Chapter 7

The HyperContext Prototype

7.1 Introduction

The HyperContext prototype has been implemented to demonstrate some of the concepts introduced and discussed throughout this thesis. It also provides a platform for experimentation with the short-term user model based on paths of traversal and from which Adaptive Information Discovery can derive a user query to locate information which may be of interest to the user.

The prototype is an interface to a HyperContext hypertext which has been automatically created by converting a World Wide Web site. Apart from discussing the prototype we also discuss some design issues for the implementation of a full-blown version of HyperContext. In Chapter 8 we describe the process of automatically converting a Web site into a HyperContext hypertext.

7.2 HyperContext features included in the prototype

The prototype is a client-server implementation incorporating the adaptive features of the HyperContext framework, together with the three search paradigms (Traditional Information Retrieval, Information Retrieval-in-Context, and Adaptive Information Discovery), the model of the user's short-term interests, and adaptive navigation services through link and path recommendation. Currently, users can inspect the short-term user model, but they cannot modify it. With further development, the prototype would include the adaptable features of the HyperContext framework which enable users to include new documents into the HyperContext hypertext, and create and modify links and interpretations.

The HyperContext prototype utilises the external services of an information indexing and retrieval system called SWISH-E [48] for all three of HyperContext's search paradigms, but it does not utilise an external user modelling system to provide a long-term user model. The short-term user model is implemented as part of the HyperContext client.

7.3 Prototype implementation details

HyperContext is implemented using Sun Microsystems' Java version 1.0.2 [87]. The choice of software platform was influenced by Sun's publicised announcement of Java's platform independence and portability. Although this has not been completely achieved (there are differences between the Java Run-Time Environments on different platforms), Java's time-to-implementation capabilities and its ease of inter-operability with Web clients and servers made it an obvious choice. The decision to develop using Java version 1.0.2 was enforced by the lack of backward compatibility of later versions of the Java compiler. Thousands of lines of Java code would have required re-writing in order to upgrade to later versions of Java. However, although HyperContext requires the Java 1.0.2 compiler, it can be executed using any version of the Java Run-Time Environment. Indeed, due to bugs present in earlier versions of the Java Run-Time Environment, it is recommended that HyperContext is run using the latest available version of Java.

HyperContext is implemented using a socket-based client-server approach, with the HyperContext server running as a stand-alone Java application listening on a communications port. The client is implemented as a Java applet which is embedded in a Web browser window frame. For security reasons, this approach limits the direct Java communication from the client to the HyperContext server which hosts the Java applet. However, a slight modification to the design of the client would allow unrestricted communication between the client and any HyperContext server (Section 7.5).

A Web site (The World Wide Web Consortium's Web site at www.w3.org) has been partially converted and is mirrored on zeus.cs.um.edu.mt, a UNIX Workstation belonging to the Department of Computer Science and Artificial Intelligence at the University of Malta. Converting a Web site into a HyperContext hypertext involves adding information to HyperContext's Object and Structure Layers - the actual HTML documents reside outside of HyperContext and are retrieved according to the specified protocol. In this case, the documents are retrieved using the HyperText Transfer Protocol (HTTP) via a Web server also running on zeus.cs.um.edu.mt. The Web-to-HyperContext conversion process itself is discussed in Chapter 8.

