Experience Report: Serving *Hypermedia and the Web Online* with HA^{3}L

Timothy Miles-Board
Intelligence, Agents, Multimedia Group
University of Southampton
Southampton, UK

March 3, 2003

Contents

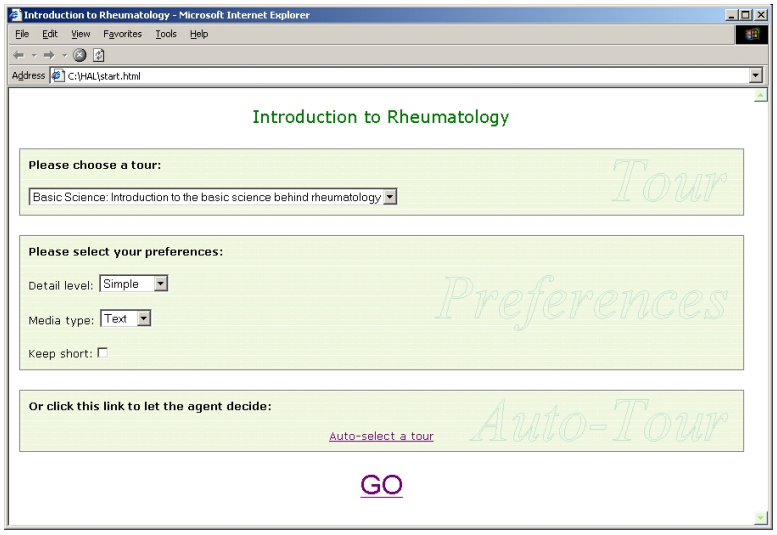


Figure 1: Choosing a tour and setting user preferences.

1 Introduction

HA^{3}L is an agent-based adaptive hypermedia system developed by (Bailey *et al.*, 2002), which uses existing technology developed by the IAM group in the form of the SoFAR agent framework (Moreau *et al.*, 2000) and the Auld Linky contextual hypermedia structure service (Michaelides *et al.*, 2001). In a nutshell, HA^{3}L provides *guided tours* of a topic which are adapted according to explicitly set user preferences of *depth*, *modality*, and *length*. Depth represents the level of detail that the user wants to see on the topic (for example, a novice user may select a high level of detail whereas an expert user may select a low level of detail). Modality represents the type of presentation the user wants to see (for example, text or video). Length is a boolean value which indicates whether or not the presented content should be 'kept short'. The content of the guided tour is augmented with generic "glossary" links leading to detailed definitions of selected terms (cf. Microcosm's *generic links* (Fountain *et al.*, 1990)). HA^{3}L also provides implicit adaptation in accordance with its user model (for example, adapting the presentation to make clear to the user concepts or links which they have already "seen"). To demonstrate the capabilities of HA^{3}L, its developer Chris Bailey has constructed a number of adaptive guided tours using information from the field of *rheumatology* (Figure).

Hypermedia and the Web (Lowe & Hall, 1999) outlines an engineering approach to the design and management of Web hypermedia applications. The authors wish to create an online resource[1] which compliments (but which does merely replicate) the content presented in the book, providing extra material and examples, and which provides adaptive presentations of its content tailored to individual visitors to the site.

[1]For the purposes of this report, this project will be referred to as the *Hypermedia and the Web Online* project.

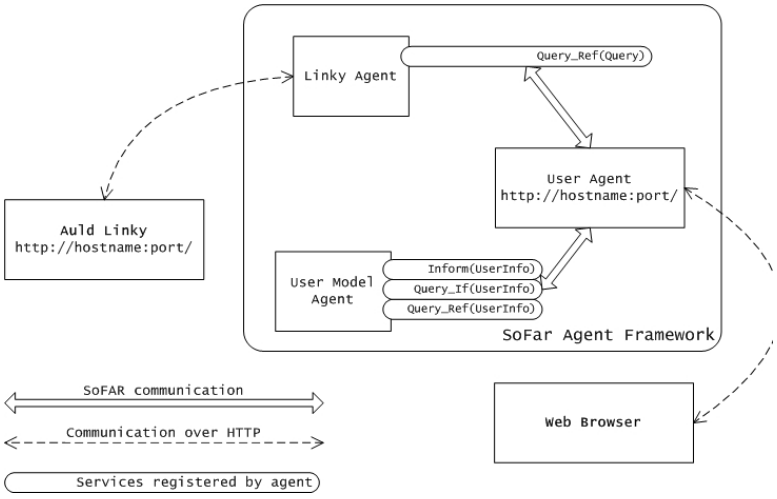


Figure 2: Basic HA^{3}L architecture.

The purpose of this report is firstly to briefly recount my experiences in getting to grips with the operation of the HA^{3}L framework (using the *Introduction to Rheumatology* example as a reference), and secondly to describe how I have applied the framework to provide adaptive tours through part of the *Hypermedia and the Web* book. The suitability of the framework for serving an online, adaptive *Hypermedia and the Web* resource is also briefly discussed.

2 Exploring HA^{3}L and the *Introduction to Rheumatology* Example

HA^{3}L runs within the SoFAR agent framework (Figure 2), and consists of three types of agent:

User Model Agent encapsulates characteristics of the user.

Linky Agent provides contextual hypermedia tour structures by acting as an intermediatry between other SoFAR agents and Auld Linky.

User Agent provides adaptive presentation to the user.

These operation of these agents is described in the following sections.

2.1 User Model Agent

The user model agent encapsulates characteristics of the user, by maintaining lists of the following items:

- The items that a user has visited on a tour (in the form of FOHM *data object* IDs, discussed later)
- The *concepts* that a user has seen (each item on a tour may have several associated concepts; a concept may be applicable to more than one item).
- The *glossary* items that a user has seen (each item on a tour may have several associated glossary links; a glossary link may be applicable to more than one item).
- The *tours* that a user has seen.
- The *level of detail* at which each tour has been seen.

2.1.1 Contract Summary

The user model agent is able to carry out three contracts within the SoFAR framework, each of which uses the UserInfo predicate (essentially a name/value string tuples to which information about the user can be arbitrarily assigned):

1. `Inform(UserInfo)` — Updates the user model according to the UserInfo predicate. For example, if the UserInfo predicate is the tuple `(visited,dataobject_id)`, the user model agent updates the list of items that the user has visited.
2. `Query_If(UserInfo)` — Queries the user model according to the UserInfo predicate. For example, if the UserInfo predicate is the tuple `(glossary,glossary_id)`, then the user model agent consults the list of glossary items that the user has seen and returns true if the user has already seen the specified item, false otherwise.
3. `Query_Ref(UserInfo)` — Gets the user model according to the UserInfo predicate. For example, if the UserInfo predicate is the tuple `(concept,)` (value unassigned), the user model agent returns an array of UserInfo objects representing all the concepts that the user has seen.

When a session ends, the user model agent serialises the user model. At the beginning of a session, the user model agent checks for the existence of a previously serialised model (loading it if available), allowing user characteristics to be persisted between sessions.

2.2 Linky Agent

The linky agent acts as an intermediatry between SoFAR agents and the Auld Linky contextual hypermedia structure service (see also Section 2.4). The linky agent carries out one contract within the SoFAR framework, with the Query predicate. The Query predicate (part of the FOHM object model (Millard *et al.*, 2000)) contains a *context* and a *structure*. The Query object is serialised as XML and sent to Auld Linky over HTTP (hostname and port of Auld Linky service configurable by an administrator). Figure 3 shows some example Query predicates (in serialised XML form).

