Developing for Web Integration in Sisyphus-IV: WebGrid-II Experience

Brian R. Gaines and Mildred L.G. Shaw
Knowledge Science Institute
University of Calgary
Alberta, Canada T2N 1N4
gaines@cpsc.ucalgary.ca, mildred@cpsc.ucalgary.ca

Abstract

The Sisyphus-IV initiative was proposed to encourage research based on the collaborative use of knowledge acquisition and management tools through the net and web and, in particular, the integration of web tools at different sites through the net. This article describes experience in the provision of web-based services for knowledge elicitation, analysis, comparison, modeling and inference to remote users accessing them through the net using a standard web browser. It describes the key features of the system architecture, provision for the support of remote users, how collaboration is supported, and the various means whereby integration is achieved with other systems.

1 Introduction

One major dimension of research reported through the Knowledge Acquisition Workshops is that theories, methodologies and techniques have been made operational through their implementation in tools that have been demonstrated to the community. It was natural also to attempt to share these tools in order to evaluate their efficacy when used by those other than their originators. In the early years the attempts to share tools were not very successful because:-

- Porting tools to different environments proved difficult
- Integrating tools from other sites proved difficult
- Supporting multiple sites was problematic
- Method of use was not well-defined by originators
- Tools were effective for originators but ineffective for others

These issues led to the development of the Sisyphus series of challenge problems in which, rather than share tools, a knowledge acquisition problem was defined and tool developers were challenged to solve it with their tools. Four Sisyphus initiatives have been defined:

Sisyphus-I: The first Sisyphus initiative was a room allocation problem, a resource allocation task in which a number of people with differing requirements as to type of room have to be allocated appropriate rooms from a number of rooms with differing characteristics.

Sisyphus-II: The room allocation problem proved to be relatively simple and the second Sisyphus initiative attempted to provide a realistic knowledge-engineering problem based on the domain knowledge for elevator configuration that was used to build the VT system (Marcus, Stout and McDermott, 1988).

Sisyphus-III: The third Sisyphus initiative is using an igneous rock classification problem to focus on the actual processes and effort involved in knowledge acquisition.

Sisyphus-IV: In recent years many knowledge acquisition tools have been ported to operate through the Internet (the *net*) and World Wide Web (the *web*), and it has become possible to provide access to tools to the community in a way that alleviates the problems noted above:

- Once tools are ported to the web they are available on all major platforms through web browsers

- The Internet provides the capability to integrate tools running on different servers
- Multiple sites can be supported technically because only the server needs updating, and support of use is more readily provided because the attempted use can be monitored at the server
- Methods of use still need to be defined but the literature and application examples have grown over the past ten years
- It is easier for the originators to collaborate with other users over the Internet which should increase the effective use of the tools

The Sisyphus-IV initiative was proposed at KAW'96 to encourage projects testing these assumptions by collaborative use of tools through the net and web and by the integration of web tools at different sites through the net.

A range of systems reported at KAW'96 already address some of the objectives of Sisyphus-IV, notably:

- Ontolingua server, Stanford's web-based tool for collaborative ontology construction (Farquhar, Fikes and Rice, 1996)
- Ontosaurus, ISI's web-based tool for browsing and editing ontologies and knowledge bases (Swartout, Patil, Knight and Russ, 1996)
- WebGrid, Calgary's web-based knowledge modeling and inference tool based on repertory grid elicitation and analysis (Gaines and Shaw, 1996)

There were also a number of web-based sub-systems demonstrated that can provide components of complete knowledge acquisition and knowledge management systems.

This article reports on the development of WebGrid-II to address the issues for Sisyphus-IV noted above and gives examples of integration with other systems.

2 WebGrid-II Architecture

WebGrid-II operates as a web server to offer knowledge elicitation, analysis, comparison, modeling and inference services to remote users accessing it through the Internet using a standard web browser such as Netscape or Internet Explorer. Its user interface is entirely through the interactive generation of active HTML documents using standard web capabilities such as typographic text, forms, images, links and clickable maps. It is designed to minimize server administration by conforming with the web's stateless protocol and storing the data being elicited in hidden fields within the documents returned to the user. This enables the user to manage their own data storage as if they were operating a local application by storing the HTML documents returned by WebGrid-II on their local disks.

Figure 1 shows the overall WebGrid-II architecture. At the right, when the server receives a document from a user client it extracts the grid from the hidden fields and extracts the command and associated data entered by the user. It then executes the command drawing upon the underlying repertory grid tools as necessary, updates the grid if appropriate and establishes the next state of the interaction. It then generates a document to return to the user using the script for that state, passing to it appropriate parameters from the grid. Under certain circumstances, such as the support of collaboration, it may draw upon other grid data cached at the server by the user.

