Chapter 10

Comparison of AWF with Previous Work

The aim of this chapter is to compare the Associative Writing Framework approach with the work in open hypermedia, hypertext writing, hypertext annotation, and link integrity, introduced over the course of the preceding chapters. A "unified framework" for the integration of AWF with existing approaches is also proposed.

10.1 AWF is an Open Hypermedia System

Chapter 4 concluded that the World-Wide Web provides only a *limited hypertext model*; the expressive capacity of the writer in creating and publishing an integrated hypertext is therefore also limited. Approaches such as Hyper-G (Andrews et al., 1995b), the Distributed Link Service (Carr et al., 1995), and WebVise (Grønbæk et al., 1999) have attempted to augment the Web's hypertext model with more advanced hypertext capabilities by retro-fixing existing open hypertext systems to the Web. More recent approaches, such as Arakne, have designed open hypertext systems specifically to facilitate Web augmentation. In addressing the "limited Web hypertext model" challenge, AWF adopts an open hypermedia approach in order to provide an environment for integrated writing through the structure management services of the AWF Server, and the integration facilities offered by established browsing (Internet Explorer) and writing (Microsoft Frontpage) tools.

By way of introducing a comparison between AWF and existing "open hypermedia meets Web" approaches, Table 10.1 lists the five different integration techniques used by open hypermedia systems as reported by (Anderson, 1997). AWF integrates with the Web using a *Web client as OHS client* strategy — the Microsoft Internet Explorer browser is extended to incorporate the Annotate toolbar which communicates with the AWF

Integration strategy	Demonstrated by:
OHS data to Web data	Microcosm
Web client as OHS client	WebVise, Arakne, **AWF**
Web server as OHS client	DLS, Chimera
OHS Server as Web Server	Hyper-G
Web protocols within an OHS	HyperDisco, Chimera

TABLE 10.1: Classifying AWF alongside open hypermedia systems using Anderson's taxonomy of integration techniques (Anderson, 1997).

Server, and other components of the framework.

As noted in Chapter 7, the latest versions of the Internet Explorer and Mozilla browsers have lent themselves particularly well to integration (Esposito, 1999), including the ability to create and register third-party browser extensions such as toolbars and sidebars; a technique also demonstrated by other *Web client as OHS client* systems such as WebVise and Arakne. This approach is useful as it enables the functionality of a popular and widely distributed Web browsing environment to be extended rather than attempt to develop an alternative browsing environment from scratch (Hyper-G's Harmony browser), or attempt to provide additional functionality through a Web proxy service (Distributed Link Service) and suffer the inherent weaknesses of this design (Vasudevan and Palmer, 1999). This kind of integration does not suffer the problems of modifying existing browser code as, for example, early experiences with the Distributed Link Service reported (Carr et al., 1995); components are cleanly separated from the browser itself, and the interface which integrates them remains consistent as new browser versions and updates are released.

More advanced structure support in approaches such as Hyper-G, the Distributed Link Service and WebVise could provide potential future extensions to the current implementation of AWF. Hyper-G allowed Web pages to be grouped into nested collections, which can be the source of target of links; the Distributed Link Service and WebVise allowed *generic* (or *global*) links to be created over pages (the creation of such dynamic links was not included in the current implementation of AWF by design, focusing instead on specific connections between ideas as identified by the writer); WebVise also provided higher-level 'guided tours' and 'contexts' (layers of related structure), and facilitated hyperstructures across Microsoft Office documents. However, it is worth noting that the AWF Server implementation differs from these approaches in its built-in ability to serve hyperstructures in the form of graphical overview maps.

Figure 10.1 illustrates how AWF fits into the Arakne model (Bouvin, 1999). AWF's Annotate, Relate, and Visualise components take the form of "navlets" in the content layer, the AWF Server resides in the structure layer, and the service layer models the communication between them (note that unlike Arakne AWF does not use a proxy for Web augmentation). This model neatly illustrates the separation between structure

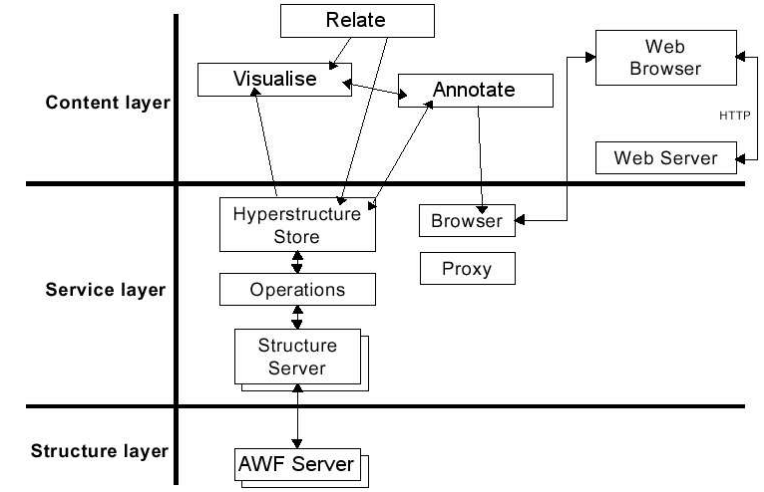


FIGURE 10.1: Fitting AWF into the Arakne model.

service and interface components; indeed a structure service from one of the approaches listed above could potentially be used in place of the AWF Server in order to provide more advanced features.

