The CARMEN Software as a Service Infrastructure

Michael Weeks, Mark Jessop, Martyn Fletcher, Victoria Hodge, Tom Jackson, and Jim Austin.
Advanced Computer Architectures Group, Dept. of Computer Science,
University of York, York. YO10 5DD. UK.
Email: {michael.weeks, mark.jessop, martyn.fletcher, victoria.hodge, tom.jackson,
jim.austin}@york.ac.uk

Summary

The CARMEN platform allows neuroscientists to share data, metadata, services, and workflows and to execute these services and workflows remotely via a web portal. This paper describes how we implemented a service-based infrastructure into the CARMEN Virtual Laboratory. A *Software as a Service* framework was developed to allow generic new and legacy code to be deployed as services on a heterogeneous execution framework. Users can submit analysis code typically written in Matlab, Python, C/C++ and R as non-interactive standalone command line applications and wrap them as services in a form suitable for deployment on the platform. The CARMEN Service Builder tool enables neuroscientists to quickly wrap their analysis software for deployment to the CARMEN platform, as a service without knowledge of the service framework or the CARMEN system. A metadata schema describes each service in terms of both system and user requirements. The search functionality allows services to be quickly discovered from the many services available. Within the platform, services may be combined into more complicated analyses using the workflow tool. CARMEN and the service infrastructure are targeted towards the neuroscience community; however it is a generic platform, and can be targeted towards any discipline.

Keywords

Software as a Service, Metadata, Collaboration, Virtual Laboratory, Neuroscience, CARMEN.

1. Introduction

In all areas of science, sharing the results of research is fundamental. Much of modern research generates large amounts of complex, highly valuable data, and develops software based methods to analyse data. Sharing these results is important but can be difficult. To be of value the data needs to be carefully recorded with the information on how it was generated and the software needs to run as intended by the developer. The CARMEN project [1] was initiated to allow such research assets to be shared easily between researchers with the ability to run the software against the stored data. The work focussed on the neuroscience community which routinely generates large amounts of data and many analysis methods. The result of the work is a system that allows neuroscientists to share data, metadata, services, and workflows and to execute these services and workflows remotely via a web portal.

During the CARMEN project, we identified three key user types within the neuroscience community; experimenters who generate data, programmers who create analysis tools, and in-silico experimenters who will exploit the data and tools. The initial phase of the CARMEN project concentrated on providing a web-based portal facility to allow data and associated metadata to be uploaded to an online data repository. Users are able to choose to share their own data and/or metadata with collaborators, or the whole CARMEN community if required. The next stage of the research was to support exploitation of the analysis code as CARMEN Services. This paper describes work toward this goal.

The user group responsible for developing the analysis tools are typically neuroscientists with domain-specific application development skills. They have a collection of legacy analysis tools that they wish to be deployed on our system. The data the services will be processing are held in the CARMEN system, and is typically orders of magnitude greater than the size of the analysis code. Rather than bring the data to the analysis tools, it is more efficient to execute the analyses local to the data. This leads to a requirement for a *Software as a Service* [2] model that can utilise generic code. The analysis software creators must be able to create their own CARMEN services from their code, without knowledge of the underlying CARMEN *Software as a Service* technology. The services must be able to be targeted at specific software development toolsets used by neuroscientists (Java, Matlab, Python, etc) and at a range of OS platforms, and be deployed and executed across a distributed cluster of powerful server nodes. Once a service has been deployed on CARMEN, it has security settings applied so that the owner of the service can allow only valid users to view and/or execute it.

Users interact with the CARMEN system through a web portal (https://portal.carmen.org.uk). To the user, running services should be simple and service implementation agnostic. A workflow tool, that we are currently integrating into the CARMEN platform, allows multiple services to be connected together to allow more complex analytical tools to be created and executed. Neurophysiology data utilises many file formats, which can provide interoperability problems when sharing and combining data and services. The NDF (Neurophysiology Data translation Format) was developed within the CARMEN project to provide a solution to this problem [3], and so the services framework must support NDF.

This paper describes the CARMEN services infrastructure that was implemented to meet these requirements.

2. The CARMEN System Architecture

The CARMEN system is a standard three tier web architecture as described by [4] and depicted in figure 1. The Data Storage Tier is shared between SQL type databases and the SRB (Storage Resource Broker [5]) storage system. The database is used to store such things as user accounts, metadata [6], system states and the relationships between artefacts and users. Physical files, such as user data, services, workflows and their outputs are stored within the SRB system.

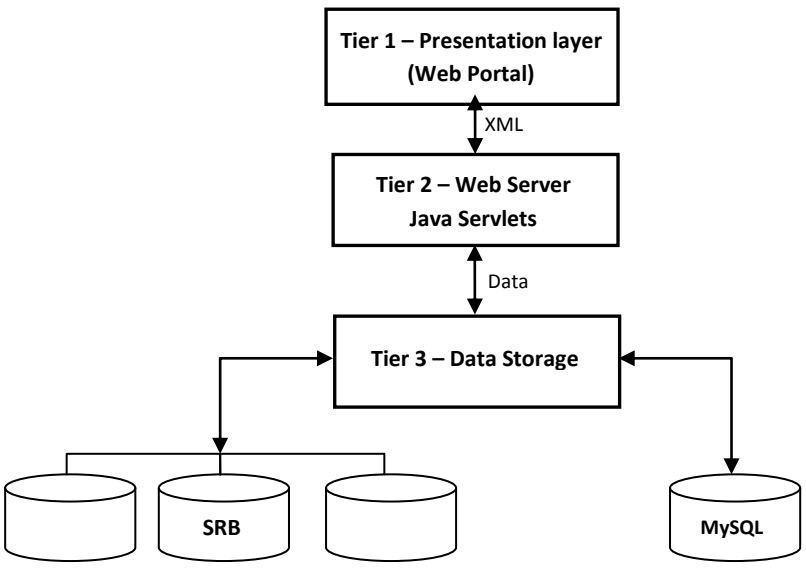

Figure 1 - CARMEN Architecture

The Application Layer (Tier 2) consists of a group of Java Servlets, as defined by Oracle [7]. These provide an API onto the Data Storage Tier which is currently exposed through a REST [8] type interface. The API, split into user and administration sections, generates XML descriptions of user interfaces (GUIs) and system states. Client applications can use these descriptions to build GUI interfaces and present information to users.

The web portal is one instantiation of a Presentation Tier, written in Java using the Google Web Toolkit (GWT) [9] (converted to JavaScript for deployment) and using AJAX (Asynchronous JavaScript And XML) [10] techniques. This allows rapid prototyping that leads to the generation of modern interfaces. These interfaces have many of the elements that make the CARMEN system look and feel more like a desktop environment, rather than a web portal. GWT also provides a development paradigm that is more akin to traditional application development, rather than that of the web. The most powerful feature of AJAX is the ability to perform asynchronous communication between browser and the Application Tier, without the need to refresh the user interface.

The CARMEN architecture is a heterogeneous environment with respect to both the user's chosen setup and that of the hosting environment. We are currently migrating the deployment of the CARMEN system to Cloud infrastructure which is supported by the architecture.

