WebGrid Evolution through Four Generations 1994-2007

Brian R. Gaines and Mildred L. G. Shaw
University of Victoria and University of Calgary
gaines@uvic.ca, mildred.shaw@ucalgary.ca

WebGrid is a port of a highly interactive conceptual modeling application to operate effectively as a web service available through any web client. It was developed in 1994, has been available as a service on the web on a 24/7 basis since January 1995, and has been used by over 20,000 different sites. A tutorial on porting interactive applications to the web using WebGrid as an example was presented at the Fourth International World Wide Web Conference in 1995. This article updates that material in the light of a decade of further experience, describing the significant lessons learned in designing and evolving WebGrid through four major generations as the underlying web technologies have evolved from the initial implementation of HTML forms through cascading style sheets and Javascript to dynamic HTML and Ajax, and as user requirements have themselves evolved.

1 Introduction

The initial concept (Berners-Lee, 1989) of the World Wide Web was that of providing ease of access through the Internet to a network of documents navigated through embedded hypertext links in a standard hypertext markup language (HTML). This concept was greatly extended when Marc Andreessen and Eric Bina in August 1993 supported embedded Motif widgets in the Mosaic browser and later used this to support HTML+ forms. The capability from HTML 2 onwards of being able to implement active documents having embedded within them the full range of basic user interface widgets enabled a wide range of interactive applications to be ported to the web. In essence, HTML came to be used as a graphic user interface programming language, and web browsers as universal interactive interfaces to a wide variety of applications.

The attractions of porting interactive applications to the web were that: they became available to any user having a web browser on any platform connected to the Internet; there was no need to distribute and update code to all users; code need be developed for only one platform; only the server code needed to be maintained and enhanced; usage could be monitored remotely, problems noted and fixed; collaborative applications could be developed as all users were on a common network; typographic, multimedia and hypertext features were automatically available; and the document model was familiar to most users.

Problems in porting interactive applications in the early days arose largely from the stateless nature of the hypertext transport protocol (HTTP), the limited range of user interface widgets supported, and the lack of local interactivity. Later developments such as Unicode which supports symbols and non-Roman fonts, cascading style sheets (CSS) that separate the document content from the way it is presented, embedded Javascript and dynamic HTML (DHTML) that together allow a document to be modified interactively, and Ajax technology that allows further material to be downloaded from the server without having to download a complete document, have alleviated many of these issues. However, it has also been surprising how much could be achieved with technologies dating from 1993, over a decade before these later enhancements had evolved.

1.1 WebGrid

WebGrid was one of the first ports of a highly interactive application to operate as a web service (Gaines, 1995; Shaw, and Gaines, 1995). It was part of a series of ports across platforms and programming language of a suite of programs developed by Shaw (1978; 1980) for eliciting conceptual models from people. Kelly (1955) had developed *personal construct psychology* as a theory of the way in which people construe their past experience in order to anticipate future experience. He also describes a technique, the *repertory grid*, through which an individual's construct systems can be elicited, modeled and compared, and this has been widely used to elicit conceptual models in a number of areas such as market research, clinical psychology, education, management, conflict resolution and knowledge acquisition. In 1975 Shaw automated and enhanced the process of elicitation through computer programs that used continuous online analysis to guide the elicitation process, and presented the results of grid analyses and comparisons in a graphic form.

Shaw's programs were originally written for the PDP12 and later ported to a PPD10 timesharing service, the Apple II, Macintosh and IBM PC to make them more widely available for remote access and on personal computers. In 1980 it was suggested (Gaines, and Shaw, 1980) that they could be used to elicit knowledge from experts in developing knowledge-based systems, and by the mid-1980s repertory grids had become a standard tool for knowledge acquisition for expert systems (Boose, 1984; Boose, 1986). A knowledge acquisition version with extended data types and rule induction, known as Knowledge Support System, KSS0 (Gaines, and Shaw, 1993a; Gaines, and Shaw, 1993b), was licensed to Neuron Data and sold by them as *Nextra*, a knowledge engineering front-end to their expert system shell, *Nexpert*.

The development of Andreessen's Mosaic web browser in 1993 supporting embedded user interface widgets for inline images and interactive forms made it feasible to offer the functionality of KSS0/Nextra across the Internet. This was attractive because there was demand for a port of KSS0/Nextra to Microsoft Windows, knowledge acquisition was generally a distributed team effort, and our integrated knowledge acquisition and modeling system involved extensive hypermedia support to manage documents, audio transcripts and images as knowledge sources (Gaines, Rappaport, and Shaw, 1992; Gaines, and Shaw, 1993a). The release of Mosaic for Unix in 1993 and for Windows and Macintosh in 1994 meant that users on most platforms could access web-based knowledge acquisition tools through similar interfaces across a common network with extensive multimedia support.

The development of Schotton's MacHTTP web server in 1993 enabled us to experiment with the use of a new release of KSS0, KSSn ported to C++, as a server extension through its common gateway interface (CGI). The only major code additions required was a module to convert the graphic outputs to graphic interchange format (GIF) images and another to manage the web page generation and form decoding. The initial prototype had some clumsy features in the user interface because different "Submit" buttons were not distinguishable in early versions of Mosaic, but once the HTML+/HTML level 2 specifications were developed and supported it became possible to offer conceptual modeling as a web service. Early in 1995 WebGrid was made available at http://tiger.cpsc.ucalgary.ca and has continued to operate as a 24/7 service ever since.

The initial usage was much higher than we had expected as users in a wide range of disciplines discovered that a conceptual modeling service was freely available on the web. In the period July-December 1995 WebGrid was accessed from 674 different sites in 30 countries. However, we had little contact with individual users since the service was robust and the online documentation appeared adequate. Occasionally users contacted us for advice and we learned more of their particular activities. For example, a graduate student in Holland studying operators' models of nuclear reactors who came across the grid elicitation system on the web in September 1995, used it with his experts to elicit their conceptual models, and included these in an additional chapter in his thesis for examination in October 1995; the web certainly accelerates research processes! Since then we know of over forty masters and doctoral theses around the world that have used WebGrid for their primary data collection, and the service has been accessed from over 20,000 user sites.

WebGrid has evolved through four major releases, each with substantial increases in functionality but all backwards compatible so ongoing users have not had to change their mode of usage unless they wish to do so. To support universal access from sites world-wide in the early years when there were major discrepancies between browser implementations, the first two releases in 1994 and 1998 were designed to rely on only HTML level 2 capabilities. WebGrid III released in 2001 made use of basic Javascript functionality to enhance the interactivity of the user interface. WebGrid IV was completely re-coded in 2006 to make full use of the latest standardized capabilities supported by a wide range of modern browsers, notably Unicode, HTML level 4, DHTML, Ajax and the Prototype cross-browser Javascript framework (Prototype, 2006). The server is script-driven with scripts in a full programming language that is incrementally compiled, so that end-users can customize its operation as they see fit.

This article discusses the design objectives established for WebGrid, their implementation, operator and user experience with a long-term web service, the evolution of the service as web technology and user requirements themselves evolved, and the current state of the art.

2 WebGrid Design Objectives

There were three major design objectives for WebGrid: in terms of overall utility, to make available through the web all of the functionality of KSS0/Nextra; in terms of software engineering, to ensure the reliability, lack of memory leaks, etc, necessary for continuous 24/7 operation and to be able to sustain this through the evolution of the service; in terms of human factors, to support users having minimal computer literacy while also enabling advanced users to customize the application and integrate it with their own applications, all this with the minimum of individual support. We were acutely conscious that making a complex interactive application open to anyone with web access might lead to a deluge of support requests, and that we needed to be highly defensive against this possibility.

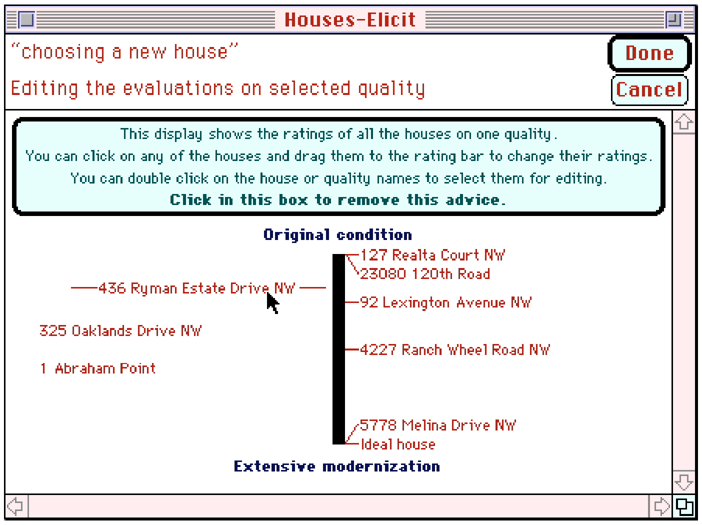
The underlying application had already been ported across eight platforms with different operating systems and to three different programming languages, and each time we had done so it had been refactored for increasing modularity. The object-oriented implementation of KSSn in C++ made the software engineering objectives relatively easy to achieve.

2.1 Porting Rating Scale Functionality

Porting the full functionality at first appeared problematic because repertory grid elicitation involves eliciting the *constructs*, the conceptual dimensions of a person's models of experience, by rating *elements*, entities in the domain of experience under investigation, along a rating scale. A major feature of KSS0/Nextra was a specialist rating scale widget that enabled this to be done by dragging elements to locations along construct dimensions as shown in Figure 1.

However, the popup menus available in HTML 2 provided similar functionality in that one could show the points on the scale and allow the user to drag the cursor to an appropriate rating as shown in Figure

2. This proved to be simple and natural to understand, particularly as popup menus came into common usage in most applications, and the advantages of a specialist widget became balanced by those of user familiarity with a standard widget. It also proved advantageous when we extended the repertory grid for purposes of knowledge-based system development with additional data types such as labeled categories, numbers, dates, etc, and found it simple to support them through HTML menus and text input fields.

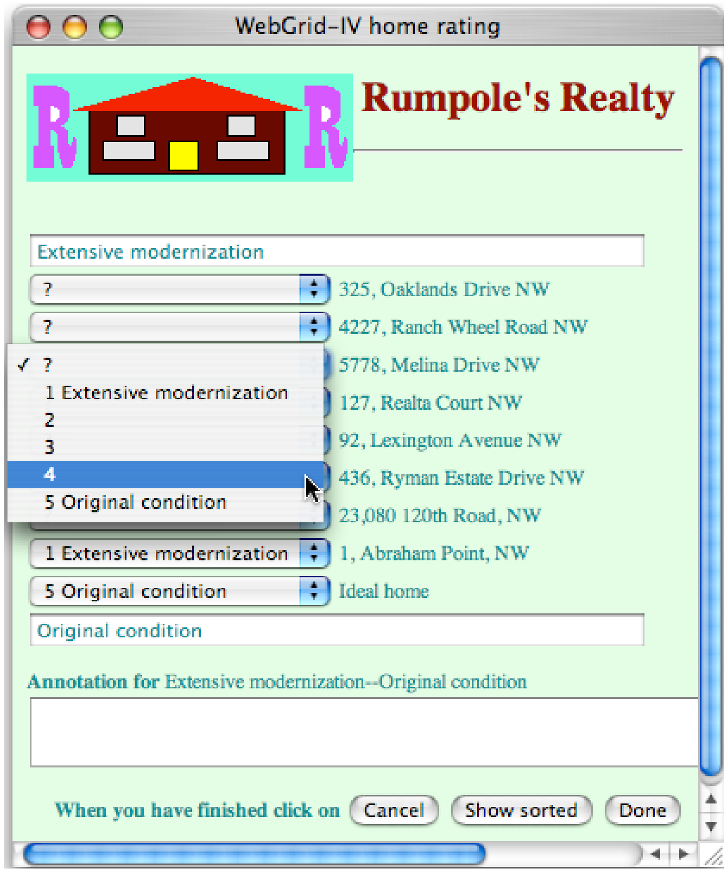

Houses-Elicit

"choosing a new house"

Editing the evaluations on selected quality

This display shows the ratings of all the houses on one quality.
You can click on any of the houses and drag them to the rating bar to change their ratings.
You can double click on the house or quality names to select them for editing.
Click in this box to remove this advice.

Figure 1. KSS0/Nextra click and drag rating scale interface


WebGrid-IV home rating

Rumpole's Realty

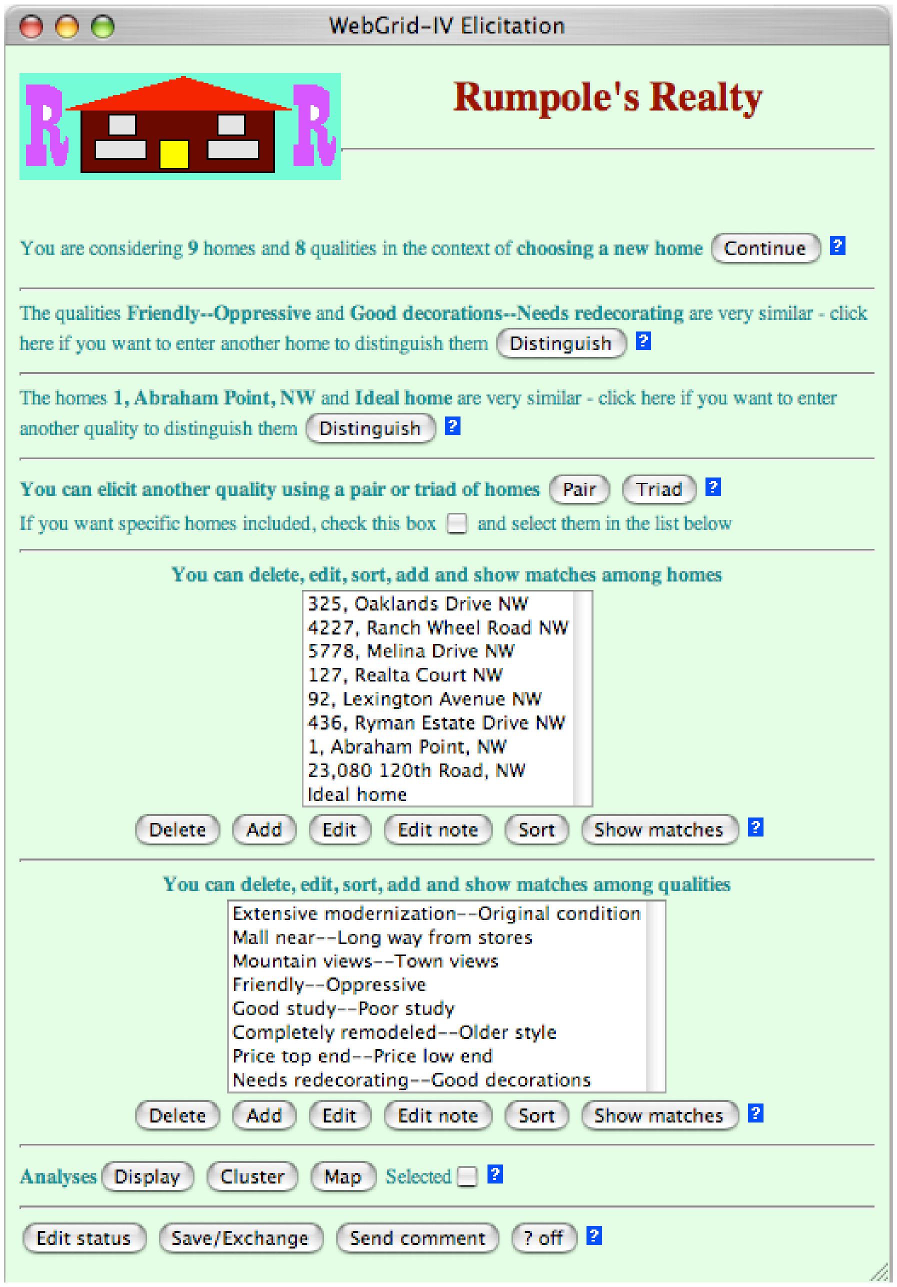
Extensive modernization	
?	325, Oaklands Drive NW
?	4227, Ranch Wheel Road NW
✓ ?	5778, Melina Drive NW
1 Extensive modernization	127, Realta Court NW
2	92, Lexington Avenue NW
3	
4	436, Ryman Estate Drive NW
5 Original condition	23,080 120th Road, NW
1 Extensive modernization	1, Abraham Point, NW
5 Original condition	Ideal home
Original condition	

Annotation for Extensive modernization--Original condition

Figure 2. WebGrid click and drag rating scale interface

2.2 Context-sensitive Help

We had intended our user community originally to be those already familiar with our standalone application on a personal computer. However, it was quickly apparent that WebGrid was developing a new community of users for whom it was their first introduction to repertory grid techniques, and that there was a need for online support integrated with the application. For WebGrid II, we developed a context-sensitive 'help' module that was called through a small "?" icon displayed in each significant sub-section of the user interface as shown in Figure 3.


