2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Goal-Directed Grid-Enabled Computing for Legacy Simulations

Ernest H. Page, Laurie Litwin, Matthew T. McMahon, Brian Wickham, Mike Shadid, Elizabeth Chang
The MITRE Corporation
7525 Colshire Drive
McLean, VA, 22102, USA
epage,llitwin,mcmahon,bwickham,mshadid,ewchang@mitre.org

***Abstract*—We describe a middleware framework conceived to enhance the effectiveness and efficiency of existing simulation applications by providing three capabilities: (1) access to grid-based and cloud-based execution, (2) access to advanced Design of Experiments (DOE) methodologies such as simulation-based optimization, and (3) access to robust data processing and visualization. The framework has been applied to a variety of simulations in both commercial and open source programming languages employing both discrete and continuous modeling formalisms. A key design objective is to minimize the workload necessary to adapt a simulation for use with the framework. User experience to date reveals that the learning curve for the framework is reasonable, but further automation of key tasks would enhance the framework's utility.**

***Keywords*-Simulation-based optimization; grid computing**

I. Introduction

Simulation is a unique problem-solving technique, with unique challenges—particularly in the areas of performance and correctness. With respect to performance, for example, you typically need to conduct many, many simulation runs in order to obtain useful results. And this business of doing many, many runs can end up taking quite a long time. Fortunately, the issue of minimizing simulation runtime is well-studied from a variety of different perspectives. The clever use of statistics, for example, allows a modeler to cull several observations from a single simulation run without incurring the overheads associated with simulation start-up [1], and can also be employed to minimize the total number of runs needed to derive an estimate for a system variable, regardless of the number of system variables of interest, their interrelationships, or underlying distributions [2], [3]. You can build a model of your model, often called a *metamodel*, that retains the essential features of your model, but with many fewer variables and therefore much shorter runtimes [4]–[6]. You can also do clever things with computers to reduce simulation runtimes. If you have enough of them, for example, you can run each replication on a different computer (since the replications tend to be independent) [7]–[9]. Or, if you are very ambitious, you can break an individual replication apart and execute it using parallel processors [10].

With respect to correctness, one of the first things simulation modelers come to understand is that "correctness" is always a negotiation between the questions being asked with the model and the model representation itself. Furthermore, simulations are *descriptive* rather than *prescriptive*. That is, the results of a simulation merely provide estimates of the value of system output variables, for some given combination of input variable values, after some amount of time passes in the system. The simulation technique doesn't tell you if any of these values are "the best" in any sense. As an analyst, it is up to you to explore the landscape of estimates that can be generated by your simulation model. In some cases, the entire landscape can be generated. In many cases, the possible combinations of model input values is too vast to explore completely. In this case, techniques from Design of Experiments (DOE) are available to assist the analyst in efficiently navigating the landscape [11].

But what if your particular simulation package doesn't provide explicit support for these enablers? What if your simulation has long runtimes and you need to perform thousands of runs? What if your only computing equipment is a laptop (and a network connection)?

This article describes a computing framework, initiated through MITRE Independent Research and Development (IR&D) funding, that seeks to enhance the effectiveness and efficiency of simulation-based analysis and experimentation by providing existing simulation applications with access to three performance enablers:

- The capability to escape the computing limitations of the engineer's desktop, via grid and cloud computing.
- Goal-directed replication management, via broad support to Design of Experiments (DOE).
- Access to the robust marketplace of data processing and visualization tools.

Our motivation for developing the framework was to be able to enhance simulation use across an enterprise. The article is organized as follows. In Section II, we describe the computing framework, the motivation for its development, the principles underlying its formulation, and its current architecture. Section III presents a brief case study of the application of the framework and a discussion of some lessons learned. Related work in grid-based support for simulation is reviewed in Section IV. Conclusions and a discussion of future directions for our framework appear in Section V.

978-0-7695-4691-9/12 $26.00 © 2012 IEEE
DOI 10.1109/CCGrid.2012.60

II. MITRE Elastic Goal-Directed Simulation Framework

In addition to performance and correctness, yet another defining aspect of simulations—at least in the defense arena—is that they tend to involve significant investment in their development. Because of this, they also tend to have very long shelf-lives, and are often re-purposed to support new objectives. A particularly interesting version of the "paradigm of re-purposing" is the simulation interoperability era that began within the United States Department of Defense (DoD) during the 1980s and resulted in the development and wide-spread adoption of High Level Architecture (HLA, see [10], [12] for discussions of the history of simulation interoperability).

