Chapter 1. Introduction

Objectives

- To introduce Requirements Engineering
- To explain why a requirements stage is necessary
- To present a definition of Requirements Engineering (RE) and to introduce a general model of the RE process
- To discuss the qualities a Requirements Engineer
- To describe various approaches to the problem of requirements
- To introduce the rationale for this book

1.1 Introduction

Requirements Engineering, is concerned with what needs to be designed rather than how it is to be designed. Requirements Engineering (RE) is also concerned with some future situation. Consider, figure 1 below, here the inputs to system design are shown as the user's present job and the technological options and the output as the future system. Drawn in this way it seems quite straightforward.

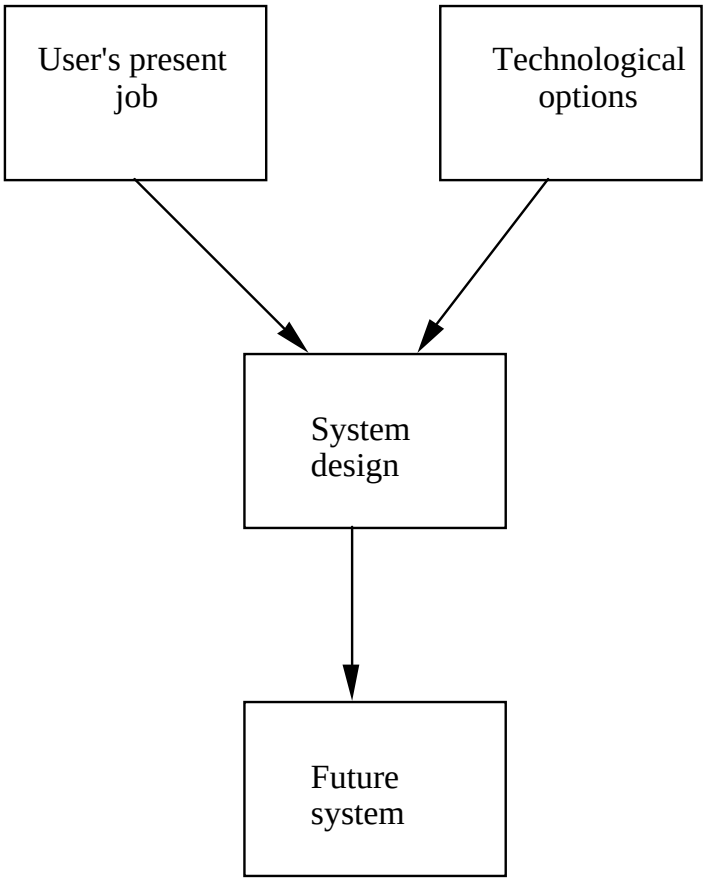


Figure 1 The process of design?

However, it is not so simple. How can the system be designed before the future situation is known? What is it that needs to be designed? In other words what are the requirements for the design?

Before design can start, there must be some knowledge of the future situation which includes the future system. Requirements Engineering is concerned with finding out about the future situation and the associated change. It is concerned with gathering information and considering possible options and with identifying what should be designed in order to meet some perceived future need.

In the waterfall model of system development, shown in figure 2, requirements is positioned before design. However, the arrows indicate that the requirements may not be a once and for all activity, but that information which affects the requirements may be discovered at later stages in the lifecycle and that the requirements may need to be changed.

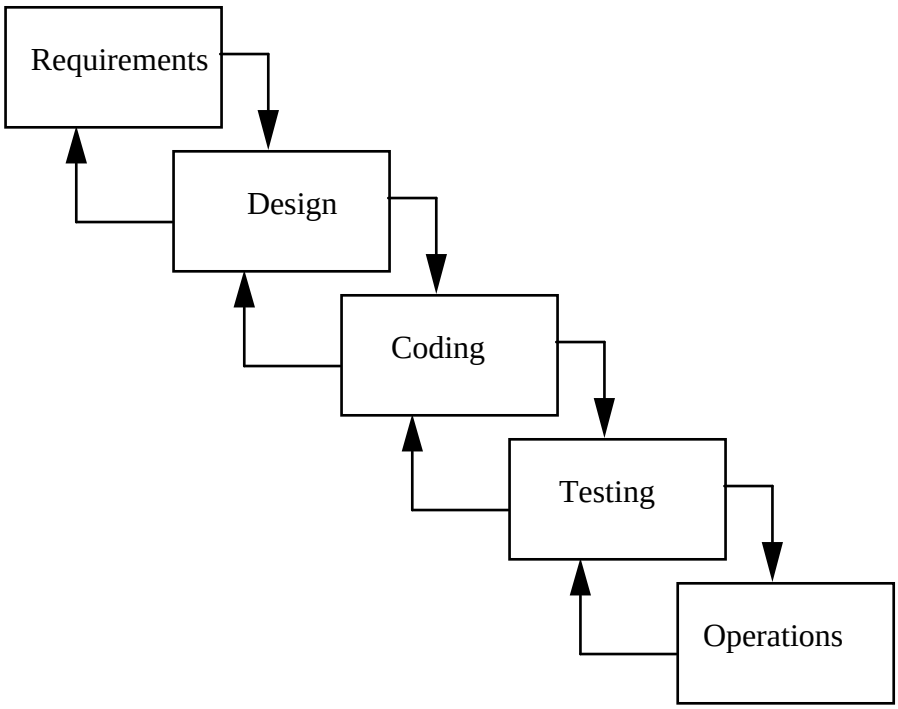


Figure 2 The position of requirements in the standard waterfall lifecycle model

A slightly different view of requirements is presented in the ‘V’ lifecycle model, see figure 3. In this case requirements is shown in two stages : the feasibility study and the product definition. The feasibility study normally involves an investigation into some customer need and the cost benefit analysis of various alternative solutions. The product definition stage is concerned with specifying the requirements for the design.

Figure 3 also shows an arrow going from product definition to evaluation and acceptance of the system. This illustrates the fact that requirements do not disappear once design starts, but that the developed system must be tested against the requirements. Thus Requirements Engineering must be concerned with describing the requirements in a way that lends itself to testing.

