10 ESTABLISHING THE REQUIREMENTS

Malcolm Eva

INTRODUCTION

> When I actually meet users ... I find that they certainly have needs, but that these do not appear in an organised form at all. The needs come out in a rush, a mixture of complaints, design decisions, interface descriptions, current situations, and from time to time specific human–machine interface requirements. It is sometimes possible to isolate chunks of this as definite functions. In short, there seems to be a gap between theory and practice. Theory says that on the one hand we'll find people who state what they want, and on the other, people who make things to please the first bunch of people. Instead, all the people and tasks and documents seem to be muddled up together.
>
> Ian Alexander (2004)

Most analysts involved in defining user requirements will probably recognise Alexander's description. Requirements, apparently regarded by business users as the uncomplicated area of a new systems development, are actually the most problematic aspect, and yet, the time allocated is often far less than for the other phases. Requirements Engineering (RE) is the Information Systems (IS) industry's response to the problem of requirements for new systems not being met, or even not being offered clearly in the first place. Tight timescales and tight budgets – both the result of constraints on the business – place pressures on the development team to deliver a product. However, without sufficient time to understand and define the requirements properly, the product that is delivered on time may not provide the solution that the business thought had been requested. Applying the process of requirements engineering should help to rectify these problems.

As many organisations are adopting an Agile mode of software development, there are new pressures on the requirements phase, where an IT systems forms part of the solution. Agile works to a completely different philosophy from the linear waterfall-based methods, and the approach chosen must be tailored accordingly. While the stages set out in the RE framework are still applicable, the level of detail defined about the requirements may be varied. Agile development adopts the principle of 'just in time' requirements. So although high-level requirements are required right from the start, the level of detail for those requirements evolves just in time. For example, at the beginning of a project, the requirements document may contain some high-level process models and a set of general business and technical requirements, supplemented by a use case diagram showing the scope of the proposed IT system. Once initial requirements have been prioritised, outline use case descriptions can be developed for the high-priority requirements and then more detailed requirements regarding the solution can be elaborated later during development. In essence, the level of detail defined for the

requirements, and the approach to the documentation, should be tailored according to the philosophy and approach to be adopted for the solution. Agile does not mean 'just do it', and an equal rigour must be enforced when working in this environment.

A similar approach to requirements definition may be adopted when the solution is to be an off-the-shelf software package. It may not be worthwhile documenting the requirements in exhaustive detail as the package will impose some constraints. However, it will be important to produce a requirements document that has enough information to evaluate the package and ensure that there is sufficient 'fit' with the requirements.

THE PROBLEMS WITH REQUIREMENTS

Studies carried out into IS project failures over the last 30 years tell a common story. The problems highlighted included:

- a large proportion of errors (over 80 per cent) are introduced at the requirements analysis stage;
- very few faults, (fewer than 10 per cent) are introduced at the development stage – developers are programming things right, but frequently not programming the right things;
- most of the project time is allocated to the development and testing phases of the project;
- less than 12 per cent of the project time is allocated to the requirements analysis phase;
- there is poor alignment of the developed system to business strategy and objectives;
- there is poor requirements management.

These findings are particularly significant because the cost of correcting errors in requirements increases dramatically if they are discovered as the work progresses further through the development lifecycle. And yet, the studies quoted above identify requirements analysis as the most error-prone stage of the development lifecycle.

Walia and Carver (2009) identified over 90 causes of requirements errors in a meta-study they carried out on 149 papers. Typical problems with requirements that these causes have brought about include:

- lack of relevance to the objectives of the project;
- lack of clarity in the wording;
- ambiguity;
- duplication and overlap across requirements;
- conflicts between requirements;

- requirements expressed in such a way that it is difficult to assess whether or not they have been achieved;
- requirements that assume a solution, rather than stating what is to be delivered by the system;
- uncertainty amongst business users about what they need from the new system;
- business users failing to identify all requirements;
- inconsistent levels of detail;
- business users and analysts taking certain knowledge for granted and failing to ensure that there is a common understanding.

The first point often results from a lack of terms of reference for the project. Clearly defined terms of reference are essential to ensure that everyone involved in defining the requirements understands the objectives, scope and constraints within which the project is to be carried out. A useful mnemonic for the terms of reference is OSCAR. This stands for:

- **Objectives** – both business and project objectives should be defined.
- **Scope** – the aspects to be covered, typically defined by specifying the activities and deliverables of the project, as well as the functional areas to be included. It is equally important to specify those areas NOT to be covered, in order to guard against scope creep during the exercise.
- **Constraints** – the budget, timescale and standards to which the project must adhere.
- **Authority** – the business authority for the project, thus ensuring that there is an ultimate arbiter to handle any conflicts between business users and their requirements. The authority is also responsible for accepting the deliverables from the project.
- **Resources** – the people and equipment available to the project.

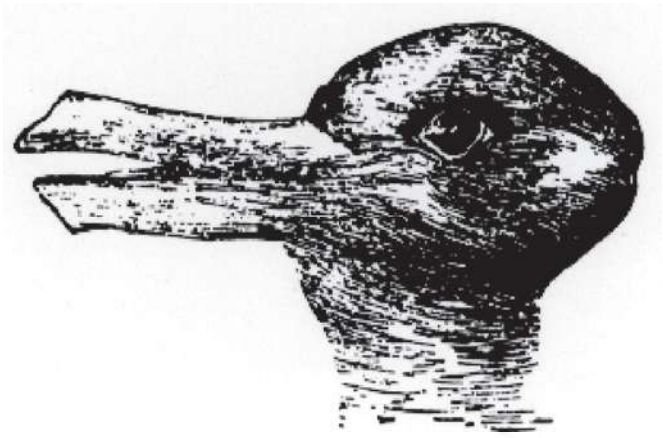
Requirements are often overlooked because they are omitted by the business user. This is usually an oversight rather than mischief, and results from a belief that the need is so self-evident they take it for granted that the analyst will understand that. The difficulty associated with taking some knowledge for granted is by no means trivial, and in a world of new business practices, business processes and new technology, by no means uncommon. The business analyst is the person who must help the business staff to visualise precisely what they need the new system to perform, and then to articulate it.