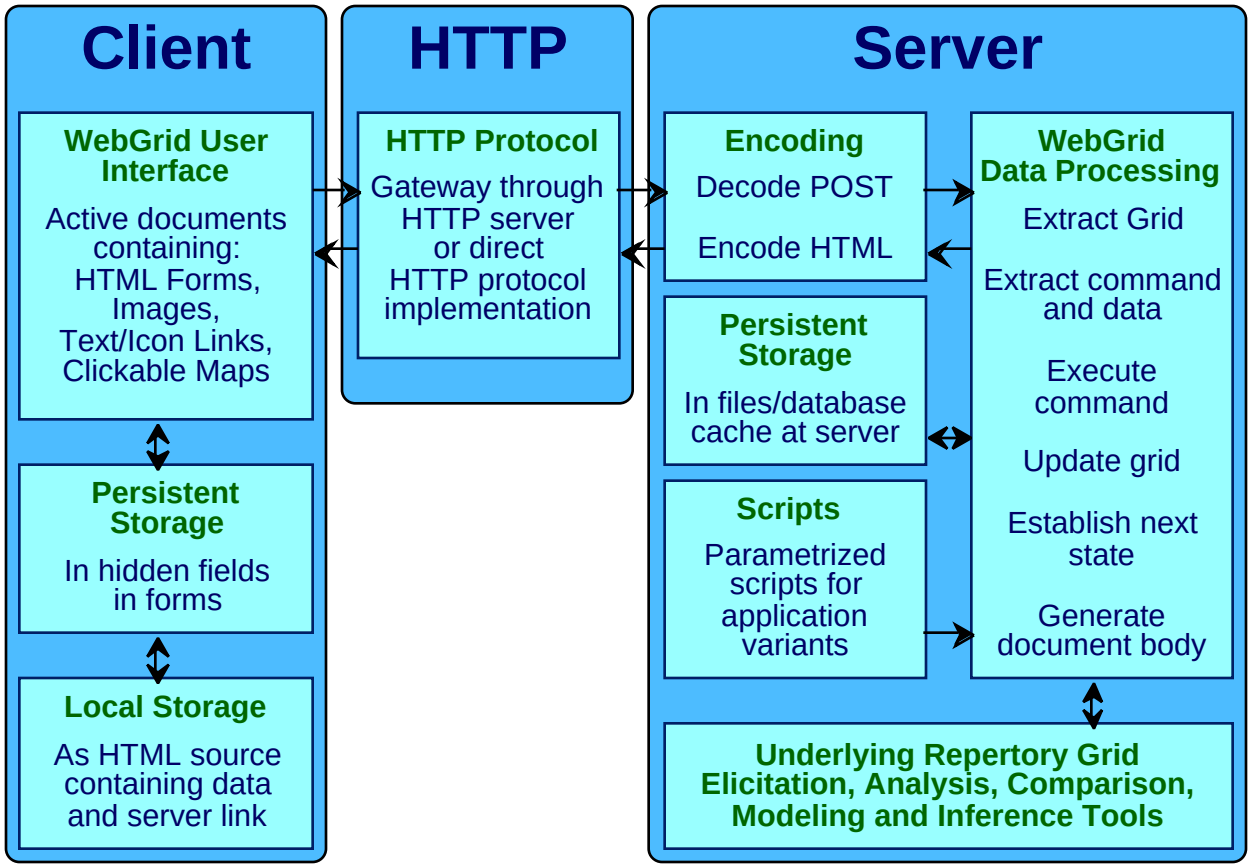

Figure 1 WebGrid-II architecture

Figure 2 shows the first part of a typical document returned by WebGrid-II to the user. Line numbers have been added for ease of reference. Line 2 is a BASE tag specifying the Internet address of the WebGrid-II server being used. This ensures that when the document is stored locally and reloaded then interaction will continue with the server without any special action on the part of the user. To the users the web browser on their machines constitute the primary 'application' and the server is an invisible resource. The documents they save locally contain sufficient information to reconnect automatically to the appropriate server. This is the only reference to the server in the document making it simple to connect to another server if required.

Line 7 is hidden data giving the version of WebGrid and the date and time at which the document was generated. The data fields are tab-separated so that the document can be loaded into a spread sheet for further analysis if required. However, in the figure the tabs have been converted to vertical lines for ease of reading.

Lines 8 through 28 contain the data in the grid. Lines 10 through 16 specify the constructs. Line 10 specifies a rating scale construct, line 11 a floating point construct, and line 14 a categorical construct (as indicated by the initial letter of the VALUE argument). Lines 17 through 28 specify the elements as a list of values on the constructs followed by the element name.