In Figure 3a, the query matches any FOHM structures in the Auld Linky linkbase which have a *relationshiptype* attribute set to *glossary* (the `missing="variable"` pragma specifies that any attributes of the association not considered will be assumed to match the query). This query has the effect of retrieving every glossary link from the linkbase.

In Figure 3b, the query matches any FOHM structures which have a *relationshiptype* attribute set to *tour*, and a *structure* attribute set to *link*. This query has the effect of retrieving all the tours from the linkbase.

In Figure 3c, the query matches the FOHM structure with the unique id *basic_science*, and then prunes the association using the *context* information (illustrated using FOHM graphical notation (Millard *et al.*, 2000) in Figure 4). In this example, any reference, binding or data FOHM objects will be pruned if the attached context does not match the constraint imposed by the query (value of *depth* attribute must be less than or equal to 10). The tour structures are actually more complicated than that implied by Figure 4, as we shall see in Section 2.4.

2.2.1 Contract Summary

1. `Query_Ref(Query)` — Communicates the serialised Query to Auld Linky over HTTP, parses the result returned by Auld Linky (also represented in XML) into a FOHM Linkbase object, and returns the result.

```
<query>
  <association missing="variable">
    <relationtype>glossary</relationtype>
  </association>
</query>
```

(a)

```
<query>
  <association missing="variable">
    <relationtype>tour</relationtype>
    <structure>link</structure>
  </association>
</query>
```

(b)

```
<query>
  <association id="basic_science" missing="variable">
    <structure>link</structure>
  </association>
  <context>
    <contextvalue key="depth" state="variable">
      <constraint>\$\_[0]<=10</constraint>
    </contextvalue>
  </context>
</query>
```

(c)

Figure 3: Example queries handled by the Linky Agent (shown in serialised XML form)

2.3 User Agent

The user agent receives requests directly from the user (via HTTP rather than internal SoFAR communication — the hostname and port which user agent 'listens' on are configurable by an administrator), and is responsible for presenting adaptive guided tours to the user, according to their preferences and the information held by the user agent model. The user agent uses the services provided by the user model agent and linky agent to achieve this. As such, the user agent is perhaps the most complex agent (in terms of program operation) in the HA^{3}L system, and deserves a more detailed examination.

User requests are sent to the user agent over HTTP from a Web browser. The user agent responds to several requests, summarised below.

2.3.1 Lookup

Example: `http://localhost:33333/lookup?topic=basic_science&depth=10&modality=text&length=short`

Starts a guided tour, given the id of the chosen tour (*topic*) and user preferences for the tour (*depth*, *modality*, and *length*). This request is initiated by the start page (Figure). The user agent responds to this request by requesting the specified tour from Auld Linky in the given context. In the *Introduction to Rheumatology* example, *depth* is represented as the numerical value 10 (*Simple*), 20 (*Extended*), or 50 (*Detailed*). If the user selects the *Simple* depth, the guided tour is pruned of any structure which has an associated context value greater than 10. If the user selects the *Extended* depth, the full guided tour is returned (no pruning takes place). The HA^{3}L system therefore provides adaptive tours by *selectively hiding content* according to user preferences.

The user agent constructs a FOHM Query object which encapsulates the selected tour and depth (Figure 3c shows the serialised XML form of the FOHM Query object constructed in response to the example URL shown above). For each agent in the framework able to undertake the contract `Query_Ref(Query)` (in the *Introduction to Rheumatology* example, a linky agent undertakes the con-

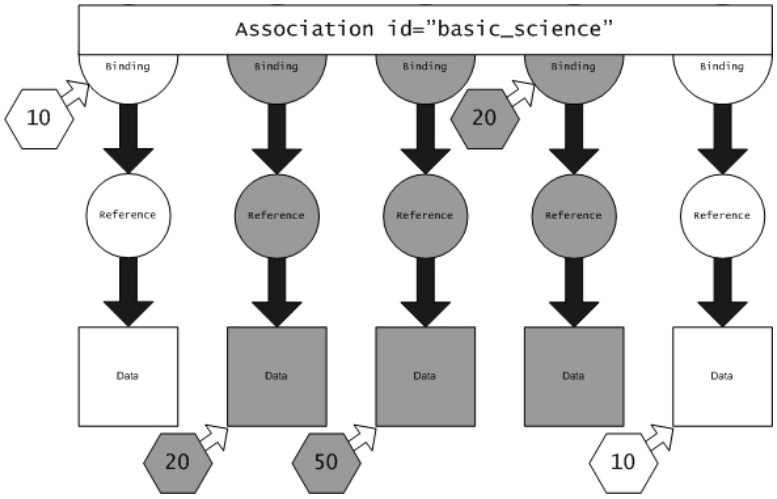


Figure 4: Pruning a FOHM association using context constraints in Query object (Figure 3c). The grey *depth* contexts have failed to match the Query context (*depth*<=10), and are therefore pruned from the association.

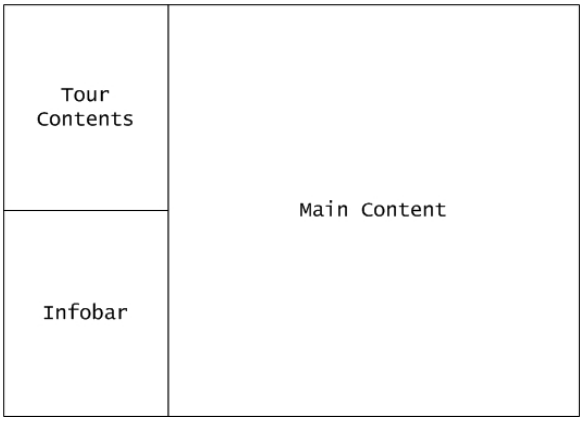


Figure 5: Page layout generated by lookup request.

tract), the user agent receives a FOHM linkbase describing the contextual tour. Provided that a contextual tour could be obtained which matched the user preferences, the user agent builds the start page of the tour, consisting of a page divided into 3 frames (Figures 5 and 6). Each frame invokes a separate request to the user agent for its content, described below.

Tour Contents — Table of contents for the chosen tour (see Section 2.3.2).

Main Content — Current page of tour (see Section 2.3.3).

Infobar — Summary of relevant concepts and glossary links for the currently displayed page of the tour (see Section 2.3.4).

The user agent also informs any agents able to undertake the contract `Inform(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) of the selected tour id and depth.

2.3.2 Table of Contents

Example: `http://localhost:33333/toc`

Builds a table of contents for the current tour using the cached result from a lookup (see Section 2.3.1). Figure 6 shows the the table of contents generated for simple, extended, and detailed tours of the *basic_science* topic.

2.3.3 Main Content

Example: `http://localhost:33333/basic_science/immune_response`

Presents the specified page of the current tour (in the example URL, the page is represented by the FOHM data object with the id `basic_science/immune_response`), with links to the next and previous pages in the tour, links to more detailed information (where appropriate), and inline links to glossary terms (also summarised in the Infobar, see Section 2.3.4).

The user agent queries each agent in the framework able to undertake the contract `Query_If(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) to determine whether the user has previously visited the specified page, and to determine the glossary links that the user has previously followed.

The user agent also informs any agents able to undertake the contract `Inform(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) that the previous page in the browsing history (where applicable) has now been seen by the user.

If the user *length* preference is set (this preference is currently maintained by the user agent for the duration of a session, rather than managed by agents such as the user model agent, and is set in the static user preferences page shown in Figure), only the first paragraph of the content is displayed, with a link which expands the text to reveal any further content (Figure 7).

