Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

DISTRIBUTED COMPUTING AND MODELING & SIMULATION: SPEEDING UP SIMULATIONS AND CREATING LARGE MODELS

Simon J. E. Taylor
Mohammadmersad Ghorbani

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

Tamas Kiss
Daniel Farkas

University of Westminster
Centre for Parallel Computing
New Cavendish Street
London, W1W 6UW, UK

Navonil Mustafee

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

Shane Kite

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

Distributed computing has many opportunities for Modeling and Simulation (M&S). Grid computing approaches have been developed that can use multiple computers to reduce the processing time of an application. In terms of M&S this means simulations can be run very quickly by distributing individual runs over locally or remotely available computing resources. Distributed simulation techniques allow us to link together models over a network enabling the creation of large models and/or models that could not be developed due to data sharing or model reuse problems. Using real-world examples, this advanced tutorial discusses how both approaches can be used to benefit M&S researchers and practitioners alike.

1 INTRODUCTION

Distributed computing gives the opportunity to design and implement applications that can run over many computers linked together via a communication network and presents interesting possibilities for Modeling and Simulation (M&S). Experimentation can be carried out faster by distributing simulation runs over many computers and larger models can be built by linking together models over a network using distributed simulation and interoperability techniques. Following previous advanced tutorials in the area (Taylor et al. 2009, 2010), this article presents a novel approach to speeding up simulation using a Grid computing approach called Volunteer Computing and discusses how this could be used in industry. An overview of motivations for using distributed simulation and interoperability is given and guidelines for the development of large models are presented. Note literature presented in this article is from the authors' experiences. Wider literature reviews on these subjects can be found in the referenced articles.

2 SPEEDING UP SIMULATION

The use of many computers to share computing workload and to speed up applications is one of the goals of Grid computing. Overall this field attempts to create managed collaborative working environments that share the resources of many organizations to create Virtual Organizations that support the computing, instrumentation and data needs of global virtual research communities. Critical to this is access to dependable, consistent, pervasive and inexpensive high-end computational resources (Foster and Kesselman 1998). It is also closely allied to the goals of national and international initiatives in e-Science Infrastructures, Cyberinfrastructures and e-Infrastructures. Many academic and industrial communities are benefitting from these huge investments in Information and Communication Technologies (ICT). Although there are some isolated examples of where Grid computing has made an impact on M&S practice, generally, M&S communities have yet to make a sustained benefit.

Many M&S Commercial-off-the-shelf (COTS) Simulation Packages (CSPs) such as AnyLogic, Arena, AutoMod, Flexsim, Simio, Witness and Simul8 only run on desktop computers (i.e., PCs and/or Macs, as opposed to Unix/Linux-based cluster computers typically found in large Grid computing infrastructures). The sub-field of Desktop Grid Computing that investigates the use of these computers and Grid computing is of particular relevance (Mustafee and Taylor 2009). Most Desktop Grid Computing works on the basis of the *Manager-Worker* principle. In 'human' terms, a manager possesses all the work that must be done. The manager hands out work to his/her workers who process their work and hands back the results. When a worker is free, the worker is given another piece of work to do. This continues until the manager has no more work. In Grid computing, the Manager is given or generates jobs or tasks which are sent out to workers. The workers have all the required resources needed to process a task. When a task is finished, the results are returned to the manager and another task is given. In terms of M&S this means that multiple computers execute simulations runs in parallel with the goal of speeding up simulation experimentation.

Taylor et al. (2010) introduced three approaches to using Desktop Grid Computing based on properties of tasks with respect to data and software. Reiterating these from an M&S task point of view we have two dimensions for tasks: *runs* and *environment requirements*. A task can either run a simulation once or multiple times. A worker environment either needs no special requirements or needs some program, model or data installed locally, i.e., simulations can either be sent to a worker in their entirety (program, model and data) or require some form of installation. Examples of these are:

- **Single run task approach**: Here each task represents a simulation run. This is a set of instructions to run the simulation, the simulation program, the model and appropriate data. These self-contained tasks therefore contain everything needed to accomplish the simulation run.
- **Multiple run task approach**: One requirement for successful speed up is that the processing time is balanced with the time taken to send out each task and to send back results. Sometimes it can take longer to send and receive this information than to process the simulation. If this is the case then speedup might not be possible. An alternative is to allow each task to perform several simulation runs (for example running several replications/trials). The concept here is that the program/model/data is used multiple times within a task and thus reduces the need to send these out repeatedly per simulation run. Increasing the computation *granularity* can increase the chance of an acceptable speedup. However, care must be taken to not overly reduce parallelism. Also, more complex instructions may be needed if different runs instead of replications are processed in a task (i.e., more parameter information is required rather than a series of random number seeds/streams).

- **Installed single run approach**: In some cases it may not be possible to encapsulate everything a task requires to fulfill its purpose. For example, models may require fixed data sources (databases/spreadsheets) or CSPs may need to be *installed* (e.g., correctly registered within Windows registry with ancillary software/libraries). In this case a task is just the set of instructions. The worker performs the task according to these instructions using these pre-installed resources. Experience has shown that there can be problems with this approach that include operating system errors in loading/running software, communication problems between COTS Simulation Package and other software packages, and software not being able to run in the background (i.e., the software appears on the screen – this is not preferable as it will interrupt the computer user).
- **Installed multiple run approach**: This combines the need for multiple runs with the need for pre-installation.

