Chapter 4

Publishing Integrated Hypertexts in the Web

Although the investigations reported in Chapter 3 uncovered some evidence of Associative Writing in the Web, this chapter starts by explaining how the hypertext model of the Web necessarily limits the expressive capabilities of the writer of such hypertexts, and ultimately the extent to which new contributions can be integrated with existing content, in comparison to the hypertext vision and the vision of Associative Writing introduced in Chapter 2. These limitations lead to two further (technical) challenges:

1. Improving support for *navigation and hyperstructures* — the Web's model of the basic building blocks of integrated hypertexts is restrictive; only uni-directional, binary links can be created, which must be embedded in the documents 'owned' by the writer. The challenge is therefore to enrich the Web's hypertext model to better support the publication of integrated hypertexts.
2. Maintaining *link integrity* — the chaotic nature of the Web (documents are edited, moved, or deleted in an ad-hoc manner) frequently causes links to 'break' (since links are embedded in documents and not maintained separately). The challenge therefore is to try and prevent integrated hypertexts from becoming permanently disconnected from the existing context on which they build.

Various approaches to these challenges using open hypertext and link management strategies are discussed.

4.1 Linking in the Web

The success of the World-Wide Web lies in the simplicity of its approach (Fensel and Musen, 2001). In order to introduce the implications of this 'simplistic' approach for

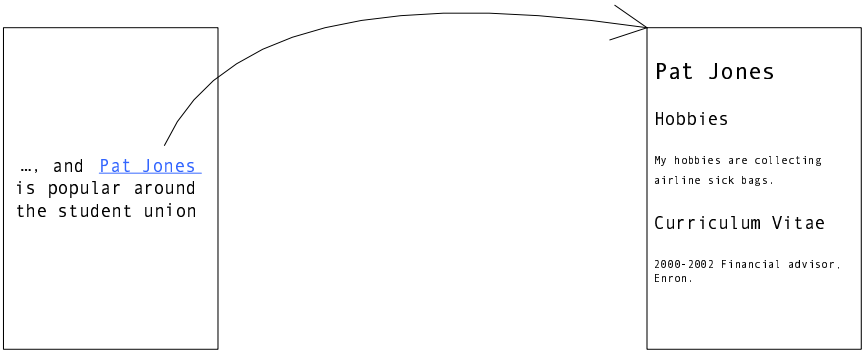
Associative Writing, this section first briefly outlines how links are encoded in Web documents using the HTML standard (W3C, 1999a).

HTML links have two ends (anchors), and a direction. A link starts at a "source" anchor in a HTML document and points to a "destination" anchor, which may be any Web resource, such as another HTML document, a specific (named) element within an HTML document, an image, a video clip, a sound file, or a program. The default behaviour associated with an HTML link is the retrieval of another Web resource. This behaviour is commonly and implicitly obtained by clicking the link in a Web browser.

Figure 4.1 demonstrates how two HTML documents could be linked in the Web. In Figure 4.1a, the link is encoded using the `A` element, with the `HREF` attribute specifying the destination of the link. In Figure 4.1b, the destination anchor is a specific element of the destination document. The destination anchor must be encoded with an anchor name by its author (the `NAME` attribute). The URL addressing this anchor must then include the name as its fragment identifier (the portion of the URL starting with the `#` character). When this link is activated, the destination document is retrieved and automatically scrolled to the relevant position by the browser. The proposed HTML+ standard (Raggett, 1994) described an additional mechanism for encoding the semantic type of the relationship between source and destination anchors using `rel` and `rev` attributes. Although the current HTML standard contains a subset of the HTML+ link type specification, none of today's popular Web browsers take these specifications into account.

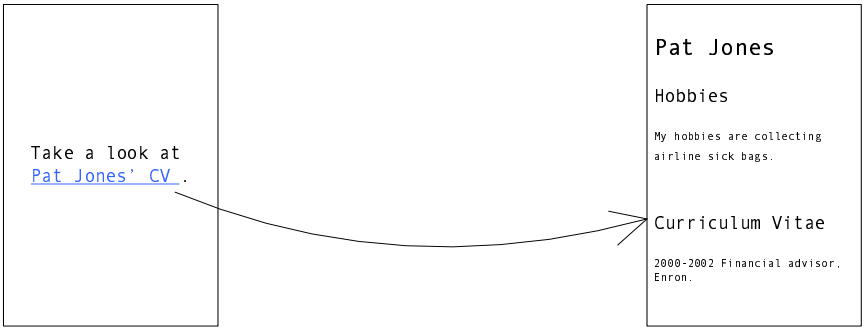
The pervasiveness of this implementation has led to the HTML link becoming the *de facto* standard definition of the hypertext link (Verbyla, 1999). However, this hypertext model is restricted in comparison with the vision of hypertext and its subsequent implementations. Researchers have often lamented that the Web does not support many of hypertext's rich structuring, navigation and annotation features (Vitali and Bieber, 1999; Bieber et al., 1997; Pam, 1995). It has even been argued that the Web is not a hypertext system at all, and has more in common with a distributed file system (Nürnberg and Ashman, 1999). Common lamentations include the lack of support for *hyperstructures* (links are embedded in HTML pages and therefore cannot easily be processed), lack of support for *navigation* (only uni-directional, binary links can be created), and the problem of *link integrity* (when Web pages are edited, deleted, or moved, links which refer to them may no longer valid). We focus first on hyperstructure and navigation issues; the issues surrounding link integrity are introduced later in Section 4.4. The specific issues surrounding the lack of support for hyperstructures and navigation afforded by the Web hypertext model can be iterated in detail:

Preservation of ownership Chapter 2 noted that the Web preserves the notion of "ownership" — pages and sites are typically under the control or ownership of a single




```
..., and <A href="students/patjones62.html">Pat
Jones</A> is popular around the student union
```

(a) Simple HTML link between two Web pages.




```
Take a look at <A
href="students/patjones62.html#cv">
Pat Jones' CV </A>
```

```
<A name="cv">Curriculum Vitae</A>
```

(b) HTML link between two Web pages using fragment identifier.

FIGURE 4.1: Creating Web links using HTML.

designer, design team, or organisation — thereby separating the roles of writer and reader (in contrast to many hypertext systems in which any reader is simultaneously a writer). With knowledge and "ownership" of Web server space, any user can create a new hypertext, and link to any other document (without having to own it). Specific content in documents not owned by the writer can only be linked to (using fragment identifiers) if the owner of that content has provided (in advance) the facilities to do so (using the `NAME` attribute). As a consequence of embedding hypertext links in HTML, only the "owner" of a document can create links leading from that document.