3. Services Specification

A survey of CARMEN neuroscientists showed that the typical neuroscientist has domain-specific development skills in Matlab, Python, C/C++ and R, and often a collection of legacy analysis code tools. They tend not to have knowledge of service technology or the necessary programming skills to achieve them. This leads to a requirement for a service framework that can utilise code that has been developed without knowledge of the CARMEN system or its implementation. In addition to the development toolset, services must also be able to be targeted at a range of OS platforms. To this end the following process was developed;

1. Services are written as non-interactive standalone command line applications. The command-line approach provides a common interface to the CARMEN system, irrespective of the implementation technology. These execute on a pool of execution nodes where data is staged in order to be used locally (in the current working directory) by the service. Inputs to the service are passed in as command-line parameters. For output data from the service, the application must print the values or filenames to the default output stream as XML formatted text to allow the system to read the values or filenames and to manage service output data;
2. A wrapping framework was needed to encapsulate the command-line application as a service so that it can be deployed on the platform. In the process it sets up environment options needed for the application to run. Java was selected as it provides an interface to the CARMEN system, provides multi-platform compatibility, and allows command line applications to be executed;
3. A service metadata schema was developed to describe the application in terms of both system and user requirement;
4. A wrapping tool to automate the creation of services; The CARMEN Service Builder.

Item 1 is easily achievable, as the CARMEN user will have already written complex domain specific analysis code and the changes necessary are minor. Item 3 requires metadata to be gathered manually from the user, whereas for item 2 the framework can use the service metadata schema to generate a wrapper environment. The CARMEN Service Builder tool was developed to simplify items 2 and 3.

The builder is a Java-based GUI application that was developed to provide simple form-based collection of service metadata, attachment of application executables, scripts, libraries, etc., and automatic generation of the service wrapping. The resultant service is registered with the CARMEN system via an administration panel on the portal.

In figure 2 we provide an example of Matlab code that is compliant with the CARMEN requirements in item 1. It has been modified so that it can be run as a command line application once compiled into a standalone executable. The program generates a random array, plots the data in a simple 2D graph and saves the graph plot to disk as a JPEG file. The program takes two parameters; a number which specifies how many data points should be generated, and a filename for the resultant graph file. There is one output, a JPEG file of the plotted data. Note the use of the XML-like "<output> </output>" tags to specify the output value or files. This is captured by the wrapping code and used to inform the CARMEN system of files or values that need to be handled as output data from the service. If multiple outputs are generated they should be inserted into the output tag string as a comma-separated list.

```
function test(num, outfile)
% generate a random vector and plot it
figure ; plot(randn(1,str2num(num)));
% save figure to file as specified filename
print ('-djpeg',outfile); close all force
% display the name of the file
disp(['<output>' outfile '</output>']);
end
```

Figure 2 - Sample Matlab service code

4. Service Wrapping Framework

Item 2 of our specification is a wrapping framework. Our initial service wrapping method utilised Web Service Technology. We evaluated several technologies, and selected InstantSOAP [11]. InstantSOAP, a collaboration project within CARMEN, allows the wrapping of command-line applications into web services. However, during the evaluation we found there were run-time errors when executing wrapped Matlab executables that utilised Matlab toolboxes, and there were concerns of on-going support. As a result, this approach was discounted.

We then opted for Metro JAX-WS [12], a Java-based technology for the creation of web services and their clients. To interface an application command-line to the JAX-WS web service wrapper, a few lines of Java interface code are required. As users may not be familiar with this, the Service Builder tool automatically generates the code, using the service metadata as a template. In this way, command line applications can be quickly wrapped inside of a web service WAR (Web ARchive) files. The resultant WAR files have a wrapping overhead of ~20MByte per service. The WAR file can be dynamically deployed into an Apache Tomcat Web Service/Servlet container [13] on an execution node, though it can take approximately 10-20 seconds before it is available for use. Unfortunately, we found that we could only run a limited number of services reliably within a tomcat container, especially when deploying and undeploying (removing) services.

Making the wrapping process as simple as possible was key. To do this the service developer is asked to specify (in a form) the input and outputs used by their application. This information is held in metadata with the application. Both JAX-WS and InstantSOAP provide client-side code that can be

configured on the fly for any particular service application and combination of inputs and outputs, as specified in the metadata. In addition, by including the wrapping method in the metadata means that we can seamlessly support multiple clients for the different service wrapping technologies. On execution, the system can examine the wrapping technology metadata and configure the correct client accordingly.

Using the JAX-WS Web Service approach, it was clear that the 20MB code overhead for each application was quite excessive. This overhead slowed loading times and limited resources such as the Apache Tomcat Servers. As our dynamic service deployment tool matured, the client and service resided on the same machine, in effect making the use of web services unnecessary and very inefficient. We removed the JAX-WS component, and adapted the Java wrapper into a dynamically loadable class. The overhead of the Java wrapping code and the resultant JAR (Java ARchive) file is only a few kilobytes, and services can be executed within a second of being deployed. It also removed the need to deploy services inside a Tomcat container. Instead, services are placed on disk, making un-deployment (i.e. deletion) of completed services unnecessary.

One of the disadvantages of the above approach is the possibility of executing malicious or unstable software on our execution nodes. We currently only allow known and trusted CARMEN users to create services. Also, the executing service is run with user-level privileges, limiting the damage it can potentially cause to the execution nodes, and the system at large. However, this will not scale as users increase. At the time of writing this paper we are moving to a private cloud hosting environment, utilising Virtual Machine (VM) technology for the execution nodes. This will enable us to create a pool sandboxed VMs to create a quarantine area. The quarantine VMs will be run new or untrusted services, and destroyed after each execution. Once a service has executed successfully a number of times, the service can be moved out of quarantine into the main pool of execution nodes.

5. Service Metadata

Item 3 of our specification is a Service Metadata schema. One of the requirements for CARMEN is to create a metadata schema for CARMEN services. The resultant MIAOWS (Minimum Information About the Operation of a Web Service) [14] document was similar to that specified in the WSDL Specification [15]. However, these were both based on an existing service at a defined endpoint location. With our need for dynamic portal interfaces, dynamic deployment of services onto a heterogeneous collection of servers, generic service clients, wrapper interfaces, and software configuration requirements, we needed to extend the metadata to be rich enough to provide the additional description capability.

It was noted that the following properties of the system would influence the metadata and drive its design;

- The requirement to hide the technology of services from the user (via the CARMEN portal), and service developers (via the Service Builder);
- Enable a user search facility to enable discovery of relevant services;
- Provide textual information to the user on the service, its usage, and its input and output parameters;
- Provide a portal interface to the user, either as a service execution form or as a workflow component;

- Select the correct client interface for the service's wrapping technology, and configure it so that the correct input data are passed to the underlying application, and outputs handled accordingly;
- Enable scheduling of services across the network of disparate execution nodes;
- Allow a service to be dynamically targeted towards specific platforms, and configure software applications as needed by a service.

