Chapter Goals

Chapter 8 looked at the stages of development and how they are combined. In this chapter, we go deeper into the practicalities of how the development might be carried out. Although we cannot provide a cohesive methodology (it will be different for every project, and the field is still not mature enough to be able to be so prescriptive), we can provide certain guidelines. The objectives of this chapter are to show that:

- The important activities in analysis are elicitation of information and formulation of the requirements. Techniques such as interviews, questionnaires, brainstorming, scenarios and prototyping are all useful.
- Structuring of information is one of the few activities where formal methods are beginning to appear. The use of these depends upon the type of information and whether legacy data exists.
- Designing associative links can be incredibly time consuming, and is currently difficult to manage other than manually.
- Design of searching and indexing methods can use well established techniques from database systems and information science.
- Design of an application's look and feel is still largely an art rather than a science.
9.1 Introduction

In the last chapter, we looked at the development process in general and the various activities we have to carry out within the development process. In this chapter we look in detail at how to carry out some of the hypermedia-specific activities.

We have already mentioned that some of the activities we have to perform are common to other disciplines of engineering, such as software or systems engineering. These activities are in the areas of analysis, planning and coding (in the case of a programming language-based product model). We do not discuss these activities further here.

When considering the practicalities of selecting and/or adapting an activity, we should also consider the cost associated with that activity. The most significant element of cost-effectiveness will be the development productivity. Two aspects which strongly influence productivity are the structuring and linking of information, and reuse and maintenance issues. The activities we are going to use when developing a hypermedia application should allow us to address these aspects in a systematic way.

9.2 Requirements Analysis

To carry out the analysis activities we can perform various tasks, including elicitiation of required information, formulation of this information into a logical structure and validating the information against client or user requirements, or for internal consistency. These activities are shown in Figure 9-1.

Elicitation

Elicitation is the task of gathering all of the information that we think is relevant to the type of analysis needed. Rather than overlooking some important information, it is better to collect all the information we think may be relevant and then eliminate those sections that we find are not. As knowledge elicitation from humans is inherently very difficult, this task can be very labour-intensive.

Techniques which are commonly used to elicit information include:

- Interviews with both the clients and prospective users of the application to determine what their requirements or expectations may
Figure 9-1
Activities associated with requirements analysis (reprinted by permission of Vithanage (1997))

be. This can include aspects such as a discussion on possible information sources, ways in which this information might be used, the different information relationships which might exist, the types of metaphors which are familiar to users, etc.

- Questionnaires and surveys can address similar aspects to interviews, but can provide a much wider coverage.

- Brainstorming can be used to identify possible alternatives, approaches, requirements, information sources, etc. Brainstorming is a useful technique in ensuring that aspects of the requirements have not been overlooked.

- Scenario analysis can be used to obtain a better understanding of how an application might be used, and therefore what functionality and information may be appropriate. For example, a scenario for the HyperBank application might be as follows:

  Customer X connects to the HyperBank main World Wide Web homepage using their favourite Web browser. From this homepage they navigate to a page providing access to personal account information. They enter their name and account number and the application provides their account balance. They follow a link to a page providing a detailed listing of their account details for the last month. They then request that all transactions in the last year relating to interest paid be totalled, and the application responds with the requested total.
The above scenario provides us with a wealth of ideas regarding requirements for the inclusion of specific types of information (such as account balances and transaction histories), functionality (such as an ability to calculate totals for specific types of transactions) and behaviour (such as the application being accessible using common WWW browsers).

- Analysis of existing documents, systems, or applications can be used to gain an improved understanding of both possible requirements, and the domain within which the application will be used.

- Prototypes can be used to assist users in understanding the implications of various forms of information and functionality. In particular, users are often unaware of the potential and limitations of new technologies such as hypermedia, and therefore are not able to adequately identify or articulate their requirements. Once a prototype has been seen, a user is often able to better understand what is feasible. Prototyping can require considerable effort, but has the potential to elicit detailed feedback.

- Literature searches can provide information related to similar systems, technological or other constraints, and other types of knowledge which might impact on the application requirements.

**Formulation**

This involves organising information that was obtained into some logical structure and eliminating redundant information. Hierarchical organisations are quite common when organising information.

**Validation**

The requirements identified need to be validated against the client’s intention, if they originated from the client. Similarly, if the requirement originated from a user, it needs to be validated against the user’s intention. Also, these requirements need to be validated against the goals of the application, and for consistency, e.g. there cannot be two requirements that contradict one another.
9.3 Information Structuring

In Section 2.1 we discussed the different types of links, and provided one categorisation: structural, associative and referential. Structural links reflect the underlying organisation of the information, whereas associative and referential links provide navigation support based on the semantic associations between the underlying concepts. In each case, we need to consider specific techniques for generating both the information relationships and the links which implement these relationships. Let us begin by considering structural links – as it is these that provide the form of the information space within which the user will be moving and navigating.

Information Structuring Methodologies

There are currently few methodologies which can be used to guide developers during information structuring. The type of approach adopted for content structuring will depend very much upon the nature of the data being used in the application, as different applications will have different information models.

West and Norris (1997) describe an approach for managing the design of small to medium scale applications. Their approach addresses the design of the document or page structure within an application particularly well (i.e. structural links as distinct from associative links). For example, they consider the development and appropriateness of alternatives such as guided tours, hierarchical structures, menu and index pages, and storyboarding. They also provide a very good overview of the management aspects of the development of the information structure. What they fail to consider, however, is the effective design and authoring of the complex associative linking that occurs in large-scale applications.

An alternative (though complementary) approach is the Relationship Management Methodology (RMM) (Isakowitz, Stohr and Balasubramanian, 1995). This is suitable for applications that have highly structured information models with a lot of information in a collection of related classes (the types of applications which have traditionally been suitable for database applications). These types of application are often based on a physical system (such as an information system for library loans, or for details on personnel in an organisation).
There is another class of large information applications based on legacy data which are not readily class-based, including applications such as those which include a large amount of information on organisational procedures. For such applications, a diagramming technique can be used. In this formal technique, the designer attempts to organise the information into a hierarchical structure. The approach is also very useful in the production stage, as it allows us to partially automate the information structuring based on the information model developed at the design stage. Figure 9-2 illustrates the use of the diagramming technique for a simple example (a manual), but it can just as easily be applied to a large complex information set which has a well defined structure (Figure 9-3). Table 9-1 explains the various symbols used in the diagramming technique.

Figure 9-2
Example elements of a manual
Where we start content design varies depending on what information is available. There can be three broad possibilities:

1. Legacy data is available (for example, if we are developing a legal information system, the relevant legislation and case histories are available in print form).
2. No legacy data is available, but a physical structure exists (for example, if we are developing an information system for an organisation there may not be legacy data to cover all aspects of the operation of that organisation, but a management or an organisational structure exists that can be used as the starting point for content design).
3. A new application where no legacy data or physical structure exists (for example, an information system for a new education module in a distance learning application).
## Table 9-1
Symbols used in the diagramming technique

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Content Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A component in a document called &quot;A&quot;, (e.g. a paragraph).</td>
<td>A</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A component which can not be decomposed further.</td>
<td>#PCDATA</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>0 (A) 1 Element A is optional or can occur once.</td>
<td>A?</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>1 (A) n A is repeated 1 to n times where n can be any integer.</td>
<td>A^*</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A and B in series (B after A).</td>
<td>A, B</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>1 (A) 2 Consists of one or two As. ie the second A can be bypassed.</td>
<td>A, A?</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>2 (A) n Two or more components of A.</td>
<td>A, A^*</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A can be bypasses or repeated – similar to 0 (A) n.</td>
<td>A^*</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A or B or C or D Selection.</td>
<td>A/B/C/D</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>Parallel components with repetition.</td>
<td>A, (B+C)</td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>Note: A, B+ C is incorrect.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Symbol A" /></td>
<td>A and B and C and D in any order.</td>
<td>X, (A&amp;B&amp;C&amp;D)</td>
</tr>
</tbody>
</table>
Let us look at each of these three cases in some detail.

**Legacy data available**

In this case the starting point of content design is obtaining all the relevant legacy data, which we then analyse. Often, legacy data tends to have a hierarchical structure, so one can use a diagramming technique as described above to capture and manipulate this structure. This approach is especially helpful if, during the implementation phase, the information is going to be marked up using SGML (Goldfarb, 1990).

**No legacy data, but a physical structure exists**

In this case the starting point is to model the physical structure. These applications tend to have a highly structured information model. The Relationship Management Methodology (RMM) (Isakowitz, Stohr and Balasubramanian, 1995) is a useful approach for applications that have a highly structured information model. This is because RMM uses an approach based on entity relationship modelling, and hence it can model highly structured information well.

**A new application with no legacy data or obvious physical structure**

If we are developing such an application, we have to use domain-dependent theories to develop the initial information structure. An example of this is the use of learning theories for educational applications. For this purpose, we can classify the different domains as shown in Table 9-2.

A final question relevant to all three of the above cases is the question of how to eliminate redundancy from the data sources. Once the existing structure of the information has been identified, it has to be refined, and any redundancies in information needs to be identified and removed. For example, it is quite common at this stage to find the name of a person and some associated details being repeated in many places within the information space. From a maintenance point of view, it is important to keep this information in one place and create links from other locations to this single location.

Another option is to identify the basic information elements and how these are combined to create various user views, store the basic information elements in some form of a database and generate the user views using a suitable script.
**Table 9-2**

Example application domains where learning theories can be applied

<table>
<thead>
<tr>
<th>Task-specific</th>
<th>Training</th>
<th>Learning</th>
<th>User-specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assists to perform a task</td>
<td>Tells how to perform a procedure</td>
<td>Provide expert viewpoint</td>
<td>Provide information on demand</td>
</tr>
<tr>
<td>Example: airline reservations</td>
<td>Example: procedure manuals</td>
<td>Example: Textbooks, scientific papers</td>
<td>Example: video, news, weather, tourist info. etc. on demand</td>
</tr>
</tbody>
</table>

**Information Database Design**

In information structuring we design the logical structure of information and the relationships among different information units. In information database design we design the physical structure in which to store the data. This information database can consist of set of mark up files and other files containing various non-text media elements such as images, video and audio clips. Other possibilities are that we may decide to use a relational database or an object-oriented database to physically store the information required for the application.

Many conventional databases and database tools have been modified to incorporate the ability to work with hypermedia systems (and in particular, Web applications). An example of this is the development of Java APIs to allow Java applications to interface (through ODBC and/or JDBC) to databases. Similarly, tools such as Microsoft Frontpage incorporate the ability to interface to databases such as Microsoft Access.

Currently, no specialised tools or techniques exist for designing the database structure specifically for hypermedia applications. Conventional database design techniques, however, will still largely be relevant, especially when used in conjunction with mechanisms such as RMM’s entity-relationship design methods, which can provide an indication of the information structure.
A main consideration when developing the information database structure is the maintenance requirements. With respect to maintainability, as both the number and size of applications expand, there will be an increasing need to take these into account. Applications will no longer be static; they will increasingly tend to evolve over time (this trend is evident in the World Wide Web even now). As a result, we need to develop applications in such a way that allows them to be easily maintained.

For adaptive maintenance it is essential to store similar information, such as names of people in an organisation and their details, physically together in a single file or database table. Then when there is a change in the physical organisation in terms of people, it is easy to change the corresponding information in the information system by just modifying a single file or database table.

9.4 Associative Linking

Whereas structural linking looks at the physical organisation of the information space, associative linking provides the ability to navigate across this structure based on conceptual or semantic associations. It is these links which provide the real strength of hypermedia systems.

The linking of related information is in most cases still done based purely on location – an anchor in a specified media item is linked to a fixed location in another media item. This is typically a constraint imposed either by the development tool or the underlying information model, implying that the author will need to know the location of associated information to create a link. When developing large hypermedia systems, the need to remember the locations becomes a major cognitive burden on the author, affecting both the quality and productivity.

Probably the most challenging aspect of the linking process is the need to mentally manage all the existing nodes within a set of information. This problem of finding relevant destination nodes for a given concept is shown graphically in Figure 9-4. When the number of information nodes in the system is small, the user can remember the information associated with a particular concept and create the links. Almost all authoring tools commercially available at present support this type of manual linking. When the information space is large (over 500 nodes is a typical figure), it becomes increasingly difficult for the author to remember information contained in all the nodes. If the system is developed by many authors
Figure 9-4

As the application size increases, managing the information space from a developer's perspective becomes increasingly difficult. The complexity involved in creating meaningful structure increases rapidly with application size. This adds to the problem. Thus, with increasing size the cognitive overload becomes unmanageable.

Techniques for managing the burden associated with linking is still an open research issue. In practical terms, however, the use of tools which can parse text (usually based on templates that have been developed for specific application domains) provides a reasonable starting point. For example, many SGML tools (such as OmniMark) provide pattern matching languages which can be used to specify the structure of legacy information, thereby semi-automating the structuring and linking process.

Designing Associative Links

In designing associative links, probably the most useful current technique is to begin by identifying a suitable basis for the creation of associative and referential links. Often, these are expressed as a set of rules. For example, with referential links we can have a rule such that if a word exists in a glossary used for the application, then there should be a link from this word anywhere in the application to the entry in the glossary.

Defining a basis for creating associative links is yet more complex. The developer first has to decide on a relevant set of associations, which could be:


- A thing and its application (e.g. in a legal information system, a law and its interpretation or case history).
- A cause and an effect (e.g. in a history information system, a war or a revolution and its social impact).
- Thing and a strongly associated property (e.g. in a store catalogue, a product and its financing plan).
- A raw material and a product.
- Two complementary and concurrent activities.
- An activity and an agent of that activity (e.g. in a history, the apartheid struggle and Nelson Mandela, a lecturer and subjects taught by the lecturer).
- Whole and part (e.g. in a science application, a diagram of a car and how each part works, a university degree program and a subject description).

From a practical perspective, a number of guidelines exist to assist in ensuring effective links. These include aspects such as ensuring consistency in associative linking, and ensuring that the link anchor provides as obvious indication of the link destination as possible. For example, consider the text:

In designing associative links, probably the most useful current technique is to begin by identifying a suitable basis for the creation of associative and referential links.

In this text it is not clear what information would be contained in the destination for the link from the text ‘associative links’. It may be a definition of associative links, it may be an essay on how to design associative links, or it may be information on how different hypertext systems (such as the WWW) implement associative links. If the latter was the case, then a more informative anchor may be:

In designing associative links (more information is available on how they are implemented in different systems), probably the most useful current technique is to begin by identifying a suitable basis for the creation of associative and referential links.

Beyond these relatively simple observations there are a number of much more complex approaches, such as the use of generic and dynamic links.
These are still not widely used, and will be discussed in detail in the final part of this book (especially in the chapters on Microcosm, Hyper-G, and the final concluding chapter).

9.5 Example of Link Design

As a simple example of how we may design specific link structures, let us consider a small subset of the HyperBank case study. We shall look at three levels of structure within this application: the high-level application structure; the detailed structural linking of one section of the application; and the associative linking within this section.

High level structure

First, let us consider the high-level structure of the HyperBank application. We can begin by considering both the potential users, and the sources of information. The potential users vary incredibly, from current and prospective customers to internal users (staff) and external users (bank suppliers and regulatory agencies). Each of these system users will have differing levels of expertise and different requirements. However, the information they will be accessing will in many cases be the same (such as both staff and customers accessing customer transaction records).

This implies that the structure of the information space should be organised in a way which allows different users to be presented with different structures for the underlying information space. Indeed, different users must be able to visualise the information space in very different ways (much as a commuter who uses a subway system will visualise the layout of a city differently from a commuter who always drives).

This indicates that at the very top level of the information space, we have a demarcation between the different potential users. Note that this does not imply that when the final Website is completed, the homepage would reflect this partitioning. It is likely that it would be considered appropriate (from a customer relations standpoint) that the perspective seen by the customer when first entering the Website is only a reflection of their viewpoint on the information space, and not the total range of different perspectives.

Having identified that the top-level structure should reflect the fact that the underlying information is organised in different ways for different users, we can look at the structure for a single user – current customers.
Given the variable level of expertise and knowledge of customers, a very simple structure to visualise is most desirable – a standard hierarchy (though we may determine that certain information belongs in numerous places within this hierarchy).

To determine the specific details of the structure, we need to consider the information we wish to provide to the customers, and their expectations of how this information is organised. This could be achieved by interviewing a range of current customers; investigating both current information to which they have access, and future potential information sources; analysing the current structure and content of information sources; developing and analysing typical customer scenarios; analysing competitors' Websites; and possibly developing prototypes. A typical resultant high-level structure of the HyperBank information space, as seen by a customer, is shown in Figure 9-5.

**Detailed structure**

Once we have a top level structure we can start to develop the detailed structure of specific information components. To demonstrate this, let us consider the information provided by the bank on the conditions of use of a specific type of bank account – say, a personal cheque account. This information already exists in a conventional document form, and is mailed out to new customers along with account details, or mailed to existing customers whenever details change.

This implies we use can the existing structure of the document to assist in defining the structural links. For example, the existing document may have the following (partial) form:

**Terms and Conditions for HyperBank Personal Cheque Account**

- Deposits . . .
- Interest . . .
- Bank Fees and Charges . . .
  - Fee types: account servicing fee, excess withdrawal fee, . . .
  - Current fees
  - Exemptions and Fee rebates
- Provision of credit
- Termination
- Error and Dispute resolution
  . . . etc.
Figure 9-5
Structure of the HyperBank information space as seen by a customer

We could immediately treat this as a suitable structure for the electronic form of the information, and simply migrate the information directly onto the WWW. Indeed, much information on the WWW (and other hypermedia applications) simply provides what is essentially access to existing documents. This case would have two disadvantages: first, it would not be making the most effective use of the hypermedia functionality available; secondly, it it does not acknowledge that many of the related documents
(on conditions for other types of accounts) will have a similar structure. This second point should indicate that we ought to identify a common structure and then map the information into this structure. The headings listed in the example above would be common for most types of bank account. The main difference may be specialised fee types for different accounts, and conditions on account balances and transaction types, etc.

We would end up with a set of terms and conditions for each account type. This could be viewed as a matrix structure (the account type, and conditions forming the two axes). This information could be mapped into a suitable SGML structure, thereby minimising the effort required to both maintain the information and to present it in different forms (the paper version of the information will still be required!) A suitable SGML Data Type Definition could be developed to define the structure of the information to be presented. Figure 9-6 shows how this could be mapped to a specific Web page design in order to present the information to the

**Figure 9-6**

High-level presentation format of the *HyperBank* Web
user. These pages could be created dynamically from the SGML information by using a suitable filter.

**Associative linking**

Once we have a structure for the information space we are creating, we need to consider the support we provide for navigation within this space using associative links. There are numerous ways this could be managed, depending upon the types of associative links we wish to develop. The first step in developing suitable associative links would be to identify the types of relationships we wish to support. To facilitate, this we analyse the information sources and develop the following association types:

- A concept and it's definition
  You are entitled to make up to 10 withdrawals from your high return savings account
  Withdrawal: A withdrawal is defined as a . . .

- An entity (such as a customer's personal account) and the conditions on that entity.
  When using your account you must always . . .
  Savings Account Terms and Conditions
  The terms of usage of a . . .

- An organisation and details of that organisation
  HyperBank has for many years had a strong association with Electronic Investments Incorporated. This . . .
  Website homepage for Electronic Investments Incorporated

- etc.

The difficulty, of course, lies in knowing when to use an appropriate link type. This will particularly be the case where a given anchor could have several different interpretations. For example, consider the following link anchor:

Your current account balance is available from . . .
  Savings Account Terms and Conditions
  The terms of usage of a . . .
or links to

Defn: Account: An account is defined as a . . .

or links to

The HyperBank offers many different types of accounts. In order to determine which is the best type of account for you needs . . .

In this example, if a user encountered the anchor account, the user would be uncertain as what information this anchor was linked to.

Three possible solutions might be to:

- Modify the anchor to make it more expressive (such as 'Your current account balance (which has various terms and conditions) is available from . . .')
- Provide a mechanism for selecting a link and obtaining more detailed information on the destination prior to activating the link.
- Provide a mechanism for having anchors link to multiple destinations.

Advanced hypermedia systems allow links to have multiple destinations (and the user is able to select the appropriate destination from a list of choices). This mechanism is not supported on the Web, and so we are forced to simply rely on design guidelines. These should be developed to be consistent. For example, the following are a typical set of guidelines for use with the HyperBank system:

- Any single words (or composites) which are part of standard text (e.g. 'When using your HyperSaver Account you must . . .') will be linked only to a glossary of terms which provides a definition of the term.
- Any bullet points or list items which incorporate links (e.g.
  - Automatic teller machines
  - Electronic Funds Transfer Point of Sale . . .'

will link only to detailed information on the given item.
- Any other associative links will utilise full anchors which make the link context and destination obvious (e.g. 'All HyperBank accounts are required by Law to incorporate a standard transaction tax. More
information on this tax is available from the Government Taxation Office homepage.

9.6 Access and Navigation Support

The design of access and navigation mechanisms includes the design of navigation structures based on user viewpoints, establishing a basis for anchor selection and link traversal and, if applicable, selecting an indexing approach for searching the information.

Design of Navigation Structures and Viewpoints

When structuring content the aim is largely to develop a data structure that will assist in fully identifying the required content, and also help to eliminate redundancies in the information. Often (indeed, most of the time) this will not be the structure a user wants to follow when accessing and utilising the information. During viewpoint design, we try to identify how users are going to access the information, and then superimpose multiple user viewpoints on top of the original information structure.

A common technique used to determine user viewpoints is to write the various headings on a set of index cards and ask a sample of users to group these based on how they would like to navigate the information space. Once we get this information from a number of users, we can do a cluster analysis to determine the dominant user viewpoints and provide hyperlinks to support them.

Design of Searching and Indexing Mechanisms

Another important activity that needs to be carried out at the design phase is to decide whether to use a search engine and, if so, what type of searching mechanism to use.

Before discussing specific approaches, it is important to understand several basic concepts related to measuring the effectiveness of searching mechanisms. When an end-user begins searching there will be certain information in the information space which will be highly relevant to their information needs some which will be useful, some marginally
relevant and some not relevant at all. Only some of this information will be returned by a search engine. We measure the performance of a searching mechanism using two ratios: recall ratio and precision ratio.

Precision is the ratio of the number of relevant documents (or nodes, pages, etc.) retrieved, to the total number of documents retrieved, i.e. the proportion of returned ‘hits’ which are relevant. Recall is the ratio of the number of relevant documents (or nodes, pages, etc.) retrieved, to the total number of relevant documents, i.e. the proportion of relevant documents which are returned as ‘hits’. This is illustrated in Figure 9-7.

It is also worth noting that, with most search engines, we have the ability to trade off precision against recall. As shown in Figure 9-8, if we increase the number of documents retrieved, then we are more likely to retrieve more of the relevant documents, and hence increase recall. At the same time, however, we will typically retrieve more inappropriate documents and hence decrease our precision.

In developing search engines we can either have a full text search or search using a selected subset of words. There are two main ways of selecting this subset: one is to use a statistical mechanism to select the words from
the text, an approach commonly known as keyword searching; the other is to use a control vocabulary to select the indexing terms.

- **Keyword searching:** for large sets of documents it is economical to assign keywords using a statistical algorithm based on the frequency of occurrence of certain words. In this approach, the keywords are automatically extracted from the document. This has the advantage that the whole process can be readily automated, making it economical for very large document sets. However, if during searching the user uses a synonym (for example, the keyword in the document is ‘compact disk’ and the user typed CD ROM), the application may fail to find the information. Therefore, this approach may not produce results as good as manual indexing based on control and entry vocabularies.

- **Searching using a control vocabulary:** in this approach we first develop or select an existing vocabulary for the subject domain. Generally, these vocabularies are hierarchically organised.

Once a user selects a vocabulary, you have to select what level of granularity is required and decide what index terms to use. For example, for an article we can assign the index term ‘Mammals’ or be more specific and say ‘Dog’. This type of indexing system will use a much larger entry vocabulary that will have pointers from synonyms to words used to index the document. For example, if a user typed CD ROM it will then point the user to compact disk.

This approach tends to give much better results in terms of precision and recall, but is very time consuming as all documents need to be
manually indexed. During the design phase, one has to weigh the advantages and disadvantages between the two methods within the application context and select a suitable indexing approach.

Design of the Overall Look and Feel

Overall screen layout

The overall screen layout should provide an appropriate visual impact, but most importantly, should make it easy for people to use the application. The designer can enhance the usability of the application in many ways with respect to look and feel. Examples include the use of metaphors, appropriate labelling of information nodes, and use of split-screens.

- **Use of metaphors:** if the designer knows about the user’s background and experience, then he can select a suitable metaphor, by which we mean the application of some form of schema to provide a basis for understanding the applications’ structure, functionality, behaviour or interface by analogy.

Metaphors are used to relate the task being performed to something with which the user is likely to be familiar (often something from the domain or environment described by the application). A very common example is the use of a book metaphor – so that the display appears as a book to the user. The user will then readily be able to relate to concepts such as turning pages, using tables of contents and indexes, etc. Another example might be laying out the main application interface to look like a kitchen, for a cooking application. Proper use of metaphors can greatly reduce the cognitive burden associated with learning how to use the application.

A designer can use a metaphor for the overall application, or for individual parts of the application. A good example at the overall application level is the common desktop metaphor used by many computer operating systems. Another example is the use of a card index metaphor for a hypermedia telephone directory. Often, users use a card index to keep a record of telephone numbers and they know how to find a telephone number quickly. When you use a card index metaphor and provide similar functionality to conventional card indexes, users can intuitively start using the application more effectively.

Use of icons such as filing cabinets and trashcans are examples of using metaphors for the individual components of an application
to enhance the usability. Appropriate metaphors can help a user understand the functionality and operation of an application. They can also assist in helping a user understand where they are within the information structure (thereby avoiding the 'lost in hyperspace' problem). For example, if we support a cooking application with a presentation metaphor of a kitchen, it can become obvious whether we are discussing ingredients (the display shows a cupboard), preparation (the display shows a workbench), cooking (the display shows a stove), etc. We can also use other mechanisms to also support a user in understanding their location within the information space. Examples include appropriate labelling, use of split screens, and information maps.

- **Appropriate labelling**: support for orientation of a user within the information space can be supported by appropriate labelling of information which is presented. A simple example is the use of structured labels within a hierarchical application – which might take the form:

  \[ \text{chapter\_title} - \text{section\_title} - \text{node\_title} \]

  For example, we might have a node labelled

  Cooking 'chicken kiev' – preparation – selecting ingredients

  Labels such as these provide valuable information to locate and contextualise the user.

- **Use of split screen**: another technique that can be used to show the user the overall information context is the use of split screens. Here the screen is divided into two or more parts, and some parts are used to provide the location and the other part for the content. An example of this approach is shown in Figure 9-9.

Such an approach allows the user to always have access to a single, well-understood mechanism for navigating. In a more advanced form, the fixed component used for navigation can change to indicate the location within the information space. Taken even further, this navigation mechanism can take the form of an explicit map of the information space.

---

**Figure 9-9**

Examples of split screens designed to facilitate navigation
The role of the internet in teaching large undergraduate classes

Abstract

The tertiary learning environment has been under significant pressure. Government funding cuts, professional accountability and student demands for flexibility and relevance have all contributed. As student numbers increase dramatically and funding formulas become more closely tied to academic research outputs, academic staff have fewer opportunities for individualized attention and interaction with students. While information and communication technologies are not a panacea for all student learning problems, they can play an important part in improving it. This is particularly so with the Internet. Internet usage is expected to improve the learning environment by increasing student interaction and access to various learning resources, enabling greater integration of

Context Diagram

The context diagram is the best place to start when attempting the functional design of a system in order to ensure that the interface to the system (tacked from the specification) is complete.

Having a complete context diagram also allows checking of the functional block diagram (to ensure that all inputs and outputs are accounted for), potentially avoiding unnecessary design flaws.
Design of interface screen elements

Once we decide on the overall screen layout, then we have to design each component of the interface. This can include the design of elements such as:

- Forms.
- Menus and Labels.
- Buttons.
- Windows/Frames.
- Icons.
- Tool bars.
- Dialogue boxes.
- Status lists.

In using these interface elements there are a number of important considerations, including how to most effectively select and group navigational buttons, making sure that the meaning of interface elements is obvious (this is related to the above discussion on the use of a suitable metaphor), minimising screen clutter and visual distractions, the suitable layout of the interface elements, etc. These aspects are well covered in most texts on user interface design.

- **Colours**: different colours convey different meanings, and this will often vary with context, culture, gender, etc. For example, red can mean danger, green can mean freshness.

- **Fonts**: most popular fonts can be classified into five categories: these are Oldstyle, Modern, Sans Serif, Script and Decorative. The general rule is that one should not use more than two categories of font types in a design – one category for labels and headings and another for the main body text.

Developing Guidelines for Content Development

There are various principles we can use to determine the guidelines for developing content. The following principles were developed by Horn (1989), based on various cognitive and user research. Four principles of node construction are:
- **Chunking principle:** whenever groups of information, concepts or choices are presented (such as lists of information, buttons across a toolbar, items in a menu), the information should be grouped (i.e. ‘chunked’) into manageable units. The number of units in a group will usually be defined by the number that humans can handle within their short-term memory limitations. A good rule of thumb is to provide a limit of $5 \pm 1$.

- **Relevance principle:** each node should be based on a single primary point or concept, which is readily conveyed to the reader. This improves understandability.

- **Consistency principle:** for similar subject matters, use similar words, labels, format, organisations and sequences.

- **Labelling principle:** label every node and group of nodes, and other interface elements such as buttons and links, according to specific and appropriate criteria. Proper labelling helps users understand their context, as well as understanding the consequences of specific operations (see Figure 9-10).

Horn also identified that most information we come across can be classified into one of seven major categories (Horn, 1989), shown in Table 9-3. This information type classification becomes very helpful when applying the chunking and relevance principals mentioned above. We can identify a consistent chunk of information as information describing a concept, classification, process, procedure, etc.

### Table 9-3

<table>
<thead>
<tr>
<th>Categories of Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>What is it?</td>
</tr>
<tr>
<td>Classification</td>
<td>Can we group or divide based on some property?</td>
</tr>
<tr>
<td>Process</td>
<td>What happens or How it works?</td>
</tr>
<tr>
<td>Procedure</td>
<td>How to carry out a task?</td>
</tr>
<tr>
<td>Structure</td>
<td>What are the sub parts?</td>
</tr>
<tr>
<td>Fact</td>
<td>Statement of data without supporting information</td>
</tr>
<tr>
<td>Principle</td>
<td>A rule that describes a relationship or what should or should not be done.</td>
</tr>
</tbody>
</table>
Figure 9-10
Labelling information chunks. Which is more informative?

Individual screen layouts

The digital medium supports a wide range of media forms, and by properly combining these we can enhance the communication of the message in our hypermedia applications. We can develop guidelines for individual screen layout based in the information types described above. For example:

- **Classification**: always specify the basis used for classification. Keep the number of items in the classification list to about 5±1, in keeping with our short-term memory capacity.
- **Structure**: use of visual representation and hyperlinks to explain the parts using text and/or audio.
- **Process**: visual representations of process (such as flowcharts, dataflow diagram, etc.) can be important. Use of animation and video can be especially effective.
- **Procedure**: step-by-step guide generally formatted using a table or flowchart. Methods for performing each step can be enhanced using other media forms (such as a video clip of the procedure being performed).
- **Other information types**: mainly presented as text.

9.7 Production Techniques

The main emphasis during the production process is on productivity. We can enhance productivity through automation, use of appropriate tools where automation is not possible and information reuse.

When developing large hypermedia information systems there are certain areas of the production process which are most amenable to automation. Most of this is in structuring (based on syntactic structure of legacy data) and algorithmic linking (creating links based on some algorithm) of textual information.
Certain production activities require decisions based on the semantics of information being processed, including structuring of mostly non-textual data and certain types of associative linking. To carry out this type of production operation, we need to use tools that will minimise the cognitive burden on the author and assist the author to increase productivity.