As an example, the AWF Server is in fact based on the Annotea (Kahan et al., 2001) annotation server, a open structure service which serves annotations rather than links. Since AWF utilised the annotation templates specified by Annotea, adopting an Annotea server seemed the natural implementation approach. However, as the implementation progressed it became apparent that extending the Annotea server to both store link information and serve overview maps would be problematic, so the more generic (and simplified) AWF Server was implemented. However, resorting to a bespoke implementation meant that Annotea's extensive filtering facilities were lost: as a result, AWF users have no mechanism for controlling heavily annotated documents. A return to using an extended Annotea server in the role of "structure service" (according to the Arakne model) may help improve matters.

It is also worth mentioning that although the XML Linking (XLink) standard (W3C, 2001b) potentially adds advanced hypertext functionality to the core of the Web, and will be universally supported (whereas users must actively seek out and install AWF), the standard was not adopted in this initial implementation of AWF due to limited browser and software support at the time. Although XLink is now arguably more mature and may eventually be adopted as a storage and interchange format, the focus of the implementation work to date has been on the writing process (how the writer makes annotations and links), rather than the storage format. Moreover, XLink will not replace the function of AWF itself; writers use AWF (which must still be sought out and installed) to create structures during the writing process which may then be represented and interchanged in XLink.

	Planning	Context Building	Structure Building	Integrated Writing
WE	·	·	●	·
SEPIA	●	○	●	○
SWT	●	●	●	·
gIBIS	·	·	●	·
MacWeb	·	·	○	·
Aquanet	·	·	●	·
AWF	○ †	●	○ †	●
VIKI	·	○	●	·
VKB	·	○	●	○
ART	·	·	○	·
Tinderbox	·	○	●	○
Wiki	·	·	·	○
StorySpace	·	·	●	·
Conn. Muse	·	·	○	·
H. Gatherer	·	○	·	○
Frontpage	○	·	○	○

†Partially supported through Microsoft Frontpage's built-in 'Task' and 'Navigation' views (Chapter 6).

●	Strength
○	↕
·	Potential weakness

TABLE 10.2: Comparing AWF with hypertext writing approaches in terms of support for Associative Writing activities.

10.2 AWF is a Hypertext Writing System

Chapter 6 concluded that the surveyed hypertext writing approaches (Table 10.2) provided little support for the context building and integrated writing activities of Associative Writing. The AWF approach of bridging document-based and map-based writing approaches was therefore designed specifically to address the weaknesses in these areas, and represents an innovative contribution to this research area (Table 10.2). As Table 10.2 also shows, AWF is a *descriptive* writing approach according to Marshall's spectrum (Marshall et al., 1994) — writers define semantics which are then available within the framework.

10.2.1 Comparing Support for Context Building

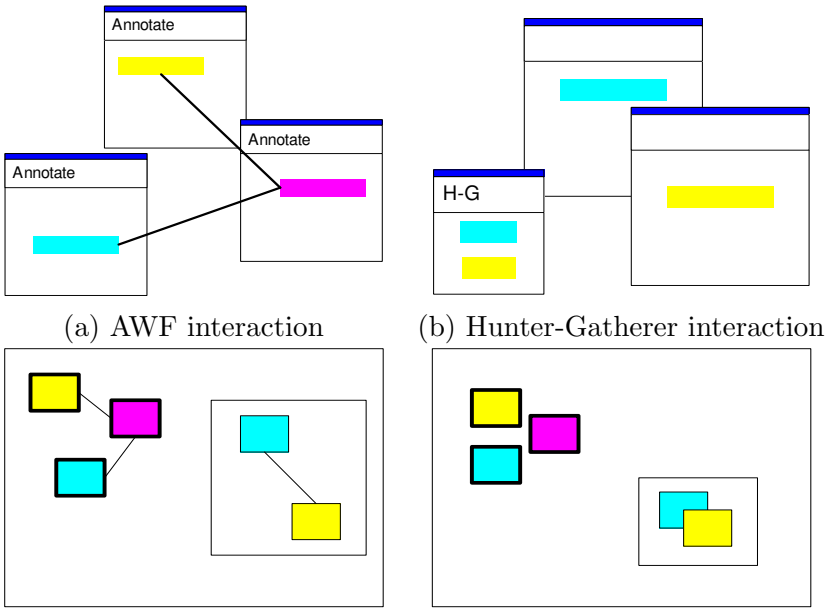
To recall from the Associative Writing model in Chapter 6, in the context building activity the writer identifies and explores the "global context" of their new contributions — the existing ideas, concepts, data, examples, description, theories, *etc.*, and the implicit connections between them —which the new hypertext will build on. AWF's Web-integrated Annotate and Relate components were designed to support this activity: Annotate allows writers to highlight relevant content in existing Web pages, optionally

annotating the highlighted text with new ideas or interpretations and assigning a semantic type to the annotation; Relate allows writers to capture relationships between highlighted content as implicit relationships are subsequently uncovered.