We now review experiences of 'grid-enabling' a simulation application using an approach developed by the Enterprise Desktop Grid Initiative (EDGI 2011) based on 'Volunteer Computing.'

3 GRID-ENABLING SIMULATION WITH VOLUNTEER COMPUTING

To illustrate how Volunteer Computing can be used to speed up simulations we use the systems biology simulation *SIMAP* introduced in Taylor et al. (2010). Briefly, 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. SIMAP (Wang et al. 2009) supports the modeling of biochemical networks, and also the simulation and analysis of the dynamic behavior of biochemical models. The tool has been developed in Java and 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 SBML ODE Solver (SOSlib 2011). Experiences on grid-enabling SIMAP with CONDOR were reported in Wang et al. (2009).

3.1 Volunteer Computing

Volunteer Computing, or desktop grid systems, are capable of offering computing resources in the magnitude of millions of PCs. The installation and maintenance of the client side software is extremely simple. The resources are donated by individuals or institutions and their computing power can be utilized for desktop grid computations whenever the processors are idle. However, this also means that desktop grid systems come with absolutely no guarantees concerning their quality of service. Desktop grids typically implement parameter sweeps (where experimental variables are incrementally changed; one simulation per variable combination) or experimentation where inter-process communication is not required. Possibly the most well-known example of this is the Volunteer Computing application supporting the SETI@HOME project (SETI 2011) and the most well-known technology is the Berkeley Open Infrastructure for Network Computing (BOINC 2011). In a previous piece of research we have successfully used BOINC to grid-enable Excel-based simulations (Zhang et al. 2007).

The SZTAKI Desktop Grid (SZDG 2011; Kacsuk et al. 2009) (Figure 1) is an extension of BOINC in order to make it more flexible, versatile and scalable in terms of enabling the interconnection of different BOINC projects and execution of parameter sweep applications from a generic, high level user interface without the intervention of the BOINC project administrator. Implementing legacy (existing) applications to a BOINC-based grid is facilitated by using a BOINC Wrapper, essentially a batch/script file used by a BOINC client to execute the application. This has several limitations. The SZDG provides a more generic solution by using the GenWrapper. This uses a POSIX-like shell script and environment that gives more flexibility for running applications and processing work.

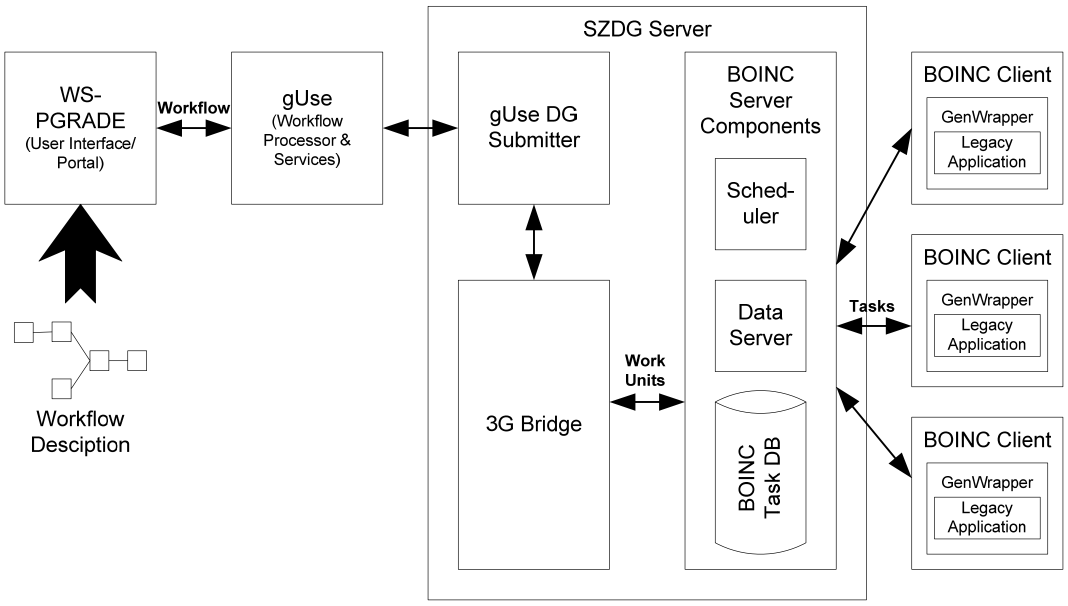

Figure 1: The SZTAKI Desktop Grid.

In Grid Computing, tasks are normally submitted through some web-based job submission portal. This is the same for the SZDG. The portal used is the WS-PGRADE portal, an important element of the Grid User Support Environment (gUSE 2011, Kacsuk 2011). gUSE is a grid virtualization environment providing a scalable set of high-level Grid services by which interoperation between Grids and user communities can be achieved. gUSE provides data management services and control services in the system. The WS-PGRADE portal supports the development of grid applications through a high-level, workflow-oriented programming approach. The nodes of the workflow represent actions (computations) and connections between nodes represent file transfers. The portal also allows local and remote desktop grid resources to be mapped to these nodes. This is supported by extending the architecture with the 3G Bridge (Generic Grid-Grid Bridge) and the DC-API (essentially a Grid to BOINC adaptor) which provides Desktop Grid access to other Grid Infrastructures. For more information on the development of this type of Volunteer Computing technology and other advances in Desktop Grid research see the websites of the European EDGeS (Enabling Desktop Grids for e-Science) (EDGeS 2011) and its follow up the EDGI (European Desktop Grid Initiative) (EDGI 2011) projects.