Information reuse is another factor that impacts heavily on productivity. It will become increasingly impractical to recreate the underlying information structures for every new application. Where possible, authoring productivity (and productivity of the overall process) can be greatly improved by reusing information structures which have already been developed for other hypermedia applications (or even applications such as databases, archives, etc.). This can be actively promoted by leaving those aspects of the information structuring which impose a specific viewpoint on the information to as late in the development process as possible (see the description of the Matilda project later in this book). Other factors which will help promote reuse include the development of suitable repositories and data models, browsing tools, standards interoperability, etc. For example, it may be appropriate to include, as part of the process, planning for a dedicated role of 'reuse engineer'.

**Information Organisation**

Capturing information (such as entering text, digitising and editing images, etc.), structuring this information (such as partitioning of the content into specific nodes), and linking the information (creating the logical structure and semantic relationships) is very time consuming. Indeed, these activities often take most of the time spent in producing hypermedia applications. Non-textual information (such as images, video and sound) can be particularly time-consuming, as these are still predominantly structured manually. We need to consider the extent to which the process helps manage these activities.

Organising the information involves activities such as breaking the information into nodes, identifying key concepts that best describe the information contained in that node, and marking anchor points. As a general rule, information within a node should be focused on a single theme. The key concepts identified during the structuring process will assist later when searching for specific information.

Various researchers have looked at automating these activities (Furuta, Plaisant and Shneiderman, 1989; Niblett and van Hoff, 1989; Rearick,
1991). A common strategy for automating the identification of nodes within paper based information is to use existing structural information. Keying on markup languages such as SGML or undertaking layout analysis of the text (Furuta, Plaisant and Shneiderman, 1989) are popular approaches. These approaches can be effective if the information is uniform in structure or has been previously marked up, but otherwise these approaches break down quickly.

Also, researchers have used information retrieval algorithms to produce key phrases and/or anchors. Although this is a step in the right direction, this approach introduces the problem of missing relevant anchors and key phrases, plus creating key phrases and anchors which are not relevant (recall and precision) (Bernstein, 1990). These projects have demonstrated that if the information contains non-structured documents or a document set with divergent structures within it, total automation is not possible. Further, we have a major problem if we try to extend these methods to non-textual media such as images, video and sound, as these do not have a well-modelled syntax.

**Information Integration**

Depending on the scale and the structure of the application the integration of various components to produce the application can either be done manually or automated. If the information has a relatively consistent structure, the various links that needs to be created can be expressed using an appropriate algorithm, and if the size of the application justifies the extra effort required to automate the process, then an automated approach can be used to integrate the various components and create the links.

If these characteristics are not present, then we have to use a manual approach to integration. This process of physically creating the application is known as authoring. Authoring involves the development of the macro structure as well as the micro structure of the application. Creation of a micro structure involves developing and formatting individual nodes. Many authoring tools have page editors to assist in creating the micro structure. The support available for creating the macro structure varies greatly from tool to tool.

Creation of the macro structure involves the integration of nodes, windows, forms, etc. into a cohesive application. This integration happens through the linking of nodes, windows, forms, etc. together. The non-linear information structure of the hypermedia applications make this
Figure 9.11

Microsoft Frontpage can be used to visualise the Web pages which are linked to, or from, given pages linking process very complex and demanding for the author. A common result is to find many errors occurring during manual linking, thus if manual linking is to be used for economic reasons, it is important to use a tool such as Macromedia Authorware which allows the creation and editing of the overall structure, or Microsoft FrontPage which shows a graphical view of the structure (Figure 9.11).

9.8 Conclusions

At the conclusion to Part Two of the book, we should revisit the concept of process. It is important to bear in mind the necessity of a suitable process, a process which is neither easy nor straightforward. Lehmann states:
‘Only by a miracle can the quality of (a) software product be better than the process that produced it . . . ’

This is just as true for hypermedia development. Unless we can get the process right, and then the activities within this process right, we will be unable to develop quality hypermedia applications. In general, there are no easy solutions or ‘silver bullets’ for the difficulties inherent in hypermedia development. Although we have discussed various techniques and methods in the last three chapters, these are still far from providing a comprehensive approach which guarantees success. Such an approach does not exist: indeed, it is unlikely that such an approach will ever exist due to the huge variation in problem, solution and development domains. Our initial aim should therefore be to develop an understanding which allows us to maximise the application quality and development productivity and minimise the risks of development.

Although the technologies to support hypermedia are changing constantly, this should not stop us from developing effective processes. To do this we need to ensure we have an understanding of not only current techniques, but where and why these techniques might be inadequate. One way of developing an improved understanding of these aspects is to consider current research and how it is approaching the development of solutions to particular problems. This is the focus of Part Three of this book.