Links must be embedded in HTML There is no inherent mechanism in the Web infrastructure for storing links externally from the source HTML document (or other link supporting media). *Link fossilisation* occurs because links cannot be changed without revising the document (Carr et al., 1995). In contrast, the separation of links and content was fundamental to open hypertext systems.

Only HTML documents can be link sources Originally, only HTML documents could contain embedded URLs. More recently other (proprietary) formats, such as Adobe's Portable Document Format (PDF), and Microsoft Office, have included support for embedded hypertext links. Other media, such as images and video, cannot contain embedded hypertext links without browser/viewer alteration. *Dead ends* occur when the user reaches a media that cannot contain hypertext links ('there is nowhere to go from here'). In contrast, open hypertext systems demonstrated how hypertext functionality could be applied to media which did not inherently support it. Without a link service, Web users can follow links from HTML documents into dead-end media such as spreadsheets, CAD documents or text; with a link service they can also follow links out of these media again (Carr et al., 1995).

Pre-determined source and destination The source and destination of Web links must be pre-determined, and manually specified by the writer. In contrast, some hypertext systems allowed the source and destination of a link to be determined dynamically. For example, the source of a *generic link* in Microcosm was determined dynamically (when the user requests a document), as was the destination of a *computed link* (when the user follows the link).

Links are uni-directional Web links lack *bivisibility* and *bifollowability* (Pam, 1995): they can only be followed in one direction and can only be seen from the originating end (the source). In contrast, many hypertext systems allowed links to be both bivisible and bifollowable, for example DeVise Hypermedia introduced some interesting notions about link directionality.

Links are binary Only one link with a single destination is permitted from any point. In contrast, many hypertext systems allowed n-ary links, which connect one-to-many, or many-to-many anchors.

4.1.1 Implications for Associative Writing

Although the investigations reported in Chapter 3 uncovered evidence of Associative Writing in the Web, the limited hypertext model of the Web restricts the expressive capabilities of the writer of such integrated hypertexts, and ultimately the extent to which new contributions can be effectively integrated with existing content.

Figure 4.2 attempts to illustrate this scenario by contrasting an integrated hypertext published in the Web with one published in the docuverse of a hypertext system which implements the hypertext vision. In both cases, solid arrows represent functional links

providing local coherence to the hypertext, and dotted arrows represent associative links integrating the writer's new contribution with the global context on which it builds.

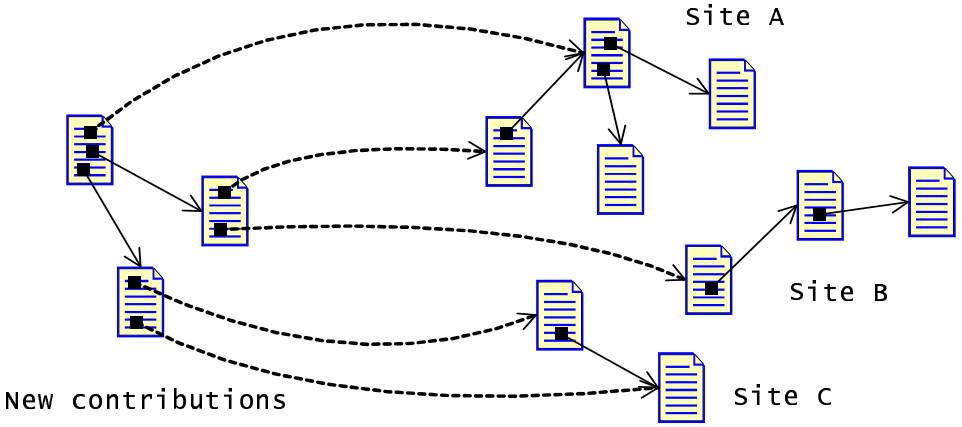
In the Web scenario, all associative links must originate from the writer's hypertext (the only document 'owned' by the writer), and can target at most one other Web page. Destination anchors target entire pages — opportunities for linking to specific content in destination pages (using fragment identifiers) have not been provided by the "owner" of the destination page. Links are also uni-directional; writers and readers can navigate from the new contribution to existing content, but the new contribution is not visible from existing content.

In the hypertext scenario, links are bi-directional, so readers can discover the new contribution from older work. Links can also have more than one destination, and destination anchors can be arbitrarily located in the destination document. Importantly, there is no notion of ownership, so the global context of the new contributions can extend to providing new links between existing content in the docuverse, perhaps to demonstrate conflicting viewpoints, or to juxtapose existing ideas in previously unthought-of ways — note also that semantic types have also been attached to each link to explicitly state its purpose.

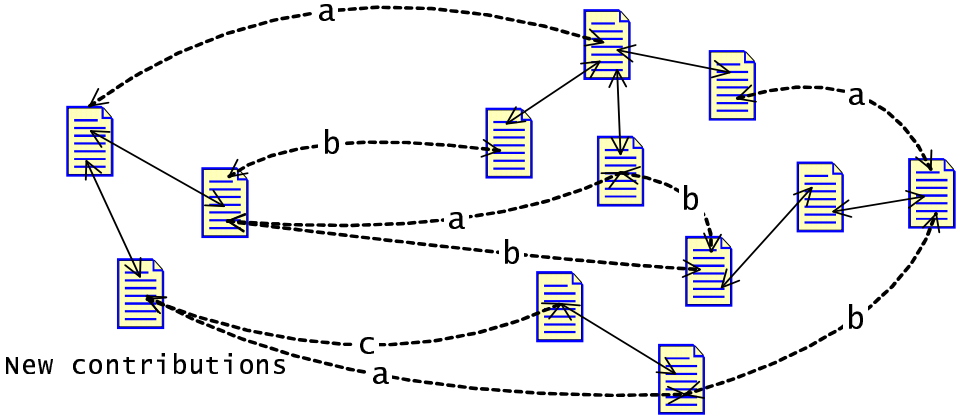
4.2 Augmenting the Web Hypertext Model

Far from restricting its success, the simplicity of the Web has in fact provided a springboard for a multitude of more sophisticated tools that add specific functionality to the Web model. Search engines index web pages and provide starting points for information seekers, and Web servers have been extended to provide for server side processing, enabling HTML to be used as a delivery mechanism for non-hypertext applications such as online shopping and banking. Similarly, much research has focused on extending the hypertext capabilities of the Web, by interfacing a standard Web browser with a powerful hypertext engine. In effect, an entire 'hypertext for the Web' technology has sprung up. As a simple example, "reverse links" can be gathered by a background process operating on the uni-directional linked Web, a technique used by Citescape (Moulthrop and Kaplan, 1995) (Figure 4.3) and some search engines, such as *Google* (Figure 4.4).