If the user *modality* preference is set (this preference is currently maintained by the user agent for the duration of a session, rather than managed by agents such as the user model agent, and is set in the static user preferences page shown in Figure), and the selected media representation is available for the selected page, the content will be presented according to the user preference (Figure 8)[2].

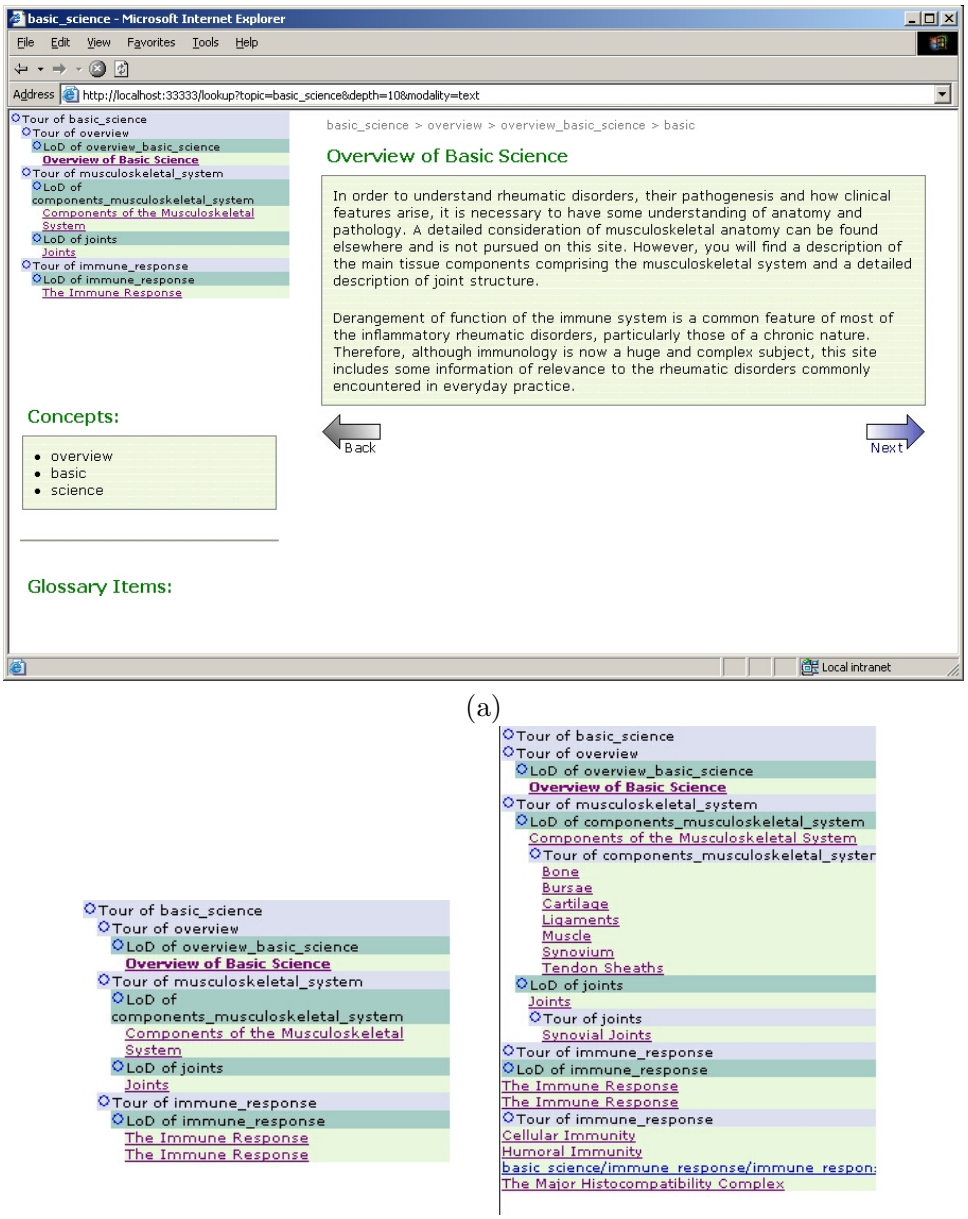
The user agent builds the content page by presenting the title and associated content of the specified data object (using appropriate presentation media). If the presentation medium is text, for each specified glossary item for the data object (see Section 2.4.2), the user agent replaces the first instance of the glossary term in the content with a link to the glossary definition. If the user has previously followed the glossary link, the link is presented in black (rather than the default blue) text to visually distinguish between visited and unvisited links (in effect *hiding* previously visited links). In all cases, the user agent also creates links to the previous and next pages on the tour, relative to the specified page, and also a link to more detailed information where the user depth preference has not caused more detailed pages to be pruned from the tour (see Section 2.3.1).

2.3.4 Infobar

Example: `http://localhost:33333/infobar/basic_science/overview`

Summarises the relevant concepts and glossary links for the currently displayed page (which may be presented using different media according to the user's modality preference) of the tour (in the example URL, the currently displayed page is represented by the FOHM data object with id `basic_science/overview`). Figure 9 illustrates the Infobar.

[2]Note that both modalities are available from the table of contents for the tour, but if the user follows the *next* links through the tour, the user modality preference will be used to determine the presentation medium of the next item in the tour (provided alternative presentation mediums are available).

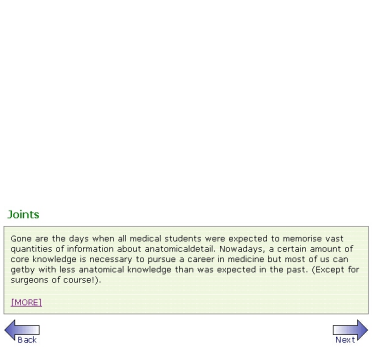


(a)

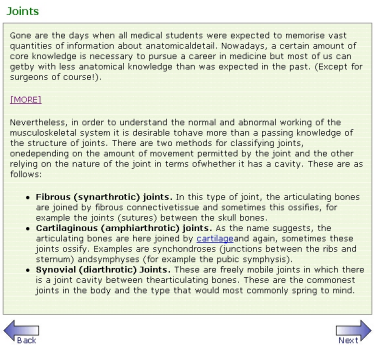
(b)

(c)

Figure 6: Start screen for simple tour (a) with detail of table of contents for extended (b) and detailed (c) tours.

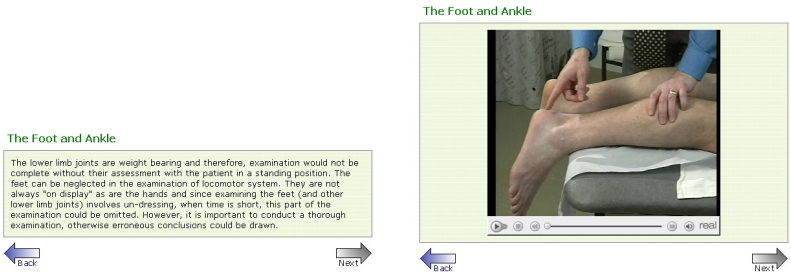


(a)



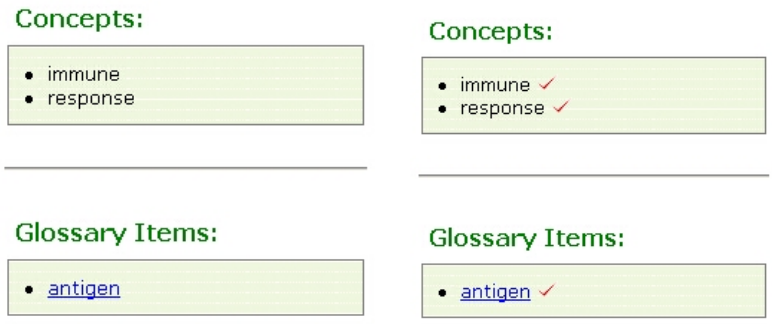
(b)

Figure 7: Main content shown with user length preference set (a). Following the *MORE* link reveals further content (b).



(a) (b)

Figure 8: Main content presented according to user modality preference of text (a) and video (b).



(a) (b)

Figure 9: Infobar presenting new concepts/glossary links (a) and concepts/glossary links that have been seen by the user before (b).

For each concept, the user agent queries each agent in the framework able to undertake the contract `Query_If(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) to determine whether the user has previously seen a page relating to this concept. If this is the case, the concept is presented in the Infobar with a "tick" in order to visually inform the user that the concept has been seen before (Figure 2.3.4b). If the concept has not been seen before, the user agent informs any agents able to undertake the contract `Inform(UserInfo)` that the user has now seen the concept. A similar process is applied to the presentation of the glossary links (Figure 9b).

2.3.5 Glossary Page

Example: `http://localhost:33333/glossaryitem/antigen`

Builds a glossary page for the specified glossary term (in the example URL, the glossary term is *antigen*). Figure 10 shows an example glossary page for the term *antigen*. Glossary pages are displayed in popup windows when the user clicks a glossary link from the Infobar or inline glossary link from the main content.

The user agent informs any agents able to undertake the contract `Inform(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) that the user has now seen the definition of the glossary term.

2.3.6 Letting the User Agent Choose a Tour

Example: `http://localhost:33333/choosetour`

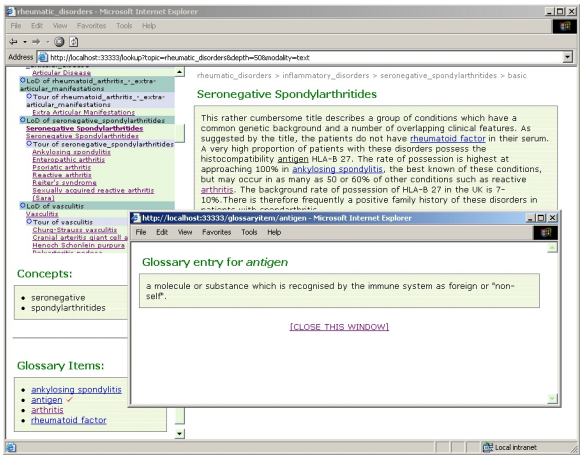


Figure 10: Glossary definition.

Similar to the *lookup* request, but the user agent chooses the tour, based on the user's previous HA^{3}L interactions.

The user agent queries each agent in the framework able to undertake the contract `Query_Ref(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) to receive a list of the guided tours (in the form of tour IDs) which the user has previously taken (in this or previous sessions) and the depth at which the user chose to view each tour.

The user agent then chooses at random a tour that the user has not seen before (if all the tours have been seen before, the user agent picks one at random). The user agent determines the depth at which to present the chosen tour by finding the most popular depth in previous tours (for example, if the user has previously seen 2 tours at *Simple* depth, 1 tour at *Extended* depth, and 1 tour at *Detailed* depth, the user agent would present the chosen tour at *Simple* depth). If the depth cannot be chosen using this method (for example, the user has not seen any tours), the depth is chosen at random.

The *length* and *modality* preferences retain their default values (length preference unset, modality preference set to "text").

The user agent then proceeds as with a *lookup* request (see Section 2.3.1), constructing a FOHM Query object which encapsulates the selected tour and depth. For each agent in the framework able to undertake the contract `Query_Ref(Query)` (in the *Introduction to Rheumatology* example, a linky agent undertakes the contract), the user agent receives a FOHM linkbase describing the contextual tour. Provided that a contextual tour could be obtained which matched the user preferences assigned by the user agent, the start page of the tour is built.

The user agent also informs any agents able to undertake the contract `Inform(UserInfo)` (in the *Introduction to Rheumatology* example, a user model agent undertakes the contract) of the assigned tour id and depth.

2.4 FOHM Structures

In the *Introduction to Rheumatology* example, data for the guided tours presented by HA^{3}L is stored in three separate linkbases (each linkbase describing FOHM structures using an XML representation):

1. `glossary.xml` — decribes the generic glossary links that HA^{3}L uses to augment the information presented in the guided tours.
2. `data.xml` — describes the data content of the tours (equivalent to describing the content of

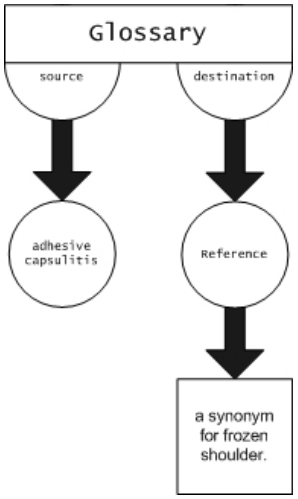


Figure 11: FOHM glossary link structure.

each Web page that the user sees on a guided tour), including the glossary links relevant to the content, the concepts covered in the content, and the context of the content (in terms of *depth* and *modality*).

3. `structures.xml` — connects the data objects defined in `data.xml` as a series of FOHM association structures representing guided tours through the topic.

2.4.1 Glossary Links

Appendix A shows how a glossary link is represented in XML as a FOHM structure (Figure 11 illustrates the same link using the FOHM graphical notation).

2.4.2 Guided Tours

In the FOHM representation, guided tours are made up from *Tours* (FOHM associations representing a set of objects designed to be viewed in sequence), *Levels of Detail (LoDs)* (FOHM associations associating multiple representations of the same object, ordered within the association from the simplest representation to the most complex), *Concepts* (FOHM associations which collect together multiple objects that represent the same conceptual entity, for example text and video presentations of the same information) and *Data* (FOHM data objects representing actual content for the tour). See (Bailey *et al.*, 2002) for further explanation of these terms.

Appendix B shows how a data object is represented in XML. The content the user sees (provided their preferences match the context specified in the *context* attribute) is stored in the *datacontent* attribute. The *behaviour* attributes are used to list *concepts* and *glossary links* appropriate to the data content.

Appendix C shows how a guided tour is represented in XML as a FOHM structure, illustrated using FOHM graphical notation by Figure 12.

3 Serving *Hypermedia and the Web Online* with HA^{3}L

This section describes how HA^{3}L was used to provide an adaptive tour of Chapter 7 ("Development Process") of the *Hypermedia and the Web* book, a chapter that seemed to lend itself particularly well