In this paper the University of Westminster Local Desktop Grid (WLDG) is used, an implementation of the SZDG. This connects laboratory PCs of the University of Westminster (London, UK) into a Desktop Grid infrastructure. The university is set over four main campuses and some additional smaller locations in Central and North-West London each of them offering a variable number of Windows based dual core PCs for teaching purposes. Over 1600 of these machines are connected to the WLDG. The WLDG can be utilized by the university's researchers to run their computation intensive tasks. It is also connected to the EGI infrastructure by the EDGeS EGI to BOINC Bridge allowing EGI users to run validated applications on the WLDG. The Westminster Grid Application Supports Service (W-GRASS) offers application porting services and runtime support for perspective users. Currently 9 different applications are supported by the WLDG from diverse disciplines, including bio-molecular simulations, 3-D video rendering, x-ray profile analysis and digital signal processing. W-GRASS implements applications on the Grid by following an application development methodology developed by the EDGeS project. This is now presented.

3.2 The EDGeS Application Development Methodology

The EDGeS Application Development Methodology (EADM) follows series of 'conventional' software development steps and concentrates only on application specific aspects needed for porting to a service grid/desktop grid infrastructure. The goal is to address necessary questions of application porting within existing technical constraints and limitations. Here we give a short overview of the methodology only.

The EADM distinguishes between five main participant roles.

- **End–users:** Users of the application. End-users may have very different technical skills from grid experts and only have basic computing skills.
- **Developers/system administrators of the original application:** Application programmers who developed the current non-Grid or non-EDGeS enabled version of the application or in case of commercial applications the system administrators who are responsible for the installation and administration of the software.
- **EDGeS Systems Analysts:** Responsible for capturing user requirements and for the conceptual design specification to port the application to the grid platform.
- **EDGeS Application Programmers:** Responsible for the implementation of the migration of the application. In most cases this group is envisaged as collaboration between developers/experts of the original application and programmers with grid-specific knowledge.
- **EDGeS Grid Operators:** Responsible for operating the Grid on which the ported applications are running.

The EADM has stages that suggest a logical order. However, the overall process is in most cases iterative allowing stages to be revisited when required (often due to participants gaining a better understanding of the needs and possibilities offered by grid computing). The stages of EADM are:

- **Analysis of current application**: Describes the existing application in detail and identifies the target user community, the problem domain, and the typical use cases and functionalities of the system. It also captures technical characteristics, such as the type of computing platform and the way of parallelism (if any) utilized by the current application, data access volume and methods, memory and hard disk usage, programming language, operating system, or security solutions. An Application Description Template has been developed to capture the above information mainly from the operators of the existing application.
- **Requirements Analysis**: Identify how the target user community will benefit from porting the application to the grid platform. The requirements towards the ported application concerning efficiency of execution and data access are analyzed from a user perspective. The target computing platform (either service grid or desktop grid) that the user wants to access as an entry point when executing the application, and the desired user interface are also identified in the User Requirement Specification document.
- **Systems Design**: Outlines the proposed structure of the system. The target computing platform, the type of user interface, and the parallelization and data access principles are designed taking both user requirements and technical feasibility into consideration. The outcome of this stage is a Systems Design Specification that identifies at a high-level how the ported application will work regarding the above aspects. The use of structural diagrams, such as a system block diagram and UML diagram techniques are typically used during this stage.
- **Detailed Design**: Provides class level specification of any required modification in the original application when porting it to a grid platform. This stage results in a Technical Design Specification that forms the basis of the next phase.
- **Implementation**: This phase carries out the implementation of the detailed design as specified in the Technical Design Specification.
- **Testing**: Both the functionalities and the performance of the ported application are evaluated and compared to the identified user requirements.

- **Validation:** This aims to assure that the application causes no harm to the computers of the grid donors and also that it conforms to the generic aims of the target grid system. This stage is required in order to deploy the application on a desktop grid platform where individuals or institutions offer their volunteer resources for the computation and to demonstrate the safety of the grid application to donating institutions.
- **Deployment**: The application is published in an application repository and deployed so it can be utilized by end-users.
- **User support, maintenance and feedback**: Provision of organized support for end-users after deployment to support and maintain the application.

3.3 Porting SIMAP to a Volunteer Computing Desktop Grid

The EADM was followed to develop a version of SIMAP that runs on WLDG. The core of SIMAP is the *SBML ODE Solver Library* (SOSlib 2011). SIMAP has a front end that allows a user to specify and run a biological model specified in SBML. Alternatively a user can use a command line interface with arguments that specify the SBML model and associated data. A single experiment is run (the simulation of the model) and results obtained. A parameter sweep uses SIMAP to run (simulate) a SBML model potentially many thousands of times. A typical usage scenario of the application is the following.

