

Modelling and Simulation for e-Social Science through the Use of Service-Orientation and Web 2.0 Technologies

Paul Townend¹, Jie Xu¹, Mark Birkin², Andy Turner², Belinda Wu²

¹School of Computing, University of Leeds, LS2 9JT

²School of Geography, University of Leeds, LS2 9JT

pt@comp.leeds.ac.uk

Abstract. This paper reports on the progress that has been made in the system infrastructure/visualisation aspect of the Modelling and Simulation for e-Social Science (MoSeS) project at the University of Leeds. Until recently, user interaction with MoSeS models has been achieved through a series of JSR-168 compliant portlets, which offers users a web-based interface to MoSeS. This paper presents the new service-enabled MoSeS architecture, and seeks to discuss both the advantages and disadvantages of the move towards service-orientation, as well as describing the use of Web 2.0 technology within MoSeS in greater detail, and discussing the benefits of using this technology in MoSeS.

1. Introduction

MoSeS (*Modelling and Simulation for e-Social Science*) is a research node of the National Centre for e-Social Science (NCeSS). MoSeS aims to use e-Science techniques to develop a national demographic model and simulation of the UK population, specified at the level of individuals and households, stretching thirty years into the future. The specific aims of the MoSeS project are as follows:

- 1) To create a flagship modelling and simulation node, in which the capabilities of Grid Computing are mobilised to develop tools whose power and flexibility surpasses existing and previous research outputs.
- 2) To demonstrate the applicability of grid-enabled modelling and simulation tools within a variety of substantive research and policy environments.
- 3) To provide a generic framework through which grid-enabled modelling and simulation might be exploited within any problem domain.
- 4) To encourage the creation of a community of social scientists and policy users with a shared interest in modelling and simulation for e-social science problems.

There are an abundance of simulation games relating to people, cities and societies (past, present and future). We pose the question of what would be the impact of transferring these simulations into a real world environment. Our specific interest is in cities and regions, with an aim of building simulation models of interactions between individuals, groups or

neighbourhoods within large metropolitan areas. Such simulations can form the basis of a wide range of applications in both e-Research and public policy analysis, with potentially substantial benefits such as:

- 1) A big policy impact through the generation of effective predictions.
- 2) A potential ‘wind tunnel’ or ‘flight simulator’ analogy: planners can gauge the effects of development scenarios in a laboratory environment.
- 3) The use of simulations as a pedagogic tool allows planners to refine understanding of systemic behaviour and alternative futures, thus aiding clarity of thinking and improved decision-making.

Specifically, MoSeS aims to develop scenarios in the domains of health, transport, business. For example, one health scenario would be to provide perspectives on medical and social care within local communities for a dynamic and ageing population. A scenario in the transport domain might concern the sustainability of transport networks in response to demographic change and economic restructuring: for example, what kind of transport network is capable of sustaining long-term economic growth in West Yorkshire, Greater Manchester, and the intervening areas – the ‘Northern Way’. A scenario in the business domain might include the impact of diurnal population movements on retail location and profitability; or the impacts of a changing retirement age on personal wealth and living standards.

The MoSeS project stands to benefit from e-Science technologies in a number of ways; in particular, the simulation model draws on diverse, virtualised data sources, deploys models which are richly specific and therefore computationally intensive, and provides outputs to a spatially distributed community of researchers and policy-makers. MoSeS is building relationships with policy users in Social Services, Health Care Trusts, urban planning, consultancy and other domains in order to demonstrate the viability and potential impact of simulation modelling, enabled by e-Science.

Until recently, user interaction with MoSeS models has been achieved through a series of JSR-168 compliant portlets [1], which offers users a web-based interface to MoSeS. The interface facilitates the creation, visualisation and analysis of simulation results. Statistics, charts, tables and simple geographic maps of the results can be generated. For example, a user can generate pie charts showing racial distribution, or maps showing car ownership in a specific area.

