Overcoming engineering challenges of providing an effective user interface to a large scale distributed synthetic environment on the US Teragrid: A systems engineering success story

Roy S. Kalawsky

Wolfson School of Mechanical, Electrical and Manufacturing Engineering,

Loughborough University, UK

r.s.kalawsky@lboro.ac.uk

tel +44 1509 635678

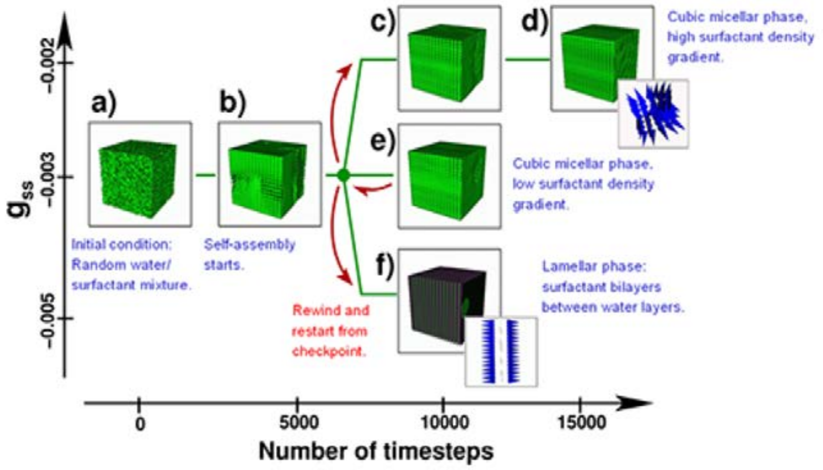
Abstract. Over recent years' large scale distributed synthetic environment enterprises have been evolving in a diverse range of scientific and engineering fields. These computer modelling and simulation systems are increasing in scale and dimension in order to allow scientists and engineers to explore the attributes and emergent properties of a given system design. Within the field of computational science, the grid has been developed to facilitate very large scale collaborative simulation enterprises. The grid is similar to Distributed Interactive Simulation/High Level Architecture (DIS/HLA) in that it supports interconnectivity but differs in the sense that it supports intercommunication of large super computing resources. An important factor in the rapid adoption of the grid has been its role in enabling access to significant supercomputing resources not usually available at a single institution. However, the major challenge for the grid has been the lack of an effective and ubiquitous interface to the huge computational resource (which can comprise over 6000 CPUs distributed across the globe) at any time and from any location. This paper describes a unique user interface built on systems engineering principles and practices to solve the problem of delivering real-time interaction (from lightweight computing devices such as personal digital assistants (commonly known as tablet devices) to high end computing platforms) with simulations delivering high resolution 3D images. The application of our work has far reaching benefits for many sectors including: aerospace, medical informatics, engineering design, distributed simulation and modelling.

Introduction

Distributed synthetic environments are increasingly being used in scientific/engineering enterprises to explore the attributes and emergent properties of complex system designs. An important feature of a synthetic environment is its ability to allow scientists/engineers to interactively explore the parameter space of a computer based model or simulation. Consequently, distributed simulation tends to support one or more of the following activities:

- Data management:- linking simulation models to remote databases (Korn, Burns et al. 1999) or real time systems (Davis 1998),
- Distribution of model/simulation: – execution of a one large simulation model (Gabbar, Shinohara et al. 2003) or execution of many separate simulation models across different computers (Zulch, Jonsson et al. 2002),
- Simulation distribution:- Distribution of the same simulation or multiple simulation scenarios (Yucesan, Luo et al. 2001),
- Collaborative environments: - simulations that can be interactively steered by many users (Liere, Mulder et al. 1997; Mulder, van Wijk et al. 1999), can also include models or simulation software that can be shared.
- Providing access to super computing resources that may not exist at a particular site.

Currently, military based distributed simulation systems typically conform to the High Level Architecture (HLA) standard because it is a relatively mature modelling and simulation standard in this domain. The HLA focuses on the implementation level and cannot readily be used to solve problems at the conceptual level, such as validation and verification of simulation models. At the systems level there are many situations where simulations from different domains need to be coupled to support cross-disciplinary investigations. Moreover, as the scale and complexity of simulation increases the demand for access to ever more powerful computing resources also increases. To meet this need an important approach for scientific research known as e-Science (Hey and Trefethen 2003) was developed in the UK which relies on the utilisation of complex distributed heterogeneous resources including experimental apparatus, large scale simulation and visualization resources. In e-Science, such resources are interconnected with a distributed computing environment known as the grid (Foster and Kessleman 1999). The grid is ideally suited to support computer simulations in a diverse range of scientific and engineering fields and has the distinct advantage of harnessing the power of distributed compute and visualization resources. For very large simulations the required computational resources frequently exceed those available at a single institution and in order to process jobs in a reasonable timeframe additional resources are employed that are geographically remote from the scientist undertaking the work. In this context large grid based simulations can typically employ hundreds of geographically distributed processing nodes over which the computation is executed (in order to increase the scale of the simulation or reduce the overall execution time). The growing need to process very complex models and large datasets (Avery 2002) is also driving the requirement for ever more powerful supercomputing resources. As a result of the increased capability, workflow strategies of the computational scientist/engineer are now tending towards submission of concurrent jobs to the supercomputing resource, please refer to Figure 1.



g_{ss}
-0.002
-0.003
-0.005
a)
b)
c)
d)
e)
f)
Cubic micellar phase, high surfactant density gradient.
Cubic micellar phase, low surfactant density gradient.
Lamellar phase: surfactant bilayers between water layers.
Initial condition: Random water/ surfactant mixture.
Self-assembly starts.
Rewind and restart from checkpoint.
0
5000
10000
15000
Number of timesteps

Figure 1: Using computational steering techniques, the scientist is able to explore the dynamics of a running simulation in real-time by steering its course towards a desired parameter space. From (Coveney 2003).