Line 34 specifies a context-sensitive information icon and the script to be called when the user clicks on it. Line 35 contains hidden data specifying the pair of constructs which have been specified as matching in case the user decides to add an element to break the match.

```
1  <HTML><HEAD>
2  <BASE HREF="http://tiger.cpsc.ucalgary.ca/">
3  <TITLE>WebGrid-II Elicitation</TITLE></HEAD>
4  <BODY BGCOLOR="FFFFDD"><IMG SRC="WebGrid/WebGrid.gif" ALIGN="left" ALT="WebGrid Icon">
5  <H1 ALIGN="center">WebGrid-II <I>Elicitation</I></H1><HR>
6  <FORM ACTION="WebGrid/Main.k" METHOD=POST>
7  <INPUT TYPE="hidden" NAME="WebGrid" VALUE="11|1|11/11/97|10:36:28 PM">
8  <INPUT TYPE="hidden" NAME="Head" VALUE="6|0|1|1|3|0|9|7|12|R|1|5|">
9  <INPUT TYPE="hidden" NAME="Labels" VALUE="MLGS|graduate students|selecting among graduate program
   applicants|MLGS|:datasets:GradProgram12a|quality|qualities|student|students">
10 <INPUT TYPE="hidden" NAME="C0" VALUE="R|32|0|10|10|1|5||admit|reject|This is what we are trying to
   predict">
11 <INPUT TYPE="hidden" NAME="C1" VALUE="F|0|7|10|10|4|0|4|gpa|0|2.99|inadequate|3|4|adequate|3.5|4|high|
   The grade point average has to be above 3 for entrance through Graduate Studies but we prefer it above 3.5">
12 <INPUT TYPE="hidden" NAME="C2" VALUE="R|0|2|10|10|1|5||good references|inadequate references">
13 <INPUT TYPE="hidden" NAME="C3" VALUE="R|0|3|10|10|1|5||willing supervisor|no supervisor">
14 <INPUT TYPE="hidden" NAME="C4" VALUE="C|1|7|10|10|2|scholarship|major|none|If the student has a major
   scholarship then we may admit if the chance of getting a supervisor on arrival looks good">
15 <INPUT TYPE="hidden" NAME="C5" VALUE="R|0|5|10|10|1|5||SE application|not SE">
16 <INPUT TYPE="hidden" NAME="C6" VALUE="R|0|6|10|10|1|5||industry experience|little experience">
17 <INPUT TYPE="hidden" NAME="E0" VALUE="1|0|3.8|0|0|1|4|4|fgh">
18 <INPUT TYPE="hidden" NAME="E1" VALUE="1|4|3.2|0|0|1|4|4|jk">
19 <INPUT TYPE="hidden" NAME="E2" VALUE="1|4|3.75|4|0|1|4|4|mrt">
20 <INPUT TYPE="hidden" NAME="E3" VALUE="1|4|3.9|0|4|1|4|4|ps">
21 <INPUT TYPE="hidden" NAME="E4" VALUE="1|0|3.5|0|4|0|4|4|lemp">
22 <INPUT TYPE="hidden" NAME="E5" VALUE="0|4|3.1|0|4|0|4|4|ml">
23 <INPUT TYPE="hidden" NAME="E6" VALUE="0|4|3.9|4|4|0|4|4|tjb">
24 <INPUT TYPE="hidden" NAME="E7" VALUE="0|0|3|0|4|1|0|0|ra">
25 <INPUT TYPE="hidden" NAME="E8" VALUE="0|4|2.9|0|4|1|0|0|pek">
26 <INPUT TYPE="hidden" NAME="E9" VALUE="0|4|3.3|4|4|1|0|0|fn">
27 <INPUT TYPE="hidden" NAME="E10" VALUE="0|4|3.1|0|4|1|4|0|ptds">
28 <INPUT TYPE="hidden" NAME="E11" VALUE="0|4|3.4|0|4|1|0|4|wsm">
29 <INPUT TYPE="hidden" NAME="Analysis" VALUE="0|I|108|100|">
30 <INPUT TYPE="hidden" NAME="Body" VALUE="<BODY BGCOLOR=^FFFFDD^>">
31 <BR CLEAR="LEFT">
32 You are considering <B>12</B> students and <B>7</B> qualities in the context of <B>selecting among graduate
   program applicants</B>
33 <INPUT TYPE="Submit" NAME="Continue" VALUE="Continue">
34 <INPUT TYPE="IMAGE" SRC="/q.gif" BORDER=0 NAME="!Main.Continue">
35 <INPUT TYPE="hidden" NAME="CPair" VALUE="5|6|">
36 <HR>
37 The qualities <B>SE application--not SE</B> and <B>industry experience--little experience</B> are very similar -
   click here if you want to enter another student to distinguish them
38 <INPUT TYPE="Submit" NAME="CMatch" VALUE="Distinguish">
39 <INPUT TYPE="IMAGE" SRC="/q.gif" BORDER=0 NAME="!Main.CMatch">
40 <HR>
   .....
```

Figure 2 WebGrid-II document with hidden fields

Figure 3 shows the script used by WebGrid to generate the output shown in Figure 2. Each part of the script consists of a field commencing with \\ followed by an identifier and terminated by \ alone on a line. The \ character followed by an argument is a substitution macro.