Another source of difficulties for the business analyst is recognising the different stakeholder viewpoints. Depending upon their role in the organisation, one person might see the business system as supporting the product, while another may think in terms of finding a 'marketing solution'; a third may see the system as being customer-focused. All three are describing their perceptions of the same system, but each sits in his or her own silo, viewing the business from that one viewpoint. It is up to the business analyst

to draw the threads together to view the system as a whole, and to meet all three perspectives as well as possible. Chapter 6 discusses the CATWOE technique and its use in analysing stakeholder perspectives.

Figure 10.1 illustrates the problem of perspectives with an illustration taken from Wittgenstein's Philosophical Investigations.

Figure 10.1 The duckrabbit



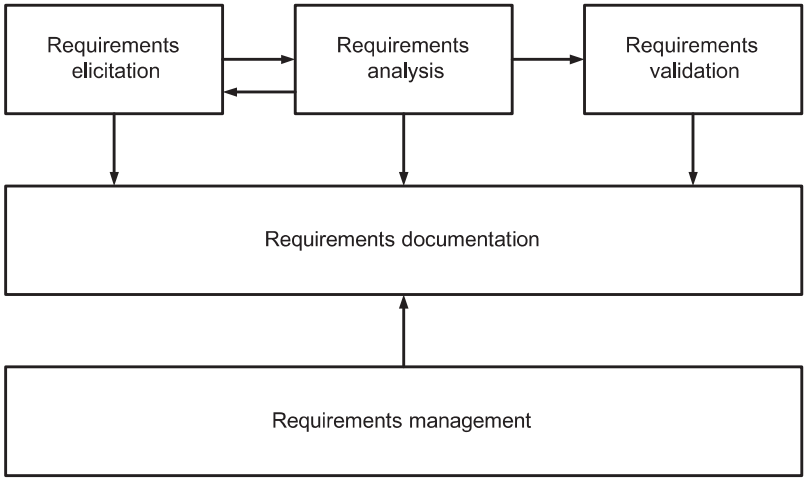
If the creature is perceived as looking to the left, it is clearly seen as a duck; if it is seen as looking towards the right, however, it is clearly a rabbit. What it is not, though, is both a duck and a rabbit. Wittgenstein named this ambiguous animal a duckrabbit. The business analyst will encounter many duckrabbits during his or her career: a sales director who sources products to satisfy customer demands sees the company as a sales organisation, while a finance director who has invested heavily in an in-house manufacturing capability will see a manufacturing organisation. They are describing the same company, but one sees a duck, the other a rabbit. Is the company there to provide satisfaction to its customers (a duck) or to sell its manufactured products (a rabbit)? The analyst or requirements engineer needs to understand all of these valid perspectives, or 'world views' (as described in Chapter 6) to be sure of capturing more than a one-sided view. Recognising the existence of a duckrabbit, or multiple world views, helps to anticipate scoping issues and potential requirements conflicts before they arise.

Another problem that business analysts face is an apparent inability on the part of the business users to articulate clearly what it is they wish the system to do for them. Some users, perhaps due to uncertainty, may even be reluctant to state their requirements. Very often they are deterred from doing so because the nature of the requirement is not susceptible to a straightforward statement. These issues may be due, at least in part, to the problem of tacit knowledge; we shall explore some of these issues in the section on 'Requirements elicitation'.

A FRAMEWORK FOR REQUIREMENTS ENGINEERING

In the same way that 'software engineering' suggests a more structured and scientific approach to the development of software than is implied by the older term 'programming', so requirements engineering encapsulates a more disciplined and rigorous approach to requirements definition. The business analyst must understand and document requirements carefully in order to avoid the mistakes and oversights identified above. To achieve this rigour, the business analyst should follow a roadmap that guides them through the key steps required to develop a well-defined requirements document. Figure 10.2 illustrates the requirements engineering framework, which provides such a road map.

Figure 10.2 Requirements engineering framework



Requirements elicitation is concerned with drawing out information and requirements from the business stakeholders. This is done using the techniques described in Chapter 5, which provides a toolkit of techniques to undertake requirements elicitation effectively. When we elicit requirements from the users they are often explained with a lack of clarity and completeness. Before the requirements are formally entered into a requirements document, the business analyst needs to subject the initial requirements list to careful scrutiny in order to ensure that they are well-formed. This is done during the requirements analysis stage which focuses on examining and organising the elicited requirements. This may involve identifying requirements that overlap, are in conflict with other requirements or are duplicates. It may also involve separating out composite descriptions into individual requirements. Once the set of unique requirements have been identified, it is important to check them against the business objectives; if a requirement is not supporting the achievement of a business objective then it is questionable that it is required. In requirements analysis, the business analyst has a gatekeeper role, ensuring only those requirements that pass the scrutiny will be entered in the document. This activity highlights any areas requiring further elicitation and, in turn, the additional information gleaned will be subject to analysis. This iterative process

continues until the analyst is content that the requirements have been defined to the desired level of detail and quality.

Requirements validation involves the external stakeholders reviewing the requirements in order to agree and sign off the requirements document. The requirements catalogue is one part of the full requirements document, which also contains models of the business processes and IT system. This document should be analysed and checked by the business analyst for completeness and consistency. Once the document is considered to be complete, it must be reviewed by business representatives and confirmed to be a true statement of the required system; at this point the document is said to be 'signed off' or 'baselined'. During this stage the reviewers examine the requirements and question whether they are well defined, clear and complete. The review will be similar to that undertaken in the analysis stage, but this time it is the business stakeholders who will carry out the checks rather than the analysts.

These three aspects – requirements elicitation, requirements analysis and requirements validation – are described in further detail below.

There are two other aspects that complete the requirements engineering work: requirements documentation is concerned with the development of a well-organised requirements document; requirements management covers the activities needed to manage any changes to the requirements. Changes occur throughout the business change and solution development lifecycles, and procedures need to be in place to deal with them. Whatever the reason, the business analyst must record, analyse and action the changes and ensure that every requirement is fully traceable from the initial recording to sign-off during user acceptance testing. Chapter 11 discusses the development and structure of the requirements document, the issue of traceability and the use of software tools to support requirements management. The approaches to developing the models of the IT system are described in Chapter 12.

ACTORS

An actor is an individual or group who is fulfilling a particular role. There are some roles we would expect to be represented during the requirements work and they represent two broad stakeholder groups:

- the business representatives;
- the project team.

The business representatives

The business is represented by the project sponsor, the subject matter expert (SME) and the business users.

