Portals and Frameworks for Web Based Education and Computational Science

Geoffrey C Fox
School for Computational Science and Information Technology
And Department of Computer Science
Florida State University
Dirac Science Library
Tallahassee Florida 32306-4130
gcf@cs.fsu.edu

Abstract

We briefly describe an architecture for portals defined as web-based interfaces to applications. In particular we focus on portals for either education or computing which we assume will be based on technologies developed for areas such as e-commerce and the large Enterprise Information Portal market. Interoperable portals should be based on interface standards, which are essentially hierarchical frameworks in the Java approach but are probably best defined in XML. We describe the underlying multi-tier architecture, the key architecture features of portals and give detailed descriptions for some computational science portals or problem solving environments.

1 Introduction

Portals have attracted a lot of attention recently – some of this is good but for others, portals are a fad. We believe that viewed as "Application-specific Object Web based distributed systems", they are very valuable and will see growing use. Java is of course critical for portal infrastructure. Here we mainly focus on computing portals (Computing Portals, web) but make some remarks on education and training portals. Note that Java is used to build a computing portal even if this is not suitable (e.g. too slow) for the application code. In section 2, we give some background on the Object Web and multi-tier distributed information systems. Section 3 shows how to customize the general web technology for portals with some special attention to collaboration as a service. Section 4 makes general remarks about computing portals in three different application areas: Mesh Generation, Space data analysis and Earthquake science. In the final section, we go through some of the computing portals, we have built at Syracuse.

2 Technology Background

2. 1 Multi-Tier Architectures

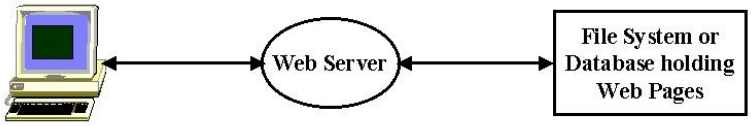

Fig 1: 3-Tier Computing Model illustrated by Basic Web Access

Modern information systems are built with a multi-tier architecture (Fox, 1998) that generalizes the traditional client-server to become a client-broker-service model. This is seen in its simplest realization with the classic web access of fig. 1, which involves 3 tiers – the browser runs on a client; the middle-tier is a Web server; the final tier or backend is the file

system containing the Web page. One could combine the Web server and file system into a single entity and return to the client-server model. However the 3-tier view is better as it also captures the increasingly common cases where the Web Server does not access a file system but rather the Web Page is generated dynamically from a backend database or from a computer program invoked by a CGI script. More generally the middle tier can act as an intermediary or broker that allows many clients to share and chose between many different backend resources.

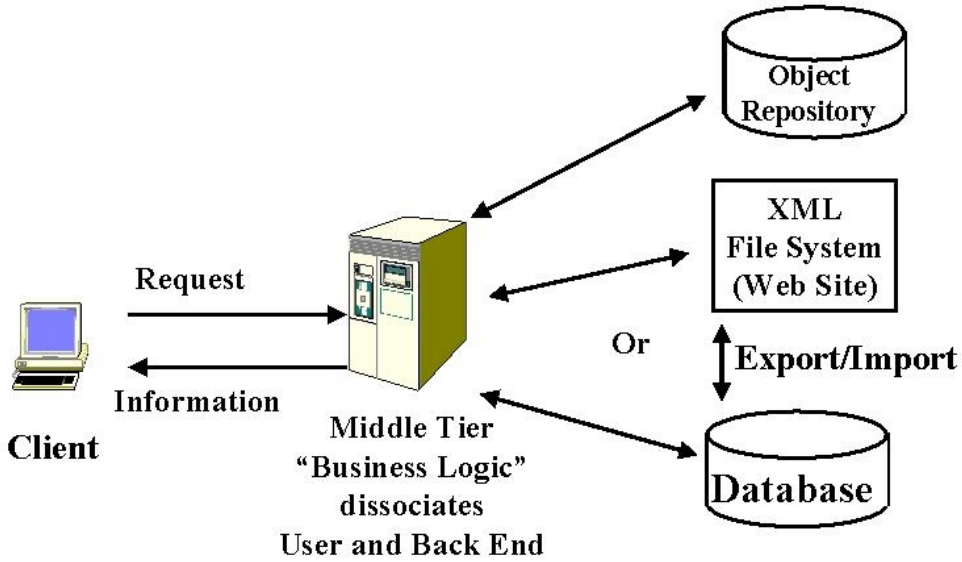

Fig.2: 3 Tier architecture supporting multiple persistent object models

This is illustrated in Figs. 2 and 3, which shows the 3-tier architecture with the two interfaces separating the layers. As we will discuss later the specific needs and resources of a portal will be expressed by metadata at these interfaces using XML technology.

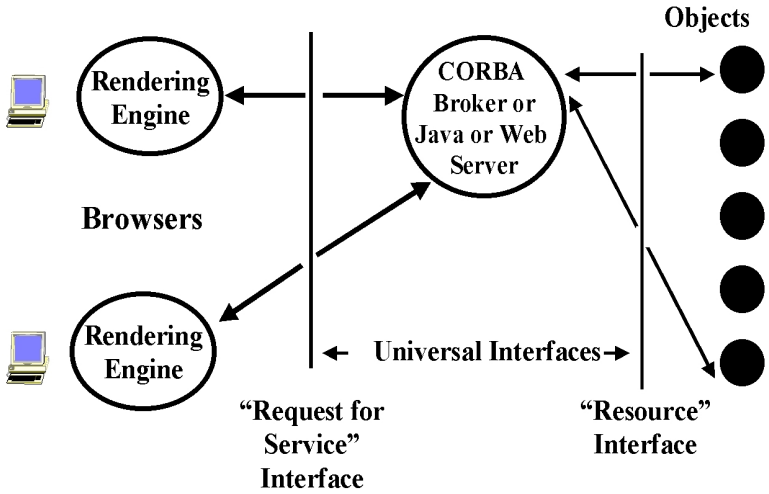

Fig 3: General Three Tier Architecture

This 3-tier architecture (often generalized to a multi-tier system with several server layers) captures several information systems which generalize the access to a web page in Fig. 1 and the invocation of a computer program from a Web or other client in Fig 4.