The family of scripts being used by WebGrid-II is identified by an argument specifying the directory in which the script is stored in the URL used to call WebGrid. If the script is not found in that directory it is fetched from a default directory. Hence, it is easy to customize particular scripts for different applications. It is also possible to give a remote researcher FTP access to a specific script directory in which they may develop and test the scripts for their application.

```
1  <TITLE>WebGrid-II Elicitation</TITLE>\
2  <IMG SRC="WebGrid/WebGrid.gif" ALIGN="left" ALT="WebGrid Icon">
3  <H1 ALIGN="center">WebGrid-II <I>Elicitation</I></H1>\l
4  \
5  \\N
6  <BR CLEAR="LEFT">
7  You are considering <B>\0</B> \1 and <B>\2</B> \3 in the context of <B>\D</B>
8  <INPUT TYPE="Submit" NAME="Continue" VALUE="Continue">
9  \?Main.Continue\
10 \
11 \\A
12 \l<B>You should add additional \E if possible </B>
13 <INPUT TYPE="Submit" NAME="EAdd" VALUE="Add">
14 \?Main.Add\
15 \
16 \\U
17 \l
18 \#0There is <B>1</B> unspecified value\
19 \#There are <B>\0</B> unspecified values\
20 <INPUT TYPE="Submit" NAME="Specify" VALUE="Specify">
21 \?Main.Specify\
22 \
23 \\m
24 \lThe \C <B>\0</B> and <B>\1</B> are very similar -click here if you want to enter another \e to distinguish them
25 <INPUT TYPE="Submit" NAME="CMatch" VALUE="Distinguish">
26 \?Main.CMatch\
27 \
.....
```

Figure 3 Script generating output of Figure 2

The WebGrid-II page whose initial sections are generated by the script in Figure 3 and encoded in HTML as shown in Figure 2 is shown on the left of Figure 4.

3 Supporting Remote Users

WebGrid-II was designed to support anonymous remote users effectively. The WebGrid-II home page has pointers to relevant background materials on repertory grids and their applications in a variety of contexts, and it assumed that users will familiarize themselves with the underlying methodologies. WebGrid provides context-sensitive on-line help through a small “?” icon which appears to the right of material about which information may be needed.

Figure 4 left shows a WebGrid-II page with “?” icons, and on the right is shown the information window which open when the “?” icon next to the “Edit note” button at the bottom of the list of constructs is clicked. The information displayed is generated from a script supporting macro substitution and hence may be customized from the data entered by the user. It gives a short account of functions available in the relevant section. It can include hypertext links to the WebGrid-II manual, articles, or sample data as appropriate. All information is targeted into one window and hence the user can position this in a convenient location, including behind the main WebGrid-II window on smaller screens since the information window automatically comes to the front when information is requested.

On-line access to context-sensitive information is one way of supporting remote users. For more direct support WebGrid provides facilities for capturing user interactions at the server if required so that a remote user can be monitored. This facility is normally off to reduce server load and respect users’ privacy. However, it is valuable for supporting a remote user on request. Timbuktu and web collaboration facilities can also be used to complement this in supporting remote users.