7.4 The HyperContext user interface

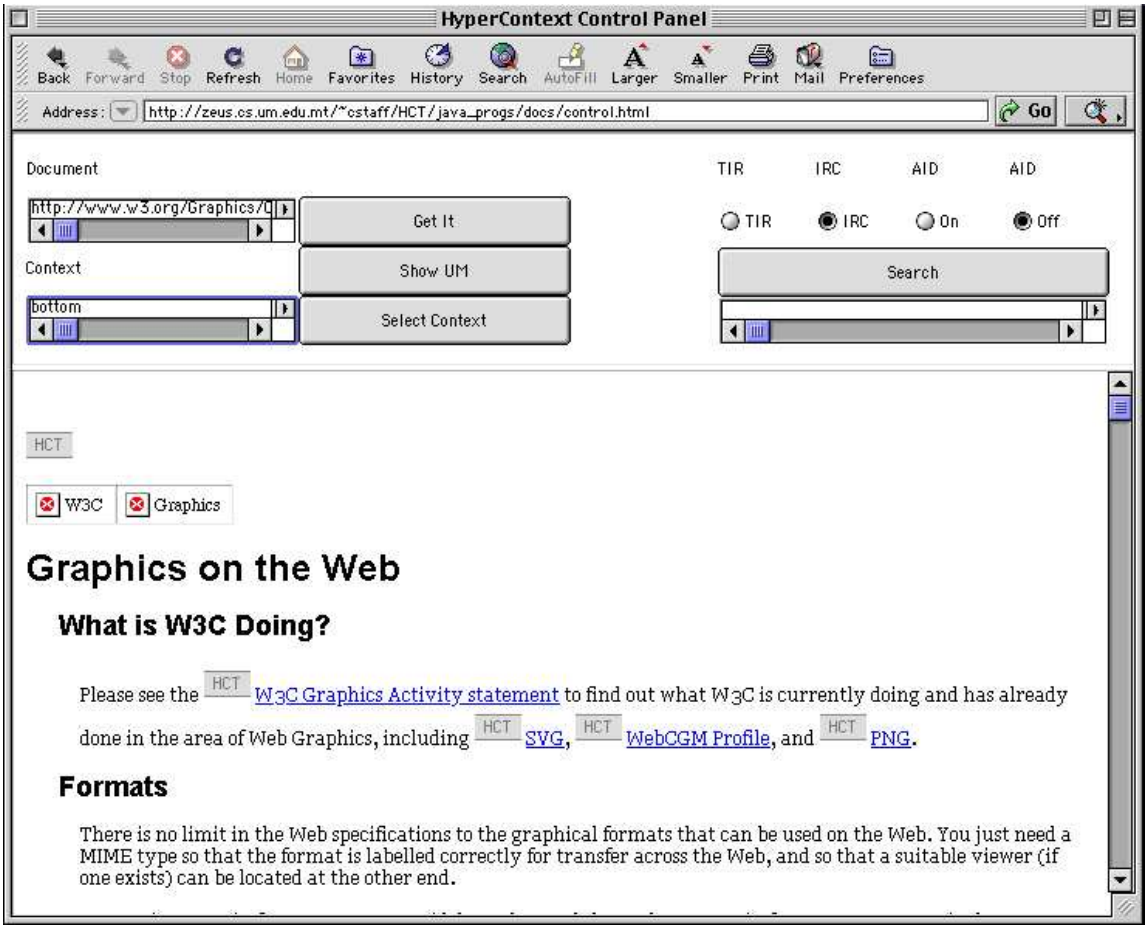

Figure 7.1: The HyperContext user interface

Figure 7.1 shows the HyperContext User Interface during a user session. The Web browser used is Microsoft Internet Explorer 4.5 running on an Apple Power Macintosh 8500. The browser window is split into two frames. The upper frame contains the HyperContext Control Panel and the lower frame, called the HyperContext Document Frame, contains an interpreted document.

7.4.1 The HyperContext Control Panel

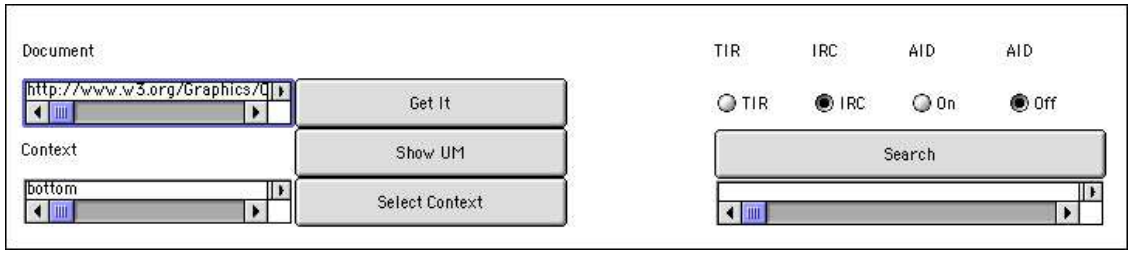

Figure 7.2: The HyperContext Control Panel

The HyperContext Control Panel is the interface to HyperContext's Presentation Layer. It allows users to supply the URL of a document and the context within which to interpret the document. The appropriate interpretation is retrieved for the user when the Get It button is clicked. The Document and Context text fields in the Control Panel show the user the URL and context used to provide the interpreted document visible in the Document Frame. The Control Panel also provides the user with a text field to enter search terms, and radio buttons to select the TIR or IRC search method. The status of Adaptive Information Discovery is shown through radio buttons. If Adaptive Information Discovery is active, a floating window contains a list of "See Also" references to superficially relevant document interpretations (Section 7.4.3).

The short-term user model can be inspected by clicking on the Show UM button (figure 7.2), and the user can change the context of the currently visible document by clicking on the Select Context button. Users can travel backwards and forwards through interpretations of documents that have already been visited in the current context session using the browser's "Back" and "Forward" menu selections, in which case the appropriate interpreted document is reloaded into the Document Frame and the URL and context text fields in the Control Panel are updated.

7.4.2 The HyperContext Document Frame

The lower frame of the browser window shown in figure 7.1 contains the interpreted document that the user has instructed HyperContext to retrieve. Users follow links that are active by clicking on the HCT button preceding the blue text. Blue text indicates that the text was source anchor text in the original HTML document. It is possible that in a different interpretation of the document the blue text is not preceded by an HCT button. This allows users to see which links are not available in the current interpretation of the document.
When HyperContext recommends a link to follow, the HCT button is replaced by an HCTr button (figure 7.3), and a navigational link is provided at the top of the document. The navigational link provides rapid relocation to the HCTr recommended link in the document.