This section focuses on the augmentation of the Web hypertext model through integration with open hypermedia technology. Although the Web was developed largely independently of the research in open hypermedia systems at the time, its popularity along with the problems inherent in its design has subsequently motivated open hypertext researchers to 'retro-fix' their systems with it, enhancing the functionality of the Web via the services of a open hypermedia system (Anderson, 1997). In turn, the Web provided a greater distribution for open hypermedia systems, and standards to increase the interoperability and ease-of-use of such systems. (Anderson, 1997) describes five



(a) Web Scenario



(b) Hypertext Scenario

FIGURE 4.2: Comparing hypertext model of the Web with typical features of systems which implement the hypertext vision.

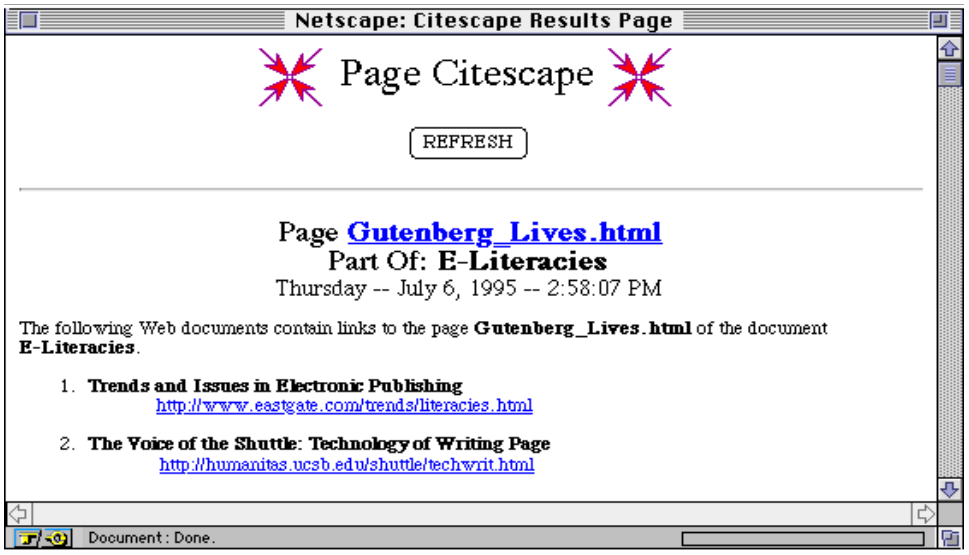


FIGURE 4.3: Reverse links computed by Citescape (Moulthrop and Kaplan, 1995)

FIGURE 4.4: Reverse links computed by *Google*

different techniques that have been used to integrate open hypermedia systems with the Web, briefly described here as a prelude to a more detailed examination of some of these systems:

OHS data to Web data The simplest integration technique — tools are produced which translate links and content within an OHS into HTML. Authors can create content within an OHS and later publish it in the Web. (Hall et al., 1996) demonstrated this approach in converting Microcosm applications into a set of linked HTML documents.

Web client as OHS client A Web client (browser) is modified to integrate with an OHS. (Hall et al., 1996) and (Anderson et al., 1994) reported such integrations using Microcosm and Chimera. The work to integrate DHM with the Web (Grønbæk et al., 1997), culminating in the WebVise system (Grønbæk et al., 1999) was also an example of this type of integration.

Web server as OHS client A Web server is extended to be a client of an OHS, enabling users to access the functionality of the OHS from a standard (unmodified) Web browser. The Web server is modified to make calls to the OHS in response to user requests for certain URLs, generating HTML which manifests the presence of the OHS. The Distributed Link Service (Carr et al., 1995) was an example of this approach, in using proxy servers to access distributed linkbases and "compile" link information into requested HTML documents on the fly. Chimera has also been integrated with the Web using this approach (Anderson, 1997).

OHS server as Web server The modification of an OHS server to masquerade as a Web server. The most notable work in this area was the Hyper-G project (Andrews et al., 1995b,a). Normally, Hyper-G structures are best viewed with the proprietary Harmony client, but Hyper-G servers can also receive requests from standard Web clients and serve translated structures as HTML.

Web protocols within an OHS This approach to integration uses Web protocols within an OHS to help leverage their distribution and standardisation benefits. HyperDisco (Wiil and Leggett, 1996) demonstrated the ability to access workspaces

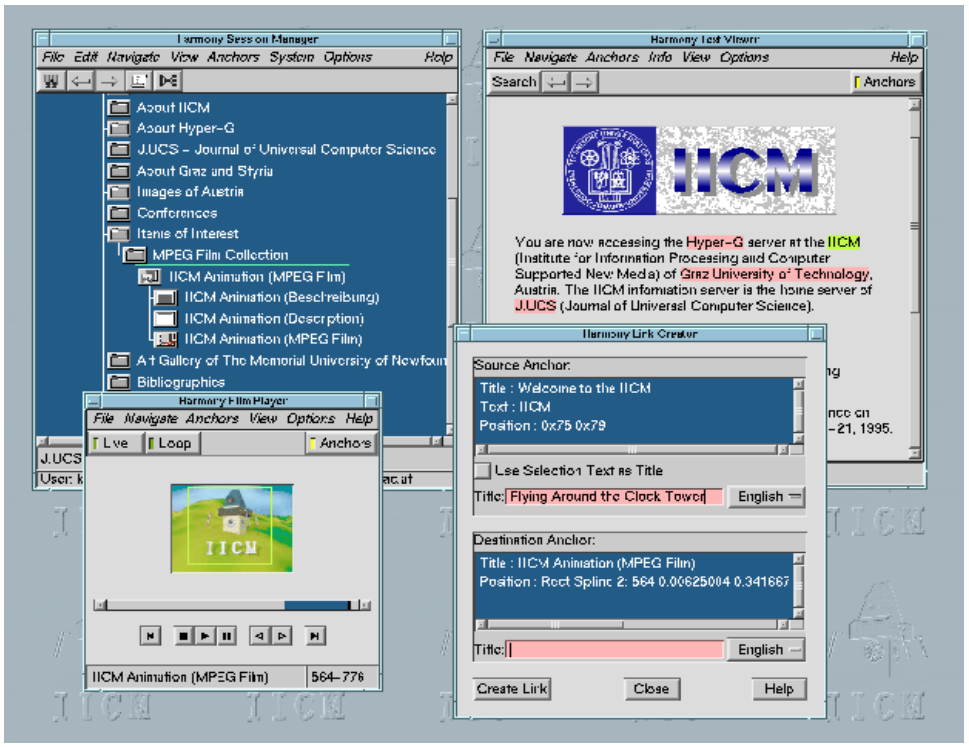


FIGURE 4.5: Interactive link creation in Hyper-G (Andrews et al., 1995a)

distributed across the Internet using this approach (Wiil and Leggett, 1997). Chimera also demonstrated a similar integration approach (Anderson, 1997).