WebGrid-IV Elicitation

Rumpole's Realty

You are considering **9** homes and **8** qualities in the context of **choosing a new home** Continue ?

The qualities **Friendly--Oppressive** and **Good decorations--Needs redecorating** are very similar - click here if you want to enter another home to distinguish them Distinguish ?

The homes **1, Abraham Point, NW** and **Ideal home** are very similar - click here if you want to enter another quality to distinguish them Distinguish ?

You can elicit another quality using a pair or triad of homes Pair Triad ?

If you want specific homes included, check this box ☐ and select them in the list below

You can delete, edit, sort, add and show matches among homes

325, Oaklands Drive NW
4227, Ranch Wheel Road NW
5778, Melina Drive NW
127, Realta Court NW
92, Lexington Avenue NW
436, Ryman Estate Drive NW
1, Abraham Point, NW
23,080 120th Road, NW
Ideal home

Delete Add Edit Edit note Sort Show matches ?

You can delete, edit, sort, add and show matches among qualities

Extensive modernization--Original condition
Mall near--Long way from stores
Mountain views--Town views
Friendly--Oppressive
Good study--Poor study
Completely remodeled--Older style
Price top end--Price low end
Needs redecorating--Good decorations

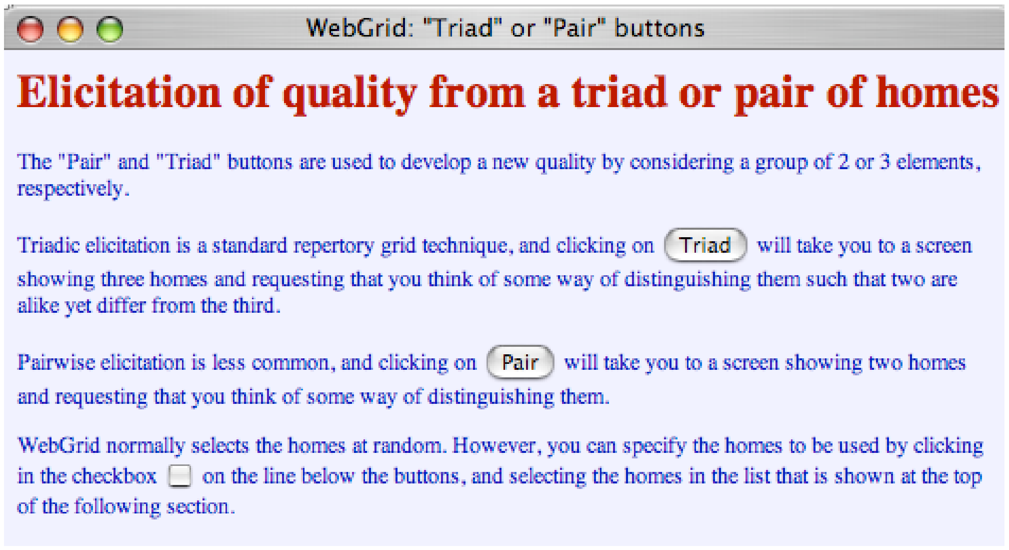
Delete Add Edit Edit note Sort Show matches ?

Analyses Display Cluster Map Selected ☐ ?

Edit status Save/Exchange Send comment ? off ?

Figure 3. Integrated context-sensitive help

For example, if the "?" icon next to the "Triad" button in the fourth section down is pressed then a help screen with relevant information pops up as shown in Figure 4.


WebGrid: "Triad" or "Pair" buttons

Elicitation of quality from a triad or pair of homes

The "Pair" and "Triad" buttons are used to develop a new quality by considering a group of 2 or 3 elements, respectively.

Triadic elicitation is a standard repertory grid technique, and clicking on Triad will take you to a screen showing three homes and requesting that you think of some way of distinguishing them such that two are alike yet differ from the third.

Pairwise elicitation is less common, and clicking on Pair will take you to a screen showing two homes and requesting that you think of some way of distinguishing them.

WebGrid normally selects the homes at random. However, you can specify the homes to be used by clicking in the checkbox ☐ on the line below the buttons, and selecting the homes in the list that is shown at the top of the following section.

Figure 4. A help window

Some of the help screens contain links to online documentation and published papers enabling users to investigate issues in greater depth if they wish.

2.3 Support of the "Back" Button as "Undo"

As is apparent in Figure 3, WebGrid provides the user with a rich range of complex functionality, which can be daunting if users feel that any of their actions may be irreversible. We needed to provide them with the reassurance of an unlimited "undo" capability whereby they could revert to any previous state. The natural way to do this in a web browser is to support the *back* button and *history* menu as *undo* capabilities. That is, if one goes back to a previous screen then the state of the system is precisely that of the system when that screen was created.

We achieved this by fully adopting the stateless protocol of HTTP in that all the data elicited from the user for the repertory grid was stored in hidden fields in the document on the screen. This was feasible because even quite large grids involved only a few kilobytes of data. If the datasets had been larger then the use of a hidden field for a state-identifier linking back to state dumps at the server would have achieved the same end, but it was not necessary in this application.

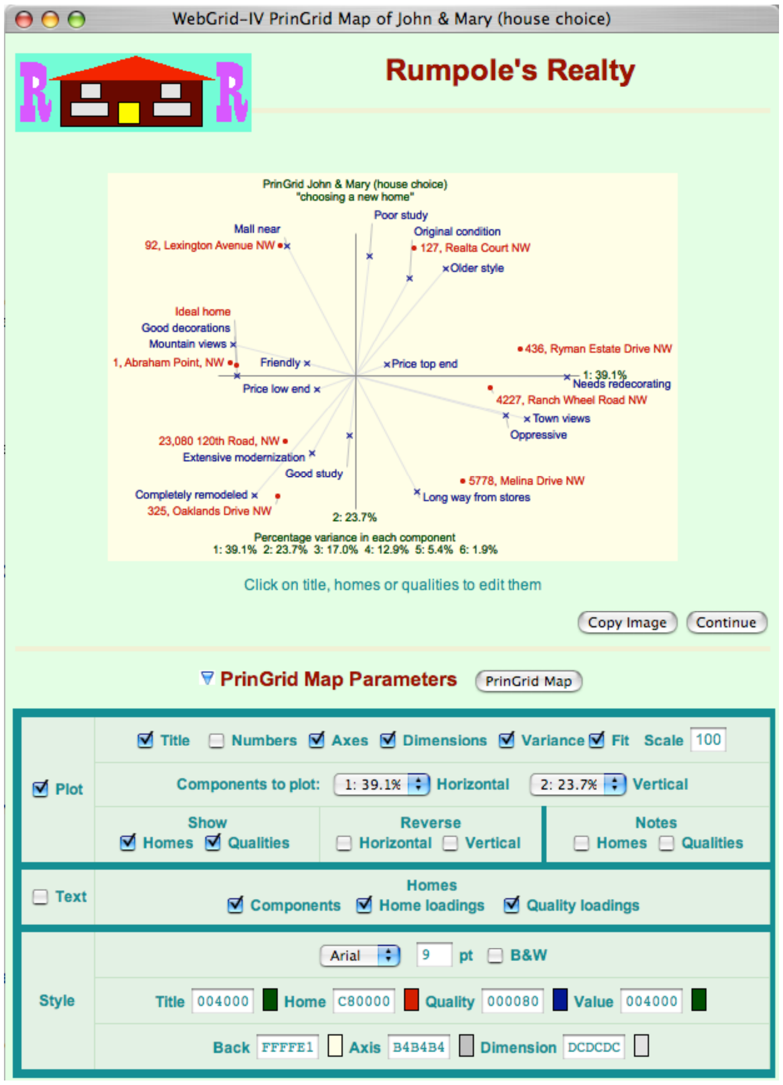
Some years later we found that the scalability of this approach was being severely tested by a few users who were entering very large datasets into WebGrid for data analysis purposes. However, by that time the browsers and server were coping with over 100 kilobytes of hidden field data without problems.

Some interesting design decisions became necessary as we began to enhance client-side interactivity, initially with Javascript and in WebGrid IV with Ajax techniques. Changes could be made to the local document that could only be undone as a whole by use of the back button. However, such a situation existed in the earliest versions since, for example, in the rating screen shown in Figure any ratings entered are lost when one reverts to a previous screen. That is, our use of the back button was more akin to that of the cancel button shown in Figure 2 in that it undid all changes made since the new page was loaded. This behavior had been accepted by our users without comment or problems, and, from discussions, with a sample of them it was clear that the user model of WebGrid interaction was that each page was, in a sense, a modal dialog that once either completed to effect a change, or cancelled with no effect.

We have supported this user model in WebGrid IV development and ensured that each of the twenty WebGrid interface documents is distinctive and its functionality clear and monolithic, a transaction-oriented model. The back button essentially cancels an ongoing transaction with no side-effects. This model is also supported from a software engineering perspective in that each document is managed through its own script with one function that generates it and another that analyzes the data returned from it and carries out the transaction.

2.4 Porting Graphic Functionality

A major feature of KSS0/Nextra was the availability of graphic analyses of the data showing the conceptual structures elicited in a readily understandable form. We supported these in WebGrid by converting the outputs to GIF (later PNG) format, caching them at the server, and embedding the cached images in the documents returned to the user as shown in Figure 5. Profiling the WebGrid I system showed that the image generation was the dominant use of processor resources, and we wrote a graphics to GIF generator with highly optimized inner loops that reduced the time to a fraction of a second even on the processors of 1995. Nowadays it is no longer an issue.



Percentage variance in each component
1: 39.1% 2: 23.7% 3: 17.0% 4: 12.9% 5: 5.4% 6: 1.9%

Click on title, homes or qualities to edit them

PrinGrid Map Parameters

☑ Plot	☑ Title ☐ Numbers ☑ Axes ☑ Dimensions ☑ Variance ☑ Fit Scale 100		
	Components to plot: 1: 39.1% Horizontal 2: 23.7% Vertical		
	Show ☑ Homes ☑ Qualities	Reverse ☐ Horizontal ☐ Vertical	Notes ☐ Homes ☐ Qualities
☐ Text	Homes ☑ Components ☑ Home loadings ☑ Quality loadings		
Style	Arial 9 pt ☐ B&W		
	Title 004000 🟩 Home C80000 🟥 Quality 000080 🟦 Value 004000 🟩		
	Back FFFFE1 ⬜ Axis B4B4B4 ▨ Dimension DCDCDC ⬜		

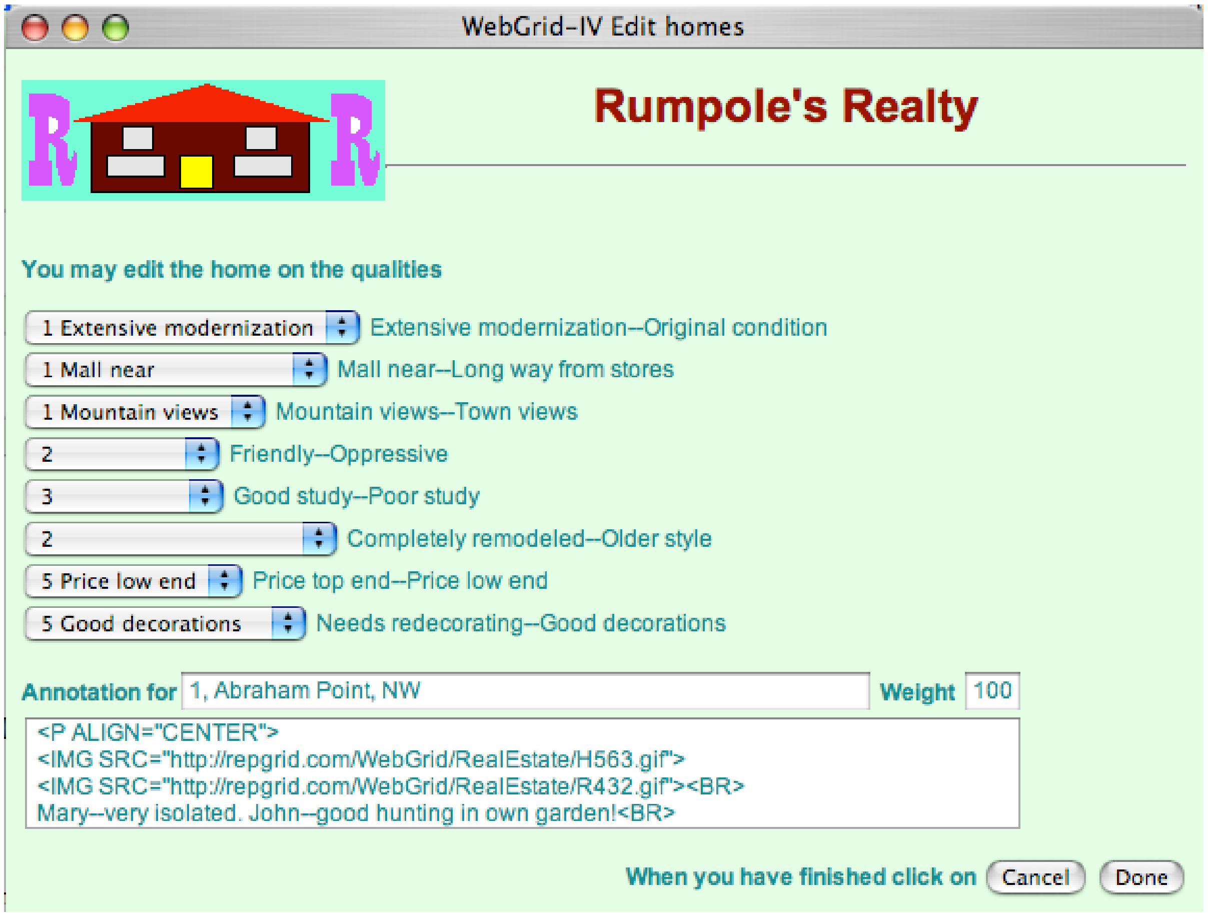
Figure 5. Graphic data analysis

The analyses were made interactive by embedding them in the forms and using the clickable input images, and using the x-y data returned from them to return to the user the appropriate document for editing the data associated with the element or construct that had been clicked.