- The user prepares the SBML model files that he/she wants to execute.
- The user specifies/uploads the files (or a single archive with the input files) through the user interface (preferably a portal interface).
- The solution creates work units from the input files, and sends them for execution to the compute nodes of the grid.
- The compute nodes run the simulation, and send back the results to the server.
- The GUI provides a way to download the results.
- The user can use the downloaded results and do further analyzing and processing.

In Grid computing the existence of a command line interface makes the task of porting an application to a Grid system a lot easier (as the alternative would be to redevelop the application to directly link to Grid software). Essentially a worker uses a batch file containing command line arguments to run the 'grid-enabled' application. The existence of a command line interface therefore offers a simple parallelization technique. Without modifying the original application at all, the application can be simply distributed with required SBML input files to the worker nodes. The application will therefore contain three parts: a generator, a set of worker, and a collector application. The generator gets input from the user and creates a work unit for each SIMAP input file. The worker runs the simulation and sends back the results to the server, where the collector creates an archive which will hold the result files. Master-worker type parallelization is possible as the simulations are independent from each other. The relationship between these parts is shown in Figure 2. The key here to a quick Grid implementation is that the SIMAP application does not require any modification as it can be accessed by a command line interface. Only a batch file wrapper is needed which implements the control flow shown in Figure 2.

The application is computation but not data intensive. Therefore, there are no special requirements on data access. The normal BOINC data distribution mechanism is sufficient. The following list summarizes the minimum set of files required by the implementation.

1. **Input ZIP file:** The archive file that will be uploaded by the user on the portal. This archive holds all of the SBML files. The size of this file can be large and will have some effect on performance. However, this should be mitigated by overall runtime.
2. **SBML model file:** the SOSlib application that runs on the worker nodes executes a simulation according to the parameters described in this file.

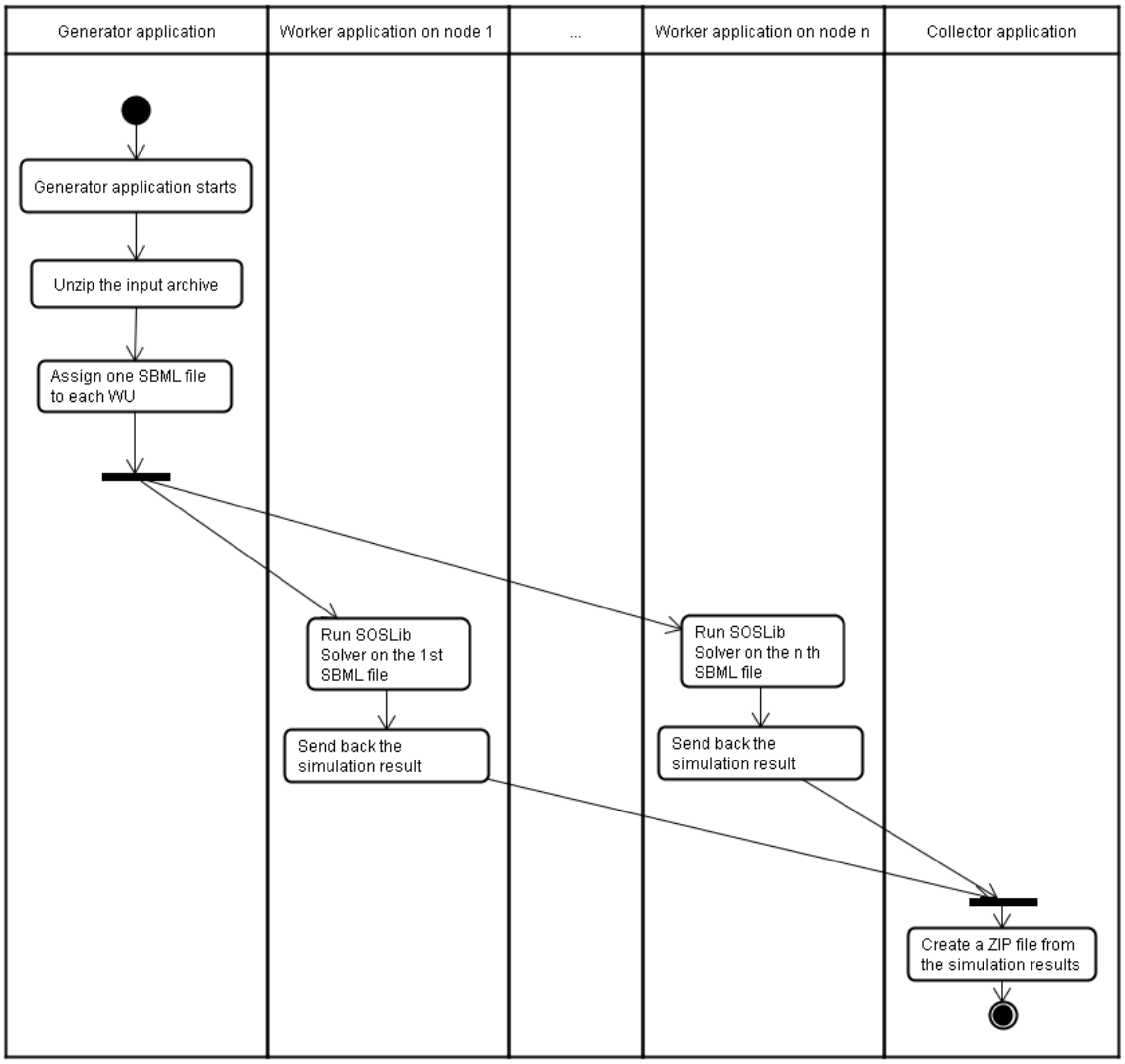
Generator application
Worker application on node 1
...
Worker application on node n
Collector application
Generator application starts
Unzip the input archive
Assign one SBML file to each WU
Run SOSLib Solver on the 1st SBML file
Send back the simulation result
Run SOSLib Solver on the n th SBML file
Send back the simulation result
Create a ZIP file from the simulation results

