Lessons Learned from the GECEM Project

David W. Walker
School of Computer Science
Cardiff University
5 The Parade, Roath,
Cardiff CF24 3AA, United Kingdom
david@cs.cf.ac.uk
WWW home page: http://users.cs.cf.ac.uk/David.W.Walker/

Abstract. The Grid-Enabled Computational Electromagnetics project (GECEM) has developed a portal for performing electromagnetics simulations. The portal is based on the GridSphere portal framework and uses JSR168-compliant portlets to access remote web services. The GECEM portal supports an execution pipeline that starts with an input geometry which is processed to generate surface and volume computational meshes, which in turn are input to a computational electromagnetics (CEM) simulation. The CEM simulation produces the final output file which consists of a vector of values at each mesh point. A distributed collaborative visualization tool has been integrated into the portal to view the CEM simulation results. This paper discusses how the GECEM portal can be extended into a more general portal for a certain class of scientific computation. A model of a scientific portal will be presented in which abstract workflows are built out of workflow patterns. The resulting workflows are then embedded into the portal for use by end-users. A virtualized data store may be used to support checkpointing and archiving.

1 Introduction

This paper describes the Grid-Enabled Computational Electromagnetics (GECEM) project, and discusses its main research outcomes and the lessons learned from the project. In particular, a model is proposed for the composition and use of distributed service-based applications that addresses the perceived and stated needs of scientific end-users who have little or no expertise in portals and service-oriented infrastructure.

The GECEM project was funded mainly by the UK's Department of Trade and Industry as part of its contribution to the e-Science Core Programme[1], with additional contributions coming from the project's industrial partners, BAE SYSTEMS and Hewlett-Packard ([1]-[3]). The other collaborating partners were Cardiff University, the University of Wales Swansea, the Welsh e-Science Centre, and the Singapore Institute of High Performance Computing. The project was completed at the end of 2005, and ran for 27 months. Further details may be found in the project final report[2].

The overarching objective of the GECEM project was to apply Grid technologies to enable large-scale scientific and engineering research across a globally-distributed extended enterprise in which the partners only partially trust each other. Subsidiary project objectives included the exploration of secure code sharing and computation capability; the grid-enablement of legacy codes by exposing them as web services; and, the development of a GECEM portal as an integrated user interface to the underlying GECEM services and tools. Computational electromagnetics was chosen as the target application area because of the particular interests and expertise of the project partners, however, the same approach and techniques used in the GECEM project could be applied equally well to other areas, such as computational fluid dynamics and structural mechanics.

The remainder of this paper is arranged as follows. Section 2 presents an overview of the GECEM application and the Grid infrastructure on which it runs. In Section 3, issues relating to trust and security in the GECEM project are discussed, and the GECEM portal is described in detail in Section 4. In Section 5 the lessons learned from the GECEM project are enumerated, and a model of a scientific portal is presented in which abstract workflows are built out of workflow patterns that can then be embedded into the portal for use by end-users. Related work is discussed in Section 6. Ideas for future work and a summary of the main points of the paper are presented in Section 7.

2 GECEM Grid and Application

The GECEM application can be viewed as a workflow, or execution pipeline, with four main stages (see Fig. 1):

1. Creation of the surface mesh from a specification of the geometry of the object to be modelled. This takes as input a file describing the geometry of the problem, typically generated by a CAD system, and outputs a file describing the resultant surface mesh.
2. Creation of the volume mesh based on the surface mesh file generated in step 1. This outputs a file containing the volume mesh.
3. Solution of the computational electromagnetics (CEM) simulation. This takes as input the surface and volume mesh files generated in steps 1 and 2, and outputs a file representing the solution.
4. Perform collaborative visualization of the output file.

[1] http://www.epsrc.ac.uk/ResearchFunding/Programmes/e-Science/default.htm
[2] http://www.wesc.ac.uk/projectsite/gecem/doc/GECEM%20Final%20Report.pdf

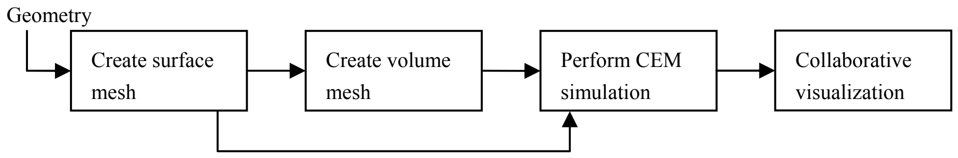

Fig. 1. The GECEM execution pipeline.

In addition to the main dataflows represented by arrows in Fig. 1, each stage in the workflow takes a small number of additional control inputs stored in files. In the GECEM project the objects to be modelled are typically quite complex, such as aircraft and ships.

The first three stages in the workflow perform the three main numerical tasks – the input geometry is converted into surface and volume meshes which are then used to carry out the CEM simulation. Each of these stages is performed by a separate executable legacy code written in Fortran. This code is called from a C wrapper which in turn is wrapped as a Java program using the Java Native Interface (JNI). This is then deployed as a service within an Apache Tomcat container.