The graphic output in Figure 5 was generated in RepNet, the interactive vector graphics component of Rep IV. In the stand-alone application the user is able to drag the labels to different positions in order to improve the appearance beyond that provided by the automatic layout algorithm. For WebGrid the graphic output is converted to a portable network graphics (PNG) file, and the only interactivity we have so far been able to provide is that clicking elements and constructs in the image accesses the appropriate editors as noted above. There are other components of Rep IV that require full user interaction with the graphics in a way that web technology until very recently has not supported. Section 4 discusses some of these components and our plans to port them.

2.5 Supporting Multimedia Annotation

We had supported multimedia annotation in KSS0/Nextra through links to material in separate hypermedia applications such as HyperCard. In WebGrid II we used the inbuilt hypermedia support of web browsers by allowing users to annotate the data they entered with comments that were then embedded in the document where appropriate. Figure 6 shows the entry of such annotation (where the use of HTML to embed images may require facilitator support), and Figure 7 its effect in an elicitation document.



You may edit the home on the qualities

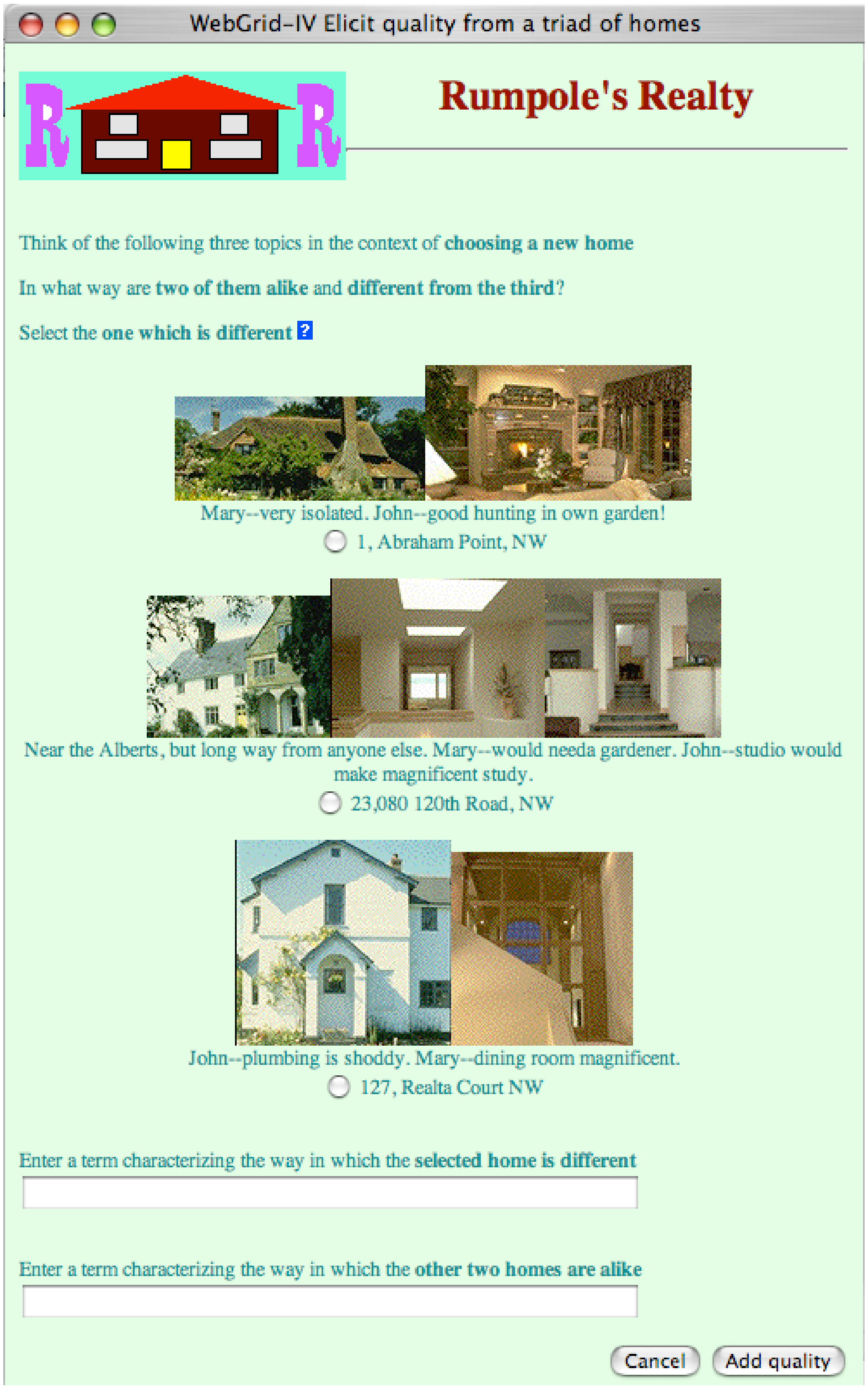
1 Extensive modernization	Extensive modernization--Original condition
1 Mall near	Mall near--Long way from stores
1 Mountain views	Mountain views--Town views
2	Friendly--Oppressive
3	Good study--Poor study
2	Completely remodeled--Older style
5 Price low end	Price top end--Price low end
5 Good decorations	Needs redecorating--Good decorations

Annotation for 1, Abraham Point, NW Weight 100

```
<P ALIGN="CENTER">
<IMG SRC="http://repgrid.com/WebGrid/RealEstate/H563.gif">
<IMG SRC="http://repgrid.com/WebGrid/RealEstate/R432.gif"><BR>
Mary--very isolated. John--good hunting in own garden!<BR>
```

When you have finished click on Cancel Done

Figure 6. Hypermedia annotation in HTML



Think of the following three topics in the context of **choosing a new home**

In what way are **two of them alike** and **different from the third**?

Select the **one which is different** ?

Mary--very isolated. John--good hunting in own garden!

( ) 1, Abraham Point, NW

Near the Alberts, but long way from anyone else. Mary--would needa gardener. John--studio would make magnificent study.

( ) 23,080 120th Road, NW

John--plumbing is shoddy. Mary--dining room magnificent.

( ) 127, Realta Court NW

Enter a term characterizing the way in which the **selected home is different**

Enter a term characterizing the way in which the **other two homes are alike**

Figure 7. Hypermedia annotations embedded in a document

The implementation of this feature involved dynamically encoding the annotation in three different formats: one for storing in the hidden fields; another for display as editable text; and another for use as embedded HTML.

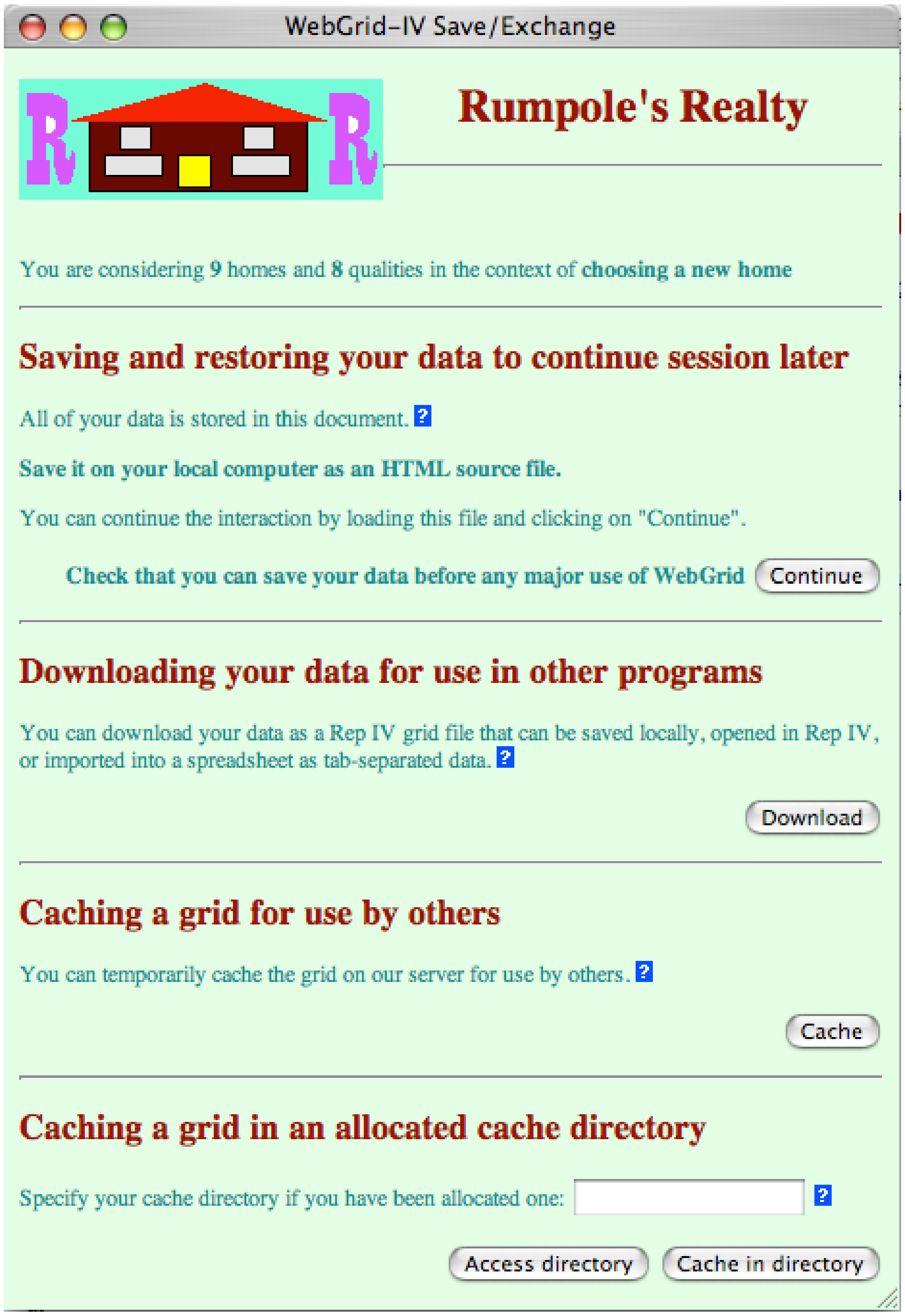
2.6 Saving Data

Our users needed to be able to save, edit and analyze their saved data, but we did not wish to manage a central storage facility requiring them to register and requiring us to be responsible for their data. This requirement was addressed through the same mechanism as that described in the previous section. Since all the data was stored in the current document, it was only necessary to save that document as HTML on the local machine to preserve the data locally. Opening the document at any time re-established the link to the server and enabled the elicitation to continue or the data to be edited and analyzed. We also made the Rep IV standalone application able to open the saved web documents and extract the data from them, so that data collected through the web could be analyzed together with data collected by other means.

Interestingly, the very simplicity of this way of saving data led to this feature requiring the most support in terms of user explanation. In most web applications users do not need to save the HTML documents on their local machines, and the notion that doing so with WebGrid documents would preserve their data was not a natural one. Once experienced it became a simple feature, but it was not the obvious thing to do.

There were some features of KSS0/Nextra that did require data storage at the server, notably the capability to compare conceptual models in two different repertory grids. We supported this in WebGrid II, as shown in Figure 8, by allowing a grid to be stored in a cache at our server for later use, automatically returning to the user a uniform resource locator (URL) containing a unique identifier for the grid. We specified that the cached data would be kept for 90 days but might expire thereafter, but in practice have never deleted such cached data.

As our user community evolved it became apparent that we needed further support for some major users who were using WebGrid to elicit conceptual models from distributed communities through the Internet, and who needed some form of centralized automatic data collection at the server. We again wanted to be able to do this without having to provide explicit control or support. Hence in WebGrid III, released in early 1998, we added the facility for users to register their own password-protected cache directories at our server, and to save and access grids in them.


WebGrid-IV Save/Exchange

Rumpole's Realty

You are considering **9** homes and **8** qualities in the context of **choosing a new home**

Saving and restoring your data to continue session later

All of your data is stored in this document. ?

Save it on your local computer as an HTML source file.

You can continue the interaction by loading this file and clicking on "Continue".

Check that you can save your data before any major use of WebGrid Continue

Downloading your data for use in other programs

You can download your data as a Rep IV grid file that can be saved locally, opened in Rep IV, or imported into a spreadsheet as tab-separated data. ?

Download

Caching a grid for use by others

You can temporarily cache the grid on our server for use by others. ?

Cache

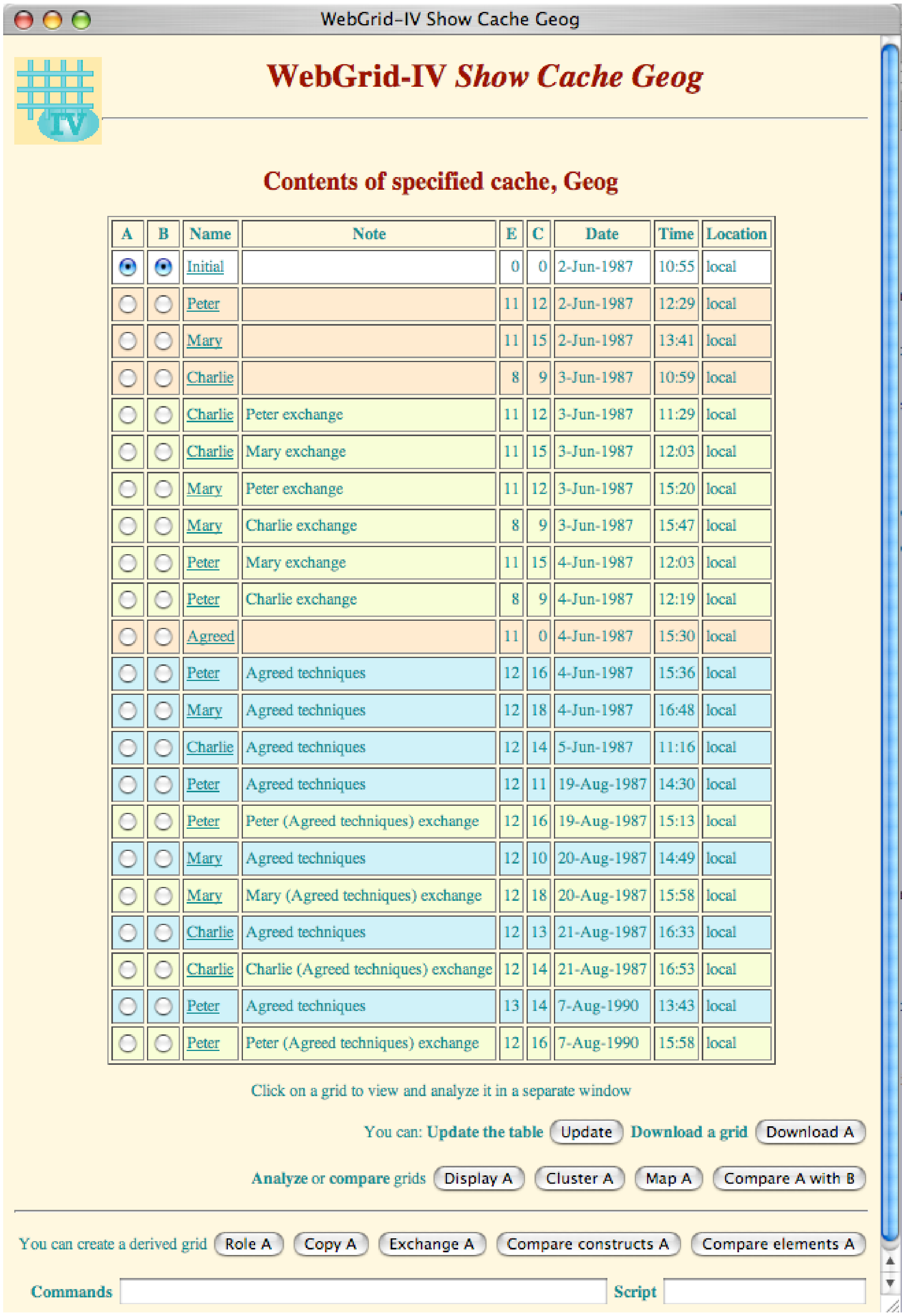
Caching a grid in an allocated cache directory

Specify your cache directory if you have been allocated one: ?