The impetus for the HLA was the need to develop multi-Service simulations to support analysis or training (sometimes both). In almost all cases, individual Service simulations existed, themselves costly investments. Given budgetary constraints, there became a natural curiosity regarding whether these existing simulations could somehow be integrated to form a larger simulation that would efficiently and effectively serve the multi-Service objectives. During the 1980s, *ad hoc* solutions supporting the integration of simulations proliferated, and were viewed as successful enough to justify the development of a standard in the mid-1990s, the HLA.

The general problem confronted here is the one of reuse—whether it is best to conceive, design and construct a new simulation expressly for newly imposed objectives/requirements, or whether to conjoin existing, so-called *legacy* simulations to accommodate new missions. The wisdom of coupling two simulations together that were not conceived to be coupled together can only be judged on a case-by-case basis. But certainly, given the costs associated with the development of new simulations and the realities of budgetary constraints, developing standards that provide the *option* to economically re-purpose legacy simulations seems a sensible course.

Inspired by the development and success of the HLA as an enabler for legacy simulations, and by recent advances in grid-enabled simulation (described in Section IV), we initiated the development of the MITRE Elastic Goal-Directed Simulation Framework, or MEG, for short.

Our objective in the development of MEG is modest: develop a middleware framework that *adds value to existing simulation applications*. As Computer Scientists working in parallel and distributed simulation, with access to lots of computing power—whose cycles were not always heavily utilized—we were initially interested in finding a convenient way for MITRE staff to access this computing with their simulations. Developing and offering grid-based replication management seemed a fairly obvious, and easy to accomplish, approach. Our initial thinking was that we could simply install Condor [13] (or another grid management framework) on our compute clusters, provide a few tutorials to the engineers on how to submit and monitor jobs, and that would be enough. But, upon reflection, we convinced ourselves that we could, and should, do more: as long as we are providing access to compute cores for executing replications, is there anything we can offer from the experimental design domain to help identify which replications to execute? Are there any tools and techniques from the robust data management and visualization marketplace that we could integrate within our framework to help engineers better understand and communicate the results of their models?

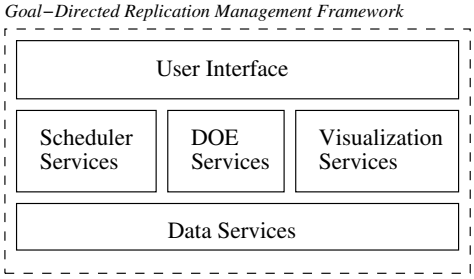

Figure 1. MEG Architecture

A. MEG Architecture

The overall architecture for MEG is depicted in Figure 1. The framework consists of approximately 20,000 lines of code, primarily Java, and is comprised of five principal sets of services: (1) user interface, (2) job scheduling (replication management), (3) DOE support, (4) visualization, and (5) data management. We briefly describe each of these below.

1) User Interface: The MEG Graphical User Interface (GUI) serves two primary functions: (1) it insulates the user from developing and manipulating the litany of scripts associated with typical grid computing software, and (2) it provides a persistent workspace and dashboard for experiment design and monitoring.

2) Scheduler Services: The MEG scheduler services architecture is depicted in Figure 2. Our objective is to support the widest range of grid schedulers available, and to permit a user to submit jobs to MEG without having to select a particular target cluster for their execution. To support this objective, we have adopted the Gridway *metascheduler* [14] which is part of the Globus Toolkit. Gridway works with many of the common Distributed Resource Management (DRM) systems, inlcuding Condor, Sun Grid Engine and the Portable Batch System (PBS). MEG currently supports execution on four named compute clusters in MITRE, two Linux clusters (one scheduled by Condor, the other by Sun Grid Engine) and two windows clusters (both scheduled through Condor). In addition, MEG employs native threading to schedule replication execution on a user's multi-core

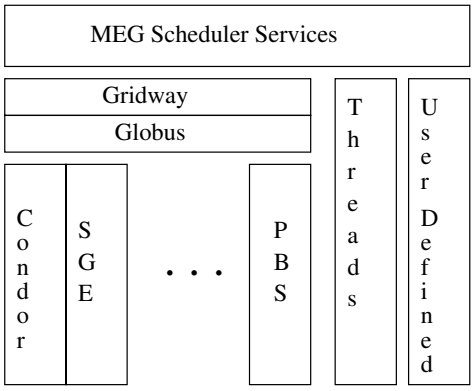

Figure 2. MEG Scheduler Services Architecture

desktop.

