# Prototyping a WebVM based WebFlow Environment for Distributed Computing and Visual Programming

*Wojtek Furmanski and Geoffrey Fox*

Northeast Parallel Architectures Center, Syracuse University, Syracuse, NY
{furm, gcf}@npac.syr.edu

To be presented at the IEEE Dual-Use Technologies & Applications Conference, Syracuse, NY, June 1996

March 15, 1996

## Summary

We review recent NPAC activities in the area of computational Web technologies. Our general strategy is to develop distributed computing environment on top of the Web. Subsequent design and development steps and technologies such as WebTools, WebVM and WebFlow are reviewed and put in the context of dual-use applications for command & control, telemedicine and distance education.

## Contents

# 1 Overview of NPAC Web Activities

At NPAC, we recognized early (in summer '94) that the software technologies of the World-Wide Web contain a promising potential for building new generation high performance computing environments. We call this concept *WebWindows* [FoFu95] to point out that it does not refer to any specific operating/windowing system (such as X Windows under UNIX or MS Windows on PC) and is developed collectively as a set of open standards and public software modules by the Web community at large.

## 1.1 WebTools – Personal Assistant

Our first contribution, developed in early '95, was packaged as a set of *WebTools* [Furm95a], based on CGI/Perl extended 'personal' web servers. The core toolkit includes Document Tree Navigator, Manager and Editor, and offers the base functionality required for interactive authoring and exploration of the document tree on the WebTools server. All tools are implemented as Perl packages and associated with PDA-style forms based front-ends. Manager supports creation and destruction of files and directories in the server document tree. Each file is created with built-in navigation bar, containing a set of control button/HTML links which are associated with suitable CGI actions at the server side. Typical actions include moving left or right in the current directory, traversing the directory tree, or launching the Editor or the Manager tool. Editor supports on-line HTML authoring for each page, maintained by a WebTools server.

WebTools were used extensively in '95 at NPAC both in the PDA mode (e.g. to develop technology information pages) and in the collaboratory mode by a group of Web enthusiasts exploring various aspects of interactive Web technologies.

We also started the development of WebMail in the WebTools framework. This task, however, turned out to be rather complex and we realized that several back-end operations, critical for reliable production version can be more efficiently implemented using commercial RDBMS support. Oracle engine for e-mail and news-group was in fact developed in another NPAC project [Cheng95] and we are now addressing the merging of WebTools and Oracle components of the mail system.

## 1.2 WebWork – Distributed Computing

Also in early '95, we developed in collaboration with Boston University and Cooperating Systems Corporation the concept of *WebWork* [FFCRC95] – a distributed computing environment based on a mesh of computationally extended Web servers. We realized that the HTTP protocol can be used to implement the point-to-point message passing between servers. We also observed that dataflow offers a natural high-level programming paradigm in this framework. We called it *WebFlow* and noticed that Java programming language could be used for developing the visual authoring tools for editing WebFlow networks, or *compute-webs*.

A simple proof of the concept WebWork application – parallel search – was prototyped in terms of a mesh of Web servers [Bala95]. A more advanced application in the area of RSA factoring on a 'World-Wide Computer' was developed by our Boston/CSC collaborators, productized at NPAC, presented at Supercomputing '95 and awarded within the Teraflop Grand Challenge contest [Cowie95].

## 1.3 WebFoil – Distance Education

In spring/summer '95, Java programming language became the driving force of new Web developments. Also at this time, Netscape initiated a major internal effort in the area of web tools, and we decided to refocus our activities on exploring the potential of Java. Following the WebWork spirit, we prototyped, using alpha Java, the front-end for the visual compute-web editor and for a sample WebFlow application — Project Manager [Vish95a].

We also observed that HotJava can be customized and extended [Furm95b] to provide an appealing tool for electronic presentation and distance education, *WebFoil*.

Our Java prototypes were successfully demonstrated during the two Web tutorials on "Interactive Web Technologies for Distance Education", presented by NPAC in '95: at HPDC4 [FFP95b] in June and at Supercomputing'95 in December [NPAC95].

The most recent version of WebFoil is based on multiframe JavaScript under Netscape 2, supports Real Audio and offers an integrated presentation framework as well as an extensive educational material, covering a broad spectrum of Web and HPCC technologies [Fox96].

## 1.4 WebSpace – Collaboratory

We are also collaborating with Argonne National Laboratory in the area of Web technologies and we developed the concept of WebSpace [FFP95a] as a Web based 'plug-in' to ANL LabSpace – a collaboratory environment to support scientific research, remote experimentation and shared teleoperation of advanced equipment such as ANL electron microscope. WebSpace will provide Web access and insight into the current advances of science for educational and small business communities.

