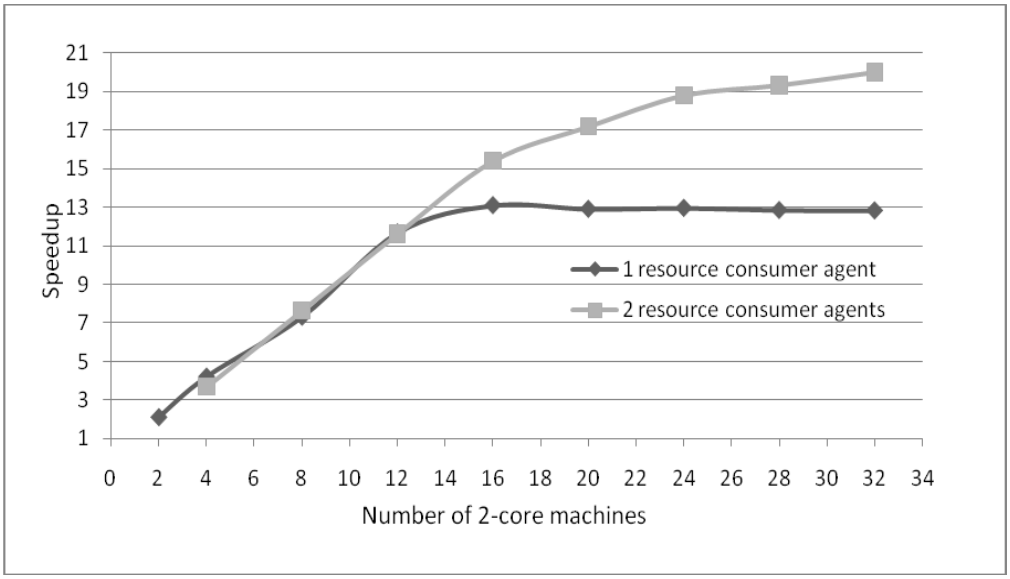


Figure 3: Relative speedup (32/16 machines) with increasing job granularity (1 resource consumer agent)

Overall what have we learnt from this study? Grid Computing using CONDOR and a Desktop Grid can be highly beneficial to Systems Biology, especially when there are a huge amount of runs. Typically small job execution times with relatively large data files is not a good 'profile' for Grid Computing. Excellent speedups are usually derived from jobs with long execution times relative to communication overheads. However, this case study has served to show that will a little 'tinkering' good speedups can be achieved. Further, this has shown that the use of desktop PCs, a resource available in many different companies and institutions, can be deployed to benefit this highly useful scientific application.

On the negative side this has also shown that using an 'off-the-shelf' middleware such as CONDOR has its drawbacks. Much of this work studied how to get the most out of CONDOR rather than SIMAP. The project team spent several months experimenting with different techniques and approaches. In the end this led to a successful result. The team also has experience in developing Grid solutions for CSP applications (Mustafee and Taylor, 2009). For technologies such as CONDOR, this leads to many new problems. Further, an expansion to combining Distributed Simulation with Desktop Grid computing may be better developed with Grid middleware dedicated to simulation. Given the specific needs of CSP software management and varying organizational security policies, it could be argued that developing Grid middleware *dedicated* to CSPs is a better approach.

The next case study takes this view and presents an in house approach to Grid Computing.

## 4 AN INDUSTRIAL CASE STUDY

Some (not all) simulation runs take a long time to execute and some simulation experiments (not all) have a large number of runs. Flexsim is a CSP that is a PC-based, object-oriented simulation tool used to model, simulate, and visualize 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 modelers the flexibility to design new objects and if required even embed their own C++ code into the tool, thus allowing Flexsim's modeling objects to be customized to exactly match the processes being studied. As with other CSPs, simulations in Flexsim can take a long time.

Until recently Flexsim, in common with most simulation software required a hardware dongle, the two most common configurations of which are 'local' and 'network'. Flexsim Software Inc have now released a version of its software which utilizes a soft license so that dongles do not need to be present. 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 any 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: 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 "become unavailable" in a 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 without further input from the user. Equally, if the failure is related to the simulation then this needs to be logged and 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 a 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 modeler that submitted the suspended job to aid them in eliminating the cause of the problem. Of course, it is a fundamental require-

ment that the individual replication can be reproduced in isolation so that the exact situation identified from a replication running on a Grid can be reproduced on a standalone machine.