As with other document-based approaches, such as Hunter Gatherer (schraefel et al., 2002) and WebVise (Grønbæk et al., 1999), the writer is engaged directly with the Web docuverse, but in using the Relate component to create links between highlighted content from different documents, the writer potentially engages with several nodes of the hypertext at once without having to switch focus to a separate (map-based) workspace. AWF's Relate component therefore bridges the gap between reading in the Web docuverse and representation in the workspace. This interaction was in fact extended from the link creation interfaces demonstrated by WebVise and Microcosm (Fountain et al., 1990). For example, to capture a connection between content in two different documents using WebVise, the writer has to (1) highlight the source 'anchor', (2) choose "New Link" from the WebVise menu, (3) highlight the target 'anchor', and (4) choose "add anchor" from the menu. Microcosm had earlier allowed several links to be created in parallel — each 'add anchor' operation displayed a small dialog which could be left on screen until the writer added a target anchor by pressing a button on the dialog. AWF was designed to extend this document-based interaction to allow anchors (annotations) and links to be created independently (user interaction is not driven by a higher-level 'link creation' process — highlighted text may serve simply as a marker of 'something' significant).

Although, as noted in Chapter 8, AWF affords a simple, informal interaction through the use of the 'none' type, further support for the gradual emergence of structure and meaning is not provided. The map-based VIKI (Marshall et al., 1994), Visual Knowledge Builder (Shipman III et al., 2001), and Tinderbox (Eastgate Systems Inc., 2002) (Map view) approaches, however, allow writers to express developing (potentially ambiguous) relationships between nodes visually and spatially. Such approaches make the AWF Server's overview map facility seem limited in comparison, but implementing such a workspace has not been the focus of AWF. The anticipation is that support for other activities (such as the well-supported structure building activity) and approaches (such as emergent structure) will be integrated into the framework in future iterations of the AWF implementation — indeed the VIKI, VKB, and Tinderbox approaches have already shown some initiative in this direction by providing some limited URL-based interactions.

Figure 10.2 attempts to illustrate some of these proposals. In Figure 10.2a, the writer captures a simple hyperstructure using AWF's document-based Annotate and Relate components. These structures then automatically appear in the writer's map-based workspace when attention is later switched to "information triage" (Marshall and Shipman, III, 1997b) (Figure 10.2a,c,d). The workspace supports the rearrangement and reinterpretation of content and structures, and possibly advanced features such as Vi-



(a) AWF interaction (b) Hunter-Gatherer interaction

(c) Map workspace (d) Spatial workspace

FIGURE 10.2: Bridging document- and browser-based hypertext writing tools using an integrated AWF/Hunter-Gatherer approach.

sual Knowledge Builder's navigable history mechanism, image objects, and suggestion manager (Shipman et al., 2002). A particular approach under consideration involves leveraging Tinderbox's XML capabilities (and also perhaps extending the URL AutoFetch mechanism to handle XPointer references to source documents). Changes made in the workspace would then be reflected back onto the desktop as the writer revisited the source pages — the importance of such smooth, ad-hoc transitions between writing activities have been emphasised (Smith et al., 1987; Streitz et al., 1989; Neuwirth and Kaufer, 1989). The contrast between AWF as a marking and linking approach and Hunter Gatherer as a information collecting approach adds a further scenario in which an AWF/Hunter-Gatherer integration approach provides both content annotation and gathering tools for writers engaged in the context building activity (Figure 10.2b,c,d).

10.2.2 Comparing Support for Integrated Writing

To recall from the Associative Writing model in Chapter 6, in the integrated writing activity the writer elaborates their new contributions into a hypertext which integrates with the underlying 'global context' identified in the context building activity. AWF's integration and extension of the Microsoft Frontpage editing tool allows the writer to build new hypertexts on the content and structures captured using the Annotate and Relate components. By extending this established editing environment, AWF allows writers to continue to take advantage of the familiar editing facilities. In contrast, many of the surveyed approaches consider writing as an "open-ended design task" (Neuwirth and Kaufer, 1989), and so provide no support for a writing/composing activity (al-

though MacWeb's "virtual documents" (Nanard and Nanard, 1991), Hunter Gatherer's linked "collection view", and Visual Knowledge Builder's "export" function provided some translation from representation to document or hypertext form). Although the Writing Environment (Smith et al., 1987) and ART (Yamamoto et al., 2002) approaches do support a writing activity, the target is a linear text, rather than hypertext. StorySpace (Bernstein, 2002) does allow writers to produce a Web hypertext, but as a primarily fiction-based tool, this hypertext is not integrated.

Of the remaining systems, SEPIA's notion of a "dual hypertext" (containing both a "surface" presentation of information about the argument with access to the writer's argumentation structure for exploring the argument at a deeper level) seems closest to the AWF approach. SEPIA also helps writers create a 'guided tour' through the argumentation network, a feature which could be incorporated in future implementations of AWF (perhaps as an added measure to reduce the effects of "muddled writing"). Wiki (Leuf and Cunningham, 2001), on the other hand, differs from the AWF approach in that writers can edit existing content directly, adding or deleting content and links (albeit subject to the limitations of the Web) — AWF annotations and links are stored separately from content by the AWF Server. Tinderbox's tools for building Web sites from notes in the workspace according to user-specified 'export templates' could be used to create links to other Web pages (based on the URL attribute of Tinderbox notes), but are limited in comparison to AWF's integration capabilities.