3) Design of Experiments Services: The MEG DOE services facilitate control of simulation input parameters and creation of simulation experiments. Once the model, input files, output handlers, and parameter definitions have been configured, a user may select from a suite of built-in *designers* to guide replication management:

- A set of "standard" Genetic Algorithms (GA) - used to optimize a model with a defined single scalar objective function, and configurable with a variety of standard GA choices.
- Constrained Genetic Algorithm - used to optimize a model where the input parameters or model outputs are subject to constraints. The constraints are referenced by name and defined via standard scripting during MEG configuration.
- Island-Model GA - used to optimize many populations of model parameters as independently evolving islands, with migration of parameters between islands at a configurable rate.
- Recursive Random-Descent Designer - based on the heuristic developed in [15].
- Full-Factorial Designer - used to conduct parameter sweeps and sensitivity studies with user-defined parameter ranges and step sizes.
- Plugin Designer - used for model validation, partial-factorial experiments, and "what-if" scenarios mediated by submitting sets of parameters directly through the MEG interface via spreadsheet.

In addition, MEG provides an API for the development of new designers, with methods supporting: (1) control of parameters - the ability to access and modify defined model inputs; (2) control of execution - the ability to notify the scheduler that a model needs to be run (as when a new GA population is ready for execution), or to query the scheduler for job completion, (3) access to objective function values - either the single scalar objective value or multiple objectives, as defined; and (4) access to constraint information - how many and which constraints were violated during a model execution.

4) Visualization Services: MEG supports two broad classes of visualization services. First, MEG's browser-based interface allows users to conveniently access grid status reports such as those provided by Ganglia [16] and Nagios [17]. The second class of visualization services supports the user's ability to interpret simulation results both as the experiment is running, and post-run. MEG provides a database that supports the Odata standard [18] which can be exploited by tools such as Excel (PowerPivot), QlikView [19] and Tableau [20]. MEG provides services that translate a user's native output files into the MEG generic data format in realtime (see below).

5) Data Services: The MEG data services are based on an application called *Data Gin* which runs as a set of web services on any operating system that supports Java. MEG users describe the nature of their output, e.g., text-based comma-separated-value (CSV) files, mySQL database, etc., within the MEG GUI. During model execution, the Data Gin registers a file system "watcher" that detects creation/modification of output files, and automatically reformulates the data appearing in these files into a MEG generic data format. The Data Gin has the ability to infer data typing, and generates the necessary tags and metadata, per run, to facilitate efficient execution of user queries.

B. MEG Concept of Operations

The typical MEG user is an engineer or scientist, with an existing simulation, often written in a commercial simulation package, who is in need of convenient access to large-scale computing or automated support for sophisticated experiment design techniques, such as simulation-based optimization. Once the simulation has been adapted as outlined above, a typical MEG use case has the following phases:

1) A user logs into the MEG system. Currently, this consists of loading a web page into a browser.
2) The user identifies the simulation model to be run, its input files and their locations. MEG automatically generates the input file templates necessary to support replication management based on the user's input files.
3) The user defines an experiment by specifying model input variables to be manipulated by MEG, and by constructing, if desired, an objective function in terms of model output variables, and selecting a search technique to guide MEG's replication management.
4) The user may view the status of available computing resources, and select one of these for his/her experiment. Or the user may allow MEG to select appropriate computing resources from those available. The user may schedule jobs for execution immediately, or for some specific time in the future.
5) MEG begins the experiment by configuring model input files, as appropriate, and moving the necessary

files into locations suitable to support execution on the target computing platforms.