To support these, the service metadata schema consists of three sections. The first section contains general information on the service, i.e.

- A name and description for the service. These are displayed to the user in the portal view;
- A keywords field to enhance discovery via search;
- Fields to specify the applications target platform type and its software configuration, if any;
- The owner of the service and sharing permissions.

The second section describes the input parameters of the application. Each input has:

- A description field that is displayed to the user in the portal;
- An input type (i.e. a variable value, a file or an NDF file);
- Filter options, such as specifying file extensions and providing some input validation;
- Whether default values are to be set;
- If the input to the service is a set of known values, a drop-down selector box can be displayed to the user.

The final metadata section describes each of the output parameters. This is restricted to a text description and an output parameter type.

6. NDF Services

With respect to CARMEN Services, NDF (Neurophysiology Data translation Format) allows a standard method for sharing data between services, especially in workflow scenarios. The portal has introduced NDF to facilitate effective service interoperability and data sharing. In workflows it allows services to interoperate effectively, by using a common format.

An NDF dataset consists of a configuration (XML) file which manages a variable number of host data files. The configuration file contains metadata and references to the associated host data files. NDF specifies a set of the most commonly used neurophysiology experimental data entities such as continuous time series, time series segments, spike times (e.g. events) and images, and may be extended arbitrarily to accommodate other data entities.

By explicitly specifying service inputs as being of a specific NDF data type, we can ensure that only the correct NDF data type is used. This is achieved at service runtime by the portal, by examining the input NDF file header for compatible data. NDF also provides some additional runtime options regarding selection of data from within an NDF data set. As such, when executing a CARMEN service from the portal, an NDF selection panel is displayed which allows the user to select the start/end time offsets for a recording, and some or all data channels within the recording. The service framework needed to take account of these additional options; in particular, a method was needed to pass these options into the service application's command line program. To achieve this we replaced any input filename parameters in the command line argument list, with the name of an XML file that

contains an NDF Service Input XML form containing all the necessary optional information, and the name of the actual input NDF file.

NDF toolboxes (in C/C++ and Matlab) [16] are available that allow the service developer to manipulate NDF data within their service application code. An extension was added to the NDF C/C++ and Matlab toolbox to allow NDF services to read the data from the NDF input file, based on the preferences provided in the NDF Service Input XML. This simplifies the programming effort substantially.

7. CARMEN Service Builder

Item 1 of our specification requires users' services to execute on a pool of processors. To enable this, the users' software needs to be compatible with these processors and their environment. Essential to this is providing a compatible environment that the user can test their code on locally; the CARMEN Service Builder provides this. It is a Java application that allows CARMEN users to wrap applications into CARMEN services though a simple directed approach. It is installed on a machine that replicates one of the CARMEN execution nodes, either as a Virtual Machine, or as remote-desktop into a pool of user development machines. In this way it can be guaranteed that a service will run on one of the execution nodes.

The Service Builder uses a project-based approach, with one project per service organised on local disk in a directory structure. Inside the top-level directory is the project configuration (XML) file, alongside subdirectories that can contain source code, compiled applications, documentation, test data, tutorials, and a subdirectory that will contain the resultant wrapped service as a JAR and the service metadata XML file.

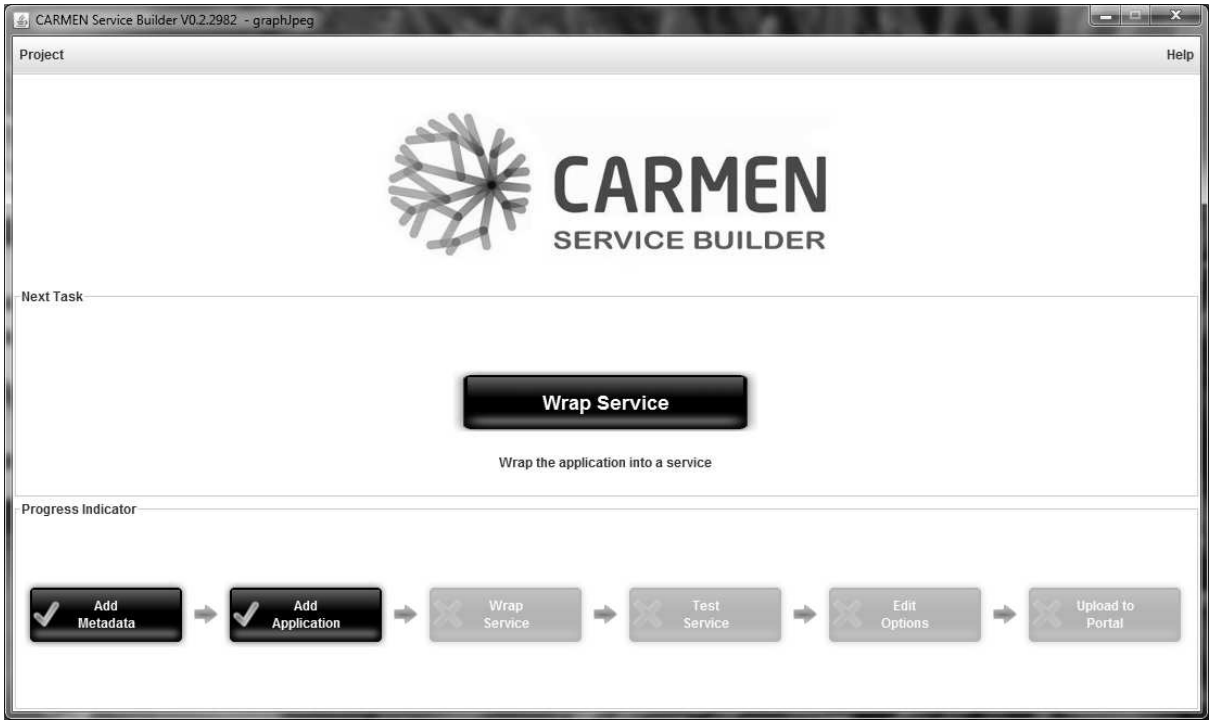

Figure 3 - CARMEN Service Builder Front Panel showing progress and next task

Once a project has been opened or created, the Service Builder displays the front panel (see figure 3). This contains a flow chart showing the steps required to create a CARMEN service, and indicates

progress. Above the progress flow chart is the “next task” button. This is the only way to advance along the process flow chart and ensure that the user follows the process to correctly wrap their application. We developed a number of alternative approaches to the service builder and this method has been shown to be simple and intuitive, even for first time users. The stages in the service builder are summarised in figure 4, and are as follows;

