CHAPTER 6

Requirements Capture

OBJECTIVES

In this chapter you will learn

- the distinction between the current and required systems
- when and how to apply the five major fact finding techniques
- the need to document requirements
- how to draw *use case diagrams* to document requirements
- how to write *use case descriptions*.

6.1 Introduction

Part of the job of the systems analyst is to find out from users what they require in a new information system. Indeed, identifying what a new system should be able to do is one of the first steps in its development, whether you are developing some simple programs for your own use or embarking on the development of a large-scale system for a commercial client. The user requirements can be classified in different ways (Section 6.2), and analysts use a range of techniques to identify and document the requirements (Sections 6.3 and 6.5). Each of the main fact finding techniques has advantages and disadvantages and is appropriate for different situations. Stakeholders are the people who have an interest in the new system and whose needs must be considered (Section 6.4). UML provides a diagramming technique that can be used to document the stakeholders' requirements. This is the use case diagram, a relatively simple diagram that is supported by written information in the form of use case descriptions (Section 6.6).

6.2 User Requirements

The aim of developing a new information system must be to produce something that meets the needs of the people who will be using it. In order to do this, we must have a clear understanding both of the overall objectives of the business and of what it is that the individual users of the system are trying to achieve in their jobs. Unless you are in the rare position of developing a system for a new organization, you will need to understand how the business is operating at present and how people are working now. Many aspects of the current system will need to be carried forward into the new system, so it is important that information about what people are doing is gathered and documented. These are the requirements that are derived from the 'current system'. The motivation for the development of a new information system is usually problems with and inadequacies of the current system, so it is also essential to capture what it is that the users require of the new system that they cannot do with their existing system. These are the 'new requirements'.

6.2.1 Current system

The existing system may be a manual one, based on paper documents, forms and files; it may already be computerized; or it may be a combination of both manual and computerized elements. Whichever it is, it is reasonably certain that large parts of the existing system meet the needs of the people who use it, that it has to some extent evolved over time to meet business needs and that users are familiar and comfortable with it. It is almost equally certain that there are sections of the system that no longer meet the needs of the business, and that there are aspects of the business that are not dealt with in the existing system.

It is important that the analyst, gathering information as one of the first steps in developing a new system, gains a clear understanding of how the existing system works: parts of the existing system will be carried forward into the new one. It is also important, because the existing system will have shortcomings and defects, which must be avoided or overcome in the new system. It is not always easy or possible to replace existing systems. So-called *legacy systems* may have been developed some time ago and may contain millions of lines of program code, which have been added to and amended over a period of time. One approach to dealing with such systems is to create new front-ends, typically using modern graphical user interfaces and object-oriented languages, and *wrap* the legacy systems up in new software. If this is the case, then it is also necessary to understand the interfaces to the legacy systems that the new *wrappers* will have to communicate with.

It is not always possible to leave legacy systems as they are and simply wrap them in new code. It was not possible to ignore the problems that faced companies at the turn of the century when it was realized that many systems were in danger of catastrophic collapse as a result of the decision to use two decimal digits to store the year. However, the process of changing the program code in such systems is a matter of understanding the internal working of existing systems rather than gathering information about the way the organization works and the way that people do their jobs.

Not everyone agrees that a detailed understanding of the current system is necessary. Ed Yourdon (1989) argues that it is a waste of time to model the current system in great detail. Yourdon points out that so much time can be spent investigating and modelling the current system that the analysts lose sight of their objective, and impatient users

cancel the project. He makes the case for concentrating on the behaviour that is required of the new system. The opposite position is taken by SSADM (Structured Systems Analysis and Design Method), which expends considerable time on the investigation and modelling of the current system. This is done in order to refine it to its logical essence and to be able to merge it with the new requirements to produce a model of the required system that includes the essentials of the existing system.

We believe that a case can be made for investigating the existing system.

- Some of the functionality of the existing system will be required in the new system.
- Some of the data in the existing system is of value and must be migrated into the new system.
- Technical documentation of existing computer systems may provide details of processing algorithms that will be needed in the new system.
- The existing system may have defects that we should avoid in the new system.
- Studying the existing system will help us to understand the organization in general.
- Parts of the existing system may be retained. Information systems projects are now rarely 'green field' projects in which manual systems are replaced by new computerized systems; more often there will be existing systems with which interfaces must be established.
- We are seeking to understand how people do their jobs at present in order to characterize people who will be users of the new system.
- We may need to gather baseline information against which we can set and measure performance targets for the new system.

For all these reasons, an understanding of the current system should be part of the analysis process. However, the analyst should not lose sight of the objective of developing a new system. In the sections on functional, non-functional and usability requirements below, we shall explain what kind of information we are gathering.

6.2.2 New requirements

Most organizations now operate in an environment that is rapidly changing. The relative strength of national economies around the world can change dramatically and at short notice; the fortunes of large companies, which may be an organization's suppliers, customers or competitors, can be transformed overnight; new technologies are introduced which change production processes, distribution networks and the relationship with the consumer; governments and (particularly in Europe) supra-governmental organizations introduce legislation that has an impact on the way that business is conducted. Some authors make the case for developing business strategies to cope with this turmoil. Tom Peters in 'Thriving on Chaos' (1988) argues that we must learn to love change and develop flexible and responsive organizations to cope with the dynamic business environment. A clear result of responding to a dynamic environment is that organizations change their products and services and change the way they do business. The effect of this is to change their need for information. Even in less responsive organizations, information systems become outdated and need enhancing and extending. Mergers and demergers create the need for systems to be replaced. The process of replacement offers an opportunity to extend the capabilities of systems to take advantage of new technological developments, or to enhance their usefulness to management and workforce. Many organizations are driven by internal

factors to grow and change the ways in which they operate, and this too provides a motivation for the development of new information systems.

Whether you are investigating the working of the existing system or the requirements for the new system, the information you gather will fall into one of three categories: 'functional requirements', 'non-functional requirements' and 'usability requirements'. Functional and non-functional requirements are conventional categories in systems analysis and design, while usability is often ignored in systems development projects. In many university courses, issues surrounding the usability of systems are taught under the separate heading of Human Factors or Human–Computer Interaction, or are only considered in the design stage of the development process. However, the lesson of human factors research is that usability considerations should be integral to the systems development lifecycle, and so they are included here.

Functional requirements

Functional requirements describe what a system does or is expected to do, often referred to as its *functionality*. In the object-oriented approach, which we are taking here, we shall initially employ use cases to document the functionality of the system. As we progress into the analysis stage, the detail of the functionality will be recorded in the data that we hold about objects, their attributes and operations. In structured methods, such as SSADM, the function may be the unit around which we structure the system, and the function is described in progressively greater detail as we move through analysis into design and implementation.

At this stage, we are setting out to establish what the system must do, and functional requirements include the following.

- Descriptions of the processing that the system will be required to carry out.
- Details of the inputs into the system from paper forms and documents, from interactions between people, such as telephone calls, and from other systems.
- Details of the outputs that are expected from the system in the form of printed documents and reports, screen displays and transfers to other systems.
- Details of data that must be held in the system.

Non-functional requirements

Non-functional requirements are those that describe aspects of the system that are concerned with how well it provides the functional requirements. These include the following.

- Performance criteria such as desired response times for updating data in the system or retrieving data from the system.
- Anticipated volumes of data, either in terms of throughput or of what must be stored.
- Security considerations.

Usability requirements

Usability requirements are those that will enable us to ensure that there is a good match between the system that is developed and both the users of that system and the tasks that they will undertake when using it. The International Standards Organization (ISO) has defined the usability of a product as 'the degree to which

specific users can achieve specific goals within a particular environment; effectively, efficiently, comfortably and in an acceptable manner'. Usability can be specified in terms of measurable objectives, and these are covered in more detail in Chapter 16 on Human–Computer Interaction. In order to build usability into the system from the outset, we need to gather the following types of information.

- Characteristics of the users who will use the system.
- The tasks that the users undertake, including the goals that they are trying to achieve.
- Situational factors that describe the situations that could arise during system use.
- Acceptance criteria by which the user will judge the delivered system.

Paul Booth (1989) describes the issues surrounding system usability in more detail.

6.3 Fact Finding Techniques

There are five main fact finding techniques that are used by analysts to investigate requirements. Here we describe each of them in the order that they might be applied in a system development project, and for each one we explain the kind of information that you would expect to gain from its use, its advantages and disadvantages, and the situations in which it is appropriate to use it.