6) As replications are executed, results are consolidated and made available to support realtime visualization of the model response surface being generated. Users may also subscribe to a variety of notification services to monitor the progress of their experiment.
7) At the conclusion of the experiment, model results are delivered to a location specified by the user.

To accommodate users with varying technical backgrounds, some of whom have limited experience with (and patience for) programming, scripting and command-line interaction, MEG has been developed with a rich graphical user interface (GUI) that provides dialogue-driven "wizard" support for most of the commonly performed tasks. Our experience and informal assessments, to date, indicate that users have found the learning curve associated with MEG to be reasonably low.

C. Adapting a Simulation for Use with MEG

The primary difficulties confronted by first-time users involve making the necessary changes to their simulation to work within the MEG framework. Our goal as framework designers is to minimize the work necessary to adapt a simulation, with the recognition that completely automating the process may never be achievable. In its current form, there are two primary capabilities that a simulation *must* have in order to work with MEG. First, the simulation must be capable of being launched and able to run to completion without any user intervention/interaction. Generally, this implies that the model must support the ability to be launched from a command-line interface (rather than a Graphical User Interface) and that any input to the running model can be read from an input file. Second, since the physical location for model execution is not generally known *a priori* in grid computing contexts, all file/data paths specified in the simulation must be relative (or parameterized) paths.

In addition, for users taking advantage of MEG's support for advanced DOE methods such as simulation-based optimization, a file that identifies the model input variables, their types and ranges must be provided. A routine that parses model outputs to compute the value of a user-defined objective function must also be developed. In both these cases, MEG provides assistance to a user in the development of these files/routines.

We adopt the "CSP-Preinstalled Approach" defined in [21]. A job launched by MEG may be *arbitrarily complex*, in that it involves many separate stages, and includes both pre-run and post-run operations, including the installation/removal of ancillary software to support model execution. As long as these operations can be done in the user's space, without user intervention, or need for root privileges, all is well. If, however, the ancillary software required to support the run needs manual intervention for its installation, or must be installed with root privileges, the software must be pre-installed on the target compute platform. MEG can take advantage of the Condor classAd mechanism (and similar capabilities in other grid schedulers) to identify compute platforms that have the necessary pre-installations.

D. MEG Design Principles

The development of MEG has been driven by five primary design principles:

- *Learn by Doing*. Get the framework in users hands early and often. Discover system requirements. Don't design in a vacuum.
- *Provide Transparency*. The reason for MEG faults and failures must be visible to end users. Even if they cannot repair/correct MEG themselves, user satisfaction with—and ultimately, their adoption of—the framework will benefit from transparent failures. Most users can accept working with system that is under development. They will tolerate occasional system failures, as long as the cause of the failures is communicated, and the repair can be effected in a reasonable time.
- *Support any modeling language or formalism*. MITRE engineers employ dozens of modeling languages (commercial, open source, general and special purpose) and a range of formalisms in both the discrete and continuous regimes. The framework should not preclude the use of any of these.
- *Provide a Low Barrier for Entry*. The framework must be sufficiently easy for the first-time user. The amount of work needed to initially adapt a simulation for use with MEG must be kept to a minimum. Automation is essential here. However, during the early phases of framework development this may mean that the MEG team must do much of the work needed to manually integrate a new simulation with MEG, until this workload can be automated.
- *A Good Idea Applies to Itself.* We will use simulation and simulation-based optimization to identify the optimal operating characteristics for any MEG installation.

While we believe the framework may eventually provide us with a platform to investigate some of the open problems in distributed simulation (see discussion in [12]), our initial goal is to provide a practical simulation-support capability.

III. Case Study: Adapting SLX Models of the National Airspace System

Some of the earliest, and most instructive—from a design perspective—users for MEG were engineers from MITRE's Center for Advanced Aviation System Development (CAASD). CAASD engineers employ a tremendous variety of models to study all aspects of the National Airspace System (NAS) in support of the Federal Aviation Administration (FAA). They have very well-established tools

and analysis methods. Among these, are a suite of tools written in the SLX programming language [22]. As an early proof-of-concept for our approach, we adapted two of these models, runwaySimulator [23] and systemwideModeler [24] for use with MEG.