Figure 2: Control flow of the ported SIMAP application

3. **Result files:** the SOSlib application on the client nodes generates a text file that is the output of the simulation. These output files should be sent back to the server to be processed by the collector application.
4. **Output file:** the final output of the workflow is again a ZIP file that holds all of the result files.
5. **SOSlib files:** all of the files that are needed to execute the SBML ODE Solver Library application on a client computer. These are sent to the worker with the model files.
6. **Log files:** log files generated by the worker application.
7. **Batch script** file: the script file detailing the models that the SOSlib simulates.

The Generator, Application (SIMAP), Collector workflow is implemented and executed through the WS-PGRADE portal. This allows users to upload the required files and specify the different parameters on an easy to use web based form (a simple HTML form) before submission.

3.4 Summary

With Systems Biology simulations only taking a short amount of time to execute (but in large numbers), this approach obviously follows the *multiple run task approach*. Speedups are modest with better performance being delivered with multiple simulations per task. For example, for a simulation that takes approximately 20 seconds to run sending out 32 simulations per job yields a speed up of around 6 and 64 simulations per job around 15. Further performance testing is currently in progress to determine if this trend carries on with larger amounts of simulations. An alternative approach using the desktop grid technology CONDOR gives comparable speed ups for single task jobs. However, speedups are significantly better when using a multiple run task approach. The modest speed up is due to sharing the desktop PCs with other applications and other users across a campus, i.e., the CONDOR implementation reported in Wang et al. (2009) uses a set of PCs on a single LAN with no other PC users present. What is attractive is the 'speed' that a Grid application can be set up on an existing BOINC-based Grid infrastructure such as the Westminster Grid and that the application can then be used on any Grid that implements the SZDG technology (which is becoming more and more widespread. We now consider the challenges of this approach, and other similar Grid approaches, in M&S industry.

4 GRID COMPUTING: CHALLENGES FOR INDUSTRY

The previous section discussed an implementation of a Systems Biology simulation. Multiple runs were needed to get faster execution than running on a single PC. Key to this, however, was the simplicity of implementation. The fact that nothing needed to be installed means that any PC can be used to run the simulations. CSPs used in academia and industry virtually all require some installation. Further, the data used by models will be in some database or spreadsheet that needs to be locally accessible by the simulation. There is certainly the need for Grid computing as time within a simulation project is always being 'squeezed' (end of project data is fixed, delays in gathering data, building models, accessing experts) and simulation studies often generate more experiments and questions than expected. It is not unreasonable for individual simulation runs to take from a few minutes to over 10 hours. If each experiment has (say) 10 replications and a scenario has (say) 10 experiments then with just one 'thread' of investigation we have 80 hours (3+ days!). This does not even include Validation and Verification (V&V). Using Grid computing to support M&S CSPs in real-world industrial environments is certainly possible and examples include WINGRID (Mustafee and Taylor 2009) and SAKERGRID (Taylor et al. 2010). Some CSPs have basic Grid elements. Simul8, for examples, allows replications to be run using a limited form of Volunteer Computing. Indeed, in a joint research project between Brunel University, Simul8 and the EDGI project (University of Westminster), research is under way to determine the extent to which the SZDG technology can support a widespread Volunteer Computing implementation of Simul8. We now briefly present the challenges that Grid computing approaches must meet if widespread utilization of these in industry is to be achieved.

- M&S is a costly endeavor and any cost increase must be balanced with a clear return on investment, i.e., there must be a clear business case for developing a Grid solution.
- Users of CSPs tend to have an operational research or management science skillset. Most will not have formal experience of programming such as one might receive in a Computer Science course and possibly will have some programming language experience such as Visual Basic. The system must be deployed within the same set.
- Experience has shown that Grid-based applications development needs a balanced and committed team consisting of Grid application developers, end users, IT management and CSP vendors. It is often difficult to find the right 'language' to implement the right solution for the right problem.
- Companies using CSPs will have either basic or sophisticated IT management. Both have things to take into account. Basic IT management needs a simple system. Sophisticated IT management has rules and compliancy policies that must be adhered to. These policies differ from organiza-

tion to organization (particularly academic vs. industry). Both may heavily restrict the type of implementation.

- More and more CSPs use an experimentation manager or consultants will develop an experimentation manager to control experimentation. This is the user interface that must be deployed with, ideally, job submission technology in-built.
- Virtually all CSPs are 'black box' applications that run in Windows and must be installed in Windows Registry to access supporting programs such as DLLs, etc.
- Virtually all simulation models use data sources such as databases and/or spreadsheets that must be reliably launched, accessed and terminated during a run (e.g., if a link between a CSP and, say, Excel breaks then the infrastructure must be capable of detecting and recovering from this).
- Models can crash due to unforeseen errors in verification, dropped links between the CSP and supporting software, supporting software crashes.
- CSPs are typically licensed and Grid solutions may be restricted to the number of licenses an organization has.
- Security requirements may restrict what machines simulations can be run on.
- Currently, there are no Grid computing implementation standards. The lack of a clear set of implementation standards will increase the cost of Grid implementation (i.e., no 'off-the-shelf' solution).

