3
Design

1 Introduction

In Part 2, you saw the importance of taking a user-centered approach to developing a user interface (UI). This approach requires an understanding of users and their requirements, and involves them in the design and evaluation of UIs.

Part 3 of this book has these main aims:

- To introduce you to conceptual design and, in particular, to show you how to create a content diagram.
- To explain the various sources of design advice and introduce four more key design principles.
- To explain the use of metaphor in UI design and the different interaction styles that are available.
- To provide you with the necessary skills for choosing suitable input and output devices and for using text, color, images, moving images, and sound effectively; to show you how to combine these to create a usable UI.
- To give you practice in designing different types of user interface. In particular, we look at three different approaches to combining the design components to create a UI. We start by looking in detail at graphical user interfaces (GUIs) and web sites. For both of these, we present a range of design issues, rules, principles, and guidelines to help you create usable designs. We then look more briefly at generic aspects of embedded systems.

1.1 Overview

In Part 2, you learned how to analyze an existing UI and hence gather the requirements for a new UI. The next step, work reengineering, is introduced in Chapter 8. Work reengineering involves identifying how a new UI can best support users' work. An important aspect of work reengineering is task allocation, in which you decide how the various tasks will be shared between the user and the computer system. We then move on to conceptual design and to creating a content diagram. A content diagram represents the underlying organization and structure of the new UI, and it can be used to inform the physical design of the UI. We demonstrate work reengineering and conceptual design through the example of a digital library system for use within a university department.

Designing UIs is a complex process, so in Chapter 9 we explore the different sources of design advice, in particular user interface standards and commercial style guides. We suggest managing all this advice by creating a customized style guide for your UI. A customized style guide can also form the basis for your design rationale, or justifications for the design decisions you make. Building on the principles of affordance, visibility, and feedback from Chapter 5, four more important design principles are presented and explained: simplicity, structure, consistency, and tolerance.

To ensure your UI is usable, you must understand the interaction process. In Chapter 10, we start by considering a model of interaction known as the human action cycle.

This model highlights the various problems that can arise when a user interacts with a computer system. We also consider the importance of the user having an accurate mental model of the system, and the use of metaphors to help users develop accurate mental models.

In Chapter 11, we use the human action cycle to help you choose between the available interaction styles, including command-line, form-fill, menu selection, direct manipulation, and anthropomorphic. We also briefly revisit the psychological principles for design set out in Chapter 5.

The main components that make up the UI can be divided into hardware components and software components. Chapter 12 looks at the hardware components — that is, the input and output devices. You may feel that you have little control over the devices you use, but using the correct device can make all the difference. A supermarket that chose not to use bar code readers/scanners would probably soon be out of business. We show you how to choose the most suitable hardware components.

Chapter 13 discusses the software components: text, color, images, moving images, and sound. These combine to provide users with feedback from the system, and we show you how to use these to best effect. You can make a lot of difference to a UI simply by changing some of the colors or the type size.

In Chapter 14, we show you how to combine the software and hardware components into a usable UI that meets the needs of your users, allowing them to carry out their tasks within their environment.

Chapter 15 illustrates how design was undertaken by Tokairo, UK, for the design and development of a worksheet system for its truck drivers.

UIs can be classified according to the way in which their various components are combined. We refer to these groupings as design areas. In Chapters 16 through 18, we identify three such design areas, and we consider the design issues that relate to each. We concentrate on GUIs and web sites, as these are the types of UI that you are most likely to be able to influence in your workplace.

Chapter 16 is about how to design usable graphical user interfaces (GUIs). In particular, we explain how to choose the most appropriate widgets, how to use each of these effectively, and how to combine them to best effect. We do this by looking in detail at an extended example based on a GUI for booking facilities at a sports center.

In Chapter 17, we show you how to design good web sites. In particular, we consider the structure of a site, the design of individual pages, and how to write the content. We do this by looking in detail at an example based on a web site for a hotel room reservation system.

Chapter 18 looks at embedded systems and small devices such as mobile phones. We consider safety-critical systems to be a type of embedded system. No new information on safety-critical systems is given, but we do provide an exercise that reinforces the relevance of the design issues, principles, and guidelines introduced in Chapters

8 through 15. We also consider a type of embedded system known as the information appliance. Chapter 18 finishes by looking in more detail at generic aspects of embedded systems, considering the issues of portability, general purpose versus specialized devices, connectivity, and the commercial environment.

Chapter 19 illustrates how the requirements were gathered and the design and evaluation of the UI for the Final Approach Spacing Tool (FAST) were undertaken by the research and development group at National Air Traffic Services (NATS). The system was developed to support air traffic controllers in their task of managing air traffic safely, efficiently, and effectively.

1.2 Learning Outcomes

After studying Part 3, you will be able to:

- Reflect upon how a new UI can better support the users' work
- Allocate tasks to either the user or the computer system for a new UI
- Create a content diagram that represents the underlying organization and structure of a new UI
- Critically evaluate a UI against the design principles of simplicity, structure, consistency, and tolerance
- Choose appropriate metaphor(s) for a UI, taking into account the potential difficulties associated with the use of metaphors
- Identify the most appropriate way for the user and computer to interact from a range of possible interaction styles
- Choose one or more input and output devices that match the requirements and constraints of the particular task, user group, and environment
- Design a UI that makes effective use of text, color, images, moving images, and sound
- Design a usable UI that combines input and output devices, text, color, images, moving images, and sound, as appropriate
- Design a GUI
- Design a web site
- Explain some considerations in the design of embedded systems

1.3 Theoretical Influences

Part 3 draws from:

- Cognitive psychology in the chapters on the human–action cycle, communicating the designer's understanding of the system, and how to use design components effectively
- Object-oriented software engineering in explaining the conceptual design process
- Graphic design and multimedia theory in the chapter on choosing text and color, sound, images, and moving images
- The experience of practitioners in creating designs

8

Work reengineering and conceptual design

1 Introduction

This chapter describes an approach to bridging the gap between gathering the requirements and creating the **physical design** of a user interface (UI). When you start planning the design of a UI, you want to improve on the design of the existing system if there is one. To maximize the benefits of developing a new UI, it is important to consider the issue of **work reengineering**. With the new UI, it may be necessary for people to work differently if they are to work effectively. This will require sensitive handling, which is another reason for involving the users in the whole of the development process: if the suggestions you make are likely to be unpopular or are really unreasonable, the users will soon tell you.

According to Mayhew (1999), the process of work reengineering has three goals:

> *Realizing the power and efficiency that automation makes possible.*
>
> *Re-engineering the work to more effectively support business goals.*
>
> *Minimizing retraining by having the new product tap as much as possible into the users' existing task knowledge, and maximizing efficiency and effectiveness by accommodating human cognitive constraints and capabilities within the context of their actual tasks. (p. 172)*

We shall focus on one particular aspect of work reengineering: task allocation. One of the most important decisions to be taken during the development of a UI is to allocate the tasks to the user or to the computer. The designer needs to establish who (or what) will provide the data or knowledge necessary to accomplish a task and who (or what) will physically accomplish the task.