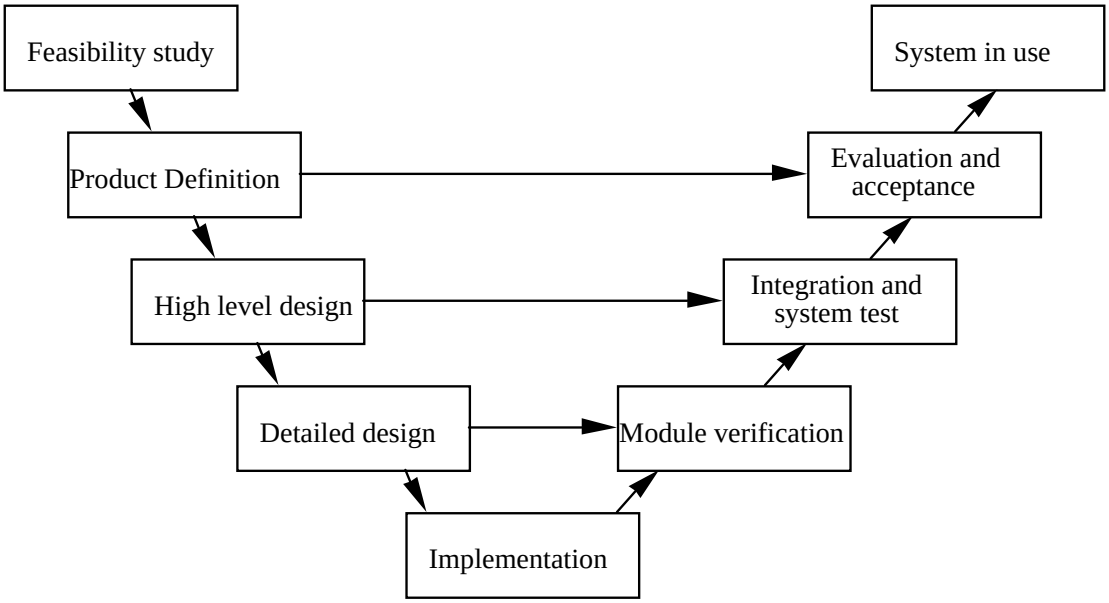


Figure 3 The Position of requirements in the standard V model

From a designers point of view a requirements stage is necessary because it results in a description of what needs to be designed.

1.2 Why is a requirements stage necessary?

From a customers point of view the situation is different. The customer lives in an ever changing environment, in which new challenges and opportunities, new problems and constraints, occur on an almost daily basis. Current systems will never be quite adequate for the future situation that the change will bring.

Figure 4 illustrates a typical system usage cycle from a customer point of view. A customer first identifies a need for a system or product. The system is purchased and installed, users are trained how to use it, jobs may be redesigned, parts of the organisation may be restructured, and if all goes well the use of the system expands from limited usage to full usage.

Eventually, because of change in the environment, the limits of the system are reached. The customer evaluates the situation and decides that there is a need for change in the supporting system.

Thus in figure 4, the requirements stage comes under the heading of ‘needs assessment’.