A demonstration based on the Globus Toolkit 2, showing the transfer of files between the different stages in the GECEM workflow and the execution of the GECEM services, was given at the UK e-Science All Hands Meeting in September 2003. This showed the basic functionality of the GECEM virtual organisation, and was subsequently developed into the GECEM portal, discussed in Section 4, based on the Globus Toolkit 3.2. The collaborative visualization capability, which is shown as the final stage of the GECEM workflow in Fig. 1, was added in the last few months of the project. This uses the Resource-Aware Visualization Environment (RAVE) which is an infrastructure based on web services for supporting collaborative visualization in a distributed environment.

The first two stages of the GECEM pipeline are performed by the Surface Mesh Generation Service and the Volume Mesh Generation Service, the executable code for which resides permanently on particular hosts. However the CEM simulation step of the pipeline is performed under the control of the CEM Migration Service. Invocation of the CEM Migration Service causes the CEM executable to be migrated to a selected target machine, together with the user-specified input files. The code then executes, its output is sent to a user-specified location, and the code on the target machine is then deleted, along with any associated datasets. The CEM Migration Service is discussed further in Section 3.

Grid infrastructure compatible with the Open Grid Services Architecture (OGSA) was used to establish a virtual organization across the project participants. This infrastructure provided for the authorization and authentication of users, the exchange of data files between sites, and the remote execution of applications. In the GECEM Grid the services for surface and volume mesh generation and CEM migration were located on machines at the University of Wales Swansea (UWS). The services for supporting collaborative visualization using RAVE were hosted at the

Welsh e-Science Centre (WeSC). The CEM Migration Service offered a choice for migrating the simulation code from UWS either to a machine at Cardiff University, or to a machine at the Singapore Institute of High Performance Computing (IHPC).

A typical use case of the GECEM Grid is shown in Fig. 2 in which a geometry file created by designers at BAE SYSTEMS is input to the mesh generation services at UWS. The resulting surface and volume meshes are then passed to the CEM Migration Service (also at UWS) which migrates the executable code and input files to a machine at WeSC. After the CEM simulation code has executed at WeSC the results are then examined in a collaborative visualization session.

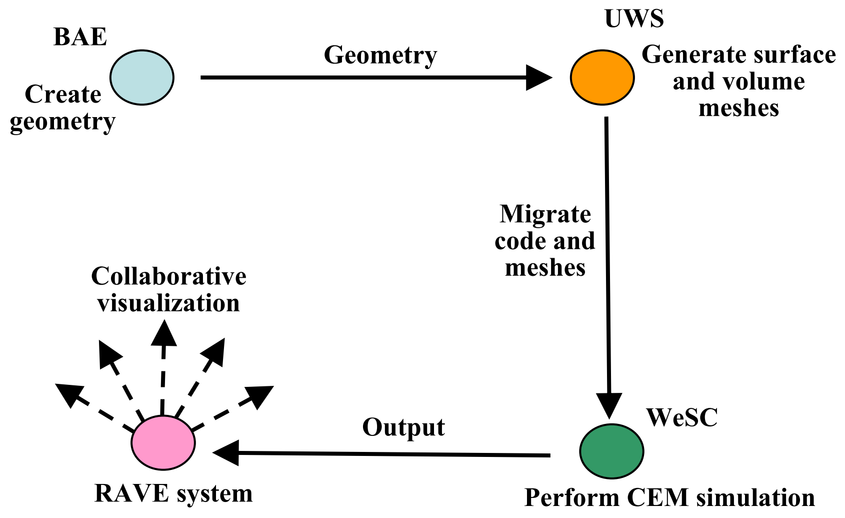

Fig. 2. Typical use case for the GECEM Grid. The dashed arrows emanating from the RAVE system represent the collaborative visualization of the CEM simulation output.

3 Trust and Security in the GECEM Grid

The GECEM Grid represents a type of extended enterprise in which the partners only partially trust each other, which places constraints on how resources are shared. For example, designers at BAE SYSTEMS may be prepared to share geometry files but not the software systems that create these files. UWS may allow authorized users to access their codes as web services, but may not want to permit users to logon to their machines to execute the codes directly from the command line. Similarly, the owners of the high performance machines at WeSC and IHPC may not wish to permanently host the CEM simulation code, but may allow it to reside on their machines on a short-term basis. In many extended enterprises it is this type of partial trust that mandates a distributed solution, since if there were complete trust between all partners all the software and data could be placed at a single location.

Authentication of users of the GECEM Grid is based on e-Science certificates. These are X.509 certificates issued by the UK e-Science certificate authority. Since the services accessed through the GECEM portal are based on GT3.2, the portal makes use of the Grid Security Infrastructure (GSI) for the authentication of users, services, and resources[3]. GSI also provides for "single sign-on" to Grid resources and the delegation of credentials. Single sign-on refers to the ability of a user to perform a single action of authentication (such as entering a password) to access the distributed resources that he or she is authorized to use. Delegation is a mechanism whereby a user or service can delegate a subset of their access rights to another service.

The GECEM portal uses the MyProxy online credential repository [4] managed by the Grid Operations Support Centre of the UK e-Science programme[4]. The MyProxy Upload Tool, developed in the CCLRC DataPortal project, is used to upload a user's proxy credentials to the MyProxy repository. The user can choose how long they wish their credentials to be kept in the repository and how long any proxies generated are valid. The user also needs to choose a username and MyProxy pass phrase, which is subsequently used to log into the portal, effectively giving single sign-on access to the GECEM resources. GridFTP is used to perform secure file transfers between sites in the GECEM Grid.