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 prioritization strategy is employed by a Grid Manager to allow more urgent jobs to supersede presently queued jobs if the need arises. Scheduling is a complex topic in any paradigm and in the case of distributing simulation experimentation 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.

### 4.1 SAKERGRID

SakerGrid has been developed by Saker Solutions in the UK as a dedicated Desktop Grid solution for CSPs. It has been configured to support a range of simulation tools. The current implementation supports the CSP Flexsim.

SakerGrid uses a CSP-preinstalled approach as (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, a Worker has been designed to integrate with CSPs by using this approach. When the Worker is started it scans the machines of the Desktop Grid for all of the available simulation packages including various versions of the same package and where appropriate, libraries, 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.

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, the motivation for developing a dedicated system is that Saker wished to supply a supported grid solution in an unknown, possibly highly restricted, security environment that is optimized for simulation and provides for future expansion to allow for distributed simulation.

As shown in figure 4, there are three components to SakerGrid; the Manager, the Worker and the Client. Each deployment 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.

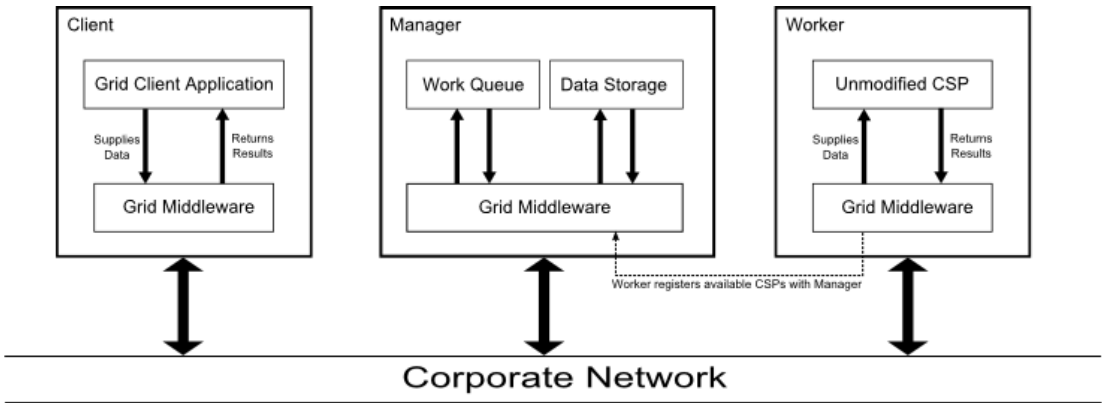


Figure 4: SakerGrid Architecture

The SakerGrid middleware isolates the CSP from the network and other implementation details of the Grid. This means that the CSP can be used without modifications. 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 5. 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, was a 'real' model and typical for a small simulation projects. This model had the advantage of allowing experimentation 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.