These facilities are effective as an initial demonstrator, but in order to exploit the full potential of e-Social Science technologies, more flexible and advanced interfaces to MoSeS are necessary; this paper explores new capabilities offered by exposing MoSeS functionality through *service-orientation* and *Web 2.0* technologies. Service-orientation is emerging as a highly useful means of developing flexible, agile, and dependable software systems. A service can be defined as “*a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.*”, whilst a service-oriented architecture can be defined as “*an application architecture within which all functions are defined as independent services with well-defined invocable interfaces, which can be called in defined sequences to form business processes.*”

This paper seeks to discuss both the advantages and disadvantages of the MoSeS project’s move towards service-orientation, as well as describing the use of Web 2.0 technology within

MoSeS in greater detail, and discussing the benefits of using this technology in MoSeS. Section 2 discusses the older, non-service-oriented MoSeS architecture, and looks at some of its inherent disadvantages. Section 3 introduces the new service-oriented MoSeS architecture, and details its benefits. New Web 2.0 functionality is discussed in Section 4. The paper is then concluded in Section 5.

2. The Older, Non-service-oriented MoSeS Architecture

To achieve the aims detailed in section 1, the software architecture employed by the MoSeS project needs to be capable of securely storing large quantities of simulation data, dynamically retrieving diverse data from spatially distributed resources, utilising the capabilities of high performance computing (HPC) resources, and visualising the results of simulation runs. In addition to this, a high level of flexibility is very much desired, to enable MoSeS software to be quickly adapted to utilise new datasets, cope with changes in the structure of existing datasets, perform simulations based on new attributes, etc. In order to achieve this, earlier stages of the MoSeS project utilised the architecture shown in Figure 1.

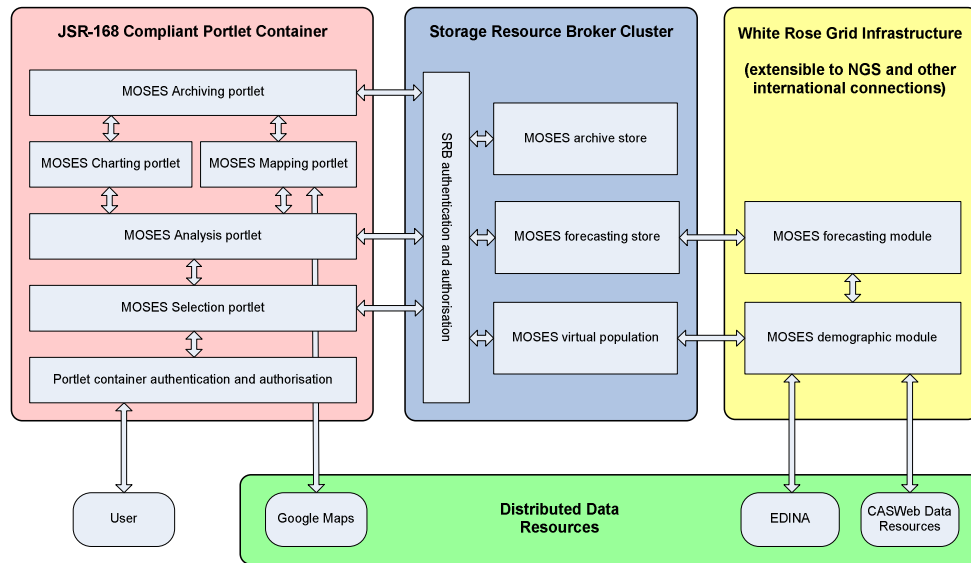


Figure 1. The older, non-service-oriented MoSeS architecture.

There were three major components to this architecture; computationally intensive demographic and forecasting modelling, virtualised storage resources brokered through the use of a Storage Resource Broker (SRB) [2] cluster, and the collection of JSR-168 compliant portlets that formed the user interface to the project. This architecture was designed to take into account the distributed and decentralised nature of e-Science research; for example, data resources are distributed both spatially and across organisational boundaries, and the computational requirements of the MoSeS forecasting and demographic modules are handled through parallel processing on the White Rose Grid [3] and the UK National Grid Service [4]. SRB resources were used to provide the considerably large storage facilities that MoSeS requires, and the user interface to MoSeS was achieved through the development of a set of specific portlets, which could be grouped according to the needs of each individual user. Visualisation of results was in part achieved through interactions with third party mapping software (specifically, Google Maps). The access control mechanisms provided by this architecture were provided through the authentication and authorisation mechanisms employed by both the JSR-168 compliant portlet container (in the case of earlier

demonstrators, Gridsphere) and the SRB installation. Security is very important for MoSeS as it incorporates confidential data (health and census data). It is imperative that access to such data and its disclosive derivatives is controlled so that only licensed parties are given access and that the license conditions of data from the system are clear.