The GECEM project explored the concept of the secure migration of applications in which an executable code is securely migrated to a remote computer, its execution is initiated and its progress monitored, and then it is deleted on completion, returning the output to the user. The intent is to leave no permanent trace of the application executable or its input and output files on the machine where the application is executed, and to ensure that no third parties (including system administrators) can access or interfere with the application or files during the migrate/execute/return cycle. Current computer architectures and operating systems allow system administrators complete control over the operating system kernel. Hence the system administrator can spy on and interfere with any application. Researchers in the area of secure remote execution are investigating the concept of "platform virtualization" in which a Virtual Machine Monitor (VMM) runs at the lowest level of the software architecture, below the operating system kernel, thereby preventing the operating system from having direct access to the machine hardware [5]. Virtualization allows the same compute hardware to run multiple operating systems simultaneously, with the VMM providing each operating system with an abstraction of the real machine hardware called a Virtual Machine (VM). The VMM ensures that each operating system running on top of a VM is kept separate, and acts as a control point restricting what an operating system can do with the hardware resources of the system. Thus, if a "guest" application is run atop its own VM it will be secure from other general users and from the administrators of any other operating systems running on the system. Some future chipsets will provide support for virtualization and hardware physical protection facilities that will allow the secure migration and remote execution of applications.

[3] http://www.globus.org/security/
[4] http://www.grid-support.ac.uk/

In the case of the Surface Mesh Generation Service and Volume Mesh Generation Service the executable code resides permanently on particular hosts. However, the CEM Migration Service differs in that it causes the executable code to be securely migrated to a selected target machine, together with any necessary input data sets. The code then executes, its output is sent to a user-specified location, and the code on the target machine is then deleted, along with any related data sets. This behaviour has many of the features required for the secure migration and remote execution of an application. However, the application code and files are vulnerable while on the remote computer – in fact, as noted above, it is currently impossible to ensure completely secure remote execution (or, indeed, secure local execution). However, the CEM Migration Service does reduce exposure to risk in remote application execution.

4 The GECEM Portal

The GECEM portal is a problem-solving environment (PSE) composed of a collection of JSR168-compliant portlets and services for mesh generation and CEM simulation. A portlet is a pluggable user interface component used with the context of a portal framework. From a user's point of view a portlet is a window in a portal that provides a specific service or function. A portlet processes requests and generates dynamic content, and the content of multiple portlets are typically aggregated together to form a portal web page. A portlet's life cycle is managed by a portlet container. Portlet standards, such as JSR-168 and Web Services for Remote Portlets (WSRP), are helping to make portlet-based portals the most common way of presenting aggregated web content to consumers [6].

The GECEM portal provides the main interface through which services are accessed. The portal supports the composition of applications from service-based components, the execution and monitoring of such applications on remote resources, and collaborative visualization, exploration, and analysis of the application results. In addition, the portal also provides an interface to meshing services and supports the collaborative visualization of meshes.

Two key decisions on the design of the portal were made early in the project. The first was to use the publicly-available open-source GridSphere[5] portal framework as the container of the GECEM portal. The second design decision was to base the GECEM services on Globus Toolkit 3.2, as this was the most recent version of the toolkit available early in the project. GT4 was not available until close to the end of the project, and it was decided not to migrate the services to this version of the toolkit. As discussed in Section 2, the GECEM services are simply wrapped legacy executables.

GridSphere 2.0.4 was used in the GECEM portal [7], together with GridPortlets 1.1 [8]. GridSphere was deployed in the Tomcat 5.0.30 servlet container. The GECEM portal has a three-tier architecture, as shown in Fig. 3. The GridPortlets are a set of portlets, developed by the same research team that developed GridSphere,

[5] http://www.gridsphere.org/

and are used for tasks such as credential management, resource browsing, file browsing, and file transfer.

The GECEM portlets in Fig.3 allow a user to set up a job as a sequence of one or more stages in the GECEM pipeline (see Fig. 1), with specified input and output files. Prior to each stage portlets are used to select the input files to be used, and the directory in which to store output files. These files and directories may be on any of the machines that are members of the GECEM virtual organisation. Services are discovered dynamically using a UDDI server at WeSC, and where multiple equivalent services are available for a particular task, perhaps employing different algorithms or numerical methods, the user can select from these. Service discovery and invocation are also controlled through GECEM portlets.

The Resource-Aware Visualization Environment[6] (RAVE) was used to provide a collaborative visualization capability within the GECEM portal. RAVE is a collaborative visualization environment that scales across visualization platforms, ranging from large immersive devices all the way down to hand-held PDAs [9, 10]. RAVE is based on web service technologies, and provides for distributed rendering on remote machines. The data to be rendered may reside on one machine, the rendering may be done on one or more other machines, and the rendered image may be displayed on yet another machine. In the RAVE architecture a Data Service is used to store and distribute the data sets to be rendered. A machine with enough power to render the data may use an Active Client, which reads directly from the Data Service and renders locally. Smaller machines, such as a laptop or PDA may use a Thin Client, which reads rendered frame buffers from an intermediate Render Service hosted on a more powerful machine. Active and Thin clients can join a single visualization session, enabling collaboration between users on vastly differing resources.