The project sponsor represents the business in ensuring that business objectives are met. The sponsor has the following responsibilities:

- to agree the project initiation document that approves the requirements engineering study;

- to deliver the specific and agreed business benefits predicted in the business case;
- to make funds and other resources available for the project;
- to accept the deliverables at the end of the project;
- to approve and sign off the requirements document as a true statement of the business needs;
- to rule on any conflicting requirements where the business analyst cannot negotiate agreement;
- to confirm that the benefits in the business case have been realised as promised.

The role of the SME is to give business advice regarding the requirements. Particular situations when this may be relevant are:

- the requirements cover more than one business function;
- the requirements relate to a redesigned business process;
- the organisation wishes to introduce a new product or process that the company does not yet fully understand;
- the organisation wishes to adopt the latest industry best practice.

The SME brings a breadth of understanding to RE, and should have experience and knowledge of industry best practice. Their level of knowledge of the business domain should help analysts distinguish between what the business and the project *need* and what a particularly forceful individual user *wants*. The SME may be an internal expert or may be an external consultant, brought in for the duration of the project. While an external SME can bring in fresh views and insights from industry, drawing on best practice as used elsewhere, there are some risks associated with their use:

- They may not understand well how this particular company works, and its specific culture. This may make their preferred approach to the solution inappropriate.
- They may be unaware of political undercurrents in the organisation that can affect the project's success or otherwise.
- They may be regarded as an outsider and so be resented by internal workers who feel they too have suitable expertise.
- There may be no knowledge transfer to internal stakeholders; the skills and knowledge gained on the project leave the organisation when the consultant SME leaves.

The business users are the individuals or groups who will need to apply the new business processes and use the new IT system. They comprise the group for whom the solution is designed. They are required to describe current procedures and documentation, highlight any difficulties they experience with current processes and identify new requirements for the system. They should be able to help the business

analyst to define the requirements in detail, by providing specific, clear information. They will be able to assist with the definition of non-functional requirements that apply to their tasks, although some specific aspects may need management involvement. For example, decisions about archiving information and the length of retention of data are likely to require the involvement of middle or senior managers.

The business user community is likely to perform several roles, each of which needs to be considered when analysing the requirements. One role from the business user community may be the 'customer' role. This will occur for any IT systems that are accessible by the end customer. For example, where an online system is to be developed and implemented. In this case care needs to be taken to ensure that the requirements are gathered accurately and appropriate techniques, such as the use of focus groups, are used to acquire the relevant information.

The project team

The project team comprises the project manager, the business analyst and the developers or technical architects. There will also be communication with representatives from the project support office and with the software testers.

The project manager is mindful of the need to meet the business requirements and satisfy the business imperatives that drive the project. The project manager will report to the project sponsor and will be concerned to:

- break the project down into identifiable and measurable pieces of work, each with its deliverable;
- allocate the pieces of work to competent people to perform;
- schedule the tasks with their start and end times, recognising dependencies between tasks;
- monitor the progress of the various tasks and be alerted of any likely slippage;
- take any corrective action should there be slippage, or risk of non-completion of a task for any reason;
- ensure that the project is completed on time and within the agreed budget.

The business analysts will be responsible for carrying out the requirements engineering work. Their key objective is to ensure that the requirements are well-formed, well-documented, complete and aligned with the business objectives. Working closely with the business staff, they gather and analyse the requirements, and are responsible to the sponsor for the quality of the requirements document.

Many organisations assign the project manager role to the business analyst on a project, but this can be problematic as the two roles call for different aptitudes, priorities and skills. There may be conflicting priorities and interests to be reconciled, which will be problematic if one individual is assigned both of these roles. The project manager is most concerned with meeting cost and time targets, while the business analyst's primary consideration is satisfying the quality criteria.

The developer will be able to check the technical feasibility of some of the requirements and help the analyst appreciate the implications of some of the requests. The developer will be able to produce prototypes of the requirements, to help the business user visualise more clearly what they have requested. This will help confirm the analysts' understanding of the requirements, as well as encouraging the users to identify new features they would like to see.

REQUIREMENTS ELICITATION

In the early days of systems analysis, this activity was often known as 'requirements capture'. The word 'capture' implies the existence of discrete entities called 'requirements' that were readily available to be spotted and caught. In those days most IT projects were concerned with developing IT systems that would perform the tasks carried out by people, mostly clerical staff. To that extent, each task performed by a person could be regarded as a requirement, or set of requirements, to be delivered by the system, so requirements capture was not an unreasonable term to use.

Nowadays, the rationale for developing or enhancing IT systems includes a need to help the organisation gain a competitive advantage, to support new or re-engineered business processes, or to install an enabling infrastructure. The more straightforward approach of using the current procedures as a basis has declined and there is a strong likelihood that the business users will not be at all clear about what they need the system to provide. The term 'capture' is less suited to the reality of today's business world, and has been superseded by the term 'elicitation'.

Whereas earlier approaches placed the onus on the business user to identify the requirements, requirements elicitation is a proactive approach to uncovering the requirements. It involves drawing out information from the users, helping them to visualise the possibilities and articulate their requirements. The requirements emerge as a result of the interaction between analyst and user with much proactive elicitation on the part of the analyst.

When devising the strategy for elicitation for any given project, one consideration to take into account is the type of knowledge the stakeholders are using.

Tacit knowledge

When developing a new system, the business users will pass on to us their explicit knowledge; their knowledge of procedures and data that is at the front of their minds, and which they can easily articulate. By tacit knowledge, we mean those other aspects of the work that a stakeholder is unable to articulate or explain. The term derives from the work of Michael Polanyi (2009) whose thesis is succinctly expressed in the maxim 'We can know more than we can tell.' In terms of understanding requirements, there are a number of elements to tacit knowledge that we must be aware of and recognise when we encounter them.

Some common elements to this unspoken knowledge that cause problems and misunderstandings are:

Skills

Explaining how to carry out actions, using words alone is extremely difficult. For example, consider how you might convey the correct sequence of actions that would be necessary to turn right at a roundabout on a dual carriageway. This would include every touch or release of the pedals, in the correct sequence, every check of the mirror, every gear change and every turn of the wheel, with the degree of turn. To build an automated vehicle that could travel along roads without a driver would need all of this information to be specified in detail. In practice, an experienced human driver would not be able explain this accurately during an interview, largely because they perform the task without having to rationalise each step; their bodies and limbs 'know' what to do without the intervention of conscious thought. If we needed to document this process, we would have to select a requirements elicitation approach that would help to uncover information about these automatic actions. Protocol analysis is a powerful way to explore these unconscious skills and gather the information to document them. This technique involves the expert performing the task in discrete steps, talking through each step as they make it, and repeating this two or three times. This is an approach often used where skills are taught, for example, when children are taught to tie shoelaces, learner drivers shown how to drive a car, and apprentices taught to use machines.