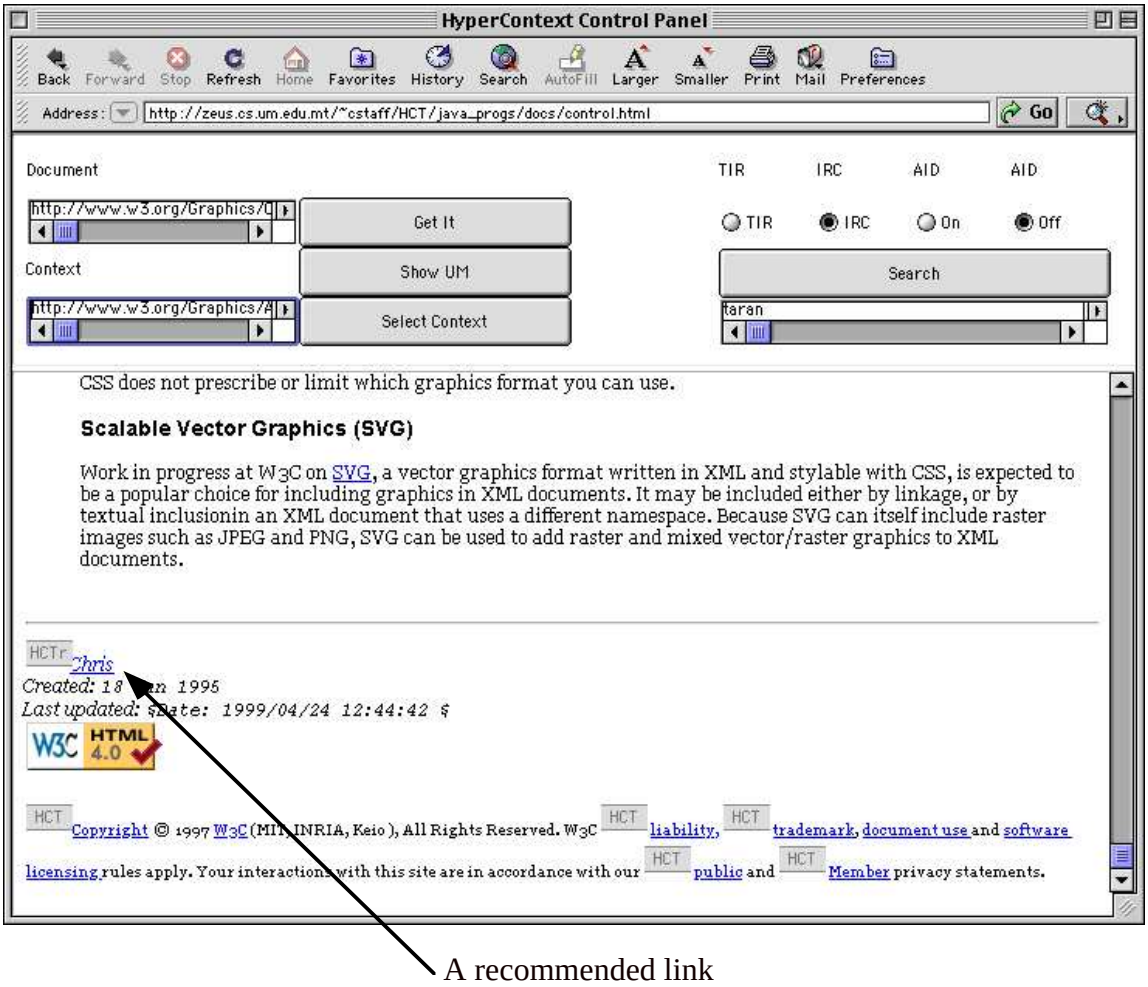

Figure 7.3: Recommended links in an interpreted document are shown as HCTr buttons

Link hiding versus link disabling

In the HyperContext prototype, we elected to disable links which cannot be followed in the current interpretation of a document so that we can see the changes that HyperContext makes to our perceptions of hyperspace as we browse. We do this by underlining the text and colouring it blue (so that it looks like a WWW hypertext link), but we do not provide a mechanism (via an HCT button) for users to actually follow the link. It is highly debatable whether link disabling would be recommended for a full implementation of HyperContext, for a number of reasons. One of the goals of adaptive hypertext is to reduce the cognitive overload on a user by reducing the number of links which can be followed from a document [32]. In HyperContext and other adaptive hypertext systems, this can be achieved by revoking the user's ability to follow links to information which is likely to be non-relevant to the user's information seeking task. However, especially in ITS-based adaptive hypertext systems, a link is made visible as soon as the user would be

able to understand the concepts contained in the destination document or as soon as the information becomes relevant [25]. The link is never again hidden from the user but it may be marked as having been already visited and not relevant to the current information seeking task [20, 28, 25]. In HyperContext, a link is disabled if it is likely to lead to information which is not relevant to the context of access. However, if a user re-visits the same document several times, each time in a different context, then, if a link hiding approach is taken, the link may appear and disappear in separate invocations. There is evidence that link disabling is disliked by and confusing to users [28]. A link hiding approach that is not used in conjunction with incremental linking may be detrimental to users because they are unable to construct a consistent mental model of hyperspace [68]. Although there may be considerable disadvantages to link disabling and link hiding the benefits to the user of hiding and disabling links to reduce the size of the perceived hyperspace are also numerous. Further research is required to determine whether a user's mental model of the hyperspace with which they are familiar can be managed separately from the reduced view of hyperspace that an adaptive hypertext system offers the user.

In HyperContext, there is an additional potential problem which may precipitate the selection of link hiding over other adaptive navigation technologies, even though it has its disadvantages. Link source anchors in different interpretations of the same document may overlap. Showing users inactive link source anchors that overlap with each other and with active link source anchors will probably be very confusing. Furthermore, the link destinations themselves may change from one interpretation to another. In HyperContext, a user can always select a different context in which to interpret the current document. It may be possible to annotate disabled links with details of the context in which they are active (together with information about the destination document to which they are connected). Further research will determine whether inaccessible links should be displayed to users and how, and, if not, how the negative effects of users constructing inconsistent mental models can be overcome.

7.4.3 HyperContext's "See Also" Links

When the user has activated Adaptive Information Discovery, HyperContext estimates the user's current interests and automatically searches for superficially relevant interpretations of information. Whenever a relevant interpretation is found, links are provided in the "See Also" references window, as shown in figure 7.4. The user can access a recommended interpretation by selecting it in the list and then clicking on the Fetch button. The interpreted document is automatically loaded into the Document Frame.

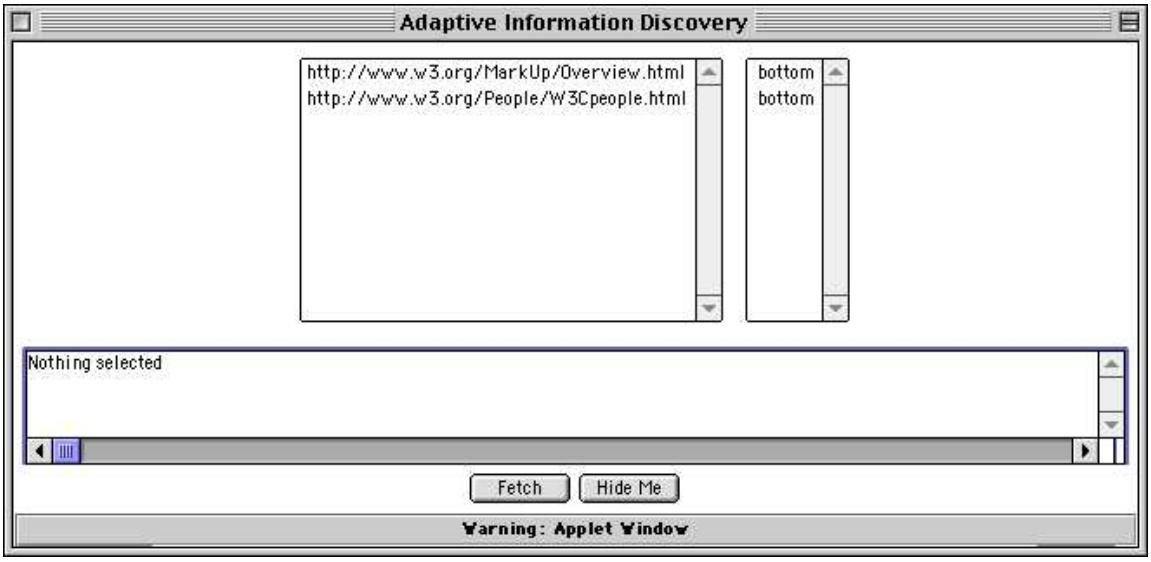

Figure 7.4: The "See Also" References floating window

7.4.4 HyperContext Status Information

A floating window contains status information about the user's current request and the HyperContext server (figure 7.5).