A RAVE Portlet was developed and integrated into the GECEM portal. The RAVE Portlet first presents the user with a list of Data Services of choose from. These are discovered dynamically using a UDDI service. The user next initiates a collaborative visualization session, which users at other locations can also join, and then selects a data set to render. Next a Render Service is selected to carry out the rendering – this is selected from a list populated from a UDDI registry. The final step is to indicate whether the local client is an active or thin client – in the former case any render service selected in the previous step is ignored and the data set is rendered on the local client. The data set will then be rendered in the GECEM portal on the local client and on any other machines that have joined the collaborative session. Users can then navigate, and interact with, the data set. Two main modes of collaborative visualization are supported:

1. Each user independently explores the same data set.
2. One user acts as "leader" and all other users view the data set from the same location as the leader.

Users are represented graphically in the visualization by an avatar, which can be seen by other users in the collaborative session.

[6] http://www.wesc.ac.uk/projectsite/rave/

Before visualization the output produced by the CEM simulation must be converted to a form that can be handled by RAVE. This is done within the CEM Migration Portlet of the GECEM portal before accessing the RAVE Portlet.

The addition of audio communication between collaborating users was considered, but it was decided that the recent advent of Voice over IP (VoIP) services, such as SKYPE[7], made it unnecessary to develop a custom solution.

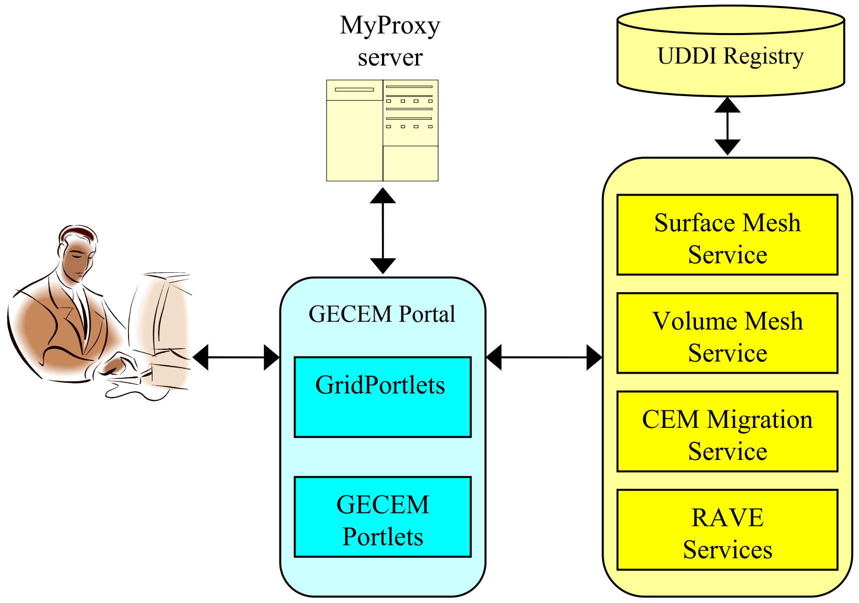

Fig. 3. Three-tier architecture of the GECEM portal.

5 Lessons Learned

The lessons learned from the GECEM project can be divided into two types: technical and non-technical. The non-technical lessons are quite generic and are mainly concerned with the management and conduct of projects with several partners in which there are interdependencies in the software development process and reliance on third-party software. In such cases it is important to avoid single points of failure whereby a particular problem can bring the whole project to a halt. Exposure to risk can be reduced by planning alternative strategies to follow if difficulties arise.

On the technical side, portals were found to be effective in providing a high-level interface for scientific users that shields them from the complexities of using distributed resources via the Grid. Portlets make it easy to integrate heterogeneous resources within a unified interface that can be accessed from any Web browser.

[7] http://www.skype.com/

The model of a distributed application model embodied in the GECEM portal consists of a linear workflow structure with a fixed number of nodes. Each node is a placeholder for a particular type of activity which is implemented by a service. Thus, there is a placeholder for surface meshing, a placeholder for volume meshing, and so on. It is the responsibility of the portal user to associate an actual service instance with each placeholder node in the workflow by making a selection from the services discovered by the UDDI portlet. The static nature of the workflow in the GECEM portal was found to be too restrictive by some of the portal's users who expressed an interest in dynamically composing service-based applications. However, even limited changes, such as adding another placeholder node to the linear workflow, require the portal to be reconfigured and new portlets to be developed by hand. This requires a degree of expertise beyond that of most end-users – indeed, it should be the aim of the portal developer that it be easily usable by those with no expertise in portal technologies. The challenge, therefore, is to support some degree of application composition by typical scientific end-users in a portal built out of portlets. There are numerous tools for composing service-based applications – examples include Triana [11], Kepler [12], and Taverna [13]. However, these are all stand-alone systems that may be difficult to embed within a portal, and that would perhaps provide more features than many end-users require. Furthermore, incorporating such composition tools would introduce unnecessary software dependencies into the portal. A better approach is to perform the workflow composition tasks external to the portal as this ensures a clear separation of form (the structure of the workflow) from content (the actual service instances and inputs used), and allows a number of third-party workflow composition tools to be used to create the initial workflow structure.