Taken-for-granted information

Even experienced and expert business users may fail to mention information or clarify terminology and the analyst may not realise that further questioning is required. The gap in our understanding may not emerge until user acceptance testing or even after implementation and this may be costly and complex to correct. This issue has been identified as a cause of many systems failures. This has also been termed 'not-worth-mentioning' information. This highlights that it is not through malice or intention that the user fails to reveal some aspect of the procedure or documentation, simply that to them it is so obvious that they take the analyst's familiarity for granted. The fact that the analyst then fails to ask a question on that aspect simply confirms to them that to talk about it is unnecessary. It is often only at user acceptance testing, and sometimes even later during deployment, that this gap becomes apparent.

Front-story/back-story

This issue concerns a tendency to frame a description of current working practices, or a workplace, in order to give a more positive view than is actually the case. The business user may do this in order to avoid reflecting badly on the staff or the organisation. An analogy could be the swan gliding gracefully across the smooth surface of a lake (front-story) while below the surface its feet are paddling frantically (back-story) to sustain the gentle momentum. This problem may occur if the analysts are perceived to represent management and as a result, may be given the favourable front-story version of how the business or department works. It is clearly important then that we build good working relationships with the business users and encourage them to trust us so that we are able to obtain details of the back-story of the reality of the business operation.

We should be most suspicious of a front story when we are told an operation runs very smoothly without any problems, and if there are problems they are always easily dealt with. Also, we might be concerned if we are told that the customers of that process are always very happy with the results. While this could be the case it is always sensible to probe a little deeper and ensure that there is quantifiable data to support the assertion.

The saying, 'If it sounds too good to be true – it probably is' is often found to be the case in organisations. The analyst must avoid seeming sceptical about what they have been told – after all, success lies in building a rapport with key stakeholders and earning their trust. Nonetheless, the true story – the 'back-story' – will in all probability give a different picture. Work shadowing, observation of the working environment, rich pictures, spaghetti diagrams and process modelling are all techniques that help to clarify what really happens.

Conceptualising requirements

If the study is required to examine a new business system and there is a lack of expertise and knowledge in our organisation, it is very difficult for the business stakeholders to imagine what they require the solution to offer. We cannot ask them to demonstrate any existing processes or procedures and yet it is important that we draw from them what they need the solution to provide. While the information about the business context, for example, the business strategy or the legal constraints, will provide a degree of understanding, it will be extremely difficult to produce a more detailed definition of the requirements. A good approach is to help the business users visualise for themselves how the business processes could work and what IS support they will require. Providing a visual representation will enable the business users to consider and articulate their requirements more clearly. Approaches that would help with this include exploring possibilities in a workshop, using business scenarios and story-telling workshops, or enlisting the developers to build some initial prototypes to show what the IT system might offer and how it might look.

'Your finger, you fool'

This is based upon an apocryphal story and tells about a European explorer who landed in the seventeenth century on an island and asked a native inhabitant the name of a prominent mountain. He pointed at the landmark he was asking about, but the islanders did not recognise the gesture of pointing to distant objects; the inhabitant assumed that what he was being asked to identify was the outstretched finger. This illustrates the difficulty of an outside party assuming that there is a common language and that the common norms of communication apply. In such situations, cultural and language differences may create many possibilities for confusion, so we should consider whether an extended investigation might be required. An ethnographic study can be an extremely useful technique in such situations although it will require an extended period of time and a great deal of effort. However, without a detailed investigation approach the analysts will not be sure that they have sufficient understanding and knowledge to be able to define the business requirements. Without this, the scope for misrepresentation of the situation can grow considerably.

Intuitive understanding

This is usually born of considerable experience. Decision-makers such as those working in medical diagnosis or geological surveys, are often thought to follow a logical, linear path of enquiry while making their decisions. In reality though, as individuals acquire decision-making skills and technical knowledge, the linear approach is often abandoned in favour of intuitive pattern recognition. Ask specialists why they made a particular judgement and you may be told about the logical steps, but there will often be a point where the logic ends and intuition takes over; this is where knowledge has been applied at a tacit level rather than explicitly. Protocol analysis aided by drawing a decision tree can help with understanding decisions, as can ethnographic studies.

With the exception of 'front-story/back-story', which is more of a reluctance to tell than an inability to articulate, these issues all occur because of the application of an individual's tacit knowledge. However, there are situations where an organisation possesses tacit knowledge and it is important for this to be recognised and understood. Examples of organisational tacit knowledge include:

- Norms of behaviour and communication – these evolve over time in every organisation. Any new process or system that threatens to conflict with these norms may face resistance.
- Organisational culture – the culture of an organisation can be expressed through the behaviour of the management and staff. The analyst needs to consider what the behaviour says about the culture of the organisation and ensure that this is taken into consideration when considering any business changes.
- Organisation stories – there will always be a shared history of events that have happened in the past. Some of these histories may relate to projects that failed or succeeded in some more spectacular fashion, and about which stories and in-jokes may have grown over the years. There could be important lessons to learn from these histories, but they are only known by word of mouth, so an analyst from outside may not be aware of this knowledge and make a faux pas as a result.
- Formal and informal networks – these are discrete groups of workers who may be related by task, or by department, by geographical location or some other factor. They have their own sets of experience, norms and practices, and as these are distinct from other groups within the organisation they are not reflected in the organisation as a whole. A network is likely to have its own body of tacit information, which the members understand well and are not accustomed to sharing openly. If a systems development project involves cross-functional requirements, or is company-wide, then the understanding of the networks is an important part of the elicitation process.

Table 10.1 shows levels of both tacit and explicit knowledge.

Table 10.1 Levels of tacit and explicit knowledge

	Tacit	**Explicit**
Individual	Skills, values, taken-for-granted knowledge, intuitiveness	Task definitions, job descriptions, targets, volumes and frequencies
Corporate	Norms, back-story, culture, networks, organisation history	Procedures, style guides, processes, knowledge sharing repositories, manuals, company reports

Tacit knowledge must be made more explicit. To do this we need to choose techniques to assist business users to articulate their tacit knowledge. Once they have articulated the tacit knowledge, we need to make this knowledge explicit by documenting it and disseminating it to the other members of the project team.