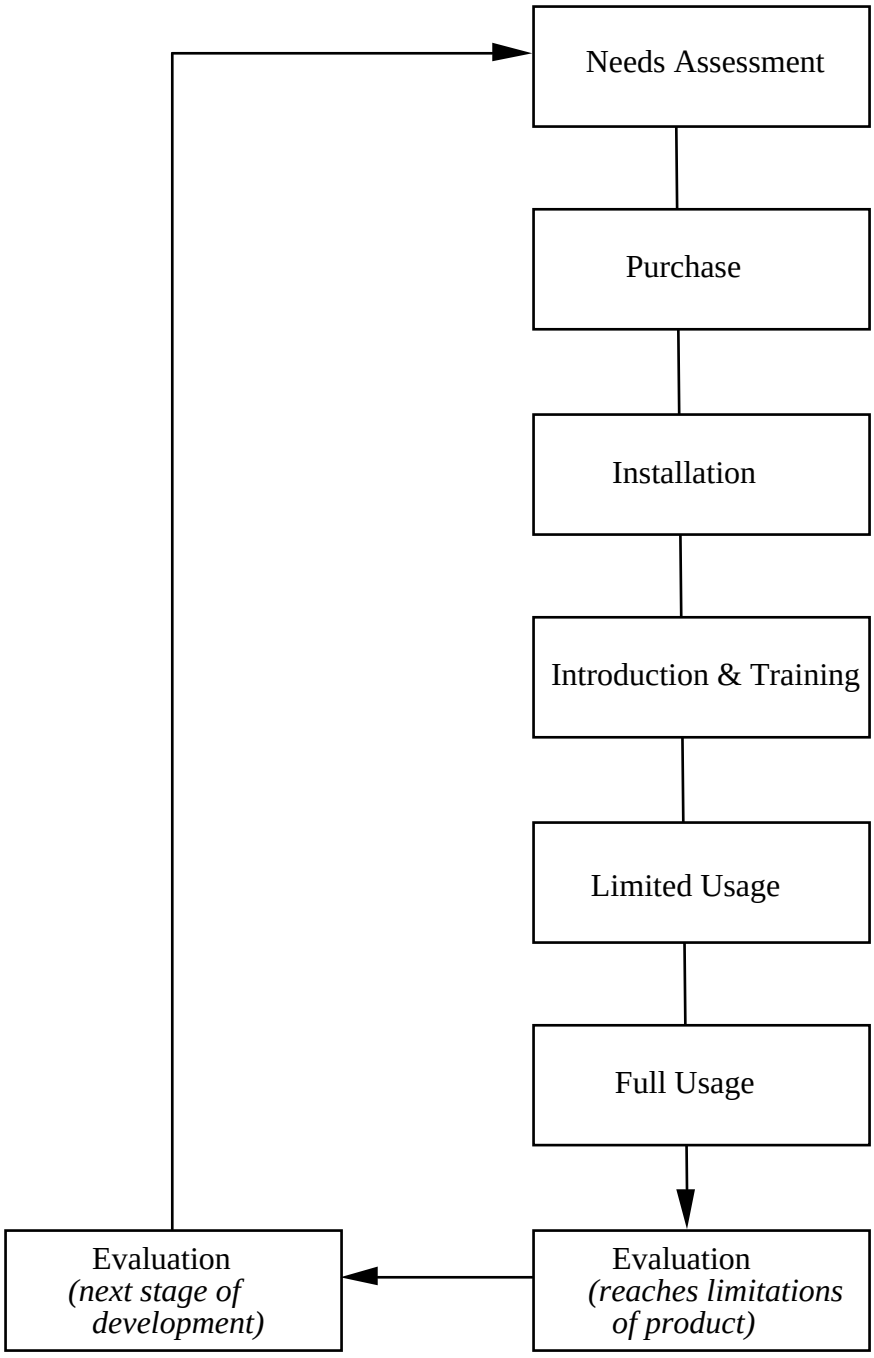


figure 4 The Customer Usage Cycle

Of course, ‘needs assessment’ may not always result in the purchase and installation of a new system. It may simply be that some additional feature is needed, or a simple upgrade

to a faster or more efficient version of the present system. In some cases it may be that the result of the needs assessment is that more effective use of personnel should be made.

Thus, from a customer's point of view a requirements stage is necessary because it helps to understand the new needs and to identify how they can be satisfied.

1.3 What is a Requirement?

In simple terms a requirement could be defined as 'something which a customer needs'. However, from a designer's point of view it could also be defined as 'something which needs to be designed'. These may or may not be the same.

There are a number of definitions of the term requirements. The most notable being the IEEE-Standard 610 (1990):

1. *A condition or capacity needed by a user to solve a problem or achieve an objective.*
2. *A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.*
3. *A documented representation of a condition or capability as in 1 or 2.*

The IEEE standard places emphasis on software requirements, although Loucopoulos and Karakostas (1995) suggest that the definition is general enough to apply to none software specific situations.

In contrast to the IEEE standard the BSI standard places the emphasis on user requirements. The BSI BS 6719 : 1986 standard guide to specifying user requirements for computer based system does not provide a definition of requirements, but rather provides a basis for describing user needs and priorities.

Both standards recommend a contents list for a requirements document which reflects their different perspectives (see chapter two for further details). The requirements document itself being a statement of the requirements.

1.4 Requirements Engineering

If the requirements document is considered to be the end product of the requirements stage then Requirements Engineering (RE) can be thought of as the process by which the requirements document is populated.

Pohl (1993) in his paper on the 'three dimensions of requirements engineering' provides one of the first definitions of RE:

"Requirements Engineering can be defined as the systematic process of developing requirements through an iterative co-operative process of analysing the problem,

documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained."

This definition suggests that it may be rather simplistic to consider RE only in terms of a process which enables us to populate a requirements document. However the contents lists of the various standards may well be helpful because they provide one view as to the end point of the requirements engineering process. (This is discussed in more detail in section 2.4).

The Pohl definition is important because it highlights some of the complexities of RE. Each part of the definition leads to a number of questions:

"the systematic process of developing requirements"

- How can the process be systematic when there are many unknown factors at the beginning of the process. How can we take a step by step approach when we don't know how many steps are needed or when it may be unclear that the end has been reached?

"through an iterative co-operative process of analysing the problem"

- How many iterations will be needed?
- How do we know when 'enough' requirements have been gathered?
- What is meant by 'enough'?
- 'Co-operative' refers to cooperation between people, who should be involved in the process? how are they to communicate with each other? how will they reach agreement on the process?
- Should users be active participants? should they be involved in decision making or should they simply be sources of information?

"documenting the resulting observations in a variety of representation formats"

- What representation format should be used?
- How should the results be documented?
- What standards and which notations should be adopted and why?

"checking the accuracy of the understanding gained"

- How will we know when the process is finished?
- How accurate do the requirements need to be? what does accuracy mean in this context?
- Will everyone who is involved in the requirements process have the same understanding?

The Pohl definition was introduced in this section and a number of questions raised concerning the RE process, these questions are revisited throughout the book. In the next section a general model of the RE process is presented.

1.5 The Requirements Engineering Process

In his book on software requirements, Davis (1993) describes two types of activity which occur during the requirements phase of a project.

The first is problem analysis, where "analysts spend their time brainstorming, interviewing people who have the most knowledge about the problem on hand, and identifying all the possible constraints on the problem's solution."

The second activity is product description when "it is time to take pen in hand and to make some difficult decisions and prepare a document that describes the expected external behaviour of the product....."

Davis recognises that these two activities are not necessarily sequential or mutually exclusive. While the first activity is characterised by uncertainty, an expansion of information and knowledge, the second activity is characterised by organisation of ideas, resolving conflicting views and eliminating inconsistencies and ambiguities.
Davis also recognises that the techniques employed by each activity will be quite different.

However, there is not just one single model of the process, but many, in chapter six , seven different models of the RE process are suggested, though not exhaustive, they highlight the fact that different situations will require different process models.

However, by way of introduction a general model of the requirements process is given below:

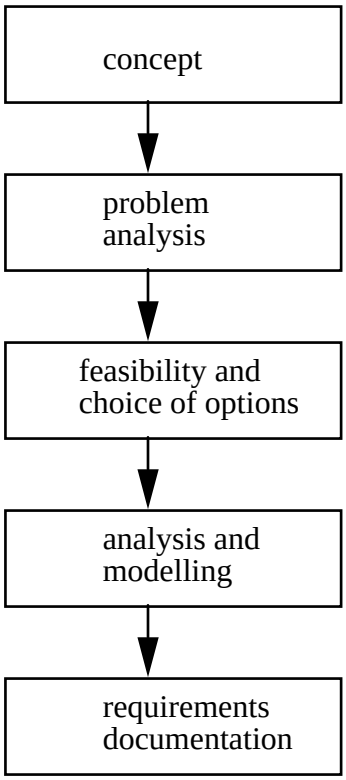


Figure 5 Activities within Requirements

The product concept provides a trigger for the requirements process to begin. That trigger might be an improvement in customer service, a future need, a small incremental improvement to the existing system or a need to use some technology which is available. Problem analysis is concerned with developing an understanding of the nature of the problem associated with the product concept. Once the problem is fully understood then possible solutions can be suggested. Feasibility and choice of options is concerned with evaluating the costs and benefits of alternative solutions. Once a given solution has been decided upon then a more detailed analysis may take place. Modelling of the application domain and of the 'solution space' may occur. Each activity should be followed by validation in order to check the accuracy of information gathered and the understanding gained. The requirements document can then be completed.