The approach to workflow composition advocated here is to use a tool that can design workflow structures out of simple workflow patterns [14, 15]. Once the desired workflow structure has been created, it would then be processed to create a new portal with the workflow embedded in it. As in the original GECEM portal, each node in the workflow would be a placeholder with which the user must associate a service instance, and inputs and outputs of the workflow would be files to be identified by the user. As an example, consider the workflow patterns on the left-hand side of Fig. 4. Pattern A, having just one input and one output, can be used to construct linear workflow structures similar to that illustrated in Fig. 1. Pattern B has two inputs and one output and can be used to construct a much more general class of binary tree structures, as shown on the right-hand side of Fig.4.

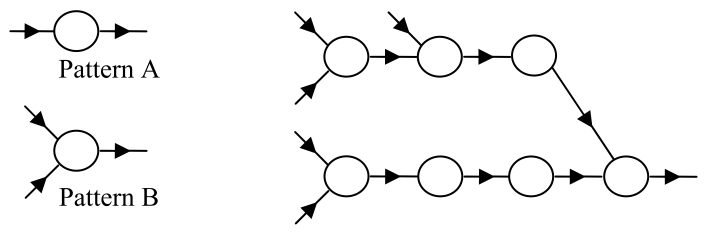

Fig. 4. The right-hand side of the figure shows a workflow structure that can be created from the two workflow patterns on the left of the figure.

The workflow structure on the right-hand side of Fig. 4 has five inputs and one output. Pattern B is a simple merge pattern; one could also consider a split pattern in which a node has one input and two outputs. Indeed, a tool that allows the user to specify the number of inputs and outputs of a node would provide for all the above patterns, and support the design of a large class of data-driven graph-structured workflows. Such abstract workflow structures can be expressed in almost any of the existing workflow description languages.

If services are virtualized and discovered within the portal from a service registry such as UDDI, then there has to be a mechanism for the portal to determine which service implementations match a given placeholder node in a workflow structure. This problem of matching concrete service instances to abstract service specifications is currently a topic of much research (see, for example, [16-18]). The simplest approach is to use a service name to perform the matching, but this works only if all service providers agree to use the same naming scheme. Other approaches make use of metadata and/or ontologies to allow independently-developed services to be discovered and matched to placeholder nodes. Rather than use the metadata and ontology approaches to service matching in end-user tools, it is probably more effective to assume that in the future "spiders" will examine the contents of multiple resource registries, and use the metadata published therein to classify and name services in order to produce meta-registries of services [19]. In a meta-registry all equivalent services will be given the same unique service type which acts as a global name. It is then necessary to distinguish between the local name a service is given in a service registry and the global name it is given in the meta-registry. If a user associates a local name with a placeholder node then the portal would be able to discover all the services with the same type in the meta-registry. These services would then be offered to the portal user who would then select one of them, thereby associating a concrete service instance with the placeholder node. Alternatively the user might browse the meta-registry to find an appropriate service to use.

In the original GECEM portal services are virtualized, but the files that act as input and output to the GECEM services are not. As discussed in Section 7, it would be useful to provide a virtualized file store from which to select input files in the portal. There is then a problem of deciding which files in the virtualized file store are compatible with the inputs of a particular service. The simplest solution is to identify different types of files by a unique file type, and to associate each input and output file of a service with one of these types. As in the problem of matching abstract and concrete services, it is possible to make use of metadata descriptions of files and service inputs/outputs to determine which files are compatible with a particular service input or output. Once again it is possible to use this metadata to associate a unique file type with each service input/output, and to perform this association independently of the workflow design tool proposed here. Thus, it can be assumed that in an abstract workflow it is possible to refer to services by unique service types and to its inputs and outputs by unique file types.

Once a workflow structure has been created, unique service types must be associated with each placeholder node. After this step the unique file type of each nodes inputs and outputs can be determined. Henceforth, the association of unique service and file types with placeholder nodes will be referred to as labeling the workflow. The final step is to embed the labeled workflow into the portal – this will be referred to as compiling the workflow. It should be noted that compiling a workflow does not bind abstract services to specific service instances - this is what the end-user does within the portal. Compiling a workflow automatically generates the portlets corresponding to each node in the labeled workflow. These portlets will be used by the end-user in the portal to specify the specific services to be used (thus, the portlets support discovery and binding). Compiling also configures the portal so these portlets are visible to the user within the portal. The compilation step can also ensure that the workflow is consistently labeled by checking that the unique file type of each node output is the same as that of the node input to which it is connected.

Functionally the design, labeling, and compilation of a workflow, together with the use of the portal itself, are independent tasks that could be performed by distinct tools and interfaces. However, there are a number of ways in which these functions could be combined. For example, the design, labeling, and compilation tasks could be merged into a single tool. In this paper it is assumed that each of these tasks is performed by a separate tool, as shown in Fig. 5. Thus, if the labeling tool accepts as input workflow structures described in a subset of BPEL, this allows existing tools to be used to design the workflow.