WebSpace technologies will include support for *CGI server* – a collaboratory server cooperating with the Web server via CGI and handling tasks such as session control or document caching. An early prototype of such a collaboratory server was developed in the alpha Java framework [Vish95b]. The model included Java standalone application, acting as a multiuser network server, and a set of HotJava clients. Participants of such a chat space were mapped on individual threads inside the Java servers. A simple textual whiteboard was developed in this framework as well as simple distributed games such as tic-tac-toe. Another prototype of a Web based chat tool was constructed more recently using Oracle database server as collaboratory engine and Netscape client pull mechanism [Cheng96]. We are now addressing the issue of combining Java and Oracle technologies (e.g. using JavaSoft's JDBC) for the collaboratory server development.

## 1.5 The Bridge and Its Dual Use

We are also working with Dave Warner on Web implementation of the Bridge [WaBa95] based Distributed Medical Intelligence. The bridge topology includes three generic components: "points of need" (or 'care portals' i.e. healthcare customers), "points of expertise" ( or 'docking stations', i.e. healthcare providers) and the intelligent switchboard (i.e. the bridge itself), providing suitable optimized routing, and offering information and interactive consulting services for the participants

We realized the a similar bridge topology can be found in the WebSpace/LabSpace architecture with the 'points on need' represented by schools or small businesses and handled by the WebSpace plug-in, with the "points of expertise' represented by scientists, and with the bridge itself provided by the ANL LabSpace.

Our major current activity which again follows the bridge architecture is Collaborative Interactive Visualization project with Rome Laboratory. Here the point of need/expertise is represented by troops/commanders and the intelligent switchboard is the electronic command and control center. We are currently developing component technologies and application modules to be integrated within a wargame scenario of a hypothetical missile attack.

We are also exploring the synergy od dual-use between the command & control and telemedicine bridge architectures and we use healthcare as the initial pilot/testcase domain. In our *WebMed* activities, we currently focus on a pilot project *CareWeb* for pediatric nursing triage, developed by a collaboration including NPAC, Syracuse School District, Syracuse University College of Nursing and SUNY Pediatrics Department. Points of need are represented by nursing stations, installed in selected schools, and using web browsers to access and operate on patient record databases. Nurse practitioner in the Nursing College (the bridge) is consulted to confirm the nurse diagnosis, and can in turn consult a pediatrician (point of expertise) in some more complex cases. Typical collaboratory actions include (WebTools based) shared development of the patient assessment form by nurse, nurse practitioner and pediatrician, and (chatboard based) interactive consulting support for the bridge and physician stations.

# 2 WebVM – Web Virtual Machine

So far, we reviewed our previous prototyping activities in the area of computational extensions of Web technologies and we outlined a set of large scale Web based distributed applications such as command & control, distance education and telemedicine, which we start to address at NPAC. In this chapter, we discuss in more detail our planned underlying infrastructure, *WebVM*, which is expected to integrate our current prototypes and set the framework for the continued development of larger scale applications.

## 2.1 Computational Web Concepts

The base support for the computational extensions of the Web is already present as part of the standard model in the form of the CGI protocol. These CGI extensions are usually exploited in conventional client-server configurations, e.g. for dynamic document generation, handling forms or accessing database servers. Typically, a CGI process receives a set of arguments from the client, performs some computation at the server side, and returns a MIME object to the client.

More elaborate services, which start to appear recently, would request an URL of the input MIME object from the client domain, fetch this object to the server domain, perform a suitable transform or filter and return it to the client as on output MIME object. Popular examples include AutoCAD DXF to VRML file convertors, remote image editing and improvement services, or remote VRML authoring kits.

A natural next step could include combining such services, e.g. by dataflow chaining. Consider for example a DXF to VRML conversion service and VRML authoring service, operated by two different providers at two different geographic locations. A Web user who owns a DXF files and wants to convert it to VRML and modify before saving, could do it most efficiently by linking the converter output to the editor input, passing URL of the DXF file to such composite service, or *compute-web*, edit the on-the-fly converted VRML file and finally save it at the client side, bypassing all intermediate formats and manual manipulations on temporary files.

At the moment, such linkage of Web services is impractical or just impossible since there is no standard methodology for publishing and connecting individual computational modules. The goal of *WebVM* is to provide a framework that would enable such interoperability. Below, we discuss our design which builds on the previous WebWork concepts and takes into account the most recent developments in the Web software industry.

In WebWork, we described the HTTP+CGI mechanism for server-to-server communication and we indicated how to generalize and improve this approach in terms of the next generation more dynamic Web server technologies. These technologies become available today from several vendors, most notably from Netscape and JavaSoft. In the following, we refine our base minimal design and we address the the issue of interoperability with the coming commercial products.

## 2.2 HTTP+CGI – Minimal Model

Minimal WebVM is given by a collection of CGI extended Web servers, typically with an independent server running on each node of a workstation cluster. The CGI extension includes a library of system CGI modules which are provided as part of the WebVM distribution, and a set of custom, user/application provided CGI modules. The overall computational paradigm is given by dataflow with user CGI processes acting as computational nodes/modules, and with the system CGI processes providing the required system control and management functions.

Each module receives a set of input MIME objects, performs an arbitrary local computation, and returns a set of output MIME objects. Both input and output objects are URL-addressable, i.e. they reside as MIME documents in a server document tree, and can be transferred between servers during the dataflow actions.

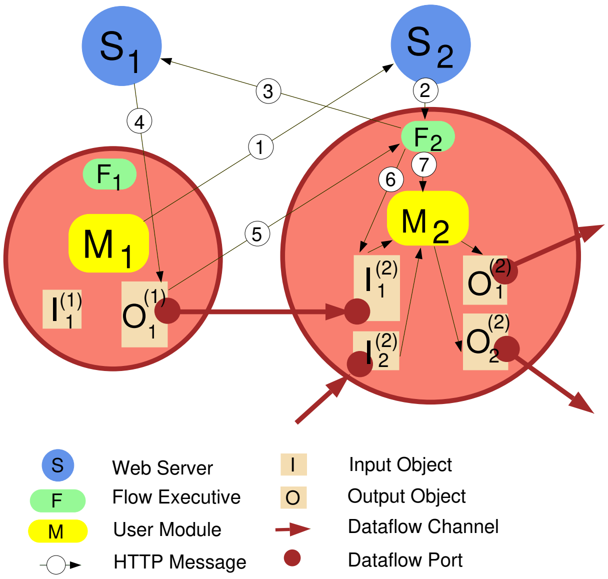


Fig. 1: Minimal model for HTTP+CGI based Server-to-Server Communication discussed in Section 2.2

Each module reads input files from and write output files to the local disk. Server-to-server object transfer is implemented via the HTTP protocol and carried out by the system CGI module called *Flow Executive*. Fig, 1 illustrates the steps involved in transferring the output object $O_1^{(1)}$ created by module $M_1$ from the server $S_1$ document tree to the server $S_2$ document tree, where it is stored as the input object $I_1^{(2)}$, ready to be used by the module $M_2$. The transfer starts after the module $M_1$ finished its computation and wrote $O_1^{(1)}$ to the local disk. After completing the write action, $M_1$ opens connection to $S_2$ and sends a POST message to the Flow Executive $F_2$, passing the current URL of stored $O_1^{(1)}$. $F_2$ sends the GET message to $S_1$, retrieves $O_1^{(1)}$, and stores it as $I_1^{(2)}$. Next, $F_2$ starts module $M_2$ in the background and exits. At this point, $M_1$ can also exit and all connections between servers $S_1$ and $S_2$ are closed. Module $M_2$ reads its input files $I_1^{(2)}$, $I_1^{(2)}$, performs local computation, writes its output files $O_1^{(2)}$,

$O_2^{(2)}$, and notifies the Flow Executive processes in the destination servers.

Fig. 1 illustrates only two neighboring nodes $M_1$, $M_2$ and one arc ($O_1^{(1)} \rightarrow I_1^{(2)}$) of a larger computational graph called *compute-web*. A given web server can host and handle modules from several compute-webs, belonging to different users. Local connectivity information is stored in configuration files in each server. Initial or idle state of a WebVM involves a set of (always) running Web servers which act as network daemons, and no other WebVM-specific active CGI processes. Dataflow is typically initiated by a browser which activates the initial module, e.g. $M_1$ in Fig. 1, via the usual URL addressing mode and the CGI convention. The subsequent $M$ processes are activated whenever the state of any of their input objects changes as a result of some previous dataflow action, and the $F$ processes are activated upon each object transfer request, issued by the $M$ processes.

A prototype implementation of the minimal model for the server-to-server communcation descibed above has been recently accomplished at NPAC [Haupt96].

## 2.3 Management Modules

The minimal model in Fig. 1 contains only one system module (Flow Executive) per node. More realistic version of WebVM will contain additional management modules. For example, the minimal model does not specify how the compute-web configuration files are being created, maintained and updated. For a small, testcase compute-web, these files can be created manually by a developer. Production level WebVM will be typically associated with some high level compute-web editing tools such as discussed in Chapter 3, WebFlow. To support automated compute-web editing, we may introduce *Link Manager* system module which will be notified by the client side authoring toolkit and will respond with the corresponding update of the compute-web configuration files. These configuration files require some management support as well to assure atomicity and concurrency control in the multi-user environment. Such responsibility could be assigned to *Configuration Manager* module which offers interface to a relational database that maintains the compute-web connectivity information.

The minimal model in Fig. 1 doesn't also specify the mechanism for assigning user modules to physical hosts in a workstation cluster. Again, for small testcase compute-webs, such placement could be hardwired in a manual mode, but a realistic production system will require *Resource Manager* module to assure fault tolerance, static resource allocation, and dynamic load balancing via module mobility. We are currently exploring several existing Cluster Management Systems (CMS) [Baker95] and selecting the most adequate solutions to be adapted to and plugged into the resource management sector of the WebVM.

## 2.4 Recent Web Developments

Minimal model is built on top of the classic Web servers which are stateless, whereas WebVM requires module state information (such as the current values of all input objects) to be available in the runtime. Minimal model implements the module state in terms of the MIME files, stored on the local disk. This, combined with the need for new socket connections to establish each dataflow channel, results in high latency of our server-to-server communication model. A series of the new products, coming from the Web software vendors such as Netscape or JavaSoft, open up new, better design and higher performance implementation avenues for the WebVM architecture.

A high performance implementation of the message passing in Fig. 1 can be realized in terms of auxiliary servers (referred to as *CGI servers* as in Section 1.4), present in all WebVM nodes, cooperating with Web servers and supporting object and configuration caching. In the minimal implementation of Fig 1., two large blobs represent document tress associated with servers $S_1$ and $S_2$, with modules as heavy–weighted processes and objects as MIME files. In the CGI server based implementation, these blobs become object-oriented, multithreaded servers with modules represented as threads and objects as class instances, cached in the dynamic memory.

Several new advanced Web technologies coming soon from the software vendors could be employed to develop such a dynamic compute-server support. We list here and discuss briefly a few most promising alternatives.

**Java Applets** Applets represent currently the most popular way of using Java. Their role in developing WebFlow visual editors is discussed in Section 3.1. However, applets can also be employed as a mechanism for easy download and moving selected WebVM modules written in Java from the server to the client domain.

**Java Applications** Java supports networking, multithreading and object-oriented programming, and hence it offers a natural implementation platform for the CGI servers discussed in Section 1.4. A Java based CGI server approach is particularily useful for WebVM modules also written in Java.

**Distributed Java** Development of Java based CGI server is most convenient in a distributed Java programming environment such as offered by the E language from Electric Communities or the RMI (Remote Method Interface) from JavaSoft.

**Java/Database** There are some analogies between Java interfaces to RDBMS and to generic computational modules in WebVM model. Java/datbase interface models such as dbKona/T3 server by WebLogic or JDBC (Java Database Connectivity) by JavaSoft can be used as design templates for Java based CGI server and also for developing specialized system modules such as the Configuration Manager.

**Java HTTPD** An extreme Java-centered approach would view the whole computational structure in a WebVM node as a single Java server, offering both the HTTPD capabilities and the server-to-server message pasing layer. Early prototypes of Java HTTPD are already available on the Web. Java HTTPD is a promising candidate for the eventual Web server programming model but is unlikely to dominate in the near term the current installation base of NCSA HTTPD servers.

**HPJava** Parallel Compiler Runtime Consortium (PCRC) led by NPAC addressed recently the role of Java in the high performance computing domain. Several concepts were contributed to the current position paper, including our previous work on MOVIE/HPFI model for distributed interpreted high performance computation. HPJava might possibly result in the high performance version of distributed Java, and hence a convenient vehicle for high performance version of the WebVM.

**Web/CORBA** World-Wide Web Consortium plans to include CORBA into the current set of Web standards and to provide a set of CORBA based Web binders to various object-oriented interpreters. This work could greaty enhance the interoperability of WebVM in the object domain.

**Netscape FastTrack Servers** Netscape announced recently the new generation servers, with the integrated support for email, newsgropus, collaboratory, Java, JavaScript and LiveWire. This model, when open enough, could provide a foundation for WebVM or in fact offer already the full solution to the challenge of Web based distributed computing we are tying to address here.

## 2.5 Portable Module API

In view of the coming Web industry offerings, a unique technology selection and detailed WebVM specification is a difficult task at the moment. Two possible strategies could be: a) select the best software platform, e.g. the latest server model from Netscape or JavaSoft and build WebVM in this framework, or b) seek for a maximally portable and scalable design, which must be then given by the common denominator of various available and coming soon approaches. We decided to follow the option b) which seems to be more safe given the current Web technology 'soup', has larger near term installation base, and is also more natural in the spirit of abstract, platform and vendor independent Virtual Machine concepts.