It is important to maximize the strengths of both the user and the computer. For example, in the UI for a bank ATM, it is difficult to imagine how the computer could provide the PIN. This must be a user action in order to meet security criteria. Similarly, you would not expect a user to have a precise knowledge of her or his bank balance before deciding whether to make a withdrawal. The computer should

Box 8.1

The Equivalent of "Work" When the UI Is for Home or Leisure

This chapter concentrates on the introduction of a new UI for a system that is used in a workplace. There are equivalent concepts for a system that is used for leisure. The "work" is whatever the users want to do with the system — for example, play a game or purchase a product. It is just as important to consider the reactions of the users, as this story illustrates:

"A few years ago, when eBay designers opted to change the background color of every page from yellow to white, they actually changed the shade of the pages in almost imperceptible increments daily for months, until the pages were finally white. 'If they'd flipped a switch and gone from yellow to white, everyone would have screamed,' said eBay senior usability engineer Laura Borns. 'But no one had any complaints about it, because it was so gradual.' "

From Mathew Schwartz, "Grow Your Site, Keep Your Users," Computerworld, June 4, 2001, www.computerworld.com/developmenttopics/websitemgmt/story/0,10801,60997,00.html, visited July 8, 2004.

provide this information. Essential use cases can be a useful tool for thinking about the task allocation process.

1.1 Introduction to the Digital Library

A university department has academic and research staff and research students, all with various teaching and research activities. All staff and research students maintain personal libraries of books, CD-ROMs, videos, and journals related to their particular research interests. As they are all in the same department, they often find that someone else in the department has interests that overlap with their own.

To make the personal libraries of the department's members accessible, the department has decided to design and develop an online digital library. It will keep track of the personal resources of each member of the department. As in an ordinary library, borrowers will be able to search its database for items of interest. Unlike in an ordinary library, items are not owned by the library but by individuals, who may need constant access to them. Therefore, for each request the owner of an item will need to agree to lend it to the borrower.

A member should be able to search the digital library for the item he or she needs. The system will therefore need to keep details of each resource, such that its owner can be contacted easily. The system will also need to be regularly updated, as members acquire new resources, join or leave the department, or change their contact details. It is the responsibility of individual members to keep their details up to date. The library will run on the department's intranet.

We shall go step-by-step through a simple example of work reengineering and task allocation, using the digital library as detailed. This involves use scenarios and essential use cases. In Chapter 4, we said that use scenarios are similar to task scenarios, but that they describe the anticipated use of the new UI rather than the use of the current UI. We start by showing how the use scenarios are likely to be quite different from the task scenarios, drawing out the impact on the working practices of the members of the computing department. We then show how to elicit essential use cases from the use scenarios, and we consider how these useful tools can help us think about the task allocation. Finally, we derive some concrete use cases from the essential use cases. These will act as a bridge to the conceptual design section, which follows.

2 Work Reengineering for the Digital Library

Figure 8.1 contains some task scenarios for the present situation within the computing department. There are likely to be many more. As you can see, the situation is rather unsatisfactory.

Task scenario. Search and request resource

Julia, a lecturer in the department, is looking for a particular CD-ROM containing examples and exercises on Object Oriented Analysis and Design. She knows that Tom, another lecturer, mainly teaches Object Oriented Analysis and Design so she knocks on his door. Unfortunately he is not there, so she leaves a note on his door. Later he returns and searches for her, finding her in the coffee bar. He tells Julia that Geoff has the CD-ROM. Unfortunately Geoff is on leave, so Julia telephones him and he promises to post it to her.

Task scenario. View updates and request resource

Mark has recently returned from six months of study leave and wants to find out what books other members of the department have bought since he left. To do this he telephones everyone in the department and arranges an appointment. He has to do this because everyone is at the university at different times. He then meets everyone individually and checks through their bookcases, asking to borrow books that interest him. He only asks for one book at a time, as he is a slow reader!

Figure 8.1 Task scenarios for the digital library.

As part of the redesign, we have drafted some use scenarios in Figure 8.2 to illustrate how the digital library might operate. Notice that there are no details about the

Use scenario. Search and request resource

Julia is looking for a particular CD-ROM containing examples and exercises on Object Oriented Analysis and Design. She accesses the digital library from home and types in the key phrase 'Object Oriented Analysis'. The system retrieves one result. Geoff owns the appropriate CD-ROM. Julia then sends an e-mail to Geoff, asking to borrow the CD-ROM.

Use scenario. View updates and request resource

Mark has recently returned from study leave and wants to find out what are the latest additions to the digital library. He selects 'check updates', identifies the books he is interested in, and sends an e-mail to the owner of the one that interests him most.

Figure 8.2 Use scenarios for the digital library.

technology that will be used, because the focus is on the users and how they will carry out their tasks. In reality, we would develop more use scenarios than this, analyzing their particular strengths and weaknesses in terms of their implication for the users.

EXERCISE 8.1 (Allow five minutes)

In what respects will the digital library be better than the present arrangements for Julia and Mark? In what respects will it be worse? What disadvantages will there be for all the members of staff who are members of the digital library?

DISCUSSION

For Julia. *Advantages:* She will not be dependent on other lecturers being in their offices, as the digital library will be available all the time. She will know for sure who has a particular resource, rather than having to guess. She can find all the information she needs and request the resource from her desk. *Disadvantages:* She will not have an excuse for a chat, and maybe coffee, with her colleagues. This may seem trivial, but important work-related discussions can occur in this sort of spontaneous manner. Such social contact can also be important if staff members are to avoid becoming isolated.

For Mark. *Advantages:* The digital library will save the huge amount of time currently spent meeting other staff members. Other staff members will not need to go through their shelves to help them remember what they have purchased in

the previous 6 months. *Disadvantages:* Again, there will be a loss of social contact with other members of the department. Also, Mark will lose the opportunity to find out what others think of a particular resource (this could be an option with the digital library, but people tend to be more honest face-to-face).

Generally. All the members of the digital library will need to enter the details of the resources they purchase. This will take time, and keeping the library up to date will take a lot of effort. Also, members may not add all their purchases to the library, particularly those they find useful. In this case, the library may be sparse and contain only the resources that no one wants.

3 Task Allocation for the Digital Library

Once we have decided on some use scenarios, we need to agree how to share the different tasks between the user and the computer. This is referred to as **task allocation**. We can specify the task allocation using essential use cases. An essential use case corresponding to the "Search and request resource" use scenarios is shown in Figure 8.3. As you can see, this does not contain any specific details, such as the technology used. For example, the search parameters could be entered using a keyboard or by voice recognition. In addition, it does not include the details of the task. For example, the essential use cases do not indicate the types of resources available or the details of the search parameters. Instead, they concentrate on showing how the tasks will be shared between the user and the system.

In this example, we have presented the use scenarios before the essential use cases, but in reality you will probably develop both in parallel. As you can see, the "Search and request resource" essential use case maps very simply onto the corresponding use scenario. This is because as we were developing the use scenarios we had an increasing sense of the form the essential use case would take. Similarly, your understanding of the tasks typically carried out by the user or system will influence the wording of the use scenarios. As with all aspects of UI development, the different activities are closely linked, and there is no single approach that will work in all

User's purpose	System responsibility
Enter search parameters	Show results
Select a resource	Show the contact details of the owner of the selected resource
Send an e-mail	Confirm the send

Figure 8.3 "Search and request resource" essential use case.

situations. The answer is always to involve the users and be prepared to keep modifying your ideas until you identify the best solution.

EXERCISE 8.2 (Allow five minutes)

Draw the essential use case for the "View updates and request resource" use scenario.

DISCUSSION

The "View updates and request resource" essential use case is illustrated in Figure 8.4. As you can see, there is a close correlation with the use scenario.

User's purpose	System responsibility
Request latest updates	Show results
Select a resource	Show the contact details of the owner of the selected resource
Send an e-mail	Confirm the send

Figure 8.4 "View updates and request resource" essential use case.

4 Conceptual Design

Conceptual design is the process of establishing the underlying organization and structure of a UI. This is an important activity, as it makes little sense to design screen layouts before you have decided what functions the screen should support. This information is represented in a content diagram. A **content diagram** is a low-fidelity prototype that represents the organization and structure of the user interface from the designer's perspective.

The network comprises nodes, referred to as **containers**, and links. Each container is an abstract representation of a part of the user's work and the functions that are required to do that part of the work. For example, a container in the digital library system may correspond to entering the search criteria for a book. The links represent how the user will navigate between the functional areas within the UI, thus showing how the functional areas need to be structured. Figure 8.5 illustrates the simplified outline of a typical content diagram.

The relation between the content diagram and the final UI will vary. In a web site, for example, each container may become a screen, and the links may become navigation elements such as hypertext links, selectable areas, or menus. Alternatively, in a

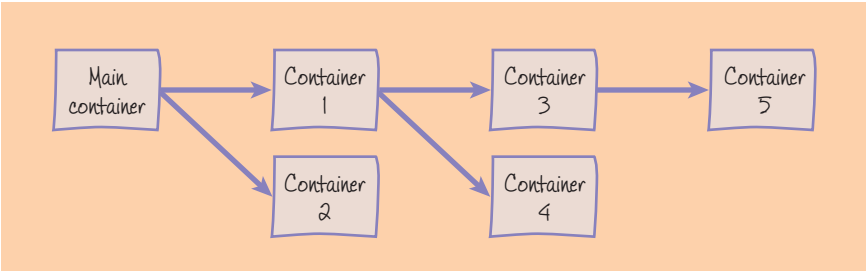

Figure 8.5 Simplified outline of a typical content diagram

graphical user interface (GUI), the containers may become windows, dialog boxes, or message boxes, and the links may become buttons and menu items. However, if you were designing a purely sound-based system, such as a voice messaging system, then the containers would become clusters of menus and their associated responses.

In some cases, the relation between the structure of a content diagram and the structure of the UI may appear less close. For example, the navigation around the UI may be slightly different, or several containers may be combined to form a single screen. The Lovely Rooms Hotel group example in Chapter 17 illustrates the former. Figure 8.20, later in this section, illustrates the latter. Thus, you should not allow the content diagram to limit your creativity when you are designing a UI.

A content diagram is based on both the findings of the requirements gathering and the concrete use cases derived from the essential use cases used during task allocation. As it is unlikely that you will have identified all the possible concrete use cases, the content diagram will probably be incomplete. However, it is still valuable for identifying the main functional areas in a UI and the essential relations between them.

To produce a content diagram, you need to do four things:

1. Derive the concrete use cases from the essential use cases.
2. Identify the primary task objects, attributes, and actions.
3. Identify the different containers and the task objects that go into each one.
4. Link the containers to show the navigation flow.

Experienced UI designers may not complete these activities in order and may omit some activities completely. Others may use less formal approaches that involve discussing diagrams drawn on flipcharts or whiteboards. As you are new to UI design, however, we recommend that you attempt the activities we describe here: they will make you aware of the issues you need to consider when creating a conceptual design.

Completing these activities is a creative process, and developing the best solution is a process of informed trial and error. To help you with this, we suggest you use some

low-fidelity prototyping tools, such as sticky notes on flipchart paper. These can be useful for the following reasons:

- Sticky notes are flexible. They can be moved around easily and discarded if necessary. This makes them ideal for experimenting with designs that may change.
- The very fact that a sticky note does not look like a UI control is a constant visual reminder that the representations generated during conceptual design are abstract. This will help you to focus on the structure of the user interface rather than on its visual appearance.
- A sheet of flipchart paper can be attached to the wall. This means that if you are working with other people, everyone in the team can see the design.

We have formulated this approach to conceptual design from a number of sources, in particular Weinschenk *et al.* (1997) and Constantine and Lockwood (1999). There is no single universally accepted approach to conceptual design. We believe the one we present here draws together many of the best features of the alternatives available.

4.1 Deriving Concrete Use Cases from Essential Use Cases

Figure 8.6 illustrates a concrete use case derived from the "Search and request resource" essential use case illustrated in Figure 8.3. This is just one of many alternatives. For example, we could have had an academic searching for a book, a

User action	System response
The academic enters one or more of the search parameters for the CD-ROM: title, year and platform	The system displays the search results
The academic selects a search result	The system displays the full details of the CD-ROM and the contact sdetails for its ownerm who is a research student
The academic chooses the e-mail address	The system displays a message area
The academic writes and sends the e-mail request	The system confirms the sending of the request

Figure 8.6 Concrete use case: "Search and request CD-ROM."

researcher searching for a journal, and so on. You should be able to see that we have added detail to the essential use case and altered the column headings, showing that we are moving toward the final design. The level of detail will vary between concrete use cases, but we suggest that you avoid any implementation details, such as references to screen elements or particular types of hardware or screen layouts. This is not necessary for conceptual design and may limit your creativity later in the process.

EXERCISE 8.3 (Allow 10 minutes)

Illustrate a concrete use case corresponding to the "View updates and request resource" essential use case illustrated in Figure 8.4.

DISCUSSION

Our solution is illustrated in Figure 8.7. Yours may contain a few differences; perhaps you included an academic rather than a research student or had the member choose a CD-ROM rather than a book. These alternatives are all correct.

4.2 Identifying Task Objects, Attributes, and Actions

See Macaulay (1995) for more about an object-oriented approach to UI design.

The next step is to identify the **task objects**, their attributes, and the actions the user will perform on them. These details will influence what goes into each of the containers in the content diagram and the links needed between the containers. We do not teach object-oriented design in detail, but the information in this section should provide you with sufficient understanding to get started.

User action	System response
The research student requests recent updates in the digital library	The system displays the availability of the latest books, CD-ROMs, videos and journals
The student selects this year's book by her favourite author: J. Nielsen	The system displays the full details of the book and contact details, including name, for the owner, who is an academic
The student chooses the e-mail address	The system displays a message area
The student writes and sends the e-mail request for the book	The system confirms the sending of the request

Figure 8.7 Concrete use case: "View updates and request book."

We start by explaining what task objects, attributes, and actions are. We then show you how to identify each from a concrete use case, and finally we discuss how to prototype your ideas.

▶ Task Objects

Primary task objects are the units of information or data with which the users interact to carry out their tasks. They are high-level objects, central to the tasks the users will be carrying out. Typically there are only a few primary task objects, and they are easy to identify. For example, if you were designing a UI for a hotel registration system, there would probably be only two primary task objects: one corresponding to the customer, the other to the room. For brevity, from now on we shall refer to primary task objects as task objects.

When you are identifying the task objects, it can be helpful to check the requirements documentation and concrete use cases for the new UI. In particular, you should check for units of information that are searched through or modified in some way. These may include artifacts, such as forms, documents, papers, and lists.

These task objects will typically be translated onto your UI design as combinations of user interface objects, such as screens, windows, dialog boxes, pull-down menus, icons, combo boxes, and so on, as you will see in Chapter 16. In embedded systems, such as mobile telephones, these may take the form of physical buttons and other simple input devices, plus output on the screen.

EXERCISE 8.4 (Allow five minutes)

List the task objects for the digital library. You will need to check the description in the introduction to the digital library presented earlier in this chapter.

DISCUSSION

The task objects we identified were book, CD-ROM, video, journal, academic staff, research staff, and research student.

It is sometimes possible to group task objects into classes. In the digital library, resource is a **class**, containing book, CD-ROM, video, and journal. Resource is an abstraction of the resource types. In other words, it contains the common elements of those types. In particular, this means the attributes that are common to all the resource types, such as keywords, title, and author.

Classes are a type of task object, because users can interact with them. For example, in the digital library, the member may want to search through all the resources. Identifying classes can help guide the decisions you make when you are designing a UI. The member could search all the resources in the digital library, using separate search screens for book, CD-ROM, video, and journal. This would be repetitive, but it would allow searches on all the attributes. A simpler solution may be to have a single screen that corresponds to the resource task object. This would contain just the common

attributes. This solution would be less powerful, but it may well be sufficient to meet the needs of the member. It may be possible to group classes into higher-level classes, thus creating a hierarchy.

EXERCISE 8.5 (Allow two minutes)

Identify a second class of task objects within the digital library. List the members of this class.

DISCUSSION

Member is a class containing academic, researcher, and research student.

▶ Attributes

A task object must have attributes. If a task object does not have any attributes, it is not an object in its own right but rather the attribute of another task object. For example, for the hotel registration system the number of people who can occupy a hotel room is an attribute of the room object rather than an object in its own right. There are two kinds of attributes:

- *Properties*. For example, in the digital library title and author are properties of the book task object.
- *Child objects*. These are task objects in their own right. For example, an attribute could indicate who owns a CD-ROM. The owner could be an academic, making the academic task object a child object of the CD-ROM task object. It is a task object in its own right because it has its own attributes, such as the name of the academic, his e-mail address, and so on.

This relation can influence the design process, as visual containment results when an attribute is a child object. In user interface design, this means that when the task object is displayed on a screen, the child object will also be displayed on the same screen. Depending on the implementation, either the whole object will be displayed or only a summary. An example of the latter would be if the screen that displays the details of a book has a line indicating the name of the person who owns it. Double-clicking on this may display another screen containing the person's details.

▶ Actions

When users carry out their tasks, they perform various actions on the task objects. For example, in the hotel registration system, the receptionist will want to allocate guests to rooms. This means the room task object will need to have a corresponding allocation action.

You can identify these actions by reviewing the concrete use cases. In addition, you should consider standard actions such as view, create, delete, copy, save, edit, and print.

4.3 Marking Up the Concrete Use Cases to Identify Task Objects, Their Attributes, and Actions

A useful technique for identifying task objects and their attributes is to mark up the concrete use cases. We suggest the following markup convention:

- Single-underline nouns that you think may correspond to task objects.
- Double-underline the attributes of these task objects.

Verbs in use cases often correspond to actions. We do not suggest marking these up as the relationships are often less direct. However, identifying the verbs can still be useful.

By way of illustration, we have marked up the "Search and request CD-ROM" concrete use case. The result is shown in Figure 8.8.

You can see from the markup that we have identified the academic, research student, and CD-ROM task objects. In addition, by implication we have identified e-mail, a further task object. This is not mentioned explicitly, but two of its attributes are e-mail address and message area. We have also identified the title, year, and platform attributes of the CD-ROM task object.

You are unlikely to identify all the task objects and attributes in this way, unless the situation is particularly simple or you use a great many concrete use cases, but it should give you most of them.

User action	System response
Academic enters one or more of the search parameters for the CD-ROM: title, year and platform	The system displays the search results
The academic selects a search result	The system displays the full details of the CD-ROM and the contact details for its owner, who is a research student
The academic chooses the e-mail address	The system displays a message area
The academic writes and sends the e-mail request	The system confirms the sending of the request

Figure 8.8 Marked-up concrete use case: "Search and request CD-ROM."

EXERCISE 8.6 (Allow five minutes)

Identify the task objects and attributes for the "View updates and request book" concrete use case in Figure 8.7.

DISCUSSION

We have illustrated our solution in Figure 8.9. As you can see, we have identified the research student, academic, book, CD-ROM, video, and journal objects, plus a number of their attributes.

Once you have identified the task objects and attributes, it is helpful to compile them, along with the actions, into a single object–action–attribute table. Table 8.1 contains the CD-ROM task object from the digital library. As you can see, some of the attributes come directly from the concrete use cases, but others come from the domain analysis. The "owned by" attribute corresponds to the academic, researcher, and research student child objects.

The actions are the standard actions, plus the reserve action, which allows the member to indicate that the CD-ROM has been reserved. You should note that the actions relate to the CD-ROM details in the digital library system, rather than to the CD-ROM itself. Thus, the table indicates that it is possible to edit these details, but it does not say it is possible to alter the CD-ROM itself.

User actions	System responses
The research student requests recent updates in the digital library	The system displays the availability of the latest books, CD-ROMS, videos and journals
The student selects this year's book by her favourite author: J. Nielsen	The system displays the full details of the book and the contact details, including name, for the owner, who is an acedemic
The student chooses the e-mail address	The system displays a message area
The student writes and sends the e-mail request for the book	The system confirms the sending of the request

Figure 8.9 Marked-up concrete use case: "View updates and request book."

Table 8.1 Object–Attribute–Action Table CD-ROM Task Object

Task object	Attributes	Actions
CD-ROM	Keywords Title Author Year Platform Owned by (academic, researcher, or research student)	View Add Print Delete Save Reserve Edit

Table 8.2 Object–Attribute–Action Table: Academic Task Object

Task object	Attributes	Actions
Academic	Name Phone number Office number E-mail address	View Add Edit Print Save Delete

EXERCISE 8.7 (Allow 10 minutes)

Draw an object–attribute–action table for the academic task object. You will need to make some assumptions, as not all the necessary information is in the concrete use cases. Explain how you identified the attributes and actions.

DISCUSSION

Table 8.2 shows the table we developed. The name and e-mail address attributes have come from the "View updates and request book" concrete use case in Figure 8.9. There are no child objects, as academic is a child object of the various resource type objects: book, CD-ROM, video, and journal. They are all standard actions.

4.4 Prototyping Task Objects, Attributes, and Actions

Marking up the concrete use cases is one approach to identifying task objects and attributes, but this should only act as a starting point. As you saw earlier, some of the task objects, attributes, and their actions may not necessarily come from the con-

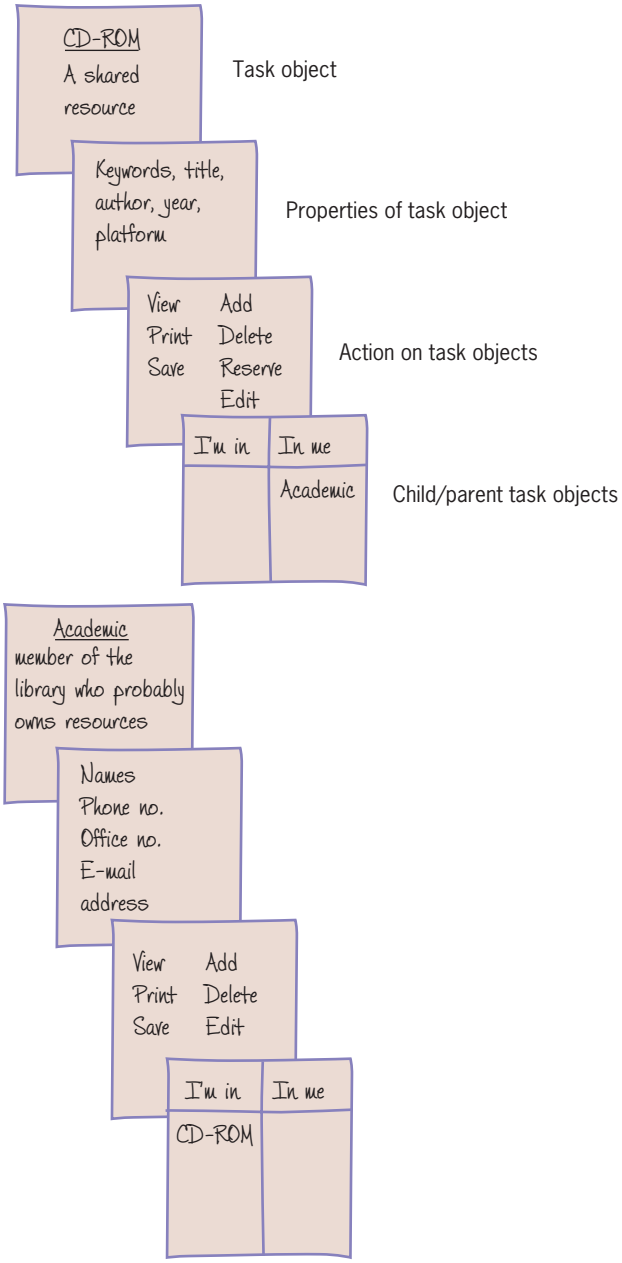

Figure 8.10 Using sticky notes to prototype task objects, attributes, and actions.

crete use cases but rather from the users' knowledge of the domain or from your own domain analysis. Consequently, the best approach is to prototype your ideas, working alongside potential users of the system.