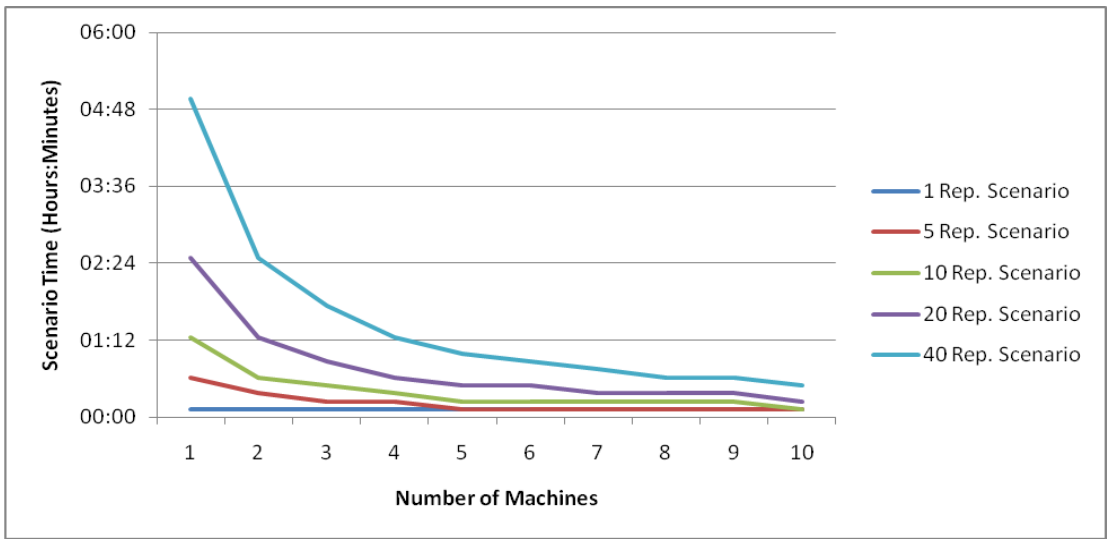


Figure 5: Overall scenario time for a model with a run time of approximately 7.5 minutes per replication

The 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 modeler 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.

## 5 SIMULATION INTEROPERABILITY

The previous sections have successfully shown how Grid Computing can benefit simulation by using many computers. Simulation interoperability presents alternative sources of benefit from using multiple computers. It can be defined and interpreted in many different ways. For many simulation applications, interoperability of a simulation system with other information systems may be a necessity. A typical scenario for this type of simulation interoperability would be the initialization of the model with input data from a database or for storing results.

A different interpretation of simulation interoperability concerns the question of how different, potentially heterogeneous simulation systems can interoperate with each other. For example:

- An automotive company is planning to build a new factory. The manufacturing line is modeled and simulated using a CSP. Experimentation investigates how many engines can be produced in one year against different levels of resources (machines, buffers, workers, etc.)
- A regional health authority needs to plan the best way of distributing blood to different hospitals. A model is built using a CSP. Experimentation is carried out to investigate different supply policies against "normal" and emergency supply situations.
- A police authority needs to determine how many officers need to be on patrol and how many need to be in the different police stations that it manages. A model is built and experiments are carried out to investigate staffing against different scenarios (football matches, terrorist attacks, etc.)

In Schumann, Schulze, and Straßburger (2000) the former form of interoperability is described as "Level 1" and the latter as "Level 2" simulation interoperability. This section discusses relevant standards and applications for both forms of interoperability and illustrates potential usage scenarios.

### 5.1 Interoperability between simulation systems and other information systems

Interoperability between a simulation system and other information systems may be required for different purposes. A very common scenario is the retrieval of simulation initialization data from a data base. In a simulation application supporting the operational control of a production system this could involve the download of order data from an operational Enterprise Resource Planning (ERP) system which then serves as the basis for verifying different short term control strategies. The feedback into a Manufacturing Execution System (MES) could be the best determined order sequence.

In a more general case, interoperability may be required to automatically create a simulation model. In such approaches of automatic (or semi-automatic) model generation a simulation model is not created manually using the modeling tools of the chosen simulator, rather it is generated from external data sources using interfaces of the simulator and algorithms for creating the model. This is often also referred to as "data-driven model generation" (Bergmann and Strassburger 2010). The promise of such approaches is that they, if successful, can reduce the amount of time needed to create a simulation model as well as the expertise needed for creating and conducting simulations. There is a wide variety of potentially relevant external data sources and IT systems. They include systems relevant during the product development, like Computer Aided Design (CAD) and Computer Aided Planning (CAP) and systems from production and shop floor control, like ERP and MES.

What is required to enable the depicted types of interoperability? Generally speaking, there must be appropriate interfaces and data formats (preferably standardized) to enable the required information exchange syntactically and semantically. First of all, the simulation systems must have the appropriate capabilities. Regarding interfaces, many simulation systems at least facilitate data exchange with spreadsheets and data bases in some way. Some also provide network access via socket interfaces. The import of XML files is also generally possible, but often has to be implemented in the simulation model itself. Most simulation systems also provide interfaces to include external code stored in libraries ("DLL interface"), although the capabilities vary quite significantly. These interfaces help to solve the syntactic interoperability problem, i.e. they provide the means to technically exchange data with other IT systems.