It is argued in this text that the model of the requirements process will depend to a large extent on the customer supplier relationship. A number of different customer supplier relationships are discussed in chapter six.

Thus requirements engineering and the requirements engineering process are to some extent situation dependent. Indeed this is one of the reasons why it is difficult to define the task of the Requirements Engineer. None the less the next section highlights some of the engineering skills and personal qualities that are expected of a Requirements Engineer.

1.6 The Requirements Engineer

As part of the research for this book the author attempted to find out more about industry's view of the requirements engineer, a short survey was undertaken of 32 companies whose representatives attended the RE'95 conference. A total of 18 responses were obtained from 17 companies. There was a diversity of industry types including telecommunications, atomic energy, defence research, electronics, banking, aerospace, nuclear fuels and software consultancies. The companies were from USA, Canada, Finland, England, Italy, Sweden and Belgium.

Recipients were asked to imagine they were looking for someone for a job which involves a significant portion of requirements engineering, and to answer the following questions:

Question 1:
'If you were recruiting someone to do requirements engineering what job title would you use?'
The responses were:
Software Engineer, Researcher, Requirements Engineer, Requirements Analyst, Systems Analyst, Systems Engineer, Systems Engineer for Business, Requirements Modeller.

Question 2:
'What job description would you provide?'
Below is an aggregation of responses:
"Experience of Requirements. Ability to undertake feasibility studies, functional studies, interviews, meetings, preparation of draft documents, network management modelling. Anything from understanding business needs to writing detailed system specs, for example, requirements elicitation, workshop facilitation, business modelling (process, data or object oriented), prototyping.
To manage requirements traceability throughout the project lifecycle providing impact assessment of changes and reporting on deficiencies in requirements coverage through the design and test phase.
Meeting the needs of the projects you work on, achieving customer satisfaction, seeking ways to improve processes, taking responsibility for your own development."

Question 3:
What personal qualities would you expect that person have?
The ability to:
Make himself or herself understood, listen, stay calm and assured under fire, quickly assimilate information, talent for sorting and analysing information, write clear, well structured documents, make presentations, chair meetings, run a group.
Also patience, perseverance, be able to live comfortably in a constant state of ambiguity, both independence and team working skills, negotiation skills, flexibility, open-mindedness, sense of humour.

Question 4:
What methods, tools or techniques would you expect that person to be familiar with?

CASE tools, process modelling, object-oriented modelling, interviewing, workshopping, viewpoint analysis, prototyping, information analysis, requirements traceability tools, appreciation of configuration management with respect to requirements.
Several respondents mentioned the need for knowledge of a range of techniques and the 'pros and cons' of their use.

Question 5:
What degree title would that person normally have?
About half said: B.Sc., MSc or PhD in a computing subject or numerate subject.
Others said not necessarily computing possibly business studies or psychology.

Question 6:
Would you expect the person to possess specific domain knowledge?
About half the respondents said yes, the other half thought it was desirable but not essential.

Question 7:
If that person was a new graduate what additional training would you expect to give them?
Training in interpersonal communication skills. Work experience in product development. Shadow experience analysts. Get them involved in difficult projects.
Courses in concurrent engineering, product planning and marketing.
Training in one or two techniques.

Question 8:
How long would you expect a new graduate to take before becoming effective in the job?
Most respondents said: To learn the ropes, 6 months; To work independently 12 to 18 months; To become a recognised 'domain expert' 2 to 4 years.

Recurring themes in the comments made by the respondents were that the Requirements Engineer should be aware of:

- Requirements engineering is not an isolated front-end activity to a software lifecycle process; rather, it is an integral part of the larger process connected to other parts through continuous feedback loops.

- The nature of requirements is such that change will necessarily occur; accordingly, requirements specifications and subsequent software lifecycles should be designed with change in mind.

- There are many sources of requirements which need to be explored and their input assimilated; there is no such thing as an all-inclusive source.

- There will be a continuous need to justify requirements and relate them to their sources; also to relate design and implementation decisions to the requirements they originate from.

- There will always be conflicts in requirements which will need to be accommodated and trade-offs that address these conflicts will need to be justified

According to the respondents the desirable skills that an Requirements Engineer should possess include:

- Interviewing skills to facilitate the acquisition of information
- Groupwork skills, including participating in meetings and the ability to work in a collaborative way
- Facilitation skills, such as the ability to lead a group
- Negotiation skills to support consensus building
- Analytical skills to support the analysis of an organisational situation prior to any proposals for solutions
- Problem solving skills to support the search for alternative solutions.
- Presentation skills, including the ability to write coherent documents using word processors and other presentation tools
- Modelling skills including business, process, data and object modelling, using a variety of notations.

Finally, the fundamental and practical knowledge required for the Requirements Engineer includes:

- Knowledge of and experience in using CASE tools
- Knowledge of and experience in using general modelling techniques, including formal languages and conceptual modelling
- Knowledge of and experience in using particular modelling languages and the ability to choose the best one among them for a given problem
- Knowledge of and experience with management and traceability tools
- Knowledge of concepts, tools and techniques in human computer interaction
- Knowledge of techniques in product planning and marketing

To summarise, the Requirements Engineer will need to be a Superman or Wonderwoman!!!!!

The problem faced by the Requirements Engineer is that a wide range of knowledge and skills are needed. Consequently a wide range of techniques may need to be employed. The next section describes potential sources of techniques.

1.7 Approaches to the problem of requirements

Approaches to requirements engineering vary depending of the interest of the groups from which they originate. Nine groups are listed below together with their principle interest in the problem of requirements:

1. Marketing :

interested in the relationship between requirements and the success of a product in the market place.