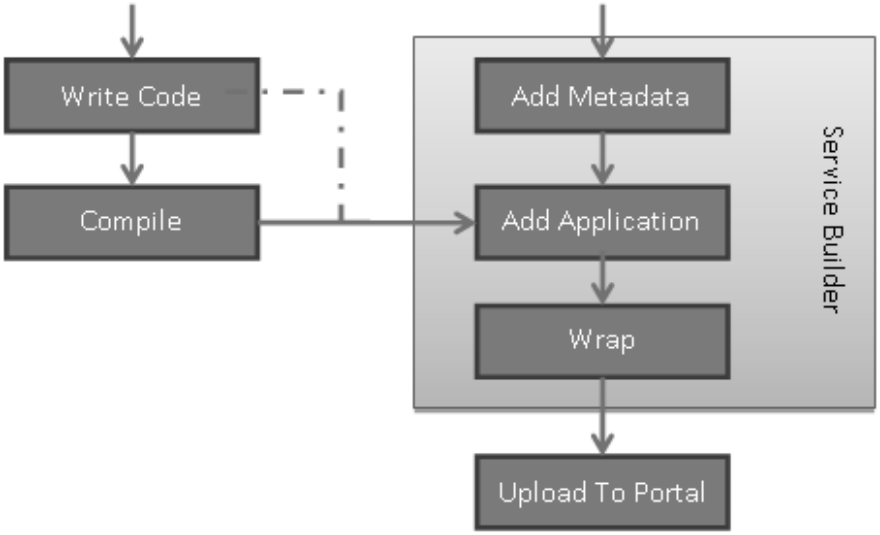

Figure 4 - Wrapping process

Assuming the application has been created, the first Service Builder process is to gather the service metadata. It consists of three panels for general metadata, input parameter and output parameter metadata entry. The general metadata panel (see figure 5) contains the name of the service along with a brief description and a keyword list. This information is for display purposes in the CARMEN portal and forms the search terms used by the system’s search facility. It also asks for general system information such as the application/package type and version number. The builder currently supports Matlab (as a compiled standalone executable), Python, R, C/C++ executables, Perl, Java and any script/executable that can be written as a command line application. The platform OS type field is used by the job manager to push the service request to the correct execution node server (both Linux and Microsoft Windows servers are supported). There is also a field for coarsely describing resource usage to provide information useful for scheduling jobs over the execution servers. For UNIX-like services that process video data, a graphics display may be required, and a field exists that informs the wrapping code to generate a virtual X-Server.

The input parameter metadata form (see figure 6) contains descriptions of each service parameter, which are used by the CARMEN portal to generate input forms appropriate to each service. A data type field allows users to select “value”, “file” or several NDF file types. If the data type selected is “value”, a text box is displayed in the portal’s “input selection” panel during execution setup. Default values can be specified, and a limited set of validation options may be selected. If a list of known value options is required, a drop-down box can be displayed in the portal instead. If “file” is selected as the data type, the portal displays a file browser allowing the user to select suitable data files from the online repository. A filter can be set to limit choice of file types based on file extension. For each output parameter, there is a description field, and a data type.

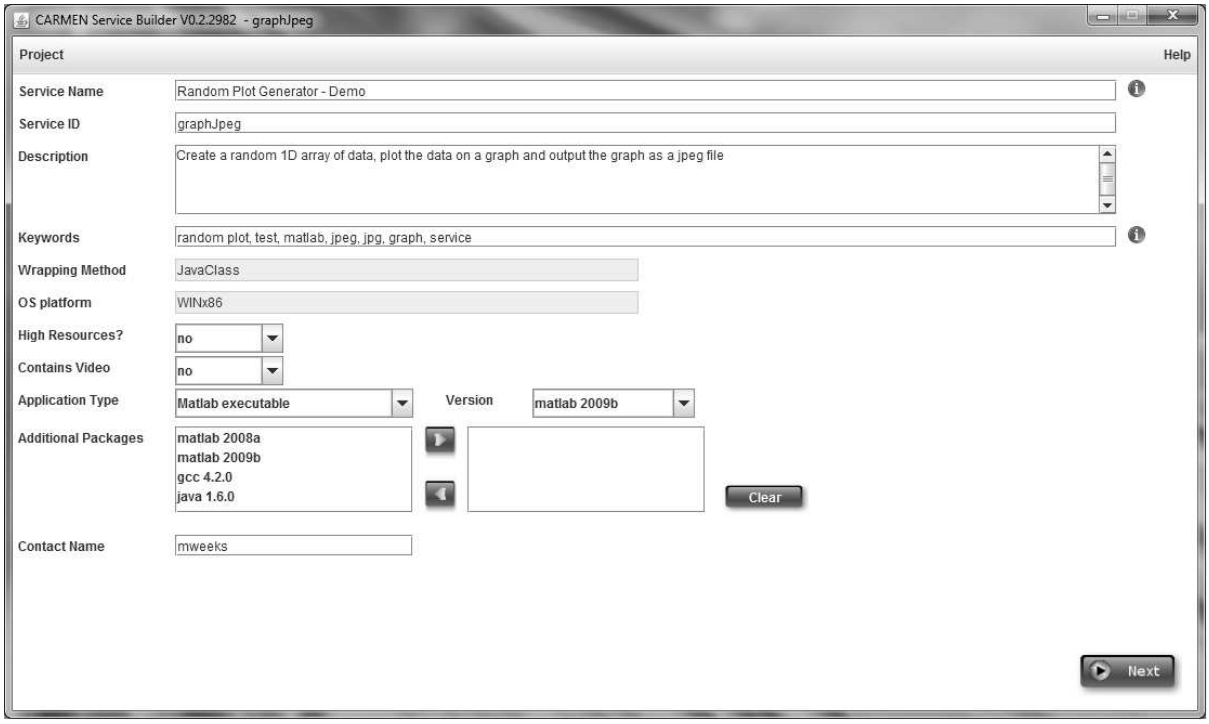

Figure 5 - CARMEN Service Builder – metadata panel.

Once the metadata gathering process is complete, the command line application for the code, whether it be a script or executable must be specified for inclusion in the wrapped service. If supplementary files, such as libraries, packages, functions, etc are required by the command line application, these must also be specified. If the application requires a system-level library or package, a custom installation can be arranged with the CARMEN administrators.