6.3.1 Background reading

If an analyst is employed within the organization that is the subject of the fact-gathering exercise, then it is likely that he or she will already have a good understanding of the organization and its business objectives. If, however, he or she is going in as an outside consultant, then one of the first tasks is to try to gain an understanding of the organization. Background reading or research is part of that process. The kind of documents that are suitable sources of information include the following:

- company reports,
- organization charts,
- policy manuals,
- job descriptions,
- reports and
- documentation of existing systems.

Although reading company reports may provide the analyst with information about the organization's mission, and so possibly some indication of future requirements, this technique mainly provides information about the current system.

Advantages and disadvantages

+ Background reading helps the analyst to get an understanding of the organization before meeting the people who work there.
+ It also allows the analyst to prepare for other types of fact finding, for example, by being aware of the business objectives of the organization.

+ Documentation on the existing system may provide formally defined information requirements for the current system.
– Written documents often do not match up to reality; they may be out of date or they may reflect the official policy on matters that are dealt with differently in practice.

Appropriate situations

Background reading is appropriate for projects where the analyst is not familiar with the organization being investigated. It is useful in the initial stages of investigation.

6.3.2 Interviewing

Interviewing is probably the most widely used fact finding technique; it is also the one that requires the most skill and sensitivity. Because of this, we have included a set of guidelines on interviewing that includes some suggestions about etiquette in Box 6.1.

Box 6.1 Guidelines on Interviewing

Conducting an interview requires good planning, good interpersonal skills and an alert and responsive frame of mind. These guidelines cover the points you should bear in mind when planning and conducting an interview.

Before the interview

You should always make appointments for interviews in advance. You should give the interviewee information about the likely duration of the interview and the subject of the interview.

Being interviewed takes people away from their normal work. Make sure that they feel that it is time well spent.

It is conventional to obtain permission from an interviewee's line manager before interviewing them. Often the analyst interviews the manager first and uses the opportunity to get this permission.

In large projects, an interview schedule should be drawn up showing who is to be interviewed, how often and for how long. Initially this will be in terms of the job roles of interviewees rather than named individuals. It may be the manager who decides which individual you interview in a particular role.

Have a clear set of objectives for the interview. Plan your questions and write them down. Some people write the questions with space between them for the replies.

Make sure your questions are relevant to the interviewee and his or her job.

At the start of the interview

Introduce yourself and the purpose of the interview.

Arrive on time for interviews and stick to the planned timetable—do not over-run.

Ask the interviewee if he or she minds you taking notes or tape-recording the interview. Even if you tape-record an interview, you are advised to take notes. Machines can fail! Your notes also allow you to refer back to what has been said during the course of the interview and follow up points of interest.

Remember that people can be suspicious of outside consultants who come in with clipboards and stopwatches. The cost-benefit analyses of many information systems justify the investment in terms of savings in jobs!

During the interview

Take responsibility for the agenda. You should control the direction of the interview. This should be done in a sensitive way. If the interviewee is

getting away from the subject, bring them back to the point. If what they are telling you is important, then say that you will come back to it later and make a note to remind yourself to do so.

Use different kinds of question to get different types of information. Questions can be open-ended—'Can you explain how you complete a timesheet?'—or closed—'How many staff use this system?'. Do not, however, ask very open-ended questions such as 'Could you tell me what you do?'

Listen to what the interviewee says and encourage him or her to expand on key points.

Keep the focus positive if possible. Make sure you have understood answers by summarizing them back to the interviewee. Avoid allowing the interview to degenerate into a session in which the interviewee complains about everyone and everything.

You may be aware of possible problems in the existing system, but you should avoid prejudging issues by asking questions that focus too much on problems. Gather facts.

Be sensitive about how you use information from other interviews that you or your colleagues have already conducted, particularly if comments were negative or critical.

Use the opportunity to collect examples of documents that people use in their work, ask if they mind you having samples of blank forms and photocopies of completed paperwork.

After the interview

Thank the interviewee for their time. Make an appointment for a further interview if it is necessary. Offer to provide them with a copy of your notes of the interview for them to check that you have accurately recorded what they told you.

Transcribe your tape or write up your notes as soon as possible after the interview while the content is still fresh in your mind.

If you said that you would provide a copy of your notes for checking then send it to the interviewee as soon as possible. Update your notes to reflect their comments.

A systems analysis interview is a structured meeting between the analyst and an interviewee who is usually a member of staff of the organization being investigated. The interview may be one of a series of interviews that range across different areas of the interviewee's work or that probe in progressively greater depth about the tasks undertaken by the interviewee. The degree of structure may vary: some interviews are planned with a fixed set of questions that the interviewer works through, while others are designed to cover certain topics but will be open-ended enough to allow the interviewer to pursue interesting facts as they emerge. The ability to respond flexibly to the interviewee's responses is one of the reasons why interviews are so widely used.

Interviews can be used to gather information from management about their objectives for the organization and for the new information system, from staff about their existing jobs and their information needs, and from customers and members of the public as possible users of systems. While conducting an interview, the analyst can also use the opportunity to gather documents that the interviewee uses in his or her work.

It is usually assumed that questionnaires are used as a substitute for interviews when potential interviewees are geographically dispersed in branches and offices around the world. The widespread use of desktop video conferencing may change this and make it possible to interview staff wherever they are. Even then, questionnaires can reach more people.

Interviewing different potential users of a system separately can mean that the analyst is given different information by different people. Resolving these differences later can be difficult and time-consuming. One alternative is to use group interviews

in order to get the users to reach a consensus on issues. Dynamic Systems Development Method (DSDM) is a method of carrying out systems development in which group discussions are used (Stapleton, 1997). These discussions are run as workshops for knowledgeable users with a facilitator who aims to get the users to pool their knowledge and to reach a consensus on the priorities of the development project.

Advantages and disadvantages

+ Personal contact allows the analyst to be responsive and adapt to what the user says. Because of this, interviews produce high quality information.
+ The analyst can probe in greater depth about the person's work than can be achieved with other methods.
+ If the interviewee has nothing to say, the interview can be terminated.
– Interviews are time-consuming and can be the most costly form of fact gathering.
– Interview results require the analyst to work on them after the interview: the transcription of tape recordings or writing up of notes.
– Interviews can be subject to bias if the interviewer has a closed mind about the problem.
– If different interviewees provide conflicting information, it can be difficult to resolve later.

Appropriate situations

Interviews are appropriate in most projects. They can provide information in depth about the existing system and about people's requirements from a new system.

6.3.3 Observation

Watching people carrying out their work in a natural setting can provide the analyst with a better understanding of the job than interviews, in which the interviewee will often concentrate on the normal aspects of the job and forget the exceptional situations and interruptions which occur and which the system will need to cope with. Observation also allows the analyst to see what information people use to carry out their job. This can tell you about the documents they refer to, whether they have to get up from their desks to get information, how well the existing system handles their needs. One of the authors has observed staff using a tele-sales system where there was no link between the enquiry screens for checking the availability of stock and the data entry screens for entering an order. These tele-sales staff kept a pad of scrap paper on the desk and wrote down the product codes for all the items they had looked up on the enquiry screens so that they could enter them into the order-processing screens. This kind of information does not always emerge from interviews.

People are not good at estimating quantitative data, such as how long they take to deal with certain tasks, and observation with a stopwatch can give the analyst plentiful quantitative data, not just about typical times to perform a task but also about the statistical distribution of those times.

In some cases where information or items are moving through a system and being dealt with by many people along the way, observation can allow the analyst to follow the entire process through from start to finish. This type of observation might be used in an

organization where orders are taken over the telephone, passed to a warehouse for picking, packed and despatched to the customer. The analyst may want to follow a series of transactions through the system to obtain an overview of the processes involved.

Observation can be an open-ended process in which the analyst simply sets out to observe what happens and to note it down, or it can be a closed process in which the analyst wishes to observe specific aspects of the job and draws up an observation schedule or form on which to record data. This can include the time it takes to carry out a task, the types of task the person is performing or factors such as the number of errors they make in using the existing system as a baseline for usability design.

Advantages and disadvantages

+ Observation of people at work provides first hand experience of the way that the current system operates.
+ Data are collected in real time and can have a high level of validity if care is taken in how the technique is used.
+ Observation can be used to verify information from other sources or to look for exceptions to the standard procedure.
+ Baseline data about the performance of the existing system and of users can be collected.
– Most people do not like being observed and are likely to behave differently from the way in which they would normally behave. This can distort findings and affect the validity.
– Observation requires a trained and skilled observer for it to be most effective.
– There may be logistical problems for the analyst, for example, if the staff to be observed work shifts or travel long distances in order to do their job.
– There may also be ethical problems if the person being observed deals with sensitive private or personal data or directly with members of the public, for example in a doctor's surgery.