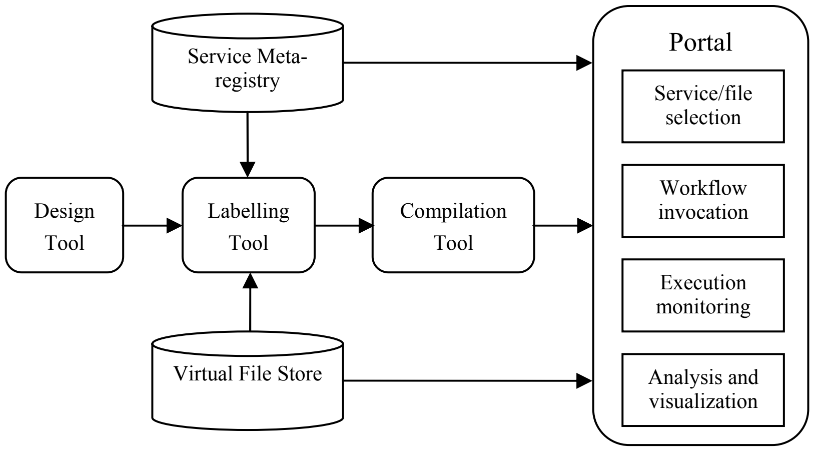

Fig. 5. The relationship between the design, labeling and compilation tools, and the portal. Also shown are some of the functions of the portal.

6 Related Work

The GridSphere portal framework is used across a range of scientific disciplines to create portlet-based portal interfaces[8]. For example, the e-Physics portal developed by researchers at The University of Melbourne has been used to perform parameter sweep studies for a magneto-hydrodynamics astrophysics code, ZeusMP [20]. Unlike the GECEM portal, where the user is responsible for selecting between semantically equivalent services, in the e-Physics portal resource selection is usually done automatically by the Gridbus Broker [21]. This difference arises from the distinct modes of use for which the GECEM and e-Physics portals were designed. The GeneGrid portal [22] is another example of a scientific portal based on GridSphere, and provides access to a virtual bioinformatics laboratory that allows users to construct experiments by either composing new workflows or reusing workflows created previously. The Astrophysics Simulation Collaboratory [23] uses the GridSphere portal framework to manage numerical relativity simulations based on the Cactus Computational Toolkit [24].

The portals mentioned in the previous paragraph all provide end-user interfaces for particular application domains. The P-GRADE portal supports the composition and execution of workflows, and is not tied to any specific application domain [25]. The P-GRADE portal is similar in some respects to the GECEM portal: both portals are based on GridSphere, both represent the input/output of data to/from a workflow node in terms of files, and both make use of certificates, GSI, and MyProxy servers in the authorization and authentication of users and resources. The P-GRADE portal provides a workflow editor that may be used to create new workflows and edit existing ones. For each node in a P-GRADE workflow the end-user must specify the client-side location of the binary executable, and its type (sequential, parallel MPI, or parallel PVM). The end-user must also select the resource that the executable is to run on by first choosing the particular Grid to be used and then the resource on that Grid. These choices are made using dropdown listboxes that are configured by the portal administrator. The P-GRADE portal user must also specify the location of the workflow’s input and output files. Thus, the P-GRADE portal differs from the GECEM portal in that the former does not provide access to virtualized services and files. In addition, the distinction between abstract and concrete workflows, discussed in Section 5, is not clearly made in the P-GRADE portal approach. Another distinction is that the P-GRADE portal uses the DAGMan workflow scheduler [26] to perform file transfers required to execute the workflow, and for the submission of jobs (representing workflow nodes) to Grid resources, whereas the GECEM portal handles these tasks directly by itself.

The concept of an abstract workflow underpins the Griphyn Virtual Data System (VDS; formerly known as Chimera) and its portal interface, Chiron [27]. Virtual data is stored in a Virtual Data Catalog (VDC), and is represented and queried by an XML-based Virtual Data Language (VDL). When queried for a data product, the VDC generates an abstract workflow for creating that data. This abstract workflow can then be used to generate a concrete workflow by mapping requests for data and computation onto actual resources – a process often referred to as “planning”. The

[8] See http://www.gridsphere.org/gridsphere/gridsphere?cid=projects

Chiron portal uses the Pegasus planner [28] to generate a concrete workflow, which is then submitted to DAGMan for execution on the Grid. The Pegasus planner uses a Transformation Catalog to map the logical names of transformations (called a "service type" in Section 5) to physical resources and executable locations, and the Metadata Catalog Service and Replica Location Service for data publication and discovery. Pegasus can also be used independently through its own portal for workflow submission and management.

The Grid Execution Management for Legacy Code Architecture (GEMLCA) provides an easy way to expose legacy application binary codes as OGSA-compatible services [29]. In GECEM this was done by wrapping the original source code, but in GEMLCA a front-end Grid service layer handles parameter passing and contacts a Globus Master Managed Job Factory Service (MMJFS) to submit the legacy application for execution. GEMLCA has been integrated with the P-GRADE portal so that legacy codes and other service components can be used to create application workflows that can then be executed on the Grid [30].

7 Future Work and Conclusions

Based on the work carried out in the GECEM project, Section 5 has introduced a set of tools for building a portal around a particular user-designed workflow structure constructed from simple workflow patterns. To achieve this, an abstract workflow is created by first generating a workflow structure using a design tool and then associating an abstract service (identified by a service type) with each node in the workflow using a labeling tool. The labeling tool also associates a file type with each service input and output. The workflow is then embedded into the portal using a compilation tool, the main task of which is to automatically generate the portlets needed to allow a portal user to select for each node in the workflow a concrete service that is consistent with the abstract service associated with the node, and also to select the input files of the workflow. Service types and file types are essentially names that are unique within the virtual organization using the portal.

