International Journal of Digital Earth
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tjde20

The challenges of developing an open source, standards-based technology stack to deliver the latest UK climate projections

Ag Stephens [a], Philip James [b], David Alderson [b], Stephen Pascoe [a], Simon Abele [b], Alan Iwi [a] & Peter Chiu [a]

[a] The British Atmospheric Data Centre, Didcot, UK

[b] School of Civil Engineering and Geosciences, Newcastle University, Newcastle-Upon-Tyne, UK

Available online: 24 May 2011

To cite this article: Ag Stephens, Philip James, David Alderson, Stephen Pascoe, Simon Abele, Alan Iwi & Peter Chiu (2012): The challenges of developing an open source, standards-based technology stack to deliver the latest UK climate projections, International Journal of Digital Earth, 5:1, 43-62

To link to this article: http://dx.doi.org/10.1080/17538947.2011.571724

International Journal of Digital Earth,
Vol. 5, No. 1, January 2012, 43–62

Taylor & Francis
Taylor & Francis Group

The challenges of developing an open source, standards-based technology stack to deliver the latest UK climate projections

Ag Stephens[a], Philip James[b], David Alderson[b]*, Stephen Pascoe[a], Simon Abele[b], Alan Iwi[a] and Peter Chiu[a]

[a]*The British Atmospheric Data Centre, Didcot, UK;* [b]*School of Civil Engineering and Geosciences, Newcastle University, Newcastle-Upon-Tyne, UK*

(*Received 20 December 2010; final version received 10 March 2011*)

To improve the understanding of local and regional effects of climate change, the UK government supported the development of new climate projections. The Met Office Hadley Centre produced a sophisticated set of probabilistic projections for future climate. This paper discusses the design and implementation of an interactive website to deliver those projections to a broad user community. The interface presents complex data sets, generates on-the-fly products and schedules jobs to an offline weather generator capable of outputting gigabytes of data in response to a single request. A robust and scalable physical architecture was delivered through significant use of open source technologies and open standards.

Keywords: geoinformatics; digital earth; climate change; geospatial science; geospatial data integration

1. Introduction

The issue of climate change is of great importance to decision-makers, businesses and the general public. In 2009, the UK launched a new set of climate projections. This paper discusses how the data management, geospatial services and web-interface were delivered in order to provide these projections to a wide and diverse audience. This section introduces the context and the projections themselves. Section 2 explains the design and implementation undertaken to deliver the projections to users. Section 3 discusses the various merits of the system design, technologies and standards employed within the project, and Section 4 draws conclusions from this work.

There is now clear evidence that global warming is happening and that anthropogenic emissions are a major contributor (Solomon *et al*. 2007). The resultant effects of the emissions problem are likely to be experienced by all, whether proportionately or disproportionately. The climate science community is being called upon to provide high-resolution projections of climate change that can be used by decision-makers to plan future strategy at all levels. This is important to both public and private sectors, and is particularly vital for government. In the UK, national and local government need the best projections of regional and local climate to feed into the UK Climate Change Risk Assessment (2010), cross-departmental and local strategies. The use of climate projections can aid understanding of the likely impacts

*Corresponding author. Email: David.Alderson@ncl.ac.uk

ISSN 1753-8947 print/ISSN 1753-8955 online

http://dx.doi.org/10.1080/17538947.2011.571724
http://www.tandfonline.com

of climate change on the economy and the environment, specifically in this case within the UK. There is also a growing requirement to place such data, and the data-production process, in the public domain. This is reflected in the Freedom of Information Act (2000) and the recent UK government enquiry into climate science.

The UK Climate Impacts Programme (UKCIP) manages the delivery of UK climate projections for the Department for Environment, Food and Rural Affairs (DEFRA), with the science being provided by the Met Office Hadley Centre (MOHC). Previous projections (UKCIP98 and UKCIP02) typically provided a small number of feasible future climate scenarios based on the MOHC regional climate models. Following a significant consultation process, the latest product represents a major departure from previous methods with the introduction of probabilistic projections. The UK Climate Projections (UKCP09, launched in June 2009 [Murphy *et al.* 2009]) provide both land and marine data sets for a range of key meteorological variables over a variety of temporal and spatial scales. Detailed information on the underlying scientific theory that underpins the UK climate projections can be found within the UK Climate Projections Science Report (Murphy *et al.* 2009). The following description provides an overview of this theory to provide context for the remainder of the paper. The main product is a 25-km gridded data set available for three emission scenarios for seven 30-year periods up to the 2080s. Further complexity lies in the probabilistic nature of the data set because each grid square, 30-year period, month (or season), variable and emissions scenario is provided as a set of plausible projections. Using Bayesian' statistical theory, the process, based around the MOHC HadCM3/RM3 suite of models (Gordon *et al.* 2000) and an ensemble of global climate simulations, and historical observations, derives a probabilistic range of possible outcomes, typically expressed as a probability density function. The complexity means there are potentially billions of unique data products available to the user. UKCP09 also includes daily and hourly time-series of future weather simulated for a given location and climate scenario. The UKCP09 Weather Generator (Jones *et al.* 2009) was developed at Newcastle University (NCL) and the University of East Anglia (UEA) to provide these simulations which are perturbed by the future climate projections from the MOHC. An effective way to make all this available to a wide audience is via a web-interface, incorporating download tools to facilitate research use of the data.

2. Design and implementation

2.1 Requirements gathering

Fixing the requirements at the start of any project is essential. In the case of the UKCP09 projections, there were many challenges in gathering requirements. Not only were many of the products new but the user community was diverse and varied greatly in its knowledge of the domain. The requirements were defined after reviews of existing services and detailed discussions with the data providers, key stakeholders and representative users. The main drivers for the requirements included provision of complex data, on-the-fly image generation, web access, high

usage and offline processing. The key system-wide requirements can be summarised as follows:

(1) Web-accessible services: A web front-end that calls other web services returning visualisations and data products.
(2) Scalability: The system must be able to grow with demand.
(3) Offline processing: A back-end capable of managing job queues.
(4) Robustness: The system must withstand high loads and should be stable over all timescales.
(5) Security and user access: The system must be secure against malicious attacks and must manage user access.
(6) High availability: Downtime of the system must be minimised.

A further set of requirements was focussed more specifically on the web-interface:

(7) Easy access to help and guidance throughout the system.
(8) The ability to produce customised (and publication-quality) outputs.
(9) The ability to save a request and return to it later.
(10) Presentation of information and data for non-expert users (Goodess *et al.* 2007, Hewitt *et al*. 2009).