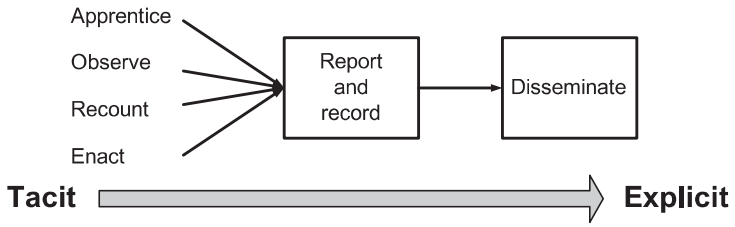
REQUIREMENTS ELICITATION TECHNIQUES

There are a number of best practice techniques available to the analyst to help elicit tacit knowledge and requirements from all levels of the organisation. Those used most often are discussed in Chapter 5.

The process for eliciting tacit knowledge shown in Figure 10.3 identifies approaches for uncovering tacit knowledge. These approaches include:

- apprentice – shadowing, protocol analysis;
- observe – observation, perhaps using the STROBE approach (see glossary), shadowing;
- recount – story-telling, scenario;
- enact – prototype, scenario role-play.

Figure 10.3 Tacit to explicit knowledge



It is important that business analysts have a toolkit of techniques so that they can tailor their approach when eliciting requirements. Table 10.2 matches some of the most popular elicitation techniques with the knowledge types that a business analyst is likely to have to handle. This mapping of techniques to knowledge types provides a good indication of where certain techniques can be particularly useful and where they can still be useful but to a lesser extent.

BUILDING THE REQUIREMENTS LIST

As we elicit the requirements from our various business stakeholders, we need to document them. This is best done in two distinct passes; building the initial requirements list, and later, developing an organised requirements catalogue. The development of the requirements catalogue is described in Chapter 11.

Table 10.2 Techniques and knowledge types (after Maiden and Rugg, 1995, and Eva, 2001)

Technique	Explicit knowledge	Tacit knowledge	Taken for granted	Front/back story	Skills	Future requirements
Interviewing	✓✓	✓	✓	✓	✓	✓
Shadowing	✓✓	✓✓	✓✓	✓✓	✓✓	✓
Workshops	✓✓	✓✓	✓✓	✓	✓	✓✓
Prototyping	✓✓	✓✓	✓✓	✓	✓✓	✓✓
Scenario analysis	✓✓	✓✓	✓✓	✓	✓	✓✓
Protocol analysis	✓✓	✓✓	✓✓	✓	✓✓	✓

The requirements list is quite simply what it says, a list containing every requirement that has been stated or elicited. The list tends to be an informal document and can be presented using three columns, as shown in Table 10.3 below, which represents a preliminary requirements list from an inventory project: the comments are added during the first pass at analysis, when each requirement is scrutinised for acceptability. This may be done at any time during elicitation while the list is being compiled. It can be useful to ask another analyst, who was not present at the elicitation session, to scrutinise the requirements as a fresh perspective can often identify clarifying questions that may have been missed during the conversation with the stakeholder.

Table 10.3 Example requirements list

Requirement	Source	Comments
1. We need an ERP System	I. Morris	
2. We need to print off a list of expected deliveries	I. Morris	
3. We need to cross-check orders against deliveries	F. Drake	
4. We need to supersede an item	J. Keen	

(Continued)

Table 10.3 (Continued)

Requirement	Source	Comments
5. The system must record the fulfilment of a purchase order on delivery	J. Keen	
6. The system must automatically re-calculate re-order quantity	J. Keen	
7. The stores controller must have the facility to adjust re-order quantity	B. Armstrong	
8. We need to create and delete contractors	K. Wynn	
9. We need to return damaged items	F. Drake	
10. We need more accurate stock levels	F. Drake	

The requirements list is begun following an initial interview or workshop and is developed further as more requirements are identified. The requirements identified at this stage will not be well-formed and the level and scope will vary considerably; some may be detailed and specific whilst others will be defined at an overview level only or may represent several potential requirements. The requirements list helps to ensure that everything that is raised is documented and the source identified. This is vital to embed traceability in the requirements.

REQUIREMENTS ANALYSIS

Once a set of requirements has been elicited and entered onto the requirements list, the next step in the requirements engineering framework can begin. This involves analysing the individual requirements.

Requirements analysis is concerned with ensuring that all of the requirements identified during requirements elicitation have been developed into clear, organised, well-documented requirements. The separation of requirements elicitation from requirements analysis is one of the strengths of requirements engineering. The analysts are encouraged during elicitation to draw out information, uncover tacit knowledge and identify requirements; during analysis, the business analysts examine the results of this work in detail in order to develop the requirement documentation that will take the project forward. Requirements analysis requires a high degree of logical thought, organisation and rigour if the documentation is to be of the required standard. In this instance, the analyst takes the role of a gatekeeper in order to ensure the quality of the requirements. Only those requirements that are clearly stated, unambiguous, atomic, feasible, aligned with the project objectives, not in conflict with any other requirements and not overlapping with or duplicating other requirements, will be documented fully for implementation.

In order to confirm that requirements are in scope, and also that the set of requirements is complete, it is helpful to draw some models that incorporate all the requirements. A use case diagram may be used to define the scope of the system and the features to be delivered; a business process model may be used to illustrate how the work is to be conducted and the flow of the process; a data model may be used to clarify the data to be held and the business rules to be implemented. If the requirement does not fit naturally into any of these three models it is probably out of scope. By cross-referring the models to each other and to the requirements it should be possible to see where there are omissions. For example, the analyst should check that there is a function for creating, modifying, deleting and accessing the data shown on the data model. If any of these functions are missing there may be a missing requirement, although it is possible that the function is not in the scope of the system. Such omissions should at the least prompt questions. The other reason for including the models is that diagrams can be much easier to makes sense of than a catalogue of several hundred entries.

As this work is carried out, many additional questions will be raised and you may need to investigate the requirements further using requirements elicitation techniques. Thus an iterative cycle of requirements elicitation and analysis develops.

