# High Speed Simulation Analytics

Simon J E Taylor and Anastasia Anagnostou
Modelling & Simulation Group
Department of Computer Science
Brunel University London
Uxbridge, Middx, UB10 8JE, UK
simon.taylor@brunel.ac.uk, anastasia.anagnostou@brunel.ac.uk

Tamas Kiss
Centre for Parallel Computing
Department of Computer Science
University of Westminster
London, W1W 6XH, UK
t.kiss@westminster.ac.uk

Abstract

Simulation, especially Discrete-event simulation (DES) and Agent-based simulation (ABS), is widely used in industry to support decision making. It is used to create predictive models or Digital Twins of systems used to analyse what-if scenarios, perform sensitivity analytics on data and decisions and even to optimise the impact of decisions. Simulation-based Analytics, or just Simulation Analytics, therefore has a major role to play in Industry 4.0. However, a major issue in Simulation Analytics is speed. Extensive, continuous experimentation demanded by Industry 4.0 can take a significant time, especially if many replications are required. This is compounded by detailed models as these can take a long time to simulate. *Distributed Simulation* (DS) techniques use multiple computers to either speed up the simulation of a single model by splitting it across the computers and/or to speed up experimentation by running experiments across multiple computers in parallel. This chapter discusses how DS and Simulation Analytics, as well as concepts from contemporary e-Science, can be combined to contribute to the speed problem by creating a new approach called High Speed Simulation Analytics. We present a vision of High Speed Simulation Analytics to show how this might be integrated with the future of Industry 4.0.

Keywords: Big Data Analytics, Cyber-physical systems, Industry 4.0, Digital Twins and Smart environments.

## 1. Introduction

Analytics can be defined as the extensive use of data, analytical techniques, models and fact-based management to drive decisions and actions (Davenport & Harris, 2007). Building on this Lustig introduces three types of analytics (Lustig, Dietrich, Johnson, & Dziekan, 2010): descriptive, predictive and prescriptive. Descriptive Analytics approaches analyse business performance on from a purely data perspective. Predictive Analytics techniques create explanatory and predictive models using both data and mathematics techniques to investigate and explain relationships between business outputs (outcomes) and data inputs. Prescriptive Analytics builds on this by evaluating alternative actions or decisions against a complex set of objectives and constraints.

Discrete-event simulation (DES) and Agent-based simulation (ABS) are widely used in industry to support decision making. These techniques are clearly cornerstones of Predictive and Prescriptive Analytics in that these techniques are used to create predictive models of systems that can be used to analyse what-if scenarios, perform sensitivity analytics on data and decisions and even to optimise the impact of decisions. In Industry 4.0

Simulation-based Analytics, or just Simulation Analytics, techniques have a major role to play in predictive and prescriptive decision making. For example, in these industrial cyber-physical systems, a simulation (or digital twin) might be constantly updated from the physical elements of the system and constantly runs in the "cloud" to predict and prescribe system behaviour (e.g. to balance manufacturing, to anticipate and prevent breakdowns, to plan and react to changes in customer/supplier behaviour, etc.) Further, these could create novel simulation applications (e.g. perpetual simulations that are always on and provide instant, pre-computed answers, symbiotic simulations that take real-time data and monitor real-world Key Performance Indicators (KPIs) against simulated ones to constantly improve system performance, etc.)

However, a major issue in Simulation Analytics is speed. Extensive, continuous experimentation can take a significant time, especially if many replications are required. This is compounded by detailed models as these can take a long time to simulate. Continuous data updates may also extend this time as statistical distributions within models need to be updated prior to simulation. For example, a detailed digital twin of a factory might take an hour (or more) to be updated and simulated. Each experiment might require (for example) 10 replications. One experiment there can take 10 hours. The goal of experimentation might be to explore efficient manufacturing strategies (e.g. a factory might have a complex product mix with several flexible routes through the machining processes of that factory) to recommend what actions should be taken within the next planning horizon (e.g. a week). This could result in several scenarios, each with multiple parameters with many values. Arbitrarily, if we say this results in 100 experiments (each with 10 replications on a model that takes an hour to simulate) then total experimentation time would be 1000 hours or around 42 days. If the planning horizon is one week then it is clearly impossible to perform the experimentation in support of this. Contemporary simulation typically uses a single computer to execute simulation. However, *Distributed Simulation* (DS) uses multiple computers to either speed up the simulation of a single model by splitting it across the computers and/or to speed up experimentation by running experiments across multiple computers in parallel. Naively, if we had, for example, 100 computers at our disposal then we could potentially speed up simulation experimentation 100 times. In the above scenario we could therefore complete the experimentation in 10 hours. In reality, various computing and networking factors reduce the efficiency of an implementation. However, major speedup is still possible.

How can we achieve High Speed Simulation Analytics? To answer this question we first review advances in DS. We then discuss one aspect of DS, high speed simulation experimentation, and how a cloud computing can be used to deliver on demand speed up. Building on these concept, we then "borrow" from contemporary e-Science to present a vision of High Speed Simulation Analytics for the future.

## 2. Distributed Simulation

DS has contributed to major successes in the simulation of large systems in defence, computer systems design and smart urban environments. The field comes from two communities (Fujimoto, 1990; 2016): the *Parallel Discrete Event Simulation* (PDES) community that focussed on how to speed up simulations using multiple processors in high performance computing systems and the *DS* community that uses PDES techniques to interconnect simulations together over a communication network. Essentially, the main goals of DS are to use parallel and distributed computing techniques and multiple computers to speed up the execution of a simulation program and/or to link together simulations to support reusability (Fujimoto, 2000). Some authors have also used DS to refer to approaches that run simulation experiments and/or replications on distributed computers in parallel with the goal of reducing the time taken to analyse a system (Heidelberger, 1986).

To reflect these various influences and goals, the following “modes” of DS can be identified (Figure 1):

- Mode A: to speed up a single simulation.
    - A model is subdivided into separate models that are simulated on different computers and interact via a communications network; speed up arises from the parallel execution of the separate simulations.
- Mode B: to link together and reuse several simulations.
    - Several simulations running on different computers are linked together to form a single simulation again with interactions between models carried out via a communications network; larger models beyond the capability of a single computer can be created. This mode enables model reuse.
- Mode C: to speed up simulation experimentation.
    - Experiments are run in parallel using multiple computers coordinated by some experimentation manager via a communication network; the parallel execution of simulation runs speeds up the experimentation thereby reducing experimentation time or increasing the number of simulation experiments possible in the same timeframe.