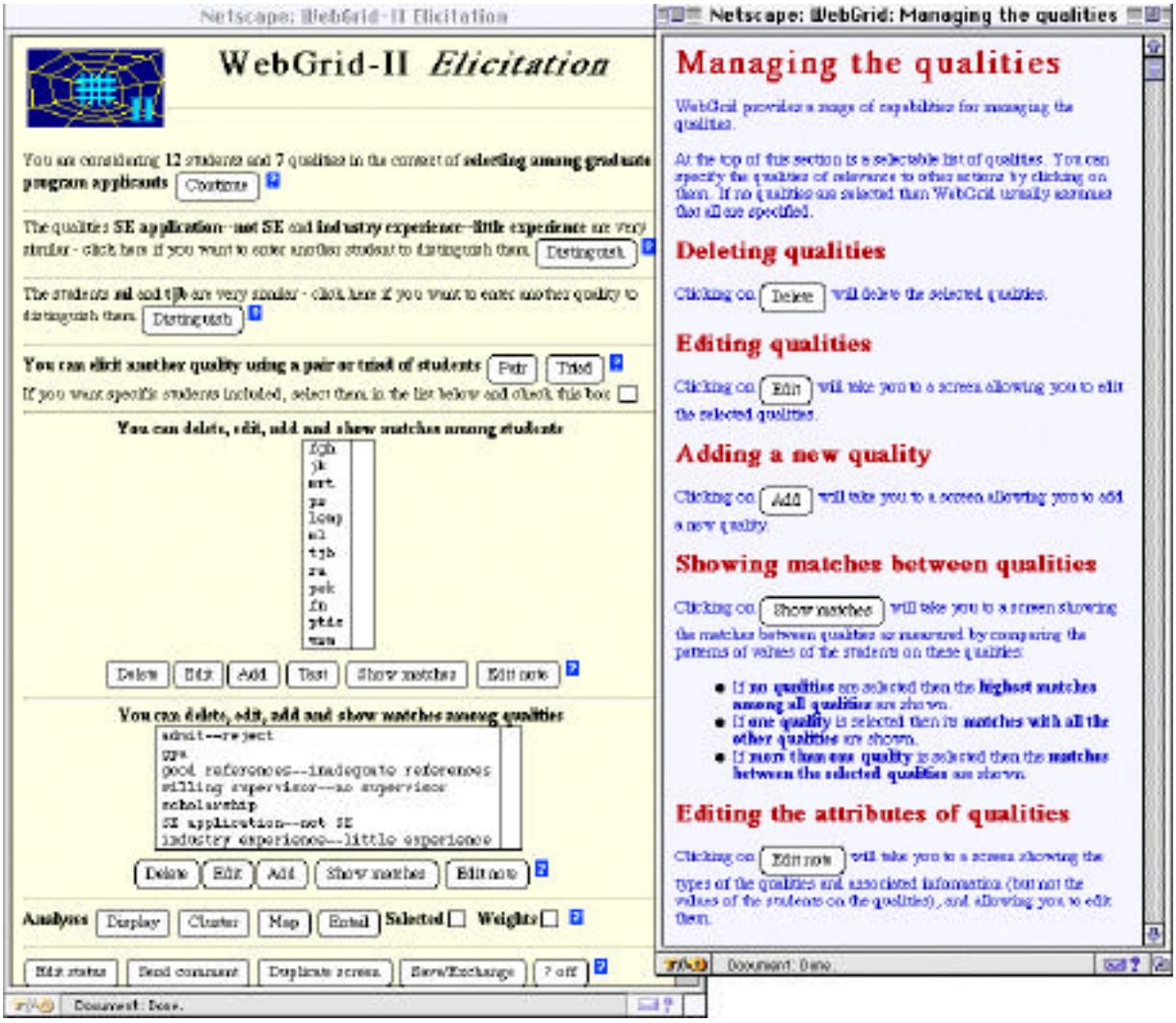

Figure 4 Context-sensitive supportive information in WebGrid-II

4 Supporting Collaboration

One of the major features of the RepGrid family of tools on which WebGrid-II is based is the capability to compare the conceptual structures of different users and analyze them in terms of consensus, conflict, correspondence and contrast (Gaines and Shaw, 1989; Shaw and Gaines, 1989). It was important that this be supported in WebGrid-II but difficult to achieve using a stateless protocol in which grid data was stored at the client site. One possibility was to store two grids in one document but this would only support pairs of users not communities. It would also be possible to give certain users password access to the server store and this was done in Web-Grid-I (Shaw and Gaines, 1995), but this was administratively undesirable in attempting to support large anonymous communities.

The solution developed for WebGrid-II is to allow users to cache their grids at the server for a period of 3 months. The time period itself is not a limitation because users may recache whenever they wish, but it allows storage utilization to be controlled through cache expiry. The caching itself has been designed to be extremely simple for the user. Figure 5 shows the screen returned by WebGrid-II when the "Save/Exchange" button at the bottom of Figure 4 is clicked. To its left is the information on caching returned when the "?" icon next to the "Caching a grid for use by others" is clicked.

Figure 5 Saving and caching grids in WebGrid-II

When the user clicks on the "Cache" button the screen shown in Figure 6 is returned. It notes that the grid has been cached for 3 months and gives 3 URL's: the first for reloading the grid into WebGrid-II; the second for loading the grid elements only so that the user can develop constructs and compare them with those in the original grid for correspondence and consensus; the third for loading the grid elements and constructs but no values so that the user can enter values and compare them with those in the original grid for consensus and conflict.

The user can issue these URL's to other users who can use them to access WebGrid-II and develop their own grids for comparison with others. A user without a web site can also use this as a means of publishing their grids. However, most users will embed the URL's in documents at their own site and use this as a basis for collaboration with others. In particular, experimenters can now implement our complete comparison methodology remotely on an anonymous basis without specific support.