4.2.1 Hyper-G

Hyper-G (Andrews et al., 1995b,a), later commercialised as HyperWave (Maurer, 1995), was an open hypertext system inspired by early developments of the Web. Documents in Hyper-G could be grouped into collections, which could be shared and combined arbitrarily. Links connected a source anchor in one document to either a destination anchor within another document, an entire document, or a collection. These externally stored links were bi-directional, and were updated and deleted automatically when documents were moved or deleted (elaborated further in Section 4.4). Every document and collection was automatically indexed upon insertion into the Hyper-G database to provide fully integrated search facilities.

Information in Hyper-G was usually structured first into overlapping collections and sub-collections, with links used for cross-references orthogonal to the collection structure. Figure 4.5 shows a link being created between a source anchor in a Hyper-G text document and a destination anchor in a video using the Harmony browser.

Hyper-G servers responded to document requests from Web browsers with a HTML representation of the requested information, preserving as much as possible of the rich

hypertext functionality of Hyper-G in the process. For example, when requested by a Web browser, each level of a collection hierarchy was converted to an HTML document containing a menu of links to its members. Hyper-G text documents were transformed on the fly into HTML documents, with any links they might have encoded into the document. Additional Hyper-G functionality such as searching was implemented via HTML forms. However, the advanced functionality of Hyper-G could only be accessed through the proprietary Harmony clients, which failed to achieve widespread acceptance.

4.2.2 Distributed Link Service

The Distributed Link Service (DLS) (Carr et al., 1995; Hall et al., 1995), later embodied commercially by WebCosm and Active Navigation's Portal Maximiser[1], applied the functionality of Microcosm (Fountain et al., 1990) to the Web, using the Web infrastructure as a communication framework.

Early implementations of the DLS required the modification of a Web browser to include a menu which allowed the user to interact with the link server. In the same way that a normal Web browser connects to a remote Web server to access a document, the modified browser could connect to a link service and request a set of links to apply to a document. The modified browser sent the link server details of the document URL and the user's selection in the document, and received back an HTML page listing the available links from currently loaded linkbases. With this modified browser, users could utilise the hypertext power of Microcosm across the global information space of the Web, creating local and generic links across arbitrary documents whether those documents supported links or not. Users could also access the link service directly, through a Web page interface, to configure settings such as which linkbases should be active during browsing.

However, modifying a third-party Web browser had its difficulties: users had to install the modified browser, and as each new browser version was released, it had to be modified and tested. Therefore, the DLS was adapted to provide an *interface-less* interaction mode whereby a proxy service masqueraded as a Web server and received all document requests, augmenting the fetched documents behind-the-scenes with all possible links from the user-selected linkbases, and then delivering the linked document back to the browser. The use of a proxy also had a number of drawbacks: the querying of linkbases on each page request inevitably slowed down the browsing process, users had to set up the browser to use the proxy, and the proxy service could only embed links in HTML documents (or other formats supporting hypertext links). These drawbacks led to the creation of the AgentDLS, in which links were displayed alongside the primary Web page rather than embedded, leaving normal browsing unaffected and boosting performance and usability considerably (Carr et al., 1998).

[1] `http://www.activenavigation.com/PortalMax/default.htm`

4.2.2.1 QuIC

A consequence of facilitating "generic" linking facilities for the Web was that links could be applied to documents "out of context" — the Web unifies without boundaries documents from many different domains describing many different topics, so generic links could potentially be applied to documents outside the intended domain or topic. The Distributed Link Service placed the onus on the reader to choose the linkbases that best matched the context of the pages that were being browsed. In contrast, Queries In Context (QuIC) attempted to automatically determine the most appropriate linkbases to use (El-Beltagy et al., 2001). Each generic link had an associated context metric which were only applied to pages with a matching context.

4.2.3 WebVise

WebVise (Grønbæk et al., 1999), building on DeVise Hypermedia (Grønbæk and Trigg, 1994) and DHM/WWW (Grønbæk et al., 1997), allowed writers to create hyperstructures such as links, annotations, and guided tours in the Web. These elements were collected into hypermedia "contexts" (layers of related structure) which were superimposed across Web documents by the system.

In order to create these structures, users had to install a WebVise client which integrated with the Internet Explorer browser and Microsoft Office applications (in contrast to the Distributed Link Service, this integration was facilitated by the extensible component model of the applications rather than by making changes to source code). The client installation also provided a separate interface for creating contexts and guided tours. Internet Explorer was extended to provide a context menu for creating externally-stored structures such as *anchored links* (links between specific endpoints), keyword-based *global links* (a generalised many-to-many variant of Microcosm's generic link), and annotations (small notes) across arbitrary Web pages. In Microsoft Office applications, WebVise integration was achieved through custom toolbars and menus. Anchored and global links could be created (and anchored) in Microsoft Word documents. In Microsoft Excel only anchored links were supported (anchors could locate ranges of cells in a worksheet). WebViseLT (Hansen et al., 1999) extended WebVise to incorporate a Link Types Editor, in which users could define a hierarchical set of link types.

The hypertext structures created with the WebVise client can be accessed in *any* Web browser (although the authoring of these structures is restricted to the Internet Explorer browser) via a proxy service interface. The WebVise proxy is similar in concept to that of the Distributed Link Service (and hence suffers from similar problems), but supports the richer set of structures provided by the WebVise service. On each document request, the WebVise proxy checks the link server for potential external structures that can be imposed upon the document. Anchored links and notes are compiled directly into

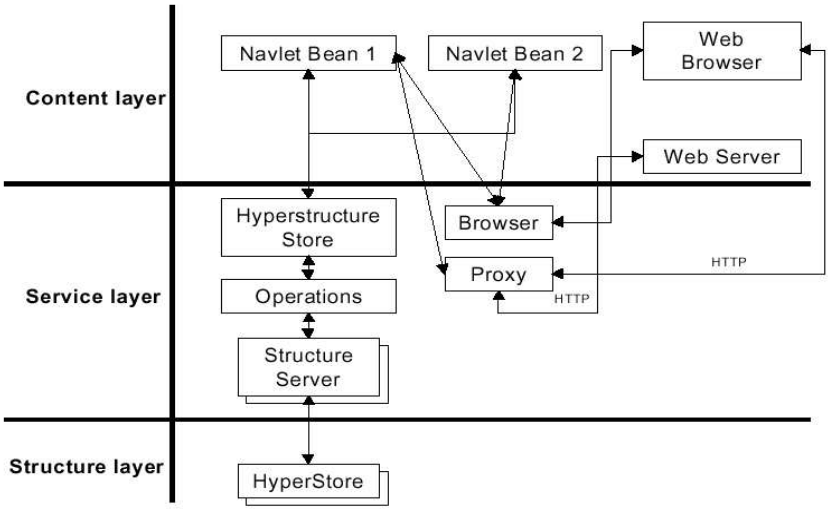


FIGURE 4.6: The Arakne framework (Bouvin, 1999).