The requirements specific to the user interface were defined following a review of existing websites, including the previous UKCIP02 project. Identifying useful features from existing systems, as well as pitfalls to avoid, gave a better understanding of the overall requirements. These requirements are revisited in Section 3 where the merits of the final solution are discussed.

2.2 System architecture

In order to create a system that was both scalable and robust, a software stack was developed that could be parallelised across multiple nodes. The approach taken replicated the hardware on high specification servers (8 Intel Xeon CPUs, 32GB Memory, 64-bit), deploying a set of identical virtual machines (VMs) on each physical server. A Xen VM image replication system was developed to allow rapid deployment and removal of each system component as required. This parallelisation is depicted in the horizontal view of the system in Figure 1. The vertical layering of the stack demonstrates the various types of VMs deployed.

The three VMs per physical machine were split based on their functional components as follows:

- User interface (UI) layer
- Service-oriented architecture (SOA) layer
- Offline processing layer

Figure 1 shows how the layered architecture is distributed across the three physical servers. The management of state is separated out into its own layer and is handled by physical servers 4 and 5. Details of each layer are outlined in the following sections.

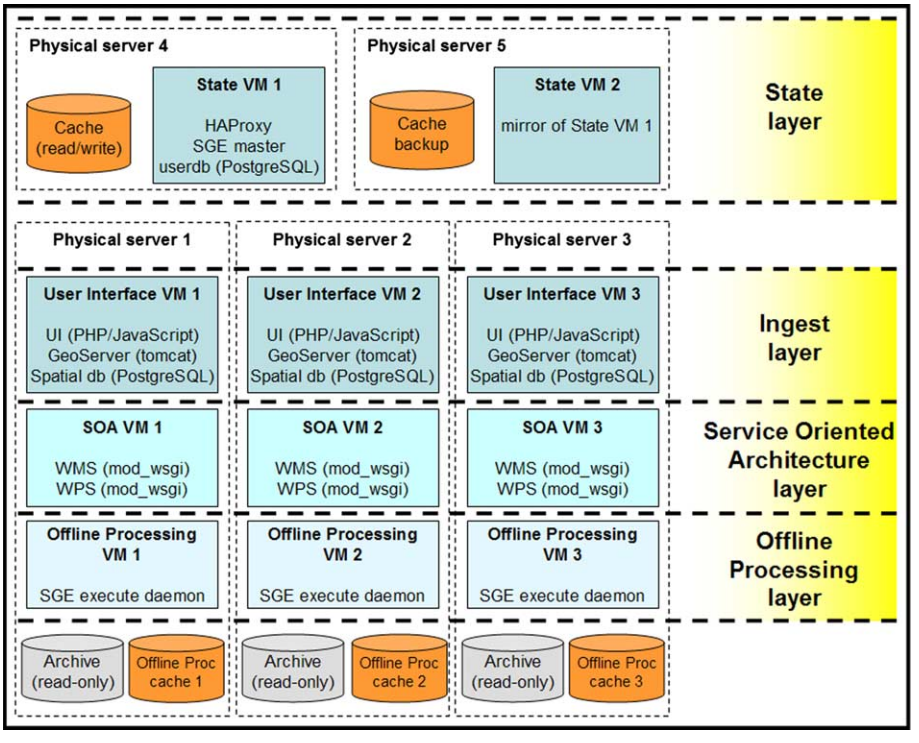

Figure 1. The UKCP09 system architecture showing the vertical (functional) layering and the horizontal (parallelised) layering.

2.3 User interface layer

The UI layer provides the web-interface that the user interacts with. The prototyping phase of the UI concentrated upon defining its layout, the process users would undertake to access the underlying data products, and considered available technologies upon which to build the interface. The UI was primarily built using PHP and JavaScript with database connections (PostgreSQL and PostGIS) to provide the presentation content for the web pages and geospatial services. The UI layer would expect to receive the highest frequency of HTTP requests. This section describes the UI design process and some of the key components employed in the final system.

The project required that the UI would function in Internet Explorer 6, Internet Explorer 7, Firefox 2+ and Mac Safari. In line with common practice, third-party JavaScript libraries were the preferred means of achieving cross-browser compatibility (Serrano and Aroztegi 2007). The Dojo and Yahoo YUI components provided the developers with a starting point for experimenting with JavaScript utility libraries. However, because of the relative infancy of these frameworks during the prototype development phase, they were subsequently replaced by the JQuery libraries. JQuery provided a framework that eased the handling of Asynchronous JavaScript and XML (AJAX) calls, and HTML document object model (DOM) interactions, and facilitated cross-browser compatibility. The extensive community

support and documentation gave the developers confidence in the longevity of the JQuery library, whilst its plug-in architecture provided access to custom add-ons developed by the community.

A prototype of the UI was developed and reviewed by prospective users in order to gain early feedback on the planned developments. The prototype attempted to encode the entire request selection logic within a single web page, as this was considered desirable following the review of other similar climate data delivery systems, such as FINESSI, a web-based tool for exploring climate change impacts in Finland. This approach was slow due to over-complex client-side scripting as it attempted to consider all the possible interactions between multiple parameters. Furthermore, the delivery of all available parameters within a single web page resulted in a cluttered interface, making it more difficult for users to understand the complex interactions between parameters. Providing fewer selections per page and more guidance simplified and improved the user experience and a new model was created, allowing the user to build a request over a series of pages. The resultant 'request builder' approach, frequently employed within internet commerce sites, allowed a request to be built from three pre-defined starting points by initially selecting a data set, variable or location (see Figure 2). This restricted the number of pathways through the interface and reduced the amount of client-side