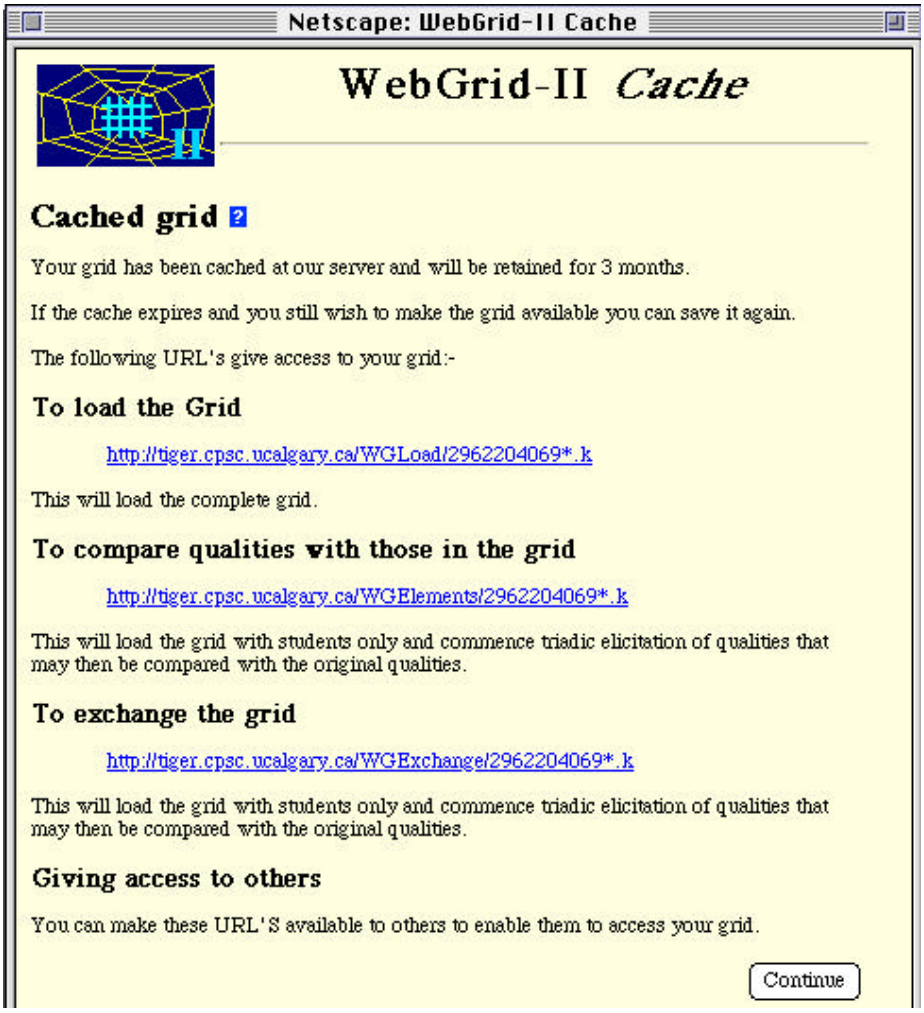

Figure 6 Accessing cached data in WebGrid-II

5 Passing Control from Another System to WebGrid and Back

One objective of the WebGrid-II design was to support users who wished to collect data on elements and constructs through their own methodology, pass them to WebGrid-II for further elicitation and analysis, and then return data and control to their own system.

Passing control to WebGrid-II is simple since all one has to do is send it a document with a data structure of the form shown in Figure 2. This is normally generated by WebGrid-II itself but is simple to output from any system that has gathered relevant data. Passing control back is also simple since WebGrid-II supports user-specified headers and trailers and these can contain JavaScript that returns the WebGrid-II data and control of the interaction to another server. WebGrid-II also supports additional hidden fields containing data from the original server which are passed back to it when control returns, thus enabling it to maintain state information.

The calling system may wish to restrict the WebGrid-II features available to the user and to exert some control over the interaction. WebGrid-II is designed such that the scripts may be simply edited to remove features and the system defaults to the expected behavior. It also supports a command language embedded in hidden fields that enable interaction to be controlled, for example, to take the user through a sequence of construct elicitations or ratings on prescribed constructs. The command language can also be used to program WebGrid-II to return control to another server when the prescribed interactions are complete.

These capabilities have been used to support a colleague in a management department in Florida who is conducting research on 360-degree survey methodologies (Jones and Bearley, 1966). He has developed his own form for element collection, and the demonstration form designed to show him how to do this is shown in Figure 7. It collects some demographic data which will not be used by WebGrid-II but just sent in with the grid data. It then asks for the names of 6 managers, characterized intensionally.

Figure 7 Customized data collection front-end to WebGrid-II

Figure 8 shows the HTML generating the document of Figure 7. It has been annotated to help in preparing the actual document used in the application. Lines 1 through 14 are the usual WebGrid-II header and initial grid data. Lines 15 through 28 are annotated commands that specify that two constructs are to be elicited from specified triads, the elements are to be rated on the two prescribed constructs, control is to be returned to a server, and the user is to be thanked. The 'server' specified is a "mailto" URL which causes the data to be mailed to the experimenter.

The experimenter edited this form to collect 10 elements specified intensionally by their type, and commenced with a grid that already had two named elements and 12 prescribed constructs. WebGrid-II enables him to undertake data collection from remote subjects without having to undertake any web programming other than the customization of the initial page as shown.