Appropriate situations

Observation is essential for gathering quantitative data about people's jobs. It can verify or disprove assertions made by interviewees, and is often useful in situations where different interviewees have provided conflicting information about the way the system works. Observation may be the best way to follow items through some kind of process from start to finish.

6.3.4 Document sampling

Document sampling can be used in two different ways. First, the analyst will collect copies of blank and completed documents during the course of interviews and observation sessions. These will be used to determine the information that is used by people in their work, and the inputs to and outputs from processes which they carry out, either manually or using an existing computer system. Ideally, where there is an existing system, screen shots should also be collected in order to understand the inputs and outputs of the existing system. Figure 6.1 shows a sample document collected from Agate, our case study company.

Agate

Campaign Summary

Date 23rd February 2002

Client Yellow Partridge
Park Road Workshops
Jewellery Quarter
Birmingham B2 3DT
U.K.

Campaign Spring Collection 2002

Billing Currency GBP £

Item	Curr	Amount	Rate	Billing amount
Advert preparation: photography, artwork, layout etc.	GBP £	15,000.00	1	15,000.00
Placement French Vogue	EUR €	6 500,00	1.61	4,037.27
Placement Portuguese Vogue	EUR €	5 500,00	1.61	3,416.15
Placement US Vogue	USD $	17,000.00	1.44	11,805.56
Total				34,258.98

This is not a VAT Invoice. A detailed VAT Invoice will be provided separately.

Figure 6.1 Sample document from the AGATE case study.

Second, the analyst may carry out a statistical analysis of documents in order to find out about patterns of data. For example, many documents such as order forms contain a header section and a number of lines of detail. (The sample document in Figure 6.1 shows this kind of structure.) The analyst may want to know the distribution of the number of lines in an order. This will help later in estimating volumes of data to be held in the system and in deciding how many lines should be displayed on screen at one time. While this kind of statistical sampling can give a picture of data volumes, the analyst should be alert to seasonal patterns of activity, which may mean that there are peaks and troughs in the amount of data being processed.

Advantages and disadvantages

+ Can be used to gather quantitative data, such as the average number of lines on an invoice.
+ Can be used to find out about error rates in paper documents.
– If the system is going to change dramatically, existing documents may not reflect how it will be in future.

Appropriate situations

The first type of document sampling is almost always appropriate. Paper-based documents give a good idea of what is happening in the current system. They also provide supporting evidence for the information gathered from interviews or observation.

The statistical approach is appropriate in situations where large volumes of data are being processed, and particularly where error rates are high, and a reduction in errors is one of the criteria for usability.

6.3.5 Questionnaires

Questionnaires are a research instrument that can be applied to fact finding in system development projects. They consist of a series of written questions. The questionnaire designer usually limits the range of replies that respondents can make by giving them a choice of options. (Figure 6.2 shows some of the types of question.) YES/NO questions only give the respondent two options. (Sometimes a DON'T KNOW option is needed as well.) If there are more options, the multiple choice type of question is often used when the answer is factual, whereas scaled questions are used if the answer involves an element of subjectivity. Some questions do not have a fixed number of responses, and must be left open-ended for the respondent to enter what they like. Where the respondent has a limited number of choices, these are usually coded with a number, which speeds up data entry if the responses are to be analysed by computer software. If you plan to use questionnaires for requirements gathering, they need very careful design. Box 6.2 lists some of the issues that need to be addressed if you are thinking of using questionnaires.

Advantages and disadvantages

+ An economical way of gathering data from a large number of people.
+ If the questionnaire is well designed, then the results can be analysed easily, possibly by computer.
– Good questionnaires are difficult to construct.
– There is no automatic mechanism for follow up or probing more deeply, although it is possible to follow up with an interview by telephone or in person if necessary.
– Postal questionnaires suffer from low response rates.

YES/NO Questions

Do you print reports from the existing system? YES NO 10
(Please circle the appropriate answer.)

Multiple Choice Questions

How many new clients do you obtain in a year? 11
(Please tick one box only.)

a) 1–10 ☐
b) 11–20 ☐
c) 21–30 ☐
d) 31 + ☐

Scaled Questions

How satisfied are you with the response time of the stock update?
(Please circle one option.)

1. Very satisfied 2. Satisfied 3. Dissatisfied 4. Very dissatisfied 12

Open-ended Questions

What additional reports would you require from the system?

_____

_____

_____

Figure 6.2 Types of question used in questionnaires.

Box 6.2 Guidelines on Questionnaires

Using questionnaires requires good planning. If you send out 100 questionnaires and they do not work, it is difficult to get respondents to fill in a second version. These guidelines cover the points you should bear in mind when using questionnaires.

Coding

How will you code the results? If you plan to use an optical mark reader, then the response to every question must be capable of being coded as a mark in a box. If you expect the results to be keyed into a database for analysis, then you need to decide on the codes for each possible response. If the questions are open-ended, how will you collate and analyse different kinds of responses?

Analysis

Whatever analysis you plan should be decided in advance. If you expect to carry out a statistical analysis of the responses, you should consult a statistician before you finalize the questions. Statistical techniques are difficult to apply to responses to poorly designed questions.

You can use a special statistical software package, a database or even a spreadsheet to analyse the data.

Piloting

You should try out your questionnaire on a small pilot group or sample of your respondents. This enables you to find out if there are questions they do not understand, they misinterpret or they cannot answer.

If you plan to analyse the data using statistical software, a database or a spreadsheet, you can create a set of trial data to test your analysis technique.

Sample size and structure

If you plan to use serious statistical techniques, then those techniques may place lower limits on your sample size. If you want to be sure of getting a representative sample, by age, gender, department, geographical location, job grade or experi-

ence of existing systems, then that will help to determine how many people to include. Otherwise it may be down to you to choose a sensible percentage of all the possible respondents.

Delivery
How will you get the questionnaires to your respondents, and how will they get their replies back to you?

You can post them, or use internal mail in a large organization, fax them, e-mail them or create a web-based form on the company intranet and notify your target group by e-mail. If you use the intranet, you may want to give each respondent a special code, so that only they can complete their own questionnaire.

Your respondents can then post, fax or e-mail their responses back to you.

Respondent information
What information about the respondents do you want to gather at the same time as you collect their views and requirements? If you want to analyse responses by age, job type or location, then you need to include questions that ask for that information.

You can make questionnaires anonymous, or you can ask respondents for their name. If the questionnaire is not anonymous, you need to think about confidentiality. People will be more honest in their replies if they can respond anonymously or in confidence.

If you ask for respondents' names and you store that information, then in the UK you should consider the provisions of the Data Protection Act (1998). (See also Chapter 12.) There are similar requirements in other countries.

Covering letter
In a covering letter you should explain the purpose and state that the questionnaire has management support. Give an estimate of the time required to fill in the questionnaire and a deadline for its return. Thank the respondents for taking part.

Structure
Structure the questionnaire carefully. Give it a title, and start with explanatory material and notes on how to complete it. Follow this with questions about the respondent (if required). Group questions together by subject. Avoid lots of instructions like 'If you answered YES to Q. 7a, now go to Q. 13.' Keep it reasonably short.

Return rate
Not everyone will necessarily respond. You need to plan for this and either use a larger sample than you need or follow up with reminders. If you use a form on the Intranet, you should be able to identify who has not responded and e-mail them reminders. Equally, you can e-mail a thank you to those who do respond.

Feedback
This needs to be handled carefully—telling everyone that 90% of the company cannot use the existing system may not go down well—but people do like to know what use was made of the response they made. They may have spent half an hour filling in your questionnaire, and they will expect to be informed of the outcome. A summary of the report can be sent out to branches, distributed to departments, sent to named respondents or placed on the company intranet.

Appropriate situations

Questionnaires are most useful when the views or knowledge of a large number of people need to be obtained or when the people are geographically dispersed, for example, in a company with many branches or offices around the country or around the world. Questionnaires are also appropriate for information systems that will be used by the general public, and where the analyst needs to get a picture of the types of user and usage that the system will need to handle.

6.3.6 Remembering the techniques

For those who like mnemonics, these techniques are sometimes referred to as SQIRO—Sampling, Questionnaires, Interviewing, Reading (or Research) and Observation. This order has been chosen to make it possible to pronounce the mnemonic. However, this is not the order in which they are most likely to be used. This will depend on the situation and the organization in which the techniques are being used.

6.3.7 Other techniques

Some kinds of systems require special fact finding techniques. *Expert systems* are computer systems that are designed to embody the expertise of a human expert in solving problems. Examples include systems for medical diagnosis, stock market trading and geological analysis for mineral prospecting. The process of capturing the knowledge of the expert is called *knowledge acquisition* and, as it differs from establishing the requirements for a conventional information system, a number of specific techniques are applied. Some of these are used in conjunction with computer-based tools.