This architecture proved to be reasonably successful; however, as the MoSeS project progressed, it became increasingly apparent that the architecture lacked the flexibility to quickly adapt to changes and new requirements. This was due to the following factors:

- 1) *A basic user interface.* The exclusive use of a JSR-168 compliant portlet interface to the MoSeS system had a number of disadvantages. Firstly, the user-interface itself was necessarily basic in order to be compatible with the JSR-168 standard; menus had to be implemented as HTML forms, while all visualisations of data (such as maps of areas) had to be represented as raster images embedded within the HTML code, with little dynamicity other than that provided by the HTML image map feature. This became an increasing impediment when richer, more interactive visualisations were required.
- 2) *A single point of entry into the system.* The ability to integrate MoSeS functionality into other systems was poor; any e-Social Science application wishing to interact with MoSeS software could only access the MoSeS software through the JSR-168 interface, and so would have to implement “web scraping” – i.e. pretending to be a web browser user, and extracting data from the returned HTML code. This is both inelegant and fragile, as such a system will break if the format of MoSeS pages changes in any way.
- 3) *Scalability.* Although the most computationally intensive aspects of MoSeS - namely the creation of a virtual population, and the dynamic simulations run on top of it - are performed “offline” from an end user’s perspective, the process of analysing the results of a simulation (aggregating values, performing comparisons, etc. on up to 60 million individual agents) is non-trivial and with the old MoSeS architecture, was performed by a single portlet located on a single machine. Thus, if a user accessed the analysis portlet whilst another user was in the midst of executing an analysis, the performance of both jobs would be affected.

In order to address these issues, it was decided to *service-enable* MoSeS functionality. Service-orientation aims to facilitate the development of complex, dynamic and inter-organisational systems. It is also designed to greatly simplify the process of integrating existing legacy systems, and has a profound impact on the software development process. The benefits of bringing this to MoSeS, as well as an overview of the actual architecture MoSeS has switched to, are explored in the following section.

3. A Service-Oriented MoSeS Architecture

By re-writing the existing MoSeS portlet functionality as a set of invocable services, a number of advantages over the older, non-service-oriented approach were introduced. These can be summarised as follows:

- 1) *Multiple entry points to the system.* By exposing MoSeS functionality as a set of invocable services, it becomes possible to quickly create new user interfaces to MoSeS. The functionality of each service can be utilised by programs written in

any language, and GUI clients can be created as front ends to both workflows (which can be defined as the description of the sequence of services that must be invoked to form a given scientific process) and individual services. Another important aspect of having multiple entry points into the systems is that other e-Social Science applications can now integrate with MoSeS much more easily, simply by invoking the required service - it is no longer necessary to interact with the system exclusively through the JSR-168 interface.

- 2) *Richer user interfaces.* In addition to the existing JSR-168 compliant interface to MoSeS, alternative user interfaces can be created in order to allow for a richer, more responsive and tactile user experience. For example, a Java application can be written as a front end to MoSeS, utilising the same service functionality that the Portlet interface uses, but with the addition of interactive maps, due to the Java GUI not being constrained by JSR-168 standards.
- 3) *Increased user flexibility.* In addition to pre-written analyses, users can now construct new MoSeS workflows through the use of high-level workflow engines such as Taverna and ActiveBPEL. This enables users to develop, share and re-use routines for producing specific maps, charts, and reports. Developing such workflows is a far easier and intuitive form of programming that is likely to be found easy for most users regardless of whether or not they have done programming before.
- 4) *Improved scalability.* MoSeS functionality no longer has to reside on a single machine; it is possible to install a MoSeS service (such as the analysis service) on multiple machines, and then perform load-balancing by dynamically binding to available/free service.
- 5) *Fault-tolerance.* Related to the benefits of improved scalability mentioned above, a level of fault-tolerance can be introduced into processing by, for example, invoking multiple MoSeS services in parallel and either cross-checking their results or else using the results of the first service to fail without raising an exception. This can also lead to performance gains, as multiple services can be invoked and the first result to be returned can be fed directly into an ongoing workflow.
- 6) *Ease of maintenance.* In spite of the increased number of entry points into the system, maintenance of core MoSeS functionality should not be affected, as changes made to each service will be reflected in each interface due to a clear separation between presentation and application logic layers of the system

The major drawback to service-enabling MoSeS was the slower performance for individual tasks (note this is unrelated to the scalability issue), mainly when passing large volumes of data using the SOAP [5] messaging protocol. Typically, data passed between different MoSeS services would need to be serialised into XML form, sent over HTTP, and then deserialised at the other end of the connection, resulting in noticeable delays when considering the size of data that is often required to be passed between services. A partial solution to this problem has been developed, whereby each MoSeS service stores data for a particular user session in SRB space, and invokes other MoSeS services by sending a reference to this data, which can then be transmitted in binary form.