Requirements analysis includes the following tasks:

- Categorisation of the requirements – as a first pass at the analysis, the requirements should be categorised into specific requirement types. There are four major types of requirements: general (business), technical, functional and non-functional requirements. These types are described in more detail in Chapter 11. Further sub-groupings may be by business area or function, or possibly by use case. Grouping requirements this way helps the analyst to examine the related requirements as a coherent group. This helps check for completeness of requirements, and makes it easier for a business stakeholder to validate a particular set of requirements.
- Drawing a set of models that reflect the requirements that have been elicited. These models provide a check that the set is complete, coherent and that no requirements have been omitted. The modelling techniques are described in Chapter 12.
- Applying a series of filters in order to ensure that the requirements are well defined.

Requirements filters

The following filters should be used to examine the requirements and build a well-formed, clearly documented requirements set.

- **Checking for overlapping or duplicate requirements**: once requirements that cover a similar area are grouped together, it is much easier to find the duplicate or overlapping requirements. Where there is duplication, the requirements should be merged; where there are overlapping requirements they should either be merged or separated into distinct requirements.
- **Unravelling multiple requirements**: some requirements may have been listed that cover a number of different aspects rather than stating one requirement.

It is important that these grouped requirements are split into individual, atomic requirements. An example of a grouped requirement is: 'the reservations clerk must be able to record, amend or cancel a booking'. In fact, there are three requirements here potentially with their own priority and acceptance criteria. Each needs to be documented.

- **Confirming relevance of the requirement**: all requirements should be aligned to the business and project objectives. They should also address a root cause of a problem rather than just a symptom of a problem. If a requirement statement does not support these areas it is likely that this is not actually a business need.
- **Feasibility evaluation**: all requirements should be evaluated to see if they would be feasible. There are three aspects to feasibility – technical, business and financial.
  - Technical feasibility is concerned with the availability of technology to fulfil a requirement. Does the technology exist to satisfy the requirement, and is it available to the organisation? If we already have the technology in place, can it cope with the expected volumes and frequency of use? Is it sufficiently robust and secure?
  - Business feasibility concerns the likely level of acceptance of the requirement by the business. Does it align with business objectives and strategy? Does it match the organisational culture? Does it contribute to the critical success factors?
  - Financial feasibility involves considering how expensive it would be to meet the requirement and consider if this is justified. Is the requirement within the budget? If not, would the extra funding be available? What would be the return on investing in this feature? Imagine a client asking an architect to build a large swimming pool into a new house design without increasing the cost. This would not be possible, but business stakeholders have been known to ask for ambitious additional features, without appreciating the cost implications. Sometimes a manager asks for information to help with a decision, which at first seems reasonable but turns out to involve building a very sophisticated Business Intelligence System (BIS) at a cost far exceeding the original budget.
- **Removing conflicts**: some requirements may contradict or conflict with other requirements such that only one may be implemented. When such a conflict is identified, the business analyst is responsible for helping to negotiate a resolution. Sometimes confirming relevance will resolve a conflict if only one of the requirements is directly aligned with the business objectives. However, if the sources of the requirements are adamant, and there is no room for compromise or discussion, then the decision will need to be passed to the sponsor or even escalated to the project board. Conflicting requirements sometimes come from different business perspectives, sometimes from organisational politics and sometimes from interpersonal tensions. The analyst will often need to tread carefully to avoid alienating either party. The approach used to analyse stakeholder perspectives described in Chapter 6 can be very helpful in understanding any differences of view at an early stage. Failing to do this is likely to result in conflicting requirements which are difficult to resolve because the opinions are so entrenched.
- **Checking for solutions**: on examination, a requirement may not be something the business needs to happen or have addressed, but may be a pre-determined

solution. Business stakeholders often express a requirement in terms of the solution, rather than in terms of what the business needs. For example, a requirement may dictate that a specific software package should be used rather than express the business requirements to be satisfied.

- **Confirm standard of quality**: all requirements should meet the quality criteria and should be:
  - **Clear** – the requirement must be expressed in clear language, avoiding vague adjectives and adverbs and using precise verbs and nouns. It is also important to avoid terms such as 'and' 'but', 'except', 'until' as each of these suggest there is more than one requirement in the statement.
  - **Concise** – the requirement must be described concisely. This is not the place to specify data items, triggers or conditions. They will come further down the line. Ideally the requirement should be expressed in one simple sentence.
  - **Consistent** – the requirement must not contradict other requirements. All requirements must be worded using the same format. This reduces ambiguity and confusion, and makes validation much more straightforward.
  - **Relevant** – the requirement must be within the scope of the project. The analyst and SME must distinguish between a business need for this project, and an individual user's 'want', which may add no value overall.
  - **Unambiguous** – the description of the requirement must not contain any ambiguity. This can arise because of the problems of terminology and jargon, or may result from poor grammar. Common sources of confusion involve the use of synonyms (two words used to mean the same thing) and homonyms (the same word used to mean different things). This is especially relevant when a project involves more than one department or division. If two people who read a requirement have a different mental picture of what is being asked for, then it is ambiguous. It is important that developers and business users compare their understanding of the requirements to be sure that there is no misinterpretation. Use of a glossary of terms for the project helps to guard against such misunderstandings.
  - **Correct** – the requirement statement must describe something that is actually required to meet the objectives.
  - **Testable** – the requirement should be described such that the solution may be tested to confirm that the requirement has been met. The statement must be worded so that a simple yes/no answer to the question 'Has it been delivered as intended?' makes sense. Later, non-functional performance measures, e.g. response time, transaction time, and security restrictions must be built into the fuller description in the requirements catalogue.
  - **Traceable** – information about the requirement must enable the traceability of the requirement. For example, the 'source' of each requirement should be recorded. Full traceability is only possible once the requirements are documented. This is discussed further in Chapter 11.