the document before being delivered to the browser. Global links (normally requested through the client's context menu) have to be accessed through a form-based interface. The proxy does not support links to and from Word and Excel documents; these links can only be browsed if the WebVise client is installed. If a document is changed outside of the WebVise system, anchored links and notes can (possibly) be repaired by using a text search strategy on the content of the anchor (elaborated in Section 4.4).

4.2.4 The Arakne Framework

In modelling and unifying the common components of integrated open hypertext/Web systems up to that point, (Bouvin, 1999) introduced the Arakne framework, an environment which attempted to provide a tool for future conceptual and practical development in this area. The Arakne framework consists of a three layered model (Figure 4.6) aimed at providing Web augmentation tools with unified access to structure servers, proxy servers, and Web browsers.

Web augmentation tools (labelled as "navlet beans" in this Java-influenced model) are dependent on four core components of the Arakne framework:

1. *Operations*: models the communication with the Structure Server component.
2. *Hyperstructure Store*: interface between "navlets" and and Operations component.
3. *Browser*: models the user's Web browser, providing access to the browser's status (for example, the URL currently displayed, the structure of the current frameset, the selected text).
4. *Proxy*: models the modification and analysis of Web content, allowing "navlets" to directly access and modify the content of Web pages as they are retrieved.

The framework was successfully used to build a system which integrated the guided tour tool Ariadne (Jühne et al., 1998) and the link creation tool Navette (Bouvin, 1998) with the Internet Explorer Web browser and DeVise Hypermedia structure server.

4.3 XLink: Next Generation Web Linking

A problem with the "hypermedia for the Web" technologies described above is that Web users must actively seek them out in order to take advantage of the extended hypertext facilities they have to offer. Consequently ignorance, and perhaps a lack of understanding of the benefits, has limited the widespread adoption of these hypertext tools (Cailliau and Ashman, 1999). However, the hypertext functionality of the Web is changed at a fundamental level by the W3C's recent XML Linking (XLink) standard (W3C, 2001b). The standard has been influenced by hypertext systems research (DeRose, 1999), and hypermedia models such as Dexter (Halasz and Schwartz, 1994) and OHP (Davis et al., 1996), and hence embodies many advanced hypertext functions: the implications of the XLink standard are therefore that hypertext is a core function of the Web which must be universally supported, rather than marginalised to third-party applications.

Section 4.1 described how HTML links are defined using the closed semantics of the `A` element. XLink standardises a mechanism to indicate that any given element is a hypertext link, and to identify properties specific to that link, including its behaviour. Whereas Web links must be embedded in documents, XLink allows external as well as internal links, and describes a mechanism for documents to declare relevant linkbases. Whereas Web links are binary and uni-directional, XLink standardises a way of describing the traversal of bi-directional, multi-headed links. Whereas URL fragment identifiers allow HTML links to reference specific parts of a target document (provided that these parts have been named in advance by the document owner), XLink embodies a more general pointing syntax enabling any node, point or selection in a target HTML, SGML, or XML document to be located.

The XLink standard has already had an impact on hypertext research. An investigation of XLink as an export format for Chimera was reported by (Halsey and Anderson, 2000). (Christensen and Hansen, 2002) go a stage further, and consider the use of XLink as a linking mechanism for open hypermedia systems in general, demonstrating how the Open Hypermedia Interchange Format (OHIF) (Grønbæk et al., 2000) used by WebVise can be successfully mapped to an XLink linkbase. In the absence of Web browsers which fully support the XLink standard — the W3C's Amaya and the Mozilla browser currently provide only a limited XLink implementation and (Christensen and Hansen, 2002) resort to using stylesheet transformations (W3C, 1999c) to convert XLink linkbases to an HTML form that can be accessed by conventional browsers — (Martin and Ashman, 2002a,b; Vitali et al., 2002) demonstrate how support for XLink can be

provided using conventional Web browsers in conjunction with an HTTP proxy. This principle is similar to proxy-based implementations of the Distributed Link Service and WebVise, and inevitably suffers from the same weaknesses of the approach.

4.3.1 Linking in XLink

The XLink model distinguishes the following concepts:

Anchors The actual data being connected by the link (documents, nodes, selections).

Locators Specify an anchor location (for example, a URL) and its properties.

Arcs Connect locator pairs by roles to indicate the usefulness of traversal and define traversal semantics for the pair. An arc can be traversed at user demand or automatically, with the target of the arc specified to replace the source, appear in a new display window, or be embedded alongside the source anchor.

Links Aggregate the other constructs into a set of a given link type: either *simple* or *extended*. A link may include multiple locators and arcs (multi-ended links) with any pattern of traversal between the ends. A link not only connects locations, but describes them and the connections themselves, using machine-interpretable *roles* and human-readable *titles*.

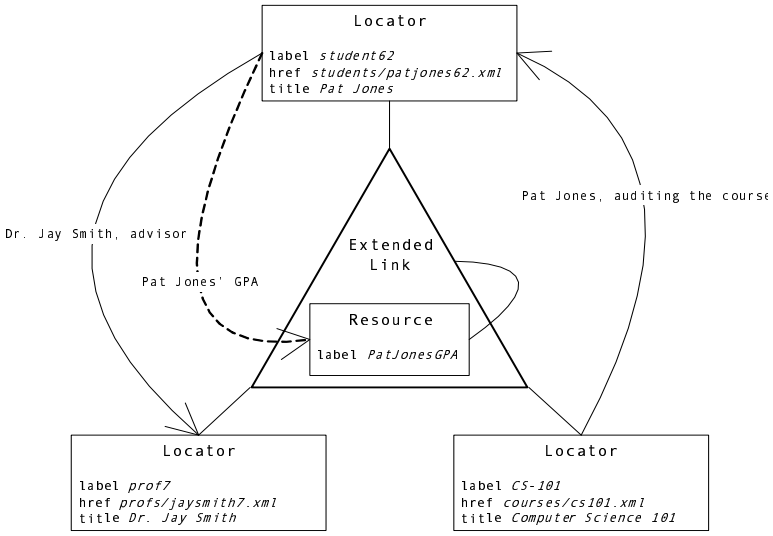
Figure 4.7 shows an example of an extended XLink with multiple endpoints (the remote resources titled *Pat Jones*, *Dr Jay Smith*, and *Computer Science 101*, and the local resource titled *PatJonesGPA*), which defines three traversal arcs between these resources:

1. The arc titled *Pat Jones' GPA* connects the remote resource *Pat Jones* to the local resource *PatJonesGPA*. This link is traversed on user request, and the target (*PatJonesGPA*) is displayed in a new window.

2. The arc titled *Pat Jones, auditing the course* connects the remote resource *Computer Science 101* to the remote resource *Pat Jones*, using the role *auditor*. This link is traversed on user request, and the target (*Pat Jones*) will replace the source (*Computer Science 101*) in the display.

3. The arc titled *Dr Jay Smith, adviser* connects the remote resource *Pat Jones* to the remote resource *Dr Jay Smith*, using the role *adviser* to describe this association. This link is traversed as soon as the source (*Pat Jones*) is displayed, and the target (*Dr Jay Smith*) is displayed in a new window.

A simple XLink is much like a Web link, connecting exactly two resources (one local and one remote) with a traversal arc going from the former to the latter (a simple



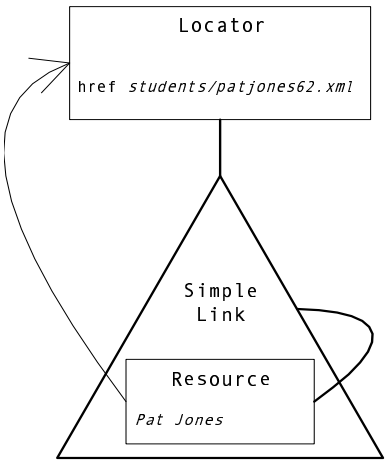

```
<extendedlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="extended">

    <!-- local and remote resources -->
    <person xlink:type="locator" xlink:href="students/patjones62.xml" xlink:label="student62"
            xlink:role="http://www.example.com/linkprops/student" xlink:title="Pat Jones" />
    <person xlink:type="locator" xlink:href="profs/jaysmith7.xml" xlink:label="prof7"
            xlink:role="http://www.example.com/linkprops/professor" xlink:title="Dr. Jay Smith" />
    <course xlink:type="locator" xlink:href="courses/cs101.xml" xlink:label="CS-101"
            xlink:title="Computer Science 101" />
    <gpa xlink:type="resource" xlink:label="PatJonesGPA">3.5</gpa>

    <!-- arc definitions -->
    <go xlink:type="arc" xlink:from="student62" xlink:to="PatJonesGPA" xlink:show="new"
        xlink:actuate="onRequest" xlink:title="Pat Jones' GPA" />
    <go xlink:type="arc" xlink:from="CS-101" xlink:arcrole="http://www.example.com/linkprops/auditor"
        xlink:to="student62" xlink:show="replace" xlink:actuate="onRequest"
        xlink:title="Pat Jones, auditing the course" />
    <go xlink:type="arc" xlink:from="student62" xlink:arcrole="http://www.example.com/linkprops/adviser"
        xlink:to="prof7" xlink:show="new" xlink:actuate="onLoad"
        xlink:title="Dr. Jay Smith, adviser" />

</extendedlink>
```

FIGURE 4.7: Describing an extended XLink (example taken from (DeRose, 1999)).

link is always an outbound link) and is intended to be a convenient shorthand for the equivalent extended XLink. Figure 4.8 illustrates a simple XLink: the local resource (anchor) titled *P. Jones* is connected to the remote resource titled *Pat Jones* by an arc which is traversed on user request, with the target replacing the source in the display — functionally equivalent to the HTML link illustrated in Figure 4.1a.

4.4 Link Integrity in the Web

Most hypertext systems assume that it is the responsibility of the system to ensure that link integrity is maintained (Davis, 1999), an assumption that is embodied in the Dexter Reference Model (Halasz and Schwartz, 1994). However, in large distributed hypertext




```
..., and <studentlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="students/patjones62.xml" xlink:show="replace" xlink:title="Pat Jones"
xlink:actuate="onRequest">P. Jones</studentlink> is popular around the student union.
```

FIGURE 4.8: Describing a simple XLink (example taken from (DeRose, 1999)).

systems such as the Web, it becomes difficult for the system to maintain this integrity; Berners-Lee has argued that letting go of the need for link consistency was a crucial design step that allowed the Web to scale (Berners-Lee, 1999).

The Web is prone to constant content change. Estimates of the average lifetime of a HTML document URL have been made at 75 days (Chankhunthod et al., 1996) and 50 days (Pitkow, 1998a). A study of AOL server logs determined that between five and eight percent of all requested links were "broken" (Pitkow, 1998b). Link decay occurs on the Web because HTML links refer to their destination anchors via a specific machine name and path name. Documents are changed or moved, often without preserving the old document. Documents may be deleted, or the domain name of a document may change. As (Jackson, 1997) succinctly points out, "there is no signed agreement that owners must make their information accessible to all for all time". Document addresses can therefore become incorrect some time after coming into use, resulting in hypertext links failing to correctly address the page they describe. The well-known "404 Error" on the Web is a classic case of invalidated references (Ashman, 2000). When a document which a link refers to is deleted, this is known as the *dangling link problem*; when the content of the document is changed (and internal fragment identifier references compromised) this is known as the "editing problem" (Davis, 1998).

(Ashman, 2000) describes a number of strategies for managing the integrity of links, which may be *preventative* (attempt to circumvent the the conditions leading to link failures), *corrective* (discover reference errors and maybe try and correct them), or *adaptive* (work out links on a just-in-time basis, so that broken links never have to be dealt with). Five of these strategies can be applied directly to the existing Web infrastructure:

1. *Do nothing*: Allow links to be wrong. Expectation is that users can perform their own corrective activity in searching for the correct target, and may also notify the link author of the error. This is the prevalent strategy on the Web.

2. *Forwarding mechanisms and gravestones*: If a document has been moved, HTTP forwarding pointers can redirect document requests to the new document location. Alternatively, the document can be replaced with a small note ("gravestone") that notifies the reader the document has been moved or deleted. This strategy is corrective: it does not repair links, but provides browsers with a means of rediscovering the correct target. The onus is on the owner of the moved or deleted document to provide the corrective measures.