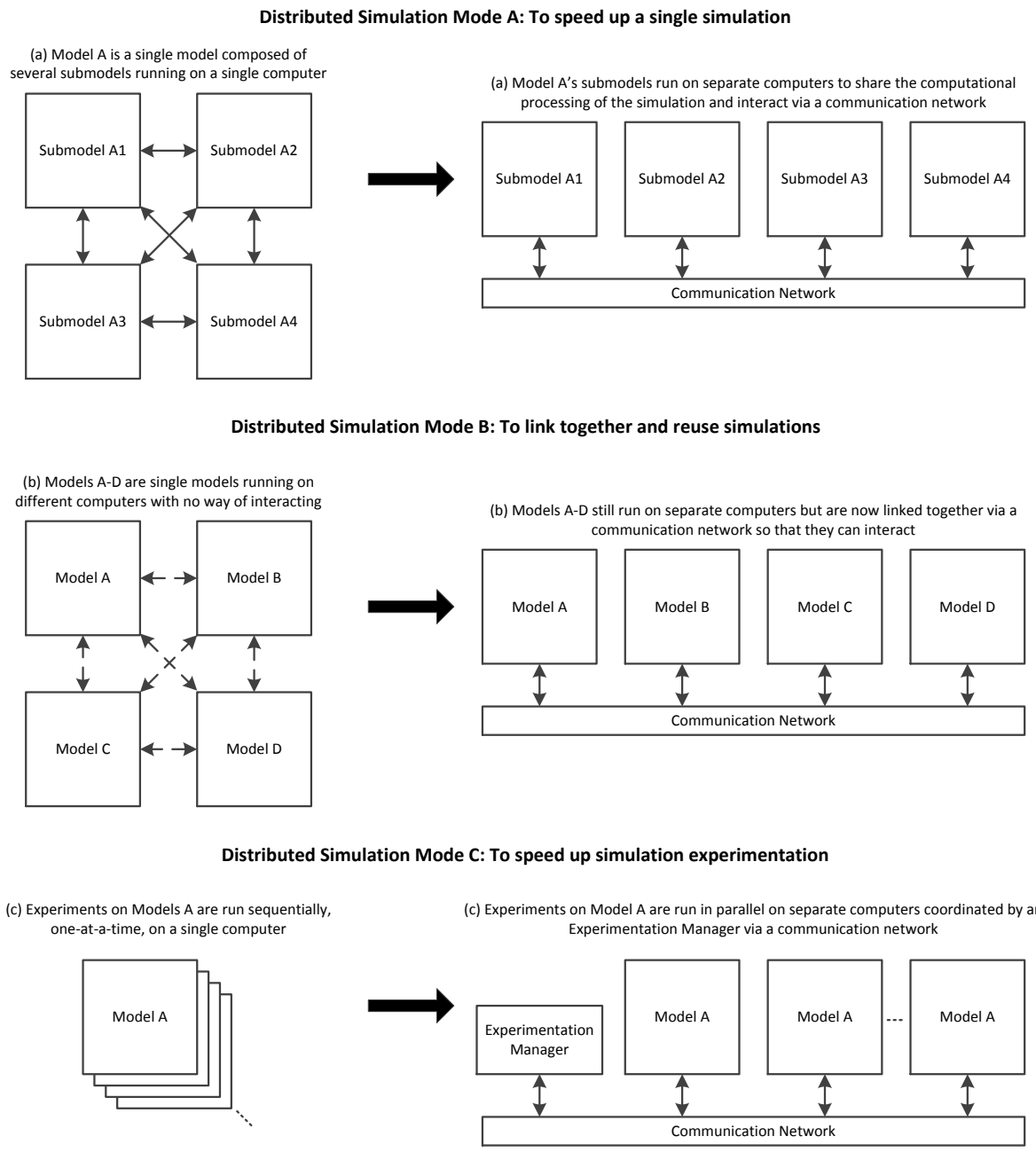


**Figure 1: Modes of Distributed Simulation**

There are various benefits that these Modes of DS offer (Mustafee, et al. 2012; Taylor, et al. 2012; Boer, de Bruin, & Verbraeck, 2009; Lendermann et al., 2007). For example:

*Execution time*. A large simulation can be slow to run. DS can be used to split the simulation across multiple computers to exploit parallel processing to speed up execution. DS may also allow simulation experimentation to be processed faster by using multiple computers.

*Model composability and reuse*. The development of a simulation can represent a significant investment in time and money. When building a new simulation it may be attractive to reuse a simulation as a sub-component. However, practical issues such as

variable name clashes, variable type incompatibility, global variables and different verification/validation assumptions might mean extensive recoding and testing. Further, if the simulations have been developed in different simulation packages or languages then it might not be possible to combine them at all without starting from scratch. It may be more convenient to just link the simulations together as a DS.

*Ownership and Management*. Following the above, if a simulation has been composed from reused simulations then it may be difficult for a simulation owner or developer to update their simulation without having to update the entire simulation. DS allows simulations to be independently managed as they are still separate.

*Privacy*. Creating a single simulation from other simulations could also mean that the entire details of a simulation would be revealed to the developer of the single simulation. If a simulation contains secrets (e.g. the confidential inner workings of a factory, hospital or military system) then these would be visible to anyone running the newly composed simulation. DS preserves this separation and allows simulations to be composed from "black boxed" simulations.

*Data integrity and privacy.* Similar to the above problems is the issue of data integrity and privacy. If a simulation requires access to a specific database then when a new simulation is created that data may have to be copied to allow the new simulation to access it. This data may be confidential. Another issue is how can the integrity of the data be preserved (how can the copy be kept up-to-date)? DS allows data to remain with the owning simulation and therefore avoids this issue.

*Hybrid simulation*. There are very few commercial simulation packages that support hybrid simulations consisting of discrete-event, agent-based and/or system dynamics elements. DS allows simulations of these different types to be linked together.

Mode A and Mode B of DS can be extremely complex to implement and, unfortunately, this presents a major barrier to its use (these Modes still represent some of the most challenging research topics in general distributed systems). Exceptions are in simulation areas where modelling teams possess advanced software engineering skills and are used to developing complex software solutions (e.g. in defence and some simulation software "houses"). Some standards and reference implementations have also been created that facilitates DS development (e.g. the High Level Architecture (IEEE, 2010) and associated interoperability issues (Taylor, Strassburger, Turner, & Mustafee, 2010)). These general standards have been adapted for process simulation used in Industry 4.0 (Anagnostou & Taylor, 2017).

DS Mode C, however, is conceptually simpler to implement (i.e. no complex synchronization) high speed simulation systems are beginning to emerge. Here the challenge is how to efficiently distribute and manage the execution of a series of single simulations over a range of computers. This is a common problem across many scientific disciplines and emerging solutions to DS Mode C are emerging with many borrowing techniques from scientific computing and e-Science. Early examples of these used grids of computers that already existed within an organisation (a desktop grid). More recent ones essentially use the same techniques but instead of fixed computing resources these use virtualised ones made available on a cloud. Examples of both of these include: the WINGRID desktop grid system that was used to speed up credit risk simulations in a well-known European bank (Mustafee & Taylor, 2009), SakerGrid, a desktop grid and computing cluster system in use today at Saker Solutions and Sellafield PLC (Kite, et al., 2011), a cluster-based high performance simulation system in use in the Ford Motor Company, a desktop grid that was used for simulations of biochemical pathways in cancer (Liu et al., 2014), and a cluster computing based grid used for a similar application (Choi, Seo, and Kim (2014)). Examples of cloud-based systems include an adaptation of the JADES platform to run agent-based simulations in parallel on cloud resources (Rak, Cuomo, & Villano, 2012) and the CloudSME Simulation Platform is used to run simulation experiments over multiple clouds (S.J.E. Taylor, Kiss, et

al., 2018). The Cloud Orchestration at the Level of Application (COLA) project[1] is developing a deadline-based auto-scaling approach for simulation experimentation on cloud with SakerCloud being the first commercial prototype ((Taylor et al., 2018). Anderson, Du, Narayan, & Gamal (2014) and Yao, et al. (2017) have developed Mode C DS that also run Mode A DS. Commercially, Saker Solutions have implemented the same in the DS of nuclear waste reprocessing.

In the following section, to illustrate the realisation of Mode C DS in support of High Speed Simulation Analytics, we present the CloudSME platform that arose from a major collaboration between e-Science developers and industrial simulation companies during the CloudSME project (www.cloudsme.eu).

## 3. Cloud-based High Speed Simulation Experimentation

Cloud computing is attractive as it offers on-demand computing resources that can be quickly "hired" and then discarded (Mell & Grance, 2011). The cost of computing resources is priced at a very attractive level. The use of these resources to power high speed simulation experimentation is therefore also very attractive. However, the complexity and variety of cloud systems and technologies can make realising these applications quite difficult and costly. Arguably, this can be prohibitive for Small and Medium-sized Enterprises (SMEs) and end user developers. Further, many cloud systems are developed for a single cloud. It is not an easy task to port from one cloud system to another.

The aim of the Cloud-based Simulation platform for Manufacturing and Engineering (CloudSME) project[2] was to create a generic approach to developing cloud-based simulation applications that enabled users to reduce implementation costs in realising commercial products and services. The project created the CloudSME Simulation Platform (CSSP) from a combination of an AppCenter, the workflow of the WS-PGRADE/gUSE (Kacsuk et al., 2012) science gateway framework and the multi-cloud-based capabilities of the CloudBroker Platform[3]. The CSSP has been used to implement a range of commercial simulation products across a many industrial domains (see the CloudSME Website[4] for examples). To show how the CSSP has been used to for high speed simulation experimentation we now describe the Platform and a representative case study.

### 3.1 The CloudSME Simulation Platform

The CSSP consists of three layers:

1. **Simulation Applications Layer** that allows software vendors deploying and presenting simulation products to end-users as SaaS (Software as a Service) in a wide range of scenarios and deployment models.
2. **Cloud Platform Layer** that provides access to multiple heterogeneous cloud resources and supports the creation of complex application workflows - a PaaS (Platform as a Service) to create and execute cloud-based simulations.
3. **Cloud Resources Layer** that represents the IaaS (Infrastructure as a Service) clouds connected to the platform.

These layers are presented in detail below (Figure 2).