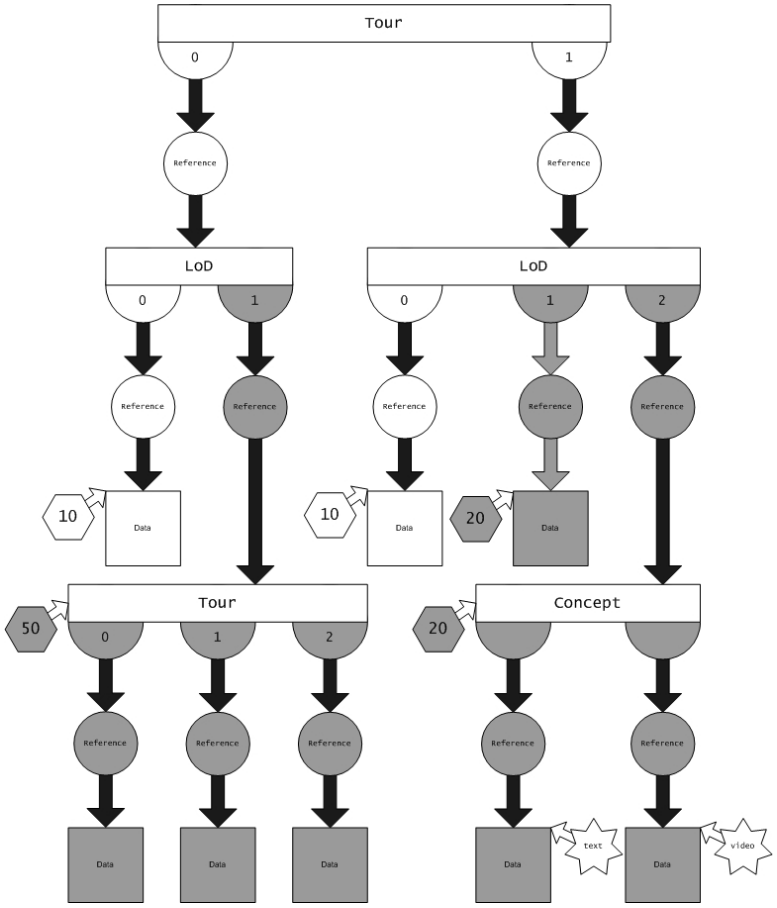


Figure 12: FOHM tour structure. The structure has been pruned using the Query context constraint (Figure 3c). The grey *depth* contexts have failed to match the Query context (*depth*<=10), and are therefore pruned from the tour.

to the tour format as it guides the reader through the steps involved in the hypermedia development process. In this small case study, the content of the book was largely used in an unedited form[3], as the purpose was merely to evaluate the process and suitability of serving content from *Hypermedia and the Web* using HA^{3}L rather than focus on elaborate content editing. I describe here the changes made to the HA^{3}L code to make the adaptive tours more suitable to the presentation of the *Hypermedia and the Web* content, and the process by which I constructed the FOHM representations of the tour structure and glossary.

3.1 Tailoring representation of User Knowledge Level

In the *Introduction to Rheumatology* example, *depth* is represented as the numerical value 10 (*Simple*), 20 (*Extended*), or 50 (*Detailed*). If the user selects the *Simple* depth, the guided tour is pruned of any structure which has an associated context value greater than 10. If the user selects the *Extended* depth, the full guided tour is returned (no pruning takes place). In this example, therefore, the HA^{3}L system provides adaptive tours by *selectively hiding content* according to user preferences.

In the *Hypermedia and the Web Online* experiment, I wanted depth to be explicitly tied to user knowledge level ("I have only a basic knowledge of hypermedia") rather than a statement of preference ("Show me only a simple level of detail") — this seemed a more natural way for the user to indicate their knowledge of a topic. I also wanted to introduce knowledge about specific topics within the tour. The Hypermedia and the Web book brings together information from a variety of disciplines, such as hypermedia, software engineering, design, and management, and may also be used by many different people (students, lecturers, hypermedia designers). To better model user knowledge of these often diverse disciplines, I extended the HA^{3}L code to allow users to indicate their knowledge of more than one topic (in the following examples, the user can indicate their level of knowledge of both hypermedia and software engineering). The *depth* value used by HA^{3}L now encodes several knowledge values as a series of key/value pairs (although it is still stored as a single value in the user model agent).

For example, in Figure 13, the user has indicated that their knowledge of both hypermedia and software engineering is *Extensive*, which corresponds to a *depth* preference of `hm_depth=10&se_depth=10` being passed to the user agent (Section 2.3.1). The encoded depth preferences correspond to a *Simple* tour in *Introduction to Rheumatology* example (*depth=10*) — the user has extensive knowledge of the topic and therefore only needs to see a basic level of detail. The 'keep short' preference of the *Introduction to Rheumatology* example has been reworded to form a further device for the indication of the user knowledge (but is still a boolean option, and remains otherwise unchanged) — "I have read the book".

3.2 Building FOHM Structures

The process of building FOHM structures describing a tour through Chapter 7 of the *Hypermedia and the Web* book (available in HTML format) involved the following steps:

1. Convert *Hypermedia and the Web* HTML pages to skeleton FOHM data objects.
2. Edit skeleton data objects to define concepts, glossary, and context attributes.
3. Create tour structure.
4. Assign depth values to data objects.
5. Create FOHM glossary linkbase.

[3]It should be noted that one of the stipulations of the *Hypermedia and the Web Online* project is that users should not be able to reproduce the content structure of the paper book from the online version.

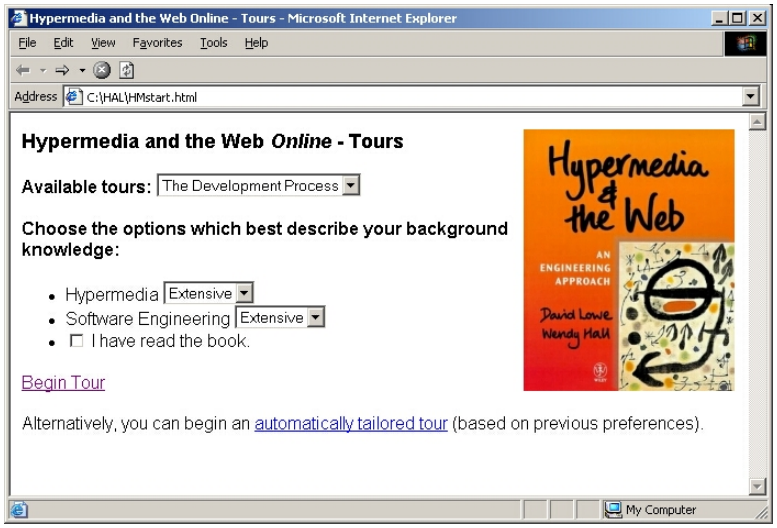


Figure 13: Choosing a *Hypermedia and the Web* tour and indicating current knowledge (cf. Figure).

3.2.1 Converting HTML pages to skeleton FOHM data objects

The first step was to convert the collection of HTML pages making up Chapter 7 into a set of skeleton FOHM data objects (using an XML representation), each data object encapsulating information on a particular topic within the chapter and representing a single page in the guided tour. The chapter was split into 7 HTML files, corresponding to each of the major sections of the chapter:

1. Key issues and goals of the chapter
2. Introduction
3. Understanding the project
4. Selecting a product model
5. Selecting a process model
6. Project planning and management
7. Framework for documentation

For each HTML file, the `html2fohmdata` script (Appendix F) extracted each section/subsection heading and its content and created a skeleton FOHM data object. For example, the *Selecting a Product Model* section contained the following subsections:

1. What is a product model?
   (a) Programming language based model
   (b) Screen based model
   (c) Information centered model
2. Selecting and adapting a product model
3. Example

```
tour foo
0 foo/bar
1 lod foo/baz
        0 foo/baz/a
        1 tour foo/baz/b
                0 foo/baz/x
                1 foo/baz/y
                2 foo/baz/z
        1 endtour
1 endlod
endtour
```

`tour foo`	start a FOHM tour structure with id *foo*. This structure represents the complete tour.
`n foo/bar`	bind a reference to the FOHM data object *foo/bar* at position n in the parent structure.
`n lod foo/baz`	bind a FOHM level of detail structure with id *foo/baz* at position n in the parent structure.
`n endlod`	end the level of detail structure bound at position n in the parent structure.
`n tour foo/baz/b`	bind a FOHM tour structure with id *foo/baz/b* at position n in the parent structure.
`n endtour`	end the tour structure bound at position n in the parent structure.
`endtour`	end the tour structure.

Figure 14: Simple notation for representing tour structure.