With a small amount of CGI programming the data collection shown in Figure 7 can become a customized series of interactions in some other system that then passes control to WebGrid-II and eventually receives the data and control back either automatically through the commands or under user control through a button in the header or footer.

```
1  <HTML><HEAD>
2  <BASE HREF="http://tiger.cpsc.ucalgary.ca/">
3  <! Title is up to you>
4  <TITLE>Management Test</TITLE></HEAD>
5  <! Body color scheme and headings are up to you>
6  <BODY BGCOLOR="FFFFDD">
7  <H1 ALIGN="center">WebGrid-II <I>Management Test</I></H1><HR>
8  <! The following material should come from a grid you develop that just contains the constructs you want to have
   used>
9  <FORM ACTION="WebGrid/Status/Manage.k" METHOD=POST>
10 <INPUT TYPE="hidden" NAME="WebGrid" VALUE="11|30911|7/13/97|11:02:01 AM">
11 <INPUT TYPE="hidden" NAME="Head" VALUE="3|0|0|0|9|2|0|R1 10|">
12 <INPUT TYPE="hidden" NAME="Labels" VALUE="Temp|manager appraisal|appraising
   managers|Temp||quality|qualities|manager|managers">
13 <INPUT TYPE="hidden" NAME="C0" VALUE="R1 0 0 10 10 1 5||ineffective|effective">
14 <INPUT TYPE="hidden" NAME="C1" VALUE="R1 1 1 10 10 1 5||follower|leader">
15 <! The following is a sequence of commands you need to specify>
16 <! The first is a triad on elements 0, 1 and 2 with a check that at least 6 elements have been entered>
17 <INPUT TYPE="Hidden" Name="Command" Value="T0 1 2 6">
18 <! The first is a triad on elements 3, 4 and 5 -- put in as many triads as you want>
19 <INPUT TYPE="Hidden" Name="Command" Value="T3 4 5">
20 <! The next two commands are for rating on each of your prescribed constructs>
21 <INPUT TYPE="Hidden" Name="Command" Value="C0">
22 <INPUT TYPE="Hidden" Name="Command" Value="C1">
23 <! The next command sets up your email address for the data -- edit this>
24 <INPUT TYPE="Hidden" Name="Command" Value="Smailto:someone@somewhere?subject=Test Data
   Collection">
25 <! The next command says the data should be sent to you as plain text>
26 <INPUT TYPE="Hidden" Name="Command" Value="Etext/plain">
27 <! The next command says send message one from the "extra" script -- a thank you>
28 <INPUT TYPE="Hidden" Name="Command" Value="X1">
29 <! The next section is your element entry -- edit it to collect as many elements as you wish>
30 <P>Please enter some demographic data<BR>
31 <P><B>Name </B><INPUT NAME="User" VALUE="">
32 <! Any fields starting with _ are yours and will be collected and returned to you>
33 <B>Title </B><INPUT NAME="_Title" VALUE="">
34 <P><B>Location </B><INPUT NAME="_Location" VALUE=""><BR><BR>
35 <HR><P><b>Enter a manager with whom you work closely</b><BR>
36 <INPUT NAME="Elements" SIZE=60><BR>
37 <P><b>Enter another manager who you admire</b><BR>
38 <INPUT NAME="Elements" SIZE=60><BR>
39 <P><b>Enter another manager who you feel is ineffective</b><BR>
40 <INPUT NAME="Elements" SIZE=60><BR>
41 <P><b>Enter another manager of your choice</b><BR>
42 <INPUT NAME="Elements" SIZE=60><BR>
43 <P><b>Enter another manager of your choice</b><BR>
44 <INPUT NAME="Elements" SIZE=60><BR>
45 <P><b>Enter another manager of your choice</b><BR>
46 <INPUT NAME="Elements" SIZE=60><BR>
47 <DIV ALIGN="RIGHT">
48 <P><B>When you are ready click on</B>
49 <INPUT TYPE="Submit" NAME="OK" VALUE="Done">
50 </DIV></FORM></BODY></HTML>
```

Figure 8 Annotated HTML generating the document shown in Figure 7

6 Using WebGrid-II as a Server

WebGrid-II is a server that receives data and commands and returns a document with embedded data and a user interface supporting further commands. It is simple to set up scripts that remove the user interface completely so that WebGrid-II appears as a service intended for use by other programs. However, it is also possible for the other programs to ignore the user interface portions of a WebGrid-II document and use it as a server anyway.

Jeni Tennison has used this approach to make WebGrid-II a service integrated with her APECKS (Adaptive Presentation Environment for Collaborative Knowledge Structuring) system on the web (Tennison, 1997a). APECKS provides a virtual environment where people who are not in the same place in the real world can meet and talk, and supports a cyclical process of construction, comparison and discussion about knowledge representations through which an organizational memory is gradually constructed.

A detailed experimental diary on Tennison's experience in integrating APECKS and WebGrid-II is available on the web (Tennison, 1997b) and an article on this research will be presented at KAW'98 (Tennison and Shadbolt, 1998).