6.4 User Involvement

The success of a systems development project depends not just on the skills of the team of analysts, designers and programmers who work on it, or on the project management skills of the project manager, but on the effective involvement of users in the project at various stages of the life cycle. The term *stakeholders* was introduced in Chapter 2 to describe all those people who have an interest in the successful development of the system. Stakeholders include all people who stand to gain (or lose) from the implementation of the new system: users, managers and budget-holders. Analysts deal with people at all levels of the organization. In large projects it is likely that a steering committee with delegated powers will be set up to manage the project from the users' side. This will include the following categories of people:

- senior management—with overall responsibility for running the organization,
- financial managers with budgetary control over the project,
- managers of the user department(s) and
- representatives of users.

Users will be involved in different roles during the course of the project as:

- subjects of interviews to establish requirements,
- representatives on project committees,
- those involved in evaluating prototypes,
- those involved in testing,
- subjects of training courses and
- end-users of the new system.

Case Study Example

The section that follows applies to what has been covered in this chapter so far to the case study.

One of the first tasks in fact finding is to draw up a plan that outlines what information is being sought, which techniques will be used, who is involved and how long the fact finding will take. A draft plan for fact finding at Agate is shown below. The jobs of the subjects are those shown in Figure A1.1 in the Agate case study.

Objective	Technique	Subject(s)	Time commitment
To get background on the company and the advertising industry	Background reading	Company reports, trade journals	0.5 day
To establish business objectives. Agree likely scope of new system. Check out involvement of non-UK offices	Interview	Two directors	2 x 1 hour each
To gain understanding of roles of each department. Check out line management and team structure in the Creative Department. To agree likely interviewees among staff	Interview	Department heads (only 1 account manager)	2 x 1 hour each
To find out how the core business operates	Interview	1 account manager 1 graphic designer 1 copy writer 1 editor	1.5 hours each
To follow up development of business understanding	Observation	2 creative staff	0.5 day each
To determine role of support/admin staff and relationship to core business	Interview	2 admin staff (based on experience with the company)	1.5 hours each
To establish what records and resources are kept	Interview/ document sampling	Filing clerk Resource librarian	2 x 1 hour each
To determine what use is made of current computer system. To determine functionality of current system	Interview	Computer manager	2 x 1 hour
To establish additional requirements for new system	Interview	2 account managers 3 staff from Creative Department	3 x 1 hour each
To establish accounting requirements for new system	Interview	Accountant Credit controller 1 purchasing assistant 1 accounts clerk	1.5 hours each

6.5 Documenting Requirements

Information systems professionals need to record facts about the organization they are studying and its requirements. As soon as the analysts start gathering facts, they will need some means of documenting them. In the past the emphasis was on paper forms, but now it is rare for a large-scale project to depend on paper-based documentation. As we have explained in Chapter 5, systems analysts and designers model the new system in a mixture of diagrams and text. The important thing to bear in mind is that within a project some set of standards should be adhered to. These may be the agreed standards of the organization carrying out the analysis and design project or they may be a requirement of the organization that is having the work done. For example, government and military projects usually require that developers conform to a specific set of standards. We are using UML to produce models of the system from different perspectives. Computer Aided Software Engineering (CASE) tools are normally used to draw the diagrammatic models and to maintain in a repository the associated data about the various things that are shown in the diagrams.

However, there will also be other kinds of documents, not all of which fit into the UML framework. In large-scale projects a librarian or configuration manager may be required to keep track of these documents and ensure that they are stored safely and in a way that enables them to be retrieved when required. Such documents include the following:

- records of interviews and observations,
- details of problems,
- copies of existing documents and where they are used,
- details of requirements,
- details of users and
- minutes of meetings.

Even in smaller projects which cannot justify a librarian, a filing system with an agreed set of conventions on how material is to be filed, and for recording who has taken items from the filing system is good practice.

In many projects, these documents will be stored digitally, using a document management system or a version control system. In this case, many people can access the same document simultaneously. The system enforces control over whether a document can be updated, and ensures that no more than one person at a time is able to 'check out' a document in order to amend it.

Not all of the documents listed above represent requirements, and it is necessary to maintain some kind of list or database of requirements. There are software tools available to hold requirements in a database, and some can be linked to CASE tools and testing tools. This makes it possible to trace from an initial requirement through the analysis and design models to where it has been implemented and to the test cases that test whether the requirement has been met.

Use cases, which are explained in the next section, can be used to model requirements, but because they focus on the functionality of the system, are not good for documenting non-functional requirements. Jacobson et al. (1999) suggest that the use case model should be used to document functional requirements and a separate list of 'supplementary requirements' (those not provided by a use case) should be kept. They say that together, the use case model and the list of supplementary requirements constitute a

traditional requirements specification. Rosenberg and Scott (1999) argue that use cases are not the same as requirements: use cases describe units of system behaviour, whereas requirements are rules that govern the behaviour of the system; one requirement may be met by more than one use case, and one use case may meet more than one requirement; some non-functional requirements are difficult to attribute to any particular use case.

Some people try to document requirements in use cases by writing long use case descriptions using templates that enable them to include non-functional requirements as well as functional requirements. One of the authors has also come across developers who use the process of brainstorming for use cases as a way of eliciting requirements. However, this tends to produce some very odd use cases.

We favour the view that use cases can be used to model functional requirements, but a separate list of requirements should be kept, containing all requirements—functional and non-functional—for the system. Where there is a relationship between a particular use case and a particular requirement, this should be recorded. Moreover, some requirements describe very high-level units of behaviour and may need to be broken down into low-level requirements that describe more precisely what is to be done. Any database of requirements should make it possible to hold this kind of hierarchical structure of requirements.

Sometimes the process of requirement gathering throws up more requirements than can be met in a particular project. They may be outside the scope of the project, over-ambitious, too expensive to implement or just not really necessary at this point in time. The process of building a requirements model for a system involves going through all the candidate requirements to produce a list of those that will be part of the current project. Figure 6.3 shows this as an activity diagram.

6.6 Use Cases

Use cases are descriptions of the functionality of the system from the users' perspective. Use case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some way to use that functionality. Figure 6.4 shows an example of a use case diagram. This is a relatively simple diagramming technique, and its notation is explained below in Section 6.6.2.

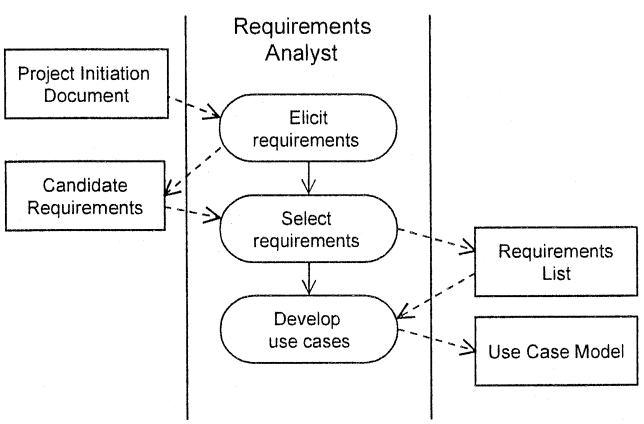

Figure 6.3 Activity diagram to show the activities involved in capturing requirements.

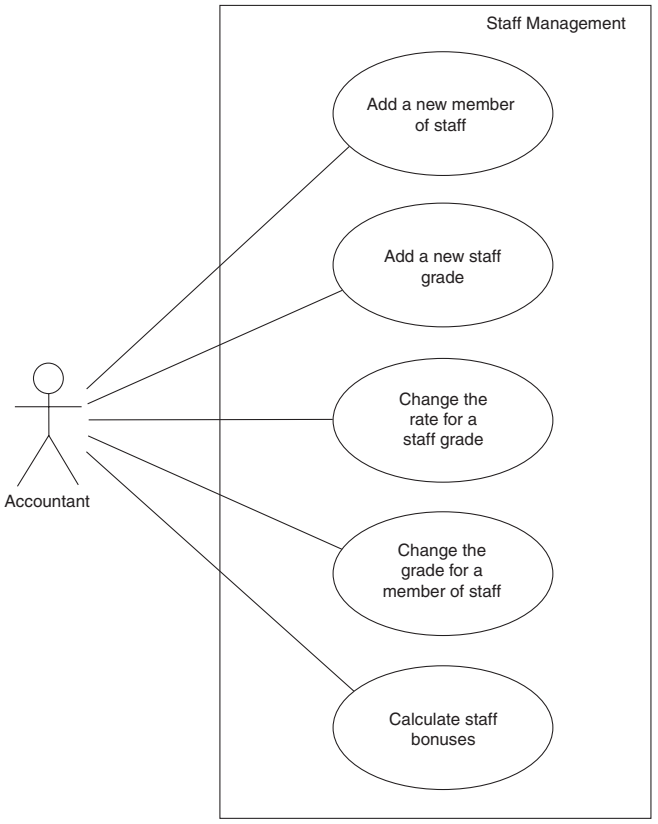

Figure 6.4 Example use case diagram.