The resulting parallel workflow is intended to facilitate comparison of results between different simulation runs rather than waiting until all jobs have been sequentially processed before an analysis is undertaken. To address this need the UK e-Science RealityGrid project (http://realitygrid.org) undertook computational studies at atomistic and mesoscale levels with the goal of improving simulation productivity by enhancing the scientist/engineer's interaction with the underlying computational model. To demonstrate the gains in increased computational power, RealityGrid was linked to the US TeraGrid (Catlett 2002; Pickles, Blake et al. 2004) to employ over 6000 distributed central processing nodes. Prior to the introduction of the grid such computing jobs would typically take days or weeks to complete. Within RealityGrid the grid has achieved a dramatic speed increase from 26 days of continuous computing to less than 48 hours (Fowler, Coveney et al. 2004). In this example, the scientist's workflow involved launching an initial application then monitoring its progress before spawning ten or more concurrent simulations. Each spawned simulation was not quite independent but was derived from a checkpoint part way through the previous simulation. As the simulations proceed the scientist/engineer could monitor the convergence or otherwise of the group average. Decisions can be taken at various intervals to terminate, extend or launch new jobs. In this application, second or further chained simulations cpuld be launched at the same time as the first, or a short while later. Consequently, even within a 48-hour simulation period the scientist/engineer had to monitor their simulation – this means being able to connect to the simulation and check the parameters of interest. Unfortunately, being tied to the desktop for a long time is a distinct disadvantage so a solution to allow the scientist/engineer to operate away from their desk was highly desirable. Crucially, the required complexity to control and interact with the distributed computing/visualization environment should not be under-estimated. Output from many

computational simulations is a complex 3D visualization which raises the question of how best to deliver and interact with the system as a remote user. Large scale computing tasks can take many hours or even days to execute and during their execution the scientist/engineer often needs to access the simulation to monitor and control (steer) their job. Simply letting a grid job run on without user intervention and steering can incur expensive computer time if the result diverges from that expected. At the moment access to these resources is achieved through reasonably high-end desktop computers but these are impractical for access whilst away from the office or laboratory.
To address these issues a user centered approach was adopted (Brooke, Coveney et al. 2003; Coveney 2003; Bradley 2004; Kalawsky, O'Brien et al. 2005) to identify required system and performance requirements. To provide a sound basis a system engineering framework was employed to define architecture requirements and support the development of a ubiquitous interface device that could facilitate 24-hour access to the computing resources when the scientist/engineer are away from their desk. The user requirement also called for interactive control of jobs as well as the provision of an interactive 3D visualization of the results as they are computed. A trade-off analysis was undertaken involving all the stakeholders and the proposed 'ideal' device, from a practical viewpoint was a personal digital assistant (PDA), commonly known today as a 'tablet device'. Clearly such devices do not possess the computational power, memory or visualization resources to even begin handling the simulation results. However, rather than abandon the idea we decided to continue with the requirement in order to drive out specific system requirements. Despite these issues we were indeed able to provide a highly effective PDA interface to grid based simulations that meets the extremely demanding user needs in terms of interactive visualization performance (which at first seemed impossible). The systems engineering process was largely responsible for characterising the required system architecture and mapping this against user needs.

Requirement for a systems engineering perspective on complex grid based computing environments

Whilst the grid facilitates submission and scheduling of jobs over a distributed computing environment there is an inherent degree of complexity in actually launching computational jobs and monitoring their progression. Consequently, the grid currently remains extremely complex and potentially unusable (Harting 2004) in terms of its configuration, infrastructure, resource scheduling, job launching and poor user interaction. However, now that grid enabled systems are beginning to deliver important scientific results there is an urgent requirement to ensure the grid is user friendly (Chin and Coveney 2004). If the uptake of the grid is to extend into the wider scientific/engineering community it must be made more accessible by providing more effective user interfaces. In the majority of cases the scientist/engineer requires to periodically observe the results from their jobs and interact with the simulation as it proceeds. Consequently, visualization has become an essential tool in order to interpret large data sets generated by scientific instruments such as telescopes, microscopes, particle accelerators, imaging machines and distributed simulations. The underlying complexity and sheer quantity of data generated by modern scientific visualization tools make it virtually impossible to process information numerically. Whilst, automated data analysis and reduction can sometimes play a role in the process, effective understanding is only achieved by human interpretation of the visualization. It is now recognised that interactive exploration of a visualization greatly increases a researcher's perception and understanding of the data space (Kalawsky, O'Brien et al. 2005). Understanding the 3D relationships in visualization imposes additional demands on the user interface because user interaction becomes a visuo-manual activity as opposed to a text input approach. The value of user interaction (viewpoint manipulation, feature flythrough and general object manipulation) should not be under-estimated. Quite often with very complicated 3D renderings the information of interest is hidden until the scientist/engineer interacts with or moves the viewpoint of the visualization. This interaction can be regarded as two categories:

- Exploration of the 3D scene. Once a visualization data set has been produced the scientist often wants to explore interactively the resulting visualization to observe areas of interest. This exploration may involve operations such as image translation, rotation or navigation through the 3D data set.

- Exploration of the visualization parameters. Parameter exploration through manipulation of the initial control parameters allows the researcher to explore 'what if?' scenarios and helps reveal otherwise occluded data and relationships.

The inherent complexity of the grid based infrastructure can make it difficult to identify the key drivers for the system design. In order to help characterise the system in terms of performance and overall architecture a systems engineering framework was adopted. In addition, a number of perspectives for the development needed to be addressed:

- The application scientist/engineer - who’s role is to engage in scientific exploration

- The computer scientist - who is concerned with the development of scientific exploration tools to facilitate the application scientist.

- The user interface designer - who’s role is to advise on the design of the interactive user interface.

At the start of the Reality Grid project it was apparent that the end users, computer scientists, user interface designers and system developers each had a different understanding of what was required. Therefore, it was important to employ a consistent approach to the derivation of an architectural framework for the visualization system. Rather than invent a new approach to the systems architecture definition a decision was taken to employ the US Department of Defense Architecture Framework (DoDAF) (DoD 2004) as the basis for the specification and development of the systems architecture. Even though this was designed for military applications it has been found to be useful for other non military activities. Only a subset of the full DoDAF viewset was applicable to the RealityGrid project. In summary the views shown in Table 1 were created.