2. Psychology and Sociology :
   interested in the relationship between requirements and needs of people as intelligent and social beings.
3. Object Oriented Analysis :
   interested in the relationship between requirements and the software development process starting from a real world objects perspective.
4. Structured Analysis :
   interested in the relationship between requirements and the software development process starting from an process and data perspective.
5. Participative Design :
   interested in requirements as part of the process of empowering users by actively involving them in the design of systems which affect their own work.
6. Human Factors and Human Computer Interaction (HCI) :
   interested in the acceptability of systems to people, the usability of systems and the relationship between requirements and evaluation of the system in use.
7. Soft Systems :
   interested in the relationship between requirements and how people work as part of a organisational system
8. Quality :
   interested in the relationship between requirements and the quality of a product, and in relation to process improvements which lead to customer satisfaction.
9. Formal Computer Science :
   interested in the relationship between requirements and software engineering's need for precision.

Each group advocates the use of specific techniques or methods within requirements engineering. A brief overview is given below:

1.7.1 Marketing

user support
User support (sometimes known as customer support, technical support or the help desk) is a valuable source of feedback on existing products. Logs of user problems, analysis of complaints and requests for upgrades can be used to inform future releases and future products.

surveys
Surveys usually take the form of questionnaires administered to a sample of customers. Designing questionnaires and choosing appropriate sample populations is a task for professionals in market surveys. The requirements engineer would need specific training in how to design questionnaires and how to analyse results.

user groups

User groups provide an opportunity for suppliers to meet with customers to discuss the use of their products. User groups enable users from different organisations to share experiences and in this way to further understand their own problems and requirements.

trade shows
Trade shows provide an opportunity for suppliers to discuss prototypes with potential customers, and for users to experience possible future technologies.

focus groups
Focus groups could form part of a User Group meeting or of a Trade Show. The focus group involves a small group of customers coming together to have a loosely structured discussion about a particular product. The group will have a facilitator. See Morgan, 1988, for further details also focus groups are discussed in chapter five of this book.

market analysis
There are many market analysis techniques such as market segmentation (see chapter two), market life cycles, product life cycles and competitor analysis (see QFD chapter four) which can be used to identify customer requirements for computer products.

1.7.2 Psychology/Sociology

Interviewing users
Interviewing is a basic tool of the requirements engineer. Focused interviews may be used in the early stages of the requirements investigation when the RE is attempting to find out about the problem domain and the concerns of users. In this case the interviewer prepares a list of topics for discussion but not precise questions. The user is informed of the list of topics at the beginning of the interview, each topic is explored in turn in order to find out factual information about the users role, the problems encountered and the functions he or she performs. The interview is normally recorded on a tape recorder to facilitate later analysis. A structured interview is most appropriate when more detailed information is needed. In this case a list of specific questions is prepared. The interviewer should have a good knowledge of the problem domain and the terminology in order to adapt the questioning according to the responses. See Moser and Kalton, 1971 for further details of the techniques and Suchman and Jordan, 1990 for a discussion of some of the problems of face to face survey interviews.

Observations
In addition to asking questions the requirements engineer could observe the user at work. This facilitates a deeper understanding of the environment and the day to day difficulties associated with the user's job. Observations need to be accompanied by some mechanism for recording the findings, either using paper and pen to keep a log of the proceedings against some predetermined proforma or through the use of video recordings. (REF**)

Videorecording
Videorecording offers a means of studying users and their work. Sophisticated video recorders and playback facilities enable efficient and in-depth analysis of recordings, for example, through use of time stamping and annotation of images. (REF**)

Think-aloud experiments
Think-aloud experiments can be used alongside other techniques such as interviewing, user responses are recorded on an audio tape. One approach is known as ‘introspection’ where the user is asked to describe in words how they would solve a particular problem. Another approach is to ask the user to identify and describe a ‘critical incident’ which occurred in the past, and to describe how it was dealt with. In contrast to this, the interviewer could think of a ‘forward scenario’ in which some future imaginary problem arises and ask the user how they would deal with this. (Hart, 1985)

Card games
Card games and card sorting can be used as a means of getting users to demonstrate their perception of certain types of problem. For example, the RE could write down the user’s problems as he or she sees them, one problem per card. The RE could then ask the user to sort the cards in order of priority. The RE could take the same cards and ask the user’s manager to sort them in order of priority. This may help in deciding which problem to tackle first. Card sorting could also be used as a means of identifying the relationships between different concepts. In this case the cards may contain some ‘high level’ concepts and some lower level concepts. The user is asked to sort the cards into piles such that there is one high level concept and the associated lower level concepts in the same pile. (Gammack and Young, 1985)

Ethnographic studies
Ethnography is a method derived from anthropology and is based on observing the behaviour of groups. Professional ethnographers are employed to observe users over a long period of time and to make detailed observations about work practices. Analysis of audio-visual recordings and results from field studies: “ reveal the delicate and complex web of interactional practices through which information is communicated and tasks accomplished.....even apparently individual tasks like reading, writing or typing into a computer are embedded in interactions with others and are designed in relation to another’s activities” (Luff et. al, 1993).

Their studies suggest that the traditional understanding of tasks as activities carried out by individuals is flawed and that all work activities involved some level of social interaction. Ethnography is not a requirements technique nor is it appropriate for a requirements engineer to carry out an ethnographic study, however, the results of ethnographic studies can be used to inform the requirements investigation, see Hughes et al., 1995.

1.7.3 Object-oriented approaches

Object-oriented analysis is a relatively young approach to requirements analysis. According to Coad and Yourdon, 1991, it is ‘based upon concepts that we learned in kindergarten: objects and attributes, wholes and parts, classes and members’. One of the advantages of a specification consisting of objects is that ‘ it can be agreed upon more readily by all (involved) in the development process.....’, Flynn, 1992.

The object-oriented approach has it's origins in object-oriented programming particularly in the language Smalltalk. The notion of object-oriented has proliferated throughout design and analysis. A recent publication of the Object Management Group (Hutt, 1994) describes twenty one object analysis and design methods ranging from the Booch method (Booch, 1991) to Coad and Yourdon's Object Oriented Analysis (1991) to Object Oriented SSADM! (Robinson and Berrisford, 1993) and the Z++ method (Lano and Haughton, 1992, 1993).

According to Davis, 1993, the primary motivation for object orientation is that as a system evolves, its functions tend to change, but its objects remain unchanged. Thus, a system built using object-oriented techniques may be inherently more maintainable than one built using more conventional functional approaches such as those described in the next section.

1.7.4 Structured Systems Analysis

Systems Analysis has the longest tradition of involvement in what is now called requirements engineering. Professional systems analysts often act as a 'go between' assisting designers in understanding user requirements and assisting users in understanding what is possible. Their task usually involves interviewing users and collecting and analysing information from various documents. They are responsible for understanding user requirements and for describing those requirements in a form that can be understood by designers.

Systems analysts often use structured systems analysis techniques. They use dataflow diagrams to represent the processes associated with the users tasks and entity relationship diagrams to describe the objects that are acted upon within those processes.
Data dictionaries are also used to store information about all data items defined in the dataflow diagrams, the analyst can use the data dictionary for checking the consistency and completeness of the dataflow diagrams and the entity relationship models.

These modelling techniques are useful in helping the analyst to communicate with designers because the entity relationship models can be used by the designer as a basis for the database design and the data flow diagrams can be used as a basis for program design. Data dictionaries can be used to generate data definitions as required by languages such as COBOL, FORTRAN and PASCAL.

This approach is often referred to a process-oriented or function-oriented. Examples of this approach can be found in:

- Structured Requirements Definition (SRD), Orr, 1980,
- Structured Analysis and Design Technique (SADT), Ross, 1977 and Marca and McGowan, 1988
- Structured Analysis and System Specification (SASS), DeMarco, 1979
- Modern Structured Analysis, McMenamin and Palmer, 1984, Ward and Mellor, 1985 and Yourdon, 1989
- Structured Systems Analysis and Design Method (SSADM), Downs et al. 1992

Davis, 1993, provides an interesting perspective on the historical development of this approach. His book also includes a review of the computer based tools which are available to support these methods.

Other methods use structured approaches within a wider context of analysis. For example CORE (Controlled Requirements Expression) Mullery, 1987. CORE undertakes the analysis by first identifying the viewpoints associated with the proposed system. A viewpoint is someone or something which is responsible for processing information. For each viewpoint a viewpoint model is developed which shows the inputs, processes and outputs using a notation similar to that of data flow diagrams. The single viewpoint models are then connected together, outputs from one model are connected to inputs to another model, to form a combined viewpoint model. This, however, is only one of a number of techniques used within CORE.

IBM's Joint Application Design (JAD), August, 1991, also uses structured analysis techniques. JAD consists of a series of facilitated workshops and uses many different techniques in it's quest for a team approach to system development. More details can be found in chapter four of this book.

1.7.5 Participatory Design

Participatory Design (PD) (Floyd et al., 1989, Greenbaum & Kyng, 1991) advocates a strong user involvement in system design, in which workers actively engage in designing the computer systems they will eventually use. (Carmel et. al., 1993). PD is often referred to as the 'Scandinavian approach' because many of it's proponents stem from the Scandinavian countries where the emphasis in the 1970s and 1980s was on workplace democratisation and empowering workers through participation in decision making processes.

Carmel et al. contrasts PD with the 'American approach': ' The software engineering approach that effectively serves as the basis for development in North America is based on fixed requirements, communication through documentation, and rules of work enforced by methods - functional foci which are de-emphasised or dismissed in the PD literature.' The PD foci is more social with emphasis on mutual learning, joint experiences, and workplace democratisation.

Greenbaum and Kyng (1991) put forward a set of design ideals which has guided their work in participatory design, the main points are listed below:

- "Computer systems that are created for the workplace need to be designed with *full participation* from the users.
- When computer systems are brought into a workplace, they should *enhance* workplace skills rather than degrade or rationalize them.
- Computer are *tools*, and need to be designed to be under the control of the people using them.

- Although computers are generally acquired to increase productivity, they also need to be looked at as a means to increase the *quality* of the results.
- The design process is a political one and includes *conflicts* at almost every step of the way. Managers who order the system may be at odds with the workers who are going to use it. Different groups of users will need different things from the system, and the system designers often represent their own interests.
- And finally, the design process highlights the issue of how computers are used in the context of work organisation. We see this as a question of focusing on how computers are used, which we call the *use situation*, as a fundamental starting point for the design process."

Floyd et. al., 1989, point to two main themes associated with PD principles. These are mutual reciprocal learning and design by doing.

Mutual reciprocal learning refers to the situation in which users and designers learn from each other about work practices and technical possibilities through joint experiences. For example designers may become 'apprentices' to users and learn by actually attempting to do the users job. Conversely users may learn about new technology through hands on training provided by the designer.

Design by doing refers to interactive experimentation, for example, by developing mock-ups of designs using cardboard boxes or other physical objects, games involving the future situation or drawings of the future workplace. By employing this approach PD practitioners have proved very innovative in engaging users in creative design.

PD practitioners (of the Scandinavian school) do not tend to be great advocates of prescriptive methods, however they do use a number of techniques, for example:

- Future Workshops are a means for focusing designers and users on visions of a future workplace
- Cooperative Prototyping in which users are actively involved in the design of prototypes as well as their evaluation.
- Design Mock-ups for generating ideas and to get feedback from users (Ehn and Kyng, 1991)
- Future games, an exercise in design by playing, linked to future workshops (Ehn and Kyng, 1991)

Details of these techniques can be found in chapter five.