The new service-oriented architecture for MoSeS is shown in Figure 2. As can be seen, MoSeS web services can be invoked by any number of end user interfaces, including Java GUIs, JSR-168 portlets, workflow enactment engines, etc. Services can be distributed

remotely as all data interactions are performed either through direct access with distributed resources over HTTP, or else through accessing the MoSeS SRB cluster, which in turn stores the results of the MoSeS demographic and forecasting modules.

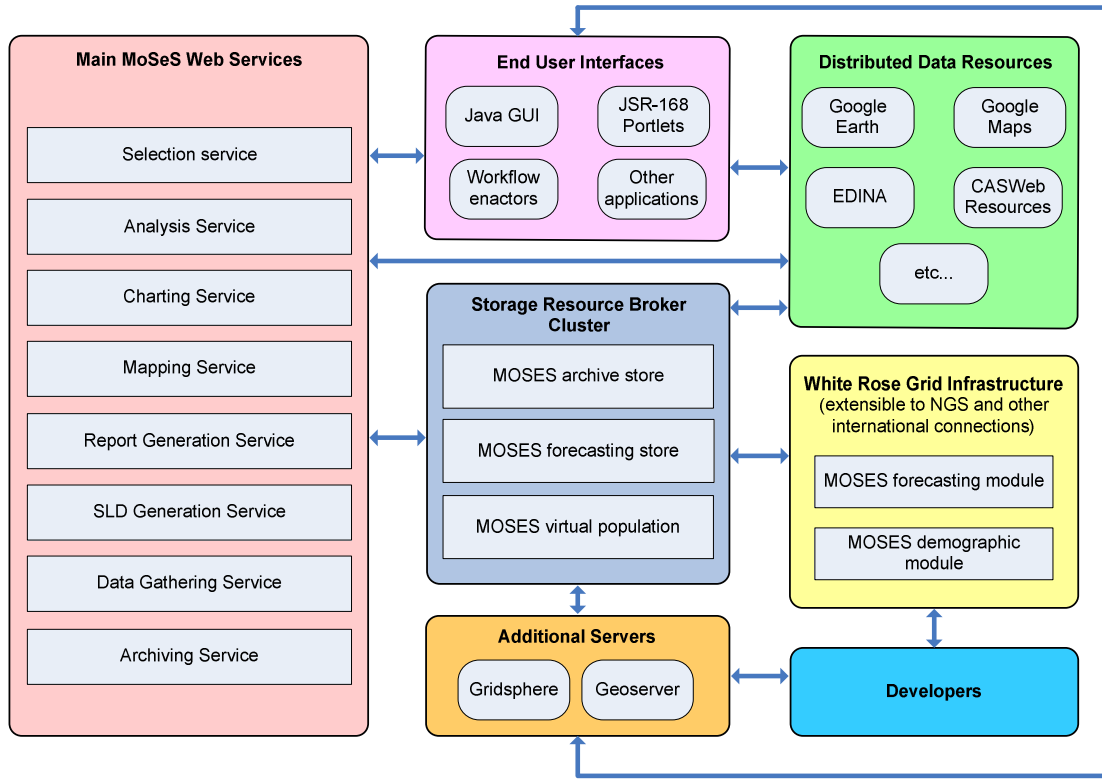


Figure 2. The new service-oriented MoSeS architecture.

Due to the new ability to use multiple end user interfaces within MoSeS, this architecture has also been designed to exploit a number of Web 2.0 technologies, particularly those related to mapping, in order to provide a more dynamic and rich user experience. Web 2.0 is defined by O'Reilly as “*the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform*” [6]. The following section details how this has been achieved.