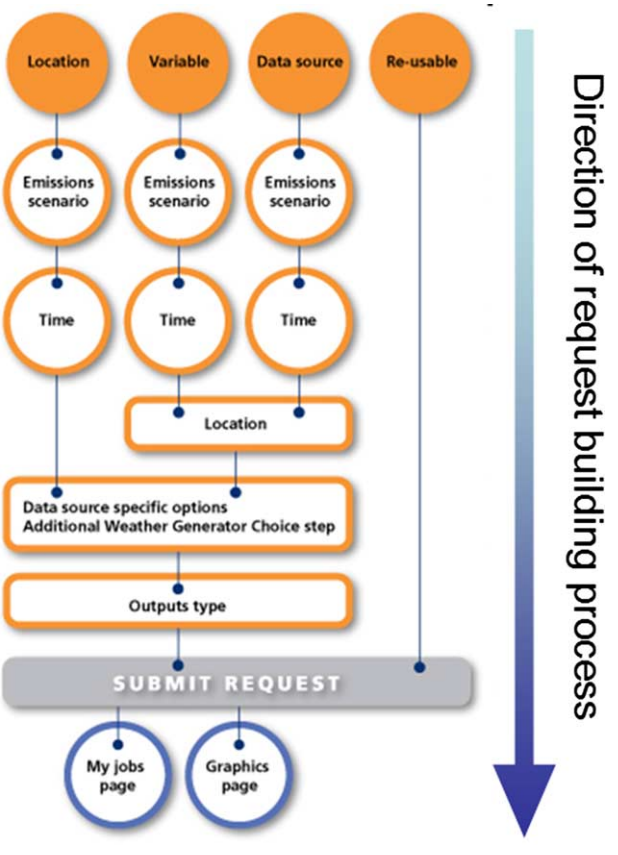

Figure 2. Schematic of the request builder process. © UK Climate Projections 2009.

scripting required to represent the relationships between parameters. The parameters themselves were stored in PHP classes with unique identifiers and delivered via a PHP module. This determined the next page within a pathway based on the previous selections and the starting point of the request. This model allowed most of the request and rule handling to be managed on the server rather than in the client (browser) code. The modified approach improved the speed and reliability of the UI, whilst also allowing the partial parameter selections to be saved during the process, giving users the opportunity to resume partially completed requests at a later date. Since many of the products served by the SOA layer were deterministic, it was vital that any two identical requests produced the same output for comparative purposes (Granell *et al*. 2010). The request builder provides a unique identifier, and re-submission URL, for each request, allowing it to be shared amongst users and referenced in reports. The UKCP09 User Interface Manual (2009) provides users with guidance on the technical process of creating a request and navigating through the interface, in order to gain access to a range of outputs.

The request builder process (as depicted in Figure 2) represents the typical workflow of a UI session where the user selects an initial parameter and subsequently narrows down the available options to select a final product. This might be a map, graph, data file (in CF-netCDF, CSV or Shapefile formats) or the outputs from a Weather Generator run. Whilst many of the pages are straightforward PHP/JavaScript/CSS rendering of HTML forms, there are three particularly important pages in terms of the user experience. These (the *Location* page, *Graphics* page and *Jobs* page) provide specific functionality detailed below.

2.3.1 Web-map client on the location page

The data available from UKCP09 is modelled at varying spatial resolutions and on different projections. Depending on the product, users can select from varying *location types*, ranging from a single grid box (5–25 km resolution) to an aggregated region (such as south-west England). Figure 3 shows a screenshot of the web-map client on the location page, alongside the functionality required to deliver this product. The OpenLayers JavaScript web-map client was chosen because it provided the appropriate utilities. Its open source nature also allowed extension and modification where necessary. The map client was customised to display multiple overlays and to perform complex selections such as the highlighting of multiple adjacent grid boxes.

2.3.2 Modifying and viewing outputs on the graphics page

A major drawback with the page-based request builder model was the problem of having to click through multiple pages to select parameters and generate output. This could become particularly frustrating when the user required very similar outputs differing by only one or two parameter selections. The graphics page was conceived as the solution to this problem as it allows limitless modification, viewing and saving of outputs without re-loading the page. Figure 4 shows the main functional elements of the graphics page. Once familiar with this page, a user can generate a request and download the outputs within a few clicks. Each new request is saved to the database and can be accessed subsequently on the jobs page (described below).

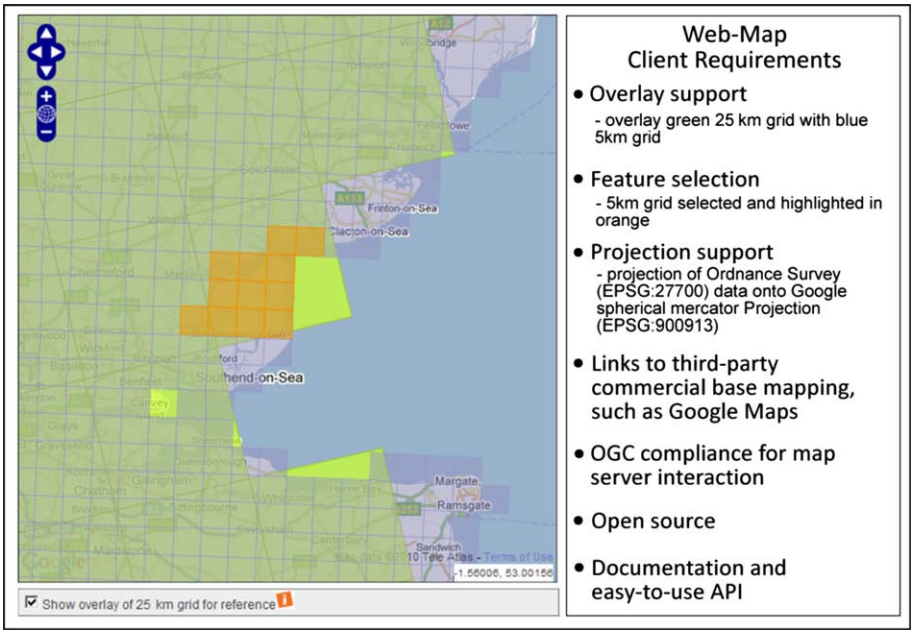

Figure 3. The web-map client on the location page and its functional requirements list.

When the user selects a 'Re-load plot…', 'Save Image As' or 'Save data As' action, the graphics page generates an AJAX request that calls services in the SOA layer. The response is either saved directly to the user's computer or, in the case of image outputs, can be previewed within the page before download. In the case of some mapped outputs, the user can zoom, pan and display values on the map-client within the graphics page. For many graphed outputs, the user can hover over the image to view the actual data values. The graphs, maps and underlying data can be downloaded in various formats, including options for publication-quality image outputs.

2.3.3 Providing access to current and previous requests via the jobs page

More complex requests, such as those for weather generator simulations, are processed asynchronously. The user is prompted to confirm a request submission based on the estimated processing time and volume of the outputs. Once the job has been submitted, the UI presents the user with the jobs page and continues to make AJAX calls to the SOA layer to provide the processing status of the offline job. When the job has completed, the outputs and metadata related to the request automatically appear in the user's list of previous jobs (on the jobs page) and the user is also notified by e-mail. The jobs page acts as a store of all requests submitted by an individual user, allowing outputs to be downloaded, requests to be re-run and post-processing of results using other UI utilities where applicable. This also provides a means to share results with other UKCP09 users and can be used as a mechanism for short-cutting the full request-building process.