Use case diagrams were developed by Jacobson et al. (1992), and the subtitle of the book in which they are presented is *A Use Case Driven Approach*. Jacobson and his co-authors offer a complete approach to the development of object-oriented software systems, but use case diagrams are the starting point for much of what follows in their approach.

6.6.1 Purpose

The use case model is part of what Jacobson et al. (1992) call the requirements model; they also include a problem domain object model and user interface descriptions in this requirements model. Use cases specify the functionality that the system will offer from the users' perspective. They are used to document the scope of the system and the developer's understanding of what it is that the users require.

Use cases are supported by *behaviour specifications*. These specify the behaviour of each use case either using UML diagrams, such as *collaboration diagrams* or *sequence diagrams* (see Chapter 9), or in text form as *use case descriptions*.

Textual *use case descriptions* provide a description of the interaction between the users of the system, termed *actors*, and the high level functions within the system, the use cases. These descriptions can be in summary form or in a more detailed form in which the interaction between actor and use case is described in a step-by-step way. Whichever approach is used, it should be remembered that the use case describes the interaction as the user sees it, and is not a definition of the internal processes within the system, or some kind of program specification.

6.6.2 Notation

Use case diagrams show three aspects of the system: actors, use cases and the system or sub-system boundary. Figure 6.5 shows the elements of the notation.

Actors represent the roles that people, other systems or devices take on when communicating with the particular use cases in the system. Figure 6.5 shows the actor `Staff Contact` in a diagram for the Agate case study. In Agate, there is no job title `Staff Contact`: a director, an account manager or a member of the creative team can take on the role of being staff contact for a particular client company, so one actor can represent several people or job titles. Equally, a particular person or job title may be represented by more than one actor on use case diagrams. This is shown in Figures 6.5 and 6.6 together. A director or an account manager may be the `Campaign Manager` for a particular client campaign, as well as being the `Staff Contact` for one or more clients.

The use case description associated with each use case can be brief:

Assign staff to work on a campaign

The campaign manager selects a particular campaign. A list of staff not already working on that campaign is displayed, and he or she selects those to be assigned to this campaign.

Alternatively, it can provide a step-by-step breakdown of the interaction between the user and the system for the particular use case. An example of this extended approach is provided below.

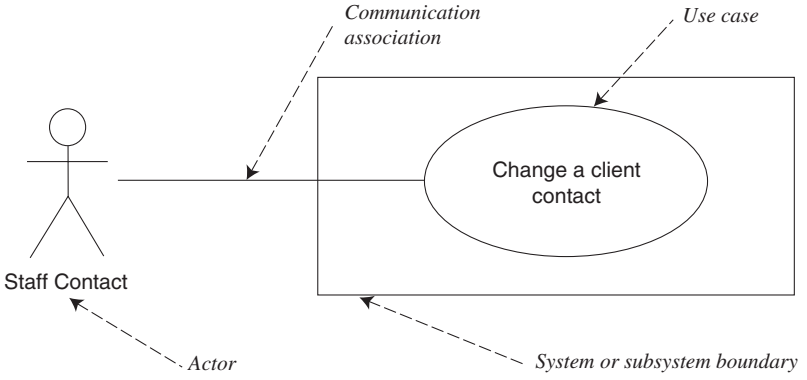

Figure 6.5 The notation of the use case diagram.

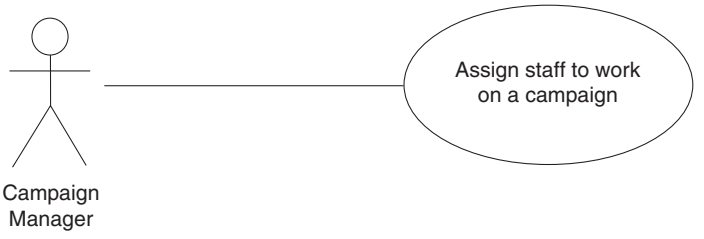

Figure 6.6 Use case showing `Campaign Manager` actor.

Assign staff to work on a campaign

	Actor Action		System Response
1.	The actor enters the client name.	2.	Lists all campaigns for that client.
3.	Selects the relevant campaign.	4.	Displays a list of all staff members not already allocated to this campaign.
5.	Highlights the staff members to be assigned to this campaign.	6.	Presents a message confirming that staff have been allocated.

Alternative Courses

Steps 1-3. The actor knows the campaign name and enters it directly.

Constantine (1997) makes the distinction between *essential* and *real* use cases. Essential use cases describe the 'essence' of the use case in terms that are free of any technological or implementation details, whereas real use cases describe the concrete detail of the use case in terms of its design. During the analysis stage, use cases are almost always essential, as the design has not yet been decided upon. In a real use case, Step 2 in the use case description for `Assign staff to work on a campaign` could be described as 'Lists all campaigns for the client in a list box, sorted into alphabetical order by campaign title.'

Each use case description represents the usual way in which the actor will go through the particular transaction or function from end to end. Possible major alternative routes that could be taken are listed as *alternative courses*. The term *scenario* is used to describe use cases in which an alternative course is worked through in detail, including possible responses to errors. The use case represents the generic case, while the scenarios represent specific paths through the use case.

As well as the description of the use case itself, the documentation should include the purpose or intent of the use case, that is to say details of the task that the user is trying to achieve through the means of this use case, for example:

The campaign manager wishes to record which staff are working on a particular campaign. This information is used to validate timesheets and to calculate staff year-end bonuses.

One way of documenting use cases is to use a template (a blank form or word-processing document to be filled in). This might include the following sections:

- name of use case,
- pre-conditions (things that must be true before the use case can take place),
- post-conditions (things that must be true after the use case has taken place),
- purpose (what the use case is intended to achieve) and
- description (in summary or in the format above).

Two further kinds of relationships can be shown on the use case diagram itself. These are the *Extend* and *Include* relationships. They are shown on the diagram using two pieces of UML notation that you will come across in other diagrams: dependencies and stereotypes.

Dependencies—a dependency is a relationship between two modelling elements where a change to one will probably require a change to the other because the one is dependent in some way on the other. A dependency is shown by a dashed line with

an open arrowhead pointing at the element on which the other is dependent. There are many kinds of dependencies in UML, and they are distinguished from one another using stereotypes.

Stereotypes—a stereotype is a special use of a model element that is constrained to behave in a particular way. Stereotypes can be shown by using a keyword, such as 'extend' or 'include' in matched *guillemets*, like «extend». (Guillemets are used as quotation marks in French and some other languages.) Stereotypes can also be represented using special icons. The actor symbol in use case diagrams is a stereotyped icon—an actor is a stereotyped class and could also be shown as a class rectangle (see Chapter 7) with the stereotype «actor» above the name of the actor. So by stereotyping classes as «actor» we are indicating that they are a special kind of class that interacts with the system's use cases. Note, however, that actors are external to the system, unlike use cases and classes.

The Extend and Include relationships are easy to confuse. «extend» is used when you wish to show that a use case provides additional functionality that may be required in another use case. In Figure 6.7, the use case Print campaign summary extends Check campaign budget. This means that at a particular point in Check Campaign Budget the user can optionally invoke the behaviour of Print campaign summary, which does something over and above what is done in Check campaign budget (print out the information in this case). There may be more than one way of extending a particular use case, and these possibilities may represent significant variations on the way the user uses the system. Rather than trying to capture all these variations in one use case, you would document the core functionality in one and then extend it in others. Extension points can be shown in the diagram, as in Check campaign budget in Figure 6.7. They are shown in a separate compartment in the use case ellipse, headed Extension points. The name of the extension point is given and a description of the point in the use case where it occurs. A condition can be shown next to the dependency relationship. This condition must be true for the extension to take place in a particular instance of the use case.

«include» applies when there is a sequence of behaviour that is used frequently in a number of use cases, and you want to avoid copying the same description of it into each use case in which it is used. Figure 6.8 shows that the use case Assign staff to work on a campaign has an «include» relationship with Find campaign. This means that when an actor uses Assign staff to work on a campaign the behaviour of Find campaign will also be included in order to select the relevant Campaign.