Access directory | Cache in directory

Figure 8. Facilities for saving and caching data

We programmed WebGrid III such that if a cached grid was used as the basis of further elicitation then the resultant grid was automatically added to the cache. We also provided multiple grid analysis facilities available when the primary user accessed his or her password-protected cache as shown in Figure 9. This supported automatic data collection from remote users, and protected access and analysis of the collected data. This facility has been widely used in a number of market research and other group conceptual modeling studies, and has been particularly attractive to researchers wishing to collect data from geographically dispersed sites.



WebGrid-IV *Show Cache Geog*

Contents of specified cache, Geog

A	B	Name	Note	E	C	Date	Time	Location
●	●	Initial		0	0	2-Jun-1987	10:55	local
○	○	Peter		11	12	2-Jun-1987	12:29	local
○	○	Mary		11	15	2-Jun-1987	13:41	local
○	○	Charlie		8	9	3-Jun-1987	10:59	local
○	○	Charlie	Peter exchange	11	12	3-Jun-1987	11:29	local
○	○	Charlie	Mary exchange	11	15	3-Jun-1987	12:03	local
○	○	Mary	Peter exchange	11	12	3-Jun-1987	15:20	local
○	○	Mary	Charlie exchange	8	9	3-Jun-1987	15:47	local
○	○	Peter	Mary exchange	11	15	4-Jun-1987	12:03	local
○	○	Peter	Charlie exchange	8	9	4-Jun-1987	12:19	local
○	○	Agreed		11	0	4-Jun-1987	15:30	local
○	○	Peter	Agreed techniques	12	16	4-Jun-1987	15:36	local
○	○	Mary	Agreed techniques	12	18	4-Jun-1987	16:48	local
○	○	Charlie	Agreed techniques	12	14	5-Jun-1987	11:16	local
○	○	Peter	Agreed techniques	12	11	19-Aug-1987	14:30	local
○	○	Peter	Peter (Agreed techniques) exchange	12	16	19-Aug-1987	15:13	local
○	○	Mary	Agreed techniques	12	10	20-Aug-1987	14:49	local
○	○	Mary	Mary (Agreed techniques) exchange	12	18	20-Aug-1987	15:58	local
○	○	Charlie	Agreed techniques	12	13	21-Aug-1987	16:33	local
○	○	Charlie	Charlie (Agreed techniques) exchange	12	14	21-Aug-1987	16:53	local
○	○	Peter	Agreed techniques	13	14	7-Aug-1990	13:43	local
○	○	Peter	Peter (Agreed techniques) exchange	12	16	7-Aug-1990	15:58	local

Click on a grid to view and analyze it in a separate window

You can: **Update the table** Update **Download a grid** Download A

Analyze or **compare** grids Display A Cluster A Map A Compare A with B

You can create a derived grid Role A Copy A Exchange A Compare constructs A Compare elements A

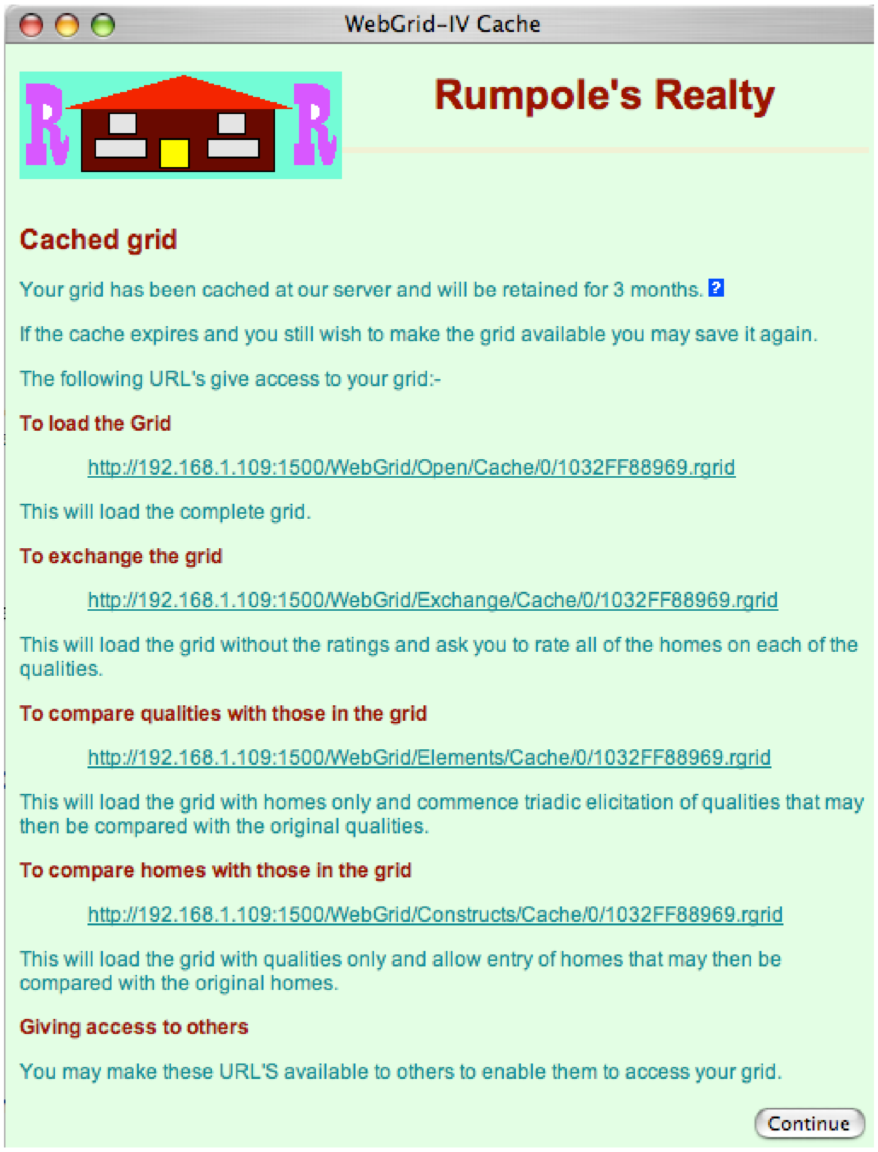
Commands **Script**

Figure 9. Access to cached data

2.7 Supporting Collaboration

One of the most important forms of grid analysis is the comparison of two or more grids from different people construing the same domain or from the same person at different times as their construct system develops and changes. Our standalone application provided extensive facilities for the graphic presentation of grid comparisons, and these were ported to WebGrid I to support some of its earliest applications to facilitating collaborative learning (Shaw, and Gaines, 1995).

As shown in Figure 8, the Save page offers the capability to cache a grid for others, and when this option is selected the grid is cached and the user is presented with a set of URL's to access the cached grid in various forms as shown in Figure 10. These may be copied and made available in web pages or sent by email, so that other people can be invited to develop related grids and compare them with those of others to see the similarities and differences in their construct systems. This capability has been used extensively to support collaborative learning in distributed communities.



Cached grid

Your grid has been cached at our server and will be retained for 3 months.

If the cache expires and you still wish to make the grid available you may save it again.

The following URL's give access to your grid:-

To load the Grid

http://192.168.1.109:1500/WebGrid/Open/Cache/0/1032FF88969.rgrid

This will load the complete grid.

To exchange the grid

http://192.168.1.109:1500/WebGrid/Exchange/Cache/0/1032FF88969.rgrid

This will load the grid without the ratings and ask you to rate all of the homes on each of the qualities.

To compare qualities with those in the grid

http://192.168.1.109:1500/WebGrid/Elements/Cache/0/1032FF88969.rgrid

This will load the grid with homes only and commence triadic elicitation of qualities that may then be compared with the original qualities.

To compare homes with those in the grid

http://192.168.1.109:1500/WebGrid/Constructs/Cache/0/1032FF88969.rgrid

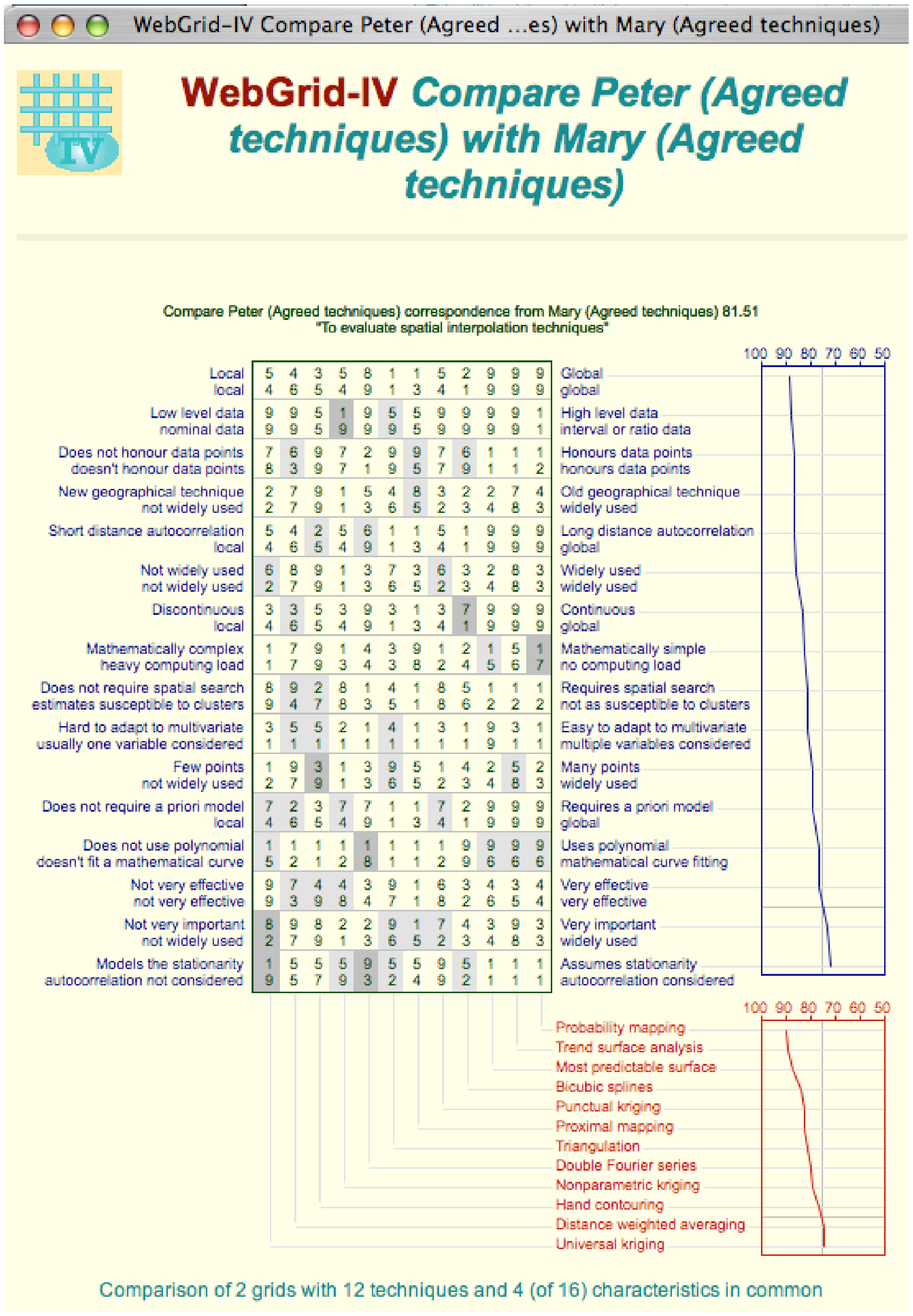
This will load the grid with qualities only and allow entry of homes that may then be compared with the original homes.

Giving access to others

You may make these URL'S available to others to enable them to access your grid.

Figure 10. Accessing a cached grid

Grid comparisons are also significant research tools in studying various communities of practice where commonalities are expected in construing but their extent may be difficult to assess (Gaines, and Shaw, 1994; Shaw, and Gaines, 1999). The support of user-managed caches introduced in WebGrid III enables a facilitator or researcher to collect grids automatically from a distributed community and analyze them online for discussion within that community. The cached data shown in Figure 9 was gathered in a study of a small professional community of geographers studying spatial mapping techniques. The radio buttons on the left enable any pair of grids to be selected and compared as illustrated in Figure 11.



WebGrid-IV Compare Peter (Agreed ...es) with Mary (Agreed techniques)

WebGrid-IV Compare Peter (Agreed techniques) with Mary (Agreed techniques)

Compare Peter (Agreed techniques) correspondence from Mary (Agreed techniques) 81.51
"To evaluate spatial interpolation techniques"

	Universal kriging	Distance weighted averaging	Hand contouring	Nonparametric kriging	Double Fourier series	Triangulation	Proximal mapping	Punctual kriging	Bicubic splines	Most predictable surface	Trend surface analysis	Probability mapping	
Local	5	4	3	5	8	1	1	5	2	9	9	9	Global
local	4	6	5	4	9	1	3	4	1	9	9	9	global
Low level data	9	9	5	1	9	5	5	9	9	9	9	1	High level data
nominal data	9	9	5	9	9	9	5	9	9	9	9	1	interval or ratio data
Does not honour data points	7	6	9	7	2	9	9	7	6	1	1	1	Honours data points
doesn't honour data points	8	3	9	7	1	9	5	7	9	1	1	2	honours data points
New geographical technique	2	7	9	1	5	4	8	3	2	2	7	4	Old geographical technique
not widely used	2	7	9	1	3	6	5	2	3	4	8	3	widely used
Short distance autocorrelation	5	4	2	5	6	1	1	5	1	9	9	9	Long distance autocorrelation
local	4	6	5	4	9	1	3	4	1	9	9	9	global
Not widely used	6	8	9	1	3	7	3	6	3	2	8	3	Widely used
not widely used	2	7	9	1	3	6	5	2	3	4	8	3	widely used
Discontinuous	3	3	5	3	9	3	1	3	7	9	9	9	Continuous
local	4	6	5	4	9	1	3	4	1	9	9	9	global
Mathematically complex	1	7	9	1	4	3	9	1	2	1	5	1	Mathematically simple
heavy computing load	1	7	9	3	4	3	8	2	4	5	6	7	no computing load
Does not require spatial search	8	9	2	8	1	4	1	8	5	1	1	1	Requires spatial search
estimates susceptible to clusters	9	4	7	8	3	5	1	8	6	2	2	2	not as susceptible to clusters
Hard to adapt to multivariate	3	5	5	2	1	4	1	3	1	9	3	1	Easy to adapt to multivariate
usually one variable considered	1	1	1	1	1	1	1	1	1	9	1	1	multiple variables considered
Few points	1	9	3	1	3	9	5	1	4	2	5	2	Many points
not widely used	2	7	9	1	3	6	5	2	3	4	8	3	widely used
Does not require a priori model	7	2	3	7	7	1	1	7	2	9	9	9	Requires a priori model
local	4	6	5	4	9	1	3	4	1	9	9	9	global
Does not use polynomial	1	1	1	1	1	1	1	1	9	9	9	9	Uses polynomial
doesn't fit a mathematical curve	5	2	1	2	8	1	1	2	9	6	6	6	mathematical curve fitting
Not very effective	9	7	4	4	3	9	1	6	3	4	3	4	Very effective
not very effective	9	3	9	8	4	7	1	8	2	6	5	4	very effective
Not very important	8	9	8	2	2	9	1	7	4	3	9	3	Very important
not widely used	2	7	9	1	3	6	5	2	3	4	8	3	widely used
Models the stationarity	1	5	5	5	9	5	5	9	5	1	1	1	Assumes stationarity
autocorrelation not considered	9	5	7	9	3	2	4	9	2	1	1	1	autocorrelation considered