Figure 7.5: The HyperContext Status window

The user is informed of improperly constructed URLs when requesting a document to be retrieved, and also whether the targeted HyperContext server is in a position to respond to requests. For the prototype only, the user is able to remotely restart the HyperContext server if it is not responding. Progress messages are also displayed to the user to indicate whether the server is currently engaged in servicing a user request.

7.5 HyperContext Prototype Implementation Model (2)

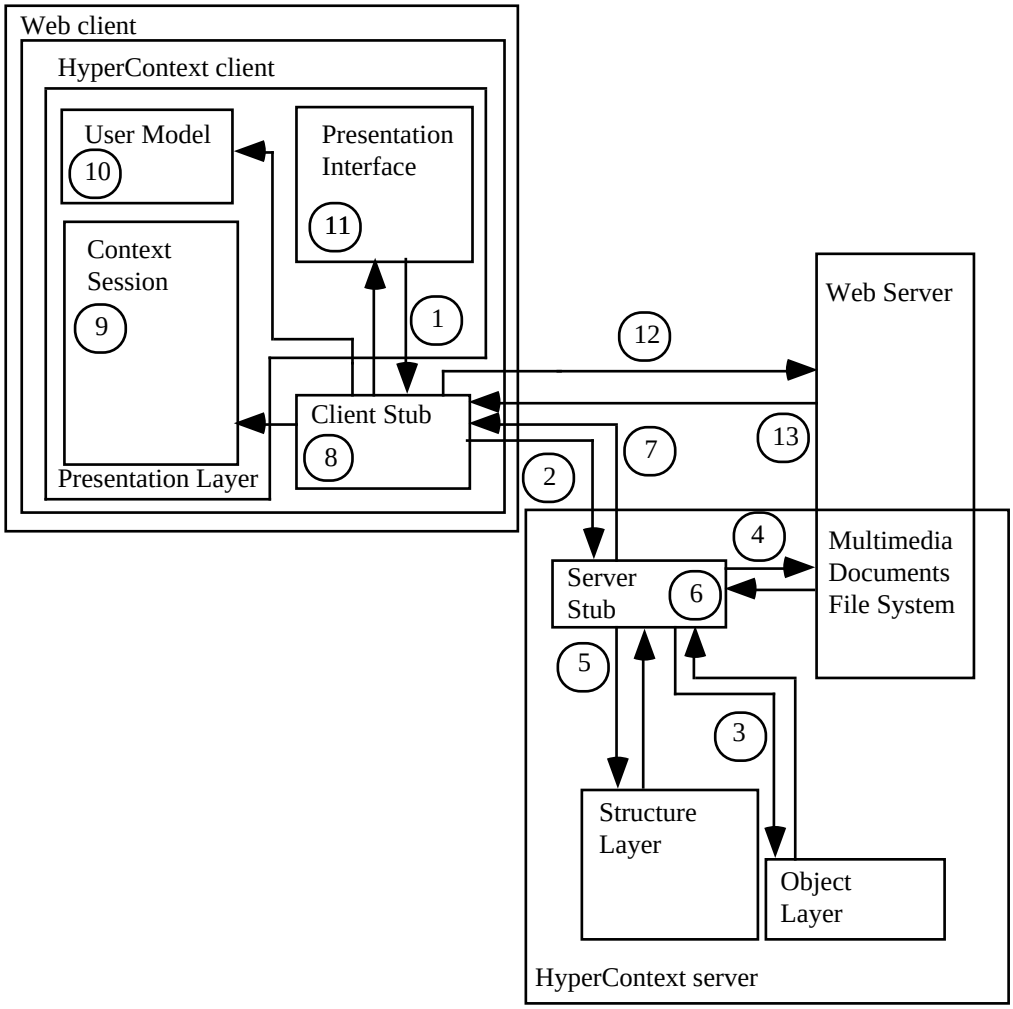

Figure 7.6: Implementation model of the HyperContext prototype

Figure 7.6 is a slight modification to the partial implementation model of HyperContext presented in figure 5.1. The model does not take into account interactions with the external information retrieval system, SWISH-E, but serves to show the information and control flow while the user browses through a HyperContext hypertext. The differences between the model described in Chapter 5.3 and the prototype are the inclusion of a Web server, and corresponding interactions between it and the HyperContext client and server. In the revised model, the HyperContext client is embedded within a Web client, steps 4 and 7 are modified, and two new steps (12 and 13) are introduced.

In the prototype, interpreted documents are not retrieved by the HyperContext client, but by the Web client, at the HyperContext client's request. The major reason for this is that the HyperContext client is embedded within a Web browser. The easiest implementation route for the display of an interpreted document is for the Web browser to request a

modified version of the HTML document which can be loaded directly into the document frame of the browser window. When a user requests a document in context through the HyperContext client (Section 7.4.1) the URL of the interpreted document must be returned to the client so that it can instruct the Web client to request the appropriate document from the Web server. Step 4 in figure 7.6 is the creation on the Web server of a temporary file containing the interpreted document. Step 7 is the client notification of the URL of this temporary file. Steps 12 and 13 represent the request for and the retrieval of the temporary file via HTTP.

The HyperContext Control Panel is accessible as a Java applet from the URL http://zeus.cs.um.edu.mt/~cstaff/HCT/java_progs/docs/control.html. Due to Java security, this restricts the client to communicating with a Java server located at the same domain. In a full implementation of HyperContext this would be a severe restriction, but it can easily be overcome by splitting the HyperContext Control Panel into two parts - an applet visible in a Web browser frame which provides the interface to HyperContext (the Control Panel), and a HyperContext client running as a stand-alone Java application on the client computer. The applet would also be loaded from the client which means that the applet can communicate freely with the HyperContext client. As the HyperContext client permanently resides on the client side, it would be free to communicate with any remote HyperContext server. At the time the prototype was first constructed, a Java Application Run-Time Environment for the Apple Macintosh platform was not available, so the HyperContext client was embedded within the HyperContext Control Panel applet.