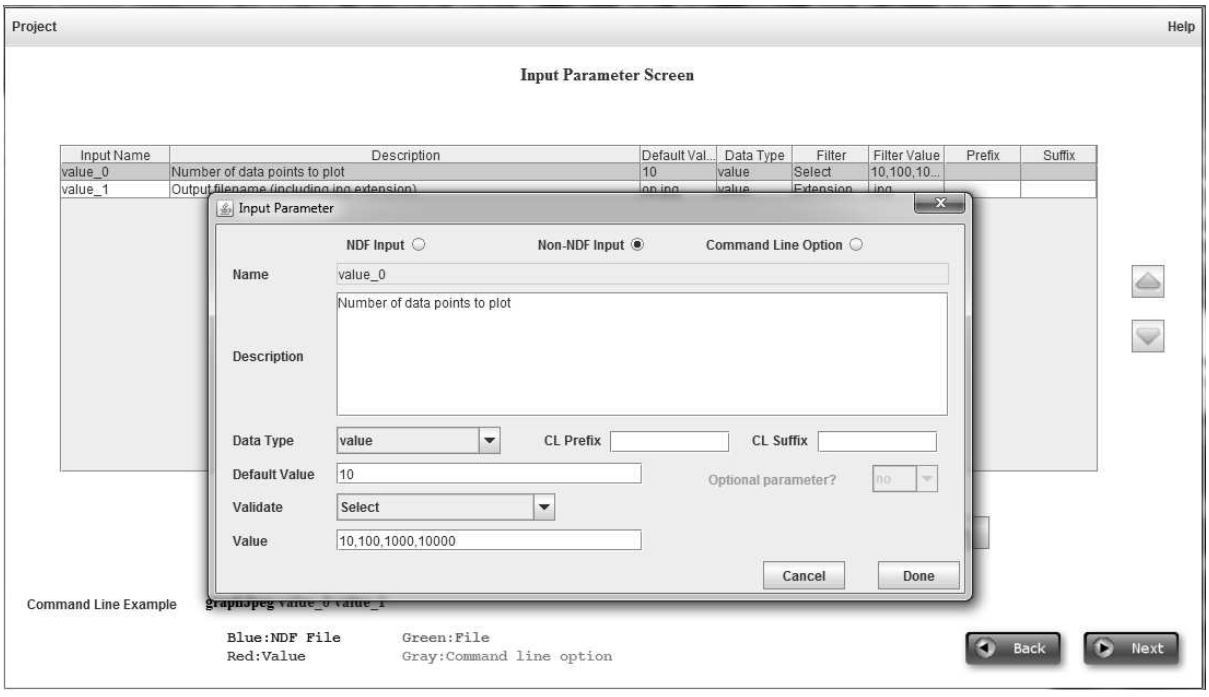

Figure 6 - CARMEN Service builder - Input parameter metadata panel

The application is now ready to be wrapped into a CARMEN service. A single button click generates the Java wrapper code from the metadata, compiles it and then bundles the application and wrapping together into a JAR file. It then gathers the metadata and formats it into an XML file. A log records the progress of the wrapping process and provides feedback should there be a problem.

After this is compete, the XML and JAR files can be published to the CARMEN system via the portal. Ownership of the service is set to a specific user who sets the sharing options for the service; private, shared with specific people or groups, or public to all CARMEN users. Given the correct permission, a newly loaded service can be immediately executed via the CARMEN portal and the results inspected via the service log or data repository.

8. Service Execution Management

The CARMEN system allows users to submit analysis code to be executed within the system as services. As discussed, services can be engineered with almost any toolset or language that can be packaged as a command line application. This application is bound up into a .jar file that contains the application and any support files that it may require. In order to allow the system to manage and manipulate these applications, each one is described by a metadata document and it is these two elements that form a CARMEN service. When a service is added to the CARMEN system it is simply stored in the SRB storage facility until an execution request is made upon which it is dynamically deployed into the execution environment.

The CARMEN execution environment is made up of a collection of machines running Linux and Windows operating system deployed both physically and as virtual machines. Each machine hosts a Service Manager Servlet that is responsible for service execution and data staging on that machine. This Servlet, as shown in figure 7, has a connection to the SRB storage facility and the SQL database that maintains the system state.

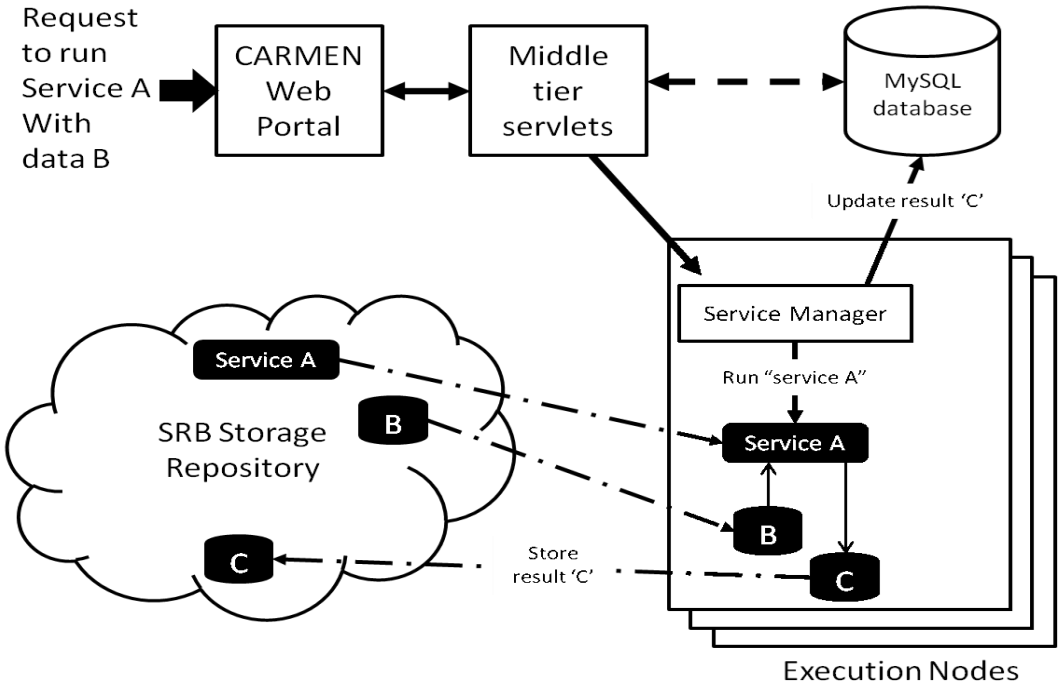

Figure 7 - CARMEN Service Dynamic Deployment

When a service execution request is made to the CARMEN system, a scheduler selects a machine within the pool based on the service type, target operating system and loading factor. The target machine then takes responsibility for managing the execution and any data staging that may be required. If the target execution machine does not have the service installed, it is copied from the SRB system and the service is installed simply by extracting the .jar file. All data for the service execution is then copied from SRB into a local directory and the service application invoked. Once the application completes the output files are then stored back in SRB and the service state within the SQL database is updated. All input and output data files are then deleted from the execution machine, but leaving the service place ready for the any further execution requests. Services are only removed when either the resource they use is needed or they are updated or deleted from the system.

9. Service Execution via the CARMEN Portal

Once a service has been published to the CARMEN system it is available for execution by authorised users. Users can discover services via the search facility, and browse their descriptions to decide upon their requirements. The user can select a particular service, and the service execution panel is displayed, as shown in figure 8. From the service metadata the portal generates user interface elements that allow the user to provide values for the input parameters for the service. The constraints associated with the parameters allow the portal to validate the user inputs which could be values, a selection of value choices from a pre-defined set or an input file. In some cases a default value is defined in the metadata. Where a parameter is a file, the portal provides a browse button so that the user can select a file from their data folders within the portal. The constraints may only allow prescribed file types to be selected.