10.2.3 Transpointing Windows in Hypertext Writing

Although AWF's visible hypertext link connections were conceived as an extension of the link creation mechanisms of WebVise and Microcosm towards map-based interactions, and applied as a novel and innovative approach to supporting hypertext writing, visible hypertext link connections themselves are not an entirely new concept. Nelson advocated visible connections between documents for side-by-side comparison as early as 1972 (Nelson, 1972), for which the term "transpointing windows" (Nelson, 1995) was more recently coined. Nelson advocates that transpointing windows address "the fundamental problem of hypertext: being able to see connections side by side", whether the user is looking at separate pages, different versions of the same document, commentaries on one document by another, or any other parallel document structure[1]. Nelson's proprietary CosmicBook Reader tool (Figure 10.3) finally demonstrated these ideas on the desktop when it was released in 2001[2]. CosmicBook Reader is a viewer for reading and displaying "flights", collections of text documents and link information bound together into a single file for easy distribution. A beta version of an editor tool for creating flights

[1] `http://xanadu.com.au/ted/TN/PARALUNE/paraviz.html`
[2] `http://xanadu.com/cosmicbook/`

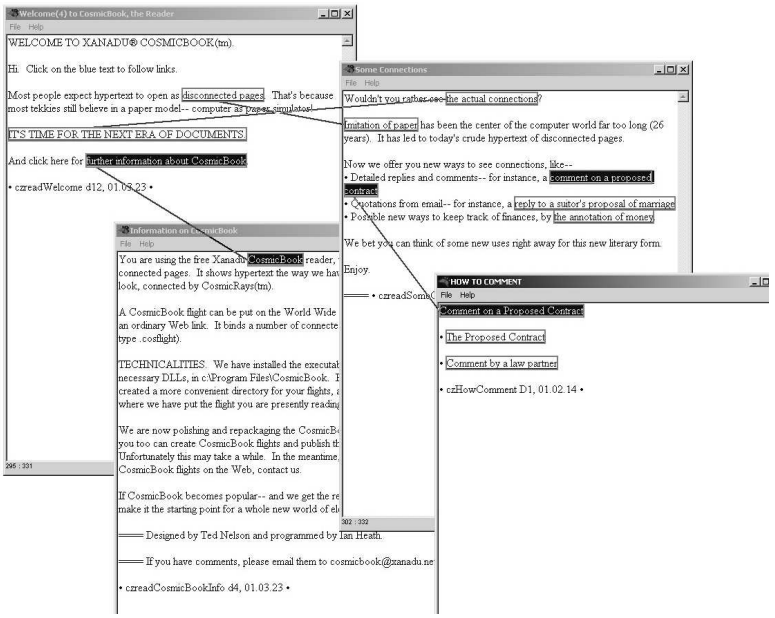


FIGURE 10.3: A CosmicBook flight.

exists, but has not yet been released for public consumption[3].

However, it is not clear whether the editor uses a 'digital ink' interaction similar to AWF, nor whether user trials of the CosmicBook reader/editor have been carried out. A related benefit of the AWF case study and 'framework evaluation' reported in Chapter 9 is therefore that the results may be of significance to researchers interested in evaluating the transpointing windows approach. Indeed, AWF demonstrates a number of features above and beyond those of the current CosmicBook (reader) release, which could in turn inform future development of the CosmicBook project:

- Web integration — AWF is implemented as extensions to Internet Explorer and uses open RDF templates for representation and interchange, rather than as a proprietary viewer and document format.
- Open hypertext approach — AWF annotations and links are managed externally, allowing different users to explore and contribute to a shared interpretation.
- Link creation — writers can create anchors (annotations) and links on existing Web pages.
- Advanced link display — AWF can display link metadata (semantic type, creator, creation time) on visible connections; and display partially obscured links as 'ghosted' connections.

[3]Personal communication with Ian Heath, lead programmer of the CosmicBook project, February 2003.

	Context building			Integrated	Semantics
	Emphasise	Annotate	Link	writing	
AWF	●	●	●	●	●
Internote	○	●	·	·	·
XLibris	●	●	●	·	○
MVA	●	●	○	·	○
Annotea	○	●	○	·	●
Fluid A.	○	●	○	·	○
MRS	○	●	·	●	·
CREAM	○	·	○	●	●
TRELLIS	●	○	●	●	●

●	Strength
○	↕
·	Potential weakness

TABLE 10.3: Comparing AWF with surveyed annotation approaches in terms of support for context building activity.

10.3 AWF is an Annotation System

Chapter 7 concluded that there seemed to be no 'complete' annotation-based approach to the challenge of supporting context building in the Web (Figure 10.3). In meeting this challenge by extending the direct annotation of documents beyond the physical boundaries of the page and onto the desktop, AWF contributes a novel interface for annotation to this research area.