Although a URL referring to any machine or domain name can be submitted by the user via the Control Panel, the HyperContext client modifies the URL to point to a mirror site hosted by zeus.cs.um.edu.mt. Consequently,

http://www.w3.org/Graphics/Overview.html

becomes

http://zeus.cs.um.edu.mt/~cstaff/HCTMirrors/neww3/www.w3.org/Graphics/Overview.html.

The user is unaware of the redirection of her request.

7.5.1 Communication between modules

Communication between the HyperContext client and the HyperContext server is always initiated by the client which waits for a response from the server on the same communication channel. The HyperContext client informs the Web client of the URL of the document to request from the Web server once the HyperContext server has created a

temporary file containing the interpreted document on the Web server. The Web client informs the HyperContext client whenever the user has changed the contents of the document frame by invoking the Web client's "Back" or "Forward" menu items to revisit interpreted documents, so that the Control Panel can update the URL and context in the Document and Context text fields. Finally, the Document Frame must also inform the Control Panel when a user follows a link in the interpreted document by clicking on an HCT or HCTr button.

Communication between components of the Web and HyperContext clients is supported through the use of a shared Java object called middle. middle contains the URL and context of the most recently invoked document interpretation. The Control Panel monitors middle and when it detects a change it either modifies the contents of the Document and Context text fields, or else it initiates a request for the interpretation of a document, depending on the type of change. Whenever an interpretation is loaded into the document frame, it announces its identity through middle, so that if the user has navigated through interpretations using the Web client's navigation tools (for example, selecting "Back", "Forward", or an interpreted document from the browser's History list), then the Control Panel's information about where the user is in the hyperspace can be maintained.

The HCT and HCTr buttons, which represent normal links and recommended links respectively, are parameterised Java applets which update middle when activated. The buttons know which document in context they lead to, and when users click on such a button, the Control Panel requests of the HyperContext server the appropriate interpretation. In addition, a recommended link identified by an HCTr button needs to know the path the user should follow to reach a contextually relevant interpretation, so that the appropriate link in intervening interpretations can also be recommended. This is achieved through the use of a parameter to the Java applet which contains the path to the contextually relevant interpretation. The interpretation to visit next is always at the head of the path, implying that as the path is traversed it is shortened, until eventually, when the user reaches the contextually relevant interpretation, the path is empty.

7.6 The HyperContext server

The HyperContext server is implemented as a stand-alone Java application, listening on port 4848. It receives messages from HyperContext clients through a duplex channel. A conversation typically includes the conversation type (to request an interpretation or a document's contexts, to conduct a search, or to request recommended links) and it passes the results back to the client on the same channel. The client periodically tests the connection status while it waits for a response from the server. If the connection drops,

the client resubmits the request, eventually reporting back to the user through the HyperContext Status Window if the request cannot be satisfied after a number of attempts or if the server has crashed.

Each document referred to by the HyperContext hypertext has a number of files associated with it, namely a profile file, an interpretations file and a links file. The profile file contains details about the associated document's name, URL, file type (HTML or TEXT), and access method (HTTP). In addition, labels are bound to specific anchor text regions in the document. Anchor text is identified through the use of <HCT> tags, a tag name which is not recognised by Web browsers (and which is consequently ignored by the browser), but which is used to attach a Java applet providing an HCT or HCTr button if the link is active in a particular interpretation. Embedding <HCT> tags into the HTML document is not an ideal solution, because it requires modification of the source document, but it is a sufficient solution for demonstration purposes. The collection of profile files form the HyperContext Object Layer for this hypertext.

A document's interpretations file contains all the interpretations of the document. An interpretation is a vector of weighted labels named by the combination of the context giving rise to the interpretation and the document's URL. The external representation of a document's interpretation in the prototype is an inverted index file. This allows the interpretation file to be rapidly searched, and, through SWISH-E, the labels and label weights can be extracted from the interpretations file for a given interpretation, so that internally an interpretation can be represented as a vector of weighted labels (see the discussion on the inter-operability of different HyperContext implementations in Chapter 4.3). The links file contains, for each interpretation of a document, the links which are active in an interpretation and the destination of each link. The interpretations and links files collectively form the HyperContext Structure Layer.

7.6.1 Interpreting a document in context

When the server receives a request to interpret a document in context from a HyperContext client, it will receive from the client the URL of the document to interpret and the context within which to interpret the document. The client will wait for the server to provide the interpretation (as a vector of label-weight pairs) and the URL of the temporary file the server creates to contain the interpreted document. The HyperContext client will then generate an HTTP request for the file through the Web client.

```
HCT Server running...
Server invoked...
INCOMING is GETURL
Server received GETURL
Server received http://www.w3.org/Graphics/Overview.html
Server received bottom
Server received
Server received getitbutton
http://www.w3.org/Graphics/Overview.html
setInt: created all the files
setInt: extracted interpretation
.
setInt: read in the profile
.
Read in the children
30 children loaded
.
Server sent http://zeus.cs.um.edu.mt/~cstaff/HCT/java_progs/923373887.html
Server sent bye
```

Figure 7.7: A typical conversation between a HyperContext client and server

Figure 7.7 contains an edited conversation, viewed from the server side, of a client's request for a document in context. Upon receiving a request to interpret a document, the server ensures that the named document exists and has an interpretation for the specified context (otherwise it will either generate an error or provide the interpretation for the document in the context bottom, respectively). If the document exists, the appropriate interpretation is extracted from the interpretations file and is sent back to the client. The links are extracted from the links file and the file containing the document is retrieved and processed to convert the appropriate <HCT> tags into parameterised Java applets representing link sources. The modified file is saved as a temporary file on the Web server, the URL for which is also sent to the waiting client.