---

[1] project-cola.eu
[2] www.cloudsme-apps.com
[3] www.cloudbroker.com
[4] http://www.cloudsme-apps.com/practical-examples/

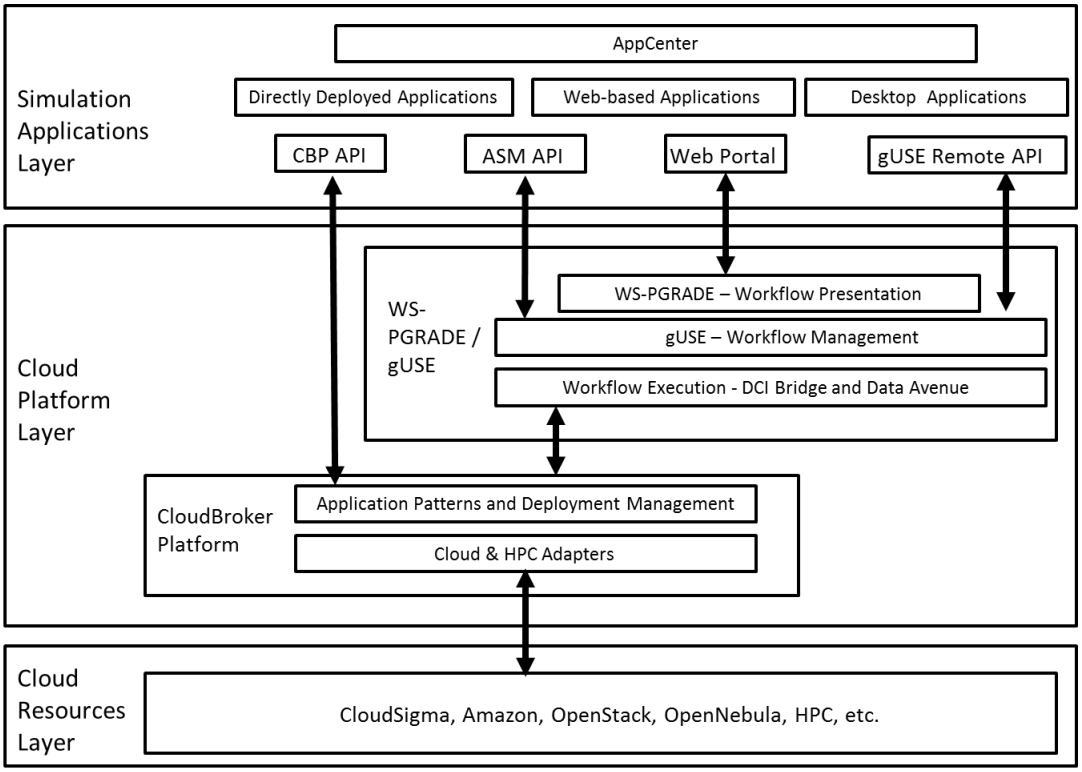


**Figure 2: The CloudSME Simulation Platform**

### 3.2 Simulation Applications Layer

This contains the CloudSME AppCenter is a web-based one-stop-shop that is the "shop window" to software products and services offered by software vendors and service providers to end users via a single consistent interface. It stores information about software products in an accessible way, provides usage scenarios for the software, and offers billing functionality that includes price setting, payment integration and tracking of users' spending. It three main deployment models: Directly Deployed Applications, Web-based Applications, and Desktop Applications. The CSSP offers a wide range of Application Programming Interfaces (APIs) to support developers. To enable the development of applications that are directly deployed in the AppCenter or the extension of desktop applications with cloud support, either the CloudBroker APIs (Java Client Library API or REST API) or the gUSE Remote API can be used. Using the CloudBroker APIs bypasses WS-PGRADE/gUSE and provides direct access from the application to the multi-cloud resources supported by the CloudBroker Platform. Using the Remote API of WS-PGRADE/gUSE enables developers to execute complex application workflows linking multiple application components together. As WS-PGRADE/gUSE is integrated with the CloudBroker Platform, multi-cloud execution capabilities are still fully utilised in this scenario. In case of web-based applications, either the ASM (Application Specific Module) API of WS-PGRADE/gUSE is used that enables the rapid development of a custom portal/gateway in the form of customised Liferay Portlets or a completely custom web interface is developed by embedding either CloudBroker API or gUSE Remote API calls. Alternatively the standard web-based interface to WS-PGRADE/gUSE can also be applied to launch workflows. All APIs are described in Akos, et al. (2013).

### 3.3 Cloud Platform Layer

The middle layer of CSSP is the Cloud Platform Layer that consists of the cloud-based services from the CloudBroker Platform and the science gateway framework WS-

PGRADE/gUSE. These components were developed prior to CloudSME and their first integration was implemented in the SCI-BUS (Scientific Gateway-based User Support) project (Kiss et al., 2014). During CloudSME this integration matured significantly and reached commercial production level.

### 3.3.1 The CloudBroker Platform

The CloudBroker Platform is a commercial PaaS that supports the management and execution of software on different cloud provider resources. The generic architecture of CloudBroker is shown in Figure 3.

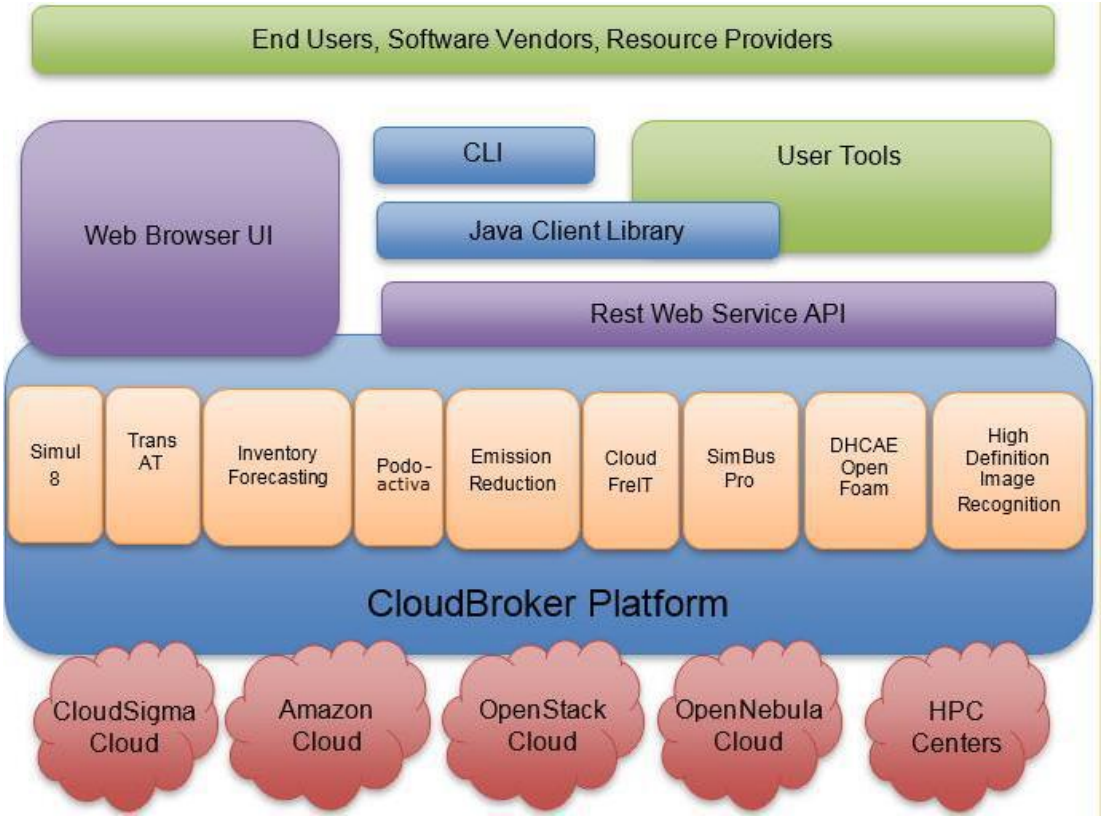


**Figure 3: CloudBroker Platform Architecture**

CloudBroker uses IaaS clouds from resource providers and incorporates adapters both to public and private cloud infrastructures. The platform provides access to a wide range of resources including open source (e.g. OpenStack and OpenNebula) and proprietary (e.g. Amazon and CloudSigma) clouds, and also various High Performance Computing (HPC) resources. CloudBroker supports non-interactive serial and parallel batch processing applications on both Linux and Windows operating systems. The platform itself consists of a set of modules that manage processes, applications, users, finance (accounting, billing and payment), and runtime issues (process monitoring, queuing, resources, storage and images). A scalability and fault handler layer supervises scalability requirements and failure issues. Cloud Provider Access Management oversees the connection to each Cloud technology and can control the number of virtual machines (VMs) started for a given application on a given cloud. Application “patterns” are deployed to CloudBroker in a form that allows the platform when instructed to run the application on a particular cloud and cloud instance type. Two typical patterns are direct installation (an application package and deployment script that allows the installation of the software on a cloud instance) or virtualisation (virtual machine image containing installed software that allows direct deployment to a cloud instance).