The skeleton data object for the *Information centered model* generated by the script is shown in Appendix D.

Script usage:

```
html2fohmdata < ch7.1.html > data.xml
```

3.2.2 Editing skeleton data objects

The next step involved manually editing the skeleton data objects. This included some minor editing of the content (removing references to other sections, examples and appendices in the book, splitting data objects with large content into two or more data objects, and summarising) and identifying suitable glossary terms and concepts relevant to the content of the data object. Appendices D and E show the same data object before and after editing.

3.2.3 Creating tour structure

The next step was to identify how the data objects could be presented within an adaptive tour structure. This involved identifying potential FOHM *Tours* and *Levels of Detail* within the chapter. The tour was described using a simple notation (Figure 3.2.3) in order to facilitate faster editing and rearranging of the tour than would be possible using the complex FOHM representation (the basis for this structure was obtained by extracting the id of each data object created in the previous stages). The `tour2fohmstruct` script (Appendix G) converted this notation into an XML representation of a FOHM tour structure.

Script Usage:

```
perl -n -e "print \"$1\n\" if /<data id="(.*?)">/;" < data.xml > tour
```

```
tour2fohmstruct < tour > tour.xml
```

3.2.4 Assigning depth values

After creating the tour structure for the chapter, "depth" values were manually assigned to each data object according to the detail of the content from the perspectives of hypermedia and software engineering. Structures such as subtours and levels of detail created in the previous stage helped identify which data objects could be adaptively pruned from the tour according the the user preferences, and which data objects were essential to the structure of the tour. As with the *Introduction to Rheumatology* example, values for *hm_depth* (hypermedia depth) and *se_depth* (software engineering depth) were either 10 (*Basic*), 20 (*Extended*), or 50 (*Detailed*). Appendix E shows a data object with manually assigned depth values.

3.2.5 Creating glossary linkbase

The `fohmdata2fohmglossary` script (Appendix H) extracted each glossary term from the XML representation of the FOHM data objects, and generated a corresponding FOHM glossary link structure (in XML), similar to that shown in Appendix A, for the term. Short definitions for each glossary term were then manually added.

Script usage:

```
fohmdata2fohmglossary < data.xml > glossary.xml
```

4 Discussion and Conclusion

This report has recounted my experiences in getting to grips with the operation of the HA^{3}L framework (using the *Introduction to Rheumatology* example). I have described how this understanding has lead to the successful demonstration of how *Hypermedia and the Web Online* content can be adapted and served using HA^{3}L. Figure 4 illustrates how a guided tour through the *Development Process* of the book is adapted according to user knowledge of hypermedia and software engineering. Figure 4 shows the *Hypermedia and the Web Online* example in action. The remainder of this section discusses the suitability of the HA^{3}L framework for serving an online, adaptive *Hypermedia and the Web* resource.

Scalability HA^{3}L currently only supports a single user interaction; however I estimate that changes required to extend the code to include multiple user support are minimal. A user model agent would be required for each user of the system, and other agents would have to locate the user agent specific to the current user request before updating the user model. User model agents may have to be spawned within the SOFAR framework in response to new user connections. Users may have to "log in" to the site in order to view personalised content.

Reponsiveness The responsiveness of HA^{3}L in both the *Introduction to Rheumatology* and *Hypermedia and the Web Online* examples was found to be slow. This slow reponse can only worsen when we consider a distributed agent platform and multiple users, and may be an artifact of the SOFAR framework rather than a specific problem with HA^{3}L's design.

Reliability During experimenting with the *Introduction to Rheumatology* example and construction of the *Hypermedia and the Web Online* example, HA^{3}L was noted to 'lock up' or crash infrequently, subsequently leading to loss of user model data.

Adaptive Features HA^{3}L is solely designed to provide adaptive guided tours through content, and as such does not provide adaptive support to users browsing an unrestricted hypertext.

User Modelling HA^{3}L's simple user modelling features, although effective in the *Introduction to Rheumatology* example, proved to be limited in serving adaptive *Hypermedia and the Web Online* content, requiring extensions as outlined in Section 3.1. There is a worry that further (non-trivial) extensions will be required to support the complex set of interactions desired for the *Hypermedia and the Web Online* resource, but these requirements need to be formalised and discussed in detail first.

FOHM Structure Construction The construction of FOHM tours using the XML representation is a particularly arduous and complicated task, and for large content bases may incur a significantly large effort. Investigation of 'intermediate representations' (such as that described in Section 3.2.3) and better tools for automatic construction are badly needed.

Despite these drawbacks, the adaptive approach which HA^{3}L embodies certainly seems promising, indeed it is feasible that the services offered by a *Hypermedia and the Web Online* resource could include a number of themed guided tours through a network of content. The next step then is to carry out a user-based evaluation of the HA^{3}L approach in order to discover whether its presentation of content is useful to readers.

References

Bailey, C, Hall, W, Millard, D E, & Weal, M. 2002. Towards open adaptive hypermedia. *Pages 36–46 of: Proceedings of the second international conference on adaptive hypermedia and adaptive web-based systems, malaga, spain.*

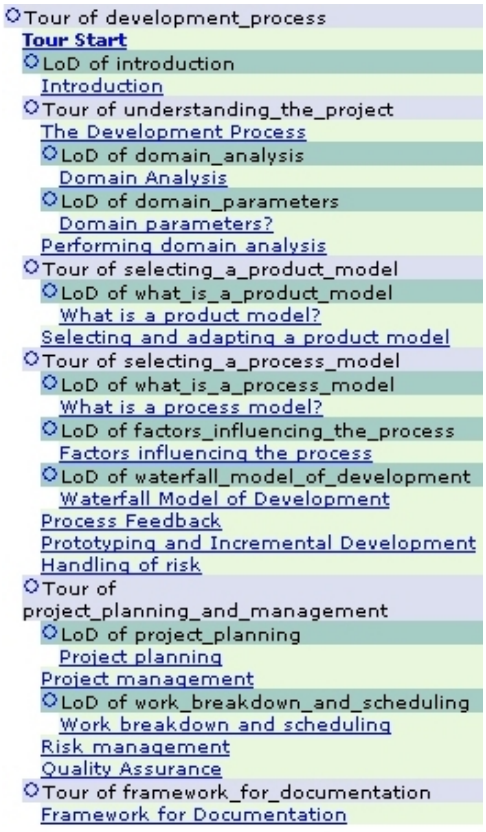
Fountain, A., Hall, W., Heath, I., & Davis, H. C. 1990 (Nov.). Microcosm: An Open Model for Hypermedia With Dynamic Linking. *Pages 298–311 of: Proceedings of the hypertext '90 conference, inria, france.*

Lowe, David, & Hall, Wendy. 1999. *Hypermedia and the web: An engineering approach.* Wiley.