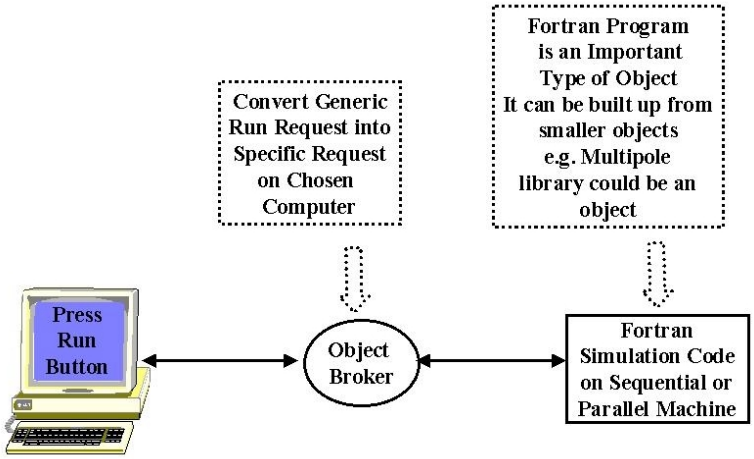

Fig 4: Simple Simulation formulated as a "distributed computational object"

The architecture builds on modern distributed object technology and this approach underlies the "Object Web" approach to building distributed systems. Let us describe this in the context of some familiar computing concepts.

2.2 Clients, Servers and Objects

A *server* is a free standing computer program that is typically multi-user and in its most simplistic definition, accepts one or more inputs and produces one or more outputs. This capability could be implemented completely by software on the server machine or require access to one or more (super)computers, databases or other information resources such (seismic) instruments. A *client* is typically single-user and provides the interface for user input and output. In a distributed system, multiple servers and clients, which are in general geographically distributed, are linked together. The clients and servers communicate with *messages* for which there are several different standard formats. *Events* are a special type of message typically with a time stamp and representing some state change. An (electronic) *object* is essentially any artifact in our computer system. As shown in Fig. 4, a computer program is one form of object while the best known distributed object is a web page. Fig. 2 illustrates database, XML and general object models.

In the example of fig. 1, a *Web Server* accepts HTTP request and returns a web page. HTTP is a simple but universal protocol (i.e. format for control information) for messages specifying how Web Clients can access a particular distributed object – the web page. Again a Database Server accepts a SQL request and returns records selected from database. SQL defines a particular "Request for Service" in the interface of fig. 3 but usually the messages containing these requests use a proprietary format. New standards such as JDBC (The Java Database Connectivity) imply a universal message format for database access with vendor dependent bridges converting between universal and proprietary formats. An Object Broker as in fig. 4 uses the Industry CORBA standard IIOP message protocol to control the invocation of methods of a distributed object (e.g. run a program). IIOP and HTTP are two standard but different protocols for communicating between distributed objects.

The Object Web signifies the merger of distributed object and web technologies, which is implicitly embodied in the discussion so far. There are four rather distinct but important Object standards. *CORBA* is the Industry Standard supporting objects in any language on any platform. New features in CORBA tend to be deployed relatively slowly as they have a cumbersome approval process and must satisfy complex constraints. *COM* is the Microsoft standard, which is confined to PC's but broadly deployed there and high performance. *Java* is

the software infrastructure of the web and although single language, the same object can be used on any computer supporting the Java VM. *XML* comes from the Web Consortium and will be briefly discussed below. It can be used to define general objects in an intuitive format illustrated in fig. 5.
The *Pragmatic Object Web* implies that there is no outright winner in the distributed object field and one should mix and match approaches as needed. For instance, CORBA Objects can use interfaces (as in fig. 3) defined in XML, with clients and servers programmed in Java, and with rendered displays using COM.
Distributed objects are the units of information in our architecture and we need to provide certain critical operating system *services* to support them. Services include general capabilities such as "persistence" (store information unit as a disk file, in a database or otherwise), and "security" as well as more specialized capabilities for science such as visualization.

2.3 Use of XML Extended Markup Language

We suggest using XML technology is used to specify both general and specific resources in portals. A good overview of the use and importance of XML in Science can be found in [SciAm] and we illustrate it below in Fig. 5, which specifies a computer program used in a prototype GEM (Earthquake) Computing Portal described later and shown in fig. 14.

```
<?xml version="1.0"?>
<!DOCTYPE application SYSTEM "ApplDescV2.dtd">
<application  id="disloc">
<target id="osprey4.npac.syr.edu">
 <status installed="Yes"/>
 <installed>
   <CmdLine command="/npac/home/webflow/GEM/JAY/dis2loc"  />
   <input>
     <inFile Path="/npac/home/webflow/GEM/JAY/" Name="disloc.output"/>
     <source Host="osprey4.npac.syr.edu" Path="/npac/home/Jigsaw/WWW/tmp"
          Name="disloc.out"/ >
   </input>
   <output>
     <outFile Path="/npac/home/webflow/GEM/JAY/" Name="simplex.input" />
     <dest Host="osprey4.npac.syr.edu"
           Path="/npac/home/webflow/GEM/JAY/simplex/" Name="s.in" />
   </output>
   <stdout Host="aga.npac.syr.edu" Path="/npac/home/haupt/webflow/history/"
          Name="job2001.out" >
   <stderr Host="aga.npac.syr.edu" Path="/tmp/" Name="haupt_job2001.err" >
 </installed>
</target>
</application>
```

Fig. 5: XML Used to describe the computational problem in the GEM computing portal

3. Portals in the Object Web

3.1 Basic Architecture

Portals have major impact in two commercially critical areas. First there are community portals like Yahoo, Excite, and Netscape Netcenter whose impact is illustrated by the $100B or so stock market valuation of Yahoo and AOL. Secondly there is the still growing use of

the portal concept to organize corporate information for customers and employees – an area termed Enterprise Information Portals (EIP). These are intended to integrate all information and related services – structured from databases, electronic mail, web pages, scanned paper – from a single user interface. Merrill Lynch has estimated that EIP technology will generate $14B in market revenues by 2002.

Portals can alternatively be looked at as a web technology implementation of a distributed system. Portals have certain common features; for a given area (community, the αβγ corporate EIP, the XYZ University Education Portal etc.) there are a collection of objects and services (operations on objects) which can be accessed from the portal (web page). One can often customize the portal functionality, choosing both, which objects to display and their parameters (which sports team score to display or which area weather map). Similarly the look and feel of the portal, background or index style, can be customized to get "my.portal". Both the basic portal objects and customization for each user must be stored persistently. The large size of the commodity portal market suggests that one should carefully architect portals to build on technologies and ideas from the large base markets and this strategy is outlined in fig. 6.