4. Web 2.0 Technology

Although the mapping functionality previously used by MoSeS (shapefiles rendered as raster images through GeoTools libraries) provided a useful visualization of MoSeS analyses, the maps exposed little additional information that a user might require, and were limited in their interactivity, the only exception being a Google Maps interface to a subset of MoSeS data, which had to be manually created. This limited interactivity was primarily due to restrictions arising from the use of a JSR-168 compliant portlet interface. A screenshot of the older style of MoSeS map is shown in Figure 3.

In order to improve upon this situation, MoSeS has been given the capability to render map information using Web 2.0 technology, allowing for the use of both dynamic 2D and 3D maps. This is performed primarily through the use of *GeoServer* technology [7]. Geoserver is an open source, freely available implementation of the *Open Geospatial Consortium* (OGC)

Web Feature Service (WFS) [8] and Web Map Service (WMS) [9] specifications, and allows map data to be uploaded and served out in a variety of formats.

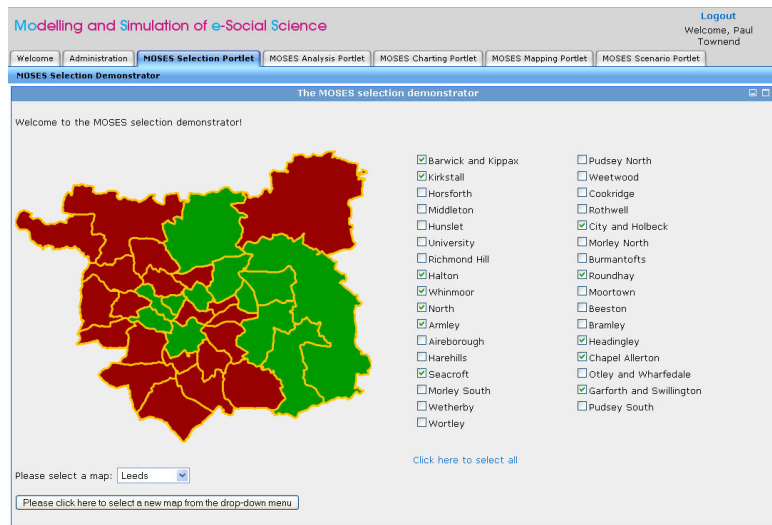


Figure 3. A static MoSeS map, inside a JSR-168 compliant portlet. Source: 2001 Census, Output Area Boundaries. Crown copyright 2003

The data from Geoserver can be viewed using *OpenLayers* [10]; this is an open-source JavaScript library for displaying map data in web browsers, with similar functionality to that offered by Google Maps [11] and Microsoft Live Search Maps [12]. MoSeS data can be viewed in OpenLayers through a new MoSeS service which can generate a *Styled Layer Descriptor* (SLD) file [13] for a given MoSeS attribute and map. An SLD file is an XML schema that describes the appearance of layers on a map; for example, should a MoSeS client wish to display a shaded map of car ownership in the Leeds area, two files will be provided: a Geotools Shapefile for Leeds, containing map information with an attribute related to Car Ownership, and an SLD file containing rules on what colour to shade a region of the map for given values of the Car Ownership attribute.

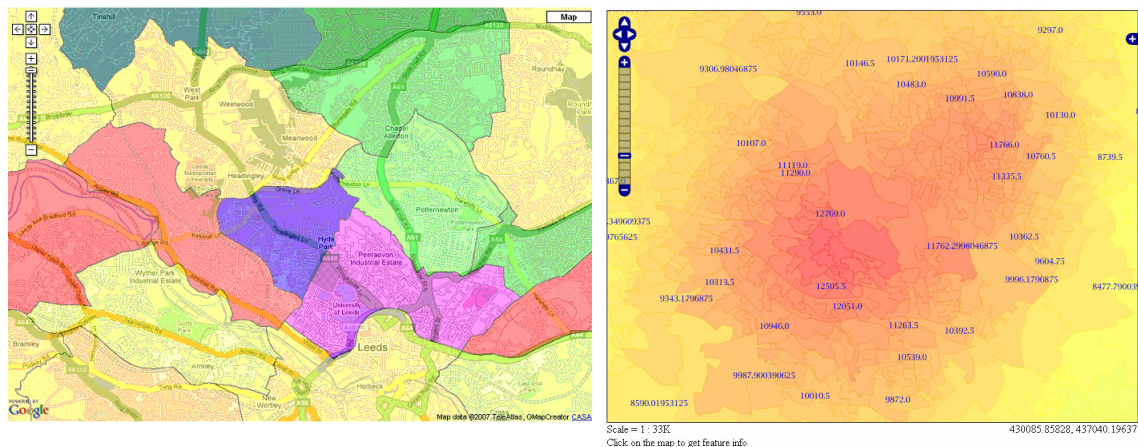


Figure 4. The MoSeS Google Maps and OpenLayers interfaces.