3. *Relative references*: This strategy is preventative. When creating links, URLs are specified using relative addresses rather than absolute addresses. A relative URL specifies how to locate the end point of the link from the current position, and does not encode server location. This strategy allows whole collections of documents to be moved while preserving the validity of their references. However, this strategy only works within a set of documents on the same server referencing each other. External links to documents on other servers can still be broken, as can external links into the collection.

4. *Embedded links*: Embedded link destinations in HTML documents (see Figure 4.1b), although contrary to open hypertext philosophy, can survive document change (provided the encoded link targets themselves are not deleted), so the editing problem can be prevented.

5. *Dereferencing*: Link authors use a document alias rather than an absolute address to link to a destination document. The aliases are paired with up-to-date absolute addresses and kept in a registry. Resolving a link alias involves a lookup in this registry to find the corresponding absolute address. Whenever the absolute address of a document changes, the registry is updated to reflect the new location. This preventative strategy does not work well for deleted documents, and depends on the cooperation of link authors to use the document alias rather than its absolute address. The document owners must also ensure that the registry is kept up to date. There are a number of implementations of this strategy on the Web, including Uniform Resource Names (Berners-Lee, 1996), Persistent URLs (Online Computer Library Center, 1996), and Digital Object Identifiers (International DOI Foundation, 1998).

A number of other strategies have also been pursued in an attempt to maintain link integrity. Hyper-G used a corrective *notification* strategy — when a Hyper-G document changed, warnings were issued to all known applications that had links over the document so that corrective measures could be taken. This meant that to successfully preserve link integrity, all document updates in Hyper-G had to be made through

the system, which then propagated the document changes to all (proprietary) Hyper-G linkservers.

Alternatively, a *detect and correct* strategy allows documents to be changed independently of the hypertext system. In this strategy, software is run over a document collection or linkbase to detect (and possibly correct) broken links. For example, Microcosm recorded a time stamp on all links and documents. When these time stamps changed in relation to each other, the system could try and repair the link or warn the user (Davis, 1995). WebVise also supported certain repairs of links that break due to the editing problem (Grønbæk et al., 1999), by using "location specifications" (*LocSpecs*) to store redundant information about specific locations within a variety of different media, including text, images, and database records. For example, the LocSpec attributes for a span of text in an HTML document are:

Reference a HTML target name (if provided by the document 'owner').

Selection the text of the anchor.

Selection Context some text surrounding the anchor.

Axis Specification a position (e.g. the starting position of the anchor) and length of anchor on a character axis.

With this redundant information, WebVise could detect and repair broken links in a variety of situations. For example, if a user edits a document whilst disconnected from the open hypermedia service, the following procedure can be performed for every LocSpec registered for the document the next time that the document is browsed using the service:

1. Highlight the span of text in the browser corresponding to the LocSpec's axis specification, and inform the user that the old target was represented by this span.
2. If the text in the highlighted span is identical to the LocSpec's selection attribute (user edits have not affected the anchor), ask the user to confirm updating of the anchor with new LocSpec information (selection context may have changed).
3. If the text in the highlighted span is not identical to the LocSpec's selection attribute, ask the user whether a search for nearby occurrences of (parts of) the string in the selection and selection context attributes should be initiated.
4. If the search is successful, ask the user to confirm updating of the anchor with new LocSpec information (reference, selection context, and axis specification may have changed).

Xanadu (Nelson, 1980) could in theory record a complete edit trail of every character insertion and deletion, or a versioning history of edits in larger chunks and time periods, so that link integrity would never be compromised (a preventative strategy).

4.4.1 Robust Hyperlinks

A recent corrective approach to the dangling link problem is Robust Hyperlinks (Phelps and Wilensky, 2000a). Whereas previous strategies have relied on authors maintaining their own links, "administrative buy-in", infrastructure creation, or agreement on conventions, Robust Hyperlinks do not rely on authors or the party administrating or 'owning' the resource. A Robust Hyperlink consists of a URL augmented with a small "signature", consisting of a few carefully chosen words from the target document which effectively identify it by its content — empirical studies showed that signatures of about five words are sufficient. If the URL fails, the signature can be used to query Web search engines in an attempt to locate the missing document.

Robust Hyperlinks have been implemented in several ways, including a proxy service which transparently makes robust all URLs that pass through it, and a "scriplet" which can be kept as a bookmark in conventional Web browsers. Figure 4.9 demonstrates the scriptlet implementation in Internet Explorer. In Figure 4.9a, the user has encountered a dangling link. Note the lexical signature of the missing target page encoded in the URL. The user activates the scriplet through a bookmark, which displays a search options page (Figure 4.9b), from which the user can choose which search engine to submit the signature of the missing page to. In Figure 4.9c, the user has chosen the Google search engine, which successfully returns the new (unique) location of the missing page.

Alternatively, if a missing page has been deleted entirely from the Web, the URL could also be used to check if the page has been archived in the Internet Archive by submitting it to the *Wayback Machine* (Feise, 2000) (as used in the investigation reported in Section 3.3.2). However, the designers of Robust Hyperlinks argue that such archives are more likely to be incomplete.

Robust Hyperlinks are not without limitations, for example the scheme is ineffective for a given document during the period in which the document has moved, but no search engine has yet indexed it in the new location. The scheme is also ineffective for document formats which are not indexed by some search engines, such as Postscript and Adobe's Portable Document Format (although Google has recently evolved an indexing strategy for such formats, and other search engines may be expected to follow suit). However, Robust Hyperlinks do have a number of desirable qualities: they are small and cheap to compute, do not require new server or infrastructure support, fit in well with the existing URL syntax, and are easy to understand.