Comparison of 2 grids with 12 techniques and 4 (of 16) characteristics in common

Figure 11. Comparing grids

2.8 Supporting Customization

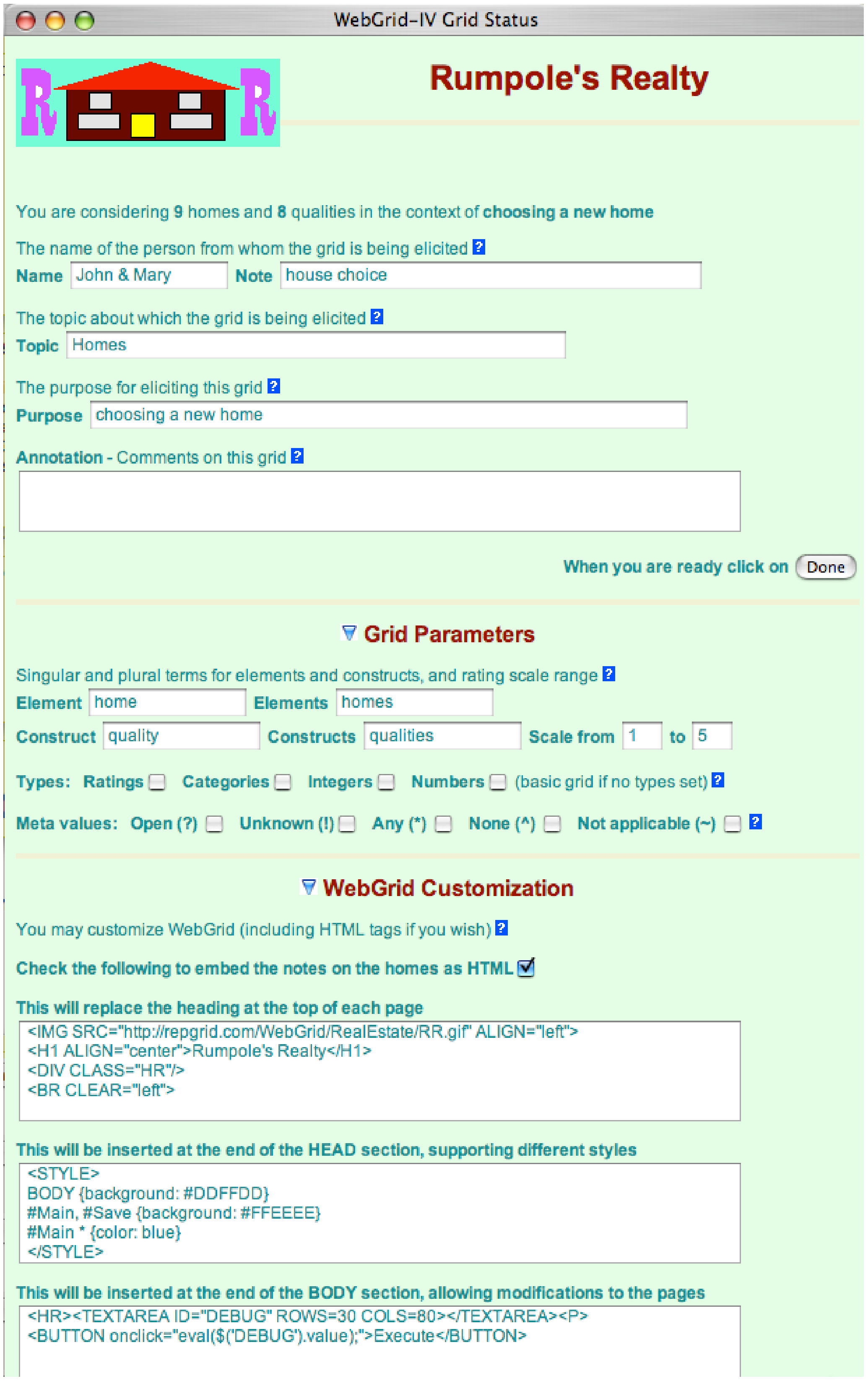
For WebGrid II, we wished to allow users to customize WebGrid in the general appearance of pages and with their own headers and trailers. The general status page allowed users to replace the BODY tag, which controls the background colour or image in each page, and the header and trailer of each WebGrid document with their own HTML code. This been used, for example, in the real estate pages shown above to change the background color and to substitute a customized header.

The BODY tag substitution was intended originally to support different background colors and images. However, it also enables user-specified HTML to be inserted at the end of the HEAD section and the beginning of the BODY section, and this has proved very powerful as HTML features have evolved. For example, when cascading style sheet support became available in browsers it was used to insert STYLE specifications customizing the interpretation of individual HTML tags, and Javascript supporting user-designed interactivity. This has made the consistency of usage of all HTML tags within WebGrid documents particularly important so that users specifying styles see precisely the effects they expect.

In WebGrid IV the BODY tag substitution has been generalized to the capability to insert arbitrary HTML in the HEAD and at the end of the BODY sections of every page. Each page has a distinct ID tag in its BODY statement and is designed such that every element within it is readily accessible through its class, its ID and through document object model (DOM) navigation. Through use of these facilities users can re-style all the WebGrid pages together or selectively, and incorporate new elements and functionality as they see fit, purely through the interface with no access to the underlying scripts or application code. Use of the Prototype (2006) Javascript framework buffers users from inconsistencies in DOM functionality across browsers.

Additional data collection can be incorporated in an integrated fashion by using these facilities to include additional DOM input elements whose names begin with an underline character, since the WebGrid server treats such data as part of the grid meta data and preserves it through all transactions, including caching and data downloads/uploads.

Figure 12 shows the customization fields in use for replacing the header, restyling pages selectively, and adding an additional field for debugging Javascript interacting with the DOM by adding a text area at the bottom of each page together with an “Execute” button that will execute any Javascript typed into that text area.


WebGrid-IV Grid Status

Rumpole's Realty

You are considering **9** homes and **8** qualities in the context of **choosing a new home**

The name of the person from whom the grid is being elicited ?

Name John & Mary **Note** house choice

The topic about which the grid is being elicited ?

Topic Homes

The purpose for eliciting this grid ?

Purpose choosing a new home

Annotation - Comments on this grid ?

When you are ready click on Done

Grid Parameters

Singular and plural terms for elements and constructs, and rating scale range ?

Element home **Elements** homes

Construct quality **Constructs** qualities **Scale from** 1 **to** 5

Types: Ratings ☐ **Categories** ☐ **Integers** ☐ **Numbers** ☐ (basic grid if no types set) ?

Meta values: Open (?) ☐ **Unknown (!)** ☐ **Any (*)** ☐ **None (^)** ☐ **Not applicable (~)** ☐ ?

WebGrid Customization

You may customize WebGrid (including HTML tags if you wish) ?

Check the following to embed the notes on the homes as HTML ☑

This will replace the heading at the top of each page

```
<IMG SRC="http://repgrid.com/WebGrid/RealEstate/RR.gif" ALIGN="left">
<H1 ALIGN="center">Rumpole's Realty</H1>
<DIV CLASS="HR"/>
<BR CLEAR="left">
```

This will be inserted at the end of the HEAD section, supporting different styles

```
<STYLE>
BODY {background: #DDFFDD}
#Main, #Save {background: #FFEEEE}
#Main * {color: blue}
</STYLE>
```

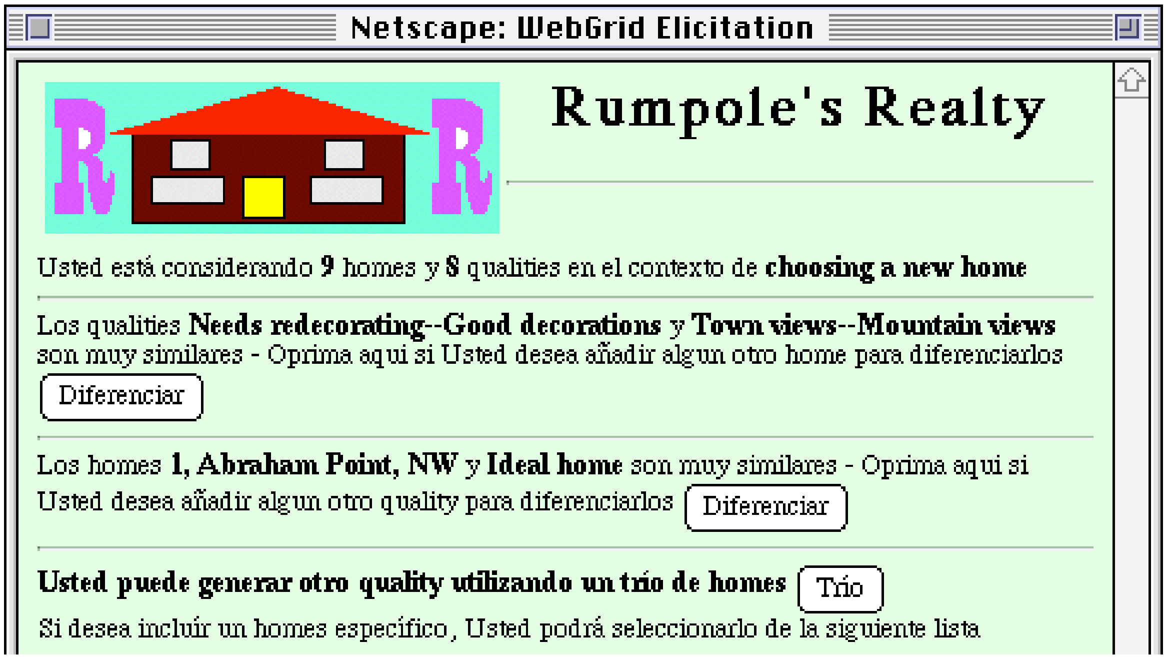
This will be inserted at the end of the BODY section, allowing modifications to the pages

```
<HR><TEXTAREA ID="DEBUG" ROWS=30 COLS=80></TEXTAREA><P>
<BUTTON onclick="eval($('DEBUG').value);">Execute</BUTTON>
```

Figure 12. Customizing WebGrid

Further customization is possible because the generation of web documents in WebGrid is from scripts that may be modified by the administrator of the server. The behavior of versions I through III was driven by scripts in a macro substitution language with variable data from the grid being available as macro parameters. WebGrid IV offers substantially more power since the entire server operation is programmed through administrator-accessible scripts in a full object-orientated programming language supported by an incremental compiler. The application programming interface (API) accessing all the grid functionality appears to the programmer as an integral library of specialist functions and datatypes. All of the HTML generation and analysis of GET and POST requests is programmed through scripts which may be modified by the server administrator. A wide range of general-purpose libraries are also available through the script language including an XML parser/generator, an SQL database manager, and graphic generation capabilities.

The family of scripts in use is specified as a parameter of the URL used to access WebGrid. This has enabled us to customize various versions of WebGrid for specific users with whom we are working closely, including translating the dialog into other languages. For example, Figure 13 shows the same grid as in Figure 3 with the URL amended to call a family of Spanish language scripts.



Usted está considerando **9** homes y **8** qualities en el contexto de **choosing a new home**

Los qualities **Needs redecorating--Good decorations** y **Town views--Mountain views** son muy similares - Oprima aqui si Usted desea añadir algun otro home para diferenciarlos

Los homes **1, Abraham Point, NW** y **Ideal home** son muy similares - Oprima aqui si Usted desea añadir algun otro quality para diferenciarlos

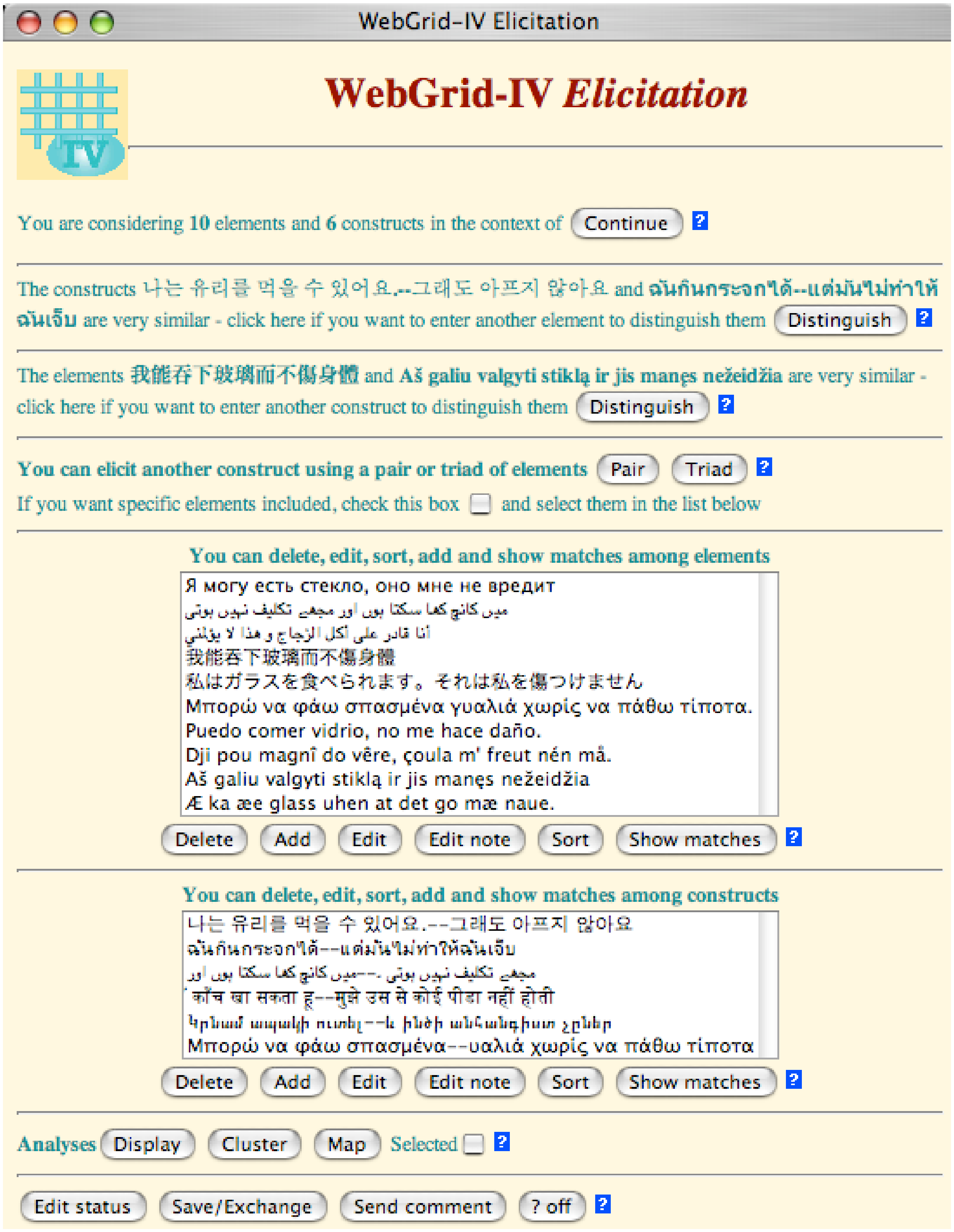
Usted puede generar otro quality utilizando un trio de homes

Si desea incluir un homes especifico, Usted podrá seleccionarlo de la siguiente lista

Figure 13. Customizing WebGrid through a script family

In recent years the availability of Unicode support in all the major browsers has made it feasible to use non-Roman languages in grids, and WebGrid IV was designed to be fully Unicode-compliant as illustrated in the test 'grid' incorporating many different fonts shown in Figure 14. This enables grid data to be entered in any language of the world, and the scripts for the interaction to be translated into non-Roman languages.