Examining the requirements in Table 10.3 in this way yields the following:

1. *We need an ERP system*. This is not a functional requirement for an inventory system, it is a strategic decision about the nature of the solution. The requirements phase is concerned with finding out the individual business requirements that such a system (if it were to be implemented) must satisfy.
2. *We need to print off a list of expected deliveries*. This is both vague (what deliveries? How soon before they are expected? Just the fact of a delivery, or cross-referencing with the orders? Listing the items to be delivered?) and a solution. The actual requirement is to produce an alert of the deliveries due within the specific time period (today, this week, this afternoon). The business requirement is not to print the notice. That might well be a good solution, but to put that into the requirement immediately disqualifies other, possibly better, solutions.
3. *We need to cross-check orders against deliveries*. This is unclear – what orders? Deliveries in or out? Cross-check for what? What is actually the business need?
4. *We need to supersede an item*. There is obviously a business function to supersede one item of inventory with another, but it is not clear what the requirement is from this project. This reads like a high-level requirement, and needs a lot of work to understand what information support is needed, which will be translated into a set of lower-level functional requirements.
5. *The system must record the fulfilment of a purchase order on delivery of the goods*. This sounds a clear statement of a business need.
6. *The system must automatically re-calculate re-order quantity (RoQ).* While this appears to be a clear requirements statement, it sounds a technically complex operation, with a series of variables to calculate and does not specify at what point the RoQ takes place. We need to investigate why it must be done automatically and when it should be done, and estimate how long it will take to design and implement, in case it proves not to be feasible in our timescales.
7. *The stores controller must have the facility to adjust re-order quantity*. This sounds as though it might conflict with item 6. Clearly they want to alter RoQ, but who is responsible – is this a political issue? Return 6 and 7 for negotiation.
8. *We need to create and delete contractors*. This is a statement of two separate requirements.
9. *We need to return damaged items*. Not clear what IS support they are asking for with this function. What exactly is the requirement?
10. *We need more accurate stock levels.* This is both imprecise and a complaint about service rather than a requirement. We need to investigate the problem of inaccurate levels. It may be a process or people issue.

While this may seem an onerous process to go through, especially when we have a long list of requirements, it is a necessary step and helps ensure that we deliver a set of requirements that are clear, and that all stakeholders agree are needed, to satisfy the project objectives. As the requirements are likely to be elicited over a period of time, this analysis can be done in parallel as a fresh list of requirements is entered, rather than waiting until the list is complete and then beginning.

We will complete the third column in the requirements list with these comments, or a précis of each.

There are several potential outcomes for each requirement from this exercise:

- to accept the requirement as it stands and document it in more detail in the requirements catalogue;
- to re-word the requirement to remove jargon and ambiguity;
- to merge duplicated/overlapping requirements and re-word them;
- to split composite requirements into their individual entries;
- to take unclear, ambiguous or conflicting requirements back to the business users for clarification.

Requirements that are in conflict, are unrealistic, or are out of alignment with the business objectives, still need to be recorded in the requirements catalogue in order to ensure that an audit trail is kept of all requirements raised and any subsequent action taken. This means that they should be considered for implementation and a business decision should be made. The structure and format of the requirements catalogue is described in Chapter 11.

Once all of the requirements have been grouped and analysed, it is useful to carry out a final, further check. The analyst needs to examine each requirement on the list to check that they are well formed and SMART (specific, measurable, achievable, relevant and time framed).

REQUIREMENTS VALIDATION

Once the analysts have completed the analysis activity and have deemed the requirements document to be complete and correct, the business and project representatives need to confirm that the document provides an accurate statement of the requirements. A review group is formed that will be responsible for checking the requirements document and confirming its suitability. Once the reviewers have been identified, the requirements document should be issued for review.

The review group must include representatives from the key stakeholder groups and different representatives will review different aspects of the requirements. The reviewers should include the following representatives:

- The business sponsor should review the document to ensure that the requirements are all in alignment with the business objectives and do not address areas that are outside the scope of the project.
- The business owners of the individual requirements, or their representatives, should review the requirements to ensure that they express the business needs clearly and correctly, without ambiguity. It is the business representatives' responsibility – and their last opportunity – to be satisfied with the requirements before accepting them.
- The subject matter expert should review the requirements to ensure that they reflect correct business practice.

- The developers should review the requirements to ensure that they are technically feasible.
- The testers should review the requirements to ensure that they are testable.
- Project office representatives should ensure that the requirements are compliant with business standards and policies, and that correct quality review procedures have been followed.

The group may meet to discuss the requirements document in a formal review meeting, may have a virtual meeting whereby comments are submitted electronically via a shared forum, or possibly, individual responses via email may be provided. A formal meeting or shared forum provides an additional aspect to a document review as the reviewers are aware of the comments from the entire group, which provides an opportunity to consider other perspectives. The use of individual emails tends to provide a more limited review approach as there is a lack of shared review and comment. However, it is still a valuable review approach and is preferable to no review. Whichever approach is taken to the review, there are two important roles to be filled: there should be a chairperson, who is responsible for controlling the review, and a business analyst, who is responsible for providing information about the requirements document or possibly presenting it to the review meeting.

A common problem regarding validation is that the stakeholders have been too busy prior to the meeting to study it and carry out the review. The analyst and project manager must make it clear that this is a task that has to be performed, and must stress the impact of not getting the requirements right at this stage. If a key stakeholder does come to the meeting with that apology, their part of the review must be re-scheduled so that they see the importance of doing the prior work. This is easier to say than enforce, but if mistakes do get through to the finished document even after all the close analysis, there will be greater expense and difficulty at a later point.

Another problem of requirements validation concerns the size of the document that is to be reviewed. Where the requirements document is large it is a good idea to review it in sections. Where the requirements have been well-organised, for example, grouped by business area or use case, it is possible to conduct a set of shorter review meetings. Only relevant stakeholders (the business owners of the particular area under discussion) need be invited to review each section of the document, which will help to save time for everyone.

There are three possible outcomes to a review:

1. The requirements document is confirmed as a satisfactory statement of the business requirements. Once the document has been agreed, it is signed off, or baselined.
2. The requirements document requires some amendment and, once these have been completed, can be signed off by the review chairperson. This is typically the business sponsor.
3. The requirements document needs significant rework and should be reviewed again once this rework has been carried out.

Once the document has been signed off, any subsequent changes should be subject to formal change management. This is described in Chapter 11.

The outcome of the review should be an agreement that the entire requirements document is complete, consistent, conformant and a true reflection of what the business requires to be delivered.

AGILE APPROACH TO REQUIREMENTS

The RE approach described above ensures that the project has understood and incorporated all business needs and user requirements before the solution is designed for implementation. It provides an audit trail for the requirements document and is meant to give all stakeholders confidence in the integrity of the development. The traceability features allow all changes to the requirements, for whatever reason, to be incorporated in the document with no loss of quality, and the reasons for those changes recorded in case of query or conflict further down the line.