Prior to our involvement, runwaySimulator and systemwideModeler analysts initiated model runs through extensions of the native SLX GUI running either on a set of Windows machines with a dedicated SLX license server, or on the analyst's local machine by accessing a network license server. To support model execution under MEG, we created a Condor pool associated with the dedicated-license machines and additional Virtual Machines (VMs) that access the network license server. The current "SLXGrid" has 32 compute cores, 24 of which are used to execute SLX jobs and 8 of which are dedicated to Condor management functions.

MEG integration for runwaySimulator was accomplished by modifying the runwaySimulator GUI to launch jobs through MEG (the user does not interact with the MEG GUI, but rather a button on the runwaySimulator GUI initiates a command-line launch of MEG). The analysis process associated with runwaySimulator involves conducting many related runs around specific design points, with specific dependencies between these runs. We incorporated Condor DAGMan [25] to describe and enforce these job dependencies.

MEG integration for systemwideModeler involved creating a templating construct for the tens of input files associated with the simulation, and developing a post-run system that supported the post-run data processing needs for the model—systemwideModeler generates several large flat-files which are parsed and loaded into an Oracle database. In addition, to support simulation-based optimization using systemwideModeler, we developed a Python script that parses the model output files, and computes an objective function value, e.g., total NAS-wide delay, which is made available to the MEG designers (part of the DOE services).

In terms of the performance of the SLXGrid, operational use of the the grid to support runwaySimulator runs show typical speedups of approximately 20 (over 24 cores). Instrumented runs show that the scaling limit is due to licence server contention. Through a temporary licensing agreement with Wolverine Software, we conducted an experiment with systemwideModeler and 138 nodes, and illustrated the ability to run 16,000 runs over a weekend, an approximately 100-fold speedup over non-MEG-based operations.

IV. Related Work

Some of the earliest versions of MEG were based on the design concepts used in the Unified Search Framework (USF) [15]. USF provides a framework where multiple heuristic search methods may be defined and used independently or cooperatively to guide a search using simulation replications launched onto a distributed computing platform using Java Remote Method Invocation [26]. USF has been used to demonstrate the effectiveness of the Recursive Random Search (RRS) technique on the problem of determining optimal values for network protocol parameters [27], [28].

Taylor et al. [21] discuss general principles for grid-based simulation support frameworks, with particular attention to the difficulties associated with deploying commercial simulation packages in grid environments. They present two case studies. The first employs Condor to distribute a Java-based simulation for systems biology, SIMAP. Here, the SIMAP application is directly adapted to submit jobs into the grid. A query-based protocol, implemented using web services, allows SIMAP to monitor the progress of jobs executing under Condor control. Heuristics are defined to throttle query frequencies in order to maintain grid performance. Similarly, the authors observe that when individual runtimes of the simulation replications are small (e.g., less than one minute) then batching is needed to overcome Condor scheduling overheads and attain scalable grid performance. For 4096 jobs (where each individual job has 20-second runtime and requires 1MB memory/disk space), the authors observe a maximum speedup of 12 using 32 cores. Additional tests indicate that this scaling could be improved if more than one Condor agent were assigned to handle the requests from SIMAP. Taylor et al. [29] describe an implementation of SIMAP on the SZTAKI Desktop Grid [30] running at the University of Westminster, reporting speedups of 15 on a grid with approximately 1600 machines registered in a volunteer computing context.

The second case study from [21], which is further detailed in [31], uses a commercial grid engine, SakerGrid [32], to distribute the execution of simulations written using the Flexsim commercial simulation package [33]. Techniques for dealing with secure execution, access to runtime licences, additional third-party software interactions, jobs with varying priority levels, and jobs that fail to run to completion are discussed. The authors report a 10-fold speedup using 10 processors for 40 Flexsim jobs, where each individual job has a 7.5 minute runtime.

Issues confronted in the "grid enabling" of Witness [34] within an industrial setting are discussed in [35].

V. Conclusions

Motivated by the success of the *value-added middleware* approach taken by the HLA, and recent work in grid-enabled simulation, we have developed a software framework that seeks to enhance the effectiveness and efficiency of simulation-based analysis and experimentation by providing existing, so-called *legacy*, simulation applications with access to three enablers:

- The capability to escape the computing limitations of the engineer's desktop, via grid and cloud computing.