Applicable Viewset	Framework	Framework Product Name	Description
All Views	AV-1	Overview and Summary	Scope, purpose, intended users

All Views	AV-2	Integrated dictionary	Definitions of all terms used
Operational View	OV-1	High level Operational Concept Graphic	High level visual/textual representation of operational concept.
Operational View	OV-2	Operational Node Connectivity Description	Operational nodes, connectivity and information exchange between nodes.
Operational View	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes.
Operational View	OV-5	Operational Activity Model	Operational activities, relationships between activities, inputs and outputs.
Systems View	SV-1	Systems Interface Description	Identification of system nodes, systems, and interconnectivity between nodes.
Systems View	SV-2	Systems Communications Description	System nodes, system items and related communications.
Systems View	SV-3	Systems –Systems Matrix	Relationship between systems.
Systems View	SV-4	Systems Functionality Description	Functions performed by systems
Systems View	SV-10a	Systems Rules Model	System constraints on functionality
Systems View	SV-10b	Systems State Transition Description	System response to events
Technical View	TV-1	Technical Standards Profile	Relevant standards that apply to system

Table 1: System Architecture Products (N.B. System also refers to a computational component)

RealityGrid was not a military based project but the use of DoDAF really helped structure and articulate the overall system requirements within a clearly defined framework. This process made it easy to derive actual system needs and communicate these to the implementation teams. In order to capture and refine user interface requirements a human factors audit was also employed to understand current practice and provide a detailed context sensitive basis of how RealityGrid scientists undertake their research. The audit was structured to capture the following information: Scientific application details, Technical environment, Task characteristics, User environment, Physical environment, User types and Special factors. Consequently, to derive an optimal design it was necessary to consider many inter-related factors such as:

- Understanding of how the users utilise their tool sets
- Consideration of what people are good and bad at
- Identifying what might improve the way people do things currently

- Listening to what users want
- Involving the end users in the design process
- Employing tried and tested user-based techniques during the design process
- Understanding of the limitations of the human physical and cognitive system

From the outset the systems engineering methodology ensured the resulting system was fit for purpose and easy to use. An iterative approach to the design of the user interface design activity was followed. The purpose of the iterative review was to verify the workflow employed by the scientist/engineer with a view to identifying key usability issues. A further particularly challenging aspect for the usability case was the need to consider a wide range of applications. Within RealityGrid a number of different scientific tools were being used and they each presented a different interface to the user. Unfortunately, the scientist was very reluctant to consider changes to the 'look and feel' of their familiar user interface. This reluctance was based on the user's familiarity with the tool and also the fact that many applications were proprietary and the underlying interface was fixed by the application. Whilst it could be argued that these tools do not necessarily represent an ideal user interface the design of these interfaces have evolved during the development of the individual tools. Ideally, what was required was a form of consistent user interface that maps across all the RealityGrid applications and is still recognizable by the scientist. An additional challenge was that the software tools used by the scientists were not necessarily compatible with each other. This potentially makes the job of designing a consistent interface very difficult. A key requirement to emerge from the human factors audit was the need for a user interface that facilitates access to the computational resources at any time and from any location.

Usability and Computational Steering

Usability is extremely context dependent (Newman W. M. and Taylor A. S. 1999) and largely determined by the interaction between users, tools and the problem domain. Improving the usability of e-Science enterprises is regarded as important (Harting 2004). The focus for our e-Science research has focussed on the need to facilitate improved computational steering and visualization. Computational steering has emerged as an iterative process (Mulder, van Wijk et al. 1999) to solve the limitations in earlier file based scripted job control systems by offering a direct, real-time user interaction mechanism. A computational steering system requires the scientist/engineer to define a series of independent variables to be manipulated during the execution of a given simulation. As the simulation proceeds the resulting dependent variables representing the state of the simulation, as well as the results of interest are displayed to the scientist. The scientist/engineer then fine tunes (steers) the computational model whilst it is running, observing results and changing input parameters until the desired results are obtained, or have reached the point where no further iteration is required (Leech 1996). The scientist/engineer is an important part of the feedback loop because they are in control of the parameter space being searched. The ability to control and manipulate steering parameters effectively and interactively is a major contributor to the usability of the system. Figure 2 illustrates the interactive elements within a typical computational steering system.

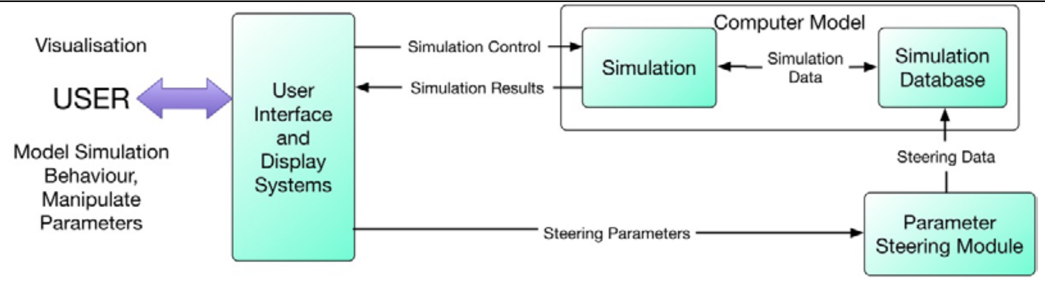
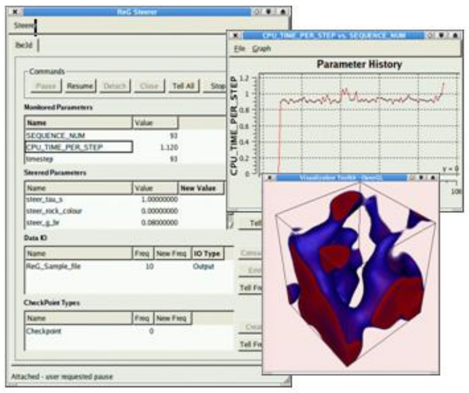


Figure 2: Manual computational steering system

A set of steering libraries have been developed that can be integrated with an application code (Brooke, Coveney et al. 2003). The steering libraries allows the user's simulation to be distributed over a grid architecture as shown in Figure 3 which shows what is involved for codes to be instrumented for computational steering through the publicly released RealityGrid Steering Library and Toolkit. A steering client user interface can be used to remotely discover and steer individual RealityGrid components on the grid, (Refer to Figure 4) through a lightweight Web Service middle tier. An important feature of the libraries is that they support dynamic connection to the steering services meaning that the user can connect and disconnect to their simulation as it executes, allowing the user to run several jobs simultaneously and switch between them. Whilst this technique improves usability there is still an onus on the user to use their desktop steering client to frequently monitor critical simulation parameters to ensure everything is running as required.