CloudBroker offers various interfaces for access. Its two main operation modes to manage and use software in the cloud are either as direct front-end, or as a back-end middleware service. For the former, the platform can be accessed directly through the Web Browser User Interface. As a back-end for advanced and automatic usage, various APIs are provided for programmatic accessibility. These include REST web service interface, Java client library and Linux shell command line interface (CLI). Via these different APIs, the CloudBroker Platform can be utilized by front-end software as middleware to allow access to applications in the cloud.

### 3.3.2 WS-PGRADE/gUSE

gUSE (Grid and Cloud User Support Environment) (Kiss et al., 2014) is an open source scientific gateway framework providing users with easy access to cloud and grid infrastructures. gUSE provides with WS-PGRADE, a Liferay based portal to create and execute scientific workflows in various Distributed Computing Infrastructures (DCIs) including clusters, grids and clouds. The generic architecture of WS-PGRADE/gUSE is presented in Figure 4.

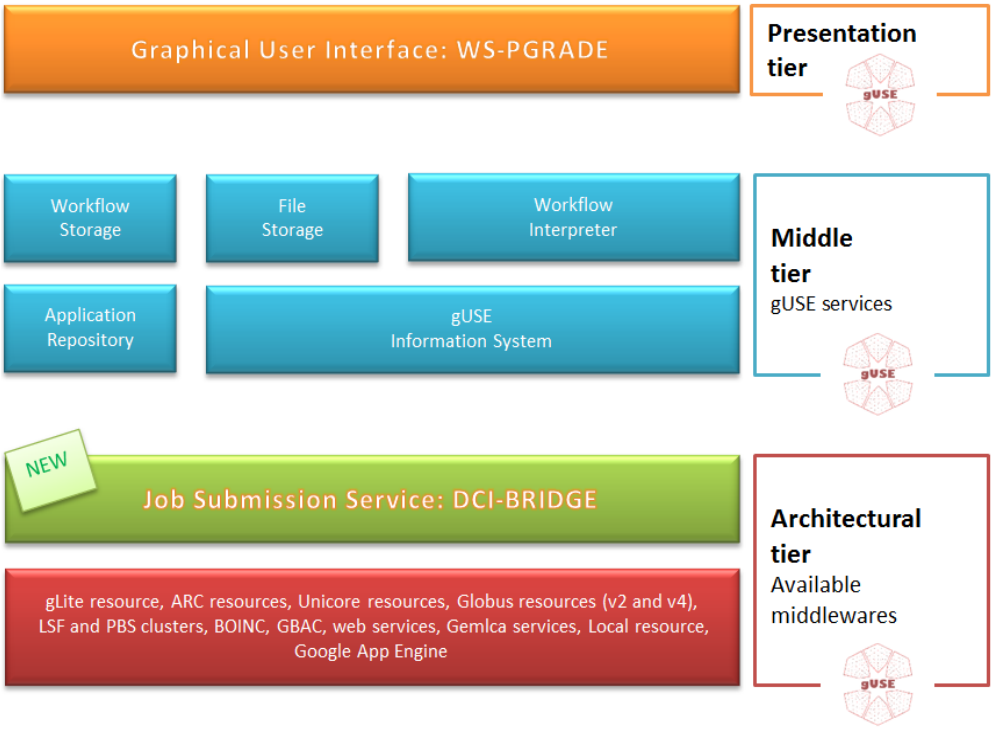


**Figure 4: Generic Architecture of WS-PGRADE/gUSE**

WS-PGRADE/gUSE consists of three layers: a top presentation layer, a middle management layer, and a bottom architectural execution layer.

The presentation layer (WS-PGRADE) includes a set of Liferay portlets to create, start and control workflows, monitor their execution on various DCIs, and present results to users. WS-PGRADE has a graph editor which can be used to build workflows and specify job configurations. A WS-PGRADE workflow is a directed acyclic graph that defines the execution logic of its components. An example for a WS-PGRADE workflow is presented in Fig. 5. The large boxes are jobs, while the smaller boxes are input and output ports representing input/output files for the jobs. The execution of a job can start when all of its inputs are available. Using this logic the WS-PGRADE workflow engine automates the execution of the workflow. For example, in case of the workflow of Figure 5 only Gen3 can

start executing when the workflow is submitted. MulCross and AddPair are waiting for the result of Gen3 and can start once the output file of Gen3 is available.

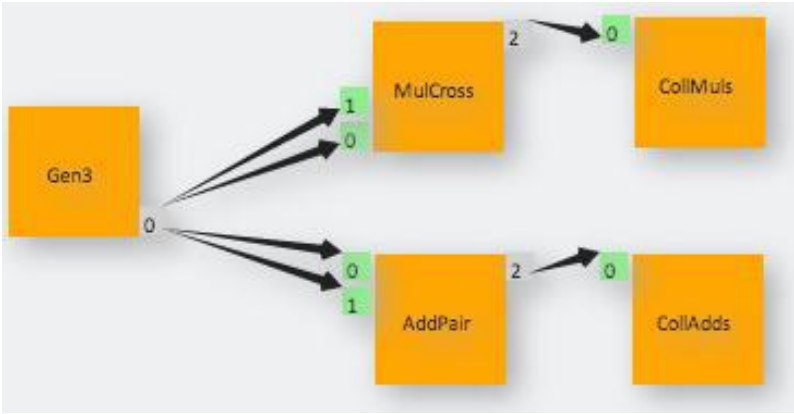


**Figure 5: Example WS-PGRADE Workflow**

The WS-PGRADE workflow concept supports multiple levels of parallelism. Each job of the workflow can in itself be a natively parallel application (e.g. using MPI). The workflow can also have parallel branches (e.g. MulCross/ColMuls and AddPair/ColAdds are in parallel branches) that can be executed in parallel of different resources. Finally, WS-PGRADE supports parameter sweep applications. Parameter sweep applications are simulations where the same simulation needs to be executed for multiple input data sets. This feature enables the same workflow to be submitted with multiple input data sets simultaneously.

A full description of WS-PGRADE/gUSE gateway framework is available in Kacsuk, et al. (Kacsuk et al., 2012), and Kacsuk (Kacsuk, 2014) gives a complete overview of WS-PGRADE/gUSE and its applications.

### 3.4 Cloud Resources Layer

The bottom layer of CSSP is the Cloud Resources Layer that consists of a range of clouds and HPC resources accessible via the CloudBroker Platform. These currently include CloudSigma and Amazon public clouds, various private clouds based on either OpenStack or OpenNebula, and the HPC resources of, for example, the Cineca Galileo Cluster or the ETH Euler Cluster.

## 4. Case Study: High Speed Simulation Experimentation

The following case study demonstrates how the CSSP can support high speed simulation experimentation. It uses the widely used open source simulation system the Recursive Porous Agent Simulation Toolkit (REPAST). This is a cross-platform, agent-based modelling and simulation toolkit and is a Java-based simulation system that is used for developing a range of simulation applications in different fields (North et al., 2013). To enable the parallel execution needed for high speed simulation experimentation it uses parameter sweeps running on multiple cloud resources via both components of the Cloud Platform Layer. CloudBroker manages deployment on multiple clouds and the parameter sweep functionality of the WS-PGRADE/gUSE workflow engine manages the execution of the simulation experiments and the parameter sweep. The deployment is first presented and then demonstrative results.

## 5 REPAST Deployment on the CloudSME Simulation Platform

The deployment of REPAST consists of two parts: deployment on CloudBroker and creation of the parameter sweep workflow on WS-PGRADE/gUSE.

Deployment on CloudBroker is done by creating an application package consisting of a deployment shell script, an execution shell script and the zipped REPAST environment. For each cloud deployment, CloudBroker is configured to create a virtual machine with a Linux Ubuntu OS image. Using its web interface, CloudBroker creates this virtual machine, transfers the application package to the virtual machine and then runs the deployment shell script. This installs REPAST, Java Runtime Environment and the execution shell script. When a job is started (i.e. a simulation run), the simulation model (a TAR archive consisting of the model source code and the simulation scenario) and the parameter sweep data (an XML file specifying the input parameters) are transferred to the virtual machine. The execution script then validates these inputs, extracts the model files and runs the simulation. Results are then added to a TAR archive for upload back to the Platform.

The WS-PGRADE/gUSE web interface is used to create the parameter sweep workflow. An abstract workflow graph is first created using the graph editor. From the graph, the concrete workflow is then created and configured to run the selected software on the selected cloud resources. The same abstract workflow can be used to create many concrete workflows by reconfiguring them. Once the graph is completed, it can be saved and used to create a concrete workflow where the jobs can be configured (e.g. the simulation software, the cloud and the region of the resources, and the instance type).

To demonstrate the performance of high speed simulation we used a well-known benchmark developed at Brunel. This is an agent-based simulation of infection disease spread (Macal, 2016). The simulation consists of three types of agents that move in an environment and interact with each other. The agents represent the susceptible, infected and recovered population. The model starts an infection outbreak with an initial population of infected and susceptible agents. Infected agents move close to susceptible agents and infect them while susceptible agents move where the least infected agents are located. Infected and susceptible agents interact with each other in every simulation time unit which is a day in our simulation. Infected agents recover after a period of time and become recovered with a level of immunity. When an infected agent gets in touch with a susceptible agent, the susceptible agent becomes infected. When an infected agent gets in touch with a recovered agent, the recovered agent decreases its immunity. When the immunity level is 0, the recovered agent becomes susceptible and can be infected again. The outbreak occurs annually. When this happens, the population changes to reflect the initial conditions taking into account the population dynamics of the previous year.

A series of experiments on two cloud infrastructures were performed: the Amazon EC2 commercial cloud and an academic cloud offered by the University of Westminster (UoW), UK. Cloud instances of various sizes were used as specified in Table 1. Each experiment was set up in WS-PGRADE/gUSE by quickly reconfiguring the workflow by selecting a different cloud/instance type.

Table 1. Cloud Resources Characteristics

| Cloud Instance | Number of vCPUs | Processor type | Memory |
|---|---|---|---|
| Amazon baseline micro (A1) | 1 | High Frequency Intel Xeon Processors with Turbo up to 3.3GHz | 0.5 GiB |
| Amazon baseline small (A2) | 1 | High Frequency Intel Xeon Processors with Turbo up to 3.3GHz | 1 GiB |
| Amazon baseline medium (A3) | 2 | High Frequency Intel Xeon Processors with Turbo up to 3.3GHz | 4 GiB |
| Amazon balanced medium (A4) | 1 | High Frequency Intel Xeon E5-2670 v2 at 2.6GHz | 3.75 GiB |
| Amazon balanced large (A5) | 2 | High Frequency Intel Xeon E5-2670 v2 at 2.6GHz | 7.5 GiB |
| UoW small (U1) | 1 | AMD Opteron 4122 Processor at 2.2GHz | 20 MB |
| UoW medium (U2) | 2 | AMD Opteron 4122 Processor at 2.2GHz | 40 MB |
| UoW large (U3) | 4 | AMD Opteron 4122 Processor at 2.2GHz | 80 MB |
| UoW XL (U4) | 8 | AMD Opteron 4122 Processor at 2.2GHz | 160 MB |

Our demonstration consisted of an experiment consisting of ten runs (i.e. ten simulations with a different parameter). We conducted ten experiments. These took approximately 200 minutes to run on a desktop PC (i5-2500 processor at 3.30GHz speed and 4.00GB RAM). We ran these experiments on one, two, five and 10 instances of each cloud type. The experiments were distributed equally when run on more than one instances. Figure 6 shows the comparative runtime by instance and Table 2 shows the speedup when compared to a single PC run. The run-time is the average of five runs. From the results, we observe that Amazon EC2 instances have relatively stable performance and the academic cloud presents a larger variation. For example, five instances of U2 perform worse than two. Also, we have a considerable increase in execution time when running on 10 instances. Types U3 and U4 show similar behaviour. This is suspected to be rooted in variations in resource availability that cause job requests to be queued until resources are available. Similar behaviour with less variation is shown by A3 where the execution time for five instances is increased. In terms of speedup, when running on a single instance for all cloud types in this experiment, apart from A5, the performance is slower than a desktop machine. This is expected since there is an overhead for starting up the virtual machines. Most of the larger instances, at least the commercial ones, present modest speedup. It is expected that for larger simulations there will be better considerable speedup as the longer processing time will compensate the overheads of setting up virtual instances on a cloud. Overall this shows how a user might investigate different cloud and instance types to choose which is the best for his or her needs.

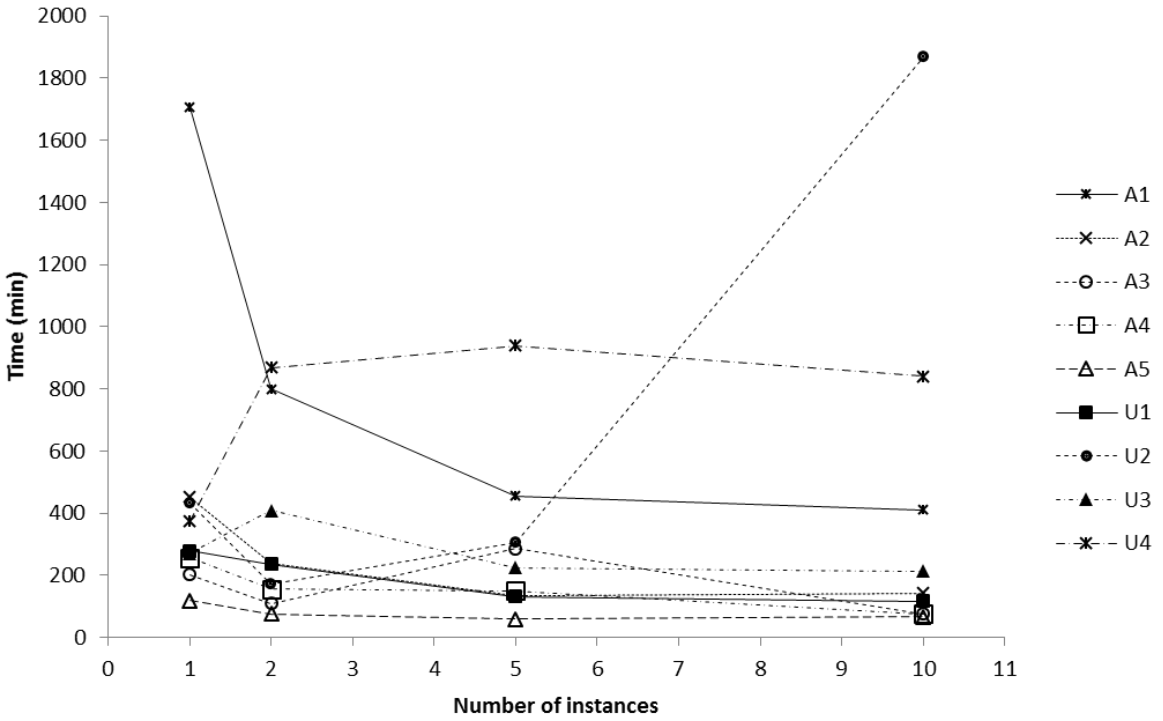


Figure 6: Cloud-based REPAST Infection Model Performance

Table 2. Cloud-based REPAST Infection Model Speedup

| Clouds | | A1 | A2 | A3 | A4 | A5 | U1 | U2 | U3 | U4 |
|---|---|---|---|---|---|---|---|---|---|---|
| Instances | 1 | 0.12 | 0.44 | 0.99 | 0.78 | 1.68 | 0.72 | 0.46 | 0.74 | 0.53 |
| | 2 | 0.25 | 0.84 | 1.82 | 1.28 | 2.67 | 0.85 | 1.18 | 0.49 | 0.23 |
| | 5 | 0.44 | 1.49 | 0.70 | 1.32 | 3.39 | 1.53 | 0.66 | 0.89 | 0.21 |
| | 10 | 0.49 | 1.40 | 2.63 | 2.67 | 2.99 | 1.71 | 0.11 | 0.93 | 0.24 |

## 6. A Vision of the Future: Towards Big Simulation Analytics

The previous sections have shown how DS and cloud computing can address high speed simulation analytics. Adding other modes of DS to this could enable high speed simulation analytics of large-scale simulations of large systems. Taking inspiration from Big Data, this move towards larger and larger systems simulations involving the analysis of diverse data suggests that we might call this emerging aspect of simulation as *Big Simulation Analytics.* We now discuss how we might realise Big Simulation Analytics.