- Goal-directed replication management, via broad support to Design of Experiments (DOE).
- Access to third-party data processing and visualization tools.

MEG is a nascent capability. To date, we have applied the framework to ten modeling activities, and have integrated simulations written in SLX, C, Java, RePast, iThink and Matlab. During the course of our efforts, we have tackled many of the issues noted in [21], [29], [31], [35] such as dealing with GUIs, license servers, and execution batching to enhance speedup. Our informal assessments of the framework suggest that the learning curve for new users is reasonable. We are focusing future efforts on further reducing the work necessary to integrate a simulation, including:

- Incorporating support for GUI-based simulations (through an ability to simulate GUI interactions via scripts such as those provided by automated GUI testing frameworks [36]).
- Developing routines to parse models and identify their input variables. (To support this feature, modelers may need to tag variable declarations with special comments.)

We are also hoping to extend MEG by:

- Incorporating support for the generation of metamodels.
- Developing and evaluating alternative approaches for dealing with "Big Data."

Most of all, we are actively seeking new users for the framework within MITRE. And we are hopeful that we will be able to make MEG available to the public domain in the near term.

ACKNOWLEDGMENTS

This work is supported through MITRE Independent Research and Development, and a variety of MITRE sponsor-funded efforts. The views expressed are those of the authors and do not reflect official positions of the MITRE Corporation or the U.S. Government. The authors thank Glenn Roberts, Zach Furness and Bahaa Fam for their vision and enthusiastic support for the research, and Pete Kuzminski, John Barrer and Billy Baden for their valuable feedback from a user's perspective. We also thank the referees for their careful reading and instructive comments on previous drafts.

REFERENCES

[1] C. Alexopoulos and A. F. Seila, "Output data analysis," in *Handbook of Simulation*, J. Banks, Ed. John Wiley and Sons, 1988, ch. 7, pp. 225–272.

[2] O. Balci and R. G. Sargent, "Validation of simulation models via simultaneous confidence intervals," *American Journal of Mathematical and management Sciences*, vol. 4, no. 3-4, pp. 375–406, 1984.

[3] S. Juneja and P. Shahabuddin, "Rare event simulation techniques: An introduction and recent advances," in *Handbooks in Operations Research and Management Science: Simulation*, S. G. Henderson and B. L. Nelson, Eds. North Holland, 2006, vol. 13, ch. 11, pp. 291–350.

[4] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *ASME Journal of Mechanical Design*, vol. 129, no. 4, pp. 370–380, 2007.

[5] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodeling techniques under multiple modeling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, 2001.

[6] R. Jin, X. Du, and W. Chen, "The use of metamodeling techniques for optimization under uncertainty," *Structural and Multidisciplinary Optimization*, vol. 25, no. 2, pp. 99–116, 2003.

[7] V. C. Bhavsar and J. R. Isaac, "Design and analysis of parallel monte carlo algorithms," *SIAM Journal on Scientific and Statistical Computing*, vol. 8, pp. s73–s95, 1987.

[8] P. Heidelberger, "Discrete event simulations and parallel processing: Statistical properties," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 6, pp. 1114–1132, 1988.

[9] Y.-B. Lin, "Parallel independent replicated simulation on a network of workstations," in *Proceedings of the 1994 Workshop on Parallel and Distributed Simulation*. Edinburgh, Scotland: IEEE, 1994, pp. 73–80.

[10] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*. John Wiley and Sons, 2000.

[11] J. P. Kleijnen, "Experimental design for sensitivity analysis, optimization, and validation of simulation models," in *Handbook of Simulation*, J. Banks, Ed. John Wiley and Sons, 1988, ch. 6, pp. 173–223.

[12] E. H. Page, "Theory and practice for simulation interconnection: Interoperability and composability in defense simulation," in *Handbook of Dynamic System Modeling*, P. A. Fishwick, Ed. Chapman and Hall/CRC, 2007, ch. 16.

[13] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the grid," in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, A. Hey, and G. Fox, Eds. John Wiley and Sons, 2003, ch. 11, pp. 299–335.

[14] Globus Alliance, "Gridway home," http://dev.globus.org/wiki/GridWay, 2012, [Online; accessed 13-February-2012].