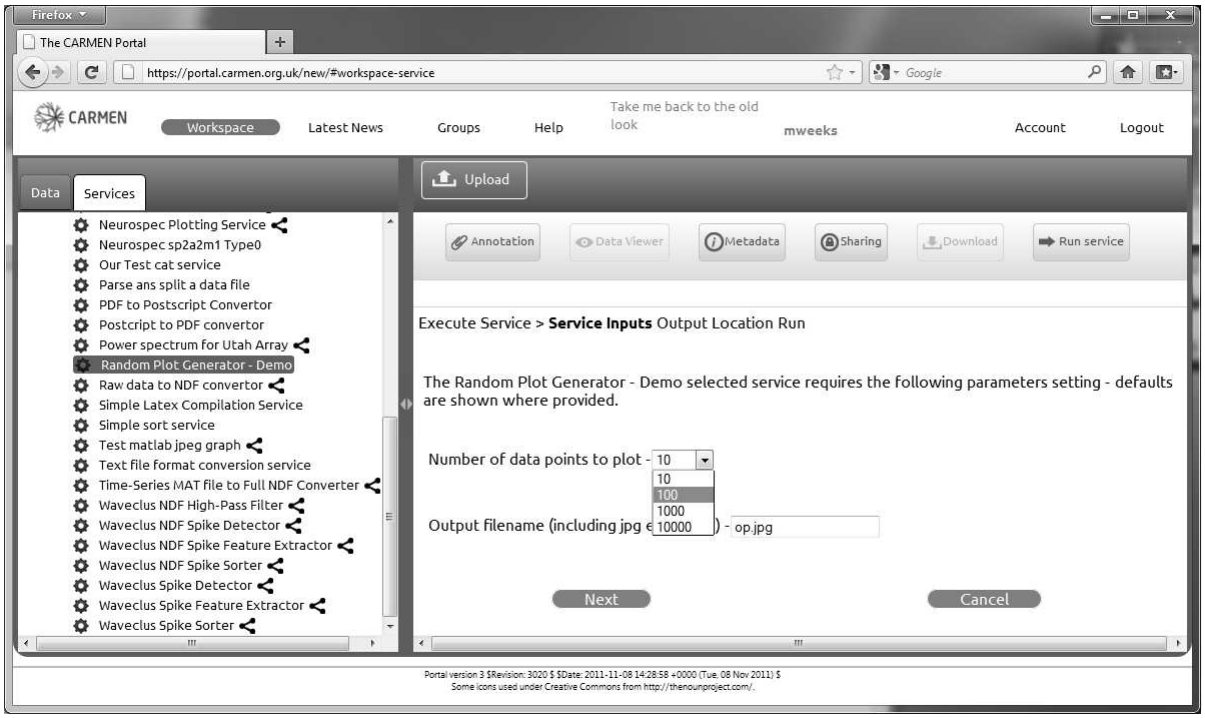

Figure 8 - CARMEN portal service execution panel

The CARMEN system uses the metadata to dynamically deploy and invoke services across the CARMEN execution environment as previously described. At execution time the metadata combined with the parameter values provided by the user enable the system to fetch the required input data and pass this along with any parameters to the service application through a client object created at run

time. The metadata also describes how to handle the output from the application so that it is stored back within the data system.

Once a service has been invoked, the portal displays a Service Log which regularly updates to show the current state of the service. Upon service completion, the output is available for viewing through the portal as shown in figure 9.

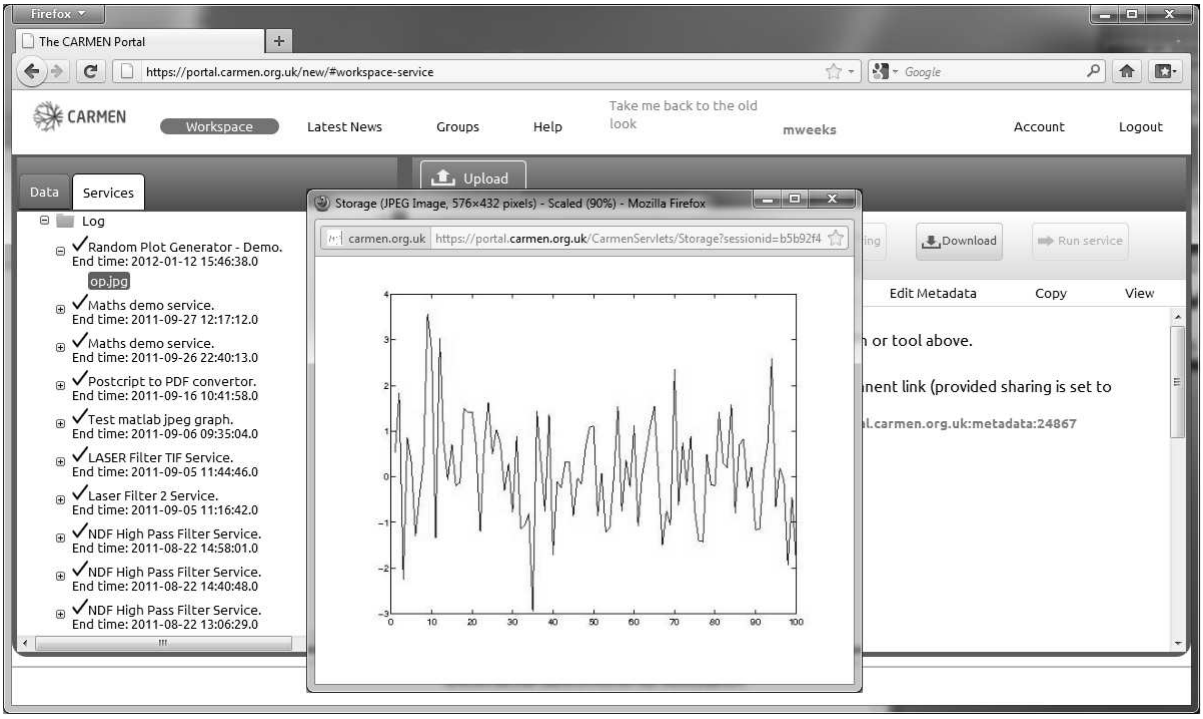

Figure 9 - CARMEN Portal - Service Log showing result file

10. Workflow Integration

A workflow tool allows multiple services to be combined to create more complex processing or analysis methods using existing services. It can also provide a more efficient way of running services for the user. For CARMEN we evaluated the Taverna [17] and E-Science Central [18] workflow systems, however two specific areas created insurmountable problems; presenting an integrated environment with a common interface, and integrating CARMEN's NDF data format.
At the time that Taverna was being evaluated the interface was a standalone Java application that would need to be installed and run outside of our existing CARMEN interface, thereby breaking with the idea of a single point of access (an internet browser client) and the need for users to not have to install any software. In order for Taverna to be capable of orchestrating CARMEN services, a plug-in was required that would allow Taverna workflow to understand CARMEN services. All attempts to generate this plug-in were unsuccessful, being dependant on a significant bug fix due in a future release of Taverna.