5 DISTRIBUTED SIMULATION AND LARGE MODELS

In terms of software limitations, if a modeler wishes to build a large model in a single CSP then s/he is only restricted by the capabilities of the package (number of entities, time limit, event processing speed, event list size, etc.) In some cases it may be convenient to create a large model from a combination of existing and new models. Conveniently one may think of 'cutting and pasting' these models together in the visual modeling environment of a CSP. However, this may not be as convenient as it sounds or even possible. This is for several reasons:

- *Privacy*: A model representing several organizations may require privacy, i.e., each organization may not wish to reveal data and/or internal operations to other organizations. A single modeling approach would by implication reveal data through shared databases and spreadsheets and internal workings by model detail. Corporate secrecy would make this impossible.
- *Data transfer/access problems*. In some situations privacy may not be an issue. A single model running on a single computer will still need data from different organizational systems represented in the model. The options are to copy data from one organization to another or to link the data source to the model over an intranet or the Internet. Databases can be large and time consuming to copy and access can be slow. Depending on the system, data when copied is instantly out of date. Running a model using copies of organization data can therefore significantly increase the execution time of a simulation and/or be inaccurate.
- *Model composability problems*. A 'cut and paste' approach may not be as easy as it sounds. Variable name clashes, global variables and different validation assumptions are three examples of the many problems of this approach. Further, if an organization needs to update its model, it has to update the single model. How do we make sure that every organization has the correct version of the single model? What if the update causes problems in another part of the single model owned by another organization? Additionally, models developed in different CSPs are simply not compatible. One cannot transfer a model developed in one CSP into another without significant effort.
- *Execution Time*. Large models will most likely develop large event lists that must be processed and updated each time an event is executed. This can take a considerable amount of time. Worse, the processing capacity of even a high specification PC may not be enough to physically

cope as the actual CSP may have an upper limit on the event list size that can be swapped in and out of virtual memory.

An alternative to cut and paste is distributed simulation (Fujimoto 2000). Here models running in their CSPs on different computers interact, or interoperate, together in a single but distributed simulation. Each CSP maintains its own event list and CSPs interact via specialist distributed simulation software. Time over the distributed simulation is handled by variants of different time management synchronization algorithms (Fujimoto 1990). The functionality of the specialist distributed simulation software is defined in the IEEE 1516 High Level Architecture (IEEE 2010) (first released in 2000 and updated in 2010). In HLA terminology, a distributed simulation is called a *federation*, and each individual simulator (in our case the combination of a CSP and its model) is referred to as a *federate*. The HLA *Federate Interface Specification* (FIS) defines distributed simulation software termed a *Runtime Infrastructure* (RTI). A distributed simulation is therefore a federation composed of many federates interacting over a communication network via RTI software (Fujimoto and Weatherly 1996).

The HLA is complex and there are few people who completely understand M&S, CSPs, distributed simulation, the HLA and RTIs. Efforts are being made to simplify the theory and practice of CSP-based distributed simulation (Taylor et al. 2006). This is being led by the COTS Simulation Package Interoperability Product Development Group (CSPI PDG) under the Simulation Interoperability Standards Organization (SISO) and is producing a suite of CSP distributed simulation standards. As part of this activity a set of four Interoperability Reference Models (IRMs) have been defined to create a common frame of reference to assess the capabilities of particular approaches and to help practitioners and vendors achieve solutions to complex interoperability problems (Taylor et al. 2007; SISO 2010). These were created by following SISO's standards development process which involves several rounds of voting/balloting and refinement before a standard is formally recognized by SISO's Standards Activity Committee and its Executive Committee. It is important to note that the balloting and standard development involved software vendors. Without this buy-in from the vendors, the standard would not have significant impact. These need to be used in conjunction with CSP handler software that links a CSP to an RTI. A brief overview of the IRMs is now presented.

6 INTEROPERABILITY REFERENCE MODELS

IRMs are a set of simulation patterns or templates that enable modelers, vendors and solution developers to specify the interoperability problems that must be solved. 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.*

There are currently four different types of IRM. These are:

- **Type A**: Entity Transfer
- **Type B**: Shared Resource
- **Type C**: Shared Event

- **Type D**: Shared Data Structure

Briefly, IRM Type A Entity Transfer deals with the requirement of transferring entities between simulation models, such as an entity *Part* leaving one model and arriving at the next. IRM Type B Shared Resource refers to sharing of resources across simulation models. For example, a resource R might be common between two models and represents a pool of workers. In this scenario, when a machine in a model attempts to process an entity waiting in its queue it must also have a worker. If a worker is available in R then processing can take place. If not then work must be suspended until one is available. IRM Type C Shared Event deals with the sharing of events across simulation models. For example, when a variable within a model reaches a given threshold value (a quantity of production, an average machine utilization, etc.) it should be able to signal this fact to all models that have an interest in this fact (to throttle down throughput, route materials via a different path, etc.). IRM Type D Shared Data Structure deals with the sharing of variables and data structures across simulation models. Such data structures are semantically different to resources, for example a bill of materials or a common inventory.