Secondly, the semantic issue, i.e., in which format data exchange should take place, must be addressed. There are only very few standards in support of this issue. They include the older SDX format (Simulation Data Exchange, see (Sly and Moorthy 2001)) and the quite young SISO standard CMSD (Core Manufacturing Simulation Data, see (SISO 2009)).

SDX is an XML-based file format for exchanging primarily geometric information. SDX is a vendor dependent format and is only supported by one major CAD system (Factory CAD) and some CSPs (AutoMod, WITNESS, Simul8, Plant Simulation). SDX can be regarded as a viable solution for exchanging geometry data, primarily in a uni-directional way from the CAD system to simulation system. The basic idea is that geometric objects can be defined in the CAD system as relevant for the simulation and that

they can carry parameters identifying some basic properties for the simulation. Most requirements that go beyond geometrical information can not be transferred using SDX.

CMSD, on the other hand, represents an attempt to create an open standard, which can transfer not only layout- and structural data of a system, but also system load data (e.g. orders), control data (resource allocations, shift plans, etc.) and several other items. CMSD is also based on XML. Its main advantage is that is represents an open standard. Limitations exist for describing the dynamic behavior of a model. Also, CMSD is currently not supported by any major system vendor. Still, it holds a large potential for facilitating interoperability between simulations systems and other information systems. It could therefore become a future standard for integrating simulation systems into the IT landscape of companies.

### 5.2 Interoperability between multiple simulation systems

Interoperability between multiple simulation systems concerns the question of how models, developed in potentially different commercial simulation packages (CSPs) can interact with each other. One example based on a practical scenario from General Motors is given in Miller et al. (2007). In this scenario, parts of the automotive plant (e.g., Body Shop, Paint Shop, General Assembly) are modelled independently and at different levels of resolution in different CSPs. The question addressed in their paper is how these models can be combined into a single model of the factory for investigating the dependencies between the models. The solution presented suggests the use of aggregation techniques to create a single model representing the overall factory at the desired aggregated (less detailed) level of resolution.

While this may have worked for this particular scenario, the more general solution from the interoperability perspective would have been to simply stick with the existing models and combine them into a distributed simulation (Fujimoto 2000).

The main motivation for using distributed simulation in this case would therefore be to facilitate model reuse by "hooking together" existing simulations into a single simulation environment, thereby potentially alleviating the cost of creating new models. Other motivations may also exist (e.g., memory and processing requirements, geographically distributed resources), but shall not be discussed in this section.

When combining models developed in multiple CSPs we again have to find solutions for the syntactic and the semantic interoperability. However, as we are trying to link simulation models together at runtime, i.e., in an on-line fashion, we have additional requirements compared to what was discussed in the previous section. We have to implement efficient synchronization mechanisms - the core problem when applying distributed simulation techniques. Several well known approaches to the problem of synchronizing logical simulation clocks exist. They can be classified into conservative and optimistic approaches. As most of the CSPs in use do not provide any efficient state-saving techniques, one is typically forced to apply conservative approaches with lookahead. Related challenges can be imposed by the capabilities of the respective CSPs (access to event lists, inclusion of external events, etc.) - see Strassburger et al. (1998) for a detailed discussion of these issues.

Distributed simulation typically requires the use of a distributed simulation middleware. A distributed simulation middleware is a software component that implements the distributed simulation algorithms to achieve synchronisation and to efficiently exchange data at runtime. Middleware such as HLA-RTI (IEEE 2000) provides the advantage of being based on an open standard. The High Level Architecture for Modeling and Simulation (HLA) defines an interface specification, which simulations have to use to interact with each other. This interface specification can be seen as a solution to the syntactic interoperability problem. The question if and how a CSP can be brought to communicate via this interface specification is a complex problem, but several CSPs have been shown to be able to achieve this task (Straßburger 2006, Taylor et al. 2006).

Towards the semantic interoperability problem, even with the HLA no final solution exists. HLA supports defining a common semantics on a very basic level with its object model templates. They allow the definition and description of the data to be shared between CSPs and their models. Using reference object models one can establish a common frame of reference for a certain domain (e.g., manufacturing,

real-time training simulation, etc.). Mechanisms to match simulation models which differently interpret certain data to be exchanged are not commonly available yet.