Figure 7.8 contains an HTML document fragment before (a) and after (b) the document is interpreted. Interpreting a document involves incorporating links into the document prior to displaying it to the user. In the prototype, this is achieved by inserting parameterised Java applets into the correct locations (shown in blue text in figure 7.8b). The Java applets will display the link and provide the link with a destination (the value of the parameter name theURL) and a context (theContext).

```
...
<h1>Graphics on the Web</h1>
<div class="main">
<h2>What is W3C Doing?</h2>
<p>Please see the <HCT2><font color=blue><u>W3C Graphics Activity statement</u></font> to
find out what W3C is currently doing and has already done in the area of Web
Graphics, including <HCT3><font color=blue><u>SVG</u></font>, <HCT4><font color=blue><u>WebCGM
Profile</u></font>, and <HCT5><font color=blue><u>PNG</u></font>.</p>
...
```

(a)

```
...
<h1>Graphics on the Web</h1>
<div class="main">
<h2>What is W3C Doing?</h2>
<p>Please see the <applet codebase="classes" code="second.class" name="second" width=30 height=25>
<param name="theURL" value="http://www.w3.org/Graphics/Activity.html">
<param name="theContext" value="http://www.w3.org/Graphics/Overview.html,W3C GRAPHICS ACTIVITY STATEMENT">
<param name="theRec" value="">
</applet>
<HCT2><font color=blue><u>W3C Graphics Activity statement</u></font> to
find out what W3C is currently doing and has already done in the area of Web
Graphics, ...
...
```

(b)

Figure 7.8: An HTML file fragment before (a) and after (b) it is interpreted

The parameter name theRec in the parameter list of the Java applet shown in figure 7.8b is used to trace the recommended path to a contextually relevant interpretation. If theRec is non-null, the associated link is recommended (through an HCTr button, as shown in figure 7.3). The client will have had to provide the previous value of theRec as part of its request to interpret a document.

7.6.2 Sending a document's contexts

A user can perform a context switch by requesting a list of a document's valid contexts by clicking on the Select Context button in the HyperContext Control Panel (figure 7.2). The server responds to this request by generating a list of all of a document's contexts extracted from the links file for the document, which contains entries for all contexts even if the corresponding interpretation has no out-links. The HyperContext client will display a floating window with all the contexts received from the server. The user can select a suitable context which causes the selected context to be loaded into the Context text field of the Control Panel (figure 7.2). When the user clicks the Get It button on the Control Panel, the client generates a request for the document to be re-interpreted.

7.6.3 Satisfying a search request

A search request can be generated in one of three ways. The user can generate a request for a TIR search or an IRC search by keying search terms into the search text field and

clicking on the Search button (figure 7.2), or the HyperContext client can generate a request for an AID search, if the AID radio button is "on". In each case, the client sends the server the search type and a list of terms (entered by the user in the case of a TIR or IRC search, or generated by the client in the case of an AID search). Upon receiving a search request, the server invokes SWISH-E to retrieve relevant interpretations.

The HyperContext site constructed for evaluation purposes is small (containing 170 Web documents which have been converted into 4,187 interpretations). Given that all documents and their interpretations are on the same host, we implemented a centralised, rather than a distributed, solution to satisfy a search request. The Traditional Information Retrieval search method requires a centralised index of all interpretations, and in the prototype the centralised index supports all of the search methods. However the results of the search are processed differently, depending on the search type.

Traditional Information Retrieval (TIR)