The E-Science Central (ESC) workflow system did not suffer these problems, but as with Taverna, the integration of the NDF data format was problematic. THE NDF format can encapsulate multiple data formats within a single container, e.g. image/video data, time-series signal data and segmented time-series data. Both systems allow the user to select the input to a service within a workflow but neither had a way to cope with the capability of NDF to encapsulate multi-format data. A primary requirement for CARMEN was the ability to select specific channels and time windows within a single multichannel data source stored within an NDF file. It was not unusual for a service to require

an NDF input file with two data formats with in it, e.g. the K-Means Spike Sorting service takes both raw time series data and segmented time series data from a single NDF file. Both Taverna and ESC represent these as separate entities although they were in the same file, and would not inspect the file types until run time, which would be too late as the user must make choices at design time. These are examples of issues that lead to the CARMEN team developing a CARMEN specific solution that was tightly coupled with both CARMEN's user interfaces and back end processes. These activities were often undertaken under advice and discussion from external teams, such as ESC and Taverna.

We have created our own workflow tool and are currently integrating it into the platform. The CARMEN Workflow Tool is Java-based and designed to make use of CARMEN Services and NDF. The workflow tool supports both data and control flow, and allows parallel execution of services. Using the Service Invocation API to invoke CARMEN services simplifies the workflow infrastructure substantially. This API allows the workflow engine to make use of our dynamic service deployment and execution system, achieving scalable heterogeneous distributed processing. The workflow tool consists of a graphical design tool, a workflow engine, and access to a library of CARMEN services and common workflow tasks.

The graphical workflow design tool is embedded into a workspace panel on the CARMEN portal. It provides the user with the ability to create workflows consisting of workflow “processors” and connections. A workflow processor can be a previously registered CARMEN service, either one belonging to the user, or a service that has been shared with them. A workflow processor can also make use of a set of pre-defined Java-based workflow services. These Java-based workflow services provide basic tasks such as workflow data input/output, control flow operations, logical and arithmetic operations, and other common workflow tasks. For data input to a workflow processor, the data files may be the user’s own data, or again, data that has been shared with them by another CARMEN user.

When creating a workflow, processors are placed in the graphical design panel. A searchable library of CARMEN services allows the user to select appropriate workflow processors, and import them into the design panel. Connections are achieved by dragging a line from a service output to the next service’s input port. Data validation to match both ends of a connection is performed on the fly so that incorrect data input to a service can be avoided. To simplify the connection process, when the user drags a connection from a service output port, all compatible service input ports are highlighted in a specific colour. Once a workflow has been created, it can be saved to the CARMEN repository as a workflow description XML form, and access permissions can be set to allow other CARMEN users to view or execute it. The workflow description XML form follows a similar form to myGrid’s SCUFL (Simple Conceptual Unified Flow Language) script [19], but modified to suit our service and data description formats. At a later date, the workflow can be retrieved from the CARMEN repository by browsing or searching the user’s or shared workspace, and re-run again.

When a workflow is to be executed, the workflow description XML form is passed to the workflow engine. The workflow engine resides on one of the CARMEN system’s backend servlets, running on one of the execution server nodes for scalability. The workflow engine parses the workflow XML form, extracting the workflow processors and their connections. The workflow engine checks that the user executing the workflow has access permissions to use the resources described in the workflow XML form. If the processor is one of the pre-defined Java services, it will be executed inside the workflow engine. If the processor is one of the user’s registered CARMEN services, its metadata is pulled from the system metadata database, and a service execution request form (XML) is generated. This form is passed to the service execution API, whereupon a request is passed to a suitable execution node for execution of the service. On completion, the result (or a reference to it) will be

returned to the workflow engine, and the workflow execution progress and results are displayed to the user in the portal's graphical view.

The workflow tool also supports NDF services and data. Data types, channels and recording time offsets can only be selected in the initial NDF input data handlers. Subsequent services process all available data, channels and recording times. NDF connection validation is also performed in the graphical workflow designer, so that ports with differing NDF data types cannot be interconnected.

11. Service Discovery

When a user wishes to run a specific CARMEN service, techniques are needed that allow suitable services to be quickly found from within the many services in the system. Search is an obvious candidate for service discovery, as it is scalable and easily achieved and can use the service metadata registered with the system's metadata database.

The search functionality is built using Apache Lucene (http://lucene.apache.org) **[**20**]**. Lucene is an open-source and high-performance Java search engine library. The CARMEN search engine parses the Services metadata to extract the XML tag (field name) and value pairs ready for searching. These pairs are stored in a Lucene index structure with one index document (set of field name and value pairs) per metadata document. In the CARMEN search engine, the full Lucene query syntax is enabled allowing Boolean operators; term modifiers: wildcard searching, fuzzy searching and proximity searching; term boosting; term/phrase grouping; and, fielded searching. CARMEN has three separate searchable sections: data, services and workflow. We have augmented Lucene with both "Did you mean?" functionality based on the functionality of Google search and auto-complete which uses Lucene in conjunction with Google Web Toolkit methods. This is needed as users often misspell search terms. "Did you mean?" spell checks each of the user's query terms against the stored Lucene index extracted from the Services metadata and provides alternative spelling suggestions for any terms that are misspelled. After finding the most similar term in the index to each misspelt query term, CARMEN displays the revised search query to the user as a clickable "Did you mean?" hyperlink which activates the CARMEN search using the corrected query string.

The CARMEN search engine is also enhanced with suggest-as-you-type and autocomplete functionality. Autocomplete lowers the chance of misspelled and redundant query terms by searching the Lucene index generated from the Services metadata. As the user types search terms into the query box, the CARMEN search engine detects when a specific number of new characters have been typed (currently set to four but configurable) and generates a list of suggested words and phrases for the new term. The user may select one of the suggestions or continue typing. In the latter case, the list of suggestions will be constantly refined as new text is typed.

The service search facility is useless without quality keyword terms being associated with a service. When the service is being constructed using the Service Builder Tool, we encourage users to add sufficient metadata keywords, including phrases. Should the user not enter enough keywords and/or phrases, a warning is automatically flagged to them, indicating that their service may not be found without amending the keywords.

Once a suitable service has been found it can be bookmarked for future use; this is similar in practice to web browser bookmarking. Bookmarking is enacted either manually from the service search results, or automatically when a service is run. The bookmarked services are displayed in a list of bookmarked services which also includes the user's own services.

12. Conclusions

Using the CARMEN Services infrastructure, we have incorporated a scalable heterogeneous computational infrastructure into the CARMEN virtual laboratory. The CARMEN Service framework and Service Builder allows quick and easy deployment of algorithms onto the CARMEN system as services. The project based approach means the service details can be amended, and the service rewrapped and deployed very quickly. All of the technical details of the wrapping process are hidden from view by the Service Builder, which is only possible due to the close integration of the metadata with the actual service implementation. This also applies to the CARMEN portal where the service user does not know or see the wrapping technology, or indeed the application or platform that the service is executing on. In this way the user only need focus on the functionality of the service.

Our dynamic deployment scheme, via the metadata system, allows services to be staged across distributed execution nodes, consisting of several platforms and application configurations. The SRB system allows fast deployment of services and data files to the execution nodes, aided by the small overhead in the service wrapping.