A group developing standards for facilitating interoperability between different CSPs is the Commercial Off-the-Shelf Simulation Package Interoperability (CSPI) Product Development Group (PDG) within SISO. A first standard of this group has been recently approved (SISO 2010) and defines reference models for common CSP interoperability problems. They build the basis for discussing interoperability problems involving multiple simulation systems and are ultimately targeted at providing a basis for integrating appropriate CSP-to-CSP interfaces into the commercially available simulation systems. The Interoperability Reference Models (IRMs) are intended to be used as follows:

- To clearly *identify* the model/CSP interoperability *capabilities* of an *existing* distributed simulation, e.g., the distributed supply chain simulation is compliant with IRMs Type A.1, A.2 and B.1.
- To clearly *specify* the model/CSP interoperability *requirements* of a *proposed* distributed simulation, e.g., the distributed hospital simulation must be compliant with IRMs Type A.1 and C.1.

An IRM is defined as the simplest representation of a problem within an identified interoperability problem type. Each IRM can be subdivided into different subcategories of problem. As IRMs are usually relevant to the boundary between two or more interoperating models, models specified in IRMs are as simple as possible to "capture" the interoperability problem and to avoid possible confusion. These simulation models are intended to be representative of real model/CSPs but use a set of "common" model elements that can be mapped onto particular CSP elements. Where appropriate, IRMs specify time synchronization requirements and present alternatives. IRMs are intended to be cumulative (i.e. some problems may well consist of several IRMs). Most importantly, IRMs are intended to be understandable by *simulation developers, CSP vendors and technology solution providers.* A previous advanced tutorial (Taylor et al. 2009) has addressed this topic.

## 6 CONCLUSIONS

Motivated by advances in e-Science, this paper has presented two threads of advanced computing as applied to M&S. There are many more such as Groupware and Cloud Computing, and the benefits of these are being felt in many areas. M&S has to a certain extent been neglected but is catching up quickly. We hope that this advanced tutorial will promote discussion and more demand of these technologies.

## ACKNOWLEDGMENTS

The authors would like acknowledge the contributions made by Jun Wang, Xuan Liu, Qian Gao and David Gilbert of Brunel University to the Systems Biology work.

## REFERENCES

Bergmann, S. and S. Strassburger. 2010. Challenges for the automatic generation of simulation models for production systems. In *Proceedings of the 2010 Summer Computer Simulation Conference*, ed. A. Bruzzone, et al. 545-549. San Diego, California, Society for Computer Simulation.

Chapman C., C. Goonatilake, W. Emmerich, M. Farrellee, T. Tannenbaum, M. Livny, M. Calleja and M. Dove. 2005. Condor Birdbath-Web Service interfaces to Condor. In *Proceedings of the 2005 UK e-Science All Hands Meeting*. 737-744.

Chen, W.V., B. Schoeberl, P.J. Jasper, M. Niepel, U.B. Nielsen, D.A. Lauffenburger, and P.K. Sorger. 2009. Input–output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular Systems Biology* 5: 239.

Choi, S., 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*, 366-371. IEEE Computer Society, Washington, DC.

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

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*. New York, NY: John Wiley and Sons Inc.

IEEE 2000. IEEE 1516 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). New York, NY: Institute of Electrical and Electronics Engineers.

Litzkow, M., M. Livny, and M. Mutka. 1988. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems,* 104-111. Piscataway, NJ, IEEE Computer Society Inc.

Miller, J.S., R. Combs, E. Foster, J. Tew, D.J. Medeiros, and O. Ulgen. 2007. Clinic: aggregating subsystem models into an automotive total plant throughput model. In *Proceedings of the 2007 Winter Simulation Conference*, ed. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J.D. Tew, and R.R. Barton. 241-249. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mustafee, N., 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*. 297-307. Birmingham, UK, Operational Research Society.

Mustafee, N., and S.J.E. Taylor. 2009. Speeding up simulation applications using WinGrid. *Concurrency and Computation: Practice and Experience* 21(11): 1504-1523.