7 Conclusions

The WebGrid development is proceeding along three major tracks: first, to expand the capabilities of WebGrid itself to support richer representation systems, modeling and inference; second, to support multi-user, multi-site collaboration; and, third, to support integration with a range of other systems. This article has focused on the integration issues, and a companion article discusses those of richer representation (Shaw and Gaines, 1998).

There are many further developments planned such as: using WebGrid-II as a front end to the various ontology tools available on the web; expanding the facilities of WebGrid-II to encompass inference with multiple knowledge structures as was done in our solution to the Sisyphus-I problem (Gaines, 1994); supporting communities of users through multiple grid analyses as was done in RepGrid-Net (Shaw and Gaines, 1993); providing a scriptable front-end for customized elicitation; improving facilities for integration with other systems in the light of the APECKS experience; and integrating concept-mapping tools supporting visual languages such as KDraw (Gaines, 1991) using java applets (Flores-Méndez, 1996).

We welcome further collaborators in using WebGrid-II as part of the Sisyphus-IV initiative.

Acknowledgments

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada.

References

Farquhar, A., Fikes, R. and Rice, J. (1996). The Ontolingua server: a tool for collaborative ontology construction. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Tenth Knowledge Acquisition Workshop**. pp.63-1-63-7 (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/farquhar-demo/farquhar-demo.html).

Flores-Méndez, R. A. (1996). Distributed concept mapping collaboration using Java. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Tenth Knowledge Acquisition Workshop**. pp.64-1-64-7 (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/flores-mendez/KAW96demo.html).

Gaines, B.R. (1991). An interactive visual language for term subsumption visual languages. **IJCAI'91: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence**. pp.817-823. San Mateo, California, Morgan Kaufmann.

Gaines, B.R. (1994). A situated classification solution of a resource allocation task represented in a visual language. **International Journal Human-Computer Studies 40**(2) 243-271.

Gaines, B.R. and Shaw, M.L.G. (1989). Comparing the conceptual systems of experts. **Proceedings of the Eleventh International Joint Conference on Artificial Intelligence**. pp.633-638. San Mateo, California, Morgan Kaufmann.

Gaines, B.R. and Shaw, M.L.G. (1996). WebGrid: knowledge modeling and inference through the World Wide Web. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Tenth Knowledge**

Acquisition Workshop. pp.65-1-65-14 (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/gaines/KMD.html).

Jones, J.E. and Bearley, W.L. (1966). **360° Feedback: Strategies, Tactics, and Techniques for Developing Leaders**. Amherst, MA, HRD Press.

Marcus, S., Stout, J. and McDermott, J. (1988). VT: an elevator designer that uses knowlede-based backtracking. **AI Magazine Spring** 95-111.

Shaw, M.L.G. and Gaines, B.R. (1989). A methodology for recognizing conflict, correspondence, consensus and contrast in a knowledge acquisition system. **Knowledge Acquisition 1**(4) 341-363.

Shaw, M.L.G. and Gaines, B.R. (1993). Group knowledge elicitation over networks. Bramer, M.A. and Macintosh, A.L., Ed. **Research and Development in Expert Systems X. Proceedings of British Computer Society Expert Systems Conference**. pp.43-62. Cambridge, UK, Cambridge University Press.

Shaw, M.L.G. and Gaines, B.R. (1995). Comparing constructions through the web. Schnase, J.L. and Cunnius, E.L., Ed. **Proceedings of CSCL95: Computer Support for Collaborative Learning**. pp.300-307. Mahwah, New Jersey, Lawrence Erlbaum.

Shaw, M.L.G. and Gaines, B.R. (1998). WebGrid II: Developing Hierarchical Knowledge Structures from Flat Grids. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Eleventh Knowledge Acquisition Workshop**. pp.submitted (http://ksi.cpsc.ucalgary.ca/KAW98S/shaw/).

Swartout, B., Patil, R., Knight, K. and Russ, T. (1996). Ontosaurus: a tool for browsing and editing ontologies. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Tenth Knowledge Acquisition Workshop**. pp.69-1-69-12 (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/ontosaurus_demo.html).

Tennison, J. (1997a). APECKS: Adaptive Presentation Environment for

Collaborative Knowledge Structuring. Department of Psychology, University of Nottingham. http://www.psychology.nottingham.ac.uk/staff/Jenifer.Tennison/APECKS/.

Tennison, J. (1997b). Linking APECKS to WebGrid-II. Department of Psychology, University of Nottingham. http://www.psychology.nottingham.ac.uk/staff/Jenifer.Tennison/APECKS/WebGrid.html.

Tennison, J. and Shadbolt, N.R. (1998). APECKS: a Tool to Support Living Ontologies. Gaines, B.R. and Musen, M.A., Ed. **Proceedings of Eleventh Knowledge Acquisition Workshop**. pp.submitted (http://ksi.cpsc.ucalgary.ca/KAW98S/tennison/).