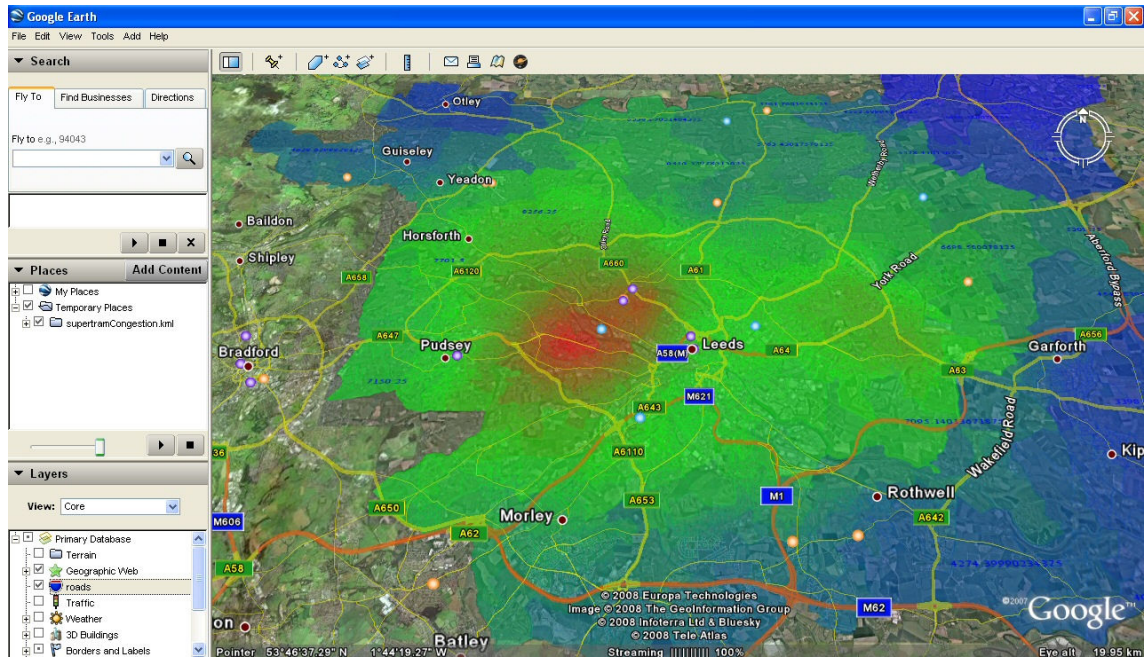


Figure 5. MoSeS data visualised in Google Earth.
Source: 2001 Census, Output Area Boundaries. Crown copyright 2003

The MoSeS Google Maps and OpenLayers interfaces are shown in Figure 4, with the Google Maps interface on the left and the OpenLayers interface on the right. In addition to the OpenLayers interface, another highly useful format served out by Geoserver is *Keyhole Markup Language* (KML) [14], an XML-based language schema for expressing geographic annotation and visualization on two-dimensional maps and three-dimensional Earth browsers. KML is the native file format of the Google Earth [15] application, and when the KML of a MoSeS map is combined with a generated SLD, impressive 3D visualisations of MoSeS data are possible, as shown in Figure 5. Once Google Earth is loaded, additional information, such as place names, roads, etc. can be layered on top of this visualization by the end user, using the Google Earth application interface.

5. Conclusions

This paper has reported on the latest progress made in the system infrastructure/visualisation aspect of the MoSeS project at the University of Leeds. Until recently, user interaction with MoSeS models had been achieved through a series of JSR-168 compliant portlets, which offered users a web-based interface to MoSeS. The functionality underpinning MoSeS has now been service-enabled, allowing for multiple entry points to the system, richer user interfaces, increased user flexibility, improved scalability, fault-tolerance, and easier system maintenance. Additionally, Web 2.0 technologies have been employed to allow for a more dynamic, rich, and tactile user experience; this has been achieved through the use of third party server software such as Geoserver, in addition to a bespoke MoSeS service for the generation of mapping styles, and allows MoSeS data to be visualised in Google Maps, Google Earth, and with OpenLayers.