As most distributed simulations with CSPs concern the transfer of entities between simulations, in this advanced tutorial we give a brief overview of the IRM Type A. Further details can be found in the full description of the standard (SISO 2010). There are currently three IRM Type A Sub-types. These are:

- **IRM Type A.1**: General Entity Transfer
- **IRM Type A.2**: Bounded Receiving Element
- **IRM Type A.3**: Multiple Input Prioritization

6.1 IRM Type A.1 General Entity Transfer

IRM Type A.1 General Entity Transfer represents the case where an entity e1 leaves activity A1 in model M1 at T1 and arrives at queue Q2 in model M2 at T2. This IRM is inclusive of cases where there are many models and many entity transfers (all transfers are instances of this IRM). This IRM does not include cases where (a) the receiving element is bounded (IRM Type A.2), and (b) multiple inputs need to be prioritized (IRM Type A.3). The IRM Type A.1 General Entity Transfer is defined as the transfer of entities from one model to another such that an entity e1 leaves model M1 at T1 from a given place and arrives at model M2 at T2 at a given place and T1 =< T2 or T1<T2. The place of departure and arrival will be a queue, workstation, etc. Note that this inequality must be specified.

6.2 IRM Type A.2 Bounded Receiving Element

Consider a production line where a machine is just finishing working on a part. If the next element in the production process is a buffer in another model, the part will be transferred from the machine to the buffer. If, however, the next element is *bounded*, for example a buffer with limited space or another machine (i.e., no buffer space), then a check must be performed to see if there is space or the next machine is free. If there is no space, or the next machine is busy, then to correctly simulate the behavior of the production process, the current machine must hold onto the part and *block*, i.e., it cannot accept any new parts to process until it becomes unblocked (assuming that the machine can only process one part at a time). The consequences of this are quite subtle. This is the core problem of the IRM Type A.2. If, for example, an entity e1 attempts to leave model M1 at T1 from activity A1, to arrive at model M2 at T2 in *bounded* queue Q2. If A1 represents a machine then the following scenario is possible. When A1 finishes work on a part (an entity), it attempts to pass the part to queue Q2. If Q2 has spare capacity, then the part can be transferred. However, if Q2 is full then A1 cannot release its part and must block. Parts in Q1 must now wait for A1 to become free before they can be machined. Further, when Q2 once again has space, A1 must be notified that it can release its part and transfer it to Q2. Finally, it is important to note the fact that if A1 is blocked the rest of model M1 still functions as normal, i.e., a correct solution to this problem must still allow the rest of the model to be simulated (rather than just stopping the simulation of M1 until Q2 has unblocked).

This IRM is therefore inclusive of cases where the receiving element (queue, workstation, etc.) is bounded. This IRM does not include cases where multiple inputs need to be prioritized (IRM Type A.3 – this is not discussed in this tutorial). Finally, a solution to this IRM problem must also be able to transfer entities (IRM Type A.1). The IRM Type A.2 is defined as the relationship between an element O in a model M1 and a bounded element Ob in a model M2 such that if an entity e is ready to leave element O at T1 and attempts to arrive at bounded element Ob at T2 then:

- If bounded element Ob is empty, the entity e can leave element O at T1 and arrive at Ob at T2, or
- If bounded element Ob is full, the entity e cannot leave element O at T1; element O may then block if appropriate and must not accept any more entities.
- When bounded element Ob becomes not full at T3, entity e must leave O at T3 and arrive at Ob at T4; element O becomes unblocked and may receive new entities at T3.
- T1=<T2 and T3=<T4.
- If element O is blocked then the simulation of model M1 must continue.

Note:

- In some special cases, element O may represent some real world process that may not need to block.
- If T3<T4 then it may be possible for bounded element Ob to become full again during the interval if other inputs to Ob are allowed.

6.3 IRM Type A.3 Multiple Input Prioritization

The IRM Type A.3 Multiple Input Prioritization represents the case where a model element such as queue Q1 (or workstation) can receive entities from multiple places. Let us assume that there are two models M2 and M3 which are capable of sending entities to Q1 and that Q1 has a First-In-First-Out (FIFO) queuing discipline. If an entity e1 is sent from M2 at T1 and arrives at Q1 at T2 and an entity e2 is sent from M3 at T3 and arrives at Q1 at T4, then if T2<T4 we would expect the order of entities in Q1 would be e1, e2. A problem arises when both entities arrive at the same time, i.e., when T2=T4. Depending on implementation, the order of entities would either be e1, e2 or e2, e1. In some modeling situations it is possible to specify the *priority* order if such a conflict arises, e.g., it can be specified that model M1 entities will always have a higher priority than model M2 (and therefore require the entity order e1, e2 if T2=T4). Further, it is possible that this priority ordering could be dynamic or specialized. This IRM is therefore inclusive of cases where multiple inputs need to be prioritized. This IRM does not include cases where the receiving element is bounded (IRM Type A.2). A solution to this IRM problem must also be able to transfer entities (IRM Type A.1). The IRM Type A.3 Multiple Input Prioritization is defined as the preservation of the priority relationship between a set of models that can send entities to a model with receiving queue Q, such that priority ordering is observed if two or more entities arrive at the same time. Note that the priority rules must be specified and the priority rules may change during a simulation if required for the real system being simulated.