Creating annotations with AWF's Annotate component is a similar process to that used by InterNote (Catlin et al., 1989), Annotea (Kahan et al., 2001), Multivalent Annotations (Phelps and Wilensky, 1997), Fluid Annotations (Bouvin et al., 2002), and MRS (Miles-Board et al., 2003a) — the writer highlights a specific text span and then chooses an 'Annotate' option from a menu or toolbar. The writer may then optionally adds a comment to the highlighted text (the highlighted text may even be part of an existing annotation comment). Other features of the surveyed approaches were beyond the scope of the AWF implementation: Annotea allowed the writer to annotate a Web document as a whole (although note that in AWF a Web document can be treated as a link endpoint), and to filter heavily annotated documents; in XLibris (Price et al., 1998b), readers annotated documents using free-form "digital ink" marks, and Multivalent Annotations offered more complex annotation mechanisms, including digital ink and "copy-edit"marks, and geometric and structural annotations.

AWF's *Relate* component allows writers to capture connections between annotations as they are displayed on the desktop. By contrast, XLibris is designed to display only one page of a document at a time, and required an innovative design of "ink anchors" and "clipping views" to facilitate the creation and exploration of links between documents.

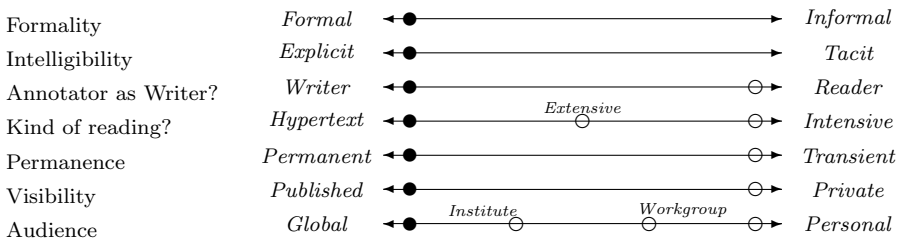


FIGURE 10.4: Overview of AWF according to Marshall's dimensions.

As a multi-document system, AWF extends the possibilities of digital ink annotation beyond the physical boundaries of the document. Annotea, Multivalent Annotations, and Fluid Annotations are also potentially multi-document annotation systems, but mechanisms for capturing relationships between annotations are limited. Capturing such relationships is better supported by CREAM and TRELLIS, although the types of connection that can be made were limited to those allowed by a domain ontology; semantics in AWF can be extended by individual users to meet specific requirements.

MRS, TRELLIS and CREAM were identified as the only annotation-based approaches which demonstrated a form of integrated writing — using annotations as a basis for creating new material: in MRS, managers created integrated hypertexts through the process of collating and importing important issues from other reports; in CREAM, integrated hypertexts could be created 'piece by piece' by dragging and dropping attributes and ontological relationships into a new document; in TRELLIS the processes of creating annotations and links produced a new hypertext. Like MRS and CREAM, AWF's integrated writing facilities extend an existing writing environment. In contrast to MRS, links in AWF are explicitly created by the writer (rather than implied by the gather/import process). AWF also extends the MRS, TRELLIS, and CREAM approaches to allow the writer to create links to larger-scale structures.

In the context of this work, AWF's Annotate component is intended as demonstration of the ideas embodied by AWF rather than as alternative bespoke solution to the existing annotation tools surveyed. The integration of existing map-based writing approaches with the framework have already been discussed — it is further anticipated that AWF's Annotate component will be replaced with a more fully-featured annotation interface, such as Multivalent Annotations.

10.3.1 Dimensions of Annotation

(Marshall, 1998) describes the "dimensions" of annotation which reflect the *form*, *function*, and *role* of an annotation approach (Figure 10.4). The *form* of annotation is described by the dimensions of formality and intelligibility.

Formality The most formal type of annotation is *metadata*, specifically metadata that

follows standard structure and naming conventions. At the other end of the spectrum, informal annotations are unconstrained (a typical example being a reader's marginalia in a textbook). AWF's adoption of the Annotea templates places the framework at the formal end of the spectrum (also CREAM, TRELLIS), in contrast to informal approaches such as XLibris.

Intelligibility Intelligibility is bounded by explicit annotations (intended for others to read) and tacit annotations (personal annotations that anyone other than the original annotator will find difficult to interpret). AWF's shared annotation environment places the approach towards the explicit end of the spectrum (with Annotea, CREAM, and TRELLIS), in contrast to "digital ink" systems such as XLibris and MVA.

The *function* of annotation reflects the degree to which the annotator has become a writer, the kind of reading that the annotator is engaged in, and the permanence of the annotations.