In large scale scientific endeavours, many scientists use grid computing or e-Infrastructures, integrated collections of computers, data, applications and sensors across different organizations (Foster et al., 2001; Bird, Jones, & Kee, 2009). There are various sophisticated software systems that exist to use e-Infrastructure facilities, typically by giving “single sign-on” secure access to multiple computers across multiple administrative domains and the ability to manage the execution of jobs on those computers (e.g. WS-PGRADE/gUSE (Kacsuk et al., 2012; Kiss et al., 2014) and the FutureGateway that has evolved from the DECIDE framework (Ardizzone et al., 2012)). E-Infrastructure applications can be created from these by first deploying the application service on the e-Infrastructure and registering it in some form for service catalogue (see below) and then accessing the service via a science gateway (a web-based system that allow scientists to use e-Infrastructures with a simple front end that has been developed for their needs) or some kind of programming interface (usually some kind of REST interface) integrated into software that

is familiar to the user (for a wide range of examples of these see www.sci-gaia.eu/community and catalog.sciencegateways.org/#/home for examples of science gateways). Software applications or services are being increasingly developed in a standard way so that they can be stored, browsed and reused from a standardized service catalogue (e.g. the EGI service catalogue (https://www.egi.eu/services/) and the INDIGO service catalogue (www.indigo-datacloud.eu)). Applications can be linked together by workflows, sequences of tasks that are translated into jobs executed on specific computing systems supported by the above software infrastructures (Deelman, et al., 2009; Liew et al., 2016). Examples of workflow systems include Pegasus (Deelman et al., 2016), Kepler (Ludäscher et al., 2006), Taverna (Wolstencroft et al., 2013), Swift (Zhao et al., 2007) and WS-PGRADE/gUSE (Kacsuk et al., 2012).

In a possible future where DS is commonly used in OR, a user might access an e-Infrastructure via a web-based science gateway, configure a workflow to execute a series of tasks and instruct those tasks to be run. As shown in Figure 7, such a workflow might have five steps: Management, Acquisition, Composition, Experimentation and Analysis.

### 6.1 Management

In this task a user first selects a pre-defined experimentation service (e.g. direct experimentation, ranking & selection algorithm, optimization, etc.) The user then configures the experimentation and then selects what *infrastructure* to run on. The choice might be an internal computing resource (e.g. a cluster), different external clouds, a dedicated high performance computing facility, etc. Cost/time information might be given for each infrastructure to help the user to decide which to select. A user might also set a deadline for experimentation and then get an estimate for how much processing resources would cost (and possibly their carbon footprint). Once the infrastructure has been selected, the user then pays if necessary (or uses some pre-loaded credit), and then instructs the management task to run the experiments. The system would then manage the experiment over the selected infrastructure, reporting to user the progress of the experimentation and when it is complete.

### 6.2 Acquisition

Experiments configured in Management use this task to acquire relevant data sources (databases, spreadsheets, etc.), update statistical distributions, obtain the latest versions of the models and simulation software, etc. needed for experimentation. In the case of Symbiotic Simulation, Cyber-physical systems or a Digital Twin, this might involve direct data collection from the sensors in a physical system. We may assume that the selection of services in this task has been predefined and the task runs these to perform the updates.

### 6.3 Composition

This task simply takes the above acquired artefacts and composes the jobs to be submitted to the infrastructure. With a single simulation this task would just ready the model and its supporting components for uploading to the infrastructure. A DS would require several models to be composed (i.e. a set of federates being composed into a federation) and a supporting workflow service could be selected to automate this(Chaudhry, Nouman, Anagnostou, & Taylor, 2016).

### 6.4 Experimentation

Jobs representing each run of a simulation (or possibly runs if these are quick but numerous) are submitted to a queue for the infrastructure to process. This task also manages the

execution of the jobs (e.g. relaunching any failed jobs) and collates the results from each job as their results are returned from the infrastructure.

### 6.5 Analysis

The final step is the *Analysis* task. Users could select from a set of services that analyse the output from experimentation. This could include, for example a service that produces summary statistics or some deeper time series-based analysis. The Analysis service could itself be workflow based and run over distributed computing resources to reduce the time taken to analyse the output. Indeed, it is possible that a user could request several analyses to be performed at the same time and the results from this be brought together in some kind of hierarchical workflow. In these cases the Management task could be extended to give further cost estimates for analysis. Similar extensions could be made to reflect the on-going cost of optimization.

### 6.6 Conceptualisation and Example

Based on this workflow, Figure 8 shows a possible conceptualization of an e-Science approach for DS that shows the workflow realized on an e-Infrastructure using a science gateway. This is influenced by the workflow system WS-PGRADE/gUSE and is based on recent experiences with the CloudSME project where several commercial cloud-based simulation systems using e-Infrastructure approaches were created.

Consider the following example. An enterprise is capable of manufacturing a range of widgets for a number of consumers. The manager of the enterprise in this supply chain wants to understand how the behaviour of her factory responds to changes in demand and supply over time. She has a discrete-event model of her factory and agent-based models of her suppliers and consumers (perhaps a more reasonable large supply chain model as this does not assume that other discrete-event simulations in the supply chain exist but does assume that the enterprise has detailed information about supplier/consumer behaviour over time). We assume that a management interface similar to a science gateway has been set up and a workflow has been defined in WS-PGRADE/gUSE. The manager might want to (for example) investigate the most reliable set of suppliers based on a 20% increase in consumption across her product range and to identify the most critical areas in her factory in terms of machine utilization and operator utilization (we assume that a mix of machines and operators are used in her factory to produce the widgets).

In the Management task, she sets up the experiments on her management interface (the equivalent of a science gateway) and chooses an analytics service that can correlate and cluster the simulation results. She then investigates the best available infrastructure to run the experiments within a reasonable amount of time (e.g. compares the cost of Amazon Cloud, Microsoft Azure and a High Performance Computing centre available in her region against running over a local desktop grid), makes her selection and begins the experimentation. The workflow then begins automatic execution by executing the Acquisition task. This executes in parallel to load the most recent data and model into the infrastructure. The *Composition* task then composes the DS by bringing together the three models with HLA standard software for time management. The *Experimentation* task would then create “jobs” based on each experiment and dispatch these through the infrastructure to (say) virtual computers running on the Amazon cloud. As results begin to come in, the infrastructure passes these onto the *Analysis* task. This task takes each set of results and, in turn, sends these jobs out for processing on the infrastructure using a clustering and classification service that runs for each job and then collates these together for display on the management interface. The manager then makes her decisions within hours rather than months. In the case of Symbiotic Simulation or Digital Twins, once set up, this process

might run constantly as the system monitors and attempts to improve the performance of the system via simulation.

## 7. Conclusions.

This article has presented the possible future of High Speed Simulation Analytics from an Industry 4.0 perspective. It has argued that the key to this is DS and high speed experimentation. A novel commercial system has been presented that demonstrates how cloud computing can be used to speed up simulation experimentation. We have then discussed how simulation analytics can borrow from e-Science and e-Infrastructures to create a vision or architecture for large-scale simulation analytics or Big Simulation Analytics. It is hope that this article has shown how the future for simulation analytics could develop and the potential functionality that emerging approaches need to urgently embrace to keep simulation relevant and at the heart of Industry 4.0. Taylor (2018) develops these themes in more detail from an Operational Research perspective, and Taylor, et al. (2018a; 2018b) give more detail on the CloudSME Simulation Platform and its simulation applications.