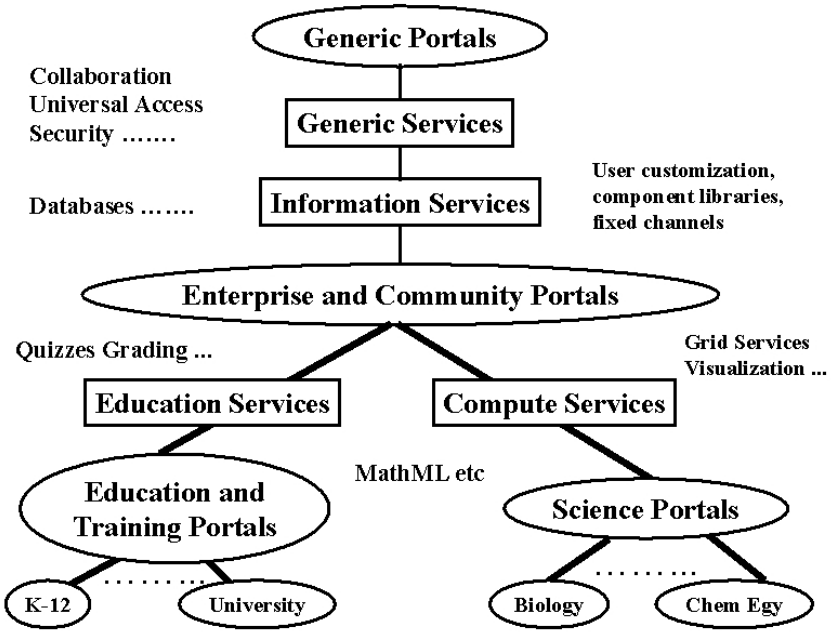

Fig. 6: Hierarchy of Portals and Their Technology

Re-use of portal technology requires careful definition of interfaces in the fashion of fig. 3. Unfortunately although the commercial world is moving with Internet time, some areas like security and even more so collaboration are not clear and so one must inevitably develop technology in areas that will later redone by the commercial whirlwind. Some application features (such as quizzes and homework for education, visualization and execution of jobs for computing) are special and can be safely studied in academia. The overall object model as described in section 2, is reasonably clear but even the area of portal authoring is still not addressed very well commercially and here again one must develop prototype solutions and expect commercial competition later. There are several interesting approaches to portals including iPlanet from Sun, portlets from Oracle, many XML/web-linked database systems, www.desktop.com, Ninja from UC Berkeley, WebSphere from IBM and e-Speak from Hewlett Packard. These address different aspects of portals with different architectures and

none has a clearly winning or complete approach. Thus here we will adopt a more general approach.

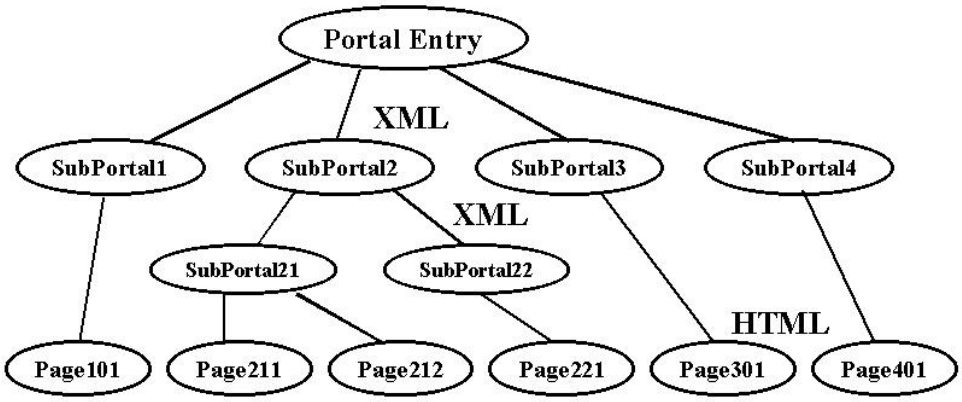

Fig. 7: Portal HTML/XML Structure

We will assume that whereas most web pages will be in HTML, these are the links accessed from portals. We can assume as shown in fig. 7, that the "control" (entry) pages of a portal can be assumed to be in XML and that this XML will define all aspects of a portal; what objects are in it and their possibly user customized properties. We will also assume that we will use XML to define the layout of the page in a fashion sketched below.

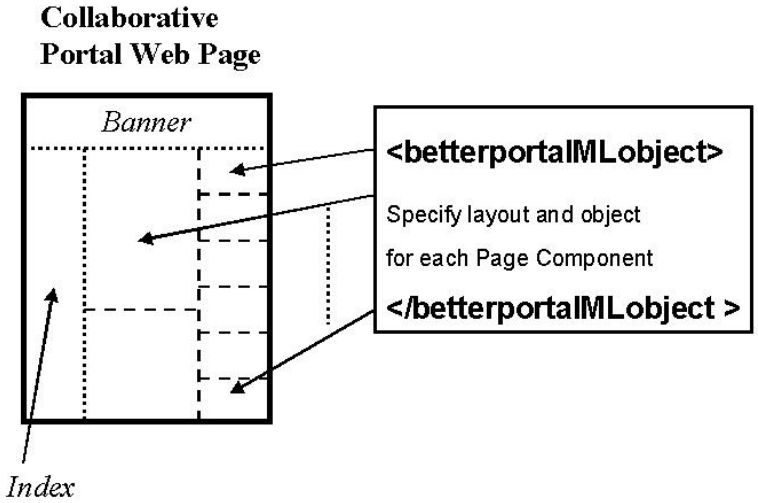

Fig. 8:A Web Page as a Collection of Collaborative Components

Maybe layout can be specified in some XML equivalent of the Java AWT layout classes with say a Grid layout specifying the classic multi-column structure of common portals today. We term this "betterportalML" above and assume some such standard will get developed. There is already a slightly specialized portalML defined and so we add the qualifier "better". We might find something like:
<portal name="" ><columns=3/><portalobject mode="forced" source=""><placeat layout="flow1" position="1" > .....
"betterportalML" would control placement of component objects on pages and specification of their parameters and location. So we might find
<portalparameter name="city" value ="Manchester" /><portalparameter name="sensor" value ="radar" />.
"betterportalML" will allow specification of your favorite nifty JavaScript or Java index and rendering preferences such as resolutions and "universal access", borders, backgrounds etc.

Portal building tools will exist for developers (architect template for portal i.e. differentiate my.pajava from my.Netscape) and users (specialize portal template). Such general standard based portal approaches will be serious competition for specialized portal systems such as WebCT for education.
The portal authoring tools will perhaps be designed like traditional PC desktop and application customization tools. Capabilities are organized hierarchically as toolbars and the user will have a GUI to allow the dragging of chosen components onto the portal page. Computing toolbars would include user profile, application area (here user would certainly customize), code choice (application specific), results, visualization (where "command" could be AVS), collaboration, programming model, (HPF, Dataflow specialized tools), resource specification, resource status etc. The same authoring tool would support commodity, computing and education "toolbars". Thus a university student could produce a single personal portal supporting his personal whims, university education and computational science research by choosing components from multiple toolbars.

3.2 Collaborative Portals

Collaboration corresponds to the sharing of objects with changes induced by one client, transmitted in some fashion to other clients. These changes are "events" and the transmittal corresponds to sending messages between clients with perhaps some sort of server intervention to route and archive events.