Our main design objective is to provide a framework which would a) allow each W3C compliant Web server to participate in WebVM computations, and b) provide hooks for more advanced servers to exploit their advantages. The Virtual Machine concept provides a natural methodology to satisfy both constraints. The essential point is the distinction between module API, i.e. "what the user sees" and the WebVM implementation layer. We already described in Section 2.2 the specific HTTP+CGI based implementation of the minimal model and pointed out that Java server implementation would look quite different. However, a generic WebVM user does not usually see or need to care about many of these details. Users develop application specific modules using their favored languages (C/C++, Fortran, COBOL, ADA, Java etc.) and they follow the dataflow metaphor. The minimal module API is built around the abstraction of a named dataflow port (heavy dot in Fig. 1) and two function calls: *inp_handle = port_read(port_name)* and *port_write(port_name,out_handle)*. For example, module $M_1$ would execute *port_read* twice to receive handles for input objects $I_1^{(2)}$, $I_1^{(2)}$, perform the computation, then execute *port_write* twice to pass handles for the output objects $O_1^{(2)}$, $O_2^{(2)}$ to the system layer, and exit. This minimal API is identical for various WebVM implementation modes, be it pure CGI, Java or Netscape/LiveWire. Various sites publishing their WebVM compliant modules can select their favored implementation strategies and in fact a given Intranet site can contain a mixture of implementation techniques which are most adequate for individual application needs.

## 2.6 Scripted Extensions

Conventional dataflow model is most useful for possibly heterogeneous but topologically static distributed applications. However, several relevant scientific computing and information processing problems require dynamically varying topologies. For example, Adaptive Mesh Refinement (AMR) algorithm for hyperbolic PDEs generates dynamically new transient tasks for refined grids, and a WebAMR programmer would like to distribute such processes efficiently over the WebVM nodes.

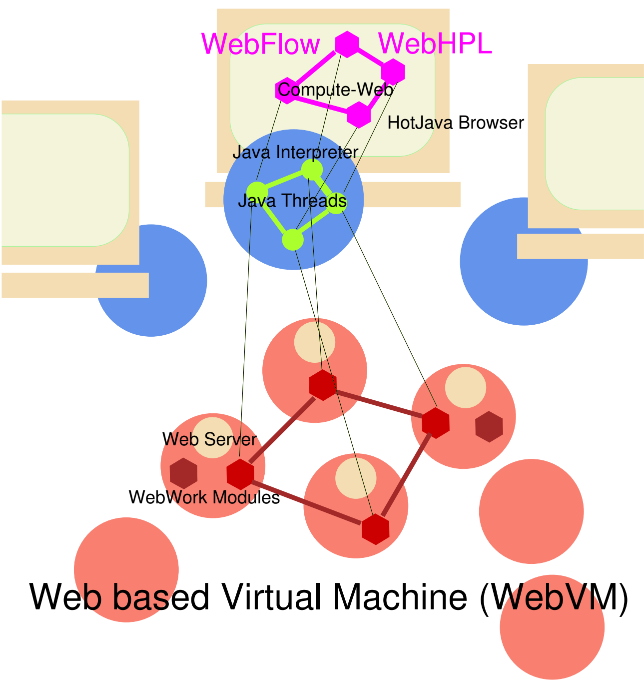


Fig. 2: Client (Java based visual editor) and Server (WebVM) components of a WebFlow environment.

Similar expansion needs arise in Intelligent Agents based information processing problems. Agents, sent to the 'cloud' in search for specific information, find it often useful to spawn other agents/slaves, send them with suitable subtasks to specialized subdomains, and then collect, compare and integrate the individual information scores. The most popular agent technology is given by Telescript, in which the information cloud in modelled in terms of a mesh of Telescript servers-interpreters. Agent objects are autonomous chunks of Telescript with methods supporting propagation in the Telescript medium, reproduction, and rule based intelligent/adaptive behavior.

We initially focus on simpler, more structured and deterministic dynamic compute-webs such as required by AMR. One possible way to handle such problems is to extend the module API by domain specific calls such as *grid_refine* etc. However, such approach mixes domain specific and generic WebVM components and increases complexity of implementation since each domain-specific call (and there will be many such calls for various domains) needs to be associated with multi-language binders for all languages supported by WebVM.

We chose instead a more elegant and problem-independent approach based on scripted extensions of the base WebVM model. In terms of the minimal model and its management modules, we introduce another system process called *Script Executive*, and one more routine in the module API – *script_execute(string)* — which passes scripts in the application-specific *little language* to the Script Executive for immediate execution/interpretation. Script Executive will offer a default set of commands such as creating, destroying or modifying the compute-web topologies, as well as mechanisms for installing new application specific commands by users or their domain system administrators.

Script Executive builds its functionality on top of the base WebVM layer, including the server-to-server message passing, and hence it can also exploit the distributed interpretation capabilities. For example, a user command to create new node in the current compute-web can be implemented with built-in dynamic load balancing: Script Executive can issue a status request to Configuration Managers, select the optimal node based on this information, and finally send the module placement request to the Link Manager in this node. An alternative and often more efficient strategy for load balancing could use periodically refreshed information maintained by local Resource Managers but the scripting channel will offer more direct insight into the real-time operation of WebVM and a platform for refining the existent and prototyping new commands in the WebVM instruction set. Also, system and user strategies can be more easily integrated in the scripted mode. For example, application may provice a special handler module to refine or even override the default node placement process and pass the URL of such a handler to the Script Executive as part of the command script.

At the current design stage we do not yet specify either the syntax or the instruction set of such coordination script. Syntax ambiguity might be soon resolved be new coming products – for example, Netscape can provide an open enough LiveWire support to be useful for our purposes, or W3C/CORBA initiative may come soon enough with standard Web bindings for various language interpreters. Initially, our needs in the scripting sector will be limited and we can tentatively develop a custom small interpreter while waiting for more stable solutions by the Web community. We also note that our scripted extension shares common aspects with other related published specifications, such as: Script Node in VRML 2.0 to specify dynamic behavior of other, more static VRML nodes; or the SQL scripting mechanism in the JDBC interface to relational databases.

# 3 WebFlow – Visual Programming

## 3.1 Visual Authoring for the Web

Several high level programming paradigms can be implemented on top of the WebVM layer discussed in Chapter 2. Based on our previous experience with AVS and Khoros, we view visual authoring of application-specific compute-webs as particularly attractive and useful system integration technique. Java capable browser such as HotJava offers a natural implementation vehicle for such high level tools. We illustrate it in Fig. 2 where the WebVM modules are associated with Java threads at the client side and visually editable via Java GUI/AWT classes. Visual feedback, corresponding to currently instantiated dataflow channels can be implemented e.g. via notification messages sent by individual Flow Executives to the JavaVM at the clent side. In Netscape 2, such implementation is currently impossible due to applet security restrictions, but some of these constraints are to be modified in the following releases, and also one can construct already now an equivalent realtime networking support by using a suitable personal server in the client domain.

We expect such *WebFlow* i.e. visual Web programming paradigm to be come increasingly popular with the growing complexity of the multi-language Web. First commercial tools in this domain are already available (e.g. JFactory for Java software engineering) and simple Java applet based visual editor prototypes are now also within the reach of the Web research community. We expect a family of such authoring tools to appear soon, specialized for various editing tasks and operating at various granularity levels, ranging from the fine grain Java class or VRML behavior editors, to coarse grain problem solving and system integration toolkits.

## 3.2 WebToolkits as Meta Modules

In our visual WebFlow design work, we plan to focus again on some minimal and expectably universal editor primitives that appear in a natural way and are useful at various granularities of the underlying compute-webs. Hierarchical design, in which computer-webs of finer grain modules are encapsulated and become coarse grain modules in a larger compute-web, will play essential role in providing the bridge and navigability between various WebFlow editor instances.

For example, we expect to be using soon an SGI editor to design 3D scenes in a televirtual collaboratory environment, a JavaSoft editor to develop the associated Java classes for Script Nodes with dynamic reactive behavior, and a Netscape editor to integrate such reality window into a multiframe window. Finally, we would use NPAC WebFlow editor to plug-in such an environment/toolkit, now encapsulated and viewed as composite module, into a larger scale distributed application such as discussed in Chapter 1.

Since our WebVM is most adequate, at least initially, for handling large coarse grain modules, we started recently the classification of such *WebToolkits* that are needed for WebWindows based larger scale system engineering. A sample grouping of various current and coming web technology products into self-contained meta modules with well defined functionality and connectivity is collected in the Table. As in the above example of developing a TVR collaboratory, we expect each such toolkit to contain a set of finer grain compute-webs, editable by suitable visual programming tools. Here, we try to identify and encapsulate the top level modules to be used in the final application or service integration.

| Toolkit | Entity | Description | Technologies |
|---|---|---|---|
| WebAgent | rule | Intelligent Agents, Distributed AI | Telescript, Safe-Tcl |
| WebAuthor | widget | Interactive GUI Authoring | JavaScript, LiveWireWebCASE |
| WebCASE | applet, module | Software Engineering | JDE, TDE |
| WebCMS | process(or) | Cluster Management | Odine, EASY, ... |
| WebConf | person | Conferencing/Chat | IRC, MOO, LiveMedia |
| WebDBMS | table/object | Relational & OO DBMS | JDBC, Oracle, Informix, ... |
| WebImg | image | Imaging | AVS, Khoros |
| WebPDA | note | Personal Digital Assistant | Netscap2, Lotus Notes, WebTools |
| WebSim | avatar | Distributed Interactive Simulations | VRML+, DIS/DSI |
| WebVoD | stream | Streamlined Video & Audio | LiveMedia |
| WebWorld | 3D object | 3D Scene/World Design | VRML 1.0 |
| WebHPL | array | multi-server sclable HPCC | Java, E, PCRC |
| WebTVR | world | Web based televirtuality | VRML 2.0, VRML+, MOO |

## 3.3 Large Scale System Integration

Fig. 3 illustrates a Mega WebFlow environment for large scale system integration in terms of WebToolkits collected in the Table. Such a framework is at the moment still far beyond our reach but hopefully soon feasible given the current explosion of the Web technologies. Each toolkit is accessible as AVS-like module icon that can be brought interactively to the visual editor area, customized in terms of the chain of finer grain tools, and integrated. The overall system architecture follows the reusable bridge topology discussed in Chapter 1. Hence, it is applicable for integrating several large scale distributed Web services such as command & control, telemedicine or distance education.

Suppose we focus on telemedicine domain, and even more specifically on our current CareWeb project. The top level encapsulation unit, i.e. an instance of a bridge meta module corresponds to a local (Syracuse) school district at the 'point of need' side, and to a specific (pediatrics) department in a local (SUNY) hospital at the 'point of expertise' side. Other bridge modules, also sketched in the figure, could represent other departments, hospitals or HMO domains that are willing to adopt similar technologies. School nurses connect to the CareWeb bridge for patient record database services, medical information or interactive consulting with a nurse practitioner. New connections are typically to be handled by the central *Agent* module which performs the initial automated triage and provides a suitable switch to an adequate service in the bridge area. A patient record request will be routed to the *DBMS* module which may in turn request additional services from specialized media database servers such as *Img*, *VoD* or *World*. Interactive consulting will be typically routed to a *Conf* module which offers chat–like services. In more complex cases, nurse practitioner may request pediatrician's help. The doctor needs an insight into the patient record and the assessment form as filled so far by the nurse and the nurse practitioner, as well as access to an independent or shared chat space for consulting with any of them or with other doctors. More advanced services are available for special complex cases: for example, radiologist's expertise might be required and the need arises for real-time medical image processing; or the surgeon, when brought to the consulting team requests a virtual surgery playback in a *TVR* session which requires a real-time *HPL* based simulation. Specific medical record databases are coupled with more generic medical information databases to guide the healthcare providers in the decision making process. Agents are routing such requests to on-board MIS databases in the bride area, and if not available, can consult other bridges or finally offer the guided access to the vast but poorly structured medical information on the World-Wide Web.

The above description represents a plausible scenario for a bridge operation in the healthcare domain as we see it today based on interactions so far with the medical community. New contacts bring new ideas and system design requirements. WebFlow based environments for visual programming in large as in Fig. 3 are expected to play essential role in prototyping new services and interactive, incremental system refinement.