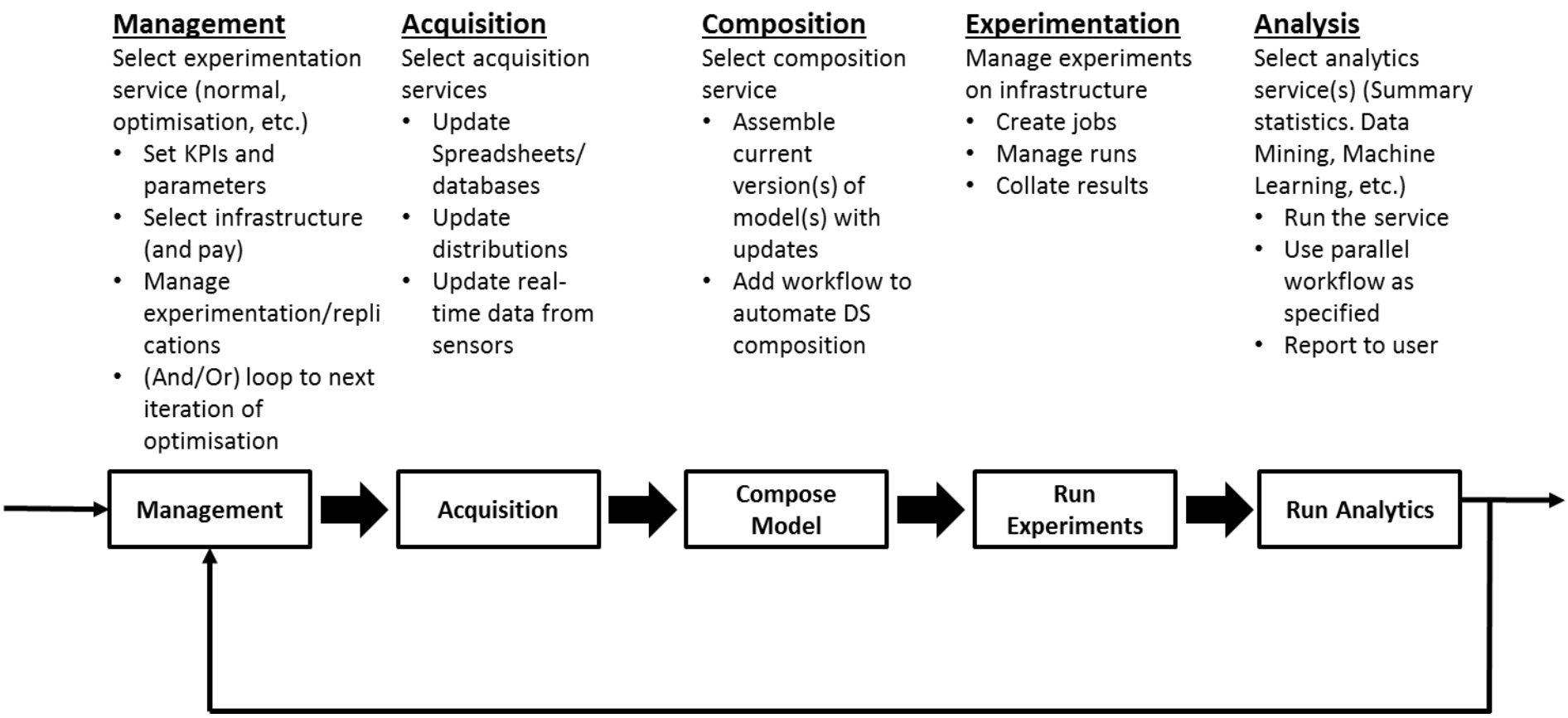


**Figure 7: A Workflow for High Speed Simulation Analytics**

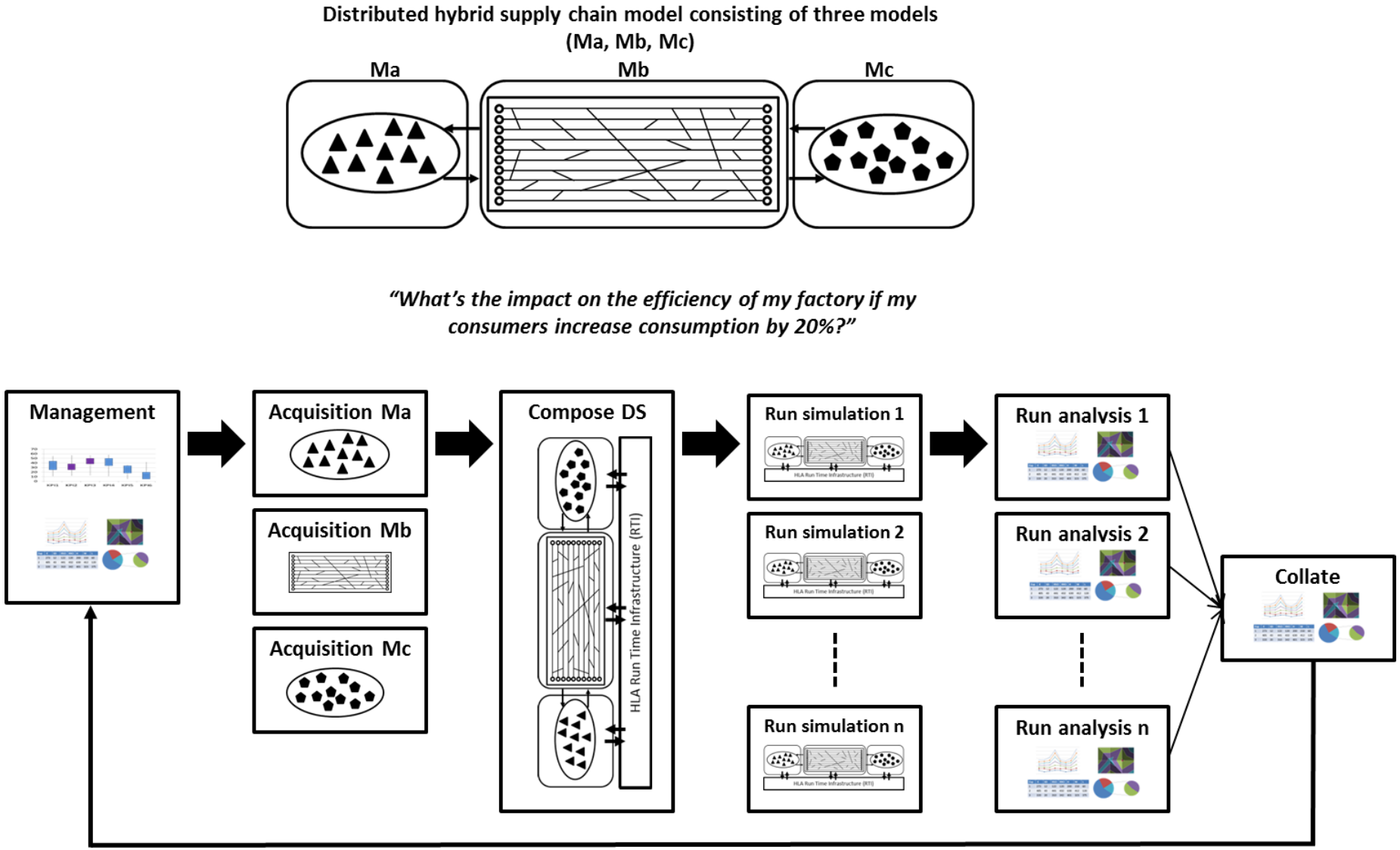


**Figure 8: An e-Science Vision for High Speed Simulation Analytics**

**References**

Akos, B., Zoltan, F., Peter, K., & Kacsuk, P. (2013). Building Science Gateways by Utilizing the Generic WS-PGRADE/gUSE Workflow System. *Computer Science*, *14*(3), 307–325. https://doi.org/10.7494/csci.2013.14.2.307

Anagnostou, A., & Taylor, S. J. E. (2017). A distributed simulation methodological framework for OR/MS applications. *Simulation Modelling Practice and Theory*, *70*, 101–119. https://doi.org/10.1016/j.simpat.2016.10.007

Anderson, K., Du, J., Narayan, A., & Gamal, A. E. (2014). GridSpice: A distributed simulation platform for the smart grid. *IEEE Transactions on Industrial Informatics*, *10*(4), 2354–2363. https://doi.org/10.1109/TII.2014.2332115

Ardizzone, V., Barbera, R., Calanducci, A., Fargetta, M., Ingrà, E., Porro, I., … Schenone, A. (2012). The DECIDE Science Gateway. *Journal of Grid Computing*, *10*(4), 689–707. https://doi.org/10.1007/s10723-012-9242-3

Boer, C. A., de Bruin, A., & Verbraeck, A. (2009). A survey on distributed simulation in industry. *Journal of Simulation*, *3*(1), 3–16. https://doi.org/10.1057/jos.2008.9

Chaudhry, N. R., Nouman, A., Anagnostou, A., & Taylor, S. J. E. (2016). WS-PGRADE workflows for cloud-based distributed simulation. In *Proceedings of the Operational Research Society Simulation Workshop 2016* (pp. 192–201).