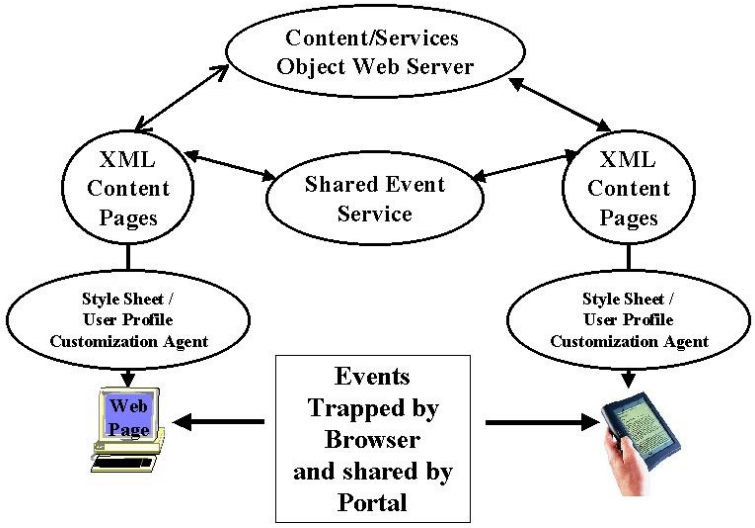

Fig. 9: Collaborative Portal Architecture

As shown above, events can be recognized on the browser (possibly in a handheld or other non PC client) and these need to be translated into a corresponding state change in the controlling server or XML page. This appears to be non-trivial and the importance of being able to link events between child and parent objects is important and should be a part of emerging W3C web document object and event standards. These events could be as trivial as a page scroll or a mouse movement; they could be a new parameter value in a form or a requested change in page URL. We can even include audio-video multi-media streams in this approach with events formed as perhaps a few hundreds of milliseconds of audio and video nuggets. The event model could include server events ("your job aborted", "remember class begins in 5 minutes") and a robust federated (between all participating clients and servers) event service underlies collaboration.

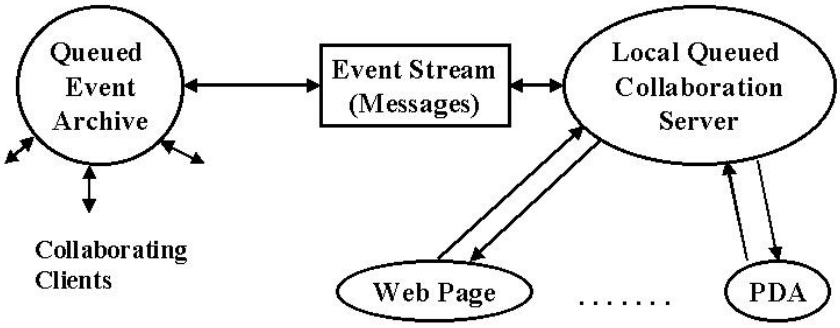

Fig. 10: Event Streams from Client to Archive

The event stream specifying the collaborative object state changes, becomes set of time stamped XML messages which we suggest should be queued and archived as in fig. 10 before dispatching to linked clients. This approach allows each client to choose between synchronous and asynchronous modes. The queued event can be delivered immediately or accessed later if so desired. This allows the system to support both real-time and "pager" modes of collaboration. As you file out of your airplane, you can rejoin your collaborative session – replaying missed events first with style sheets to format for your handheld client and then when you reach home, you can examine session in detail – either from archive or join the active session synchronously. As demonstrated by Beca from Syracuse University, one can use attributes in "betterportalML" to specify the collaborative structure of each object in your portal. This would specify both the allowed options – which collaborative modes are supported by the system – and the customized choice of each user.

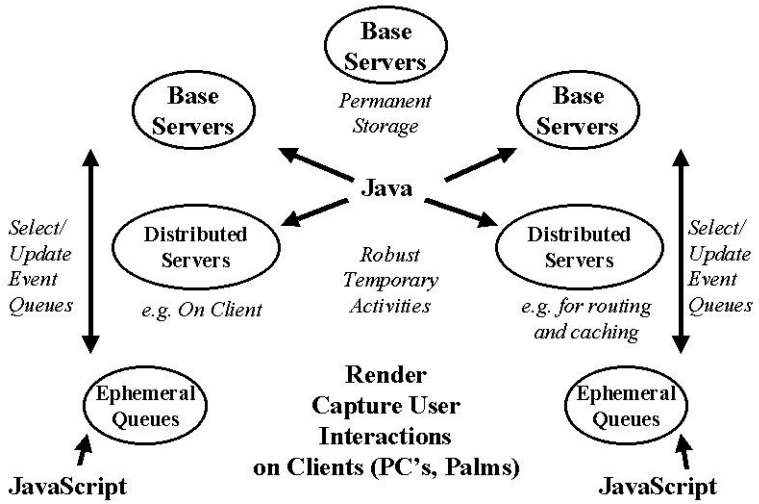

Fig. 11: Distributed Event Subsystem

Currently we are designing collaborative portals where we add an appropriate robust event service as in fig. 11 to one or more existing portal technologies. Here we view events as stored in a distributed database with appropriate caching to support high performance.