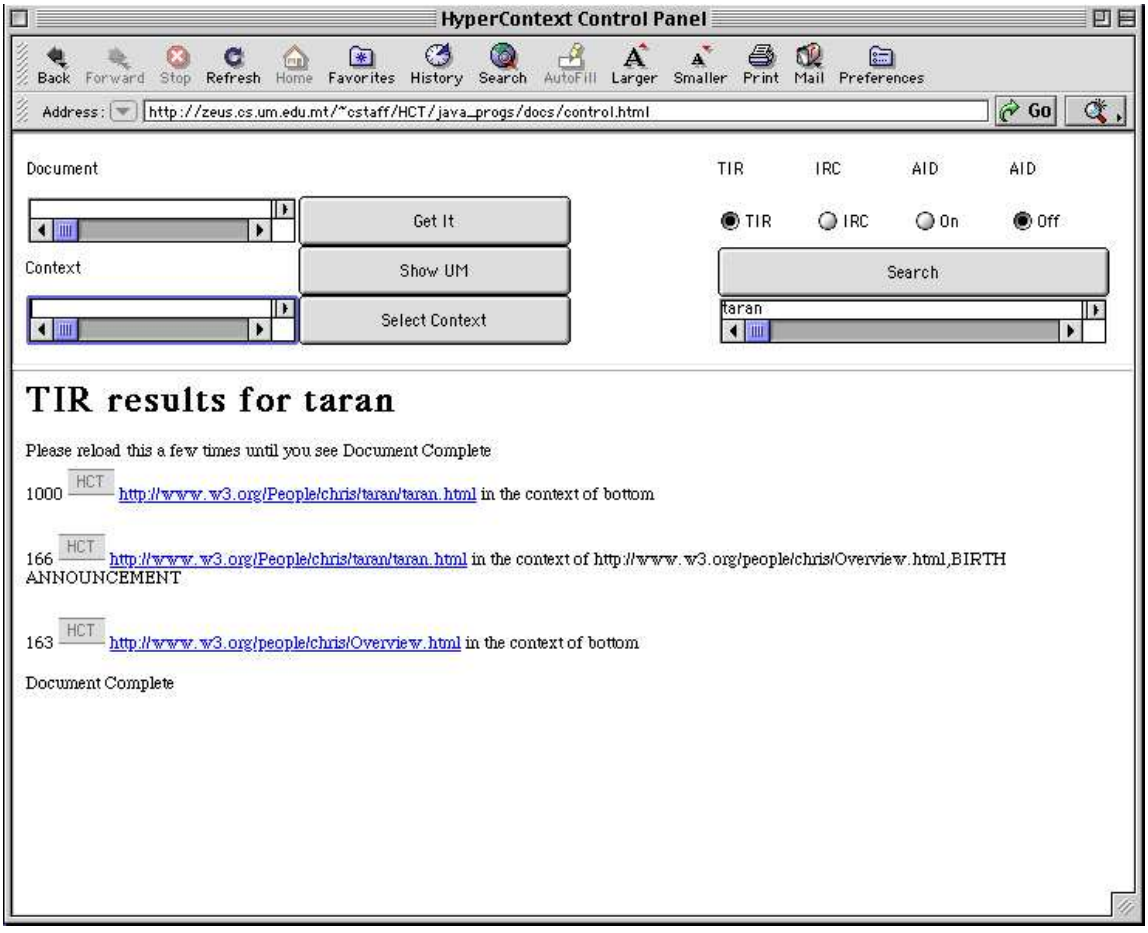

Figure 7.9: Example TIR results

If the user has selected TIR on the HyperContext Control Panel, keyed in some search terms, and clicked the Search button, the client will send the server the search type and search terms. The server then generates a query against the central index, creates a separate thread of control within which SWISH-E runs as a C process, and waits for SWISH-E to terminate. As SWISH-E produces output, the data stream is copied into a temporary file which is processed once SWISH-E has terminated. SWISH-E's output is modified to insert HyperContext HCT button before each relevant interpretation (figure 7.9). The interpretations are ranked in order of relevance (the score is shown on the left-hand side of a relevant interpretation). The URL is highlighted in blue, and the context of the relevant interpretation is also displayed. When a user follows a link to a relevant interpretation, she is taken to the appropriate interpretation of the document. Following a link indicates the start of a new context session. The salient interpretation (Chapters 5.8.1 and 6.7.1) is derived using formula 5.6.

Information Retrieval-in-Context (IRC)

The HyperContext server also uses SWISH-E to provide search results which can then be processed to inform the user of contextually relevant information. As in TIR, the server constructs a query from the user-selected search terms sent to it by the client. The query is submitted to SWISH-E, which searches through the central index for all relevant interpretations. IRC requires that the user nominates an interpretation which will act as the root of the context sphere (Chapter 6.6.2). By default, this is the interpretation the user is currently visiting. SWISH-E processes the IRC query as though it is a TIR query. The results of the search are copied to a temporary file which is then processed to identify all interpretations that are contextually relevant to the root interpretation of the context sphere. The contextually relevant interpretations are presented to the user as shown in figure 7.10.

The user can be guided to a contextually relevant interpretation by clicking on the HCTr recommended link associated with the interpretation. The user is also informed of the path that will be followed by clicking on the HCTr button. By following the link, the user is first taken to the interpretation nominated as the context sphere's root, and the recommended link in each interpretation on the recommended path is indicated by an HCTr link (see figure 7.3).

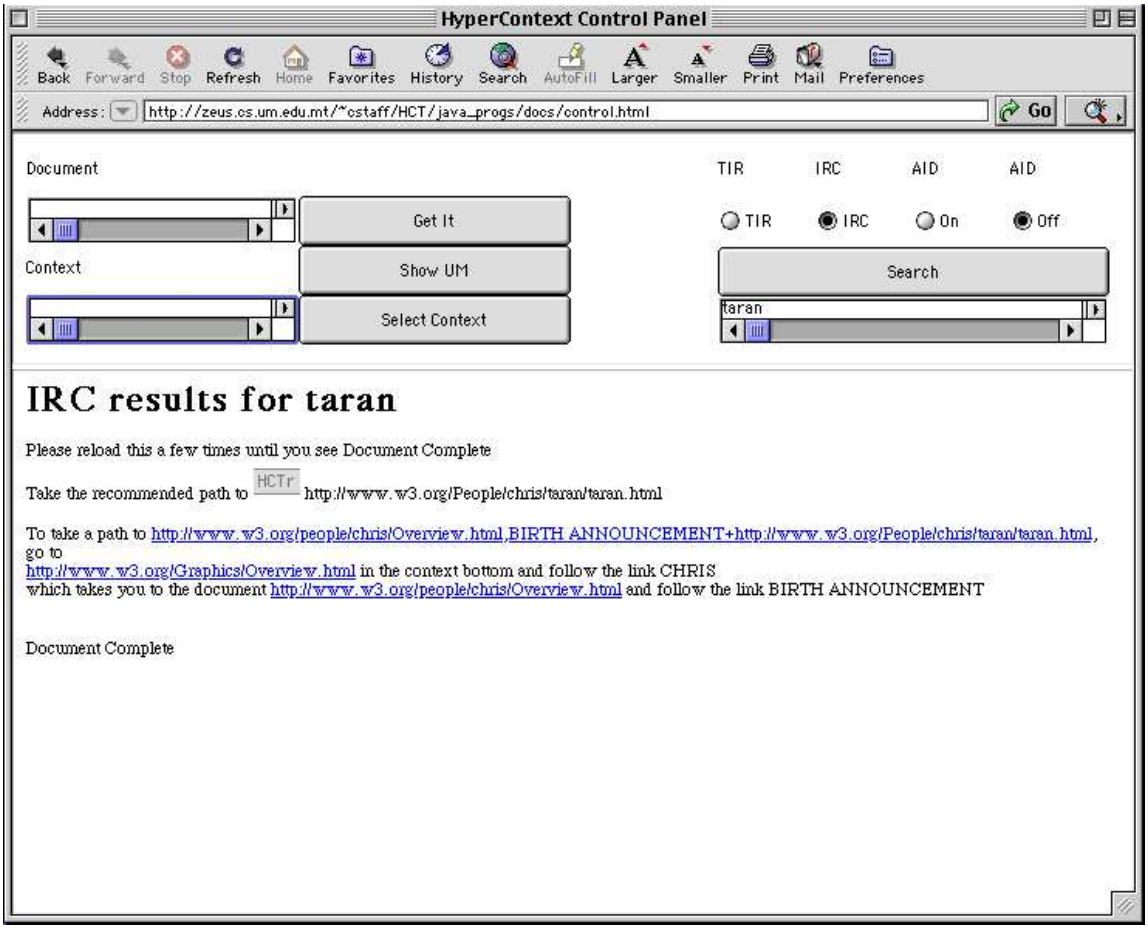

Figure 7.10: Example IRC results displayed in the Document Frame

An interpretation is contextually relevant to the root interpretation of the context sphere if the interpretation is a descendant of the root interpretation and is relevant to the user query. In the prototype, context resolution has been implemented as a linear, rather than a parallel, solution. Consequently, to ensure the user receives some response within a reasonable time, the solution is time bounded. A context path between the root interpretation and the relevant interpretation must be identified within a fixed number of seconds, otherwise the context expansion is aborted. Context expansion is performed in a bottom-up fashion. Starting from a relevant interpretation's parent (identified from its context), generations of ancestors are investigated in turn until the root interpretation or bottom is encountered, or until the time limit is reached.