Choi, C., Seo, K.-M., & Kim, T. G. (2014). DEXSim: an experimental environment for distributed execution of replicated simulators using a concept of single simulation multiple scenarios. *SIMULATION*, *90*(4), 355–376. https://doi.org/10.1177/0037549713520251

Davenport, T. H., & Harris, J. G. (2007). *Competing on Analytics: the New Science of Winning*. Boston, MA: Harvard Business School.

Deelman, E., Gannon, D., Shields, M., & Taylor, I. (2009). Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, *25*(5), 528–540. https://doi.org/10.1016/j.future.2008.06.012

Deelman, E., Vahi, K., Rynge, M., Juve, G., Mayani, R., & Da Silva, R. F. (2016). Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Computing*, *20*(1), 70–76. https://doi.org/10.1109/MIC.2016.15

Foster, I., Kesselman, C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of High Performance Computing Applications*, *15*(3), 200–222. https://doi.org/10.1177/109434200101500302

Fujimoto, R. M. (1990). Parallel Discrete Event Simulation. *Commun. ACM*, *33*(10), 30–53. https://doi.org/10.1145/84537.84545

Fujimoto, R. M. (2000). *Parallel and Distributed Simulation Systems*. New York: John Wiley & Sons.

Fujimoto, R. M. (2016). Research Challenges in Parallel and Distributed Simulation. *ACM Transactions on Modeling and Computer Simulation*, *26*(4), 1–29. https://doi.org/10.1145/2866577

Heidelberger, P. (1986). Statistical Analysis of Parallel Simulation. In *Proceedings of the 1986 Winter Simulation Conference (WSC)* (pp. 2278–2288).

IEEE. (2010). *IEEE 1516-2010 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. IEEE Computer Society Press. https://doi.org/10.1109/IEEESTD.2010.5953411

Kacsuk, P. (Ed.). (2014). *Science Gateways for Distributed Computing Infrastructures*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-11268-8

Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., & Marton, I. (2012). WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities. *Journal of Grid Computing*, *10*(4), 601–630. https://doi.org/10.1007/s10723-012-9240-5

Kiss, T., Kacsuk, P., Takacs, E., Szabo, A., Tihanyi, P., & Taylor, S. J. E. (2014). Commercial use of WS-PGRADE/gUSE. In P. Kacsuk (Ed.), *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by*

*Scientific User Communities* (pp. 271–286). Spinger Cham. https://doi.org/10.1007/978-3-319-11268-8-19

Kite, S., Wood, C., Taylor, S. J. E., & Mustafee, N. (2011). SAKERGRID: Simulation experimentation using grid enabled simulation software. In *Proceedings of the 2011 Winter Simulation Conference (WSC)* (pp. 2278–2288). https://doi.org/10.1109/WSC.2011.6147939

Lendermann, P., Heinicke, M. U., McGinnis, L. F., McLean, C., Strassburger, S., & Taylor, S. J. E. (2007). Panel: distributed simulation in industry - A real-world necessity or ivory tower fancy? In *Proceedings of the 2007 Winter Simulation Conference (WSC)* (pp. 1053–1062). https://doi.org/10.1109/WSC.2007.4419704

Liew, C. S., Atkinson, M. P., Galea, M., Ang, T. F., Martin, P., & Hemert, J. I. Van. (2016). Scientific Workflows: Moving Across Paradigms. *ACM Computing Surveys*, *49*(4), 1–39. https://doi.org/10.1145/3012429

Liu, X., Taylor, S. J. E., Mustafee, N., Wang, J., Gao, Q., & Gilbert, D. (2014). Speeding up systems biology simulations of biochemical pathways using Condor. *Concurrency Computation Practice and Experience*, *26*(17), 2727–2742. https://doi.org/10.1002/cpe.3161

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., … Zhao, Y. (2006). Scientific workflow management and the Kepler system. *Concurrency Computation Practice and Experience*, *18*(10), 1039–1065. https://doi.org/10.1002/cpe.994

Lustig, I., Dietrich, B., Johnson, C., & Dziekan, C. (2010). The Analytics Journey. *Analytics Magazine*, (November/December), 11–13. Retrieved from http://analytics-magazine.org/the-analytics-journey/

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, *10*(2), 144–156. https://doi.org/10.1057/jos.2016.7

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing. National Institute of Standards and Technology*. Gaithersburg, MD. Retrieved from http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Mustafee, N., & Taylor, S. J. E. (2009). Speeding up simulation applications using WinGrid. *Concurrency Computation Practice and Experience*, *21*(11), 1504–1523. https://doi.org/10.1002/cpe.1401

Mustafee, N., Taylor, S., Katsaliaki, K., Dwivedi, Y., & Williams, M. (2012). Motivations and barriers in using distributed supply chain simulation. *International Transactions in Operational Research*, *19*(5), 733–751. https://doi.org/10.1111/j.1475-3995.2011.00838.x

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., & Sydelko, P. (2013). Complex adaptive systems modeling with Repast Simphony. *Complex Adaptive Systems Modeling*, *1*(1), 3. https://doi.org/10.1186/2194-3206-1-3

Rak, M., Cuomo, A., & Villano, U. (2012). mJADES: Concurrent simulation in the cloud. In *Proceedings of the 2012 International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)* (pp. 853–860). https://doi.org/10.1109/CISIS.2012.134

Taylor, S. J. E. (2018). Distributed simulation: state-of-the-art and potential for operational research. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2018.04.032

Taylor, S. J. E., Anagnostou, A., Kiss, T., Kite, S., Pattison, G., Kovacs, J., & Kacsuk, P. (2018). An Architecture For An Autoscaling Cloud-Based System For Simulation Experimentation. In *2018 Winter Simulation Conference*. IEEE Press.

Taylor, S. J. E., Anagnostou, A., Kiss, T., Terstyanszky, G., Kacsuk, P., Fantini, N., … Costes, J. (2018). Enabling Cloud-based Computational Fluid Dynamics with a Platform as a Service Solution. *IEEE Transactions on Industrial Informatics*. https://doi.org/10.1109/TII.2018.2849558

Taylor, S. J. E., Kiss, T., Anagnostou, A., Terstyanszky, G., Kacsuk, P., Costes, J., & Fantini, N. (2018). The CloudSME simulation platform and its applications: A generic multi-

cloud platform for developing and executing commercial cloud-based simulations. *Future Generation Computer Systems*, *88*, 524–539. https://doi.org/10.1016/j.future.2018.06.006

Taylor, S. J. E., Strassburger, S., Turner, S. J., & Mustafee, N. (2010). *SISO-STD-006-2010 Standard for COTS Simulation Package Interoperability Reference Models*. Orlando.

Taylor, S. J. E., Turner, S. J., Strassburger, S., & Mustafee, N. (2012). Bridging the gap: A standards-based approach to OR/MS distributed simulation. *ACM Transactions on Modeling and Computer Simulation*, *22*(4), 1–23. https://doi.org/10.1145/2379810.2379811

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., … Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, *41*, W557-61. https://doi.org/10.1093/nar/gkt328

Yao, Y., Meng, D., Zhu, F., Yan, L., Qu, Q., Lin, Z., & Ma, H. (2017). Three-level-parallelization support framework for large-scale analytic simulation. *Journal of Simulation*, *11*(3), 194–207. https://doi.org/10.1057/s41273-017-0057-x

Zhao, Y., Hategan, M., Clifford, B., Foster, I., Von Laszewski, G., Nefedova, V., … Wilde, M. (2007). Swift: Fast, reliable, loosely coupled parallel computation. In *Proceedings of the 2007 IEEE Congress on Services* (pp. 199–206). https://doi.org/10.1109/SERVICES.2007.63

**Glossary**

| | |
|---|---|
| Agent-based Simulation | ABS |
| Application Programming Interfaces | APIs |
| Discrete-event Simulation | DES |
| Distributed Simulation | DS |
| High Level Architecture | HLA |
| High Performance Computing | HPC |
| Information and Communication Technologies | ICT |
| Infrastructure as a Service | IaaS |
| Innovation for Manufacturing SMEs | i4MS |
| Institute of Electrical and Electronics Engineers | IEEE |
| Internet of Things | IoT |
| Modelling & Simulation | M&S |
| Modelling & Simulation as a Service | MSaaS |
| Operational Research | OR |
| OR/MS | Operational Research/Management Science |
| Parallel and Distributed Simulation | PADS |
| Parallel Discrete Event Simulation | PDES |
| Platform as a Service | PaaS |
| Run Time Infrastructure | RTI |
| Simulation Interoperability Standards Organization | SISO |
| Small-to-Medium Enterprise | SME |
| Software as a Service | SaaS |