4: Architecture of Computing Portals

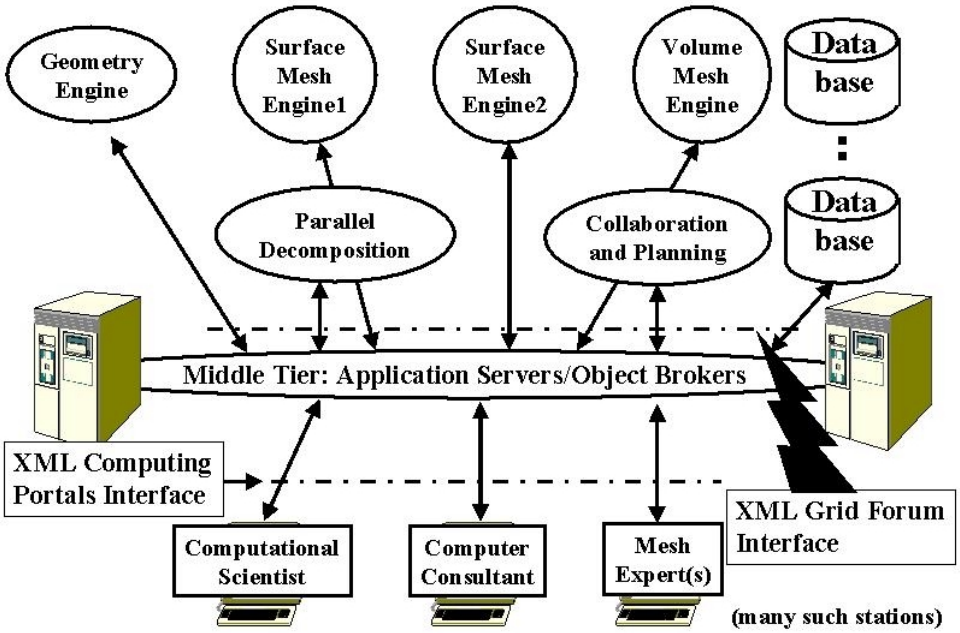

Fig. 12: A Portal for Mesh Generation

A Computing Portal – or equivalently a web-based Problem Solving Environment or PSE is an application that integrates access to the data, computers and tools needed for a particular computational science area (Computing Portals, web). It must exhibit several services including security, fault tolerance, object lookup and registration, object persistence and database support (as in EIP's), event and transaction services, job status (as in HotPage from NPACI and myGrid from NCSA), file services (as in NPACI Storage Resource Broker, support for computational science specific metadata like MathML and XSIL, visualization, application integration (chaining services viewed as backend compute filters), "seamless access" and integration of resources between different users/application domains, parameter specification service (get data from Web form into Fortran program wrapped as backend object), and finally collaboration. We illustrate in figs. 12 through 14 the functional architecture of conceptual portals in three areas – Mesh generation, Space data analysis and Earthquake Science.

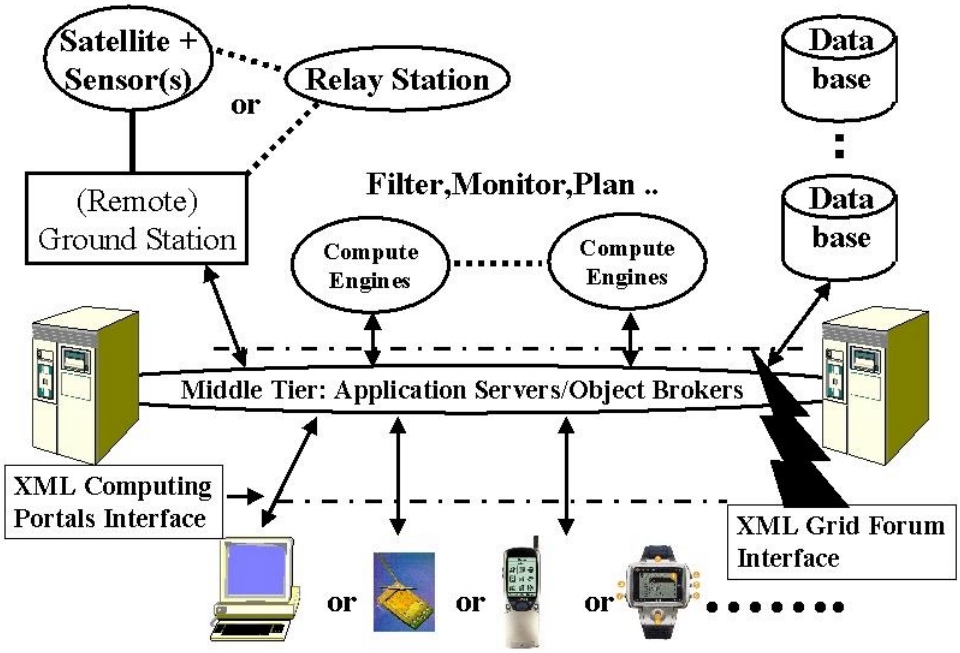

Fig 13: SMOP or Space Mission Operations Portal

In fig. 12, one emphasizes collaboration with experts as mesh generation is still hard to automate and we assume that will a distributed center that supports this function that underlies solution of partial differential equations from essentially any field. We have the same interfaces already introduced in fig. 3 in all portals. In fig. 13, we indicate the importance of multiple user interfaces as space missions will not wait and interaction with users can be needed at any time. Further we try to isolate access to the Space Internet as one service but otherwise use infrastructure (databases, compute servers) from less esoteric applications. Let us focus on the last case GEM (General Earthquake model) which is a portal supporting all aspects of earthquake science from real time interaction with data from the "big-one" to decade long theoretical studies of the underlying nonlinear systems (Fox, 2000). Most of the features of GEM (which we have studied quite deeply) translate to other areas such as those of fig. 12 and 13.

In GEM, everything is a "distributed object" whether it be a simulation on a supercomputer; the basic GEM Web pages; the notes from a field trip entered on a palm top; CNN real-time coverage of the latest earthquake or the data streaming in from sensors. GEM provides an integrated view of these diverse resources with XML definitions for the raw objects themselves and the data they produce. The services shown in fig. 14, from collaboration, security, object discovery, visualization and computer access, are generic to all computing portals.

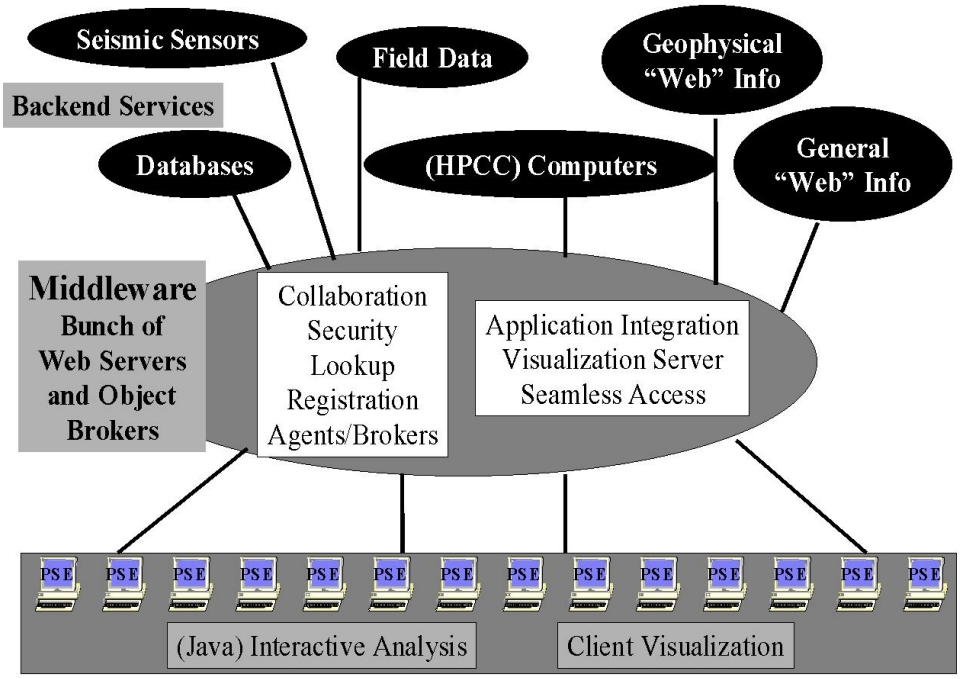

Fig. 14: 3 Tier Architecture of a GEM Computing Portal

Building GEM using the same approach and tools as other portals ensures the availability of these services. They will require customization as for instance there are many different visualization packages and each requires non trivial work to include in such a portal. Again collaboration corresponds to sharing distributed objects, and as discussed in section 3, this can currently only be automated for some objects. Many web pages can be shared using generic techniques illustrated in fig. 5 but sharing say the control and output of a general simulation can require quite a lot of custom modifications.