However, the RE approach can be time-consuming and make demands on resources; many organisations need to produce new systems in a shorter timespan than this allows so have adopted an Agile approach to development.

The key differences between defining requirements when using an incremental or waterfall approach and the Agile philosophy, is in the level and type of documentation that is produced. If RE is employed in an incremental or waterfall-style project, the aim is to produce a baselined set of good, complete, relevant well-formed requirements before a solution is specified and designed; this ensures that the development is driven by business requirements, and not by the technology. Agile, on the other hand, follows a practice of evolving the requirements during development so much of the detail is not specified in advance. However, it is good practice to have a definition of the requirements at a level that enables the Agile development to commence. This requires a set of requirements to be selected for development within a defined timebox. Despite the less detailed nature of the requirements definitions, it is still important to use the RE framework to ensure that the requirements are clear and meet the quality criteria described earlier. The details of the requirements evolve during the development process and, if desired, the detailed definition of the requirements is completed once the development has been completed.

The advantage of this is that it reduces the maintenance overhead of managing a detailed requirements document while the software is being developed. It also removes the redundant work that may be involved in specifying requirements early on, only to see them change, and change again, before implementation. The drawbacks are that it could lead to scope creep as focusing on individual requirements might lead the developers to overlook the big picture of the business objectives. Also, unless there is a strong project structure in place, there is a danger that traceability is compromised, and in future years, maintainability could present a problem where the documentation is insufficiently detailed. Agile software developments are best suited to a dynamic environment where requirements are unclear at the outset, and where there is a short timescale for initial delivery.

Levels of Agile requirements

An Agile project will normally begin with an intensive workshop, possibly a hothouse workshop as described in Chapter 5. The business participants at a hothouse will usually

be at a senior level, which should help ensure that the development is aligned with the business strategy, and the functionality identified will contribute to that strategy rather than simply being a 'nice idea'. The outcome of the workshop should be the scope, a set of high-level requirements, a priority for implementing the requirements and some way of representing which actors need to interact with the system. While there is no prescribed format for this output, a commonly used notation is the use case diagram. These diagrams are illustrated in Chapter 12. Each use case represents a piece of functionality that is needed to meet an actor's needs, and may contain a number of requirements to satisfy them; these requirements need not be defined absolutely until just before they are due to be built. An alternative approach is to employ the 'user stories' technique, which identify the features required by an actor to be fulfilled by a system. The stories themselves may be at two levels: 'epic' and individual story. 'Epic' describes a story that, because of size, cannot be implemented in a single development period and so must be broken down into smaller, lower-level stories. The lower-level user stories are inputs to the development process, which is carried out in a series of 'timeboxes' or 'sprints' which typically may last between 15 days and 4 weeks, depending upon the scale of the development. The developer, using a prototyping approach and technology, sits with the business user and business analyst, to elaborate on the requirements that will deliver the user story and meet its defined goal.

The user story will generally be framed to answer the questions, Who? What? Why? and be expressed in the format: 'As a {user role} I want {feature} so that I can {reason}.'

'As a warehouse clerk I want to record the quantity of an item that has been picked so that I can know how much stock is left in the bin.'

This format has the following benefits: **who** – helps us be sure we build a usable product, as the actor concerned will be with the developer as they construct the prototype; **what** – defines the functionality that is being provided; **why** – identifies rationale for the function; this may be in terms of enabling a dependent activity or adding value by the performance of the action. The **why** feature enables decisions about the priority of the stories. Once the story has been agreed, it is often documented on an index card (paper or electronic). Business rules, pre-conditions and constraints are then elicited and added to the story, so that they can form part of the prototype.

There is discussion among Agile proponents as to whether user stories, use cases or a mixture best represent the requirements. There is not a single correct answer, just as there is not a single best approach to Agile developments. Each team and each organisation that uses this approach will have its own standards and preferences. What is more important is that the process is tightly controlled, so that the successive sprints are planned and always in accordance with the architecture and the business objectives.

It is important to recognise that Agile offers an alternative philosophy rather than a rigid methodology. It has evolved over several years, beginning with early approaches to development such as RAD, and in the tradition of systems thinking and Lean. It is likely to continue evolving as the world and the technology keep advancing. If a system is likely to have a short life before needing to be replaced, a 'light touch' RE approach followed by Agile software development is an obvious approach to use; if the system is likely to form the backbone of the company's operations for a substantial time or is highly complex, a linear development lifecycle preceded by a more rigorous RE approach would be beneficial and would make future maintenance and enhancements much less risky.

SUMMARY

Requirements engineering is the approach by which we ensure that the process of understanding and documenting the business requirements is rigorous and ensures the traceability of each requirement. This chapter has considered the RE stages of elicitation, analysis (which feeds back into the elicitation), and validation; the supporting stages of documentation and management are considered in Chapter 11. All of these stages contribute to the production of a rigorous, complete requirements document. The core of this document is a repository of individual requirements that is developed and managed and provides the basis for deciding upon business and software changes. Where organisations place insufficient emphasis on defining requirements it is to the detriment of the implemented solutions and can leave business stakeholders unable to do their jobs effectively. Requirements engineering is an approach by which we can deliver solutions that truly meet business needs.

REFERENCES

Polanyi, M. (2009) *The Tacit Dimension*, Re-issued edn. University of Chicago Press, Chicago, IL.

Walia, G. S. and Carver, J. C. (2009) 'A systematic literature review to identify and classify software requirement errors', *Information and Software Technology*, 51. 1087–1109.

Maiden, N. A. M. and Rugg, G. (1996) ACRE: selecting methods for requirements acquisition, *Software Engineering Journal*. 183–192.

Eva, M. (2001) Requirements acquisition for rapid applications development. *Information & Management*, 39. 101.

FURTHER READING

Alexander, I. and Beus-Dukic, L. (2009) *Discovering Requirements. How to Specify Products and Services*. John Wiley and Sons Ltd, Chichester.

Alexander, I. and Stevens, R. (2002) *Writing Better Requirements*. Addison-Wesley, Harlow.

Cadle, J., Paul, D. and Turner, P. (2014) *Business Analysis Techniques*: *99 essential tools for success,* 2nd edn. BCS, Swindon.

Gause, D. and Weinberg, G. (1990) *Exploring Requirements, Quality before Design*. Dorset House, New York, NY.

Kulak, D. and Guinney, E. (2003) *Use Cases, Requirements in Context*. