TADZEBAO AND WEBONTO: DISCUSSING, BROWSING, AND EDITING ONTOLOGIES ON THE WEB

John Domingue
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA, UK
http://kmi.open.ac.uk/people/domingue/
J.B.Domingue@open.ac.uk

ABSTRACT

In this paper I describe two systems which, in different ways, address the shortcomings of current approaches to enabling ontology construction and use via the World-Wide Web. The first system Tadzebao, enables knowledge engineers to hold synchronous and asynchronous discussions about ontologies. Tadzebao addresses the fact that an integral part of communal design, dialogue, has largely been ignored by the community. The second system WebOnto uses a Java based client to alleviate the acknowledged problems of creating interfaces in 'vanilla HTML'.

1.   INTRODUCTION

Within the knowledge acquisition community the use of ontologies in the construction of knowledge intensive systems is now widespread. Ontologies are used to facilitate knowledge sharing and reuse (Borst, Akkermans, and Top, 1996), agent interoperability (Falasconi, Lanzola, and Stefanelli, 1996), and knowledge acquisition (van Heijst, 1995).

Ontology construction is often a collaborative effort, because the knowledge contained within an ontology represents a common view shared within a community. The collaborative construction and maintenance of any software by a community is a non-trivial task, especially if the community is geographically spread. There are problems of distribution, version management and hardware and software platform inconsistencies. Recently the World-Wide Web (WWW) has been seen to offer a way of easing these problems. The Ontolingua group at Stanford (Farquhar, Fikes, Pratt, and Rice, 1995) under the Knowledge Sharing Effort (Neches, Fikes, Finin, Gruber, Patil, Senator, and Swartout, 1991) and more recently the work on the Ontosaurus project (Swartout Patil, Knight, and Russ, 1996) have exploited WWW technology for delivering a service for collaboratively browsing and editing ontologies.

Although the approach used by the two groups was a forward step, the use of WWW technology for communal knowledge construction is still in its infancy, and as such there remain issues to be addressed and problems to be solved. In this paper I describe two systems which, in different ways, begin to fill in the gaps.

Dialogue, for example to reach consensus or to disseminate information, is an important activity within communal design (Fischer, Lemke, McCall, and Morch, 1991) that has not been addressed by current approaches. In the next section I give an overview of Tadzebao a web based tool which supports synchronous and asynchronous discussions on ontologies.

Both the Ontolingua and Ontosaurus systems use HTML for the client interface, and both groups acknowledge that the use of HTML generates interface design problems. In section 3 of this paper I review these problems. In section 4 I describe WebOnto a tool for collaboratively browsing and editing ontologies, which uses a Java (Sun, 1996) client to alleviate the interface problems generated by HTML. The architecture of Tadzebao and WebOnto is described in section 6. In section 7 I discuss the principles underlying the design of the tools and how the approach supports

ontology building communities in particular those concerned with software reuse. A short section on related work is followed by a summary.

2. TADZEBAO: SUPPORT FOR DISCUSSIONS ON ONTOLOGIES
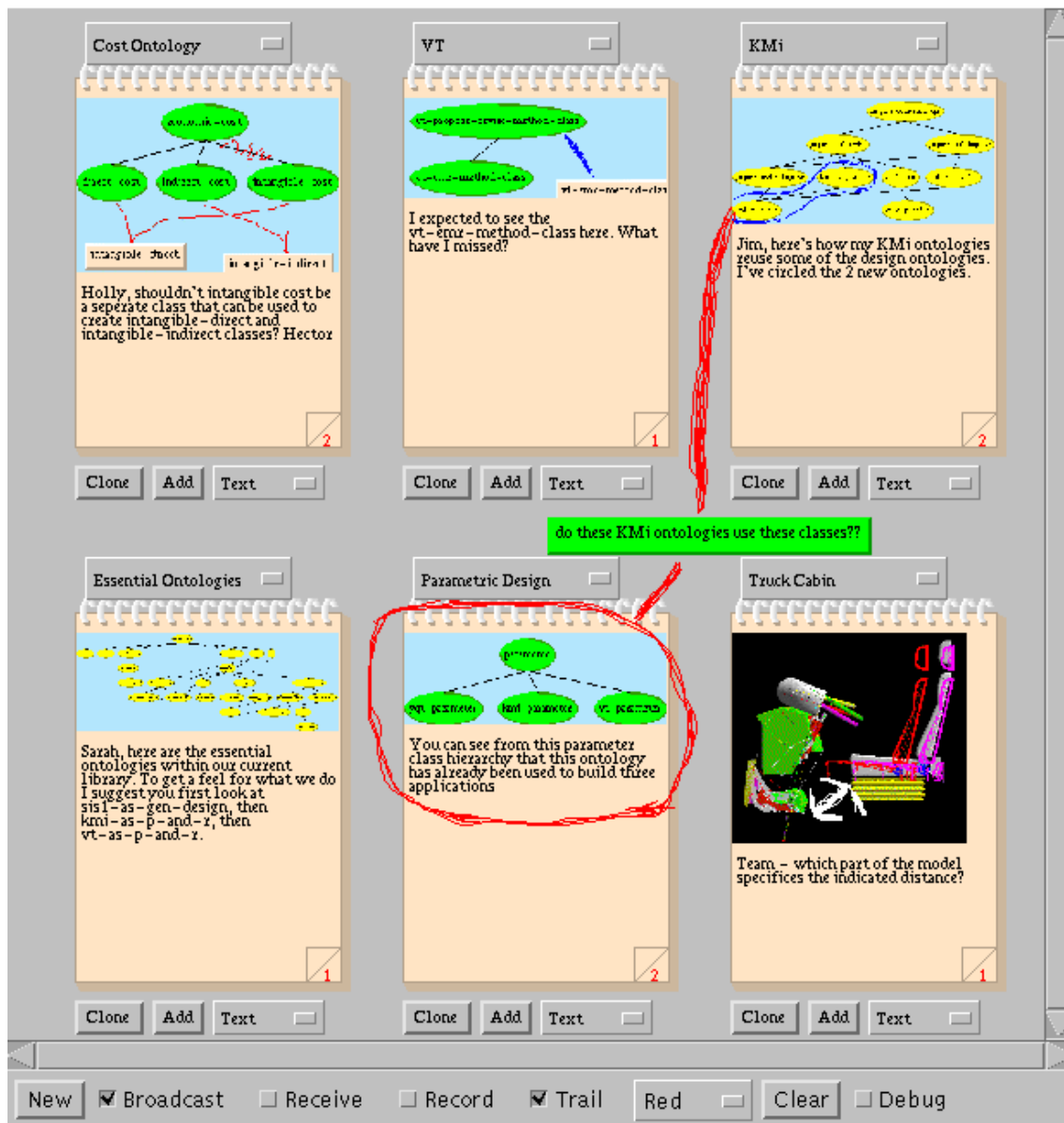
Figure 1. A screen snapshot of Tadzebao in use.

The overall design goal for Tadzebao was to support both asynchronous and synchronous discussions on ontologies. Asynchronous discussions are necessary because an ontology design team or community may be spread over large time zones. Synchronous discussions are useful a) when a dialogue is composed of many small utterances, b) when there is time pressure, or c) to give users a sense of belonging to a team or community.

Because design dialogues are mainly about the artefact under construction it was important to allow knowledge engineers to refer directly to ontologies in their messages. This meant that Tadzebao had to integrate tools, such as text editors and drawing tools, for expressing general ideas with tools for manipulating ontologies.

Before describing the system itself a brief explanation of the name is necessary. Tadzebao, which literally means "Big Character Poster", is the Chinese word for the type of poster used to support political debate during the Cultural Revolution. During the Cultural Revolution a political argument or ideology would be expressed through the placement of a poster (a tadzebao). Rebuttals to or comments on the initial argument would be expressed by additional posters (tadzebaos) on top of or around the original poster.

Tadzebao uses an extension of this metaphor for guiding knowledge engineers around ongoing dialogues on ontologies. Within Tadzebao, dialogues are centred around a *notepad* which contains a series of pages. Each notepad page can contain a mixture of text, GIF images, hand drawn sketches and ontology components represented in OCML (Motta, 1998). Dialogues can be browsed by clicking on the bottom right corner of a notepad.

Figure 1 shows six notepads each containing a design dialogue. I shall now briefly describe the top page of each of these notepads in turn.

Cost Ontology - Hector and Holly are collaboratively constructing a medical cost ontology. Holly had left a part of the class hierarchy representing the concept of economic payoff on the first page (page 1) shown in figure 2a below. She created the page shown in figure 2a by selecting "Ontology" from the selector on the lower right of the figure, clicking on the "Add" button, creating the hierarchy in WebOnto, quitting WebOnto, and finally clicking below the figure and typing her text into the editor. Hector created the page shown in figure 1 (page 2) by cloning page 1 (by clicking on the "Clone" button), clicking on the image, adding the sketched red lines and the labels "intangible-direct" and "intangible-indirect" using WebOnto's sketching tools, then quitting WebOnto and adding the text. Figure 2b shows Holly's reply to Hector which was created in a similar fashion. Figure 2c shows an unscaled version of the image captured in figure 2a, and figure 2d shows an unscaled version of the image captured in figure 2b. This ontology (Falasconi, Lanzola, and Stefanelli, 1996) is currently under construction as part of the ongoing EU funded Telematics project HCReMa (Health Care Resource Management) (Project HC 3103) (HCReMa, 1997).
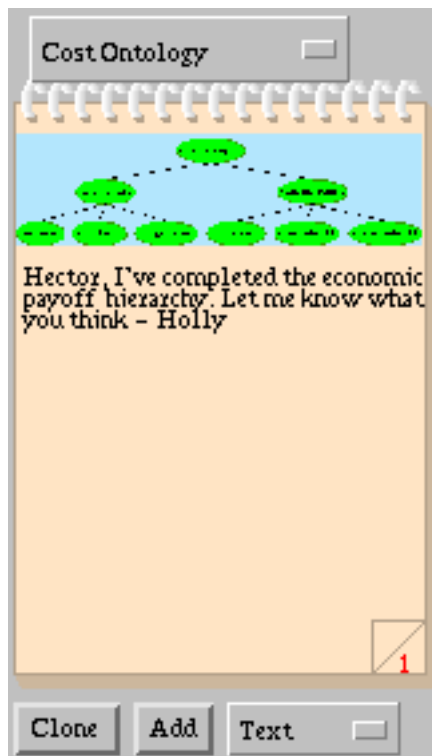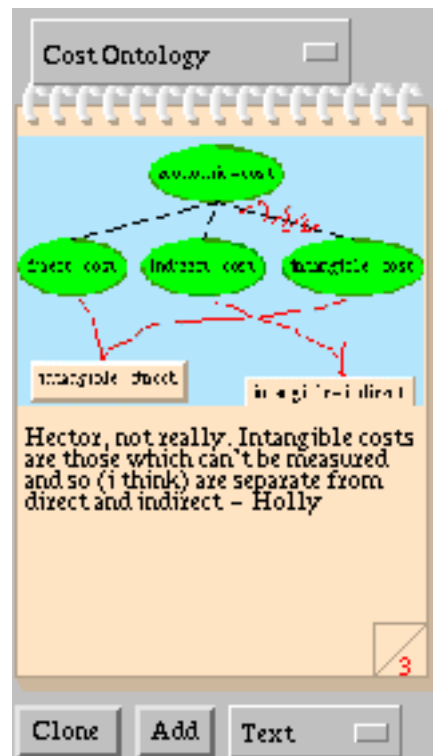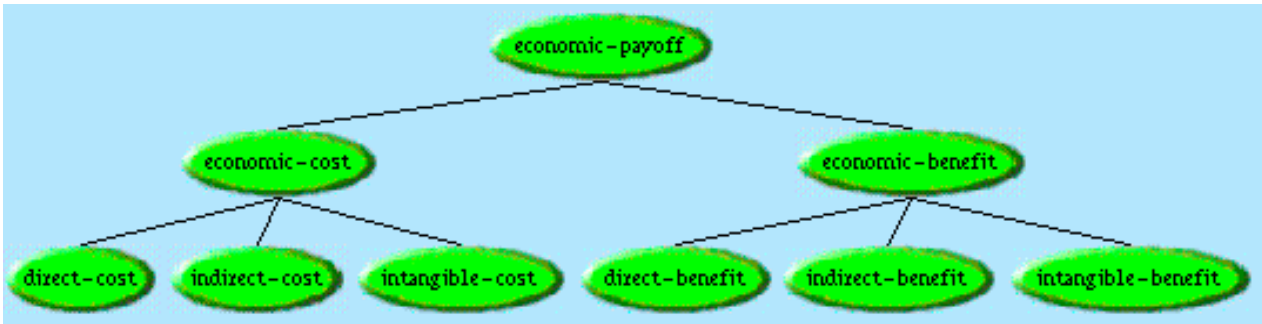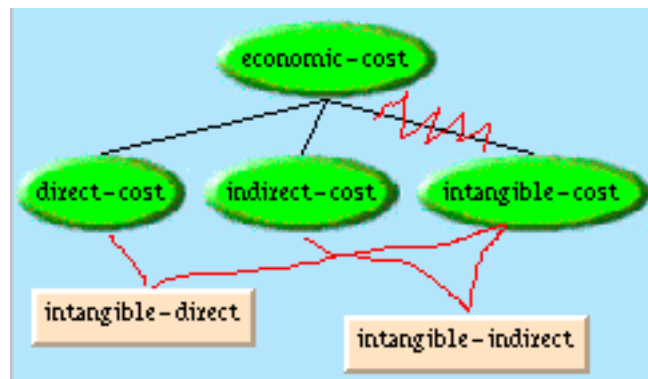

Figure 2a

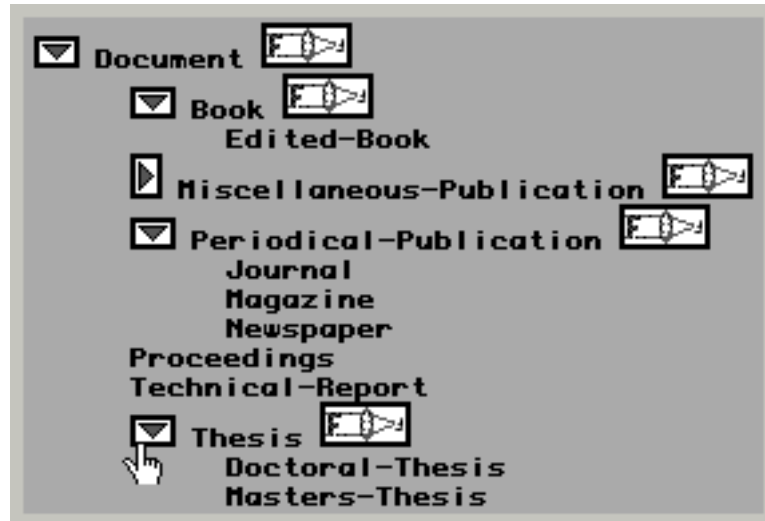Figure 2b

Figure 2c

Figure 2d

Figure 2a) shows page 1 of the first notepad in figure 1 and figure 2b) shows page 3. Figure 2c) shows the image captured in 2a) without scaling. Figure 2d shows the image captured in 2b) without scaling.

VT - in this page a user is looking for help from the designers of the VT ontology, which was constructed as a solution to Sisyphus II (Yost, 1994), a lift configuration design problem. The user posed their query by annotating part of the existing class hierarchy with a class which she expected to see. Responses to the query would be articulated as new (possibly cloned) pages.

KMi - the top part of this page shows a hierarchy of ontologies. Jim has been left with a message indicating, via annotations, which ontologies were reused to construct an application which this summer was used to allocate the members of the Knowledge Media Institute to offices within a new building.

Essential Ontologies - Sarah is a new knowledge engineer within a department. A colleague has left a message for her showing the hierarchy of ontologies currently within the department's library. In the text below the image the colleague describes which ontologies give the best flavour of the department's work.

Parametric Design - A user has left a message containing part of a class hierarchy showing how a class from a parametric design ontology has been inherited in classes within three separate applications. The red sketched circle around this page, the red lines to the ***KMi*** page, and the label "do these KMi ontologies use these classes?" are part of a live broadcast. Selecting the "Broadcast"

Figure 3. A class hierarchy displayed in Ontolingua's class browser. The triangles open ▷ and close ▽ subgraphs, and the 🔲 widget changes the root node. Taken from (Rice, Farquhar, Piernot, and Gruber, 1996) (c) 1996 ACM by permission.

button in the bottom control panel broadcasts all interface actions (including all sketching, page turning and the adding of text, GIF images or ontologies) to *receivers,* Tadzebao's with the "Receive" button selected, in real time. The user is asking if the KMi ontologies use any of the classes shown in the parametric design notepad.

Truck Cabin - The page contains a GIF image showing the output from a CAD package. The image shows the positioning of components within a truck cabin, which was partially designed using the parametric design ontology (Banecek, Drvota, and Valasek, 1996). A user has used the drawing tools to draw a white arrowed line to indicate the distance mentioned in the text. Clicking on an image creates a new window showing a full size version.

The Tadzebao in figure 1 is displaying 6 notepads. Users can opt to browse 4 notepads or a single notepad if they wish. Dialogues are chosen using the selector above each notepad (currently showing "Cost Ontology", "VT", "KMi", etc.). In section 4 I describe a scenario showing Sarah using WebOnto to browse one of the ontologies contained in the image on the top page of the "Essential Ontologies" notepad. I shall now review the problems for interface designers when using HTML to create their interfaces.

3. ACKNOWLEDGED PROBLEMS WITH HTML INTERFACES

As I mentioned earlier both the Ontolingua and Ontosaurus groups accept that using 'vanilla HTML' imposes numerous problems for interface design. In this section I review the problems described in (Rice, Farquhar, Piernot, and Gruber, 1996) and (Swartout, Patil, Knight, and Russ, 1996) using three categories.

3.1. Centralisation of All Data

Because all data is stored at a central server and the link between the server and the client is of limited bandwidth, small system state changes are not feasible. This restriction means that intermediate user activities can not be stored and tightly couple interfaces which provide immediate feedback or use direct manipulation techniques are ruled out. A consequence of not having direct manipulation is that interface designers have to use repeated control widgets. An example is shown in figure 3 - a snapshot of the Ontolingua browser in use. Ontolingua uses a metaphor similar to the Macintosh Finder for displaying class hierarchies. The triangular widgets open and close subhierarchies. The 🔲 widget makes the current node the root of the tree. In a direct

manipulation interface the root of the tree would be altered by selecting the desired node and an appropriate menu item. Since this is not possible the icon for the command has to be repeated cluttering the display.
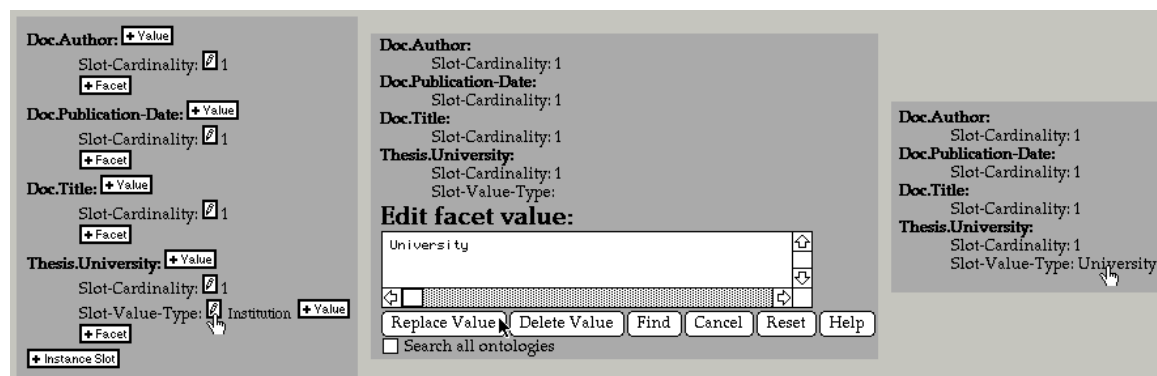
Figure 4. Three screen snapshots showing (1) a user clicking on the ▧ icon to edit the type of the Thesis.University slot, (2) changing the value from "Institution" to "University", and (3) removing the screen clutter by inhibiting all edit widgets. Taken from (Rice, Farquhar, Piernot, and Gruber, 1996) (c) 1996 ACM by permission.

3.2. One-shot Connections

Servers can only respond to web page requests. Once a server has sent the data for a web page to a browser the connection is closed. This means that the interface can only respond to user actions. It is not possible to incorporate asynchronous communication, for example, to periodically prompt to save data. Additionally, all interface updates have to be at the page level. This means that interface designers are limited to mapping inner structure views of ontology objects, such as concepts and instances, to web pages.

Figures 4 and 5 show the Ontolingua and Ontosaurus interfaces for editing classes and concepts.

Figure 4 shows three screen snapshots showing (1) a user clicking on the ▧ icon (note that this icon has to be repeated as for the ▭ widget) to edit the type of the Thesis.University slot, (2) changing the value from "Institution" to "University", and (3) removing the screen clutter by inhibiting all edit widgets. Figure 5 shows the form used in Ontosaurus to edit concepts and instances. Fields are provided to edit role restrictions and superconcepts. More complex representations are edited in the text windows.

Notice how in both systems the interface designer has chosen to map a web page to a single object. If the display contained more than a one object then the redisplay time, the time taken to transfer and render a whole page, would become unacceptable. In addition to waiting for a page update users are cognitively burdened by having to remember previous pages.

3.3. The Tyranny of the Browser

To facilitate the display of multimedia objects across a wide range of platforms the amount of control over the browser available from HTML is limited. This impacts on interface designers in several ways. The exact layout and appearance of components can not be prescribed. The widget set available within HTML is also limited. Both the Ontolingua and the Ontosaurus clients display data using a limited number of purely typographical features. Graphics (icons) are used purely for control widgets.
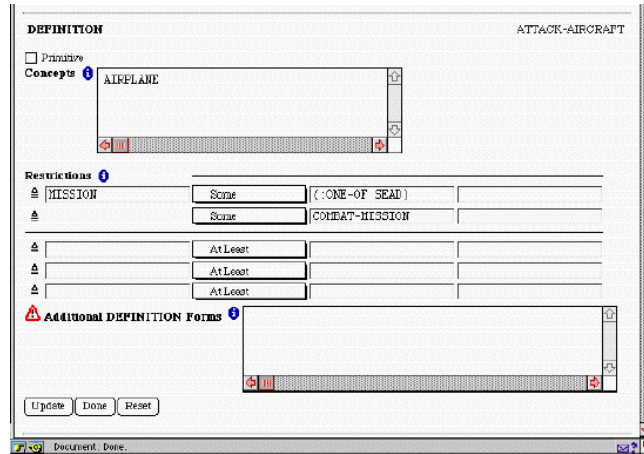
Figure 5. A screen snapshot showing the Ontosaurus editor for editing concepts and instances. Fields are provided to edit role restrictions and superconcepts. More complex Loom representations are edited the text windows. Taken from (Swartout, Patil, Knight, and Russ, 1996) by permission.
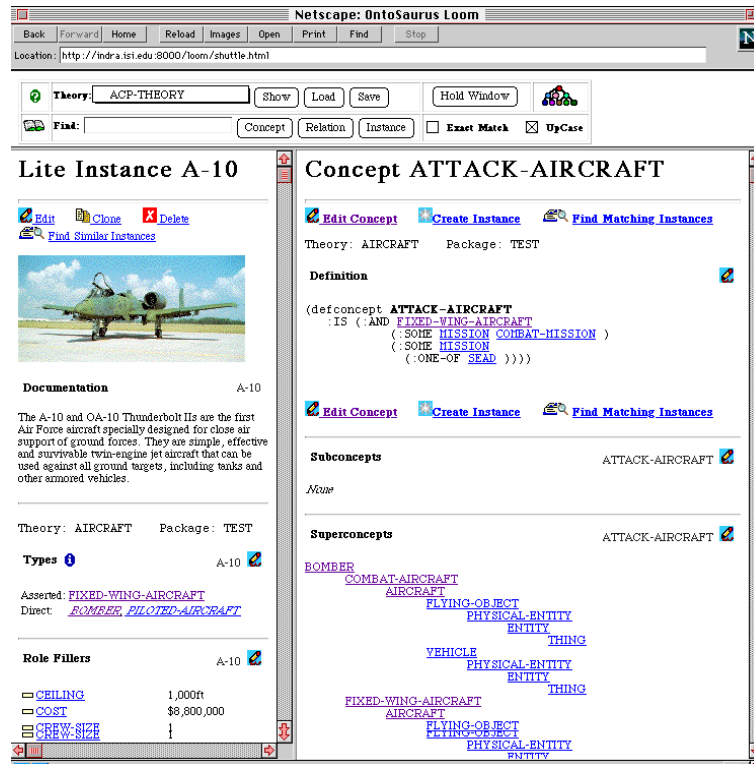
Figure 6. A screen snapshot showing the Ontosaurus browser in use. The browser is made of 3 frames. The top frame is a control panel. The lower right frame is used to display the contents of an ontology whilst the lower left frame is used for navigation purposes. Taken from (Swartout ,Patil, Knight, and Russ, 1996) by permission.

Figure 6 shows a screen snapshot from the main Ontosaurus browser. As can be seen from figure 6 the Ontosaurus browser is made up of 3 frames. The top frame is a control panel. The lower right frame is used to display the contents of an ontology whilst the lower left frame is used for

navigation purposes. Apart from the image which is used for documentation, all the graphics within the interface represent commands (edit concept, create instance, clone, etc.). The data is displayed plain, italic and bold styles of the same font.

Interface designers are also hindered by the fact that any undesirable functionality automatically provided by browsers can not be inhibited. For example, the "Back" button available in most browsers enables users to move to out-of-date states where deleted objects can still be referenced.

In the next section of this paper I describe WebOnto, a web based tool for collaboratively browsing, creating, and editing ontologies, designed to overcome some of the above problems by using a Java based client. It should be emphasised that the above criticisms are not aimed at the interface designers for the two systems described. I believe that in both cases the designers did a remarkable job considering the severe constraints imposed.

4. WEBONTO: SUPPORT FOR ONTOLOGY BROWSING, CREATION AND EDITING

WebOnto was designed to support the collaborative browsing, creation and editing of ontologies without suffering from the interface problems described in the previous section. In particular, WebOnto was designed to provide a direct manipulation interface displaying ontological expressions using a rich medium. WebOnto was aimed to be easy-to-use, yet have facilities for scaling up to large ontologies. Finally, WebOnto was designed to complement the ontology discussion tool Tadzebao.

WebOnto's architecture is composed of a central server and clients written in Java. The detailed description of the architecture is given in section 5. In the remainder of this section I will describe a scenario showing how Sarah uses WebOnto to browse an ontology. Sarah clones the top page (using the "Clone" button) within the "Essential Ontologies" notepad in figure 1 and then clicks on the image. This action creates the window shown in figure 7 below.
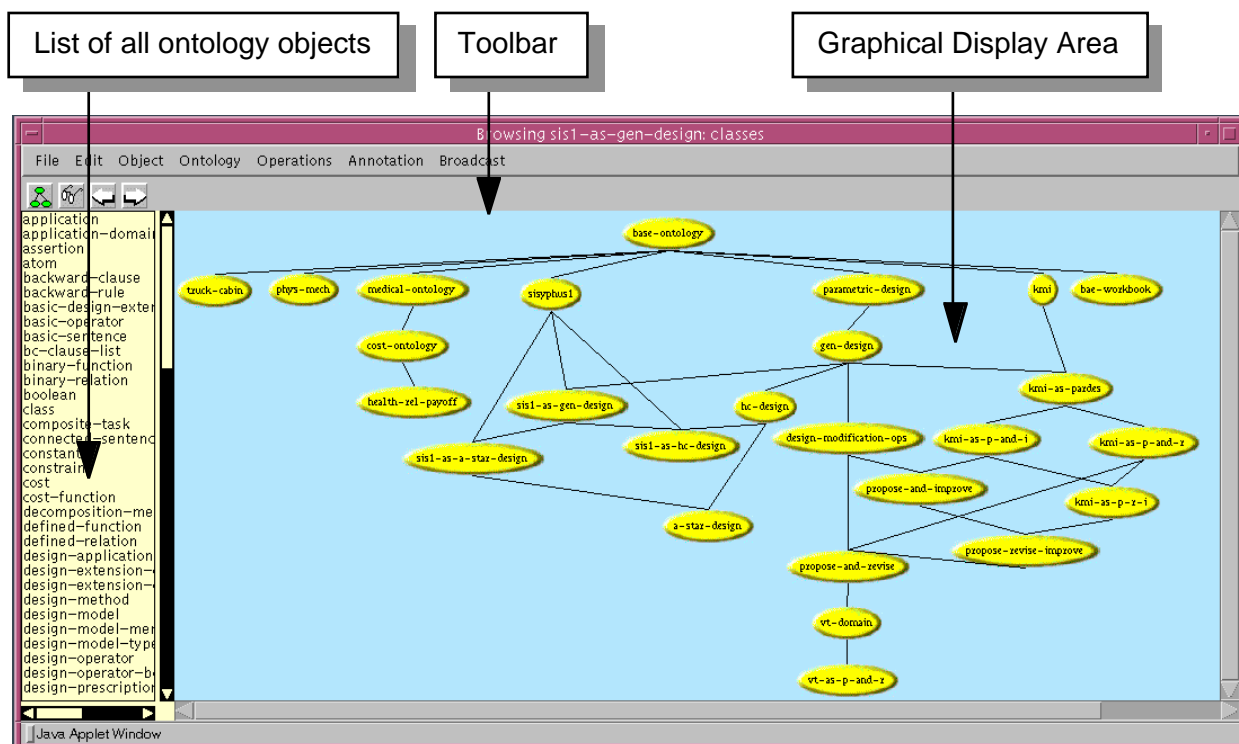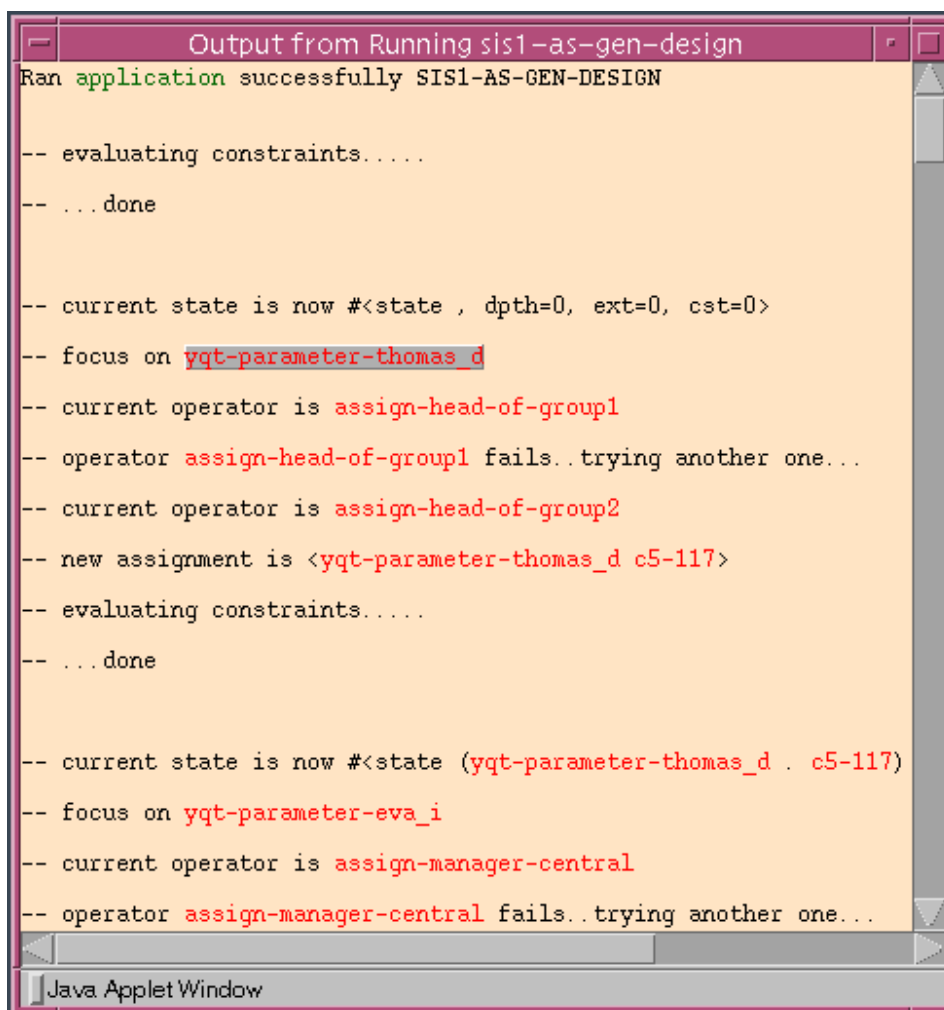

Figure 7. A screen snapshot showing the WebOnto window created by clicking on the image in the top page of the "Essential Ontologies" notepad in figure 1.

The graphical display area in figure 7 shows the hierarchy of currently loaded ontologies. There are several features of the display that should be emphasised:

- data (ontologies) are displayed using graphics,
- the icons can be moved (with rubber banding) or selected and operated on (i.e. figure 7 shows a direct manipulation interface with immediate feedback),
- local rendering enables the transfer of large images (the graph in figure 7 can be viewed as a large image).

Sarah decides to browse the sis1-as-gen-design ontology (the ontology representing a solution to the Sisyphus I room allocation problem (Linster, 1992)) by selecting the oval labelled "sis1-as-gen-design" and choosing "View Ontology" from the "Ontology" menu. The topleft window shows a filtered list of objects within the selected ontology. Sarah can elect to view all of the items within the ontology or one of: the classes, instances, functions, procedures or rules. She can also hide items which were inherited from ancestor ontologies. In figure 7 Sarah is viewing all the classes included in the sis1-as-gen-design ontology or one of its ancestors (gen-design, sisyphus1 etc.).

Figure 8. A screen snapshot showing the output from running the sis1-as-gen-design ontology. The text in red are the names of instances which can be selected and viewed.

As mentioned in the previous section, ontological expressions are represented in OCML (Motta, 1998) an operational knowledge modelling language originally developed in the context of the VITAL project (Shadbolt, Motta, and Rouge, 1993) to provide operational modelling capabilities for the VITAL workbench (Domingue, Motta, and Watt, 1993). One benefit of using OCML is that users can browse application models by running them and starting from a trace of the output. Sarah runs the sis1-as-gen-design application by choosing "Run" under the "File" menu. The ontology is run on the server and the output displayed in the window shown in figure 8.
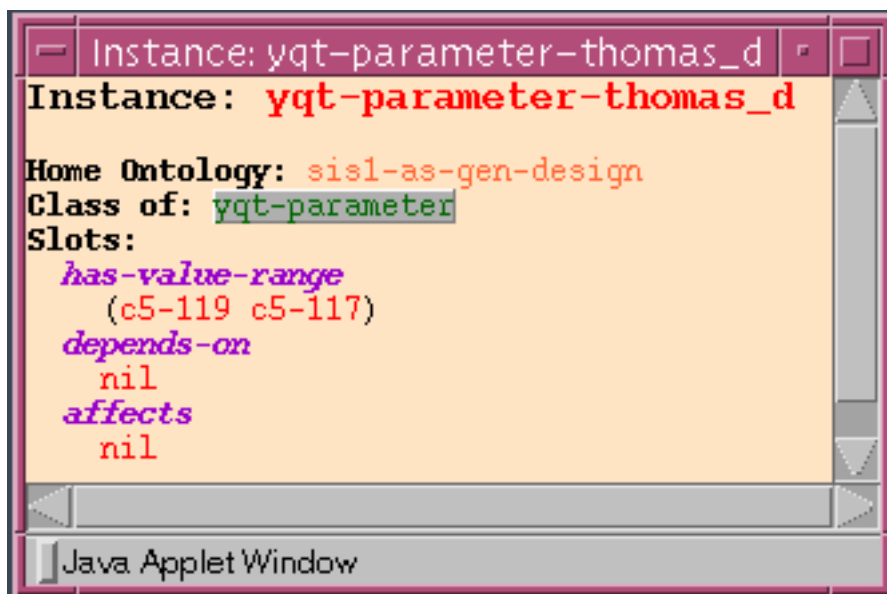
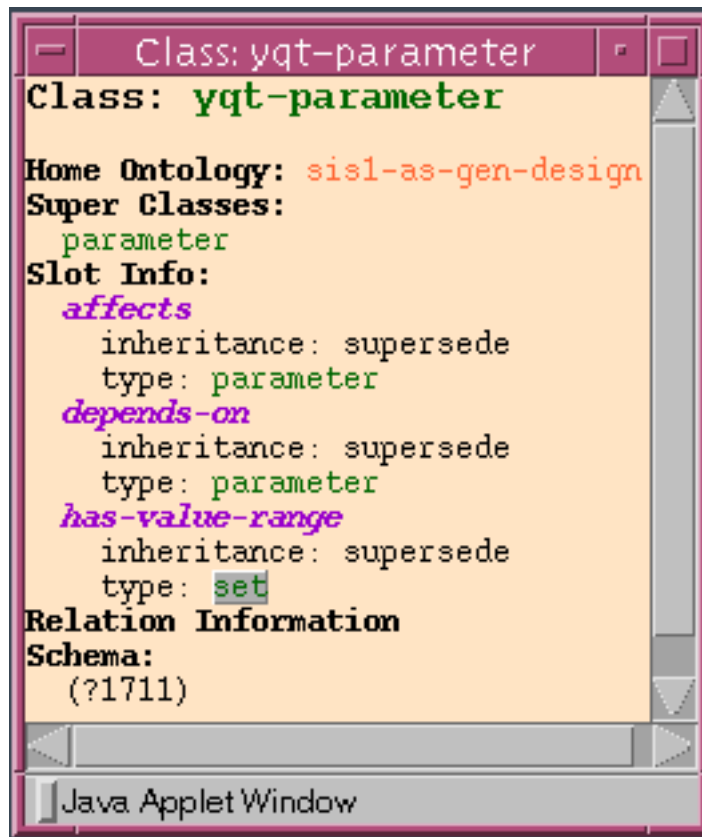**Figure 9. A screen snapshot showing the display of the instance "yqt-parameter-thomas_d". The colours indicate the following OCML types: red - instance, green - class, purple - relation, and orange - ontology.**

The window shown in figure 8 automatically highlights terms which are part of the currently viewed ontology. Highlighting is achieved by drawing the text in a colour, where the colour denotes the type of OCML construct represented. Within figure 8 the red items represent instances. Although it is possible to display coloured text in HTML, in current approaches (described in section 3) the display would have to be generated at the server. Within WebOnto the interface data is kept locally. This means that changes to the appearance of the display, for example, only highlighting certain types of OCML items, or changing the appearance of a highlighted item, can be made incrementally and without a call to the server.

Sarah selects the first occurrence of the instance yqt-parameter-thomas_d. The selection is shown as the grey box (this would not be possible with an HTML interface). Clicking on the 🔍 (inspect item) button changes the display in figure 8 to show the structure of the instance as shown in figure 9. We can see in figure 9 that yqt-parameter-thomas_d:

• was defined in the sis1-as-gen-design ontology,

• is an instance of the class yqt-parameter,

• has 3 slots which are: has-value-range depends-on and affects, with values (c5-c119 c5-c117), nil and nil.

Again note the use of colour to indicate type. Selecting the class yqt-parameter and clicking on the 🔍 button displays the structure of the class shown in figure 10.

Figure 10. A screen snapshot showing the class yqt-parameter.

Sarah selects the class "set" in figure 10 and then clicks on the ▲ (draw descendants) button changing the graphical display in figure 7 to show the class hierarchy for the class set as shown in figure 11. Note that the graphical display can be used because the image is rendered locally, minimising the amount of data which had to be transferred. After displaying the hierarchy Sarah decides that she would like to edit part of the hierarchy. She selects "base-ontology" in the list of currently used ontologies in the bottom left window, then selects "View Ontology" from the "Ontology" menu. Sarah then selects "Edit Mode" under the "Edit" menu. Changing to edit mode alters the appearance of the window in two ways, generating a direct manipulation environment for creating and editing OCML constructs. First a well appears next to the graphical display area containing a class and an instance icon. Second two edit specific icons appear:

▨ - edit current selected object, and

◩ - undo last edit.

Sarah starts to edit the class "set-as-list". She does this by selecting the class in the graphical area, then clicking on the ▨ button. The source code for the class is retrieved from the server and displayed in the new window at the bottom of figure 11. Once the text has been edited within the window Sarah can update the source on the server by clicking on the "Done" button. After editing an existing class Sarah decides to create a new class "ordered-set". She does this by dragging a new class icon from the well and then clicking on the icon with the "Shift" key held down. A new text edit window appears into which she types the text "ordered-set". At this point the window has the appearance shown in figure 11. Adopting an approach where the editing environment is very similar to the browsing environment follows the 'Edit-in-Place' metaphor (Rice, Farquhar, Piernot, and Gruber, 1996) adopted for the Ontolingua browser. The motivation for this metaphor is to minimise the number of different windows, icons and menus offered to the user. With WebOnto

the edit-in-place metaphor is pushed further as switches can be made between a 'pure' browse mode and an edit mode with only incremental changes to the display (i.e. the user does not have to wait for an entire page to be re-rendered).

Sarah finishes the session by quitting WebOnto. At this point the image in page 2 of the "Essential Ontologies" notepad changes to show the altered set class hierarchy. She then leaves a comment describing her alteration on the same page.
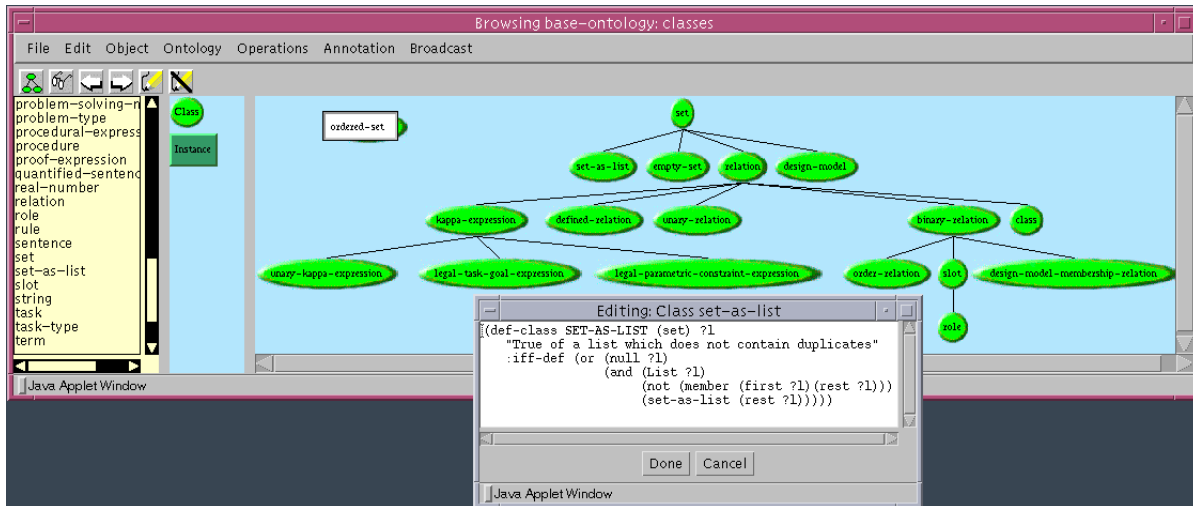
Figure 11. A screen snapshot showing a user editing part of the set class hierarchy in the base-ontology.

In the scenario outlined in this section we have seen how Sarah browsed the sis1-as-gen-design ontology from a trace of its output and then edited an ancestor ontology, adding a new page to the asynchronous discussion on "Essential Ontologies". It should be noted that WebOnto incorporates the sketching and synchronous communication tools described in section 2 enabling users to hold synchronous discussions about the details of an ontology. The synchronous communication tool relies on the fact that the architecture allows connections between a server and its clients to be left open.

5.   ARCHITECTURE

I shall now describe the architecture of the Tadzebao and WebOnto Server and of the Tadzebao and WebOnto Java clients. The servers are responsible for storing and maintaining the knowledge content and user dialogue, the clients are the interfaces to the stored content.

5.1. The Server

As can be seen in figure 12 the server is built on top of the LispWeb HTTP server, (Riva and Ramoni, 1996) a specialised HTTP server written in Common Lisp (Steele, 1990). In addition to implementing the standard HTTP protocol, the LispWeb server offers a library of high-level Lisp functions to dynamically generate HTML pages, a facility for dynamically creating image maps, and a server-to-server communication method.

The LispWeb passes on HTTP requests from the WebOnto and Tadzebao clients to the appropriate servers. The WebOnto server interacts with the ontology libraries represented in OCML and is responsible for:

• loading requested libraries,

• running applications,

• converting OCML structures into ASCII strings which can be sent via HTTP,

- sending requested OCML source code to WebOnto clients,
- updating server source code from WebOnto clients, and
- locking an ontology when it is being edited (sub-ontologies remain unlocked).

Locking an ontology ensures that conflicting edit operations, such as two users changing the same OCML structure, can not occur. WebOnto is designed to be used by collaborating teams and there are therefore no restrictions on who can edit an ontology.

The Tadzebao server is responsible for maintaining the notepad library, which contains the content of all the notepads, and delivering notepads to requesting Tadzebao clients.

The broadcaster sends incoming HTTP requests from a broadcasting client to registered receiving clients. In figure 12, seven clients are connected to the server. Clients 1 and 2 are using Tadzebao to construct dialogues. The third client is using WebOnto to edit an ontology. The fourth client is broadcasting to clients 5-7 through an open HTTP stream shown in red.
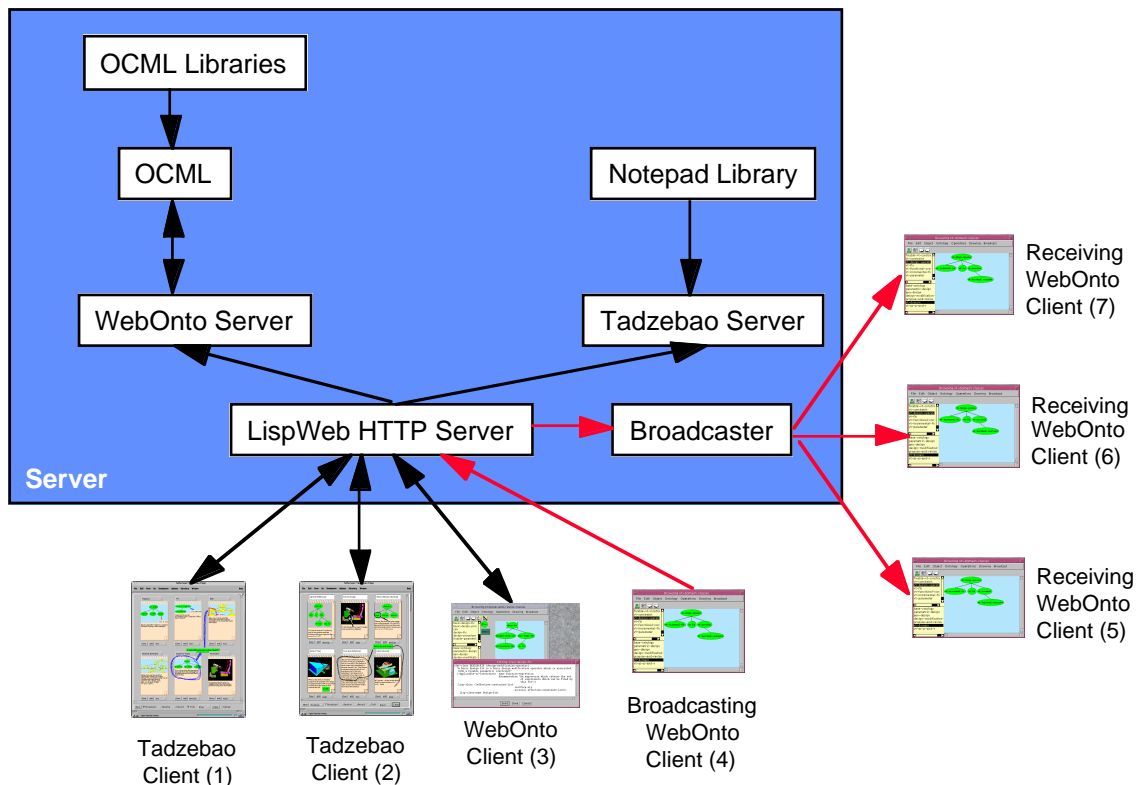
Figure 12. The architecture of the Tadzebao and WebOnto Server.

5.2. The Tadzebao Client

Figure 13 shows the architecture of the Tadzebao client. The client is responsible for presenting a consistent view of the selected notepads. The client ensures consistency by obtaining updates from the notepad library on the server when user actions which require an up-to-date view are performed. These special user actions include moving to the page beyond the current last page, selecting a notepad to view or cloning a page. Changes to a notepad such as adding a text comment result are immediately uploaded to the server and stored in the notepad library. Changes to a page can only be made in the session in which it was created.

The central part of the client is the Command Handler. Input events from the user are converted by the Event Handler to commands. The Command Handler dispatches drawing and labelling

commands to the Sketcher module and page specific commands such as cloning a page to the Notepad Handler. The display of a notepad page is controlled by the Page Displayer which uses a Text, GIF and OCML component drawing modules.

If the Tadzebao client is in broadcasting mode, the Command Handler copies all the commands to the Broadcaster. The broadcaster converts the commands into an ASCII string which is then passed to the HTTP Connector. The HTTP Connector deals with low level tasks such as opening up connections with the server, and listening for incoming data. In receive mode the receiver converts incoming ASCII strings into commands and passes them onto the command handler. The commands within the Tadzebao client are similar to the command objects (Gamma, Helm, Johnson, and Vlissides, 1995) used in the KSIMapper Netscape plug-in (Kremer, 1996) to facilitate the realtime sharing of concept maps.
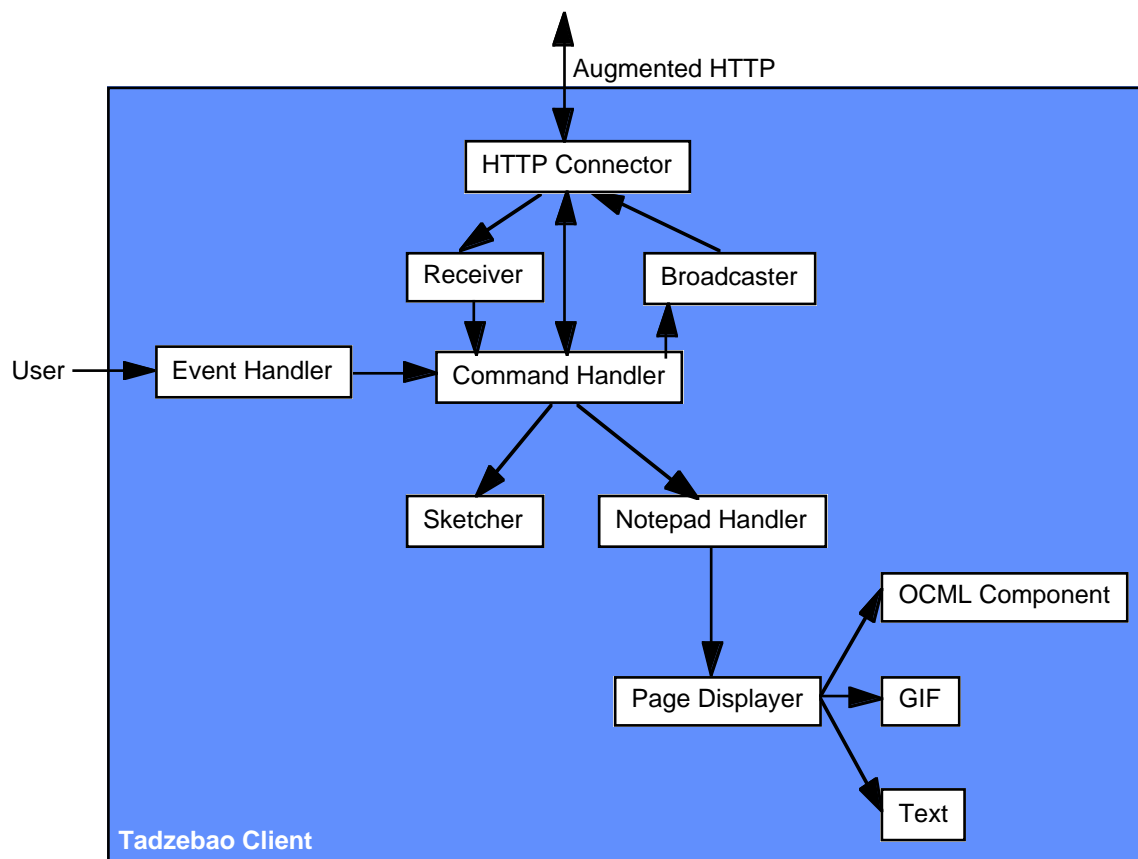
Figure 13. The architecture of the Tadzebao client.

5.3. The WebOnto Client

Figure 14 shows the architecture of the WebOnto client. It shares the Event Handler, Sketcher, HTTP Connector, Receiver and Broadcaster modules with the Tadzebao client which were described in the previous section.

Ontology browsing commands such as viewing a new ontology or inspecting an OCML structure result in requests being sent to the server for the relevant data. The server encodes OCML objects as ASCII strings. This string representation is unpacked by the OCML Structure Parser. When the server receives a request to view an ontology O it sends back the ontologies which O uses and headers for all the OCML items in O. Each header contains the item's name, type and if relevant the names of related items (such as a class' parents). The structure parser converts the string

representation of the headers into internal Java objects which are stored in the Headers Store module. The store is used by the Graphical Displayer to draw graphical representations of ontologies and by the Items Display to present the list of items. The store is also used by the

Structure Displayer. When a user clicks on the 🔍 button a request is sent to the server for detailed information on the selected OCML item. The Structure Parser sends the server's reply to the Structure Displayer, which formats the information as seen in figures 9 and 10. When displaying any output the Structure Displayer uses the Header Store to recognise OCML items.
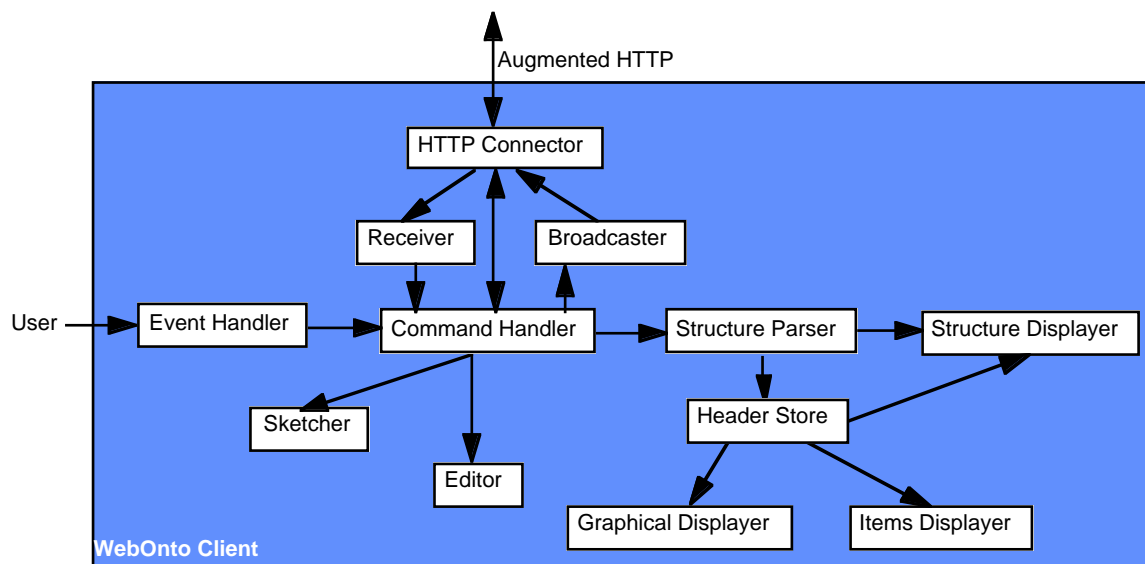
Figure 14. The architecture of the WebOnto client.

The architecture was designed maximise the interactiveness of Tadzebao and WebOnto whilst minimising the required bandwith. The principles underlying the design of the architecture are described in the next section.

6. DISCUSSION

In this section I will describe how the approach enabled interfaces to be constructed which did not suffer the difficulties described in section 3, after describing some additional issues which influenced the design of Tadzebao and WebOnto, I will then discuss how the approach supports ontology building communities.

6.1. Overcoming the Problems of 'Vanilla HTML'

6.1.1. Centralisation of All Data. A key aspect of the approach adopted in WebOnto was to move some data from the server onto the client. In general splitting the data is a tradeoff. Data kept on the server benefits from:

- always being consistent (there is only one copy), and
- reducing the computation/space requirements for the user.

but suffers from:

- being (relatively) slow to access, and
- scaling issues, no matter how large a central server, there will be a point at which the size of the user base causes too many simultaneous requests.

The approach used for splitting the data was based on four simple principles:

- keeping all interface data local,
- only transferring minimal data to support the interface,
- the central server holding the sole representation of OCML object structures, and
- embedding knowledge about OCML into the Java client.

The localisation of interface data is an obvious principle since a direct manipulation environment demands fast access to interface data. The data traffic between the server and the clients is kept to a minimum by only sending the minimally required data to support user interactions. Consistency problems are reduced by only having detailed representations of OCML objects on the server.

Embedding knowledge about OCML into the client (in the Structure Parser and the Structure Displayer) reduces the amount of data required to be sent by the server to represent ontological entities, and thus also minimises data traffic.

6.1.2. One-shot Connections. Just as with 'vanilla HTML' interfaces connections to the server are initiated by user interface actions. The difference is that the Tadzebao and WebOnto clients support incremental changes to the display, and connections can be left open to support synchronous communication. The command object representation of user actions minimises the data sent when broadcasting. For example, in broadcast mode moving the top node in the graphical display area of figure 11 would result in the ASCII string "move node 'set-as-list' to 129 217" being sent to the receivers.

Although it is possible for clients to periodically request information (e.g. after a period of time or after a pre-set number of user interface actions) the problem that new connections can only be initiated by the client remains, as Java applets can not create server sockets.

6.1.3. The Tyranny of the Browser. Java gives developers extensive control over the functionality, appearance and layout of the interface. In fact the Ontolingua browser now uses Java to draw class hierarchies and the next version will have a completely new user interface written in Java (KSL, 1997). The only remaining problems are those concerned with security. For example, it is not possible to write files to a local machine from within a Java v1.0.2 applet.

6.2. Additional Design Issues

In this paper I have considered ontology construction as a collaborative design problem viewed from an HCI perspective. Apart from overcoming the problems associated with HTML the guiding principle for the design of Tadzebao and WebOnto was that the tools for creating ontologies and the tools for discussing ontologies should be tightly integrated. In a number of discourse environments, such as e-journal publishing (Sumner and Buckingham Shum, 1998) and design rationale (Fischer, Lemke, McCall, and Morch, 1991) it has been shown that users need to be able to move easily between discussions and the artefact being constructed. Within a dialogue, utterances should be able to refer *directly* to the artefact. A consequence of this principle is that generic shared whiteboard tools, such as those found in Microsoft's NetMeeting (1998), were not considered.

The relationship between the Java based clients and the Lisp based server is similar to that of an interface and an underlying application. Technologies such as CORBA and Java's RMI (Sun, 1997) provide mainly *syntactic* support for linking *resuable object system components* (RMI is design to facilitate Java-to-Java communication). I therefore consider these technologies inappropriate for the type of connection required for Tadzebao and WebOnto and, moreover, of little consequence from an HCI perspective.

6.3. Supporting Ontological Communities

As I described at the start of section 2, the overall design goal for Tadzebao was to support both asynchronous and synchronous discussions on ontologies, allowing both free expression, using text and sketching tools, and direct reference to ontological constructs. Before moving further there is a need to reflect on the needs of our users. It is clear that ontologies are used by different sorts of

users with different requirements. In a separate context I have been investigating how the needs of novice programming communities can be supported on the web (Domingue and Mulholland, 1997a), and how the lessons learned could be applied to expert programming communities (Domingue and Mulholland, 1997b). There is a need to classify ontology users and to assess their needs. This type of study has been carried out in various design communities including software programmers (Curtis, Krasner, and Iscoe, 1988) and telephone engineers (Sumner, 1995).

In terms of technology support I believe that our goal should be to build *socioware* (Chen and Gaines, 1996) for the ontology community. Where socioware is defined as:

> "computer-mediated environments for supporting community -wide processes which expedite virtual interactions" Chen and Gaines (1996)

where virtual interactions are virtual in the sense of being in a virtual space and in the fact that the intention to collaborate may not have existed when the artefact was created.

6.3.1. Software Reuse. Software reuse involves finding segments of code which implement a desired *behaviour*. Ontologies are useful because they can be used to provide a multi-level formal description of code stored within a library (Neches, Fikes, Finin, Gruber, Patil, Senator, and Swartout, 1991). Ontologies can also aid in configuring software components. Users search for software components by browsing ontologies. This can be problematic for users who are not familiar with the ontologies. The approach described in this paper provides two forms of extra help for these unfamiliar users. Firstly, because OCML is operational and provides hooks to Common Lisp (Steele, 1990), running applications can be made available on the Web. Users can examine the behaviour of applications using the trace output and by direct manipulation move to the relevant part of an ontology. Future work will examine how the trace output can be improved using the Software Visualization (SV) techniques developed within the Vital Workbench (Domingue, 1995) and the WWW-SV infrastructure constructed for teaching programming on the web (Domingue and Mulholland, 1997b). The dialogues stored with Tadzebao form a design rationale (Moran and Caroll, 1996) for ontologies and are thus a valuable source of extra documentation for users.

The application of Tadzebao and WebOnto to software reuse will carried out on the IBROW[3] project (IBROW[3], 1997). IBROW is an Esprit Open LTR project with the objective of developing intelligent brokers that are able to distributively configure reusable components into knowledge systems through the WWW.

7. RELATED WORK

The Ontolingua server (Farquhar, Fikes, Pratt, and Rice, 1996) was a landmark system for the ontology community. It has over 1000 registered users of which 150 are described as serious (Rice, Farquhar, Piernot, and Gruber, 1996). The overall design goal for the Ontolingua server was to facilitate the collaborative development of ontologies. As the name suggests the architecture consists of a central server which uses the HTTP protocol to communicate with remote users logged on via a web browser. The Ontolingua interface was created using HTML widgets, and is described in more detail in section 3.

The Ontosaurus server (Swartout, Patil, Knight, and Russ, 1996) is implemented using CL-HTTP (Mallery, 1994) a Common Lisp based Web server, the Loom knowledge representation system (Macgregor, 1991) and Lisp code to interface between the two. Like the Ontolingua server the interface was created using HTML widgets and is described in more detail in section 3.

The Knowledge Science Institute at the University of Calgary has carried out considerable work on sharing concept maps on the Web. Concept maps are a diagrammatic form of knowledge representation which can be used informally or can have formal semantics (Gaines, 1991). Access to KMap (Gaines, and Shaw, 1995), a Macintosh based concept mapping tool written in a C++ class library (Gaines, 1994), over the web was provided by registering KMap as a client helper with Netscape, defining a new MIME type, and using Apple events. Access to concept maps on non-Macintosh platforms was enabled through the use of CGI scripts and a concept map to GIF converter. More recent work has resulted in the KSIMapper program (Kremer, 1996) being

available as a Netscape plug-in using JavaScript (Netscape, 1996a) and LiveConnect (Netscape, 1996b). The plug-in uses command objects to facilitate the realtime sharing of concept maps. A Java implementation of the KSImapper has also been completed (Flores-Mendez, 1997).

The SHADE project (McGuire, Kuokka, Weber, Tenenbaum, Gruber, and Olsen, 1993) supports the sharing of engineering knowledge through the internet using a variety of technologies including the Ontolingua server, the KQML agent-communication language (Finnin, Weber, Widerhold, Genesereth, Fritzson, McKay, McGuire, Shapiro, and Beck, 1992), and ServiceMail™.

8. SUMMARY

Since ontologies represent a shared point of view ontology construction is often carried out by a group of people. The use of WWW technology to facilitate communal construction when the group is spread over large distances has attracted considerable attention. In this paper I have presented two systems which in different ways address deficiencies with current approaches to using WWW technology for communal ontology construction. An integral part of the communal design process is dialogue about the artefact under construction. It is important that discussions about the artefacts, and representations of the artefact itself are integrated (Fischer, Lemke, McCall, and Morch, 1991). Tadzebao supports this type of integrated dialogue using multimedia notepads which can contain a mixture of text, GIF images and ontologies.

Using HTML to create interfaces imposes severe constraints on interface designers. There are problems caused by: the need to centralise all of the data, the restriction of one-shot connections, and the fact the interface sits within a browser which can only be partially controlled. WebOnto was designed to support the collaborative browsing, creation and editing of ontologies without suffering from the interface problems generated by HTML. In section 4 I described how WebOnto provides a direct manipulation interface using graphical representations to present ontology constructs.

Currently dialogues within WebOnto and Tadzebao consist of ontology representations, graphics and text. Future work will integrate a streaming audio system currently being constructed as part of the KMi Stadium project (Eisenstadt, Buckingham Shum, and Freeman, 1996). KMi Stadium is a Java based application that explores the use of large-scale teleprescence focused on the broadcasting of realtime audio.

Both Tadzebao and WebOnto are currently being evaluated in-house, within the confines of our ongoing TELEMATICS project HCReMa, and by the University of Madrid. Once the feedback from the evaluations have been incorporated into the tools we will set up a dedicated server within KMi for open use. It is our hope that this will be another step to creating socioware for the ontology construction community.

REFERENCES

Banecek, J., Drvota, J., and Valasek, M. (1996) Knowledge Level Description of Initial Vehicle Design. Encode: Environment for Configuration Design Technical Report TR-Encode-DccS-2-96. Copernicus Project CIPA-CT94-0149.

Borst, P. Akkermans, H. and Top, J. (1996) Engineering Ontologies. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14, 1996.

Chen, L., L., and Gaines, B. R. (1996) Knowledge Acquisition Processes in Internet Communities. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14, 1996.

Curtis, B., Krasner H. and Iscoe N. (1988) A Field Study of the Software Design Process for Large Systems. Communications of the ACM, Vol. 31, Iss. 11, pp. 1268-1287.

Domingue, J., Motta, E. and Watt, S. (1993) The Emerging Vital Workbench. In Ed. Aussenac, N., Boy, G., Gaines, B., Linster, M., Ganascia, J.-G. and Kodratoff, Y. Knowledge Acquisition for Knowledge-Based Systems 7th European Workshop, EKAW'93 Toulouse and Caylus, France, September, pp. 320-339, Springer-Verlag.

Domingue, J. (1995) Using Software Visualization Technology in the Validation of Knowledge Based Systems. Proceedings of the 9th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, February 26-March 3, 1995.

Domingue, J., and Mulholland, P. (1997a) Teaching Programming at a Distance: The Internet Software Visualization Laboratory. The Journal of Interactive Media and Education. 7 (1).

Domingue, J., and Mulholland, P. (1997b) Fostering Debugging Communities on the Web. Communications of the ACM, Vol. 40, No. 4, pp. 65-71 April, 1997.

Eisenstadt, M., Buckingham Shum, S. and Freeman, A. (1996) KMi Stadium: Web-based audio/visual interaction as reusable organisational expertise. In: Proceedings of Workshop on Knowledge Media for Improving Organisational Expertise, 1st International Conference on Practical Aspects of Knowledge Management, Basel, Switzerland.

Falasconi, S. Lanzola, G. and Stefanelli, M. Using Ontologies in Multi-Agent Systems. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14, 1996.

Farquhar, A., Fikes, R., Pratt, W., and Rice, J. Collaborative Ontology Construction for Information Integration. Knowledge Systems Laboratory Department of Computer Science, KSL-95-63, August 1995.

Finnin, T., Weber, J., Widerhold, G., Genesereth, M., Fritzson, R., McKay, D., McGuire, J., Shapiro, S. & Beck, C. (1992) Specification of the KQML Agent-Communication Language. The DARPA Knowledge Sharing Initiative External Interfaces Working Group.

Fischer, G., Lemke, A. C., McCall, R. and Morch, A. (1991) Making Argumentation Serve Design. Human Computer Interaction, Vol. 6, Iss. 3-4, pp. 393-419.

Flores-Mendez, R., (1996) Distributed Concept Mapping Collaboration using Java. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14.

Gaines, B. R. (1991) An Interactive Visual Language for Term Subsumption Languages. Proceedings of IJCAI-91, Sydney, Australia, August 24030, 1991.

Gaines, B. R. (1994) Class library implementation of an open architecture knowledge support system. International Journal of Human-Computer Studies 41(1/2) 59-107, 1994.

Gaines, Br. R., and Shaw, M. L. G. (1995) WebMap: Concept Mapping on the Web. World Wide Web Journal 1(1) pp. 171-183.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, Reading Massachusetts.

HCReMa (1997) Project HC 3103, Telematics Applications Programme http://www.vision.auc.dk/CHI/projects/HC-REMA/HCREMA.html

IBROW[3] (1997) IBROW3: An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web. http://www.iiia.csic.es/~richard/esprit/ibrow3-sum.html

Kremer, R. (1996) Toward a Multi-User, Programmable Web Concept Mapping "Shell" to Handle Multiple Formalisms. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14, 1996.

KSL (1997) KSL Services Forthcoming Release notes at http://www-ksl-svc.stanford.edu:5915/doc/release-notes.text.

Linster, M. (1992) Sisyphus '92: Models of Problem Solving. GMD Technical Report.

MacGregor, R. (1991) The Evolving Technology of Classification-Based Representation Systems. In Ed. J. Sowa Principles of Semantic Networks: Explorations n the Representation of Knowledge. Morgan Kaufmann.

Mallery, J. C. (1994) A Common Lisp Hypermedia Server, in Proceedings of the First International Conference on the World-Wide Web. Geneva, CERN, May 25, 1994.

McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., Olsen, G. R. (1993) SHADE: Technology for Knowledge-Based Collaborative Engineering. Journal of Concurrent Engineering: Applications and Research (CERA), 1(2), September, 1993.

Microsoft Corporation (1998) http://www.microsoft.com/netmeeting/.

Moran, T. P. and Carroll, J. M. (1996) Design Rationale: Concepts, Techniques, and Use, Ed., Lawrence Erlbaum Associates, Hillsdale, NJ.

Motta E. (1998) Reusable Components for Knowledge Models. PhD Thesis. Knowledge Media Institute. The Open University, UK. Available at (http://kmi.open.ac.uk/people/motta/thesis.html).

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. R. (1991) Enabling Technology for Knowledge Sharing. AI Magazine, 12(3) pp. 16-36.

Netscape Communications Corporation (1996a) Netscape JavaScript. http://home.netscape.com/comprod/products/navigator/version_2.0/script/index.html.

Netscape Communications Corporation (1996b) Developer Information: LiveConnect. Netscape Navigator 3.0 Developer Information, http://home.netscape.com/comprod/products/navigator/version_3.0/.

Rice, J., Farquhar, A., Piernot, P. and Gruber, T. (1995) Using the Web Instead of a Window System. Knowledge Systems Laboratory Department of Computer Science, KSL-95-69, September 1995. http://www-kslsvc.stanford.edu:5915/doc/papers/ksl-95-69/ksl-95-69.html

Riva, A. and Ramoni, M. (1996) LispWeb: a Specialised HTTP Server for Distributed AI Applications. Computer Networks and ISDN Systems, 28,7-11 (1996), 953-961.

Shadbolt, N., Motta, E., Rouge, A. (1993) Constructing Knowledge-Based Systems. IEEE Software, 10, 6, pp. 34-39, November 1993.

Steele, G. L. (1990) Common Lisp the Language. 2nd Edition. Digital Press.

Sumner, T., (1995) The High-Tech Toolbelt: A Study of Designers in the Workplace. In the Proceedings of Human Factors in Computing Systems (CHI '95), Denver, CO pp. 178-185, May 7-11, 1995.

Sumner, T. and Buckingham Shum, S. From Documents to Discourse: Shifting Conceptions of Scholarly Publishing. Proceesings of Human Factors in Computing Systems (CHI '98), Los Angeles, April 18-23.

Sun Microsystems (1996) The Java Programming Language, Sun Microsystems Inc. http://java.sun.com/

Sun Microsystems (1997) Remote Method Invocation. http://java.sun.com/products/jdk/1.2/docs/guide/rmi/

Swartout, B., Patil, R., Knight, K. and Russ, T. (1996) Toward Distributed Use of Large-Scale Ontologies, USA. Proceedings of the 10th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, November 9-14, 1996.

van Heijst, G. (1995) The Role of Ontologies in Knowledge Engineering. PhD Thesis. University of Amsterdam.

Yost, G. R. (1992) *Configuring Elevator Systems.* Workplace Integration Technologies Group, Digital Equipment Corporation, 11 Locke Drive (LM02/K11),Marlboro, MA 01752. December. Unpublished.