All these features have gone a long way to making WebGrid IV both a complete grid elicitation and analysis system, and an extremely flexible, open-architecture porting environment for interactive applications.


WebGrid-IV Elicitation

WebGrid-IV *Elicitation*

You are considering **10** elements and **6** constructs in the context of Continue ?

The constructs 나는 유리를 먹을 수 있어요.--그래도 아프지 않아요 and **ฉันกินกระจกได้--แต่มันไม่ทำให้ฉันเจ็บ** are very similar - click here if you want to enter another element to distinguish them Distinguish ?

The elements **我能吞下玻璃而不傷身體** and **Aš galiu valgyti stiklą ir jis manęs nežeidžia** are very similar - click here if you want to enter another construct to distinguish them Distinguish ?

You can elicit another construct using a pair or triad of elements Pair Triad ?

If you want specific elements included, check this box ☐ and select them in the list below

You can delete, edit, sort, add and show matches among elements

Я могу есть стекло, оно мне не вредит
میں کانچ کھا سکتا ہوں اور مجھے تکلیف نہیں ہوتی
أنا قادر على أكل الزجاج و هذا لا يؤلمني
我能吞下玻璃而不傷身體
私はガラスを食べられます。それは私を傷つけません
Μπορώ να φάω σπασμένα γυαλιά χωρίς να πάθω τίποτα.
Puedo comer vidrio, no me hace daño.
Dji pou magnî do vêre, çoula m' freut nén må.
Aš galiu valgyti stiklą ir jis manęs nežeidžia
Æ ka æe glass uhen at det go mæ naue.

Delete Add Edit Edit note Sort Show matches ?

You can delete, edit, sort, add and show matches among constructs

나는 유리를 먹을 수 있어요.--그래도 아프지 않아요
ฉันกินกระจกได้--แต่มันไม่ทำให้ฉันเจ็บ
مجھے تکلیف نہیں ہوتی --میں کانچ کھا سکتا ہوں اور
काँच खा सकता हूँ--मुझे उस से कोई पीडा नहीं होती
Կրնամ ապակի ուտել--և ինծի անհանգիստ չըներ
Μπορώ να φάω σπασμένα--υαλιά χωρίς να πάθω τίποτα

Delete Add Edit Edit note Sort Show matches ?

Analyses Display Cluster Map Selected ☐ ?

Edit status Save/Exchange Send comment ? off ?

Figure 14. Unicode support of non-Roman fonts

2.9 Integration with Other Applications

KSS0/Nextra was designed to integrate with knowledge-based and hypermedia systems in such a way that a user experienced a single, relatively seamless application (Gaines, and Linster, 1990; Gaines, Rappaport, and Shaw, 1992). We planned for WebGrid to offer similar capabilities and built facilities into the server for communication with other applications across the Internet. However, most of the integration was achieved by linking additional applications to WebGrid at the server. For example, we ran our rule induction (Gaines, 1994) and description-logic inference engines (Gaines, 1996) on the same server as WebGrid and linked them through inter-application communication protocols to provide a integrated knowledge acquisition and inference capability integrated with documents (Gaines, and Shaw, 1999) that has been widely used in computer science teaching (Gaines, and Shaw, 1997).

An interesting approach to integration with WebGrid was adopted by Tennison in the development of APECKS, tool for collaborative ontology construction (Tennison, O'Hara, and Shadbolt, 2002). APECKS, as well as providing internal knowledge acquisition (KA) support, was designed to interface with web-accessible KA tools, thereby allowing theoretically unlimited KA support for users. The sytem used WebGrid-II for external KA support, and the paper cited discusses issues involved in integrating APECKS and WebGrid in detail.

Tennison wished to show that her approach could be used to integrate web-based KA services with APECKS without any modification to those services, even if they had been designed primarily for human interaction and there was no collaboration with their designers. She packaged APECKS ontologies as if they were grids embedded in WebGrid forms and posted them to the WebGrid server using the HTTP protocol. She re-packaged the document returned as an APECKS document for presentation to her users. Figure 15 shows an example of WebGrid being used to compare two APECKS ontologies.

WebGrid IV incorporates script-driven XML encoders and decoders supporting the offering of web services targeted on computer-computer interaction that make it simpler to achieve the kind of integration that Tennison developed, as well as to offer access to a far wider range of web services based on the knowledge acquisition, modeling and management tools available on the WebGrid server. However, the capability that Tennison demonstrated, of being able to re-purpose one web service in order to integrate some of its capabilities into another web service even if the service used was not designed with this in mind, remains an important option.

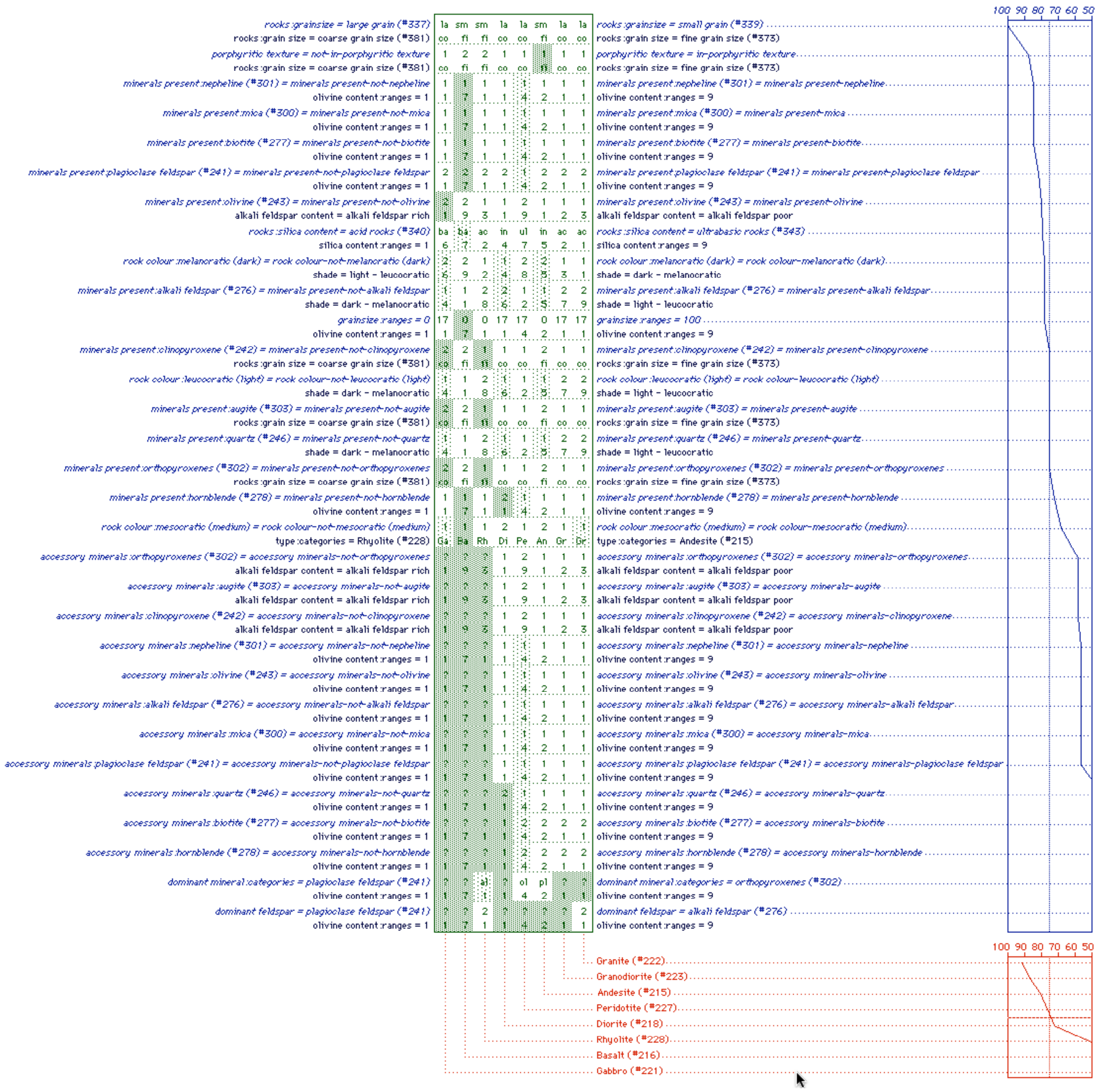

rocks :grainsize = large grain (#337) — rocks :grainsize = small grain (#339)
rocks :grain size = coarse grain size (#381) — rocks :grain size = fine grain size (#373)
porphyritic texture = not-in-porphyritic texture — porphyritic texture = in-porphyritic texture
minerals present :nepheline (#301) = minerals present-not-nepheline — minerals present :nepheline (#301) = minerals present-nepheline
olivine content :ranges = 1 — olivine content :ranges = 9
minerals present :mica (#300) = minerals present-not-mica — minerals present :mica (#300) = minerals present-mica
minerals present :biotite (#277) = minerals present-not-biotite — minerals present :biotite (#277) = minerals present-biotite
minerals present :plagioclase feldspar (#241) = minerals present-not-plagioclase feldspar — minerals present :plagioclase feldspar (#241) = minerals present-plagioclase feldspar
minerals present :olivine (#243) = minerals present-not-olivine — minerals present :olivine (#243) = minerals present-olivine
alkali feldspar content = alkali feldspar rich — alkali feldspar content = alkali feldspar poor
rocks :silica content = acid rocks (#340) — rocks :silica content = ultrabasic rocks (#343)
silica content :ranges = 1 — silica content :ranges = 9
rock colour :melanocratic (dark) = rock colour-not-melanocratic (dark) — rock colour :melanocratic (dark) = rock colour-melanocratic (dark)
shade = light - leucocratic — shade = dark - melanocratic
minerals present :alkali feldspar (#276) = minerals present-not-alkali feldspar — minerals present :alkali feldspar (#276) = minerals present-alkali feldspar
grainsize :ranges = 0 — grainsize :ranges = 100
minerals present :clinopyroxene (#242) = minerals present-not-clinopyroxene — minerals present :clinopyroxene (#242) = minerals present-clinopyroxene
rock colour :leucocratic (light) = rock colour-not-leucocratic (light) — rock colour :leucocratic (light) = rock colour-leucocratic (light)
minerals present :augite (#303) = minerals present-not-augite — minerals present :augite (#303) = minerals present-augite
minerals present :quartz (#246) = minerals present-not-quartz — minerals present :quartz (#246) = minerals present-quartz
minerals present :orthopyroxenes (#302) = minerals present-not-orthopyroxenes — minerals present :orthopyroxenes (#302) = minerals present-orthopyroxenes
minerals present :hornblende (#278) = minerals present-not-hornblende — minerals present :hornblende (#278) = minerals present-hornblende
rock colour :mesocratic (medium) = rock colour-not-mesocratic (medium) — rock colour :mesocratic (medium) = rock colour-mesocratic (medium)
type :categories = Rhyolite (#228) — type :categories = Andesite (#215)
accessory minerals :orthopyroxenes (#302) = accessory minerals-not-orthopyroxenes — accessory minerals :orthopyroxenes (#302) = accessory minerals-orthopyroxenes
accessory minerals :augite (#303) = accessory minerals-not-augite — accessory minerals :augite (#303) = accessory minerals-augite
accessory minerals :clinopyroxene (#242) = accessory minerals-not-clinopyroxene — accessory minerals :clinopyroxene (#242) = accessory minerals-clinopyroxene
accessory minerals :nepheline (#301) = accessory minerals-not-nepheline — accessory minerals :nepheline (#301) = accessory minerals-nepheline
accessory minerals :olivine (#243) = accessory minerals-not-olivine — accessory minerals :olivine (#243) = accessory minerals-olivine
accessory minerals :alkali feldspar (#276) = accessory minerals-not-alkali feldspar — accessory minerals :alkali feldspar (#276) = accessory minerals-alkali feldspar
accessory minerals :mica (#300) = accessory minerals-not-mica — accessory minerals :mica (#300) = accessory minerals-mica
accessory minerals :plagioclase feldspar (#241) = accessory minerals-not-plagioclase feldspar — accessory minerals :plagioclase feldspar (#241) = accessory minerals-plagioclase feldspar
accessory minerals :quartz (#246) = accessory minerals-not-quartz — accessory minerals :quartz (#246) = accessory minerals-quartz
accessory minerals :biotite (#277) = accessory minerals-not-biotite — accessory minerals :biotite (#277) = accessory minerals-biotite
accessory minerals :hornblende (#278) = accessory minerals-not-hornblende — accessory minerals :hornblende (#278) = accessory minerals-hornblende
dominant mineral :categories = plagioclase feldspar (#241) — dominant mineral :categories = orthopyroxenes (#302)
dominant feldspar = plagioclase feldspar (#241) — dominant feldspar = alkali feldspar (#276)
100 90 80 70 60 50
Granite (#222)
Granodiorite (#223)
Andesite (#215)
Peridotite (#227)
Diorite (#218)
Rhyolite (#228)
Basalt (#216)
Gabbro (#221)

Figure 15. WebGrid ontology comparison in APECKS

2.10 Instrumentation, Monitoring and Performance

Since WebGrid was developed initially as proof of concept in a research environment, we instrumented it to track all performance parameters under various conditions of loading, and this has been useful in optimizing all later developments. WebGrid I and II ran on a Mac II computer with a 16MHz, 6820 processor, 8 Mbytes main memory and an 80 MByte hard drive, that had been discarded as obsolete. This was replaced in 1999 with WebGrid III running on a 15” flat panel iMac running OS X with the MacHTTP (renamed WebStar) server and KSS0 as a CGI running under the OS 9 emulator. WebGrid

IV is a cross-platform implementation with one version that runs as a universal application under OS X and another than runs under Windows XP and Vista, all compiled from the same source code.

WebGrid I through III had a transaction time of less than one second and, from the server logs, typically supported some six simultaneous users, with peak loads of 15 or so. Most transactions involve the user in substantial local activity taking some 15 to 30 seconds so users saw little impact of server loading with these loads. WebGrid IV running on current machines takes typically less than 100 msec a transaction and can support higher loads, but since the server is now embedded in the stand-alone application we expect much usage will be on local computers and small intranets and a major pattern of use may become the support of collaboration among small groups, either local or distributed, with each project having its own server.