Michaelides, Danius, Millard, David, Weal, Mark, & Roure, David De. 2001. Auld linky: A contextual open hypermedia link server. *In: Proceedings of the acm hypertext 2001 conference, arhus, denmark.*

Millard, David E., Moreau, Luc, Davis, Hugh C., & Reich, Siegfried. 2000. FOHM: A fundamental open hypertext model for investigating interoperability between hypertext domains. *In: Proceedings of the '00 acm conference on hypertext, may 30 - june 3, san antonio, tx.*

Moreau, Luc, Gibbins, Nick, DeRoure, David, El-Beltagy, Samhaa, Hall, Wendy, Hughes, Gareth, Joyce, Dan, Kim, Sanghee, Michaelides, Danius, Millard, Dave, Reich, Sigi, Tansley, Robert, & Weal, Mark. 2000 (Apr.). SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. *Pages 369–388 of: The fifth international conference and exhibition on the practical application of intelligent agents and multi-agents.*



(a)

(b)

Figure 15: Tours presented to users with extensive (a) and basic (b) hypermedia and software engineering backgrounds

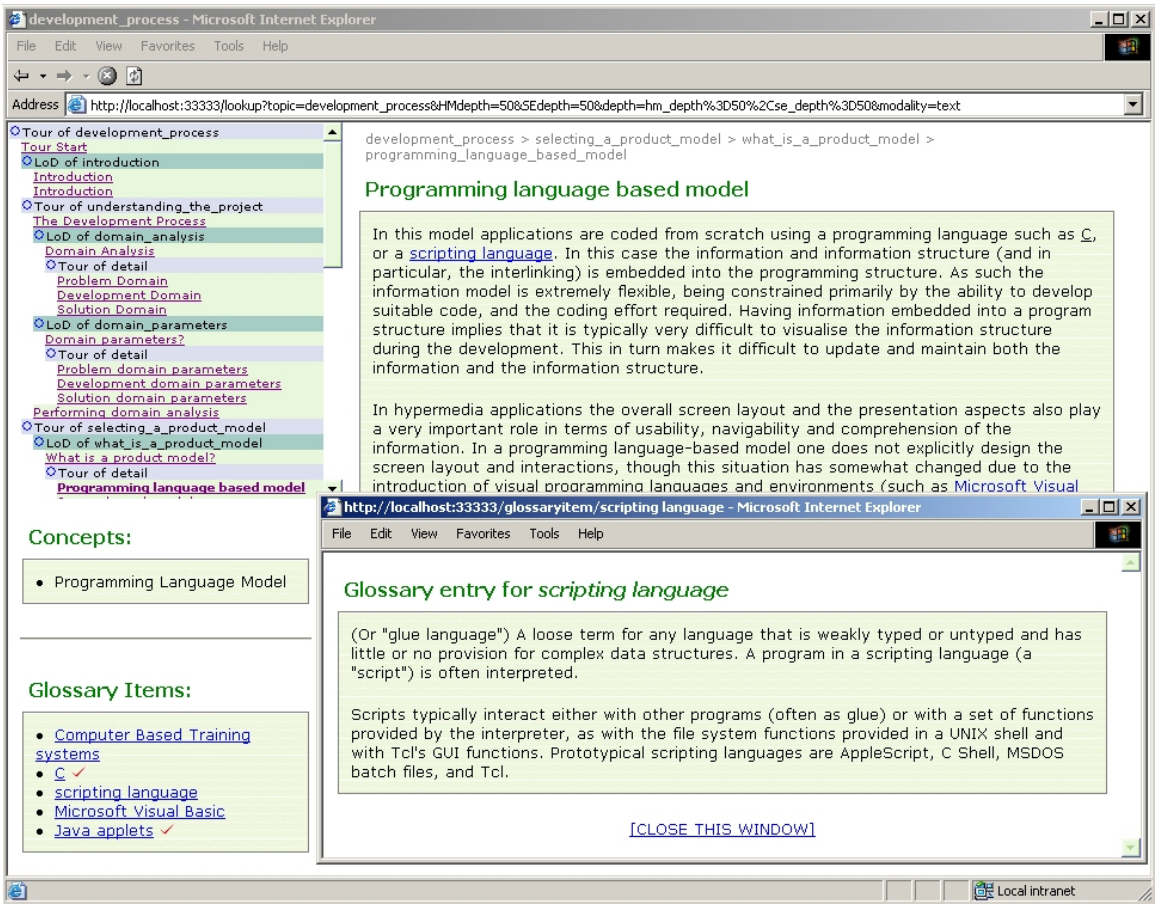


Figure 16: Following the *Development Process* tour.

A FOHM Glossary Link Structure (*Introduction to Rheumatology* example)

```
<association id="glossary/adhesive capsulitis">
  <relationtype>glossary</relationtype>
  <description>Glossary entry for adhesive capsulitis</description>
  <structure>link</structure>
  <feature>direction</feature>
  <binding>
    <reference missing="variable">
      <locspec>
        <regioncontent>adhesive capsulitis</regioncontent>
      </locspec>
    </reference>
    <featurevalue feature="direction">source</featurevalue>
  </binding>
  <binding>
    <reference>
      <data>
        <datacontent>A synonym for frozen shoulder.</datacontent>
      </data>
    </reference>
    <featurevalue feature="direction">destination</featurevalue>
  </binding>
</association>
```

B FOHM Data Object Structure (*Introduction to Rheumatology* example)

```
<data id="approach_to_patient/technique/overview">
  <behaviour>
    <event>display</event>
    <behaviourvalue key="title">Technique</behaviourvalue>
    <behaviourvalue key="concept0">technique</behaviourvalue>
    <behaviourvalue key="concept0">examination</behaviourvalue>
    <behaviourvalue key="glossary0">musculoskeletal system</behaviourvalue>
  </behaviour>
  <context>
    <contextvalue key="depth">10</contextvalue>
  </context>
  <datacontent>In order to carry out an examination of the musculoskeletal system quickly
  and efficiently and with least disturbance to the patient, it is necessary that both patient
  and examiner are relaxed and comfortable.  As with other aspects of physical examination,
  it is important not to inflict unnecessary pain or discomfort on the patient.  You must always
  be alert to the possibility that whatever you are doing may cause discomfort to the patient.
  This may be achieved partly by ensuring that, instead of focusing your attention on the part
  that you are examining, you observe the patient's facial expression during the examination.
  This will tell you whether what you are doing is acceptable or not.</datacontent>
</data>
```

C FOHM Tour Structure (*Introduction to Rheumatology* example)

```
<association id="approach_to_patient">
  <relationtype>tour</relationtype>
  <feature>position</feature>

  <binding>
    <reference>
      <association id="approach_to_patient/technique">
        <relationtype>lod</relationtype>
        <structure>link</structure>
        <feature>position</feature>

        <binding>
```

```
        <reference>
          <data id="approach_to_patient/technique/overview" state="id"/>
        </reference>
        <featurevalue feature="position">0</featurevalue>
      </binding>

      <binding>
        <reference>
          <association id="approach_to_patient/technique/detailed">
            <relationtype>tour</relationtype>
            <structure>link</structure>
            <feature>position</feature>
            <context>
              <contextvalue key="depth">50</contextvalue>
            </context>

            <binding>
              <reference>
                <data id="approach_to_patient/technique/detailed/feel" state="id"/>
              </reference>
              <featurevalue feature="position">0</featurevalue>
            </binding>

            <binding>
              <reference>
                <data id="approach_to_patient/technique/detailed/look" state="id"/>
              </reference>
              <featurevalue feature="position">1</featurevalue>
            </binding>

            <binding>
              <reference>
                <data id="approach_to_patient/technique/detailed/move" state="id"/>
              </reference>
              <featurevalue feature="position">2</featurevalue>
            </binding>

          </association>
        </reference>
        <featurevalue feature="position">1</featurevalue>
      </binding>

    </association>
  </reference>
  <featurevalue feature="position">0</featurevalue>
</binding>

<binding>
  <reference>
    <association id="approach_to_patient/examination">
      <relationtype>lod</relationtype>
      <structure>link</structure>
      <feature>position</feature>

      <binding>
        <reference>
          <data id="approach_to_patient/examination/overview" state="id"/>
        </reference>
        <featurevalue feature="position">0</featurevalue>
      </binding>

      <binding>
        <reference>
          <data id="approach_to_patient/examination/hand" state="id"/>
        </reference>
        <featurevalue feature="position">1</featurevalue>
      </binding>

      <binding>
        <reference>
          <association id="approach_to_patient/examination/ankle">
            <relationtype>concept</relationtype>
            <structure>link</structure>
            <feature>media</feature>
            <context>
```

```
              <contextvalue key="depth">20</contextvalue>
            </context>

            <binding>
              <reference>
                <data id="approach_to_patient/examination/ankle/text" state="id"/>
              </reference>
              <featurevalue feature="media">text</featurevalue>
            </binding>

            <binding>
              <reference>
                <data id="approach_to_patient/examination/ankle/video" state="id"/>
              </reference>
              <featurevalue feature="media">video</featurevalue>
            </binding>

          </association>
        </reference>
        <featurevalue feature="position">2</featurevalue>
      </binding>

    </association>
  </reference>
  <featurevalue feature="position">1</featurevalue>
</binding>

</association>
```

D FOHM Data Object skeleton generated from *Development Process* Chapter

```
<data id="development_process/selecting_a_product_model/what_is_a_product_model/information_centered_model">
      <behaviour>
            <event>display</event>
            <behaviourvalue key="title">Information centred model</behaviourvalue>
            <behaviourvalue key="glossary0"></behaviourvalue>
            <behaviourvalue key="concept0"></behaviourvalue>
      </behaviour>
      <context>
            <contextvalue key="hm_depth"></contextvalue>
            <contextvalue key="se_depth"></contextvalue>
      </context>
      <datacontent><![CDATA[
            <P>In this model the information and its structure...
      ]]></datacontent>

</data>
```

E FOHM Data Object after manual editing

```
<data id="development_process/selecting_a_product_model/what_is_a_product_model/information_centered_model">
      <behaviour>
            <event>display</event>
            <behaviourvalue key="title">Information centred model</behaviourvalue>
            <behaviourvalue key="glossary0">Hypertext Markup Language</behaviourvalue>
            <behaviourvalue key="glossary1">SGML</behaviourvalue>
            <behaviourvalue key="glossary2">Multimedia Viewer</behaviourvalue>
            <behaviourvalue key="concept0">Information Model</behaviourvalue>
      </behaviour>
      <context>
            <contextvalue key="hm_depth">10</contextvalue>
            <contextvalue key="se_depth">50</contextvalue>
      </context>
      <datacontent><![CDATA[
            <P>In this model the information and its structure...
      ]]></datacontent>
```

```
</data>
```

F html2fohmdata listing

```
while (<STDIN>) {

        if (/<H3>(<A NAME='.*?'><\/A>)?(.*?)<\/H3>/i) {
                $h3 = $2;
                next;
        }
        if (/<H4>(<A NAME='.*?'><\/A>)?(.*?)<\/H4>/i) {
                doContent($id,$title); # data object for previous section
                $h4 = $2;
                $id = "$h3/$h4";
                $title = $h4;
                next;
        }
        if (/<H5>(<A NAME='.*?'><\/A>)?(.*?)<\/H5>/i) {
                doContent($id,$title); # data object for previous section
                $h5 = $2;
                $id = "$h3/$h4/$h5";
                $title = $h5;
                next;
        }
        if (/<H6>(<A NAME='.*?'><\/A>)?(.*?)<\/H6>/i) {
                doContent($id,$title); # data object for previous section
                $id = "$h3/$h4/$h5/$1";
                $title = $1;
                next;
        }

        $datacontent .= $_;
}
doContent($id,$title);

sub doContent {
        my ($id,$title) = @_;
        next if $datacontent eq "";
        $id =~ s/ /_/g;
        $id = lc($id);
        print <<END;
<data id="development_process/$id">
        <behaviour>
                <event>display</event>
                <behaviourvalue key="title">$title</behaviourvalue>
                <behaviourvalue key="glossary0"></behaviourvalue>
                <behaviourvalue key="concept0"></behaviourvalue>
        </behaviour>
        <context>
                <contextvalue key="hm_depth">10</contextvalue>
                <contextvalue key="se_depth">10</contextvalue>
        </context>
        <datacontent>
                $datacontent
        </datacontent>
</data>

END
        $datacontent = "";
}
```

G tour2fohmstruct listing

```
print "<linkbase>\n";
while (<STDIN>) {
        chomp;
        if (/^(\t*)tour (.*?)$/) { # start of main tour
```

```
                print<<END;
$1<association id="$2">
$1      <relationtype>tour</relationtype>
$1      <structure>link</structure>
$1      <feature>position</feature>
END
                next;
        }
        if (/^(\t*)endtour$/) { # end of main tour
                print<<END;
$1</association>
END
                next;
        }
        if (/^(\t*)[0-9]+ lod (.*?)$/) { # start of lod
                print<<END;
$1<binding>
$1      <reference>
$1              <association id="$2">
$1              <relationtype>lod</relationtype>
$1              <structure>link</structure>
$1              <feature>position</feature>
END
                next;
        }
        if (/^(\t*)([0-9]+) endlod$/) { # end of lod
                print<<END;
$1              </association>
$1      </reference>
$1      <featurevalue feature="position">$2</featurevalue>
$1</binding>
END
                next;
        }
        if (/^(\t*)[0-9]+ tour (.*?) \[(.*?)\]$/) { # start of subtour (with context)
                print<<END;
$1<binding>
$1      <reference>
$1              <association id="$2">
$1                      <relationtype>tour</relationtype>
$1                      <structure>link</structure>
$1                      <feature>position</feature>
$1                      <context>
END
                foreach $context (split(',',$3)) {
                        @c = split '=', $context;
                        if (scalar(@c) == 2) {
                                print<<END;
$1                              <contextvalue key="$c[0]">$c[1]</contextvalue>
END
                        }
                }
                print <<END;
$1                      </context>
END
                next;
        }
        if (/^(\t*)[0-9]+ tour (.*?)$/) { # start of subtour
                print<<END;
$1<binding>
$1      <reference>
$1              <association id="$2">
$1                      <relationtype>tour</relationtype>
$1                      <structure>link</structure>
$1                      <feature>position</feature>
END
                next;
        }
        if (/^(\t*)([0-9]+) endtour$/) { # end of subtour
                print<<END;
$1              </association>
$1      </reference>
$1      <featurevalue feature="position">$2</featurevalue>
$1</binding>
END
```

```
                next;
        }
        if (/^(\t*)([0-9]+) (.*?)$/) { # data reference
                print<<END;
$1<binding>
$1      <reference>
$1              <data id="$3" state="id"/>
$1      </reference>
$1      <featurevalue feature="position">$2</featurevalue>
$1</binding>
END
                next;
        }
}
print "</linkbase>\n";
```

H fohmdata2fohmglossary listing

```
while (<>) {
        if (/<behaviourvalue key="glossary[0-9]+">(.*?)<\/behaviourvalue>/i) {
                $glossary_items{$1}++;
        }
}

print "<linkbase>\n";

foreach $term (keys %glossary_items) {
        print <<END;
<association id="glossary/$term">
        <relationtype>glossary</relationtype>
        <structure>link</structure>
        <feature>direction</feature>
        <binding>
                <reference missing="variable">
                        <locspec>
                                <regioncontent>$term</regioncontent>
                        </locspec>
                </reference>
                <featurevalue feature="direction">source</featurevalue>
        </binding>
        <binding>
                <reference>
                        <data>
                                <datacontent><![CDATA[Definition of $term.]]></datacontent>
                        </data>
                </reference>
                <featurevalue feature="direction">destination</featurevalue>
        </binding>
</association>

END
}
print "</linkbase>\n";
```