Schumann, M., T. Schulze, and S. Straßburger. 2000. Different forms of interoperability for harbor models. In *Proceedings of the International Workshop on Harbour, Maritime & Multimodal Logistics Modelling and Simulation (HMS)*. 97-105.

SISO. 2009. Standard for: Core Manufacturing Simulation Data – UML Model (Draft Version from May 11, 2009), Simulation Interoperability Standards Organization (SISO) – CMSD Product Development Group.

SISO. 2010. Standard for Commercial-off-the-shelf Simulation Package Interoperability Reference Models (SISO-STD-006-2010), SISO COTS Simulation Package Interoperability Product Development Group.

Sly, D., and S. Moorthy. 2001. Simulation data exchange (SDX) implementation and use. In: *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 1473-1477. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Straßburger, S., T. Schulze, U. Klein and J.O. Henriksen. 1998. Internet-based simulation using off-the-shelf simulation tools and HLA. In *Proceedings of the 1998 Winter Simulation Conference*, eds. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, 1669-1676. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Straßburger, S. 2006. The road to COTS-interoperability: from generic HLA-interfaces towards plug-and-play capabilities. In *Proceedings of the 2006 Winter Simulation Conference*, eds. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1111-1118. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Taylor, S.J.E., X. Wang, S.J. Turner, and M.Y.H Low. 2006. Integrating heterogeneous distributed COTS discrete-event simulation packages: an emerging standards-based approach. *IEEE Transactions on Systems, Man & Cybernetics: Part A* 36(1):109-122.

Taylor, S.J.E., N. Mustafee, S.J. Turner, K. Pan and S.Straßburger. 2009. Commercial off the shelf simulation package interoperability: issues and futures. In *Proceedings of the 2009 Winter Simulation Conference,* ed. M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin and R.G. Ingalls, 203-215. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wang J, X. Liu, N. Mustafee, Q. Gao, S.J.E. Taylor and D. Gilbert. 2009. Grid-enabled SIMAP Utility: motivation, integration technology and performance results. In *Proceedings of the UK e-Science All Hands Meeting.*

Wood, C., S. Kite, S.J.E. Taylor and N. Mustafee. 2010. Developing a grid computing system for commercial-off-the-shelf simulation packages. In *Proceedings of the Operational Research Society Simulation Workshop (SW10)*. 184-191. Birmingham, UK, Operational Research Society.

## AUTHOR BIOGRAPHIES

**SIMON J. E. TAYLOR** is the Founder and Chair of the CSPI-PDG under SISO. He is the co-founding Editor-in-Chief of the UK Operational Research Society's (ORS) *Journal of Simulation* and the Simulation Workshop series. He was Chair of ACM's SIGSIM (2005-2008). He is a Reader in the School of Information Systems, Computing and Mathematics at Brunel and leads the Distributed Systems Research Group. He has published over 100 articles in modeling and simulation. His recent work has focused on the development of standards for distributed simulation in industry. His email address is <simon.taylor@brunel.ac.uk>.

**NAVONIL MUSTAFEE** is a lecturer in Information Systems and Operations Management at the School of Business and Economics, Swansea University (UK). His research interests are in grid computing, parallel and distributed simulation, and healthcare simulation. His e-mail address is <n.mustafee@swansea.ac.uk>.

**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>.

**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>.

**STEPHEN JOHN TURNER** is Professor of Computer Science and Head of the Computer Science Division in the School of Computer Engineering at Nanyang Technological University (Singapore). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his MSc and PhD in Computer Science from Manchester University (UK). His current research interests include: Parallel and Distributed Simulation, Grid Computing, High Performance Computing and Multi-Agent Systems. He is also Secretary of SISO's COTS Simulation Package Interoperability PDG. His email address is <Steve@pmail.ntu.edu.sg>.

**STEFFEN STRAßBUGER** is a professor at the Ilmenau University of Technology in the School of Economic Sciences. Previously he was working as head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and as a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Ph.D. and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He is a member of the editorial board of the *Journal of Simulation*. His research interests include distributed simulation as well as general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Development Group. His web page and email address are <www.tu-ilmenau.de/wi1> and <steffen.strassburger@tu-ilmenau.de>, respectively.