7 CONCLUSIONS

Motivated by advances in distributed computing, 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 to acknowledge the contributions of Jun Wang, Xuan Liu, Qian Gao and David Gilbert of Brunel University (SIMAP) to the Systems Biology work, Chris Wood (SakerGrid) and the members of SISO's CSPI PDG.

REFERENCES

BOINC. 2011. Berkeley Open Infrastructure for Network Computing. Accessed March 24. boinc.berkeley.edu.

EDGeS. 2011. "Enabling Desktop Grids for e-Science." Accessed May 1. www.edges-project.eu.

EDGI. 2011. "Enterprise Desktop Grid Initiative." Accessed May 1. www.edgi-project.eu.

Foster, I., and C. Kesselman. 1998. *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann.

Fujimoto, R.M. 1990. "Parallel Discrete Event Simulation." *Communications of the ACM* 33(10)30-53.

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

Fujimoto, R.M., and R. M. Weatherly. 1996. "Time Management in the DoD High Level Architecture." In *Proceedings of the 10th Workshop on Parallel and Distributed Simulation Workshop*, 60-67. Washington, DC: IEEE Computer Society.

gUSE. 2011. "Grid User Support Environment." Accessed March 24. www.guse.hu.

IEEE. 2010. *IEEE 1516-2011 IEEE Standard for Modeling And Simulation (M&S) High Level Architecture (HLA)*. New York, NY: Institute of Electrical and Electronics Engineers.

Kacsuk, P., J. Kovacs, Z. Farkas, A. C. Marosi, G. Gombas, and Z. Balaton. 2009. "SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System." *Journal of Grid Computing* 7(4):439-461.

Kacsuk, P. 2011. "P-GRADE Portal Family for Grid Infrastructures." *Concurrency and Computation: Practice and Experience* 23(3): 235-245.

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

SETI. 2011. "Search for Extra Terrestrial Intelligence Project." Accessed March 24. setiathome.ssl.berkeley.edu/.

SOSlib. 2011. "SBML ODE Solver." Accessed May 1. www.tbi.univie.ac.at/~raim/odeSolver/.

SISO. 2010. *Standard for Commercial-off-the-shelf Simulation Package Interoperability Reference Models (SISO-STD-006-2010)*. Simulation Interoperability Standards Organization, Orlando, Florida.

SZDG. 2011. "SZTAKI Desktop Grid." Accessed March 24. www.desktopgrid.hu.

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. Straßburger, S. J. Turner, M. Y. H Low, and J. Ladbrook. 2007. "The SISO CSPI PDG Standard for Commercial Off-The-Shelf Simulation Package Interoperability Reference Models". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 594-602. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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,* edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 203-215. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers Inc.

Taylor, S. J. E, N. Mustafee, S. Kite, C. Wood, S. J. Turner, and S. Strassburger. 2010. "Improving Simulation through Advanced Computing Techniques: Grid Computing and Simulation Interoperability."

In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 216-230. Piscataway, New Jersey: Institute of Electrical and Electronic 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* 2009, Oxford, UK. 7-9 December 2009.

Zhang, J., N. Mustafee, J. Saville and S. J. E. Taylor. 2007. "Integrating BOINC with Microsoft Excel: A Case Study." In *Proceedings of the 29th Information Technology Interfaces Conference*, 733 – 738. Institute of Electrical and Electronics Engineers, Inc.

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 ICT Innovation 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**.**

MOHAMMADMERSIN GHORBANI is a PhD student in the ICT Innovation Group and Centre for Synthetic and Systems Biology, both in the School of Information Systems, Computing and Mathematics at Brunel University. He is studying methods for using Grid Computing to support researchers in Biology. His email address is mohammadmersin.ghorbani@brunel.ac.uk.

TAMAS KISS is a Senior Lecturer at the Department of Business Information Systems, and a researcher at the Centre for Parallel Computing at the School of Electronics and Computer Science, University of Westminster, London. His research interests include parallel and Grid computing, and he has extended experience in the area of legacy code deployment, interoperation of Grid systems, and application porting to service and desktop Grid systems. He led the design and development activities resulting in the Grid Execution Management for Legacy Code Architecture (GEMLCA) solution, now a Globus incubator project, within the UK EPSRC founded OGSA Testbed project. He contributed to the CoreGrid Network of Excellence project as the leader of the Legacy Code Wrapping and Deployment Methodologies Research Group within the Institute on Grid Systems, Tools and Environments, and led the Grid Application Support Service activity within the European EDGeS project. He is currently work package leader in the European EDGI (Enabling Grids for e-Science) and DEGISCO (Desktop Grids for International Scientific Collaboration) projects coordinating application porting and user support activities. He has been member of the program committees of several international grid and distributed computing conferences (e.g., PDP, EGI Technical Forum, IWSG, HealthGrid conference) and co-authored over 60 scientific papers published in journals and conference proceedings. His email address is t.kiss@westminster.ac.uk.

DANIEL FARKAS is Research Associate in CPC at the University of Westminster. His research interests include distributed and parallel computing, service and desktop grid computing. He works on the EDGI project and he is involved in application support and infrastructure management. His email address is d.farkas@westminster.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.

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.