The major drawback to the service-enabling of MoSeS has been the lower performance and complexity caused by the serialisation and XML transfer of complex MoSeS datasets over the SOAP protocol. Additionally, problems were encountered when trying to generate *Web*

Service Description Language (WSDL) [16] representations of the complex data contained within MoSeS data structures. A partial solution to this issue has been developed, whereby data is stored in an SRB cluster, and references to this data are passed between services; this reduces the overhead in converting MoSeS data into XML, and does not require extensive modification to the automatically generated WSDL that our current development tools were producing (which did not adequately represent the complex MoSeS data).

As part of our future work, the W3C *Message Transmission Optimization Mechanism* (MTOM) [17], which allows for the sending of binary data between Web Services, will be investigated as an optional alternative to the use of SRB. We will also be investigating methods for allowing MoSeS to be used whilst “offline”, as the current service-oriented approach assumes an internet connection is always present. We are also investigating methods for improving the security of MoSeS data further; currently, we use SRB username/password authentication to protect access to stored MoSeS data, but plan to investigate the use of *Grid Security Infrastructure* (GSI) [18] within SRB. Additionally, we hope to leverage the technology we developed in the *CROWN-C* dependable and secure Grid middleware system [19] to increase instance level security within MoSeS web services, and also to assess the robustness of the overall system using CROWN-C fault injection technology.

6. Acknowledgements

MoSeS is funded by ESRC Grant RES-149-25-0034.

7. References

- [1] JSR 168: Portlet Specification, <http://jcp.org/en/jsr/detail?id=168>
- [2] Rajasekar, A., Wan, M., Moore, R., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhu, B., Chen, S.-Y., Olschanowsky, R. (2003): *Storage Resource Broker—Managing Distributed Data in a Grid*, Computer Society of India Journal, Special Issue on SAN 33, No. 4, 42–54
- [3] Dew, P.M, Schmidt, J.G., Thompson, M., Morris, P. (2003): *The White Rose Grid: Practice and Experience*, in Proc. UK e-Science 2nd All-Hands Meeting, Simon J. Cox Eds, Nottingham Conference Center, U.K., ISBN 1-904425-11-9.
- [4] The UK National Grid Service website, <http://www.ngs.ac.uk>
- [5] W3C Recommendation (2007): *SOAP Version 1.2 Specification*, <http://www.w3.org/TR/soap12-part1/>
- [6] O’Reilly, T. (2006): *Web 2.0 Compact Definition: Trying Again*, <http://radar.oreilly.com/archives/2006/12/web-20-compact-definition-tryi.html>
- [7] Geoserver website, <http://geoserver.org/display/GEOS/Welcome>
- [8] The Open Geospatial Consortium (OGC) Web Feature Service (WFS) standard, <http://www.opengeospatial.org/standards/wfs>
- [9] The Open Geospatial Consortium (OGC) Web Map Service (WMS) standard, <http://www.opengeospatial.org/standards/wms>
- [10] OpenLayers website, <http://www.openlayers.org/>
- [11] Google Maps website, <http://maps.google.com>
- [12] Microsoft Live Search Maps website, <http://maps.live.com/>

- [13] The Open Geospatial Consortium (OGC) Styled Layer Descriptor (SLD) standard, <http://www.opengeospatial.org/standards/sld>
- [14] The Open Geospatial Consortium (OGC) Keyhole Markup Language (KML) standard, <http://www.opengeospatial.org/standards/kml>
- [15] Google Earth website, <http://earth.google.com/>
- [16] W3C Recommendation (2007): *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, <http://www.w3.org/TR/wsdl20/>
- [17] W3C Recommendation (2005): *SOAP Message Transmission Optimization Mechanism*, <http://www.w3.org/TR/soap12-mtom/>
- [18] Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S. (2003): *Security for Grid Services*, in International Symposium on High Performance Distributed Computing, pp. 48-57, Seattle.
- [19] Townend, P., Looker, N., Zhang, D., Xu, J., Li, J., Zhong, L., Huai, J. (2007): *CROWN-C: A High-Assurance Service-Oriented Grid Middleware System*, in 31st IEEE High Assurance Systems Engineering Symposium, Dallas, USA.