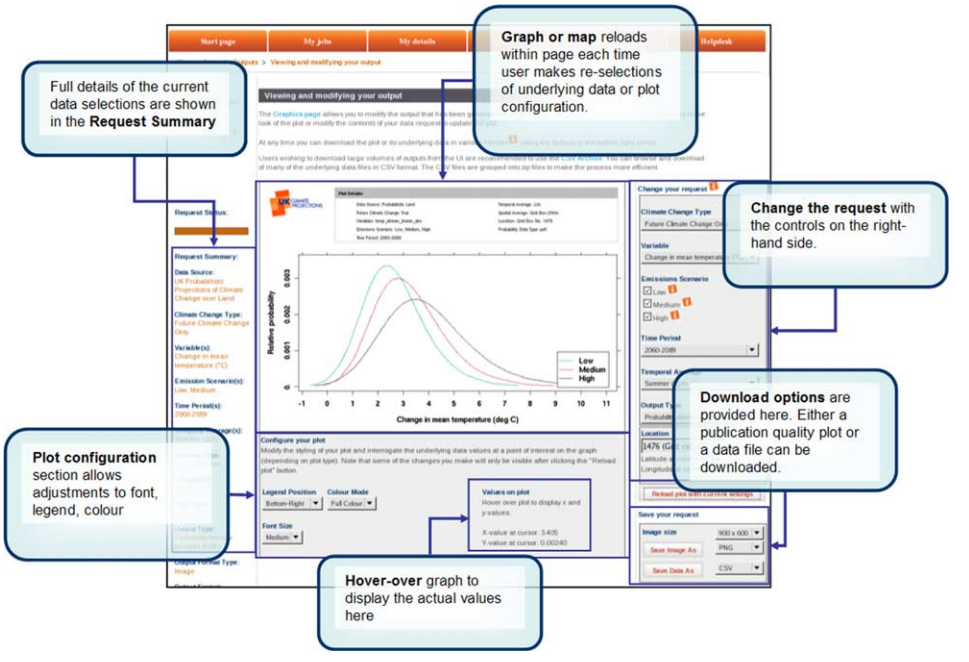

Figure 4. The graphics page of the UKCP09 UI, with the key functional elements highlighted.

2.4 Service-oriented architecture layer

The SOA layer runs the web services that generate all the maps, graphs and data products within the system. This layer operates as a set of Python applications using the Pylons framework running inside of the mod_wsgi Apache module. The SOA layer receives HTTP GET or POST requests for a product from the UI and either farms them out to one of its multi-processes, dispatching a response within 15 seconds, or scheduling the job with the offline processing layer.

The requirement for the SOA layer was to deliver a range of different products including map tiles, graphs and data files. The Open Geospatial Consortium (OGC) specialises in the development of standards for geospatial data storage and exchange. For generating tiles for the UI interactive map-client, the OGC Web Mapping Service (WMS 2006) was an obvious technology choice. WMS is an accepted and commonly used standard, providing interoperability via a well-described interface. The developers also had pre-existing in-house software components that could be re-factored to provide the UKCP09 WMS capability.

Catering for jobs that would run for different durations and produce a range of different outputs required a solution that managed both in-process execution of processes and asynchronous job management. When selecting a tool or framework to deliver this flexibility, the OGC Web Processing Service (WPS 2007) specification was considered a suitable candidate. Whilst still in its infancy, the WPS interface provides a means of wrapping up arbitrary processes (such as *GetAPlot* or *RunAModel*) in a common framework with the following advantages:

- Web service interface, using POST or GET
- Asynchronous reporting and control of jobs
- A defined XML interface for responses, including exceptions
- A common format for passing arguments to the server
- Job status interrogation

The available open source implementations of WPS were not well developed at this time. Since the project developers already maintained an open source toolkit containing a WMS implementation (CEDA OGC Web Services or COWS [Pascoe *et al.* 2010]), it made sense to add the WPS capability to this stack. The resulting tool, known as the COWS-WPS, includes the ability to:

- inform users when a job has completed (or failed).
- provide a configuration system that allows new processes to be added via a simple configuration file and a single Python interface module. This feature is available only to the WPS administrator in the current implementation and new processes are identified after the service is restarted. However, the 'plug-in' methodology could be extended to allow authorised users to add processes at runtime.
- connect to a scheduling tool that communicates with external offline processing nodes. The SOA layer and the offline processing layer share a file system which allows either layer to read/write status information to a single location.
- run a multiple-processor server application with 'quick' tasks within a worker pool of connected processes.
- make a 'costonly' (or 'dry-run') request before submitting a job that results in an estimate of the response size and duration without actually executing the job. This is implemented as an additional argument to the WPS Execute Request, making it available for use with all processes.
- report a job history for the entire system or the individual user.
- zip up groups of output files and report details of their contents in the XML response. This functionality enabled selecting and viewing of thumbnail plots within the UI graphics page.

A specific feature of the system that required a great deal of integration between the PHP/JavaScript UI and the Python WPS was the tracking of user jobs via the UI jobs page. The WPS specification provided a mechanism for presenting a persistent URL that returns the status of a given job. The UI employed AJAX technology to poll this 'status URL', parsing the XML and displaying a table of outputs that can be interrogated by the user. For accessing information about previous jobs, an additional process was deployed under the COWS-WPS that returns an XML-encoding of the metadata for previous jobs issued to the WPS by a given user. This extension of the WPS capability is likely to be an essential requirement of any implementation that includes asynchronous job management.

2.5 Offline processing layer

The offline processing layer was created in order to handle large data extractions and weather generator runs that might take hours to process and could potentially

require significant computing resources. It was considered that the majority of users would require access to graphical outputs, and therefore to reduce the impact on the UI and SOA layers providing these outputs a separate and controlled environment was developed for offline jobs. Also, controlling the number of concurrent jobs and queue management can be better handled by tools dedicated to that purpose. The processes running in this layer are managed by Sun Grid Engine (SGE), a scheduling tool configured to manage a *fast* and a *slow* queue to handle requests of different sizes from the WPS in the SOA layer. Whilst the offline processing layer limits access on a one job-per-user basis, the SOA layer is unrestricted, and so users can continue to generate image and small data products whilst a large job is running.

Each job is managed via a Python wrapper module that has the ability to report its status to a simple text file. Lengthy processes routinely update this file with information on the percentage of the task that has been completed. When the UI polls the WPS for information about a current job, the WPS uses the status URL to identify the job and reads the contents of the status file. The current status is then serialised into the Execute Response XML document (specified in the WPS standard) returned to the UI.

2.6 State management layer

Whilst the read-only components of the front-end, web services and the offline processing could be parallelised, the management of state needed to be treated differently. There are two main areas of state management, the individual UI session (stored in a database) and the longer-term cache of outputs requested by users (stored on hard disk). A user session, or user-driven process, has no specific binding to a given VM, and so a series of interactions can be dealt with by any node in the parallelised system. This required the database and cache disk to be accessible, and identical, on all nodes. The preferred solution was to provide a single instance of each state component connected to all VMs.

The three components of the system required to interact with all nodes in the system were as follows:

- **Load-balancer:** The HAProxy load-balancer tool distributes HTTP requests to the UI and SOA layers. All requests are received by HAProxy initially before forwarding to the appropriate VM in a round-robin configuration.
- **Database:** A PostgreSQL database managing the state and history of user information, sessions, requests and offline jobs.
- **Cache disk:** Request outputs are written to a cache disk which is NFS-mounted across VMs in the offline processing and SOA layers.

The danger of managing state in this manner was the introduction of single points of failure. This was mitigated by housing the load-balancer, database and cache disk on a pair of VMs in active/passive failover mode. The ‘state’ VM runs the live services whilst the ‘backup state’ VM routinely copies its state from the live version. Through the combined use of database dumps and Rsync mirroring of the cache disk, it was possible to continually synchronise the ‘backup state VM’ within 1 hour of the live

system. Subsequent testing showed it was possible to reinstate the backup system within 2 hours of a failure to the 'state' VM.

2.7 System overview

The overall system functions by a significant amount of interactions between the different components described in the previous sections, whether for a weather generator simulation run, or for access to one of the other data sources. The connections between the UI layer, the SOA layer and offline processing layer are highlighted in the following system interactions involved in an example of a weather generator request:

(1) User logs into UI and selects to run a weather generator job.
(2) User steps through the request builder pages configuring the request parameters.
(3) The UI contacts the WPS in the SOA layer to request the 'cost' of the job.
(4) The WPS returns an XML response that the UI parses to inform the user of the output size and job duration.
(5) User confirms submission of the job and is forwarded to the UI jobs page.
(6) The WPS submits the job by contacting the SGE 'submit host'.
(7) The WPS sets the job status as 'accepted'.
(8) The SGE 'submit host' decides an appropriate queue to submit the job to.
(9) The job is placed in the SGE queue.
(10) The job is run on one of the SGE 'execute hosts' in the offline processing layer.
(11) The job is run from the command-line on the 'execute host' and it routinely updates the status file with the percentage completion time.
(12) The UI keeps track of, and routinely polls, the status URL of the request to get the status of the job from the WPS layer.
(13) On completion of the job the offline processing layer updates the status of the job and mails the user.
(14) The UI polls the status URL and identifies that the job is finished at which point it becomes visible via the jobs page.

Figure 5 provides a view of the main outputs delivered by the UKCP09 user interface. The raw data encompasses weather generator simulations, large data subsets and small requests for the data underlying graphical outputs. Once users have navigated through the request builder, the final page always presents a range of available output types, filtered based on their previous selections.

3. Discussion

This section presents a discussion of the approaches described in Section 2. It begins with a review of the requirements, followed by comments on the use of standards, free and open source software (FOSS) and the provision of a robust system.

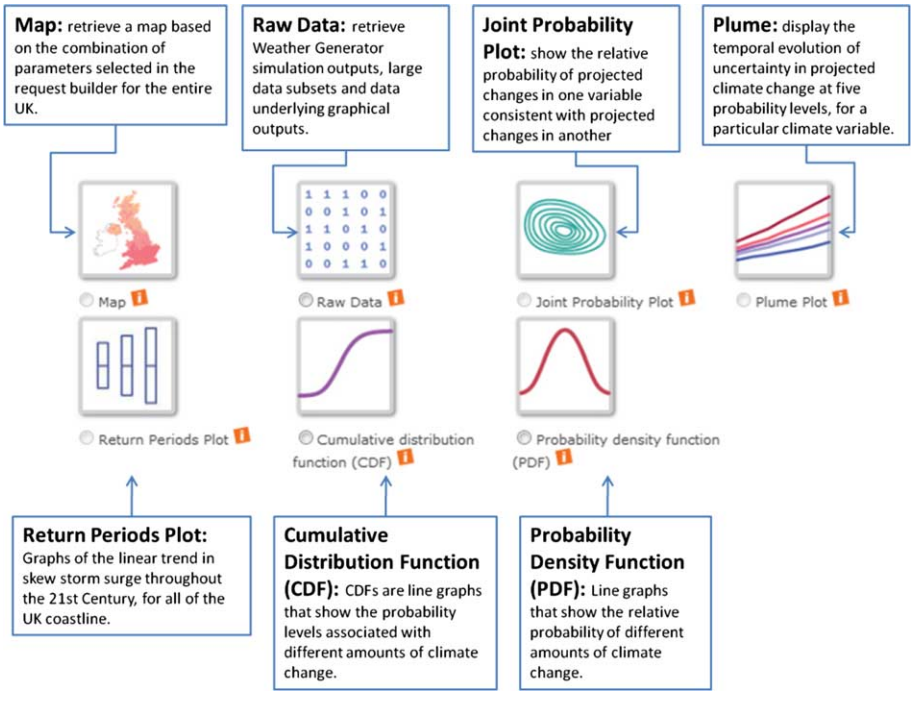

Figure 5. The main outputs delivered by the UKCP09 user interface.

3.1 Meeting the requirements

The process of identifying end-users for the UK climate projections highlighted the varied nature of those needing to consider climate change impacts in their decision making. A profile of the current users includes national and local governments and commercial, public and academic sectors working in fields ranging from nature conservation to construction. The requirements of each user group vary from generating a small set of temperature and rainfall maps to the ability to download almost the entire probabilistic data set to use as inputs for Climate Impacts Models.

The requirements for this project are still evolving as the scientists, stakeholders and end-users discover what questions need to be asked and how they can be answered via this novel interface. A significant challenge in writing the user guidance was attempting to provide both summary and detailed information about the underlying system and the science. The layout and delivery of the support documentation needed to be well organised and logically accessible to prevent users from being overwhelmed with information (Street *et al.* 2009). The complex relationships between the data sets and their parameters further complicated the data delivery process as various routes through the system involved navigation through multiple pages. However, the 'request builder' approach was preferred because it constrained the amount of guidance required per page and allowed the information to be related to particular parameters displayed on each page, rather than displaying all at once.

In response to the six main requirements in Section 2.1, it is clear that the parallelised and layered architecture was constructed to ensure scalability, robustness

and high availability (requirements 2, 4 and 6). The design was compartmentalised into various functional elements to ensure a clear separation of concerns that would guarantee each component an appropriate allocation of computing power. The SOA layer and the UI layer provided a set of web-accessible services (requirement 1) that could interact with the offline processing layer (requirement 3) for handling large requests. Issues of security and user access (requirement 5) are discussed later in this section.

In generating the requirements for the UI part of the system, the project undertook a review of other websites delivering climate information and products. This highlighted many desirable and undesirable aspects about a range of sites. The key aspects of the review are summarised in the UI requirements listed in Section 2.1 and are subsequently reviewed here.

The diverse user community required easy access to help and guidance focussed both around the UI and the underlying science behind the available data products (requirement 7). UKCIP provided the majority of the scientific guidance about UKCP09 and identified the importance of providing information in uncomplicated language in a hierarchical fashion. This enabled both expert and non-expert users (requirement 10) to dig down into the support information at their convenience. Furthermore, information icons (i) facilitated access to both the UI Manual and scientific guidance as hover-overs. The data itself were mainly delivered through the graphics page, offering users the ability to customise and download publication-quality graph and map outputs within a single web page (requirement 8). As the user generated these requests within the request builder process, each selection was subsequently saved after each page, enabling users to return to a partially complete request at their convenience without having to restart (requirement 9).

3.2 Use of standards and free and open source technologies

Web standards help to define rules for storing, integrating and using data, services and source code. The World Wide Web Consortium (W3C) drives the development and use of standards within the web community. The successful delivery of this project can be partially attributed to the extensive use of standards to aid interoperability between the chosen components. The OGC Web Mapping Service and Web Processing Service have been employed within the system to disseminate and remotely process geospatial data via the web. Furthermore, a common communication layer between multiple web applications was provided via the Python Web Server Gateway Interface (WSGI). The underlying six-dimensional geospatial climate data set was encoded using the *de-facto* NetCDF standard, a platform-neutral storage format which is supported by a range of free libraries and tools. The use of these standards allowed the development of a distributed architecture from a range of independent components.

The definition and use of standards is key in the quest for interoperability. However, many standards are still in flux, and so most projects select those that they consider important. Within this project, standards were adhered to where possible but in some cases were modified in order to meet the required functionality. A particular example was the OGC WPS specification. Still at version 0.4.0 when the project began, the WPS 1.0 specification was approved in February 2008 but it is still relatively immature (Michaelis and Ames 2009). WPS provided a very useful

generic container for a host of 'processes' that might have otherwise been deployed as bespoke web services. The implementation within the COWS-WPS extended the specification in terms of adding the new arguments to the execute request signature ('costonly' and 'inform') as well as a process for accessing previous job details. WPS 1.0 introduced the concept of application profiles as a means of providing and publishing domain-specific processes to aid interoperability in terms of building clients and utilising the publish/find/bind paradigm (Lanig and Zipf 2010). In the case of this project, the outputs were highly tailored, and so in most cases the processes were unlikely to be generally applicable to other climate-related data sets. However, the development of application profiles for more general climate-plotting processes would be beneficial as there are common requirements for plotting tools outside the two-dimensional view covered by WMS. The tools developed within the COWS stack are now being taken on by other projects both in-house and externally.

The underlying climate projections were over half a terabyte in size and for this reason a binary file format was considered appropriate for managing the data archive. The CF-NetCDF format was chosen because of its widespread adoption demonstrated by its usage by the international research community for exchange of climate simulations in the third Climate Model Intercomparison Project (CMIP3 [Meehl *et al*. 2007]). An alternative approach would have been to use the emerging Sensor Web standards such as the Sensor Observation Service (2007) or a profile of the OGC Observations and Measurements (2007) standard. The CF-NetCDF option was considered more appropriate because the tooling and experience already existed in the scientific user community.

FOSS can describe code that can be viewed, modified and redistributed by developers without imposing the same property right restrictions that are synonymous with proprietary software (Waring and Maddocks 2005). The advantages and disadvantages of using FOSS are well documented and summarised in Waring and Maddocks (2005), Cerri and Fuggetta (2007) and Kemp (2009). The combination of using open source and standards-compliance enabled heterogeneous software components to be integrated into one system within this project. The FOSS tools utilised were generally platform-agnostic, enabling client and server-side components to be developed and tested without being tied to a particular platform or environment. The full technology stack is presented in Table 1, illustrating a range of open source technologies deployed within this project. The integration of FOSS and in-house software facilitated delivery of a scalable, flexible and robust system.

All the technologies employed have stood up well within the production system. Some deserve a particular mention because of the impact they had on the development cycle and the deployment process:

- JQuery: provides robust cross-browser support for a range of useful JavaScript functions. It also works side-by-side with other JavaScript libraries such as OpenLayers.
- The Web geo-stack of OpenLayers, TileCache, GeoServer and PostGIS: provides an incredibly powerful toolkit for complex map applications employing OGC standards where appropriate.
- Pylons: provides a Python web development environment that rivals Ruby on Rails. It allows rapid deployment of Python-based websites and web services

Table 1. The technology stack and standards used within this project.

Type of tools and/or technologies	Tool technology or language*	Relevant standards
Load-balancing and caching	HAProxy, TileCache	HTTP
Web-interface	Apache, PHP, JavaScript†, JQuery, OpenLayers, Zend Lucene, Proj4JS	XML, HTML, ECMAScript, CSS, WMS, WMS-tiling
Database	PostgreSQL, PostGIS	SQL, OGC Simple Features for SQL
Web services, data manipulation and visualisation code	Apache, mod_wsgi Apache module, GeoServer, Tomcat, Pylons, SQLAlchemy, cows-wps‡, cows-wms‡, Python, R, MatPlotLib, CDMS, NumPy, Shapelib	WPS, WMS, WSGI, XML
Scheduling and processing	Sun Grid Engine†, Fortran (F/V), Java†, Python	
Testing and monitoring	Nmon, JMeter, System check‡, Pylot, Python Nose, Python PyUnit, Apache Benchmark, NetBeans, Firebug, Selenium, IETester, MultipleIEs†, OpenSSH	
Management tools	Trac, Subversion	
System and deployment tools	Xen, VM image replication tools‡, Rsync, Python, Buildout, Python easy-install	
Operating system	OpenSUSE 10.3	

*Most tools, technologies and languages are considered free (by which we mean 'licenced for usage without a fee') and open source, except those indicated below.
†Not open source but licensed for usage without a fee.
‡Tools developed within the project – made available on an open source licence.

using the WSGI standard. It also plugs and plays with Apache and other standard tools.

- Subversion: provides an invaluable means for the team, distributed across three sites, to work on a common code base. With clients for various platforms each developer used its own IDE of choice and was able to deploy local versions of services for testing.

3.3 Performance and reliability

Since the launch in June 2009 there has been extensive use of the UKCP09 system. During this period, there has been very little downtime ($<0.5\%$), most of which has been pre-scheduled. The parallelised architecture has stood up very well to both high-demand and usage over time, allowing rapid bug-fixing, VM management and maintenance without disruption to the user experience. A significant success during the launch period was the introduction of an additional 12 VMs at very short notice to handle high demand. This involved the temporary acquisition of

six physical servers from another project, installation of the VM hosting software, deployment of VMs and re-configuration of the load-balancer, all of which was achieved in less than a week. Other factors that have significantly contributed to the robustness of the system are discussed below.

3.3.1 System protection

All web-accessible computing resources require some level of protection against malicious attacks as well as high demand that can overload and compromise the system. Protection against malicious attacks requires rigorous protection of servers and configuration of firewalls. Locking down access to servers and databases by IP address is a typical approach to this problem.

In order to protect against very high demand, the following measures were put in place:

- Over 500,000 requests were pre-run and cached within the system. These were typically requests for graphs which would take from 2 to 20 seconds to produce. Pre-caching the most likely requests reduced the load on the servers and allowed more concurrent sessions to be handled.
- The parallelised system was another means of reducing the impact of high demand by spreading the load across multiple VMs.
- A limit (1000) was set on the number of concurrent sessions allowed. Any attempts to login over that number would be politely asked to try again later.

3.3.2 Testing, benchmarking and monitoring

As the project neared completion, the use of testing and benchmarking techniques became critical to highlighting design issues, bugs in the system and performance factors. Automated testing tools such as Python Nose and Selenium were employed to rapidly test front- and back-end components. Two particular aspects of testing that paid dividends in delivering the final system are discussed briefly here. During early deployment, integration tests of the WPS plotting processes demonstrated that a third-party library was not thread-safe. This was a silent failure in that the processes all ran to completion without reporting any errors. It was only upon inspection of the image outputs that the problem was visible, with plot axes and annotation being randomly ‘shared’ between different outputs generated by concurrent WPS processes. This required modification of the core WPS code to move from a multi-threaded application server to a multi-process model using a worker pool. The WSGI environment and application stack allowed this change to be ‘plugged-in’ at the web server layer with minimal change to the code base. The second very significant testing scenario involved a third-party contractor employed to assess the overall system performance under varying loads. This performance testing used the FOSS tool JMeter to deploy very heavy loading on both the UI and SOA layers of the system. This test was used to assess the capacity of the parallelised services and

provided stakeholders with assurance that the system would stand up to intensive usage during the launch period.

Another key factor in protecting the system was constant monitoring. A set of system-checking tools was developed to handle monitoring of disk space, access to servers, access to cross-mounted disks, responses from known URLs, response times and database access. In addition, access from outside the system was monitored to highlight any issues that might be firewall related. Furthermore, all warnings and errors recorded by the system-checker, and by the UI code itself, resulted in an e-mail and/or SMS message to the administrators. This set of checks provided, and still provides, an almost instantaneous indication of the state of the entire system.

4. Conclusion

This project demonstrates how a heterogeneous web application can be built with different technologies and languages using common standards within a service-oriented architecture. The UKCP09 project involved a large number of stakeholders directing the course of a novel user interface to deliver cutting-edge climate science to both sophisticated and non-expert users. The resulting UI fronts a multi-layered parallelised architecture built on a range of open source tools and technologies, adhering to standards where appropriate. The technology stack employed within the project consists of many exemplary tools. The use of such FOSS alternatives to proprietary software is likely to increase with time in publicly-funded IT projects as public sector managers are encouraged to avoid long-term contracts that can lead to vendor tie-in.

A range of exceptional technologies now exist in the free and open source communities that, combined with the power of modern browsers and high-speed internet, provide a basis for delivering very sophisticated tools for presenting scientific information. As internet technologies and the science of climate prediction continue to evolve, there is every reason to be excited about the interactive tools that will be used to deliver future climate projections. The ability to disseminate this type of scientific data across the internet, through the combination of open source technologies, can help a wider range of decision-makers understand the potential impacts of climate change in the UK, and recognise the consequences this may have on the environment and future economic development.

Acknowledgements

We acknowledge all members of the UKCP09 Project Management Group, and the following individuals for their help and efforts within the project: Geoff Jenkins, David Sexton, Vassilis Glenis, Chris Kilsby, Phil Jones, Colin Harpham and Peter Norton.

Notes on contributors

Mr Ag Stephens is the Collaborative Projects Manager at the British Atmospheric Data Centre and its parent organisation CEDA. His work involves overseeing collaborations with the UK Met Office and other research groups funded by the Natural Environment Research Council and UK Government departments. Ag led the development of the UK Climate

Projections (UKCP09) User Interface and data services. He is currently developing the Python-based COWS Web Processing Service as a general service container and is keen to explore the wider role of WPS in the OGC Web Services landscape.

Mr Philip James is a Senior lecturer in GIS at Newcastle University. He graduated from Newcastle University in 1987 with a degree in Japanese and worked at Nissan for a number of years before returning to the University as a systems programmer in the Department of Surveying. He primarily works with colleagues in climate and transport within the School of Civil Engineering and Geosciences integrating data through information technology and geospatial technologies and standards.

Mr David Alderson graduated with a first class Honours degree in Geographic Information Science from Newcastle University, in 2005. He was subsequently appointed Research Associate in GeoInformatics at Newcastle University. He has worked on many projects, including helping to develop the components of the UK Climate Projections '09 (UKCP09) user. His research has focussed around the management, delivery and dissemination of geospatial data, specifically focussing on the use of open source products and tools. He is currently employed as a named researcher on the UK Infrastructures Transitions Research Consortium (ITRC), funded by the Engineering and Physical Sciences Research Council (EPSRC).