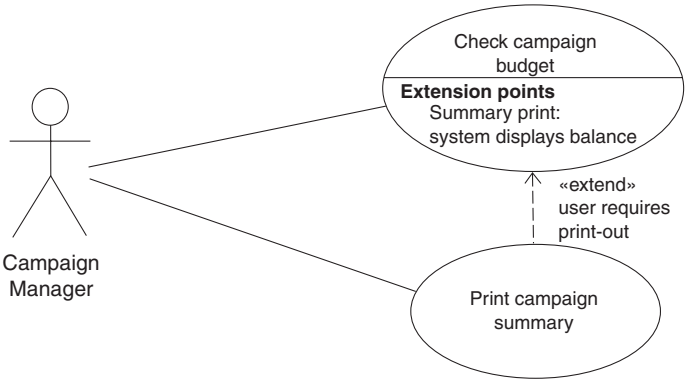

Figure 6.7 Use case diagram showing «extend»

As well as describing the use cases, it is worth describing who the actors are in terms of job titles or the way in which they interact with the system. Although at the moment, we are concentrating on requirements, later we shall need to know who the actual users are for each high level function that is represented by a use case. This may help in specifying the security for different functions or in assessing the usability of the functions.

Bear in mind that actors need not be human users of the system. They can also be other systems that communicate with the one that is the subject of the systems development project, for example, other computers or automated machinery or equipment.

Figure 6.9 shows a use case diagram for the Campaign Management subsystem with both Extend and Include relationships. Note that you do not have to show all the detail of the extension points on a diagram: the `Extension points` compartment in the use case can be suppressed. Of course, if you are using a CASE tool to draw and manage the diagrams, you may be able to toggle the display of this compartment on and off, and even if the information is not shown on a particular diagram, it will still be held in the CASE tool's repository.

In Chapter 4, the concepts of generalization, specialization and inheritance were introduced. They are explained in more detail in Chapter 8. However, generalization and specialization can be applied to actors and use cases. For example, suppose that we have two actors, `Staff Contact` and `Campaign Manager`, and a `Campaign`

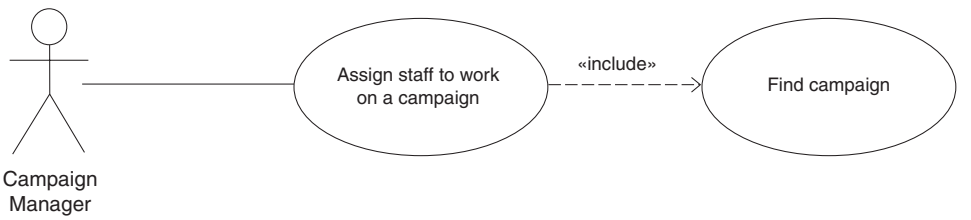

Figure 6.8 Use case diagram showing «`include`»

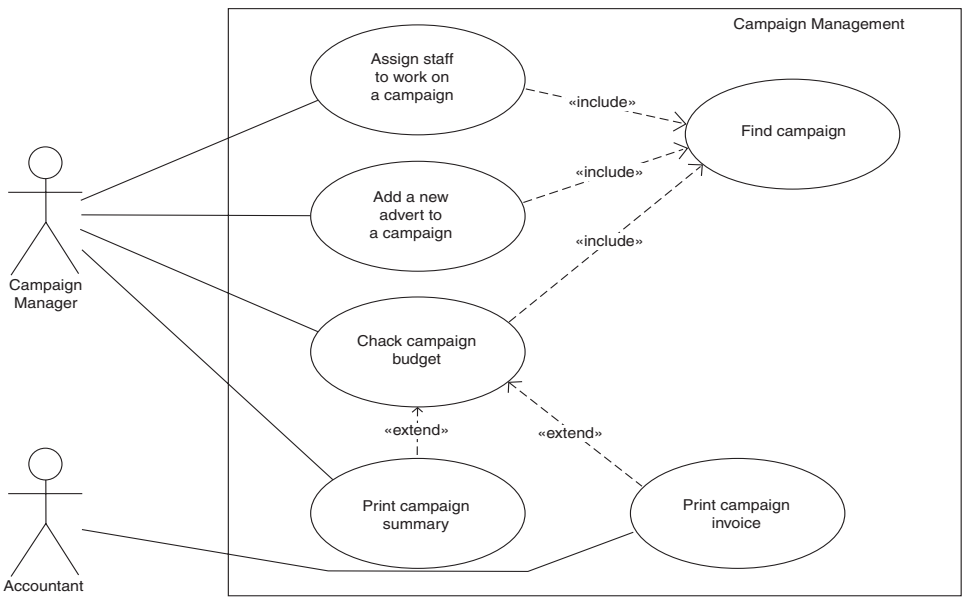

Figure 6.9 Use case diagram showing both «`extend`» and «`include`».

Manager can do everything that a Staff Contact can do, and more. Rather than showing communication associations between Campaign Manager and all the use cases that Staff Contact can use, we can show Campaign Manager as a specialization of Staff Contact, as in Figure 6.10. Similarly, there may be similar use cases where the common functionality is best represented by generalizing out that functionality into a 'super-use case' and showing separate. For example, we may find that there are two use cases at Agate Assign individual staff to work on a campaign, and Assign team of staff to work on a campaign, which are similar in the functionality they offer. We might abstract out the commonality into a use case Assign staff to work on a campaign, but this will be an abstract use case. It helps us to define the functionality of the other two use cases, but no instance of this use case will ever exist in its own right. This is also shown in Figure 6.10.

6.6.3 Supporting use cases with prototyping

As the requirements for a system emerge in the form of use cases, it is sometimes helpful to build simple prototypes of how some of the use cases will work. A prototype is a working model of part of the system—usually a program with limited functionality that is built to test out some aspect of how the system will work. (Prototypes were discussed in Section 3.2.2 and are explained in more detail in Chapter 17 on the design of the user interface.)

Prototypes can be used to help elicit requirements. Showing users how the system might provide some of the use cases often produces a stronger reaction than showing them a series of abstract diagrams. Their reaction may contain useful information about requirements.

For example, there are a number of use cases in the Campaign Management subsystem for Agate that require the user to select a campaign in order to carry out some

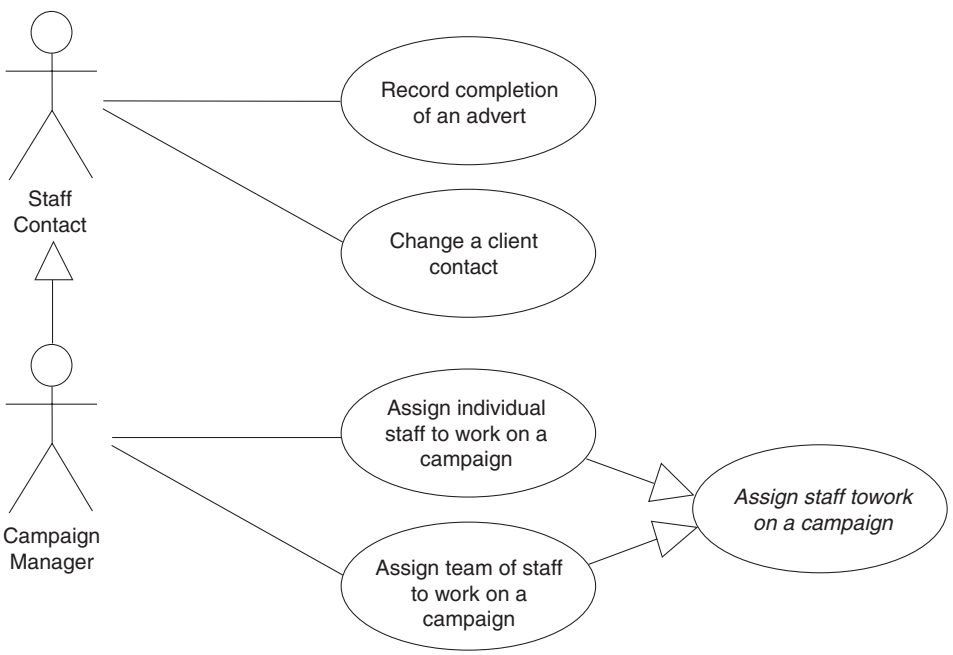

Figure 6.10 Generalization of actors and use cases.

business function. The use case diagram in Figure 6.9 reflects this in the «include» relationships with the use case `Find campaign`. The use case `Find campaign` will clearly be used a great deal, and it is worth making sure that we have the requirements right. A prototype could be produced that provides a list of all the campaigns in the system. A possible version of this is shown in Figure 6.11.

Showing this prototype interface design to the users may well produce the response that this way of finding a campaign will not work. There may be hundreds of campaigns in the system, and scrolling through them would be tedious. Different clients may have campaigns with similar names, and it would be easy to make a mistake and choose the wrong campaign if the user does not know which client it belongs to. For these reasons, the users might suggest that the first step is to find the right client and then only display the campaigns that belong to that client. This leads to a different user interface—shown in Figure 6.12.

The information from this prototyping exercise forms part of the requirements for the system. This particular requirement is about usability, but it can also contribute to meeting other, non-functional requirements concerned with speed and the error rate: it might be quicker to select first the client and then the campaign from a short-list than it is to search through hundreds of campaigns; and it might reduce the number of errors made by users in selecting the right campaign to carry out some function on.

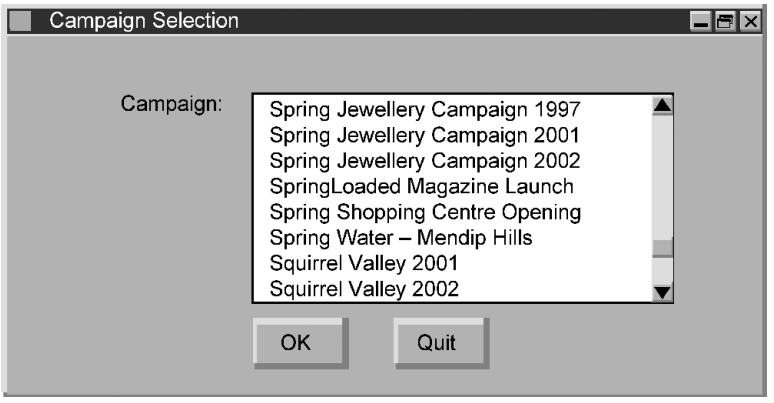

Figure 6.11 Prototype interface for the `Find campaign` use case.

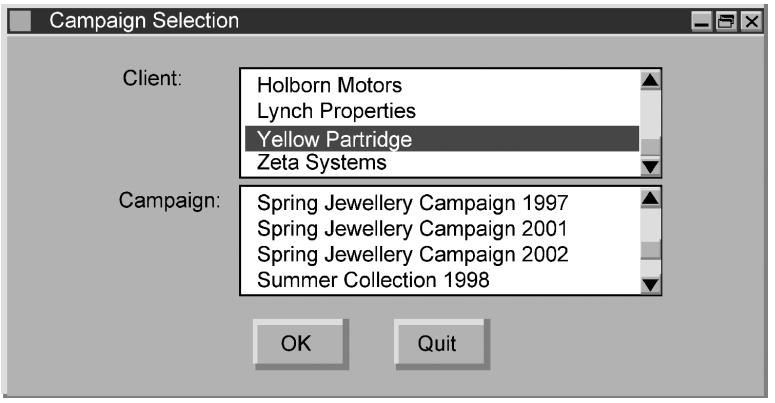

Figure 6.12 Revised prototype interface for the `Find campaign` use case.

Prototypes can be produced with visual programming tools, with scripting languages like TCL/TK, with a package like Microsoft PowerPoint® or even as web pages using HTML.

Prototypes do not have to be developed as programs. Screen and window designs can be sketched out on paper and shown to the users, either formally or informally. A series of possible screen layouts showing the steps that the user would take to interact with a particular use case can be strung together in a storyboard, as in Figure 6.13.

6.6.4 CASE tool support

Drawing any diagram and maintaining the associated documentation is made easier by a CASE tool, as described in Section 3.6.

As well as allowing the analyst to produce diagrams showing all the use cases in appropriate subsystems, a CASE tool should also provide facilities to maintain the repository associated with the diagram elements, and to produce reports. Automatically generated reports can be merged into documents that are produced for the client organization. The behaviour specification of each use case forms part of the requirements model or requirements specification, which it is necessary to get the client to agree to.

6.6.5 Business modelling with use case diagrams

We have used use case diagrams here to model the requirements for a system. They can also be used earlier in the life of a project to model an organization and how it operates. Business modelling is sometimes used when a new business is being set up, when an existing business is being 'reengineered', or in a complex project to ensure that the business operation is correctly understood before starting to elicit the requirements.

In the examples that we have shown above, the actors have all been employees of the company interacting with what will eventually be at least in part a computerized system. In business modelling, the actors are the people and organizations outside the company, interacting with functions within the company. For example, Figure 6.14 shows the `Client` as an actor and use cases that represent the functions of the business rather than functions of the computer system.

A full business model of Agate would show all the functions of the company, and the actors would be the other people and organizations with which Agate interacts, for example the media companies (TV stations and magazine and newspaper publishers) from which Agate buys advertising time and space, and the subcontractors that Agate uses to do design work and printing.

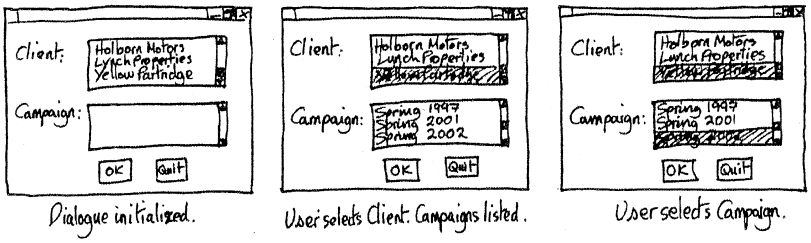

Figure 6.13 Prototype storyboard.

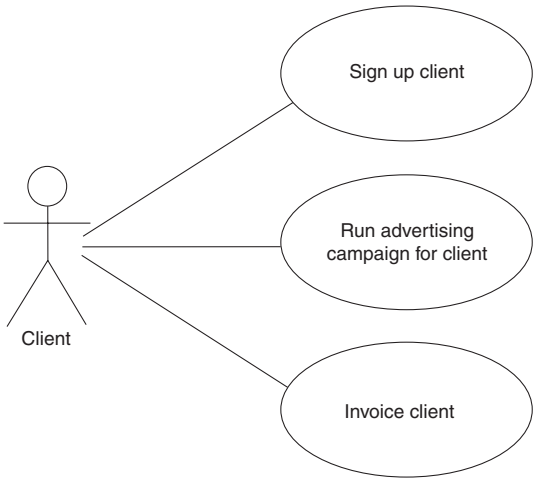

Figure 6.14 Example of business modelling with use cases.

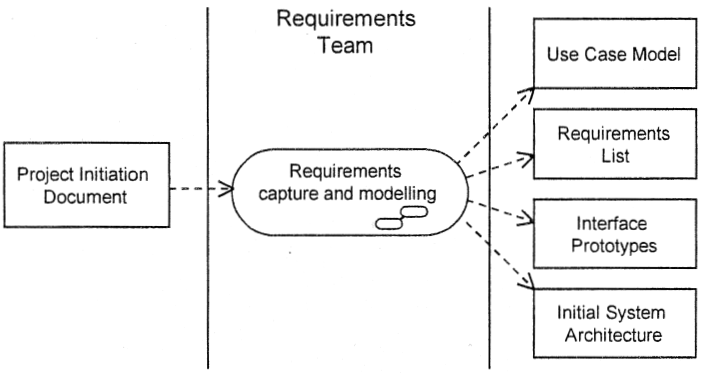

Figure 6.15 Activity diagram for `Requirements capture and modelling`.

6.7 Requirements Capture and Modelling

The first stage of most projects is one of capturing and modelling the requirements for the system. As we progress through the book, we shall include activity diagrams to illustrate the main activities in and products of each phase. These diagrams link back to the table in Figure 5.18, which summarizes the approach that we are taking in this book. Figure 6.15 shows the first such diagram.

In this case we have not broken the activity `Requirements capture and modelling` down into more detail, though it could potentially be broken down into separate activities for the capture of the requirements (interviewing, observation, etc.) and for the modelling of the requirements (use case modelling, prototyping, etc.).

We have used object flows to show the documents and models that are the inputs to and outputs from activities, and swimlanes to show the role that is responsible for the activities. In this case, one or more people in the role of `Requirements Team` will carry out this activity. In a small project, this may be one person, who carries out

many other analysis and design activities; in a large project or organization, this may be a team of requirements analysis specialists taking more specialist roles.

The Case Study Chapter A2, which follows this one, provides more extended examples of the outputs of the `Requirements capture and modelling` activity, and the book website provides a full use case model.

6.8 Summary

Analysts investigating an organization's requirements for a new information system may use five main fact finding techniques—background reading, interviews, observation, document sampling and questionnaires. They use these to gain an understanding of the current system and its operation, of the enhancements the users require to the current system and of the new requirements that users have for the new system.

Using agreed standards to document requirements allows the analysts to communicate these requirements to other professionals and to the users. Use case diagrams are one diagramming technique that is used to summarize the users' functional requirements in a high level overview of the way that the new system will be used.