Monitored Parameters

Name	Value
SEQUENCE_NUM	93
CPU_TIME_PER_STEP	1.120
timestep	93

Steered Parameters

Name	Value	New Value
steer_tau_s	1.00000000	
steer_rock_colour	0.00000000	
steer_g_br	0.08000000	

Data IO

Name	Freq	New Freq	IO Type
ReG_Sample_file	10		Output

CheckPoint Types

Name	Freq	New Freq
Checkpoint	0	

Attached - user requested pause

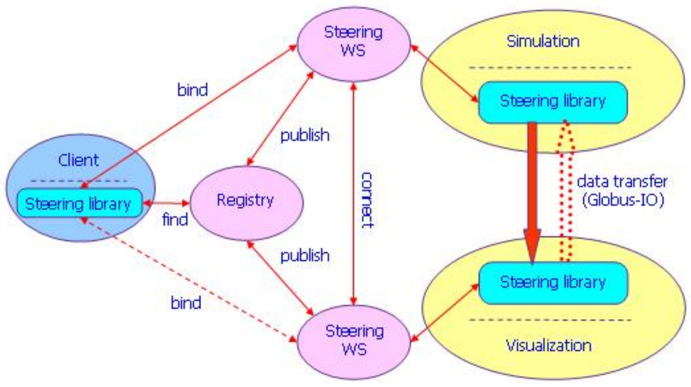


Figure 3: Simple RealityGrid Steering (Right) (Brooke, Coveney et al. 2003)). The RealityGrid Steerer provides a visual front-end to all of the Grid-based computational steering (Left).

Computational steering will remain partially supervised (until automatic goal orientated systems become available) where the output of the simulations must be checked on a regular basis for continued valid output. The job submission and resource management systems do not always guarantee a particular time-slot when the scientist's job is run. Invariably, other jobs with a higher priority seem to jump the queue. This means the scientist has to connect and disconnect regularly with the simulations in order to inspect the developing results. Unfortunately, this ties the scientist down to their desktop computer and is not very productive, especially if jobs take hours or days to execute.

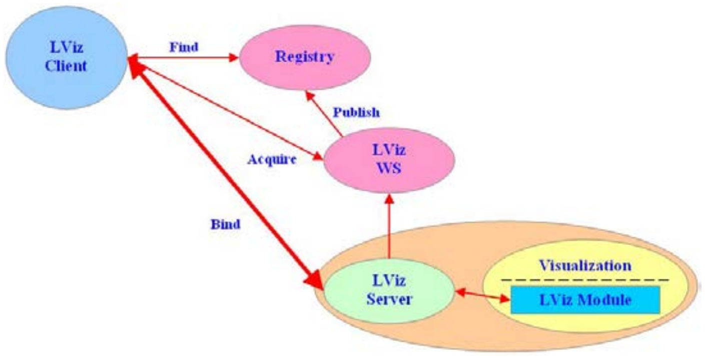


Figure 4: RealityGrid Lightweight Visualization provides a set of Grid-enabled software components and middleware

The RealityGrid Steering Library and Toolkit enables each grid component (typically a simulation but potentially a visualization) to be remotely steered through a lightweight Web Service middle tier, by means of a steering client user interface. Computational steering is achieved by exposing a set of pre-registered internal simulation code parameters over the grid. Once exposed, these parameters can be adjusted through the remote user interface and updated inside the grid-based application in real-time. Usable grid infrastructure and middleware is fundamental to the successful adoption of e-Science technology for scientific computing.

Lightweight remote visualization

The importance of remote visualization was first recognised by (Parulkar 1991) who noted the increasing reliance on the visualization of data by the scientific and engineering communities. Interactive scientific visualization requires a consistent, low-latency representation of the visualization and high band-width computation. Latency is a problem when analysing and interacting with large datasets and rendering 3-D structures, especially where the scientist is required to interactively manipulate the steering parameters of the computation. Early approaches to remote visualization relied on a client-server paradigm. The distributed nature of grid resources and indeterminate network performance can introduce serious human computer interaction issues as discussed earlier in this paper. Various techniques are available to reduce these time delays by compressing data sent over the network, thus leading to a reduction in network bandwidth. Unfortunately, this is not always commensurate with a guaranteed decrease in time delay or latency. Given the desirability to support interactive exploration it is considered good practice to separate the visualization rendering stage from the main compute loop because this will not tie down the visualization system response time (during an interactive session) to the long execution time of the computation. If the user does not have local high-end visualization systems, it is then feasible to employ a remote visualization or distributed system. It is a reasonable assumption that any operation on a data stream has the potential to add a delay into the overall system pipeline, or possibly worse, to introduce a degree of distortion into the original dataset. The real question is how to trade off frame rate versus compression loss. Various techniques exist for supporting distributed visualization and specialised hardware has become available in terms of the processing that can be performed within dedicated hardware and this has led to a novel technique of remotely generating a compressed image and transmitting this to a user over a network. However, particular care has to be exercised in the use of data compression for images. There are both lossy and lossless compression techniques, the lossy compression techniques are ideal for low latency situations because of their low network bandwidth overhead requirements, but they should only be used where image compression can be tolerated.

An important user interaction issue is the effect of time delay between user input and the corresponding output from the computation. Depending on overall grid architecture the interaction time delay can be anything from fractions of a second to several minutes, or in extreme cases several hours. General advice regarding what constitutes an ideal response time in an interactive system has largely been the same for the past three decades (Card and Mackinlay 1991), (Miller 1968). For a truly responsive computational steering system (in usability terms) it is necessary to be able to update visualization viewpoints at a rate of at least 0.1s (10Hz). Unfortunately, it is impractical to expect all parts of the grid to be capable of updating at this rate. Indeed, in the case of extremely large scale computation a single iteration may still take many hours to execute. However, this does not imply that all parts of the computation need to be tied to this long lead time. By separating the visualization from the computation loop it is possible to increase the responsiveness of the user's interaction. However, the iteration rate of the visualization machine then becomes an important factor in the usability of the system. As soon as the visualization is separated from the main computation, and the user is allowed to interactively manipulate parameters which affect the final visualization, a number of important user interface requirements need to be considered. Depending on the nature of the task, close coupled interaction with the intermediate data set can be very useful, (if not essential) in understanding whether the computation needs to continue.