During the execution of a service, both the service, the data it operates on and resultant data reside purely on the CARMEN platform. The user's access device needs only to be powerful enough to run a web browser. The portal has been successfully used to execute processor intensive services on large datasets from thin client devices such as an HTC Desire S Smartphone and an Apple iPad tablet.

The CARMEN system is currently hosted on a small cluster of Linux servers. At this point in time a Cloud infrastructure is being installed to replace the current hosting environment. A view is also being taken to expanding CARMEN to the Eduserv Education Cloud [21]. A clear issue when expanding the system to external Cloud providers is the implications for sensitive data such as human data, the use of which is highly regulated. This is a hot topic for discussion within the CARMEN project and ties in with a use case for a federated CARMEN where sensitive data can be stored within owning organisations, such as the UK National Health Service (NHS).

The main benefits of moving to Cloud hosting from a CARMEN perspective include increased server utilisation, and the ability to dynamically manage execution node types, matching whatever OS and software is currently in demand. Also, a VM repository containing all of the OSs we have ever used will support software sustainability. With old operating systems, there is the possibility that they will become a security risk when they are no longer supported. Normally, we would have to retarget the service at a newer OS. However, if it is not a trivial task to migrate the service to a newer platform, we can simply run the service using the old OS as a VM in a secure sandbox. If there is a problem with a service that corrupts a VM, the VM can be simply deleted and a new one instantiated in its place, and overall the system remains unaffected. The secure sandbox method of running VMs can also be used to run new, untrusted, or script-based services.

One of the aims of the follow on funding for CARMEN is to explore the possibilities for a federated CARMEN system. To date some initial work has been undertaken to understand the requirements and implementation details for a federated capability. An early prototype was built that allowed users to connect to independent CARMEN installations and view resources across the federation. However, without a specific use case or request for this capability the work has been of a lower priority. Recent discussions have raised the prospect of an actual requirement for a federated CARMEN and so the functionality is being re-appraised.

At the time of writing (July 2012), there are 85 neuroscience-based services registered with the CARMEN system. A breakdown of the number of services per operating system, and application are shown below in Table 1 and Table 2. The majority of services are running on Linux, however this is more a reflection of our physical machines being Linux-based, rather than the user's choice of OS. As we move towards a VM-based architecture, we expect this to change and to reflect the user's OS of choice.

Windows	Scientific Linux 4	CentOS 5 Linux
1	11	73

Table 1 - Number of services per OS platform type

Matlab executable	R script	Python script	C/C++ executable	Other executable
63	5	1	2	14

Table 2 - Number of services per application type

The CARMEN system is targeted towards the neuroscience community; however it is a wholly generic platform, and could in fact be targeted towards any disciplines. The YouShare [22] project has expanded the concept of CARMEN further, offering a general platform available to UK University researchers. YouShare users are members of a larger community of mixed disciplines, though they can create or join subgroups related to a subject area or organisation. YouShare also provides the ability to create branded groups, which allows group projects to be set up within YouShare, which can look and feel like a separate system. CARMEN has now been transparently moved into one such branded group within YouShare, and the technology we have developed for CARMEN services is now fully incorporated inside YouShare.

13. Acknowledgements

CARMEN was developed under funding provided by the UK EPSRC on contract number EP/E002331/1 and is now supported by the UK BBSRC under contract BB/I000984/1. The funding for YouShare is provided by the HEFCE University Modernisation Fund.

14. References

[1] Watson, P., Jackson, T., Pitsilis, G., Gibson, F., Austin, J., Fletcher, M., Liang, B., and Lord, P. 2007 The CARMEN Neuroscience Server. UK e-Science 2007 All Hands Meeting, Nottingham, September 2007.
[2] Papazoglou, M.P. 2003 Service-oriented computing: concepts, characteristics and directions, Proceedings of the Fourth International Conference on Web Information Systems Engineering. WISE 2003, pp. 3- 12, Dec. 2003.
[3] Liang, B., Fletcher, M., and Austin, J. 2010 The design and implementation of a Neurophysiology Data translation Format (NDF), ninth UK e-Science All Hands Meeting (AHM 2010), City Hall, Cardiff, UK. 16th September 2010.
[4] Eckerson, W. 1995 Three Tier Client/Server Architecture:Achieving Scalability, Performance, and Efficiency in Client Server Applications, Open Information Systems, vol. 10, Jan. 1995.
[5] Baru, C., Moore, R., Rajasekar, A., and Wan, M. 1998 The SDSC Storage Resource Broker, Proc. CASCON'98 Conference , Nov.30-Dec.3, 1998, Toronto, Canada, see http://www.sdsc.edu/srb/index.php/Main_Page.
[6] Jessop.M., Weeks, M., and Austin, J. 2010 CARMEN: a practical approach to metadata management, pp. 4147-4159, Phil. Trans. R. Soc., 368, published by The Royal Society on 1- 9- 2010.
[7] Oracle 1994 Java Servlet Technology: http://www.oracle.com/technetwork/java/javaee/servlet/index.html – retrieved 17th January 2012.

[8] Fielding, R. T. 2000 Architectural styles and the design of network-based software architectures. PhD Dissertation, Dept. of Information and Computer Science, University of California, Irvine.
[9] Google Web Toolkit (GWT): http://code.google.com/webtoolkit/ - retrieved 17th January 2012.
[10] Garrett J.J. 2005 Ajax: A New Approach to Web Applications. http: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. Retrieved 17th January 2012.
[11] InstantSOAP Project. http://instantsoap.sourceforge.net/. Retrieved 17th January 2012.
[12] Metro JAX-WS: http://jax-ws.java.net/ - retrieved 17th January 2012.
[13] Apache Tomcat: http://tomcat.apache.org/ - retrieved 17th January 2012.
[14] Gibson, F. et al. 2008 Minimum Information about the Operation of a Web Service (MIAOWS). Published in Google Knol, http://knol.google.com/k/minimum-information-about-the-operation-of-a-web-service-miaows, November 2008.
[15] Web Services Description Working Group: http://www.w3.org/2002/ws/desc/ - retrieved 13th December 2011.
[16] NDF Toolbox and documentation: http://www.carmen.org.uk/software/descriptions - retrieved 20th January 2012.
[17] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P. and Oinn, T. 2006 Taverna: a tool for building and running workflows of services., *Nucleic Acids Research*, vol. 34, iss. Web Server issue, pp. 729-732, 2006.
[18] Woodman, S., Hiden, H., Watson, P. and Cała, J. 2009 Workflows and Applications in e-Science Central. All Hands Meeting, December 2009, Oxford, UK.
[19] myGrid SCUFL: http://www.mygrid.org.uk/dev/wiki/display/developer/2010-07+SCUFL2 - retrieved 12th July 2012.
[20] McCandless, M., Hatcher, E., and Gospodnetic, O. 2010 Lucene in Action, Second Edition: Covers Apache Lucene 3.0. Manning Publications Co., Greenwich, CT, USA, 2010.
[21] Eduserv Education Cloud: http://support.cloud.eduserv.org.uk/home - retrieved 12th July 2012.
[22] YouShare: http://www.youshare.ac.uk/ - retrieved 12th July 20122