The approach described in Section 5 targets end-users who want to perform "what-if" styles of computational investigation in which the particular service that is bound to a node in a workflow may be selected via the portal shortly before workflow execution. For example, this allows a user to experiment with different algorithms for performing a particular task. On a longer timescale the user also wants to create new workflows and to create portals for their execution and management. It is assumed that a user will run many workflows before generating a new one, so it appears sensible to create the abstract workflow external to the portal. This also allows third-party workflow composition tools to be used, provided the compilation tool is able to recognize how they represent abstract workflows. It might be argued that, rather than having a separate portal for each workflow structure, it would be better to manage all the workflows to be used by a Virtual Organization through a single portal interface. This does not appear to raise any technical difficulties, and it would be quite simple to write a portlet for each workflow to integrate them into a single portal.

The portal generated through the design-label-compile process is similar to the GECEM portal in that it provides mechanisms for the user to specify the files to be input to the portal workflow and the services to be invoked. In the current GECEM portal services are virtualized, but not the related input and output data sets. Thus, users currently must identify data sets by specifying a specific file on a specific machine (by using portlets based on the GridPortlets file browser portlet). In future work a virtualized file store will allow authorized users to manage, access, and use data sets without needing to refer to which physical resource they are actually stored on. All input, intermediate, and final data products will be stored in the virtualized file store. This will allow a workflow to be started from any intermediate point for which the necessary input files are held in the virtualized file store. In addition, files will be annotated to support metadata-based searches of the virtualized file store. The metadata would include provenance information (how, why, when, and by whom the file was produced) and other data deemed necessary to provide a description of a file. In particular, the metadata should include sufficient information to distinguish files of different types. The simplest way to do this would be if the files were in XML format – then files conforming to the same XML Schema would be of the same type. The advantage of being able to distinguish files by their type is that it is then possible to match files to the inputs and outputs of services, so when the portal user is deciding what file to select as the input to a service, only those files of the correct type are presented to them.

This paper has shown how end-users, with no expertise in Grid computing or portal development, can make you of simple tools to compose scientific workflows that can then be automatically embedded within a portal. The design, labeling, and compilation tools are used to specify a workflow and generate a portal that supports the end-user in selecting specific services and files for executing an instance of the workflow. The integrated view presented here of scientific workflow composition and portal design addresses concerns raised by end-users of the GECEM portal, and supports a common mode of use of distributed resources based on the input/output of files to/from services. Future work will investigate the use of this approach in other scientific application areas, such as computational fluid dynamics and structural mechanics.

Acknowledgements

The author would like to acknowledge the efforts and assistance of the other members of the GECEM project team, and would particularly like to thank Dr Maria Lin for her work on implementing the GECEM portal, and Dr Yu Chen for supporting the GECEM services. It is also a pleasure to thank Dr Ian Grimstead and Mr Ganesan Subramaniam for their work on developing the RAVE system and the RAVE portlet, Dr. Omer Rana for commenting on a draft of the paper, and Dr Alan Davies for helpful discussions on the GECEM portal.

References

1. D. W. Walker, J. P. Giddy, N. P. Weatherill, J. W. Jones, A. Gould, D. Rowse, and M. Turner, "GECEM: Grid-Enabled Computational Electromagnetics," in Proceedings of the UK e-Science All Hands Meeting, 2003. ISBN 1-904425-11-9. http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/105.pdf.

2. M. Lin, D. W. Walker, Y. Chen, and J. W. Jones, "A Grid-Based Problem-Solving Environment for GECEM," in Proceedings of the Fifth International Symposium on Cluster Computing and the Grid, held in Cardiff, UK, 9-12 May 2005. ISBN 0-7803-9075-X.
http://users.cs.cf.ac.uk/David.W.Walker/papers/CCGrid2005Final.pdf.

3. M. Lin and D. W. Walker, "A Portlet Interface for Computational Electromagnetics on the Grid," in Proceedings of the 17th IMACS World Congress, held in Paris, France, 11-15 July 2005. ISBN 2-915913-02-1.
http://users.cs.cf.ac.uk/David.W.Walker/papers/IMACS2005Final.pdf.

4. J. Novotny, S. Tuecke, and Von Welch, "An Online Credential Repository for the Grid: MyProxy," in Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, pp. 104-111. August 2001.
http://www.globus.org/alliance/publications/papers/myproxy.pdf;

5. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and , Warfield, "Xen and the Art of Virtualization," in Proceedings of the ACM Symposium on Operating Systems, October 2003.
http://www.cl.cam.ac.uk/Research/SRG/netos/papers/2003-xensosp.pdf.

6. F. Bellas, "Standards for Second-Generation Portals," IEEE Internet Computing, Vol. 8, No. 2, March 2004.

7. J. Novotny, M. Russell, and O. Wehrens, "GridSphere: A Portal Framework for Building Collaborations," Concurrency and Computation: Practice and Experience, Vol. 16, No. 5, pp. 503-513, April 2004.

8. M. Russell, O. Wehrens, and J. Novotny, "The GridPortlets Web Application: A Grid Portal Framework," in Proceedings of the Sixth International Conference on Parallel Processing and Mathematics, published by Springer as Lecture Notes in Computer Science, Volume 3911, 2006.