Part of the instrumentation has been to keep a copy of each grid developed under WebGrid in a password-protected cache. We have treated such material as protected by user privacy, and it has been only used to monitor usage in terms of sizes of grid, topics, language used, and so on, and the data has never been made available to third parties or published. However, as WebGrid became used by researchers for distributed data collection we have been asked to provide facilities that go beyond the caching of completed grids as described in Section 2.7, and allow the researcher to track the process of grid development. In WebGrid IV it is possible to collect each page delivered to a user in an SQL database and review them as a series of snapshots of user interaction.

We have also been asked whether it is possible to provide such facilities in real-time. In classical repertory grid elicitation, whether involving computers or not, the facilitator often works with the person from whom the grid is being elicited, prompts them, and discusses the evolving grid. Shaw's (1980) PEGASUS elicitation program emulates some of this facilitation by continually feeding back matches between elements and constructs and prompting further elicitation, and WebGrid implements this, but it would also be useful for a human facilitator to be able to 'look over the user's shoulder' and communicate with the user at a remote location. We have done this with Timbuktu remote screen-sharing software, but this requires specialist software to be installed on the client machine, and it would be useful to be able to do it with standard web clients. We are experimenting with such a capability in WebGrid IV using standard chat and voice over internet protocol (VOIP) facilities for communication and an Ajax polling client for the facilitator that mirrors pages sent to the user on the facilitator's machine. There are obvious privacy issues that need to be addressed, but these are no different from those of any other web usage since such monitoring at the server is always possible.

3 The Future: WebNet, Porting the RepNet Interactive Graphics Component

This paper has focused on what has been involved in porting the repertory grid representation of conceptual models, which was feasible in 1994 with existing technology, and the enhancements of that port to date. However, there are components of the stand-alone application which it has not been feasible to port so far because there has been no support for them within the web standards framework until very recently. These components all involve interactive vector graphics which were not included in the family of Motif user interface widgets that Eric Bina incorporated in Mosaic to support text entry, and became the basis for HTML forms.

3.1 Graphic Representation of Conceptual Models

In the stand-alone application the RepNet tool provides a generic shell for many different knowledge representation networks allowing different types of nodes and links to be defined and styled for presentation, with the logical structure and the appearance being separately managed. We have used the tool across many disciplines to represent bond graphs, PERT charts, concept maps, conceptual graphs, semantic networks and construct nets. Figure 16 shows RepNet representing a concept map from a classic educational study (Novak, and Gowin, 1984; Gaines, and Shaw, 1995b).

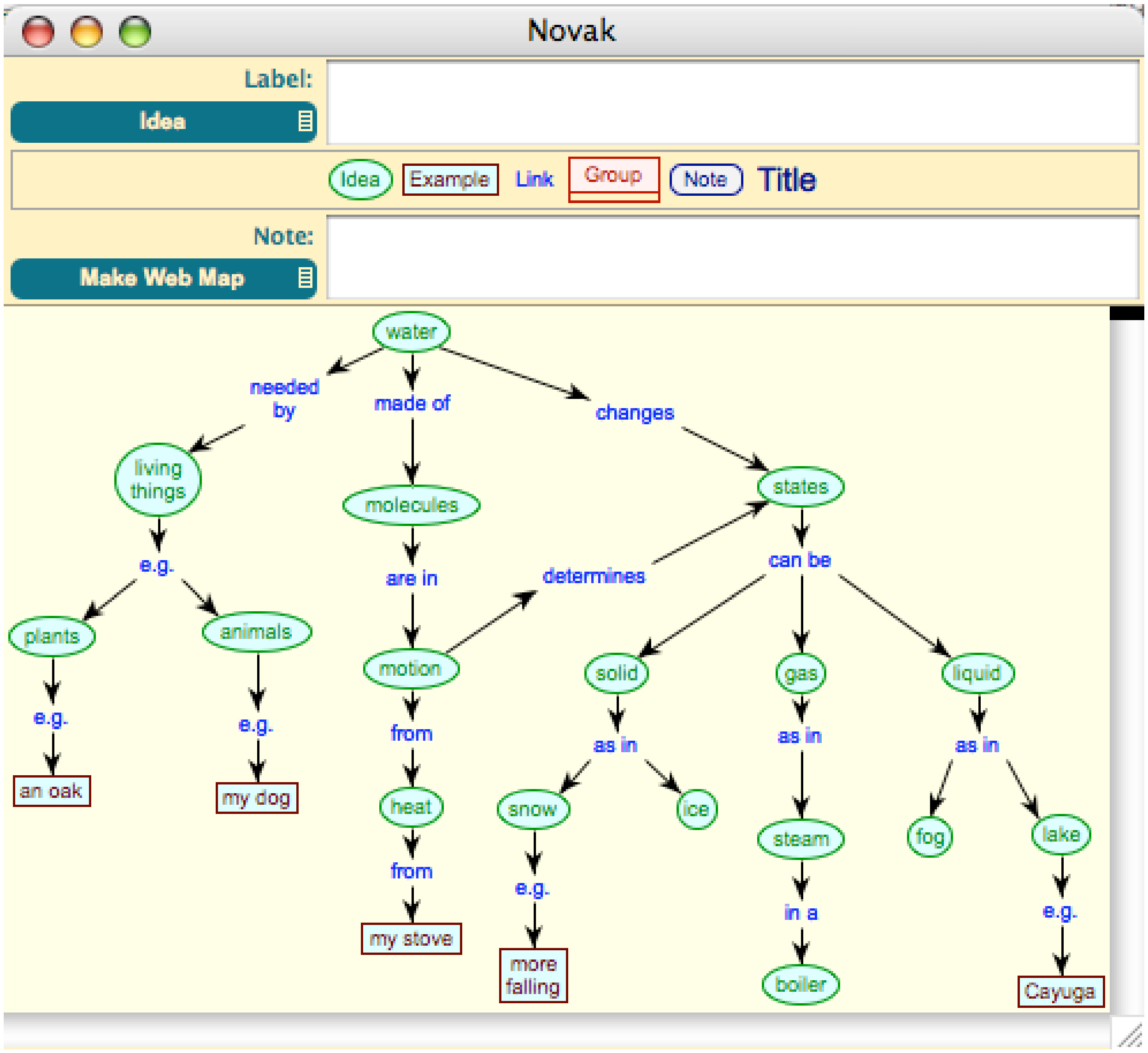


Figure 16. Concept map in RepNet

RepNet graphics are interactive in two distinct ways: first, in their creation as graphical structures; and secondly in their continuing detection of user actions such as clicks, the response to which is fully scriptable based on the same open-architecture scripting system used to program WebGrid. The various grid elicitation and analysis programs in Rep IV all use RepNet for their graphic output, and, as noted in Section 2.4, ongoing interaction with this output is important to understanding, exploring and using it.

For example, SocioGrids, the multiple grid analysis components of Rep IV, which is important to supporting collaborative usage, make major use of RepNet interactivity. We have been able to port the comparison of pairs of grids as illustrated in Figure 11, but the full SocioGrids analysis relies on user interaction with the graphic output in order to explore and understand it fully. Figure 17 shows a socionet in RepNet generated by SocioGrids, where the links between grids represent the modeled capability of the conceptual structure represented in one grid to be used to construe the world in the same way as that represented in another (Shaw, 1980).

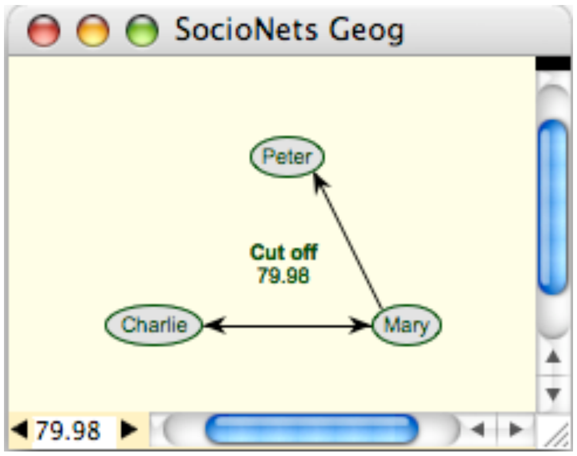


Figure 17. Socionet in RepNet generated by a Rep IV SocioGrids analysis

All links are possible at some level of comprehension, and those actually displayed are determined by the cut off applied to the strength of the links. This is adjustable through control at the bottom left, letting the user see the links come and go as the cut off is adjusted, and hence explore the structural relationships in the community represented, perhaps producing a composite graphic showing this as illustrated in Figure 18. We would like to port these capabilities to the web.

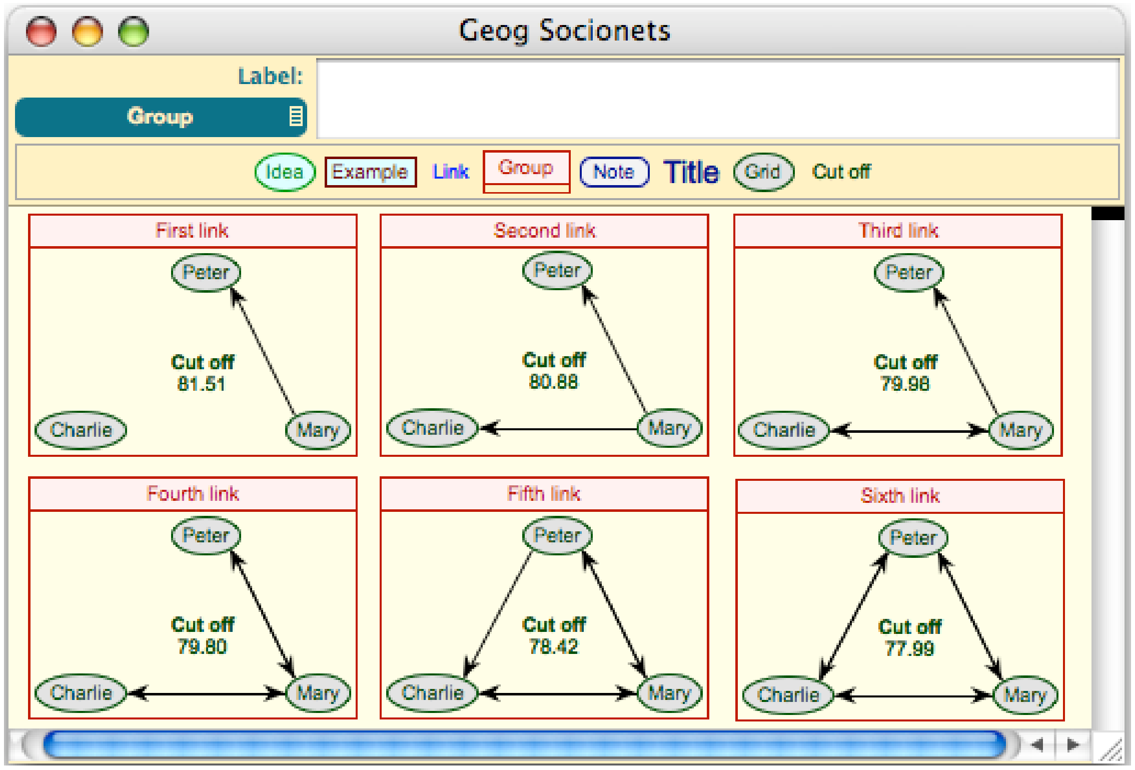


Figure 18. Socionets composite

3.2 Construct Nets

There are also important graphic representations of conceptual structures that Rep IV supports, and that we wish to port to the web. For example, a complementary representation to that of the grid is that through *construct nets* in which the conceptual model comprises labeled nodes connected by directed links (Shaw, and Gaines, 1992b; Gaines, and Shaw, 1993b). Such nets are used informally as *concept maps* in education (Novak, and Gowin, 1984) and with well-defined semantics as *semantic networks* in formal knowledge representation (Gaines, 1991). We have demonstrated the wide range of uses of such graphic network representations in many stand-alone and web-based applications (Gaines, and Shaw, 1995a), and the stand-alone Rep IV application provides the capability to represent construct systems in both grids (RepGrid) and networks (RepNet).

Figure 19 shows a construct net derived from a repertory grid representing a set of sample cases of contact lens prescriptions based on the well-known study of rule induction by Cendrowska and Bramer (1987). The popup menu is a scripted response to the event generated by clicking in the net outside a node. Clicking within a node generates a different scripted menu that enables the node to be assigned a truth value of true or false. Selected "Infer" in the popup menu shown triggers a scripted description logic inference engine to infer from the truth values of some nodes the truth values of other, thereby operating as an 'expert system.' The inference network shown may also be tested on the repertory grid holding the cases to make inferences about each case. The details are described in the description logic literature (Gaines, 2004).

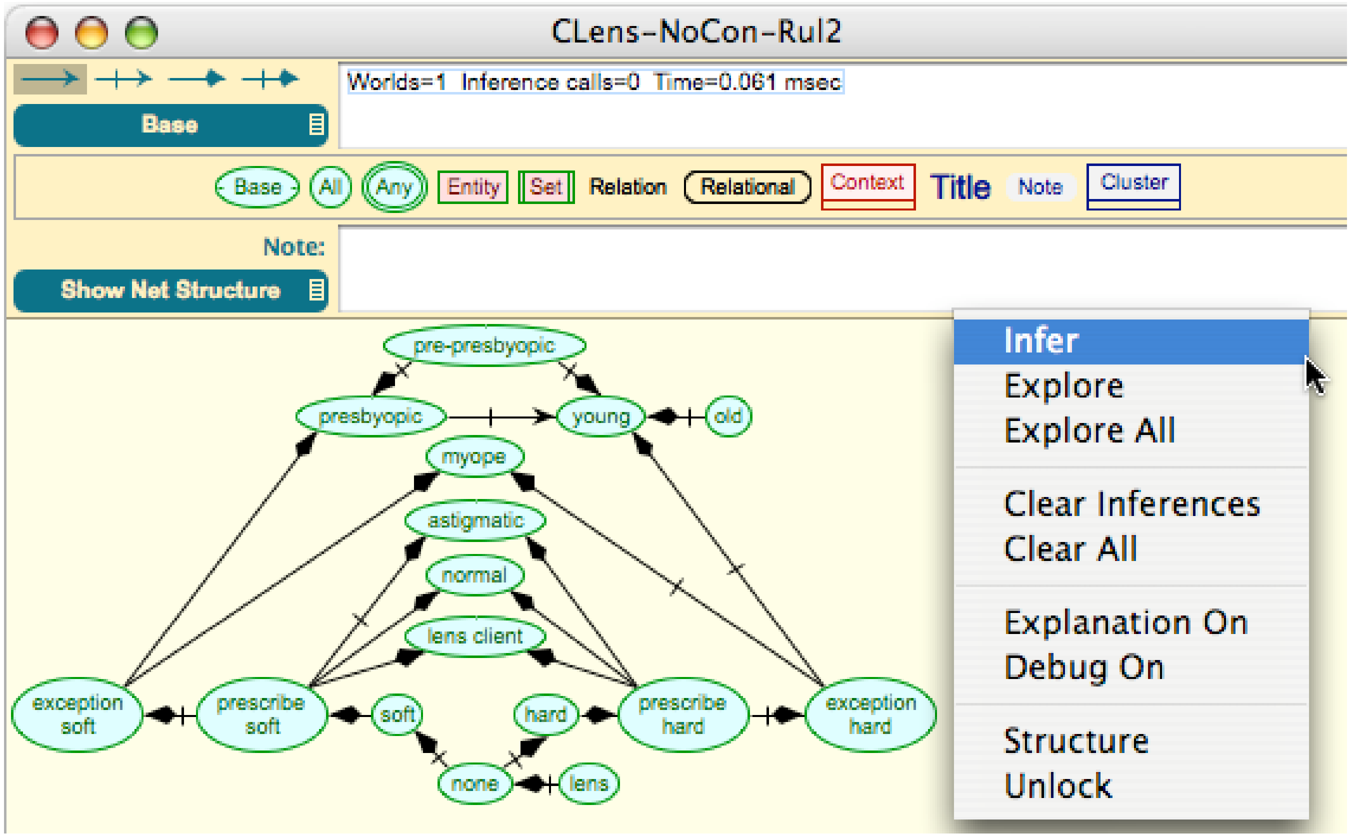


Figure 19. Construct net solving the contact lens prescription problem

3.3 Porting the Interactive Graphic Component

Although the current WebGrid port has been very heavily used and has supported a wide range of research activities, we shall not regard it as complete until we have also ported the graphic functionality shown in Figures 16 through 19. This requires some form of interactive scalable vector graphics on the web that has not been so readily available as the form widgets and static bitmap graphics used to implement the repertory grid component. RepNet can output its networks as a PNG file and this is how the images shown in Figures 5 and 11 were generated for WebGrid. It can also output a client-side clickable map data structure allowing any net to be used on the web and a URL to be accessed when a node is clicked. The WebGrid server does all this dynamically when it services a request to fetch a RepNet file. However, while this is useful, it only makes the static end-product of the process of developing a graphic representation available, not the interactive development or use.