One needs to define the entities in the GEM environment as distributed objects and for computer programs this implies a rather arcane process termed "wrapping the program as a distributed object". Operationally this implies allowing a middle-tier server (the CORBA object broker or Java application Server) to be able to run the program on one or more machines, specify the input files and either specify output files or access them as streams of data in the fashion of UNIX pipes. Our strategy is to define all relevant properties of computer programs in XML as illustrated in fig. 5. These properties are used to generate either statically or dynamically the needed object wrappers. This approach requires the user specify what they know – the properties of their program, while the filter copes with the obscure syntax of each object model. Obviously this also allows one to support all object models – COM CORBA Java, by changing the filter and so one can adapt to changes in the commercial infrastructure used in the middle tier.

One must apply the XML object definition strategy to all entities in GEM; programs, instruments and other data sources and repositories. This gives the metadata defining macroscopically the object structure. However equally usefully, one needs to look at the data stored in, produced by or exchanged between these objects. This data is itself a typically a stream of objects – each an array, a table or more complex data structure. One could choose to treat the data at some level as an unspecified (binary) "blob" with XML defining the overall structure but detailed input and output filters used for the data blobs. As an example, consider the approach that an electronic news organization could take for their data. The text of news flashes would be defined in XML but the high volume multimedia data (JPEG

images and MPEG movies) would be stored in binary fashion with XML used to specify <IMAGEOBJECT> or <MOVIEOBJECT> metadata. Systematic use of XML allows use of a growing number of tools to search, manipulate, store persistently and render the information. It facilitates the linkage of general and specific tools/data sources/programs with clearly defined interfaces. This will help the distributed computing portal users to separately develop programs or generate data, which will be easily able to interoperate. More generally as shown in fig. 6, XML standards will be defined hierarchically starting with distributed information systems, then general scientific computing and finally application specific object specifications. For example GEM would develop its own syntax for seismic data sensors but could build on general frameworks like the XSIL scientific data framework developed at Caltech (Williams, 1998). XSIL supports natural scientific data structures like arrays and the necessary multi-level storage specification. Another example is MathML which provides XML support for the display and formulation of Mathematics. We can expect MathML to be supported by tools like Web Browsers and white boards in collaborative scientific notebooks and allow one to enhance theoretical collaboration in GEM. There will for instance be modules that can be inserted into applications for parsing MathML or providing graphical user specification of mathematical formulae. One can also use MathML in high level tools allowing specification of basic differential equations that are translated into numerical code. This has been demonstrated in prototype problem solving environments like PDELab but have so far not had much practical application (Houstis, 1998). Greater availability of standards like MathML should eventually allow more powerful interchangeable tools of this type. Finally we can mention a set of graphical XML standards such as X3D (3 dimensional objects) SVG and VML which are vector graphics standards, which can be expected to be important as basis of application specific plot and drawing systems.

5 Building Computing Portals

Here we discuss some existing experience from my research in integrating such objects and tools that manipulate them into an overall environment. The Syracuse team led by Tom Haupt (Akarsu, 1999) has built several exemplar computing portals for both the NSF and DoD HPCMO (High Performance Computing Modernization Office) supercomputer centers. In fig. 15, we show some useful tools including a first cut at a “wizard” that helps produce the distributed object wrappers described in section 4.

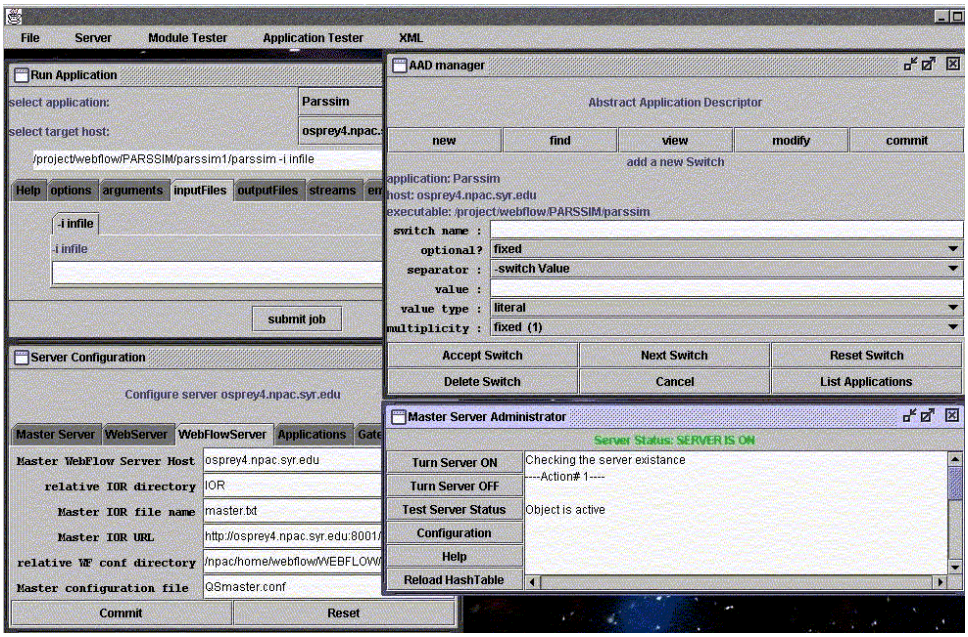

Fig. 15: This illustrates several components of the WebFlow system from Syracuse. (Counter-clockwise from bottom-left): The Server Configuration and Master Server Administrator are WebFlow administrative tools that allow one to configure and monitor the middle tier servers. AAD manager (Abstract Application Descriptor) is a tool to incorporate new applications to the system: the interface shown simplifies creation of an XML definition of the application.
This definition can be used for dynamical creation of front-end application interfaces (window "Run Application").