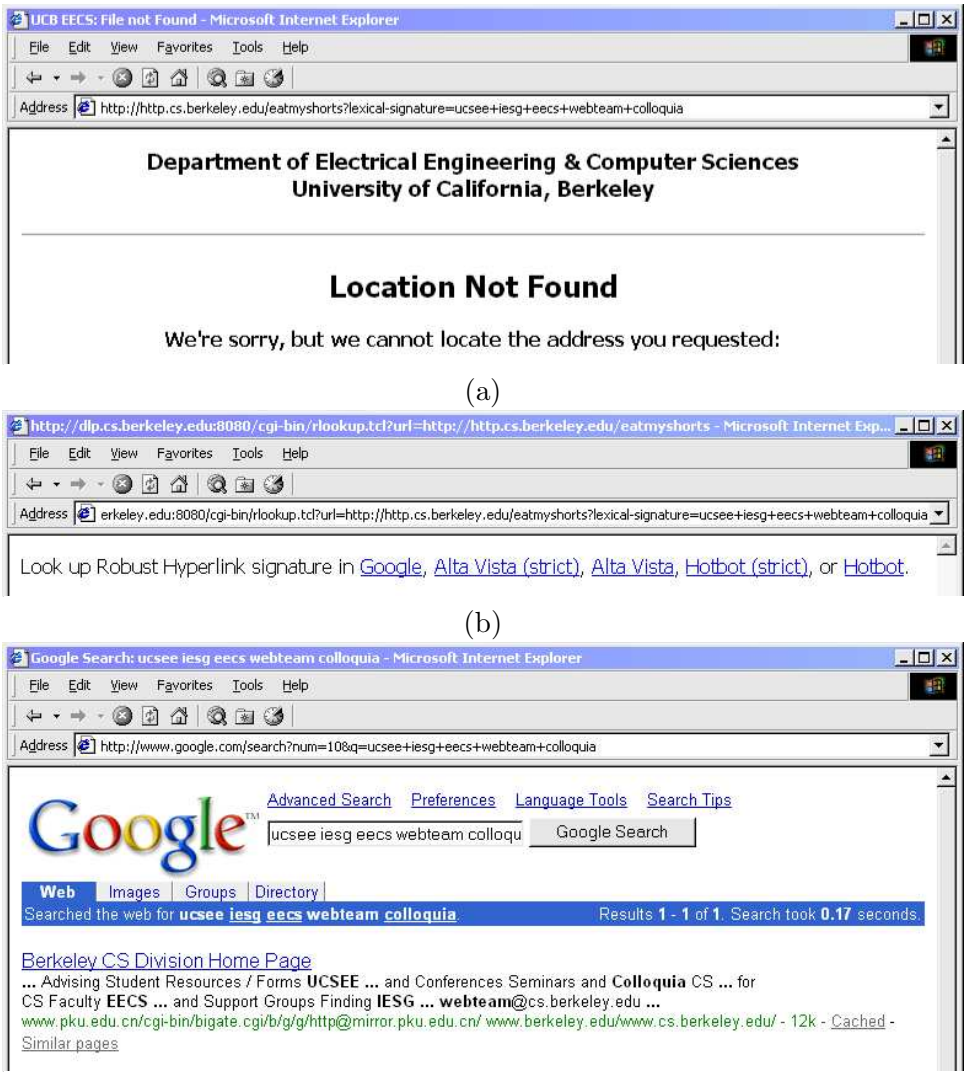


FIGURE 4.9: Robust Hyperlinks implemented using bookmark scriptlet (Phelps and Wilensky, 2000a).

4.4.2 Link Integrity in XLink

The XML Linking standard implicitly allows "dangling Xlinks": links may have only one participating resource, or none at all, and simply be untraversable. Such a link may still be useful, for example, to associate properties with a single resource by means of XLink attributes, or to provide a placeholder for link information that will be populated eventually. XLink uses the URL addressing scheme to locate existing documents, but leaves to the client the choice of what to do if the referenced document has been moved or deleted. Robust Hyperlinks describe a mechanism for monitoring the referenced URL for correctness and attempting to discover the new location of the page — Robust Hyperlink "signatures" could be feasibly be incorporated into XLink specifications as part of the URL reference, integrating the two approaches.

The XPointer specification used by XLink provides some level of robustness against the editing problem by locating anchors relative to a nearby element which has been assigned an ID, but (Phelps and Wilensky, 2000b) note that the task of recovering failed location pointers in the face of change seems to be outside the domain of the XPointer standard. XPointer describes many means for traversing a document tree, including node position, and id and attribute values, but leaves to the client the choice of what to do if a previously stored tree path is no longer available in an edited document.

4.5 Discussion: Challenges

The limitations of the Web as a publishing medium for integrated hypertexts leads to the proposal of two technical challenges: to enrich the Web's hypertext model to better support integrated hypertexts, and to prevent these integrated hypertexts from becoming permanently disconnected from the global context on which they build. Various approaches to these challenges have been examined.

Work on integrating the Web with open hypermedia systems has shown that open hypertext technology can be used to enrich the Web's hypertext model. For example, Hyper-G allowed writers to create bi-directional links between arbitrary documents and ad-hoc document collections, the Distributed Link Service brought Microcosm's generic links to the writer's toolkit, and WebVise successfully integrated the Web with Microsoft Office applications, allowing users to create more advanced hyperstructures across arbitrary Web and Office documents. The Arakne framework, a model of common components used in such approaches, may provide a useful basis for further development. However, the success of such systems depends on users actively seeking out and using them — their widespread adoption has subsequently been limited. The XML Linking standards, by contrast, bring advanced hypertext capabilities to the core of the Web, and must be universally supported, so adoption of this standard may also be useful. The automatic selection of links "in context" by the QuIC system may not be so useful in meeting this challenge since this work focuses on hand-crafted links between specific anchors rather than generic links.

A number of Web link management strategies have been examined which may be useful in meeting the second challenge. Such strategies included forward mechanisms and gravestones, relative references, embedded links, dereferencing (URNs, PURLs, DOIs), notification (Hyper-G), and detect-and-correct (WebVise). Robust Hyperlinks as an approach for correcting "broken links" seems particularly useful since it does not place the onus of link maintenance on writers or 'owners'. By postfixing a Web URL with a few carefully chosen words that uniquely identify the referenced document, this "signature" can be submitted to a search engine to determine the new location of the document when the original URL fails. Robust Hyperlinks also complement the XML Linking standard.

4.6 Summary

Following the reports of evidence of Associative Writing in the Web in Chapter 3, this chapter has discussed how the Web's limited hypertext model restricts the expressive capabilities of the writer of such integrated hypertexts, and ultimately the extent to which new contributions can be integrated with existing content. A discussion of these limitations, and of approaches to overcoming them, led to the introduction of two technical challenges for this work.

Firstly, the Web's model of the basic building blocks for integrated hypertexts is restrictive; only uni-directional, binary links can be created, which must be embedded in the documents 'owned' by the writer. The challenge is therefore to enrich the Web's hypertext model to better support the publication of integrated hypertexts. This chapter has shown how this could be achieved by integrating the Web with open hypertext technology, or by adopting the recent 'next generation' XML Linking standards.

Secondly, the chaotic nature of the Web frequently causes links to 'break' — documents are edited, moved, and deleted in an ad-hoc manner, breaking any links that refer to them. The challenge therefore is to try and prevent integrated hypertexts from becoming permanently disconnected from the existing context on which they build. This chapter has shown how this could be achieved by adopting one of several link management strategies, including Robust Hyperlinks, a low cost approach which does not place the onus of link maintenance on the writer or 'owner' nor require new server or infrastructure support.