9. I. J. Grimstead, N. J. Avis, and D. W. Walker, "RAVE: The Resource-Aware Visualization Environment," accepted for publication in Concurrency and Computation: Practice and Experience, 2006.

10. I. J. Grimstead, N. J. Avis, R. N. Philp, and D. W. Walker, "Resource-Aware Visualization Using Web Services," in Proceedings of the UK e-Science All Hands Meeting, September 2005.
http://users.cs.cf.ac.uk/I.J.Grimstead/RAVE/AHM2005-full.pdf.

11. I. Taylor, M. Shields, I. Wang, and A. Harrison, "Visual Grid Workflow in Triana," Journal of Grid Computing, Vol. 3, Nos. 3-4, pp. 153-169, September 2005.

12. B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific Workflow Management and the Kepler System,"

Computation and Concurrency: Practice and Experience, Vol. 18, No. 10, pp. 1039-1065, August 2006.

13. T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in Creating a Workflow Environment for the Life Sciences," Computation and Concurrency: Practice and Experience, Vol. 18, No. 10, pp. 1067-1100, August 2006.

14. O. F. Rana and D. W. Walker, "Service Design Patterns for Computational Grids," in *Patterns and Skeletons for Parallel and Distributed Computing*, pp. 237-264, published by John Wiley, ISBN: 1-85233-506-8, 2003.

15. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, published by Addison-Wesley, ISBN: 0-201-63361-2, 1995.

16. S. Balzer and T. Liebig, "Bridging the Gap between Abstract and Concrete Services: A Semantic Approach for Grounding OWL-S," in Proceedings of the Third International Semantic Web Conference, published by Springer as Lecture Notes in Computer Science, Volume 3298, 2004.

17. S. Majithia and D. W. Walker, "Automated Composition of Semantic Grid Services," in Proceedings of the UK e-Science All Hands Meeting, September 2004.

18. R. Lara and D. Olmedilla, "Discovery and Contracting of Semantic Web Services", in Proceedings of the W3C Workshop on Frameworks for Semantics in Web Services, Innsbruck, Austria, 9-10 June 2005.

19. T. Goodale, S. A. Ludwig, W. Naylor, J. Padget, and O. F. Rana, "Service-Oriented Matchmaking and Brokerage," in Proceedings of the UK e-Science All Hands Meeting, September 2006.

20. B. Beeson, S. Melnikoff, S. Venugopal, and D. G. Barnes, "A Portal for Grid-Enabled Physics," in Proceedings of the 2005 Australian Workshop on Grid Computing and e-Research, pub. ACM Press, pp. 13-20, 2005.

21. S. Venugopal, R. Buyya, and L. Winton, "A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids, " in Proceedings of the Second International Workshop on Middleware in Grid Computing, pub. ACM Proess, pp. 75-80, 2004.

22. N. Kelly, P.V. Jithesh, S. Wasnik, R. McLaughlin, F. Fragoso, P. Donachy, T. Harmer, R. Perrott, M. McCurley, M. Townsley, J. Johnston, and S. McKee, "The GeneGrid Portal: A User Interface for a Virtual Bioinformatics Laboratory," in Proceedings of the UK e-Science All Hands Meeting, September 2005.

23. R. Bondarescu, G. Allen, G. Daues, I. Kelley, M. Russell, E. Seidel, J. Shalf, and M. Tobias, "The Astrophysics Simulation Collaboratory Portal: A Framework for Effective Distributed Research", Future Generation Computer Systems, Vol. 21, No. 2, pp. 259-270, 2005.

24. T. Goodale, G. Allen, G. Lanfermann, J. Masso, T. Radke, E. Seidel, and J. Shalf, “The Cactus Framework and Toolkit: Design and Applications,” in Proceedings of the Fifth International Conference on Vector and Parallel Processing,” pub. Springer, 2003.

25. P. Kacsuk and G. Sipos, “Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal,” Journal of Grid Computing, Vol. 3, No. 3-4, pp. 231-238, 2005.

26. D. Thain, T. Tannenbaum, and M. Livny, “Distributed Computing in Practice: The Condor Experience,” Concurrency and Computation: Practice and Experience, Vol. 17, Nos. 2-4, pp. 323-356, 2005.

27. Y. Zhao, M. Wilde, I. Foster, J. Voeckler, J. Dobson, E. Gilbert, T. Jordan, and E. Quigg, “Virtual Data Grid Middleware Services for Data-Intensive Science,” Concurrency and Computation: Practice and Experience Vol. 18, No. 6, pp. 595-608, 2006.

28. G. Singh, E. Deelman, G. Mehta, K. Vahi, and M.-H. Su, “The Pegasus Portal: Web-Based Grid Computing,” in Proceedings of the 2005 ACM Symposium on Applied Computing, pub. ACM Press, 2005.

29. T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G. Z. Terstyanszky, and S. C. Winter, “GEMLCA: Grid Execution Management for Legacy Code Architecture Design,” in Proceedings of the 30th Euromicro Conference, pub. IEEE Press, pp. 477-483, 2004.

30. L. Bitoni, T. Kiss, G. Terstyanszky, T. Delaitre, S. Winter, and P. Kacsuk, “Dynamic Testing of Legacy Code Resources on the Grid,” in Proceedings of the Third Conference on Computing Frontiers, pub. ACM Press, 2006.