This AAD (Abstract Application Descriptor) can be extended to allow specification of all needed input parameters of an application, with an automatic generation of input forms respecting default values and allowed value ranges. The object wrappers of an application should not only invoke the code and allow parameter specification but have built in description/help systems.
One major Syracuse Computing Portal was built for the Landscape Management System (LMS) project at the U.S. Army Corps of Engineers Waterways Experiment Station (ERDC) Major Shared Resource Center (MSRC) at Vicksburg, MS, under the DoD HPC Modernization Program, Programming Environment and Training (PET). The application can be idealized as follows.
A decision maker (the end user of the system) wants to evaluate changes in vegetation in some geographical region over a long time period caused by some short term disturbances such as a fire or human's activities. One of the critical parameters of the vegetation model is soil condition at the time of the disturbance. This in turn is dominated by rainfall that possibly occurs at that time. Consequently as shown in fig. 16, the implementation of this project requires:

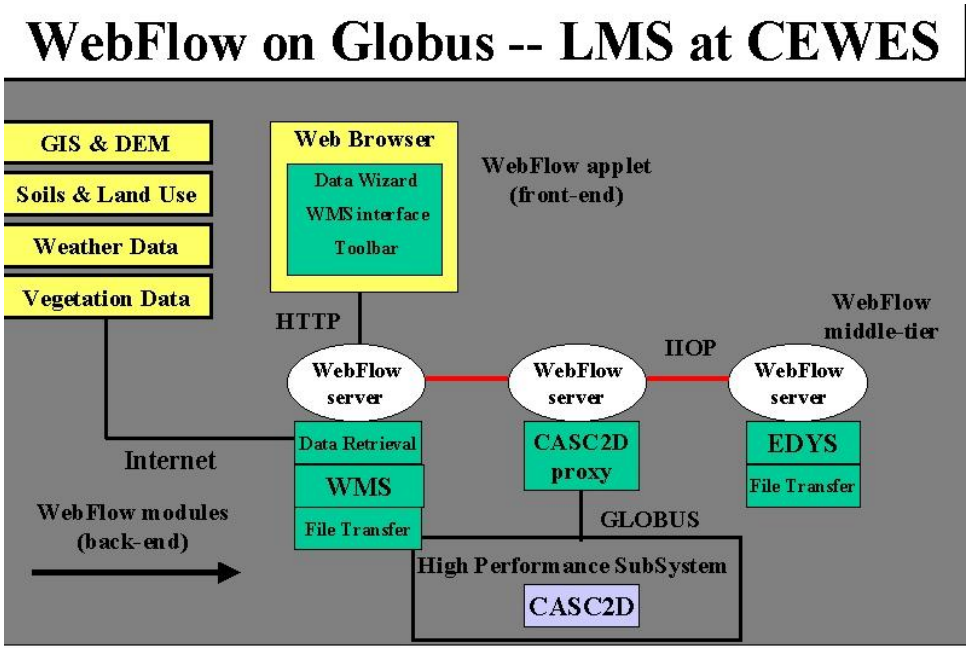

Fig 16: 3 Tier Architecture of DoD "Land Management" Application whose front end is shown in fig. 17

- Data retrieval from remote sources including DEM (data elevation models) data, land use maps, soil textures, dominating flora species, and their growing characteristics, to name a few. The data are available from many different sources, for example from public services such as USGS web servers, or from proprietary databases. The data come in different formats, and with different spatial resolutions.
- Data preprocessing to prune and convert the raw data to a format expected by the simulation software. This preprocessing is performed interactively using WMS (Watershed Modeling System) package.
- Execution of two simulation programs: EDYS for vegetation simulation including the disturbances and CASC2D for watershed simulations during rainfalls. The latter results in generating maps of the soil condition after the rainfall. The initial conditions for CASC2D are set by EDYS just before the rainfall event, and the output of CASC2D after the event is used to update parameters of EDYS and the data transfer between the two codes had to be performed several times during one simulation. EDYS is not CPU demanding, and it is implemented only for Windows95/98/NT systems. On the other hand, CASC2D is very computationally intensive and typically is run on powerful backend supercomputer systems.
- Visualization of the results of the simulation. Again, WMS is used for this purpose.

One requirement of this project was to demonstrate the feasibility of implementing a system that would allow launching and controlling the complete simulation from a networked laptop. We successfully implemented it using WebFlow middle-tier servers with WMS and EDYS encapsulated as WebFlow modules running locally on the laptop and CASC2D executed by WebFlow on remote hosts. Further the applications involved showed a typical mix of supercomputer and computationally less demanding personal computer codes. LMS was originally built using specialized Java Servers but these are now being replaced by commercial CORBA object brokers but in either case the architecture of fig. 16 is consistent with the general structure of fig. 3.

For this project we developed a custom front-end shown in fig. 17, that allows the user to interactively select the region of interest by drawing a rectangle on a map. Then one could select the data type to be retrieved, launch WMS to preprocess the data and make visualizations, and finally launch the simulation with CASC2D running on a host of choice.

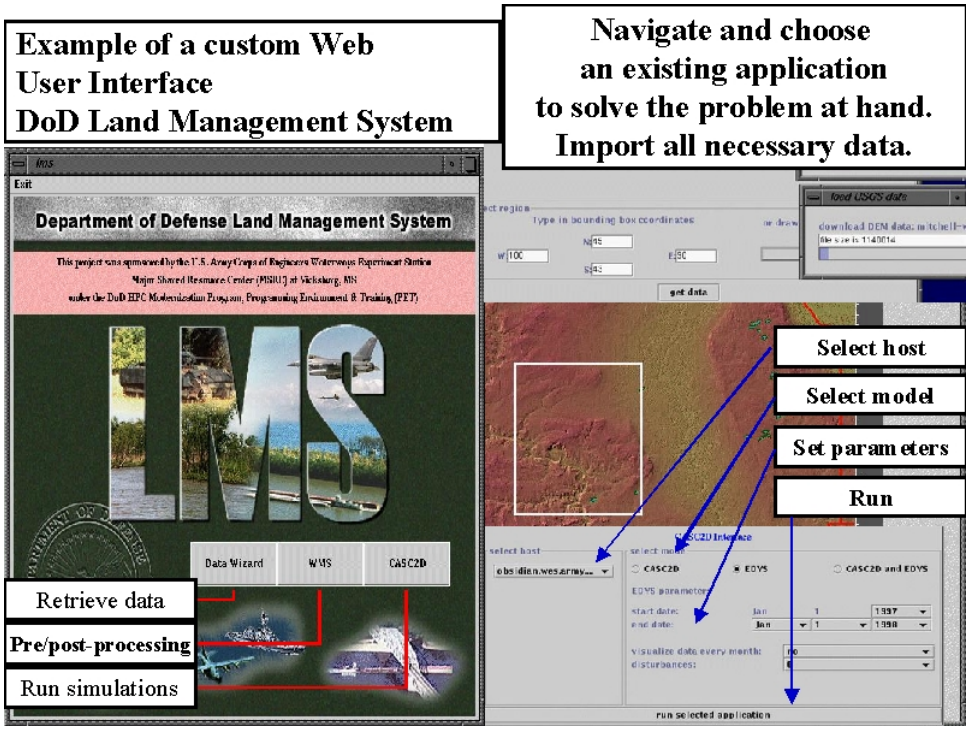

Fig 17: Example of a Web Interface for a "Land Management Problem Solving Environment" built by Syracuse for the Department of Defense ERDC Laboratory in Vicksburg, Ms.