In the early days of Java applets, which allowed one to run Java applications within a web document, we experimented with the implementation of our graphic network tools as interactive applications on the web, but found the applet support in the major browsers was unreliable. Now with improved applet support, or alternative techniques such as W3C's (2007) *scalable vector graphics* standard that is beginning to be supported in some browsers, Adobe's Flash technology, and Zorn's (2005) Javascript vector graphics library that uses standard HTML elements to provide a graphic drawing environment in web pages, it is possible to support interactive graphic applications on the web as demonstrated by the Open-jACOB workflow editor (Open-jACOB, 2007). We plan to use these technologies to make the RepNet tool in Rep IV available interactively on the web integrated with the WebGrid environment. We expect this implementation to evolve as vector graphics become an increasingly standardized feature in web browsers, with the underlying graphic data structures used in RepNet being presented in improved ways as the web's graphic technology evolves.

When this port is completed it will provide the capability on the web to induce inferential knowledge structures from grids and use them to emulate human anticipation, inference and problem solving (Shaw, and Gaines, 2005). It will be possible to take an informal concept map such as that of Figure 17 and evolve it to be a formal knowledge structure from which inferences may be drawn, modeling the processes whereby informal human idea become increasingly formally defined in the process of scientific development (Nersessian, 1989). We have used a combination of grids and nets in many courses to facilitate students clarifying their ideas, comparing them others, and studying their consequences (Shaw, and Gaines, 1992a; Shaw, and Gaines, 1998). Our overall objective is to make all these capabilities available on the web to distributed communities, anytime, anywhere.

4 Conclusions

When we decided to experiment with the porting of our repertory grid and other knowledge acquisition tools to operate through the World Wide Web in 1994 it was primarily a curiosity-driven venture with little expectation of achievement. Web browsers were still fairly primitive and unreliable, and the potential to support highly interactive applications with the basic widgets available seemed very limited. We proceeded with the venture in major part because others in our field, such as the Ontolingua research team at Stanford (Rice, Farquhar, Piernot, and Gruber, 1996), were pursuing the development of web-based interaction with their tools with great enthusiasm.

However, WebGrid when we released it as a proof of feasibility demonstration early in 1995 came into widespread use very rapidly, and that usage has continued and grown during the past decade. We had to accept very early on that we were serving a large and diverse, world-wide community of users who were largely anonymous to us, and that we needed to focus on supporting them with a reliable, self-explanatory service that necessitated little direct interaction with our users. This article has described some of the major lessons learned in designing a service with these objectives in mind.

The success of the approach adopted is probably most evinced in the periods when the workload of other projects has led to us neglecting to monitor WebGrid usage for periods of a year or more. What generally reminds us to take note of the WebGrid service after such a period is email from a user concerned about disruption of their research or teaching through failure to access the server, usually during the Calgary winter when power outages may disrupt the network through which the server is connected to the Internet.

4.1 WebGrid IV

WebGrid has gone through three major upgrades since its release in 1994. WebGrid II was released in January 1998 and offered multimedia annotation, user-defined customization, caching grids for use by collaborators, and rule induction and inference facilities. WebGrid III was released in February 2001 and offered advanced analysis features, improved expert system capabilities, and the capability to register password-protected caches for remote data collection and analysis. Each new version has been completely backwards compatible with earlier versions so that data collected by them can be aggregated with later data and analyzed using later tools.

WebGrid IV, released for testing in June 2006, was completely re-written using current web technologies within a Web 2.0 conceptual framework with aspirations to play a significant role in the evolution of Web 3.0. It is completely script-driven and this, together with its support of Unicode, makes it easy to re-program it to operate in any of the languages of the world without modification to the application source code. Its documents have a modular structure supporting CSS styling so that customization of its appearance is simple for end users. Interactivity has been improved through the use of Javascript and dynamic HTML.

For example, in the original stand-alone application the relative positions of elements along a rating scale was immediately visible as shown in Figure 1. The use of HTML menus as shown in Figure 2 was effective for individual ratings but lost this relative positioning information, and a “Show Sorted” button was added that reordered the elements to regain this feedback to the users. In WebGrid IV, Javascript is used to restructure the DOM so that the ratings are always sorted immediately after a new rating is entered. This and similar enhancements, have made the web application more supportive of its users.

WebGrid IV is one module in Rep IV (http://repgrid.com/), a new cross-platform implementation of our repertory grid, concept mapping, and other conceptual modeling tools. All the applications were refactored yet again and built on an object-oriented library of classes that simplify the development, maintenance and enhancement of these types of tools (Gaines, 1994). A web server having the architecture described in Section 3 and a web browser based on the HTML widget of the underlying platform are part of Rep IV.

The wheel has now come a complete circle in that the standalone application is cross-platform to Macintosh OS X and Microsoft Windows, so that a web service is no longer necessary to achieve cross-platform access. However, the need to support distributed communities of users across the Internet, and the advantages of HTML as a graphic user interface language remain and are highly significant. The standalone application can act as its own client/server combination and also as a server to standalone web browsers on the same machine or across the Internet. The application also provides modes of knowledge elicitation and analysis on the local machine that are not yet readily supported through existing HTML widgets, but which we plan to port to the web as its underlying technology continues to evolve.

4.2 Lessons Learned and Future Directions

Looking back over the decade of the web application and the three decades of porting the repertory grid tools, the most obvious lesson learned are the need to cope with the limitations of the technology of each era, and to evolve the applications to take advantage of the rapidly evolving features of information technology. As the Red Queen informed Alice, one has to run very fast to keep still. In the early years of the standalone application we had to make major efforts to fit the programs into the small memories of the computer and to optimize them to be reasonably interactive given low processor speed. In the early years of the web port we had to keep image data small so as not to overload limited communication facilities, and to limit the use of HTML features to the relatively small subset supported by virtually all browsers. Until recent years the technologies for cross-platform implementation were complex, expensive and unreliable, and became rapidly obsolete. The ease of implementation nowadays is astonishing and the only limits seem to be human imagination.

For the future, the port of interactive graphics as discussed in Section 4 is our first priority, and thereafter we envision the next major refactoring and implementation as being one where the components of WebGrid IV are separated as individual services communicating through standardized protocols and able to operate in a distributed fashion on multiple platforms across a network. This would address one of our long-term objectives of supporting heterogeneous integration of a wide range of knowledge acquisition, modeling and management tools.

Research progress is generally much slower than it could be because individual projects across the world result in a plethora of systems whose operation generally cannot be sustained in the long-term. Research could be systematically accelerated if the significant advances in such systems could be factored into highly specific modules designed to interoperate with other modules across a network through well-defined protocols. This would facilitate the rapid prototyping of new, experimental systems, and would help to make the outcomes of research much more cumulative.

The underlying technologies of the web and the Internet are structured in this way, and we need to accelerate the movement towards structuring application technologies in the same fashion. This applies both to tools such as WebGrid and to content, for example, educational materials when it is used in constructivist learning environments. We envision WebGrid in its increasingly modular and customizable implementation as becoming an integrated component of many online learning environments where learners can, at any time, elicit their conceptual models of a domain, use them for problem-solving and compare them with those of their peers and subject matter experts.

We welcome collaboration with colleagues having related objectives.

Acknowledgements

Over the past thirty years the development of our conceptual modeling programs has benefited from interactions with colleagues who have collaborated with us in various ways, used them in their own studies, and provided insights that have contributed to their evolution. It is impossible to mention them all but, in our retirement, we treasure the memories and the contributions they have made. Thank you.

5 References

5.1 Web Access

Papers by Gaines or Shaw may be accessed at http://www.cpsc.ucalgary.ca/~gaines/reports/

Rep IV may be accessed at http://repgrid.com/RepIV/

WebGrid III may be accessed at http://tiger.cpsc.ucalgary.ca/

The current beta of WebGrid IV may be accessed at http://gigi.cpsc.ucalgary.ca :1500/WebGridIV.html

5.2 Bibliography

Berners-Lee, T. (1989). **Information Management: A Proposal** (Report Number http://www.w3.org/History/1989/proposal.html). CERN, Geneva.

Boose, J. H. (1984). Personal construct theory and the transfer of human expertise. In **Proceedings AAAI-84**, pp. 27-33. American Association for Artificial Intelligence, California.

Boose, J. H. (1986). **Expertise Transfer for Expert Systems**. Elsevier, Amsterdam.

Cendrowska, J. (1987). An algorithm for inducing modular rules. **International Journal of Man-Machine Studies** 27, 349-370.

Gaines, B. R., and Shaw, M. L. G. (1980). New directions in the analysis and interactive elicitation of personal construct systems. **International Journal Man-Machine Studies** 13, 81-116.

Gaines, B. R., and Linster, M. (1990). Integrating a knowledge acquisition tool, an expert system shell and a hypermedia system. **International Journal of Expert Systems Research and Applications** 3, 105-129.

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

Gaines, B. R., Rappaport, A., and Shaw, M. L. G. (1992). Combining paradigms in knowledge engineering. **Data and Knowledge Engineering** 9, 1-18.

Gaines, B. R., and Shaw, M. L. G. (1993a). Eliciting knowledge and transferring it effectively to a knowledge-based systems. **IEEE Transactions on Knowledge and Data Engineering** 5, 4-14.

Gaines, B. R., and Shaw, M. L. G. (1993b). Basing knowledge acquisition tools in personal construct psychology. **Knowledge Engineering Review** 8, 49-85.

Gaines, B. R. (1994). Class library implementation of an open architecture knowledge support system. **International Journal Human-Computer Studies** 41, 59-107.

Gaines, B. R., and Shaw, M. L. G. (1994). Using knowledge acquisition and representation tools to support scientific communities. In **AAAI'94: Proceedings of the Twelfth National Conference on Artificial Intelligence**, pp. 707-714. AAAI Press/MIT Press, Menlo Park, California.

Gaines, B. R. (1995). Porting interactive applications to the web. In **4th International World Wide Web Conference Tutorial Notes**, pp. 199-217.

Gaines, B. R., and Shaw, M. L. G. (1995a). Concept maps as hypermedia components. **International Journal Human-Computer Studies** 43, 323-361.

Gaines, B. R., and Shaw, M. L. G. (1995b). Collaboration through concept maps. In **Proceedings of CSCL95: Computer Support for Collaborative Learning** (Schnase, J. L., and Cunnius, E. L., eds.), pp. 135-138. Lawrence Erlbaum, Mahwah, New Jersey.

Gaines, B. R. (1996). Transforming rules and trees into comprehensible knowledge structures. In **Knowledge Discovery in Databases II** (Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., eds.), pp. 205-226. AAAI/MIT Press, Cambridge, Massachusetts.

Gaines, B. R., and Shaw, M. L. G. (1997). Knowledge acquisition, modeling and inference through the World Wide Web. **International Journal of Human-Computer Studies** 46, 729-759.

Gaines, B. R., and Shaw, M. L. G. (1999). Embedding formal knowledge models in active documents. **Communications of the ACM** 42, 57-63.

Gaines, B. R. (2004). Understanding ontologies in scholarly disciplines. In **Proceedings 2004 International Workshop on Description Logics, DL2004** (Haarslev, V., and Möller, R., eds.). CEUR-Workshop Proceedings, http://CEUR-WS.org/, Whistler, B.C.

Kelly, G. A. (1955). **The Psychology of Personal Constructs**. Norton, New York.

Nersessian, N. J. (1989). Conceptual change in science and in science education. **Synthese** 80, 163-184.

Novak, J. D., and Gowin, D. B. (1984). **Learning How To Learn**. Cambridge University Press, New York.

Open-jACOB. (2007). **Open-jACOB web based workflow editor**. http://www.openjacob.org/draw2d.html.

Prototype. (2006). **Prototype JavaScript Framework**. http://www.prototypejs.org/.

Rice, J., Farquhar, A., Piernot, P., and Gruber, T. (1996). Using the web instead of a window system. In **Proceedings of CHI'96**, pp. 103-117. ACM, New York.

Shaw, M. L. G. (1978). Interactive computer programs for eliciting personal models of the world. In **Personal Construct Psychology 1977** (Fransella, F., ed, pp. 59-67. Academic Press, London.

Shaw, M. L. G. (1980). **On Becoming A Personal Scientist: Interactive Computer Elicitation of Personal Models Of The World**. Academic Press (now only available from http://repgrid.com or http://www.gallowglassbooks.com), London.

Shaw, M. L. G., and Gaines, B. R. (1992a). Mapping creativity with knowledge support tools. In **AAAI-91 Workshop on Creativity: Models, Methods and Tools**, pp. 32-45. AAAI, Menlo Park, California.

Shaw, M. L. G., and Gaines, B. R. (1992b). Kelly's 'Geometry of psychological space' and its significance for psychological modeling. **New Psychologist** 23-31.

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

Shaw, M. L. G., and Gaines, B. R. (1998). A research-based masters program in the workplace. **Proceedings of WCCCE'98: Western Canadian Conference on Computing Education**

Shaw, M. L. G., and Gaines, B. R. (1999). Modeling the social practices of users in Internet communities. In **UM99: User Modeling: Proceedings of the Seventh International Conference** (Kay, J., ed, pp. 77-86. Springer, New York.

Shaw, M. L. G., and Gaines, B. R. (2005). Expertise and expert systems: emulating psychological processes. In **The Essential Practitioner's Handbook of Personal Construct Psychology** (Fransella, F., ed, pp. 87-94. Wiley, Chichester, UK.

Tennison, J., O'Hara, K., and Shadbolt, N. R. (2002). APECKS: using and evaluating a tool for ontology construction with internal and external KA support. **International Journal Human-Computer Studies** 56, 375-422.

W3C. (2007). **Scalable Vector Graphics (SVG) 1.1 Specification**. http://www.w3.org/TR/SVG11/.

Zorn, W. (2005). **High Performance JavaScript Vector Graphics Library**. http://www.walterzorn.com/jsgraphics/jsgraphics_e.htm.