arrangeMeeting

USER INTENTION	SYSTEM RESPONSIBILITY
arrange a meeting	
	request meeting attendees and constraints
identify meeting attendees and constraints	
	suggest potential dates
choose preferred date	
	book meeting

Figure 7.10 An essential use case for arranging a meeting in the shared calendar application.

above. Scenarios are concrete stories that concentrate on realistic and specific activities. They therefore can obscure broader issues concerned with the wider organizational view. On the other hand, traditional use cases contain certain assumptions, including the fact that there is a piece of technology to interact with, and also assumptions about the user interface and the kind of interaction to be designed.

Essential use cases represent abstractions from scenarios, i.e., they represent a more general case than a scenario embodies, and try to avoid the assumptions of a traditional use case. An essential use case is a structured narrative consisting of three parts: a name that expresses the overall user intention, a stepped description of user actions, and a stepped description of system responsibility. This division between user and system responsibilities can be very helpful during conceptual design when considering task allocation and system scope, i.e., what the user is responsible for and what the system is to do.

An example essential use case based on the library example given above is shown in Figure 7.10. Note that the steps are more generalized than those in the use case in Section 7.6.2, while they are more structured than the scenario in Section 7.6.1. For example, the first user intention does not say anything about typing in a list of names, it simply states that the user identifies meeting attendees. This could be done by identifying roles, rather than people's names, from an organizational or project chart, or by choosing names from a list of people whose calendars the system keeps, or by typing in the names. The point is that at the time of creating this essential use case, there is no commitment to a particular interaction design.

Instead of actors, essential use cases are associated with user roles. One of the differences is that an actor could be another system, whereas a user role is just that: not a particular person, and not another system, but a role that a number of different people may play when using the system. Just as with actors, though, producing an essential use case begins with identifying user roles.

ACTIVITY 7.5 Construct an essential use case "locateBook" for the user role "Library member" of the library catalog service discussed in Activity 7.4.

Comment

locateBook

USER INTENTION	SYSTEM RESPONSIBILITY
identify self	
	verify identity
	request appropriate details
offer known details	
	offer search results
note search results	
quit system	
	close

Note that here we don't talk about passwords, but merely state that the users need to identify themselves. This could be done using fingerprinting, or retinal scanning, or any other suitable technology. The essential use case does not commit us to technology at this point. Neither does it specify search options or details of how to initiate the search.

7.7 Task analysis

Task analysis is used mainly to investigate an existing situation, not to envision new systems or devices. It is used to analyze the underlying rationale and purpose of what people are doing: what are they trying to achieve, why are they trying to achieve it, and how are they going about it? The information gleaned from task analysis establishes a foundation of existing practices on which to build new requirements or to design new tasks.

Task analysis is an umbrella term that covers techniques for investigating cognitive processes and physical actions, at a high level of abstraction and in minute detail. In practice, task analysis techniques have had a mixed reception. The most widely used version is Hierarchical Task Analysis (HTA) and this is the technique we introduce in this chapter. Another well-known task analysis technique called GOMS (goals, operations, methods, and selection rules) that models procedural knowledge (Card et al., 1983) is described in Chapter 14.

7.7.1 Hierarchical task analysis

Hierarchical Task Analysis (HTA) was originally designed to identify training needs (Annett and Duncan, 1967). It involves breaking a task down into subtasks and then into sub-subtasks and so on. These are then grouped together as plans that specify how the tasks might be performed in an actual situation. HTA focuses on the physical and observable actions that are performed, and includes looking at actions that are not related to software or an interaction device at all. The starting point is a user goal. This is then examined and the main tasks associated with achieving that goal are identified. Where appropriate, these tasks are subdivided into subtasks.

Consider the library catalog service, and the task of borrowing a book. This task can be decomposed into other tasks such as accessing the library catalog, searching by name, title, subject, or whatever, making a note of the location of the book, going to the correct shelf, taking it down off the shelf (provided it is there) and finally tak-

0. In order to borrow a book from the library
   1. go to the library
   2. find the required book
      2.1 access library catalog
      2.2 access the search screen
      2.3 enter search criteria
      2.4 identify required book
      2.5 note location
   3. go to correct shelf and retrieve book
   4. take book to checkout counter

plan 0: do 1-3-4. If book isn't on the shelf expected, do 2-3-4.
plan 2: do 2.1-2.4-2.5. If book not identified do 2.2-2.3-2.4-2.5.

Figure 7.11 An HTA for borrowing a book from the library.

it to the check-out counter. This set of tasks and subtasks might be performed in a different order depending on how much is known about the book, and how familiar the user might be with the library and the book's likely location. Figure 7.11 shows these subtasks and some plans for different paths through those subtasks. Indentation shows the hierarchical relationship between tasks and subtasks.

Note how the numbering works for the task analysis: the number of the plan corresponds to the number of the step to which the plan relates. For example, plan 2 shows how the subtasks in step 2 can be ordered; there is no plan 1 because step 1 has no subtasks associated with it.

An alternative expression of an HTA is a graphical box-and-line notation. Figure 7.12 shows the graphical version of the HTA in Figure 7.11. Here the subtasks are represented by named boxes with identifying numbers. The hierarchical relationship between tasks is shown using a vertical line. If a task is not decomposed any further then a thick horizontal line is drawn underneath the corresponding box.

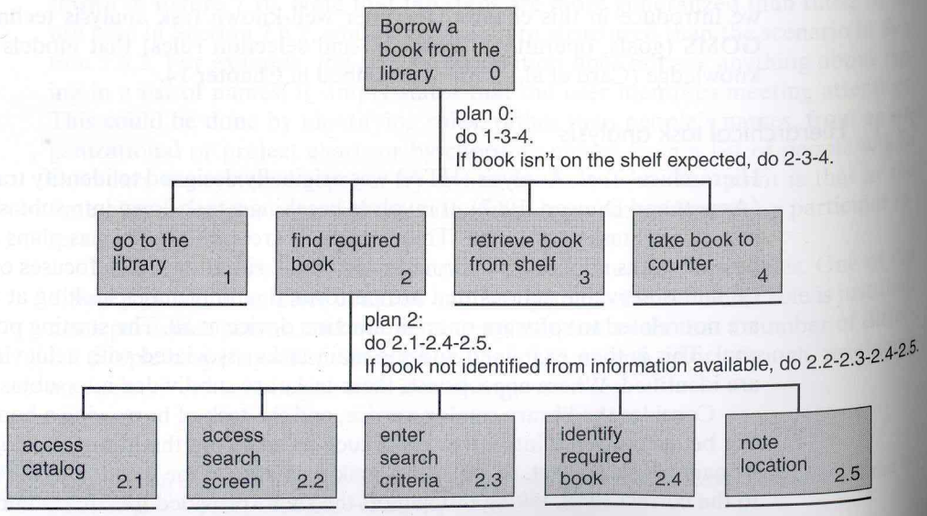

Figure 7.12 A graphical representation of the task analysis for borrowing a book.

Plans are also shown in this graphical form. They are written alongside the vertical line emitting from the task being decomposed. For example, in Figure 7.12 plan 2 is specified next to the vertical line from box 2 "find required book."

ACTIVITY 7.6 Look back at the scenario for arranging a meeting in the shared calendar application. Perform hierarchical task analysis for the goal of arranging a meeting. Include all plans in your answer. Express the task analysis textually and graphically.

Comment

The main tasks involved in this are to find out who needs to be at the meeting, find out the constraints on the meeting such as length of meeting, range of dates, and location, find a suitable date, enter details into the calendar, and inform attendees. Finding a suitable date can be decomposed into other tasks such as looking in the departmental calendar, looking in individuals' calendars, and checking potential dates against constraints. The textual version of the HTA is shown below. Figure 7.13 shows the corresponding graphical representation.

0. In order to arrange a meeting
   1. compile a list of meeting attendees
   2. compile a list of meeting constraints
   3. find a suitable date
      3.1 identify potential dates from departmental calendar
      3.2 identify potential dates from each individual's calendar
      3.3 compare potential dates
      3.4 choose one preferred date
   4. enter meeting into calendars
   5. inform meeting participants of calendar entry

plan 0: do 1-2-3. If potential dates are identified, do 4-5. If no potential dates can be identified, repeat 2-3.

plan 3: do 3.1-3.2-3.3-3.4 or do 3.2-3.1-3.3-3.4

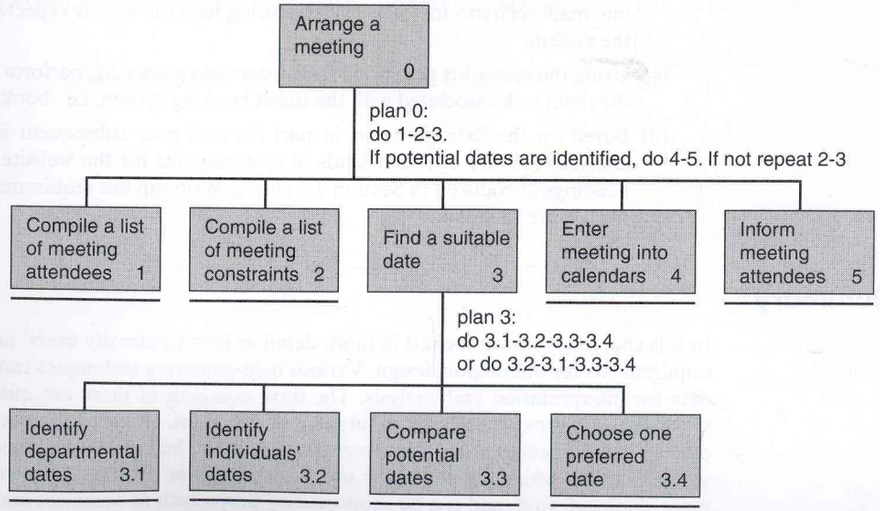

Figure 7.13 A graphical representation of the meeting HTA.