Dr Stephen Pascoe is a software engineer specialising in data access systems for the Earth System Sciences. He has a PhD in Atmospheric Chemistry from the University of Leeds and has worked at the British Atmospheric Data Centre for the past 7 years. Stephen is a principle developer of the CEDA OGC Web Services Framework (COWS) and develops operational websites on top of COWS' WMS, WCS and WPS support. He is active in various international collaborations designing technology standards and implementations including being BADC's technical lead in the IS-ENES EU Framework-7 Project and member of the Earth System Grid Federation technical committee.

Mr Simon Abele is a Research Associate in Geoinformatics/GIS at the School of Civil Engineering & Geosciences (CEG), Newcastle University, UK. He previously helped to build the UKCP09 User Interface (UKCP09-UI) and was primarily responsible for implementing geospatial web services to support geospatial data storage, processing and delivery. His research interests include; Open Source geospatial standards, real-time geospatial data access, processing, storage and analysis, geospatial metadata, search and discovery, vernacular geography, geospatial cloud computing, and their application in addressing the problems encountered in engineering and geosciences. He holds a degree in Geographic Information Science (GIS) from Newcastle University.

Mr Alan Iwi is a research scientist at the British Atmospheric Data Centre. Since completing his D.Phil. in atmospheric physics at Oxford University he has worked on a number of projects involving climate models. Alan is currently a named researcher on the VALOR project, funded by the Natural Environment Research Council.

Mr Peter Chiu is the Systems Support Manager for the British Atmospheric Data Centre. He is the administrator of a network of over 200 servers running a range of Linux-based operating systems as well as a data archive of around a petabyte. Peter has significant experience in deploying virtual machine environments for robust scalable systems.

References

CEDA OGC Web Services, 2010. Available from: http://cows.badc.rl.ac.uk/cows_wps.html [Accessed 10 December 2010].

Cerri, D. and Fuggetta., A., 2007. Open standards, open formats, and open source. *Journal of Systems and Software*, 80 (11), 1930–1937.
FINESSI Web Tool, 2007. Available from: http://www.finessi.info/finessi/ [Accessed 10 December 2010].
Freedom of Information Act, 2000. Available from: http://www.opsi.gov.uk/acts/acts2000/ukpga_20000036_en_1 [Accessed 10 December 2010].
Goodess, C.M., *et al*., 2007. Climate scenarios and decision making under uncertainty. *Built Environment*, 33 (1), 10–30.
Gordon, C., *et al*., 2000. The simulation of SST, sea ice extents and ocean heat transports in a version of the Hadley Centre coupled model without flux adjustments. *Climate Dynamics*, 16, 147–168.
Granell, C., Díaz, L., and Gould, M., 2010. Service-oriented applications for environmental models: reusable geospatial services. *Environmental Modelling and Software*, 25 (2), 182–198.
Hewitt, C.D., Goodess, C.M., and Betts, R.A., 2009. Towards probabilistic projections of climate change. Proceedings of the Institution of Civil Engineers. *Municipal Engineer*, 162 (1), 33–40.
Jones, P., *et al.*, 2009. *Projections of future daily climate for the UK from the Weather Generator*. Newcastle-Upon-Tyne, UK: Newcastle University.
Kemp, R., 2009. Current developments in open source software. *Computer Law and Security Review*, 25 (6), 569–582.
Lanig, S. and Zipf, A., 2010. Proposal for a Web Processing Services (WPS) application profile for 3D processing analysis [online]. *In: Second international conference on advanced geographic information systems, applications, and services, Geoprocessing 2010*, 10–16 February 2010, Netherlands, Antilles. Available from: http://www.computer.org/portal/web/csdl/doi/10.1109/GEOProcessing.2010.25 [Accessed 4 March 2011].
Meehl, G.A., *et al*., 2007. The WCRP CMIP3 multi-model data set: a new era in climate change research. *Bulletin of the American Meteorological Society*, 88 (9), 1383–1394.
Michaelis, C.D. and Ames, D.P., 2009. Evaluation and implementation of the OGC Web Processing Service for use in client-side GIS. *GeoInformatica*, 13 (1), 109–120.
Murphy, J., *et al*., 2009. *UK climate projections science report: climate change projections*. Exeter: Met Office Hadley Centre.
OpenLayers, 2010. Available from: http://openlayers.org/ [Accessed 10 December 2010].
Pascoe, S., Stephens, A., and Lowe, D., 2010. Pragmatic service development and customization with the CEDA OGC Web Services framework [online]. *In: European Geosciences Union General Assembly*, 2–7 May 2010, Vienna. Available from: http://cedadocs.badc.rl.ac.uk/773 [Accessed 10 December 2010].
Python Web Server Gateway Interface (WSGI), 2010. Available from: http://www.wsgi.org/wsgi [Accessed 10 December 2011].
Serrano, N. and Aroztegi, J.P., 2007. Ajax Frameworks for Interactive Web Apps. *IEEE Internet Computing*, 24 (5), 12–14.
Solomon, S., *et al*., 2007. *Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. UK: Cambridge University Press.
Street, R.B., Steynor, A., Bowyer, P., and Humphrey, K., 2009. Delivering and using the UK climate projections. *Weather*, 64 (9), 227–231.
The Open Geospatial Consortium (OGC), 2010. Available from: http://www.opengeospatial.org/ [Accessed 10 December 2010].
The Open Geospatial Consortium (OGC) Observations and Measurements, 2007. Available from: http://www.opengeospatial.org/standards/om [Accessed 4 March 2011].
The Open Geospatial Consortium (OGC) Sensor Observation Service, 2007. Available from: http://www.opengeospatial.org/standards/sos [Accessed 4 March 2011].
The Open Geospatial Consortium (OGC) Web Mapping Service (WMS), 2006. Available from: http://www.opengeospatial.org/standards/wms [Accessed 10 December 2010].
The Open Geospatial Consortium (OGC) Web Processing Service, 2007. Available from: http://www.opengeospatial.org/standards/wps [Accessed 10 December 2010].
The World Wide Web Consortium, 2010. Available from: http://www.w3.org/ [Accessed 10 December 2010].

UK Climate Change Risk Assessment, 2010. Available from: http://www.defra.gov.uk/environment/climate/adaptation/ccra/index.htm#assessment [Accessed 4 March 2011].

UKCP09 UI manual – User interface manual: 1 Introduction, 2009. Available from: http://ukcp09.defra.gov.uk/content/view/1145/537/ [Accessed 10 December 2010].

Waring, T. and Maddocks, P., 2005. Open source software implementation in the UK public sector: evidence from the field and implications for the future. *International Journal of Information Management*, 25 (5), 411–428.

Xen, 2010. Available from: http://www.xen.org [Accessed 10 December 2010].