When a user follows a recommended link the context session is extended. The salient interpretations of successively accessed interpretations on the recommended path are calculated using formula 5.4. The user, however, is free to follow other links which have not been recommended.

Adaptive Information Discovery (AID)

The prototype HyperContext server makes no distinction between an AID search and a TIR search. In the prototype, an AID query is treated as a TIR query. The search results are written to a temporary file on the server just as if the client had requested a TIR search, and the name of the file is returned to the client. However, rather than the HyperContext client generating an HTTP request through the Web client to display the search results, the HyperContext client instead requests that the HyperContext server sends it the temporary file. The details of the relevant interpretations (without distinction between contextually relevant and superficially relevant interpretations) are processed and loaded into AID's "See Also" References Floating window (see figure 7.4 and Sections 7.4.3 and 7.7.3). Of course, in AID search the query used will have been generated from the user model representing the user's short-term interests, rather than from user-supplied search terms.

If the user chooses to access an AID recommended interpretation, the context session is initialised, with the accessed interpretation acting as the root of the new context session. The salient interpretation, however, is abstracted using formula 5.6. In an enhanced version of the prototype, the user would be able to create a link to an accessed recommended interpretation and include the new interpretation within the current context session, and AID would recommend links and paths to contextually relevant interpretations.

7.7 The HyperContext client

The HyperContext client is responsible for providing the user with an interface to HyperContext; managing the context session and the user model; generating AID search requests; maintaining the list of "See Also" recommendations; and communicating user requests to the HyperContext server. User requests include requesting a document interpretation either by following a link or by supplying a document URL and context and clicking on the Get It button; changing the context of the current document; and performing a TIR or IRC search. The client activities involving communicating user requests to the server have been referred to during the description of the HyperContext server (Section 7.6). In this section, the description of the HyperContext client revolves mainly around the management of the context session and the user model, and the automatic generation of queries for Adaptive Information Discovery.

7.7.1 The context session

The context session is initialised when a user hyperleaps to an interpretation of a document. A new context session is started when the user clicks on the Get It button in the Control Panel; follows a link to an interpretation in the TIR Search results; or follows a "See Also" link recommended by AID.

The context session is a linked list of accessed interpretations. Each element of the list contains the accessed document's URL, the document's context, and vectors representing I_{sel} and I_{ave}, the selected and average interpretations of the document (see Chapters 5.8.1 and 6.7.1). As long as the user continues to access interpretations by following links, the context session is extended. The context session always represents a path, so if a user returns to an interpretation already accessed earlier in the same context session, then all interpretations which had subsequently been accessed from that interpretation are removed from the context session.

7.7.2 The short-term user model

The short-term user model is initialised and modified whenever the context session is initialised and modified. The user model is implemented as a linked list of salient interpretations. If the first interpretation in the context session is accessed by clicking through from the results of a TIR search, or from AID's "See Also" links, then the salient interpretation is derived using formula 5.6, otherwise formula 5.4 is used. Each salient interpretation is weighted according to the confidence scale discussed in Chapters 5.8.2 and 6.7.2. Although the user model is maintained throughout a context session, it is used only if the user has activated Adaptive Information Discovery.

7.7.3 Adaptive Information Discovery: query formulation and link recommendation

Adaptive Information Discovery can be activated and de-activated by the user at any point during a context session. If AID is active, then each time a user accesses an interpretation, AID will derive a query from the short-term user model and search for relevant information. Any relevant interpretations found will be presented to the user as "See Also" links, regardless of whether the interpretation is contextually or superficially relevant. If a user decides to access a recommended interpretation, the user is taken directly to the selected interpretation, the context session and user model are initialised, and the salient interpretation is derived using formula 5.6. In the prototype, AID will

recommend "See Also" links from the first interpretation accessed in a context session, rather than waiting for a later access.

In the prototype, an AID query is generated by combining the salient interpretations in the user model using formula 5.5, and extracting at least seven highest scoring terms (more than seven terms are used if the score of the seventh term is also held by other terms). These terms are ANDed in the server-side query submitted to SWISH-E. If there are terms in the user model which have negative scores, then at least three terms with the lowest scores are extracted and NOTed in the query. The HyperContext client then invokes the HyperContext server to perform a TIR search against the central index. However, rather than presenting the TIR results through HyperContext's document frame (as in figure 7.9), the HyperContext client reads in the results and processes them prior to presenting them through AID's "See Also" References window (figure 7.4).

It is possible that more than one interpretation of the same document is relevant to an AID query. In this case, only the highest scoring, or most relevant, interpretation is displayed to the user. Additionally, once multiple document references have been removed, only the five most relevant interpretations are recommended to the user.

7.8 A non-adaptive version of HyperContext

The experiments discussed in Chapter 9 require that the performance of adaptive HyperContext is compared against a benchmark provided by a non-adaptive hypertext. The HyperContext client has been modified to provide such a control hypertext.

HyperContext can be converted into a non-adaptive hypertext by replacing the Control Panel with a version that does not provide an interface to the Information Retrieval-in-Context and Adaptive Information Discovery search mechanisms. In addition, all document requests are satisfied in the context bottom, even if the user requests a document by following a link.

The user still has access to the Traditional Information Retrieval search mechanism, but only documents interpreted in the context bottom are recommended to the user.

By accessing all documents in the context bottom, the non-adaptive HyperContext client will always show each user precisely the same interpretation of a document, regardless which link is used to access the document. This is precisely the experience shared by users of a non-adaptive hypertext, such as the World Wide Web. Consequently, the

modifications to the HyperContext Control Panel suitably provide a non-adaptive control version of the adaptive hypertext used to obtain experimental results.

7.9 Summary

We have presented the prototype of a HyperContext hypertext which is implemented using a client-server architecture. The HyperContext server is a stand-alone Java application, and the client is a Java applet embedded within a Web browser frame.

The prototype is a partial implementation of the HyperContext framework, which includes the adaptive features of the framework, but which excludes the adaptable features.