As an alternative to presenting high quality rendered images whilst performing interactive tasks such as navigation and image rotation it is feasible to render an image which is much simpler or has reduced quality. This means the image is purposely degraded to a point where sufficient visual cues remain and are adequate for the user to navigate around the dataset. As soon as the user interacts less with the visualization and finds the point of interest, the visualization is progressively rendered at a much higher level of fidelity or with no compression. This technique has the advantage of minimising network bandwidth with increased frame rate during periods of high user interactivity. This visualization technique is known as ‘adaptive rendering’ and is based on a very similar approach that has already been successfully applied to virtual environments (Funkhouser and Sequin 1993), (Watson 1998). In grid based environments, different algorithms are required to control the degree of adaptivity in order to maintain an adequate level of accuracy in the display during interaction and when viewing the image from a static viewpoint. These and other more sophisticated rendering techniques are able to make a significant contribution to reducing network bandwidth whilst providing an appropriate level of image fidelity during interaction.

The case for a lightweight computational steering and visualization system

The DoDAF architectural views allowed the application scientists to express their requirements in a format that was easy for them and which were highly readable by the computer scientists/system developers. Analysis of the data from the RealityGrid systems engineering process identified the following requirements for a lightweight user interface for grid applications:

- Remote access to grid based resources at any time and from any location
- Wireless connectivity
- Intuitive interaction – with minimal learning
- Consistent user interface between standard desktop scientific applications
- Connectivity with standard (proprietary) and bespoke computation grid enabled applications
- High resolution, highly interactive 3D visualization
- Ability to graph data as it is computed
- Ability to interactively steer a computation through a set of user selectable input parameters
- Collaborative visualization – shared visualization with other users
- Minimum latency during interaction

Therefore, to extend the established computational steering capabilities of RealityGrid and other grid based applications, a lightweight visualization environment was created to provide a set of grid-enabled software components and middleware. These were designed to allow high-end visualization applications to run over a lightweight software-based middle tier. The basis of lightweight visualization was to facilitate efficient and collaborative remote user access to high-end visualization on the grid. The resulting system can be hosted on a PDA, tablet, mobile phone, laptop or desktop computer and comprises four Grid-enabled components (Refer to Figure 5):

a) A Lightweight Visualization Module (LViz Module) is embedded within a visualization code to expose an external means of accessing the internal visualization functionality.

b) The Lightweight Visualization Server (LViz Server) manages remote client connections, communicates client interaction commands to the visualization and serves the remote client with visually rendered image data.

c) The Lightweight Visualization Web Service (LViz Server) provides resource discovery capabilities and helps the client to establish a direct connection to a visualization application on the grid.

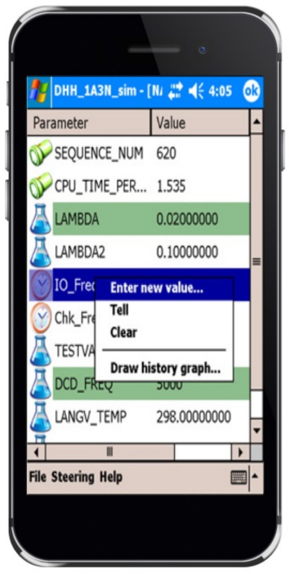
d) The client or user interface through its interaction with the LViz WS configures itself to provide a lightweight extension of the visualization application's indigenous user interface. This approach to providing tailor-configured user interfaces has enabled RealityGrid Lightweight Visualization to provide a wealth of visualization functionality through the many bespoke and commercial visualization applications that are currently deployed in RealityGrid.

Implementation of the RealityGrid PDA client

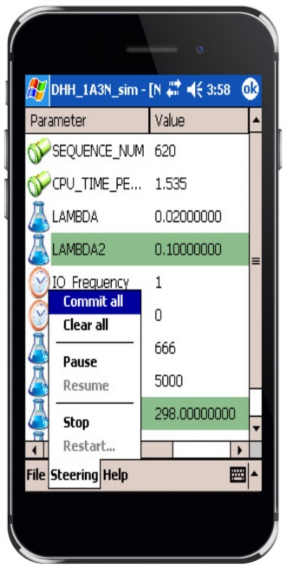
The RealityGrid PDA Client has been designed as an intuitive visual front-end, enabling the user to discover their applications on the RealityGrid, steer their simulations in real-time and interact with high-end visualizations as if the supercomputing applications were running locally. The RealityGrid PDA Client provides convenient, remote, handheld access to the three primary aspects of RealityGrid supercomputing functionality:

Resource discovery enables the scientist to find experiment software components on the grid and bind the remote user interface to the grid applications in order to support real-time interaction. The actual location of a RealityGrid application remains transparent to the user throughout the resource discovery process. The RealityGrid PDA Client's first task at start-up is to contact a user-specified RealityGrid Registry and retrieve its contained list of all currently deployed and active jobs on the grid. Through the PDA interface the user is then able to select an individual job to interact with. The RealityGrid PDA Client will then establish a remote connection or 'attach' to the job over the grid and display the appropriate computational steering or visualization interface, depending on the type of job selected.

Real-time computational steering is supported by a built-in computational steering interface (Refer to Figure 5). RealityGrid computational steering functionality remains within a constant subset between each individual simulation code. Thus the client's computational steering interface has been developed as a generic front-end to each separate RealityGrid simulation. The computational steering interface of the RealityGrid PDA Client has been developed to specifically provide the same level of usability plus all the facilities that the user can access on their desktop or laptop computer through the established RealityGrid Steerer application (Refer to Figure 3).
Whilst attached to a running simulation the RealityGrid PDA Client updates its user interface with fresh simulation parameter data (Refer to Figure 5), which is requested and retrieved from the SWS at an interval of once every second. This received data can also be plotted, as it is received, in a 2D line graph (Refer to Figure 5). The tasks of retrieving parameter data, updating the user interface and graph plotting are all each processed separately by the RealityGrid PDA Client using multiple threads. The use of multithreading in the RealityGrid PDA Client has enabled the computational steering interface to remain active and responsive to the user at all times, whilst constantly updating simulation parameter data from the grid in the background.
The user remotely steers a simulation by entering new parameter values into an input dialog, which is displayed within the RealityGrid PDA Client interface. The inputted new parameter values are then despatched to the SWS where the appropriate action is then taken on the grid to update the relevant parameters within simulation. Once the required steering activity has been instigated within the simulation it is reflected back within the PDA interface, through a client-requested parameter update, almost instantaneously.


DHH_1A3N_sim - [N/ 4:05

Parameter	Value
SEQUENCE_NUM	620
CPU_TIME_PER...	1.535
LAMBDA	0.02000000
LAMBDA2	0.10000000
IO_Freq	
Chk_Fre	
TESTVA	
DCD_FREQ	5000
LANGV_TEMP	298.00000000


DHH_1A3N_sim - [N 3:58

Parameter	Value
SEQUENCE_NUM	620
CPU_TIME_PE...	1.535
LAMBDA	0.02000000
LAMBDA2	0.10000000
IO_Frequency	1
	0
	666
	5000
	298.00000000

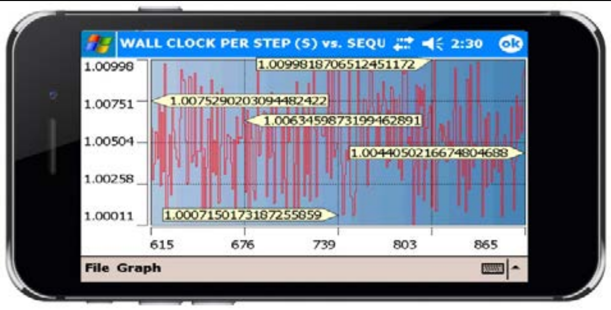


Figure 5. The computational steering interface of the RealityGrid PDA Client

Visualization is an accompaniment to computational steering. The most innovative feature of the RealityGrid PDA Client is its ability to provide remote handheld user access to the high-end 3D visualization capabilities of RealityGrid (Refer to Figure 6). RealityGrid provides grid-based hosting support for a wide range of existing or established scientific visualization codes. The RealityGrid PDA Client has initially been developed to provide a remote handheld interface to the RealityGrid-instrumented VMD application (Refer to Figures 6 and 7). The client's user interface is automatically configured to the indigenous VMD user interface when the user attaches the client to an instance of the VMD visualization application on the grid. The RealityGrid PDA Client's configurable visualization interface has been specifically designed and implemented to provide the same user services and level of user interaction as that of the integrated VMD font-end (Refer to Figure 7), or any other supported visualization code, creating a sense of continuity for the user on their remote PDA interface.

The RealityGrid PDA Client remotely controls a visualization application on the grid by remotely issuing text-based commands to the LViz Server, derived from captured user interactions in the interface. Commands are executed within the visualization code by the embedded LViz Module, which is also responsible for capturing two-dimensional rendered visualization image sequences. Rendered images are encoded and served back to the client over a connecting TCP/IP socket. The RealityGrid PDA Client incorporates a generic interface, allowing the user to remotely configure the image encoder settings of the LViz Server. This facility has been implemented in the RealityGrid Lightweight Visualization architecture to enable the user to apply varying levels of image compression, in order to adapt the client for use on low or medium bandwidth wireless networks with higher levels of image compression yielding increased image serving throughput on slower networks. Rendered frames are encoded by the LViz Server using a combination of the Portable Network Graphics (PNG) and the Joint Photographic Experts Group (JPEG) compression algorithms, providing a means for the user to view visualized data on the PDA in both lossless and lossy formats.

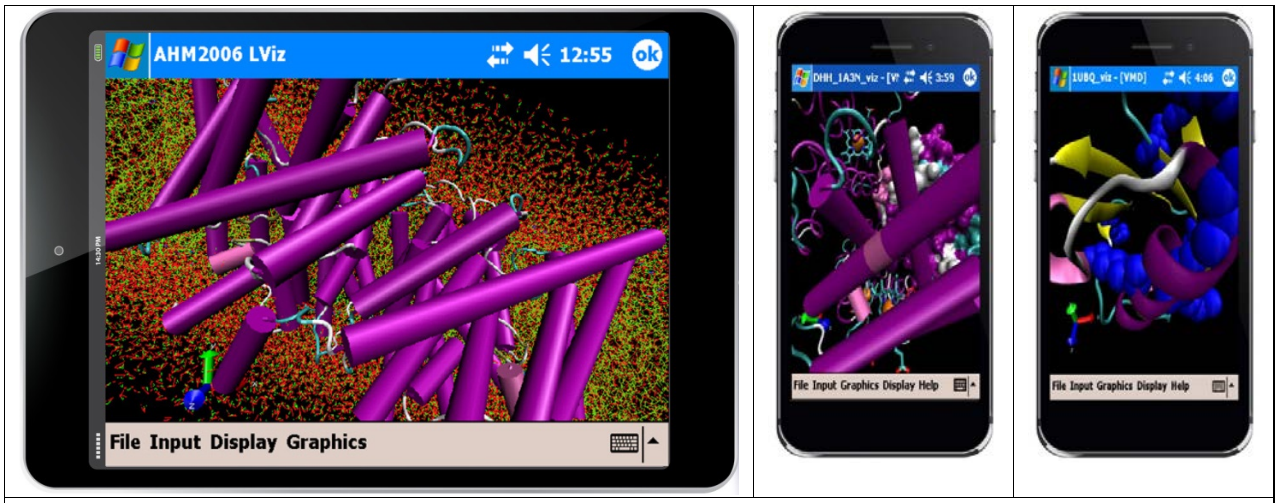
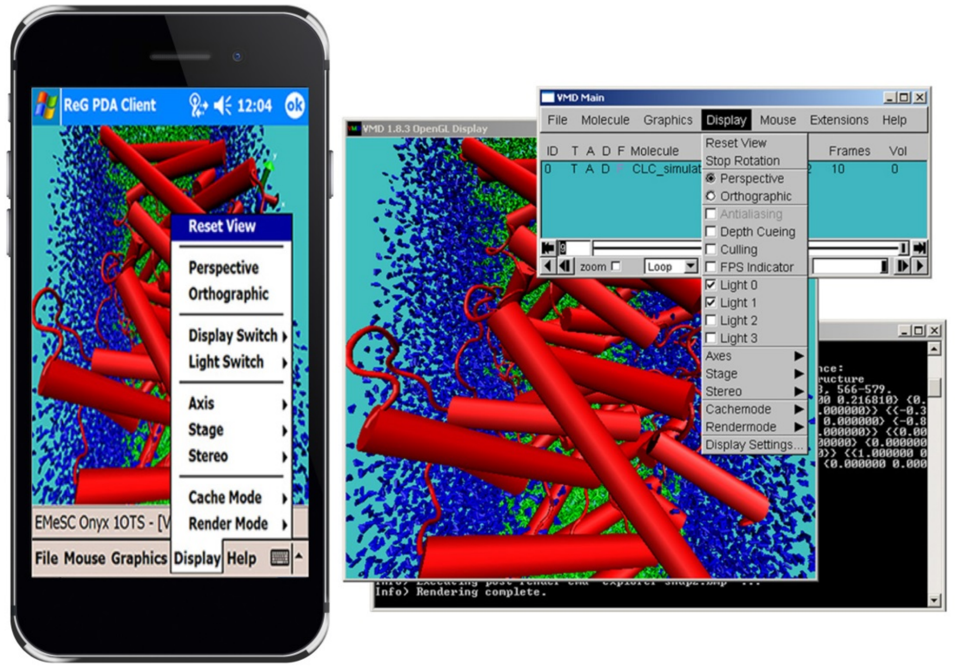

AHM2006 LViz
12:55
ok
File Input Display Graphics
DHH_1A3N_viz - [V
3:59
ok
File Input Graphics Display Help
1UBQ_viz - [VMD]
4:06
ok
File Input Graphics Display Help

Figure 6: The RealityGrid PDA Client is able to configure itself to provide remote user access and interaction for the VMD visualization application on the Grid.

When designing the visualization interface for the RealityGrid PDA Client, it was essential to provide the user with a real sense of engagement with their visualization application and not leave them feeling remote or disconnected from it in any way when using the PDA. Traditional user interaction with scientific visualization (on a desktop or laptop computer) typically involves the user dragging the mouse cursor across the display, and the visualization viewpoint (or model) subsequently responds by rotating, panning, zooming (in or out) or translating. A key objective for the RealityGrid PDA Client was to present an impression that the visualization was running locally on the handheld device, in much the same way as it would be on a desktop system (either locally or through a remote user interface). To help achieve this desired effect, the RealityGrid PDA Client implements a user interaction mechanism derived from the conventional desktop interaction model.

The user interacts remotely with visualization through the RealityGrid PDA Client by dragging their PDA stylus either horizontally or vertically across the display screen. The client captures the stylus movement and transmits an appropriate interaction message over the grid to the visualization using one of three pre-defined and user specifiable interaction modes: rotate, scale or translate. The RealityGrid PDA Client also provides two additional user-specifiable modes for capturing the movement of the stylus. These are incremental and terminal. The incremental capture mode despatches continuous interaction instructions as the stylus is gradually moved across the display. In contrast, the terminal capture mode waits until the user has completed a full movement, signified by the stylus being lifted away from the display screen, before despatching a single instruction for the whole movement to be executed in one cycle. Incorporating different modes of input capture as with image encoding, has enabled the user to adapt the performance of the RealityGrid PDA Client to remain responsive to changeable wireless network conditions.



```
Info> Rendering complete.
```

Figure 7. The RealityGrid PDA Client visualization interface has been specifically designed to provide the same user services as the integrated VMD front-end

Concluding Remarks

In an attempt to make the grid an easier tool for scientists/engineers to use on an everyday basis we developed a grid-enabled steering and visualization lightweight client. The underpinning systems engineering architecture framework helped identify key requirements from different perspectives and in a form understandable by the end users and system developers alike. The use of DoDAF and UML has helped to articulate real system needs and expected system performance in addition to definition of the functionality of the user interface. Prior to this the system developers had difficulty in relating the implementation to the context of use. This approach has led to a system that now seriously competes with the desktop system in terms of performance and which exceeds it in terms of operation anywhere and at any location. The systems engineering approach we have followed has enabled us to fully understand the stakeholder requirements for a ubiquitous user interface. It is important to note that the systems architecture framework acted as a platform independent blueprint for the PDA client and server systems.

The concept of using a PDA to interact with a series of jobs across a supercomputing resource at first may seem strange, but when scientists/engineers realise they can use these devices to connect to their computation wherever there is a wireless (or cellular phone) network the idea becomes very compelling. The PDA's lightweight and compact design has made it an ideal candidate to deliver the much needed flexibility and convenience to the grid-based user interaction experience; offering scientists a significantly increased mobility advantage over the laptop PC, by the way in which the device can be easily transported 'hands free' tucked inside a coat or trouser pocket, or elsewhere carried discreetly about one's person. Although the PDA has been developed for computational science applications there is no reason why the same infrastructure cannot be employed elsewhere. We have successfully demonstrated an interface to a medical scanner thereby providing the healthcare consultant and surgeon with a very useful time saving device.

To qualify the effectiveness of a PDA-based user interface on the grid: the roaming computational steering client that was initially developed as part of the RealityGrid project (and upon which the research described in this paper has been founded) provided scientists with a highly usable handheld facility to monitor and computationally steer (fine tune) their high-end computational simulations, in real-time, from virtually any location at any time of day (or night). Our original concerns about not being able to provide a low latency interactive visualization have been unfounded. The use of user interfaces that conform to those used in specific applications has made it straightforward for new users to adapt to the PDA solution. This means that users can operate in a familiar environment. Finally, our approach has been to solve the user interface problem for the PDA, the methods used also map directly onto any other platform thus making the system truly ubiquitous. Others have tried and failed to produce real-time 3D visualizations on low power PDA devices. In addition, our approach is fully scalable from the PDA through desktop computers to high end computing resources. However, we believe that the systems engineering approach we have followed has enabled us to gain a much greater insight into the aspects of the system design that are crucial to the successful realisation of our concept. Finally, we have recently successfully applied the same approach to the implementation of a PDA based medical visualization system (Project supported by The Loughborough University / University Hospitals of Leicester NHS Trust Medical Research Council Interdisciplinary Bridging Award.) leading to first prize in Da Vinci Health Technology Innovation awards for developments in 3D medical visualisation.