Case Study Example

You have already seen several examples from the case study in this section. The use cases are determined by the analyst from the documentation that is gathered from the fact-finding process. What follows is a short excerpt from an interview transcript, which has been annotated to show the points which the analyst would pick up on and use to draw the use case diagrams and produce the associated documentation. The interview is between Dave Harris, a systems analyst, and Peter Bywater, an Account Manager at Agate. It is from one of the interviews with the objective 'To establish additional requirements for new system' in the fact finding plan in the earlier case study section in this chapter.

Dave Harris: You were telling me about concept notes. What do you mean by this?
Peter Bywater: At present, when we come up with an idea for a campaign we use a word-processor to create what we call a concept note. We keep all the note files in one directory for a particular campaign, but it's often difficult to go back and find a particular one.
DH: So is this something you'd want in the new system?
PB: Yes. We need some means to enter a concept note and to find it again.
(This sounds like two possible use cases. Who are the actors?)
DH: So who would you want to be able to do this?
PB: I guess that the staff working on a campaign should be able to create a new note in the system.
DH: Only them?
(Any other actors?)
PB: Yes, only the staff actually working on a campaign.
DH: What about finding them again? Is this just to view them or could people modify them?
PB: Well, we don't change them now. We just add to them. It's important to see how a concept has developed. So we would only want to view them. But we need some easy way of browsing through them until we find the right one.
(Who are the actors for this?)

DH: Can anyone read the concept notes?
PB: Yes, any of the staff might need to have a look.
DH: Would you need any other information apart from the text of the concept itself?
(Thinking ahead to Chapter 7!)
PB: Yes. It would be good to be able to give each one a title. Could we use the titles then when we browse through them? Oh, and the date, time and whoever created that concept note.
DH: Right, so you'd want to select a campaign and then see all the titles of notes that are associated with that campaign, so you could select one to view it?
(Thinking about the interaction between the user and the system.)
PB: Yes, that sounds about right.
...
From this information, Dave Harris is going to be able to develop the use case descriptions for two use cases:

Create concept note,
Browse concept notes.

The use case diagram is shown in Figure 6.16. The use case descriptions will be as follows.

Create concept note.
A member of staff working on a campaign can create a concept note, which records ideas, concepts and themes that will be used in an advertising campaign. The note is in text form. Each note has a title. The person who created the note, the date and time are also recorded.

Browse concept notes.
Any member of staff may view concept notes for a campaign. The campaign must be selected first. The titles of all notes associated with that campaign will be displayed. The user will be able to select a note and view the text on screen. Having viewed one note, others can be selected and viewed.

The interaction here is quite straightforward, so we shall not need a more detailed breakdown of the interaction between user and system.

Note that in Figure 6.16, because Campaign Staff is a specialization of Staff, we do not need to show a communication association between the Campaign Staff actor and the Browse concept notes use case.

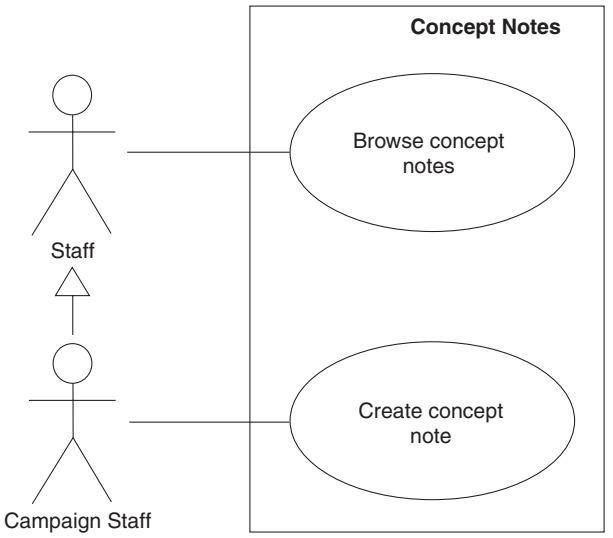

Figure 6.16 Use cases for `Concept Notes` subsystem.

Review Questions

6.1 Read the following description of a requirement for FoodCo, and decide which parts of it are functional requirements and which are non-functional requirements.

The allocation of staff to production lines should be mostly automated. A process will be run once a week to carry out the allocation based on the skills and experience of operatives. Details of holidays and sick leave will also be taken into account. A first draft Allocation List will be printed off by 12.00 noon on Friday for the following week. Only staff in Production Planning will be able to amend the automatic allocation to fine-tune the list. Once the amendments have been made, the final Allocation List must be printed out by 5.00 pm. The system must be able to handle allocation of 100 operatives at present, and should be capable of expansion to handle double that number.

6.2 Name the five main fact finding techniques and list one advantage and one disadvantage of each.

6.3 Imagine that you will be interviewing one of the three staff in Production Planning at FoodCo. Draw up ten questions that you would want to ask him or her.

6.4 What is the purpose of producing use cases?

6.5 Describe in your own words the difference between the «extend» and «include» relationships in use case diagrams.

6.6 What is the difference between an 'essential' and a 'real' use case?

6.7 Write a use case description in the extended form, used for the `Assign staff to work on a campaign` example in Section 6.6.2, for either `Create concept note` or `Browse concept notes`.

6.8 Think of the different possible uses you could make of a library computer system and draw a use case diagram to represent these use cases.

6.9 List some non-functional requirements a library computer system (as in Question 6.8) that you would not model using use cases.

6.10 In what way are use case diagrams different when used for business modelling?

Case Study Work, Exercises and Projects

6.A Refer to the material for the second case study—FoodCo (introduced in Case Study B1). Draw up your initial fact finding plan along the lines of the plan given above.

6.B Read the following excerpt from a transcript of an interview with one of the production planners at FoodCo. Draw a use case diagram and create use case descriptions for the use cases that you can find in this information.

Ken Ong: So what happens when you start planning the next week's allocation?
Rik Sharma: Well, the first thing to do is to check which staff are unavailable.
KO: Would that be because they are on holiday?
RS: Yes, they could be on holiday, or they could be off sick. Because staff are handling raw food, we have to be very careful with any illness. So factory staff often have to stay off work longer than they would if they were office workers.
KO: So how do you know who's off sick and who's on holiday?
RS: They have to complete a holiday form if they want a holiday. They send it to the Factory Manager, who authorizes it and sends it to us. We take a copy, and enter the details into our system. We then return the form to the member of staff.
KO: What details do you enter?
RS: Who it is, the start date of the holiday and the first date they are available for work again.
KO: What about illness?
RS: The first day someone is off sick they have to ring in and notify us. We have to find someone to fill in for them for that day if we can.
KO: Right. Let's come back to that in a minute. How do you record the fact that they're off sick for your next week's production plan?
RS: We make an entry in the system. We record which member of staff it is, when they went off sick, the reason and an estimate of how many days they're likely to be off.
KO: Right, so how do you get at that information when you come to plan next week's allocation?
RS: Well, we run off three lists. We enter Monday's date, and it prints us off one list showing who is available all week, a second list showing who is not available all week, and a third list showing who is likely to be available for part of the week.
KO: Then what?
RS: Then we start with the people who are available all week and work round them. We pull each operative's record up on the screen and look at two main factors—first their skills and experience, and second which line they're working on at the moment and how long they've been on that line. Then we allocate them to a line and a session in one of the three factories.
KO: So you can allocate them to any one of the three factories. Do you enter the same data for each one?
RS: No, there are slight variations in the allocation screen for each of the factories—mainly for historical reasons.
...

Further Reading

- Booth (1989) Chapter 5 describes the issues surrounding the usability of systems in more detail than we can here, and explains the process of Task Analysis.
- Oppenheim's book (2000) provides a very detailed coverage of questionnaire design for survey purposes. It is aimed mainly at social science and psychology students, but has some relevant chapters on how to formulate effective questions. Many books for students on how to carry out a research project cover fact-gathering techniques such as interviewing and questionnaire design. Allison et al. (1996) is an example, but most university libraries and bookshops will have a selection of similar books.
- Hart (1997) gives a detailed explanation of the techniques that are specific to the development of expert systems.
- Roberts (1989) addresses the role of users in a systems development project. This book is one of a series of guides written for civil servants in the UK government service, and is relatively bureaucratic in its outlook. However it ranges widely over the issues that users may face. Yourdon (1989) discusses users and their roles in Chapter 3.
- Jacobson et al. (1992) present the original ideas behind use cases as an analysis technique. For a more recent view, look at Rosenberg and Scott (1999) or Cockburn (2000).
- Examples of use cases from the Agate case study are available on the book website. This includes references to examples of templates for use case descriptions.