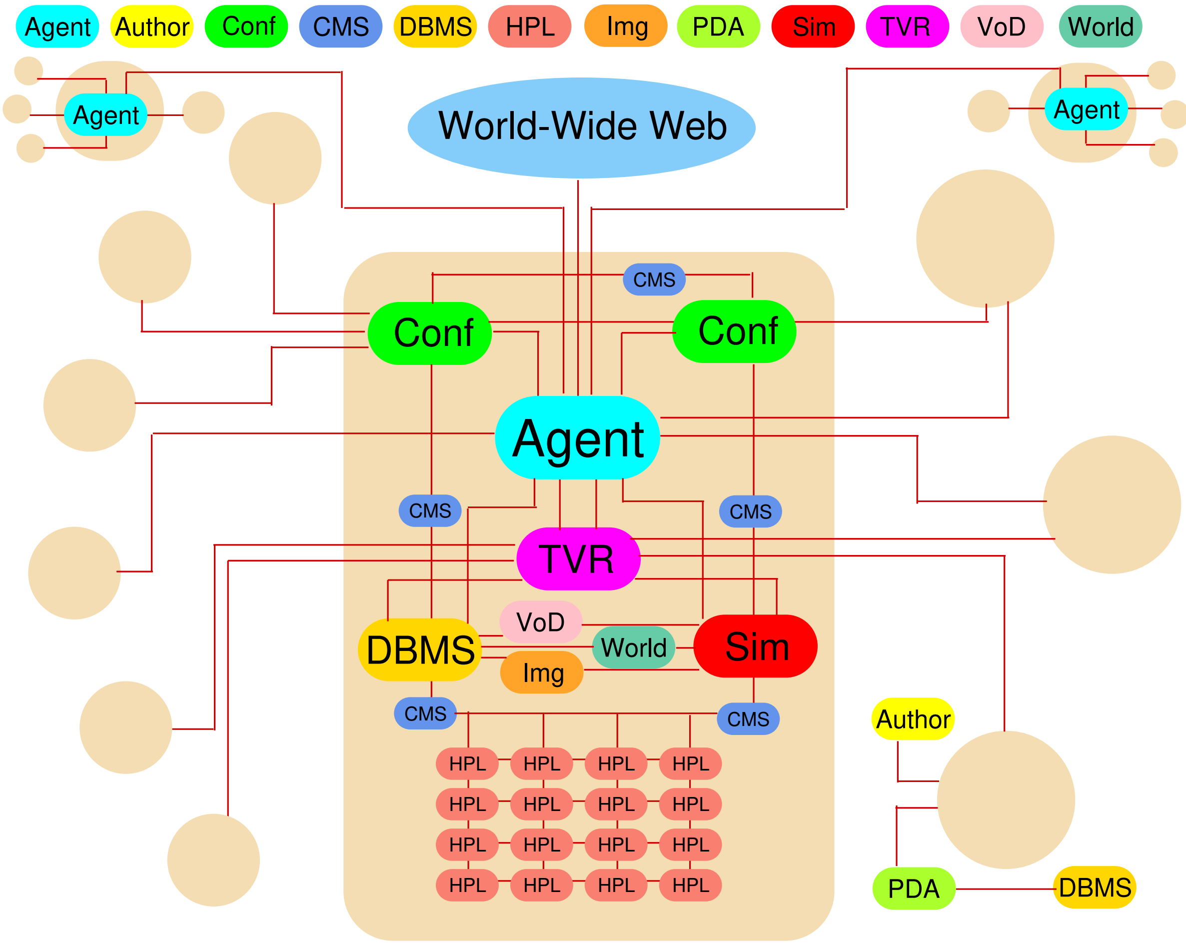


Fig. 3: Example of using WebFlow for large scale software integration. Modules correspond to WebToolkits collected in the Table.

## 4 References

[Baker95] Marek Baker, "Cluster Computing Overview", http://www.npac.syr.edu/techreports/hypertext/sccs-0748/

[Bala95] Balaji Velikandanathan, "Parallel WebSearch", http://kayak.npac.syr.edu:2007/HPDC/Demos/WebWork.html

[BeOl84] M. Berger and J. Oliger, *J. Comp. Phys.* 53, 484 (1984)

[Cheng95] Gang Cheng, "NPAC Oracle 7 Database Server", http://kayak.npac.syr.edu:1963

[Cheng96] Gang Cheng, "AskNPAC Chat-Board", http://kayak.npac.syr.edu:1963/cgi-bin/chat/chat.html

[Cowie95] Jim Cowie, "RSA Factoring by Web", http://www.npac.syr.edu/factoring/

[FFCRC95] Geoffrey Fox, Wojtek Furmanski, Marina Chen, Claudio Rebbi and Jim Cowie, "WebWork: Integrated PRogramming Environment Tools for National and Grand Challenges", NPAC Technical Report, SCCS-715., http://www.npac.syr.edu/techreports/html/0700/abs-0715.html

[FFP95a] Geoffrey Fox, Wojtek Furmanski and Marek Podgorny, "WebSpcae: A WebWindows Based Gateway to ANL Labspace", http://ww.npac.syr.edu/projects/webspace/

[FFP95b] Geoffrey Fox, Wojtek Furmanski and Marek Prodgorny, " Interactive Web based HPDC Technologies for Distance Education", http://ww.npac.syr.edu/users/gcf/hpdc95master/

[FFPCR95] Geoffrey Fox, Wojtek Furmanski, Marek Prodgorny, Gang Cheng and Roman Markowski, " Interactive Web Technologies for Distance Education", http://ww.npac.syr.edu/users/gcf/sc95tutorial/

[FoFu95] Geoffrey Fox and Wojtek Furmanski, "The Use of the National Information Infrastructure and High Performance Computers in Industry", NPAC Technical Report SCCS-732, July 1995, http://www.npac.syr.edu/techreports/html/0700/abs-0732.html

[Fox96] Geoffrey Fox, "WebFoil", http://www.npac.syr.edu/users/gcf/webfoil/Main.html

[Furm95a] Wojtek Furmanski, "WebTools", http://king.syr.edu:2006/WebTools.html

[Furm95b] Wojtek Furmanski, "HotJava WebFoil", http://www.npac.syr.edu/users/gcf/rcihalloween

[Haupt96] Tom Haupt, "Early experiments towards definition of a WebFlow interface", http://aga.npac.syr.edu:1995

[Vish95a] Vishal Mehra, "WebFlow Editor Front-End", http://king.syr.edu:2006/HPDC4/Demos/WebFlow.html

[Vish95b] Vishal Mehra, "Alpha Java Chat Server and Client",

[WaBa96] David Warner and David Balch, "Medicine Meets VR: BRIDGE", http://www.telemed.med.ecu.edu/bridge/bridge_1.htm