.

Acknowledgements

The author would like to thank the dedicated teams of application and computer scientists within the RealityGrid project (University College London, University of Manchester, University of Oxford, Loughborough University, Edinburgh Parallel Computing Centre and Imperial College). Special thanks also go to Simon Nee and Ian Holmes (both formally of Loughborough University) for their contribution towards early developmental research on the RealityGrid PDA Client as well as to Andrew Porter of Manchester Computing for his invaluable assistance regarding the RealityGrid computational steering framework. This work was carried out under grant GR/R67699 from the EPSRC and within the RealityGrid project.

References

Avery, P. (2002). "Data Grids: a new computational infrastructure for data-intensive science." Philosophical Transactions: Mathematical, Physical and Engineering Sciences **360**(1795): 1191 - 1209.

Bradley, D. (2004). RealityGrid - Real Science on computational Grids." e-Science 2004: The Working Grid.

Brooke, J. M., P. V. Coveney, et al. (2003). Computational Steering in RealityGrid. Proceedings of the UK e-Science All Hands Meeting, Nottingham.

Card, S. K., Robertson, G. G., and J. D. Mackinlay (1991). The information visualizer: An information workspace. Proc. ACM CHI'91 Conf, New Orleans, LA.

Catlett, C. (2002). The Philosophy of TeraGrid:Building an Open, Extensible, Distributed TeraScale Facility. 2nd IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society.

Chin, J. and P. V. Coveney (2004). Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware.

Coveney, P. V. (2003). Computational Steering on Grids - A Survey of RealityGrid. Proceedings of the Second Annual RealityGrid Workshop, University College London, UK.

Davis, W. J. (1998). On-line Simulation: Need and Evolving Research Requirements. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. J. Banks, John Wiley**:** 465-516.

DoD, A. F. W. G. (2004). DoD Architecture Framework Version 1.0. **1, 2:** 87.

Foster, I. and C. Kessleman (1999). The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers.

Fowler, P. W., P. V. Coveney, et al. (2004). Exact calculation of peptide-protein binding energies by steered thermodynamic integration

using high performance computing grids. EPSRC e-Science All Hands Meeting, Nottingham.

Funkhouser, T. A. and C. H. Sequin (1993). Adaptive display algorithms for interactive frame rates during visualization of complex virtual environments. Computer Graphics (SIGGRAPH '93), Los Angeles, CA.

Gabbar, H., S. Shinohara, et al. (2003). "Experiment on Distributed Dynamic Simulation for Safety Design of Chemical Plants." Simulation Modeling Practice and Theory **11**(2): 109-123.

Harting, J., Venturoli, M., Coveney, P.V. (2004). "Large-scale grid-enabled lattice Boltzmann simulations of complex fluid flow in porous media and under shear." Philosophical Transactions: Mathematical, Physical and Engineering Sciences **362**(1821): 1471-2962.

Hey, T. and A. E. Trefethen (2003). The Data Deluge: An e-Science perspective. Grid Computing - Making the Global Infrastructure a Reality. F. Berman, G. Fox and A. J. G. Hey, Wiley.

Kalawsky, R. S., J. O'Brien, et al. (2005). "Improving scientists interaction with complex computational-visualisation environments based on a distributed grid infrastructure." Phil. Trans. R. Soc. Lond. A **363**(1833): 1867-1884.

Korn, S., G. R. Burns, et al. (1999). "The Application of Multiparadigm Simulation Techniques to Manufacturing Processes." International Journal of Advanced Manufacturing Technology **15**(12).

Leech, J., Prins, J.F., Hermans, J. (1996). "SMD: Visual steering of molecular dynamics for

protein design." IEEE Computational Science & Engineering **3**(4): 38–45.

Liere, R. v., J. D. Mulder, et al. (1997). "Computational steering." Future Generation Computer Systems **12**(5): 441-450.

Miller, R. B. (1968). Response time in man-computer conversational transactions. Proc. AFIPS Fall Joint Computer Conference.

Mulder, J. D., J. J. van Wijk, et al. (1999). "A survey of computational steering environments." Future Generation Computer Systems **15**(1): 119 – 129.

Newman W. M. and Taylor A. S. (1999). Towards a Methodology employing Critical Parameters to deliver Performance Improvements in Interactive Systems. Interact '99 Conf.

Parulkar, G. M., Bowie, J., Braun, H. Guerin, R., Stevenson, D. (1991). Remote visualization: challenges and opportunities. Proceedings of the 2nd conference on Visualization '91, San Diego, California, IEEE Computer Society Press.

Pickles, S. M., R. J. Blake, et al. (2004). The TeraGyroid Experiment. Workshop on Case Studies on Grid Applications - held in conjunction with GGF10, Berlin, Germany.

Watson, B., Walker, N., Ribarsky, W.R., Spaulding, V. (1998). "The effects of variation in system responsiveness on user performance in virtual environments." Human Factors **40**(3): 403-414.

Yucesan, E., Y.-C. Luo, et al. (2001). "Distributed Web-based Simulation Experiments for Optimization." Simulation Practice and Theory **9**(1): 73-90.

Zulch, G., U. Jonsson, et al. (2002). "Hierarchical Simulation of Complex Production Systems by Coupling Models." International Journal of Production Economics **77**(1): 39-51.

BIOGRAPHY

Professor Roy S. Kalawsky Ph.D, M.Sc., B.Sc., C.Eng, FIET, FRSA is Director of the Advanced VR Research Centre at Loughborough University. Before joining Loughborough University in 1995 he worked for BAE Systems and was responsible for Advanced Cockpit Research across the Military Aircraft Division. In 1993, he received the Royal Aeronautical Society medal for developing the UK's first virtual cockpit. Kalawsky joined BAE Systems in 1978 as a systems engineer and researched advanced cockpits for fast jet aircraft.