[15] T. Ye and S. Kalyanaraman, "A unified search framework for large-scale black-box optimization," ECSE Department, Rensselaer Polytechnic Institute, Tech. Rep., 2003.

[16] Ganglia, "Ganglia monitoring system home," http://ganglia.sourceforge.net/, 2012, [Online; accessed 13-February-2012].

[17] Nagios, "Nagios home," http://www.nagios.org/, 2012, [Online; accessed 13-February-2012].

[18] Odata, "Open data protocol home," http://www.odata.org/, 2012, [Online; accessed 13-February-2012].

[19] QlikTech International AB, "QlikView home," http://www.qlikview.com/, 2012, [Online; accessed 13-February-2012].

[20] Tableau Software, "Tableau home," http://www.tableausoftware.com/, 2012, [Online; accessed 13-February-2012].

[21] S. J.E.. Taylor, N. Mustafee, S. Kite, C. Wood, S. J. Turner, and S. Straßburger, "Improving simulation through advanced computing techniques: Grid computing and simulation interoperability," in *Proceedings of the 2010 Winter Simulation Conference*. Baltimore, MD: IEEE, 2010, pp. 216–230.

[22] J. O. Henriksen, "Introduction to SLX," in *Proceedings of the 1997 Winter Simulation Conference*. Atlanta, GA: IEEE, 1997, pp. 559–566.

[23] J. Barrer, "A new method for modeling complex runway systems," in *Proceedings of the 7th USA/Europe Air Traffic Management Research and Development Seminar*, Barcelona, Spain, 2007.

[24] W. A. Baden, Jr., D. J. Bodoh, A. G. Williams, and P. C. Kuzminski, "systemwideModeler: A fast-time simulation of the NAS," in *Proceedings of the Integrated Communications, Navigation and Surveilance Conference*. Herndon, VA: IEEE, 2011, pp. F4–1 – F4–6.

[25] Condor, "DAGMan home," http://research.cs.wisc.edu/condor/dagman/, 2012, [Online; accessed 13-February-2012].

[26] Oracle Corporation, "Java remote method invocation home," http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html, 2012, [Online; accessed 5-January-2012].

[27] T. Ye and S. Kalyanaraman, "A recursive random search algorithm for large-scale network parameter configuration," in *SIGMETRICS '03*. San Diego, CA: ACM, 2003, pp. 196–205.

[28] T. Ye, H. T. Kaur, S. Kalyanaraman, and M. Yuksel, "Large-scale network parameter configuration using an on-line simulation framework," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 777–790, 2008.

[29] S. J.E.. Taylor, M. Ghorbani, T. Kiss, D. Farkas, N. Mustafee, S. Kite, S. J. Turner, and S. Straßburger, "Distributed computing and modeling & simulation: Speeding up simulations and creating large models," in *Proceedings of the 2011 Winter Simulation Conference*. Phoenix, AZ: IEEE, 2011, pp. 161–175.

[30] Sztaki, "Desktop grid home," http://szdg.lpds.sztaki.hu/szdg/, 2012, [Online; accessed 5-January-2012].

[31] S. Kite, C. Wood, S. J. E. Taylor, and N. Mustafee, "SakerGrid: Simulation experimentation using grid enabled simulation software," in *Proceedings of the 2011 Winter Simulation Conference*. Phoenix, AZ: IEEE, 2011, pp. 2283–2293.

[32] Saker Solutions, "Homepage," http://www.sakersolutions.com/Index.html, 2012, [Online; accessed 5-January-2012].

[33] Flexsim, "Homepage," http://www.flexsim.com/, 2012, [Online; accessed 5-January-2012].

[34] Lanner, "Witness home," http://www.lanner.com/en/witness.cfm, 2012, [Online; accessed 6-January-2012].

[35] N. Mustafee, A. Alstad, B. Larsen, S. J. E. Taylor, and J. Ladbrook, "Grid-enabling FIRST: Speeding up simulation applications using WinGrid," in *Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications*. Malaga, Spain: IEEE, 2006, pp. 157–164.

[36] Wikipedia, "List of GUI testing tools — wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/List_of_GUI_testing_tools, 2012, [Online; accessed 13-February-2012].