Annotator-as-writer The process of annotation in AWF is part of a higher-level integrated writing task, and so in this sense AWF could be placed towards the 'annotator-as-writer' end of the scale. However, AWF could also be used simply as a reading aid, placing it at the 'annotator-as-reader' end of the spectrum with XLibris

Kind of Reading (Levy, 1997) describes hyperextensive, extensive, and intensive reading. Annotations during hyperextensive and extensive reading are (respectively) link- and structure-oriented, in which two or more documents are involved and the entire hypertext is in the foreground. Annotations in intensive reading are within-document annotations, in which engagement is primarily with a single document, and the hypertext is in the background. AWF is first and foremost an example of a hyperextensive approach (also CREAM, TRELLIS), although its use in an extensive (also Annotea and Fluid Annotations) or intensive scenario (also Internote, XLibris) is also possible.

Permanence Typically, AWF's annotations are permanently retained by the AWF Server, and as such the approach seems less transient than that of Internote, XLibris and MVA. However, in the case that a (private) AWF Server is maintained locally by a user, annotations may be removed if required.

The *role* of annotation reflects its visibility and target audience.

Visibility In the vision of an integrated Web, AWF, along with other approaches such as InterNote, Annotea, Fluid Annotations, CREAM, and TRELLIS publishes annotations on public servers. A local AWF installation however, provides private annotations, as does XLibris.

Target Audience In the vision of an integrated Web, AWF annotations are published on public servers, accessible by any Web user (global audience); however, it is also possible that an AWF Server be managed and shared within an Intranet (institutional audience) or community of collaborating writers (workgroup audience), or even used locally by an individual user (personal audience). Annotea, Fluid Annotations, CREAM, and TRELLIS could also target a workgroup, organisation, or global audience; InterNote and XLibris target a workgroup or personal audience.

10.4 Link Integrity in AWF

Chapter 4 concluded that the chaotic nature of document changes in the Web frequently compromises link integrity. Numerous link management strategies have been applied by different approaches (Ashman, 2000), in an attempt to keep track of moved pages and correctly reposition link anchors (or annotations) in modified pages. In addressing the "link integrity" challenge, AWF currently adopts two preventative strategies in an initial attempt to provide some protection against integrated hypertexts becoming disconnected from the existing content on which they build:

1. The "editing problem" (Davis, 1998) in the writer's new hypertext is prevented by using an "embedded links" strategy (Davis, 1998). During editing, AWF annotation and link information is embedded in the new hypertext (although hidden from the writer in Frontpage's 'Normal' view) during editing, to prevent location references from being broken as the text changes (maintaining such references externally would cause problems as the content is continually revised). When the finished hypertext is published, a manually triggered 'update' process crawls the hypertext and stores extracted annotation and link metadata from each page on the AWF Server (however, this process must be performed each time the content or location of the hypertext is changed).

2. By adopting the Annotea templates (Kahan et al., 2001) for representing annotations (and extending the template architecture to also represent links), AWF takes advantage of the natural robustness properties of XPointers as a means to partially overcome the "editing problem" in existing pages which the writer has annotated. As a simple example, Figure 10.5 shows how an XPointer may survive document changes. Changes of any magnitude to sibling subtrees at the level of the `FONT` element are safely contained (provided that no new `FONT` elements are added *before* the first one). Similarly, changes of any magnitude to sibling subtrees at the level of the `P` element are safely contained (provided that no *P* elements are added or deleted *before* the sixth one). However, this strategy is not guaranteed to succeed, and AWF currently adopts no corrective strategy if an XPointer fails.

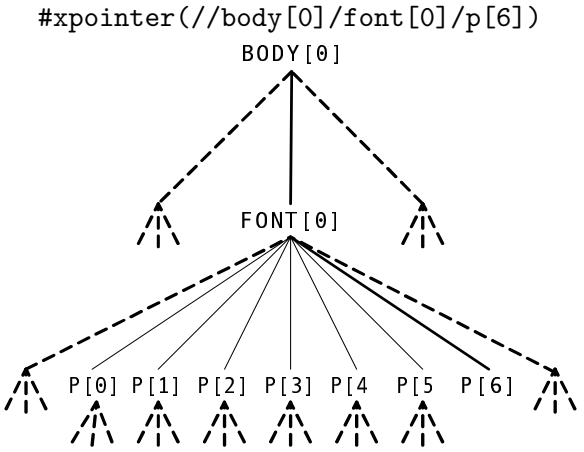


FIGURE 10.5: XPointer provides some natural robustness to document changes.

10.4.1 Comparing current preventative strategies

AWF's manually-triggered "update" approach has some similarities to the notification strategy used by Hyper-G (Andrews et al., 1995b), in which warnings were automatically issued to all known applications holding links over a changed document (provided changes were made through the Hyper-G system). Future implementations of AWF could therefore include an automatic detection of document changes (perhaps by storing the last modified date each annotated document, and checking this date each time AWF structures are added to the document — a technique used by Microcosm).

Other preventative strategies such as relative references and dereferencing were deemed unsuitable for the AWF approach, and indeed for Associative Writing in general. A relative referencing strategy (as opposed to absolute document addresses) works well within local collections of documents, but cannot be applied when links integrate many different documents from different servers. A dereferencing strategy (where aliases are used to lookup an up to date document address in a registry) could be applied in an integrated writing scenario, but would rely on the original authors of each integrated work to keep the address registry up to date.