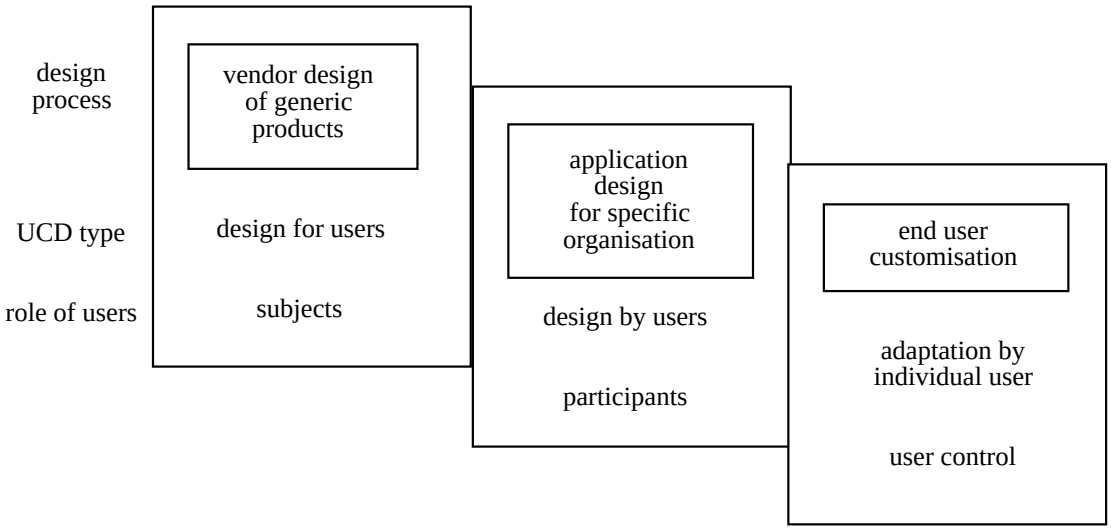
Enid Mumford who is also a PD practitioner has adopted a more 'American approach' in that step by step guides on how to use the PD method ETHICS are available (Mumford, 1986, 1989). ETHICS places emphasis on job satisfaction, good job design and good organisational design. Users participate in the requirements process by analysing their problems at work, completing job satisfaction questionnaires and setting objectives for efficiency, effectiveness and job satisfaction.

Details of Mumford's approach can be found in chapter three.

1.7.6 Human Factors and HCI

In common with Participative approaches, Human Factors and HCI (Human Computer Interaction) practitioners are concerned with user centred system design. User Centred Design (UCD) places the user at the centre of the design process, and includes techniques and procedures for designing usable systems. Woodson, 1981, suggests that UCD is concerned with "design from the human-out" and that the designer should "make the design fit the user", as opposed to "making the user fit the design". UCD includes techniques which cover not only the requirements stage of a project but the whole life of a product from development through to every day usage, (Rubin, 1994). Gould and Lewis (1985) advocate three key principles for UCD these are: (I) an early focus on users and tasks, (ii) empirical measurement of product usage and (iii) iterative design whereby a product is designed, modified, and tested repeatedly.

Eason, 1992, suggests that different forms of User Centred Design are needed depending on the purpose of the product. When generic products are being developed for a large number of customers it is difficult to get users to participate in the design process, Eason suggests that in this case, users are subjects, who can be observed or questionned, take part in user reviews or usability evaluations. When an application is being designed for a specific organisation then users can participate in the design process, probably as part of a design team, a number of alternative design team structures are discussed in chapter two. The third type of UCD is when users have purchased a generic product and wish to adapt it to meet their individual needs, in this case the user is in control of the changes.



Three Forms of User-Centred Design
Figure 6 after Eason, 1992

UCD incorporates a whole range of user-centred techniques, including techniques for undertaking cost benefit analyses from an organisation and user perspective,

requirements capture and analysis (Macaulay, ..., HUFIT.....)task analysis (Johnson....), dialogue specification (Monk....) and usability evaluation (Rubin, 1994, Monk et al. ....). Details of selected techniques can be found in chapters three, four and five.

1.7.7 Soft Systems

Many research traditions recognise that all human actions take place within wider contexts or situations Suchman (1989), Winograd and Flores (1986), Preece (1994) and from Checkland (1981) systems theory.

The Soft Systems Methodology (SSM) of Checkland (1990) and Wilson (1990) is a general problem solving technique that was developed as an engineering approach to management problems (Dobbin and Bustard, 1994). The basic elements of the technique were establish in the early 1970s and refined over several years by a process of 'action research'. This involved applying the technique to real problem situations, assessing its effectiveness and making adjustments accordingly. It is not a precisely defined method 'but rather a collection of concepts that analysts may use in whatever way they find effective', (Dobbin and Bustard, 1994).

SSm is important because it has a number of features which are not explicitly addressed by more traditional analysis methods, Dobbin and Bustard, 1994 presents a summary of these features (reproduced with permission):

"Treatment of the Problem Situation
SSM is concerned with analysing the entire problem situation, by considering the wider system environment as well as the system under investigation. SSM does not examine a problem but the situation in which there is perceived to be a problem.

Emphasis on Behaviour
SSM focuses on identifying the purpose (or purposes) of a system and the activities necessary to achieve those purposes. It explicitly avoids a consideration of system structure initially.

Emphasis on Change
SSM is a methodology which is based on the idea of bringing about change in a problem situation. The proposed system model is compared to the actual system in order to determine the necessary changes.

Multiple Perspectives
The essence of SSM is its analysis of the problem situation from a number of different perspectives or viewpoints. Systems usually serve a number of different purposes and an acknowledgement of the multiple viewpoints provides SSM with a mechanism for identifying and resolving conflicts.

Goal-driven

SSM is a goal-driven approach; in other words, it focuses on a desirable system and how to reach it, rather than starting with the current situation and considering how to improve it.

Emphasis on Control and Monitoring
SSM explicitly recognises the importance of control in any system, by requiring the presence of a monitoring and control activity."

The basic SSM is discussed in chapter three of this book.

1.7.8 Quality

The manufacture of mass produced goods during the industrial revolution created the necessity for control of the production process, in particular in the car and engineering industries. Growing customer demands, intensive international competition, products which are parts or components of systems which are becoming increasingly complex and the considerable cost savings, are all reasons put forward by Wallmuller (1994) for paying attention to quality management within software engineering.

Philip Crosby, an internationally recognised expert in the field of quality assurance, argues that not only top management but that every individual is responsible for quality (Crosby, 1985). Zultner, 1993, confirms that a fundamental component of total quality management is for each person or group to improve their work on a regular basis.