Our use of XML standards at the two interfaces in fig. 3, allows us to change front end and middle-tier independently. This allowed us the middle tier upgrade described above which will bring security and "seamless access" capability to our computing portals.

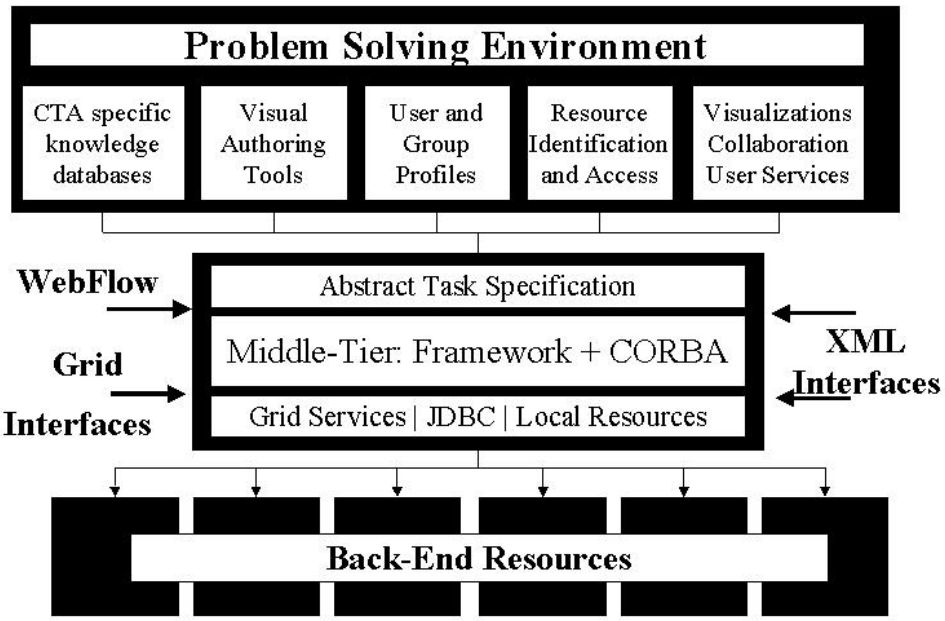

Fig. 18: Architecture of Current Syracuse WebFlow-based Computing portals

Seamless access is an important service, which is aimed at allowing applications to be run on an arbitrary appropriate backend. Most users wish their job to just run and usually do not mind what machine is used. Such seamless capability is rather natural in the architecture of figs. 3 and 18. Essentially the front end defines the "abstract" job passing its XML specification to a middle tier server. This acts as a broker and instantiates the abstract job on one of the available backend computers. The middle-tier backend XML interface is used to specify both the available machines sent to the servers and the "instantiated job" sent from the servers. We build the back end mapping of jobs to machine using the important Globus technology (Globus, web). Globus is a distributed or meta-computing toolkit providing important services such as resource look-up, security, message passing etc.

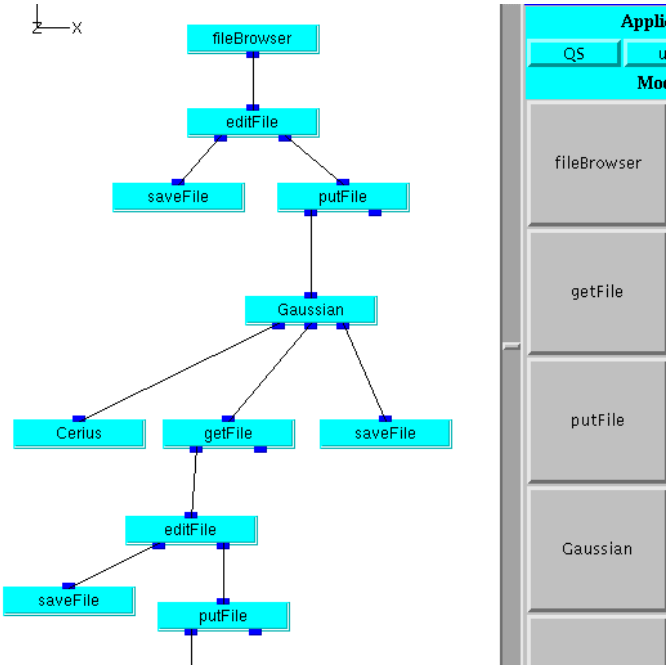

Fig 19: Fragment of WebFlow Composition Tool linking modules in a Quantum Simulation (Chemistry) application

One can view WebFlow as a "kit" of services, which can be used in different ways in different applications. In fig. 19, we show another capability of WebFlow, which supports the ability to compose complex problems by linking different applications together with data flowing between different modules chosen from a palette. This WebFlow service is supported by a Java front end and a middle tier service matching abstract modules to back end computers and supporting the piping of data between the modules. The space data portal of fig. 13 could use WebFlow like services to concatenate various filters needed in data analysis applications.

References

(Akarsu, 1999) Erol Akarsu, Geoffrey Fox, Tomasz Haupt, Alexey, Kalinichenko, Kang-Seok Kim, Praveen Sheethaalnath, and Choon-Han Youn, Using Gateway System to Provide a Desktop Access to High Performance Computational Resources, Proceedings of HPDC-8 Conference, Redondo Beach Ca., Aug 3-6, 1999, IEEE Press.
http://www.osc.edu/~kenf/theGateway/ and http://www.npac.syr.edu/users/haupt/WebFlow/

(Bosak, 1999) Jon Bosak and Tim Bray, XML and the Second-Generation Web, Scientific American May 99, http://www.sciam.com/1999/0599issue/0599bosak.html

(Computing Portals, web) Computing Portals Community Activity
http://www.computingportals.org/

(Fox, 1998) Fox, G. C., and Furmanski, W. "High Performance Commodity Computing," chapter in "The Grid: Blueprint for a New Computing Infrastructure", C. Kesselman and I. Foster, editors, Morgan Kaufmann, 1998

(Fox, 2000) Fox, G.C. and Hurst K., Computational Science and Information Technology meets GEM, a chapter in AGU monograph on Physics of Earthquakes, edited by John Rundle and published by AGU in 2000.

(Globus, web) Globus Metacomputing Toolkit, home page: http://www.globus.org

(Houstis, 1998) E. N. Houstis, J. R. Rice, S. Weerawarana, A. C. Catlin, P. Papachiou, K. Y. Wang, and M. Gaitatzes. PELLPACK: A Problem Solving Environment for PDE Based Applications on Multicomputer Platforms. ACM Transaction on Mathematical Software, 24:30-73, 1998. See http://www.cs.purdue.edu/research/cse/pdelab/pdelab.html

(Williams, 1998) R.D. Williams, Caltech http://www.cacr.caltech.edu/SDA/xsil/index.html