One approach is to use sticky notes. Write a one-sentence definition of the task object on one sticky note, the task object's properties on another, and the actions that can be applied to it on a third. We have discussed two kinds of attributes: child objects and properties. Divide a fourth sticky note into two columns, one headed "In me" and the other headed "I'm in"; write any child objects under the column "In me" and any parent objects under "I'm in." Once you have done this for each task object, you can stick these on a sheet of flipchart paper. Figure 8.10 illustrates some sticky notes developed as part of the conceptual design process for the digital library system.

You can check the completeness and correctness of the paper prototype by walking through a selection of concrete use cases with the users and verifying that the identified task objects, attributes, and actions will satisfy the tasks they plan to carry out.

4.5 Creating the Content Diagram

The content diagram represents the underlying organization and structure of the UI. It is a network made up of containers and links. The next step is to identify the containers that are needed for the content diagram. Each container collects functions and task objects into a coherent place in the system to support a part of the user's work. Typically, these become screens, windows, dialog boxes, or message boxes in the UI. We also need to identify the links between the containers. These indicate the navigation around the UI.

▶ Template for Containers

Figure 8.11 shows a template for containers.

As you can see, each container has the following elements:

Name — The name you choose for this container

Purpose — A phrase indicating its purpose in supporting the user's task.

Functions
- Indicates functions that are invoked by the user to perform the work.
- Indicates functions that are automatically invoked by the system.

Links — The links with other containers, indicating the name of the container linked to and its purpose. There are two types of **link**.
Single links ▸ A **single link** indicates that the user moves to another container and then that new container becomes the focus of the user's activities. For example, when you login to your PC, you get a login dialog box. After you enter your login name and password, the system validates it and then shows the main desktop window.
Double links ▸▸ A **double link** indicates that the work done in a second container needs the context of the first container and that the user will switch back and forth between the two; an example is

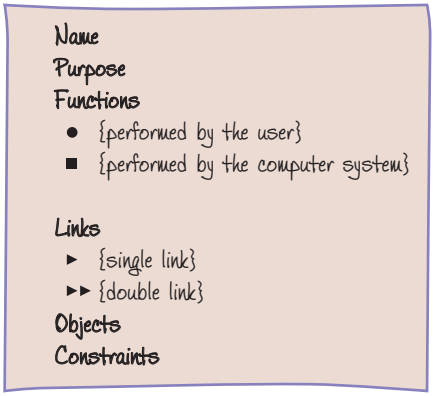

Figure 8.11 Template for containers.

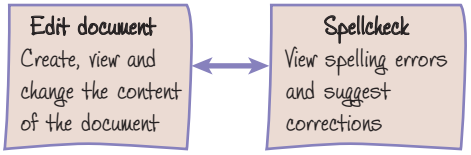

Figure 8.12 An example of a double link between containers.

when you are working in a Word document and you invoke the spell checker. This is a double navigation link because the focus keeps moving between the two windows when you perform the task. Figure 8.12 illustrates this situation. The containers are illustrated in outline form.

- **Objects** The task objects whose attributes and actions are required for the users to complete their tasks.
- **Constraints** Any constraints for that container, such as speed, reliability, and availability.

The Main Container

To find out more about launch pad windows, see Chapter 16.

The first container we need to specify is the **main container**. This represents the first thing the users encounter and will be central to their work. In a GUI, this might be a launch pad window containing icons corresponding to the main tasks that can be carried out.

When you are designing the main container, you will typically include links to the following:

- *Vital tasks.* The user must perform these tasks quickly, even under stress. An example would be a network supervisor who continually monitors the traffic on a network. If something goes wrong (for example, if someone cuts a cable), it is important that the supervisor is able to assess the situation quickly and reroute the traffic.

- *Frequent tasks.* Those tasks that users spend the majority of their time performing must be fast to access. For example, the software used by the telephone operators who deal with emergency calls needs to offer good support for the limited number of tasks that the operators perform during each call.
- *Navigational aids.* The users need to understand quickly and easily what the application is capable of doing and how to accomplish their tasks. For example, information kiosks need to be easy to learn and use without written instructions. This might be achieved using a map or some other metaphor.

Typically the section of the UI corresponding to the main container will not perform any of these tasks. Instead, it will provide links to the containers that do. The main container for the digital library system is illustrated in Figure 8.13. This container lists functions that correspond to the most frequent tasks, as elicited by the requirements gathering:

- To search for a resource, identify who owns it, and send an e-mail to that person.
- To identify the current updates to the digital library, choose a recently added resource, and e-mail the owner.
- To contact the system support team.

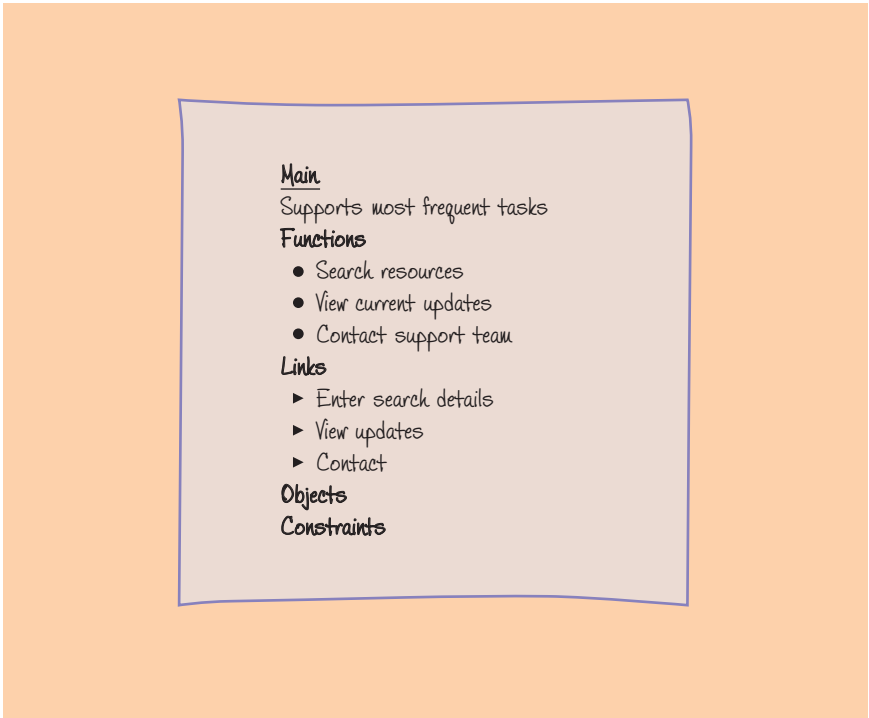

Figure 8.13 The main container for the digital library system.

There is a corresponding link for each of these functions. As the purpose of the main container is to access other containers, it does not contain any of the functionality or the details relating to any of the task objects. There are no constraints, apart from the obvious ones, such as reliability and availability. We have not included these, as they apply to all the containers.

This container does not represent all the information needed to start developing the physical design of the screen, but it does represent the essential elements and helps us to focus our thinking on the other containers that will be needed.

▶ Other Containers

Other containers are usually derived from the concrete use cases. Each concrete use case shows the sequence of steps needed to accomplish a particular task. The functionality needed to support these steps can be divided between one or more containers.

Consider the "Search and request CD-ROM" concrete use case in Figure 8.6. The first user action is to enter the search criteria. Rather than have separate search screens for each of the different resource types, we decided to have a single search screen allowing the user to search through all the resource types. The corresponding "Enter search criteria" container for the digital library system is illustrated in Figure 8.14.

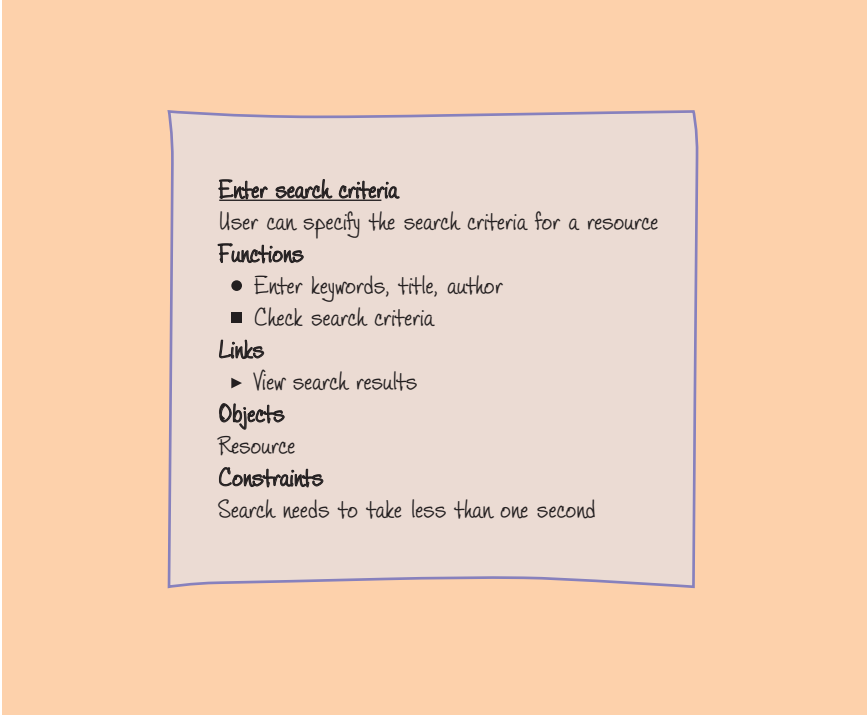

Figure 8.14 The "Enter search criteria" container for the digital library system.

The purpose of this container is to allow users to specify the search criteria. From the functions, we can see that the UI is responsible for checking that the search criteria are syntactically correct. There is only one link: this is to the container that displays the results of the search. Unlike the main container, "Enter search criteria" relates to a task object: the resource object. In the UI, the attributes common to all the resource types will form the search criteria. For example, the corresponding screen might be as illustrated in Figure 8.15.

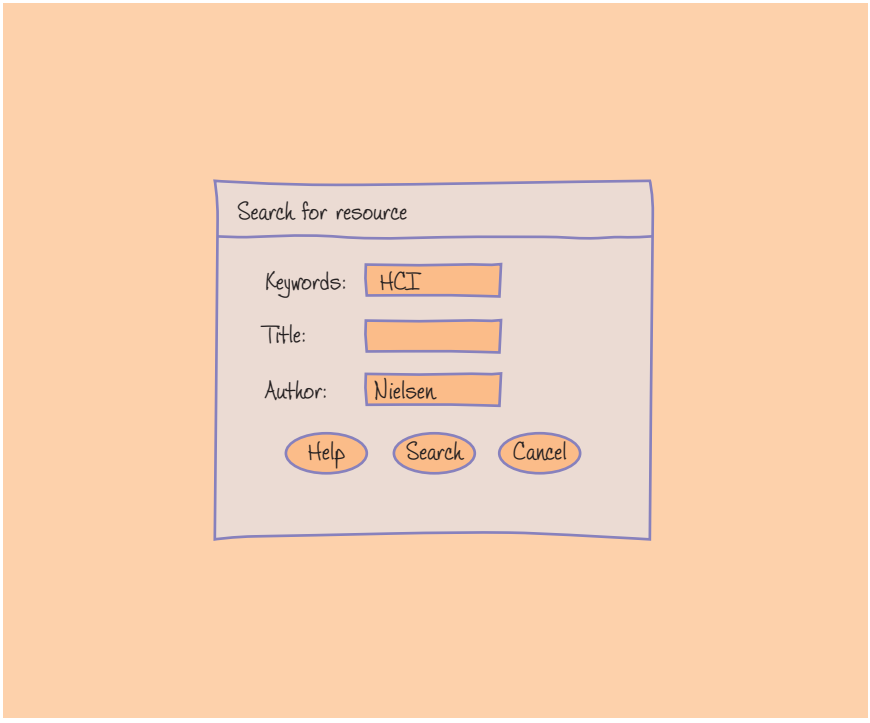

Figure 8.15 A possible screen design for the "Enter search criteria" container.

This is a very limited implementation. It may have been better to have included additional fields corresponding to the attributes that are particular to the different resource types, such as the length of the video or the platform needed for the CD-ROM (Mac or PC).

Ideally you should not anticipate the more detailed aspects of your UI design during the conceptual design phase, as it can constrain your thinking; nevertheless, it can help you if you are inexperienced.

EXERCISE 8.8 (Allow 10 minutes)

Illustrate the "View search results" container, identifying the functions, links, and associated task object(s). Explain the decisions you have made and draw a simple screen mockup, showing how the screen might look. How would you

implement the container for an audio output UI, such as a telephone menu system?

DISCUSSION

Figure 8.16(a) illustrates the container we designed. In our design, the "View search results" container shows the results of the search, allows users to select the one they want, and then allows them to move on to the next container, so that they can view the contact details. The resource object is associated with this container because the attributes will occupy most of the screen, as Figure 8.16(b) illustrates. The screen is simpler than it will be in the final design, because it does not contain information such as the resource type for each title.

If this container were to be implemented using an audio-output UI, each result would need to be read out, with an associated number. Users would then choose the number of the result for which they require the contact details and either speak the number or press the appropriate key on the telephone keypad.

▶ Links

Once we have identified the containers we need to link them together to reflect the navigation flow, the way in which the users will move through the UI to achieve their goals. We refer to the resulting diagram as the content diagram; it is the outcome of the conceptual design process. For clarity, we have presented the steps of identifying and linking the containers as separate stages, but in reality you would develop the two in parallel.

When you are drawing the content diagram, draw single links as single-headed arrows ▸ and double links as double-headed arrows ▸▸. These arrows correspond to the links each container indicates are necessary. If any conditions determine the navigation flow to a particular container, label the arrow. Such labels are known as **conditions of interaction**. These links and the conditions of interaction in the content diagram will help you determine whether the user interface architecture supports the users' tasks as they navigate from one container to another.

Identifying the links should not be too difficult, as they will reflect the order of the actions in the concrete use cases. However, producing the content diagram for a complex UI is not a trivial task, as it will probably involve combining a large number of concrete use cases into a complex network of containers and links.

Figure 8.17 illustrates the whole of the relevant section of the content diagram corresponding to the "Search and request CD-ROM" concrete use case in Figure 8.6. As you can see, there is a condition of interaction between View details and Write e-mail message. This is necessary because the user may choose to telephone the owner or to knock on the owner's office door rather than to send an e-mail.

▶ Prototyping Containers and Links

Because of the potential complexity of the task, it is important to prototype your ideas when you are developing the content diagram. As with the prototyping of

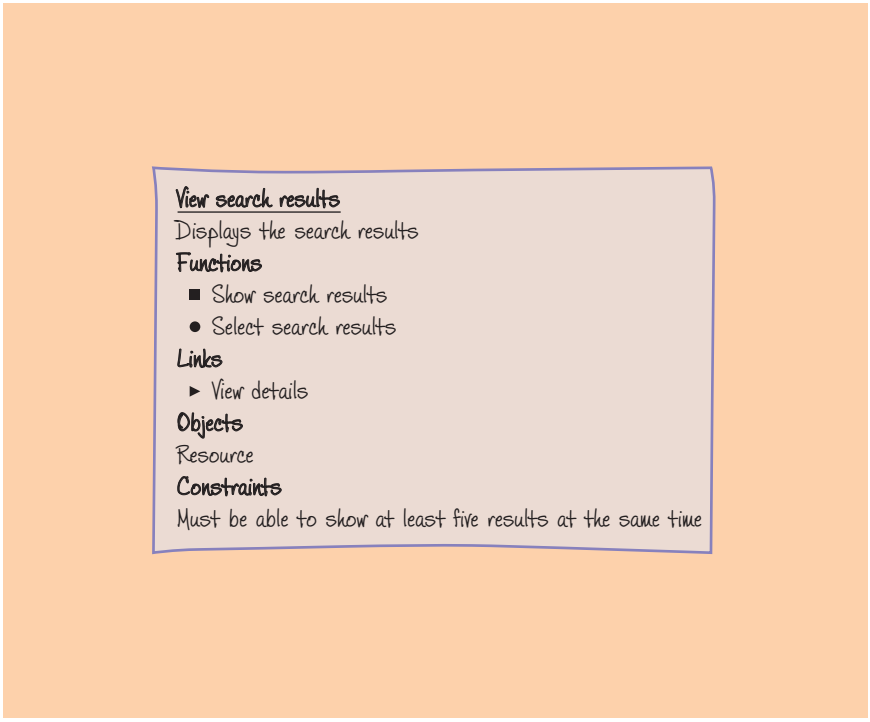

Figure 8.16 (a) The "View search results" container for the digital library system. (b) A possible screen design.

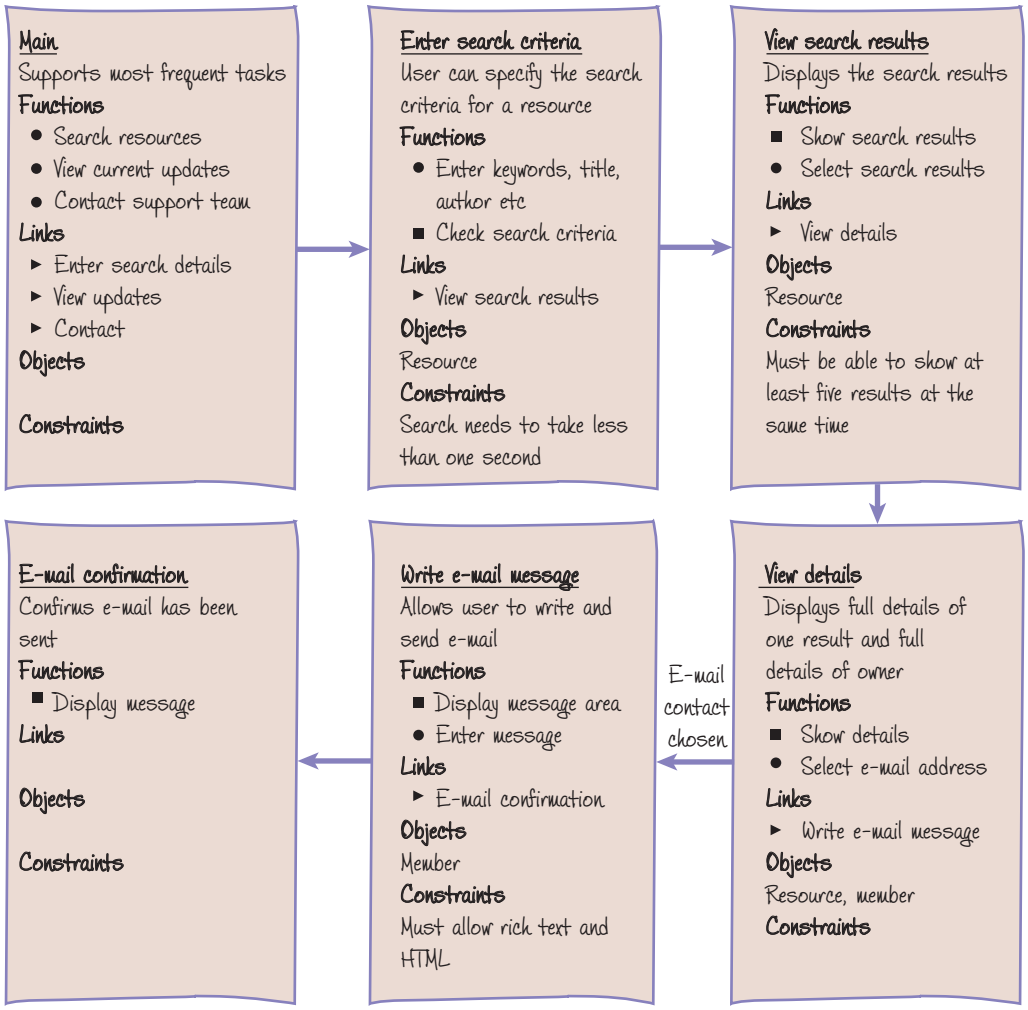

Figure 8.17 Section of the content diagram corresponding to the "Search and request CD-ROM" concrete use case.

task objects, attributes, and actions, one way of doing this is to put sticky notes on a flipchart sheet. Each sticky note should represent a container or a link. You do not need to add all the details for the containers at this stage; just the name and purpose will be sufficient. You can then move the sticky notes around, adding and removing containers and links as appropriate. This representation can help you discuss matters with the users. In particular, you can step through some of the concrete use cases to ensure that the containers represent the necessary functionality and that the links allow for all the sequences of actions needed to carry out the users' tasks.

EXERCISE 8.9 (Allow 10 minutes)

Draw the section of the content diagram corresponding to the "View updates and request book" concrete use case in Figure 8.7. You should base the content of the containers upon the task objects, attributes, and actions you identified earlier. As for the content diagram corresponding to the "Search and request CD-ROM" concrete use case, assume that the implementation will be at resource level rather than having separate update screens for books, CDs, videos, and journals.

DISCUSSION

Figure 8.18 illustrates the section of the content diagram corresponding to the "View updates and request book" concrete use case. As you can see, four of the containers are the same as those in Figure 8.17.

▶ Final Thoughts on Conceptual Design

Figure 8.19 illustrates the whole content diagram based on the two concrete use cases. For brevity, we have given only the titles of the containers. We have combined the containers common to the two content diagrams illustrated in Figures 8.17 and 8.18. You can see how the network of containers is starting to evolve.

Transforming the content diagram into a user interface is a creative activity. There are no strict mappings from the content diagram onto the UI design, but for GUIs and web sites, most containers will translate into screens, windows, or dialog boxes. The task objects and their attributes will become UI controls such as list boxes, combo boxes, radio buttons, data fields, and so on, and the actions will become menu items or items on the tool bar in the windows. The navigation from one container to another will typically be achieved by selecting a link, tool bar button, and so on.

The navigation flow, as represented by the links, will underlie any corresponding UI, but the relation may not be as straightforward as you expect. It is sometimes possible to have a one-to-one relation, but that may not be ideal. For example, Figure 8.20 illustrates a web page that corresponds to three containers: the main container, the "Enter search criteria" container, and the "View updates" container. Thus, we have translated three containers into a single screen while maintaining the same navigation flow. Similarly, a single container may correspond to several screens.

In Chapters 16 and 17, this process of transformation is demonstrated for graphical user interfaces and web user interfaces. As you will see, the content diagram informs the UI design, but there are still numerous design decisions to make.

5 Summary

In this chapter, we discussed work reengineering and conceptual design. Work reengineering involves deciding how a UI will help users carry out their tasks. An

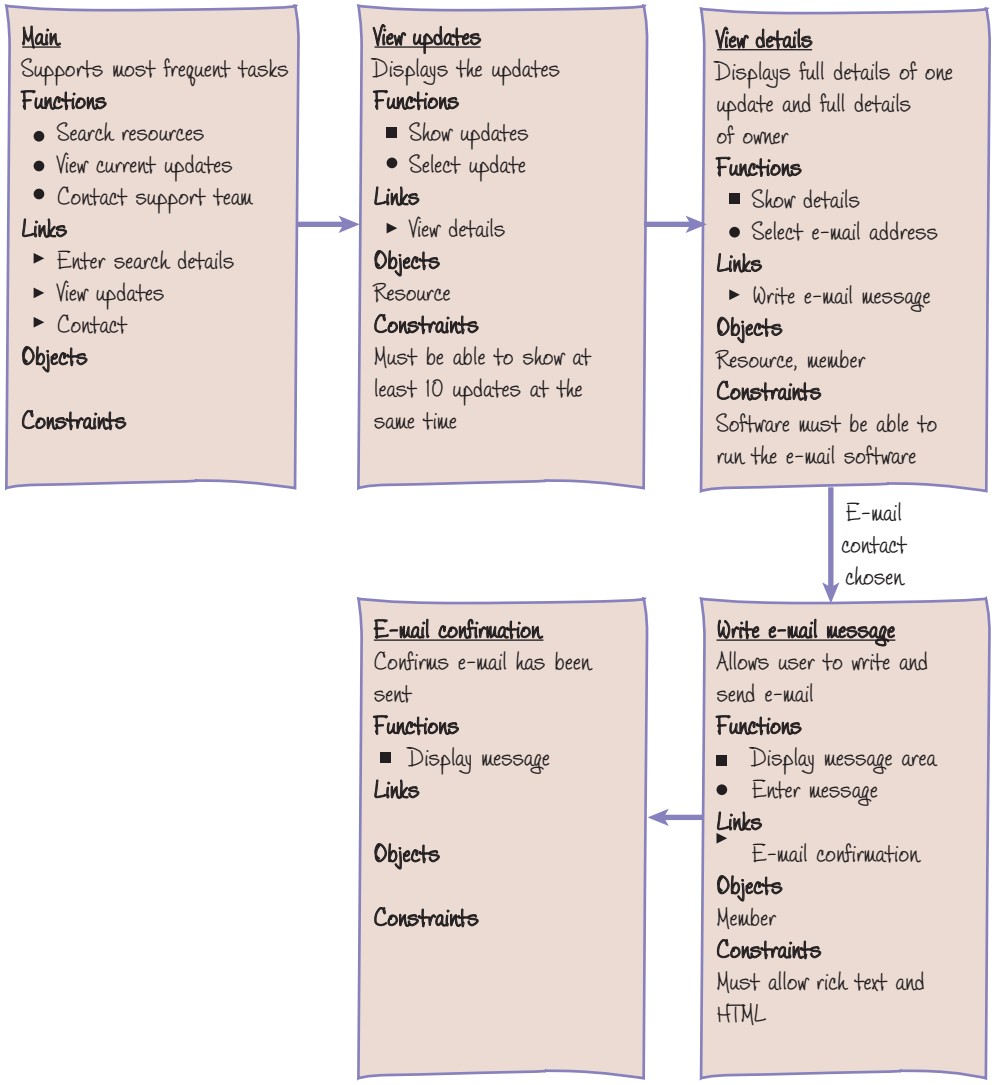

Figure 8.18 A section of the content diagram corresponding to the "View updates and request book" concrete use case.

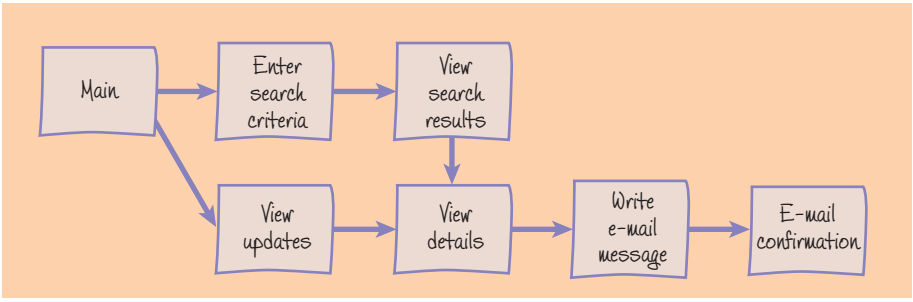

Figure 8.19 The content diagram for the digital library.

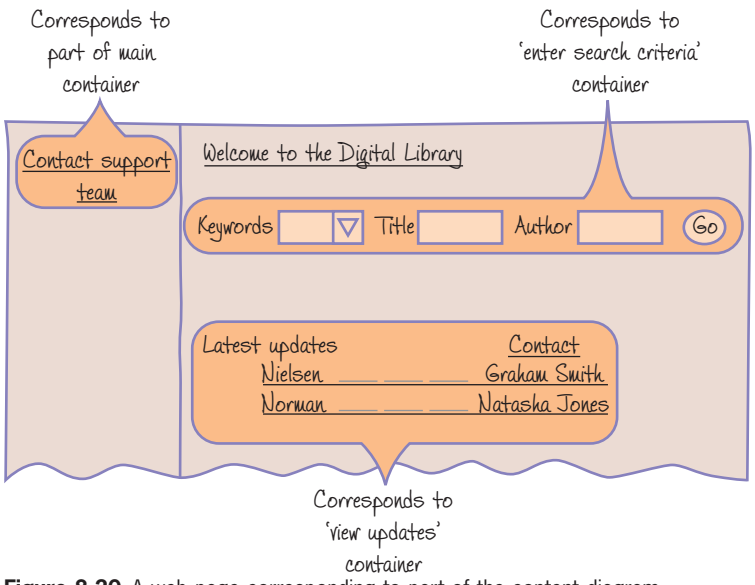

Figure 8.20 A web page corresponding to part of the content diagram.

important part of this process is task allocation, by which the various activities are shared between the user and the computer system. We have shown how use scenarios and essential use cases can help us think about work reengineering.

We then discussed the process of conceptual design. Conceptual design results in a content diagram, which represents the underlying structure and organization of the UI. We showed you how to develop the content diagram; first the essential use cases are translated into concrete use cases; then the task objects, attributes, and actions are identified from the concrete use cases; and finally the containers and links for the content diagram are identified.

Work reengineering and conceptual design are demanding but important tasks. For these reasons it is important to prototype and check your ideas with the users.

In Chapter 10, we begin to discuss the physical design of the user interface.