Zultner identifies a strong link between requirements and quality, he advocates the use of QFD (Quality Function Deployment) as a means of maximising customer satisfaction from the engineering process. The focus of QFD is on " preventing dissatisfaction by having a deeper understanding of stated requirements and implied customers' needs, and then deploying these expectations downstream in order to design value into the system." (Zultner, 1993). Three types of requirements are identified (Kano et al. 1984):

- Normal requirements: these are what we typically get by just asking customers what they want.
- Expected requirements : these are often so basic the customer may not think to mention them - until we fail to deliver them. Their presence in the system meets expectations, but does not satisfy customers. Their absence however is very dissatisfying.
- Exciting requirements : The presence of these provoke the customer reaction "wow. it even does this..." features can succeed in satisfying customers so well that that they boast about their software. (Zultner, 1993)

Zultner describes how to use QFD and other quality control tools to improve the software development process. The basic QFD procedure is described in chapter four of this book.

1.7.9 Formal Computer Science

A key problem within requirements is that of specifying requirements which are complete, unambiguous and consistent. Formal discrete mathematics-based specification

methods have a role to play in this, for example, Lutz, 1993 suggests the use of formal specification techniques alongside natural language specifications. Lutz' paper reports on the problems of software requirements errors in safety critical embedded systems where lack of precision and incomplete requirements have led to component failures.

Producing formal descriptions forces the requirements engineer to think more carefully about the nature of the system being defined and how exactly it will operate, Bustard and Lundy, 1995. Formal languages are typically difficult to learn and often are applied only to some critical subset of the specification. The reader is referred to Gehani and McGettrick, 1986, for further details.

summary.....link

1.8 Rationale for this Book

summary of what has been read
Thus instead of presenting the reader with a pre-defined process for Requirements Engineering, the author adopts the view that the process will be dependent upon the nature of the project, and that what the Requirements Engineer needs is a portfolio of techniques.

It can be seen from the nine approaches described above that the portfolio could include potentially a large number of techniques. It is not the author's intention to provide descriptions of all possible techniques. Indeed many of the techniques mentioned in this book have received adequate coverage elsewhere. Thus this book will seek to introduce the reader to a number of specific and useful techniques which have not previously been included within Computer Science literature.

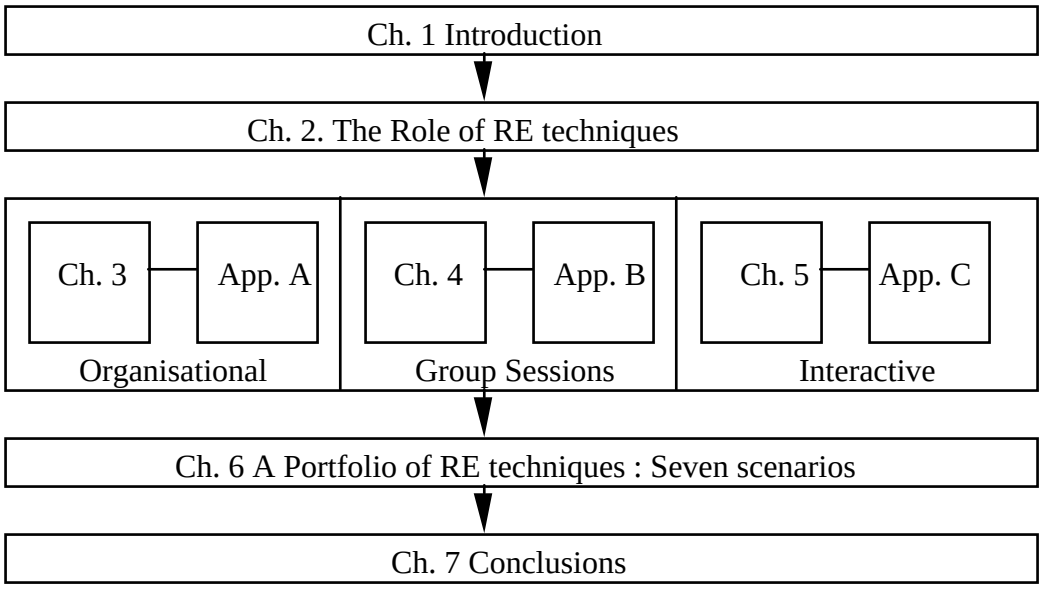


figure 6 The structure of this book

Chapter two presents a detailed discussion on the role of requirements engineering techniques and attempts to establish what techniques are needed. (expand)

Chapters three, four and five focus on specific problems within requirements engineering. Each chapter begins with an illustrative problem situation. There then follows a number of specific techniques which could be used to help the requirements engineer to deal with the problem. Also attached to each chapter is a user guide to one particular technique. It is intended that each guide contains sufficient detail for the reader to attempt to use the technique on their own project.

In chapter three the key problem is that there is a failure to realise that the goals of a system are defined within the total context of an organisation and its political and social environment and not just in relation to technology. This problem is illustrated through an analysis of the failure of a computer aided despatch system at the London Ambulance Service. The techniques described in this chapter encourage the requirements engineer to consider the social, political and organisational issues as part of the requirements investigation. The techniques described belong to Soft Systems, ETHICS and Eason's User Centred System Design. Appendix A contains a 'user guide' to Eason's "Cost benefit assessment of the organisational impact of a technical system proposal."

In chapter four the key problem is that different interest groups do not communicate effectively with each other, each seeking to exert power and influence over the other. The illustrative problem situation is a study of the development of a Computer Aided Learning system in which two interest groups fail to reach agreement about the

requirements despite the fact that two prototypes of the system were developed. The techniques described in this chapter encourage the requirements engineer to consider the importance of facilitation in the RE process. The techniques described are JAD, QFD and CRC. Appendix B contains a ‘user guide’ to CRC (Cooperative Requirements Capture) stage one.

In chapter five the key problem is that designers do not fully understand the work of users. The illustrative problem situation is.....
The techniques described in this chapter encourage the requirements engineer to interact with the users. The techniques described include future workshops, design mock-ups, cooperative prototyping and Cooperative Evaluation. Appendix C contains a ‘user guide’ to Cooperative Evaluation.

Chapter six presents a discussion of how the reader might develop a ‘portfolio’ of requirements techniques. Seven different scenarios are explained. For each scenario the customer supplier relation is explained, a typical process model is presented and techniques are suggested for the portfolio.

Conclusions - chapter seven

1.9 Summary

link ch 1 to ch 2