The Web annotation approaches HotOffTheWeb and CoNote (Davis and Huttenlocher, 1995) applied preventative strategies by restricting the annotation options available to the writer. HotOffTheWeb's geometric annotations were anchored in position by fixing the dimensions of the browser window (although changes to the underlying content would still 'break' annotation positions), whereas CoNote only allowed writers to add annotations to positions in the document predefined by the original author. AWF enforces no such restrictions: by keeping track of window interactions, annotations and links are not 'broken' by resizing browser windows, and writers are free to annotate arbitrary content in existing documents.

10.4.2 Towards corrective strategies

To better prevent integrated hypertexts becoming disconnected from the existing content on which they build, future implementations of AWF should incorporate corrective strategies for both 'dangling links' (when a target page is deleted or moved) and the 'editing problem' (when the location of an annotation within a page cannot be resolved because the page has been edited).

Robust Hyperlinks (Phelps and Wilensky, 2000a) as a strategy for correcting dangling links seems ideally suited for integrated writing, since Robust Hyperlinks do not rely on writers maintaining their own links, nor require any infrastructure changes. The computation of a page 'signature' is straightforward, and since Robust Hyperlinks use a URL format, the Robust Hyperlink can easily be represented and encoded within the existing AWF annotation templates. AWF itself could provide support for Robust Hyperlinks in the absence of browser-based support — AWF's Annotate component is already embedded in the browser. However, a vendor-supported, browser-based implementation would mean that readers could take advantage of Robust Hyperlinks whilst browsing integrated hypertexts without having to install AWF.

A problem may arise if the signature resolves to multiple results from the search engine (or no results at all). If an existing page is moved during the Associative Writing process, it is reasonable to expect the writer to be able to correctly identify the new location of the page from the search results, but how does the reader of the published hypertext, when confronted with this option, correctly identify the new location of the page which the author originally annotated? A possible solution may be to have AWF automatically examine each new page location suggested by the search engine, attempt to resolve that page's annotation locations (possibly using a corrective reattachment algorithm — see below), and then choose the page in which all (or most) annotations can be resolved.

In terms of the editing problem, WebVise and Robust Locations (Phelps and Wilensky, 2000c) have demonstrated corrective strategies that could also be integrated into the AWF approach. WebVise (Grønbæk et al., 1999) combined 'LocSpecs' with a reattachment algorithm to relocate link anchors broken due to document changes. By recording 'redundant information' about a link anchor (such as the text of the anchor, and some of the text surrounding the anchor), as well as an 'axis specification' (the anchor location, similar to AWF's XPointer representation of annotation locations), WebVise could repair broken links (in close co-operation with the user, who had to direct or confirm each stage of the process) by searching the document for occurrences of fragments of the recorded text. Equivalent 'redundant information' (annotated text, text surrounding annotation) is not currently recorded in AWF, but again the open and extensible RDF templates adopted by AWF would easily facilitate the encoding of this information within the existing framework.

Robust Locations demonstrated a more advanced (and more automated) variation of the WebVise approach in reattaching broken annotation locations in the Multivalent Annotation (MVA) system (Phelps and Wilensky, 1997). Span annotations in MVA were modelled as two 'tree walks' leading to the start and end of the span (in contrast, AWF's equivalent XPointer representation describes the entire span using a single 'tree walk', although two tree walks can be encoded using XPointer representation). Again, redundant information was recorded in the form of the text surrounding the start and end points of the span. A reattachment algorithm used both structural and text search strategies in an attempt to reattach broken annotations, marking annotations with a 'confidence' value if they could be successfully reattached.

However, just as a non-unique search result presents a (reader) problem in the Robust Hyperlinks approach, adopting Robust Locations in AWF also presents a reader problem in the case of orphaned annotations (annotations which cannot successfully be reattached after the original document has undergone a series of changes). Typically, orphaned annotations are made available for manual reattachment by the user; ComMentor placed orphan annotations at the end of the document, whereas unresolved Robust Locations were presented in separate pop-up windows. Again, it is reasonable to expect the writer to be able to reattach orphan annotations during the Associative Writing process, but when the new hypertext is published, can readers be expected to perform this function? (Brush et al., 2001) noted that users found it difficult to reattach orphaned annotations that they had not created, even when part of the original text associated with an annotation was available in the document. In the absence of a versioning strategy, in which every version of a published work is available — cf. Xanadu (Nelson, 1980), Chimera (Anderson et al., 1994) — the original annotated text may be at least preserved and presented (stored 'redundant information'). Otherwise, integration with a Web versioning facility, such as WebDAV (Whitehead Jr and Goland, 1999), may be the only way of protecting the potential 'fragility' of AWF's integrated hypertexts.

10.5 Bridging Approaches: Unification of Proposed Future Developments

In comparing the AWF approach with various systems from different research areas, a number of proposals for the integration of existing approaches within the framework have been suggested. The purpose of this section, therefore, is to discuss how these ideas could be unified into a coherent AWF architecture.

Figure 10.6 attempts to bring these proposals together into a unified approach, based on the Arakne three layer model (Bouvin, 1999). The Relate and Visualise components are retained from the current AWF implementation (currently implemented AWF components shown with bold outlines), with Annotate relegated to the service layer where

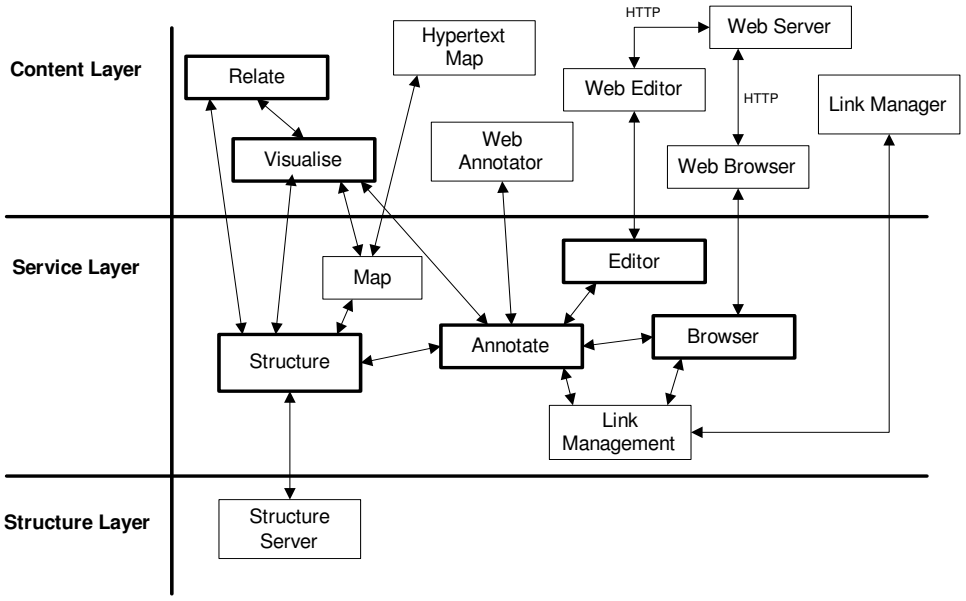


FIGURE 10.6: Proposed integration approach.

it acts as an common interface between other components and a third-party annotation tool such as Annotea. Additionally, Map, Link Management, Browser, and Editor services provide common interfaces to a third-party Hypertext Map, Link Manager, Web Browser, and Web Editor tool respectively. A number of opportunities for future integration are discussed below:

Structure Server Provides hyperstructure storage and management through the common Structure interface. For example, an integrated WebVise structure server could provide 'guided tour' and 'context' management facilities; an integrated Annotea server could provide advanced filtering facilities for controlling heavily annotated documents.

Web Annotation Provides Web-based annotation interface. Storage and retrieval of annotation structures is handled by the Annotate service, which also provides the Web Annotation component with access to a common Web Browser interface (through the Browser service). AWF-specific functionality is handled by the Annotate service: the Web Annotation tool need only provide an annotation interface to the writer. For example, an integrated Multivalent Annotations component could provide many different annotation options, including free-form 'digital ink' annotations, span annotations, and lens annotations; an integrated Hunter-Gatherer component could facilitate document-based collection building in addition to annotation.

Hypertext Map Provides map-based interface to emerging hyperstructures created with Web Annotation and Relate tools. The service-layer Map component is responsible for synchronising Hypertext Map interactions with on-screen links drawn

by Visualise component (so that links and annotations created using the Relate and Web Annotation components appear in the Hypertext Map, and changes made in the map interface are then reflected by Visualise). The Map service is also responsible for providing access to the Structure service to store and retrieve the writer's hyperstructures. For example, an integrated Visual Knowledge Builder component could provide support for emergent structuring as the writer moves away from the document-based early stages of writing and representation towards map-based structuring and reflection.

Link Manager Provides corrective link management services to the framework, when notified by the Annotate or Browser services of a 'broken' document URL or annotation location. For example, an integrated Robust Hyperlinks component could compute document signatures when annotations are created, submitting the signature to a search engine if the link breaks, redirect the Web Browser to the correct page (via the Browser service), and return the updated URL to the framework; an integrated Robust Locations component could attempt to reattach failed annotation locations (and subsequently instruct the Web Annotation service to recalculate the location specifier of the reattached annotation).

Web Server Serves Web pages to the Web Browser, including pages published through the Web Editor. For example, an integrated WebDAV server could facilitate document versioning, in order that broken annotations are not orphaned.

Web Editor Web page editing interface, abstracted through the Editor service which facilitates integration with other AWF components. The AWF framework currently integrates the Microsoft Frontpage editor, but integration with other HTML-aware editors such as Microsoft Word may be achieved.

Web Browser User interface to Web docuverse, abstracted through the Browser service which allows access to browser functions and DOM, and which also receives browser events. The AWF framework currently integrates the Internet Explorer browser, but integration with other browsers such as Netscape/Mozilla may be achieved.

10.6 Summary

This chapter has compared the Associative Writing Framework approach with the work in open hypermedia, hypertext writing, annotation, and link integrity, introduced over the course of the preceding chapters.

A number of proposals for the integration of existing approaches within the framework have been suggested, and unified into a coherent proposal for a future AWF implementation.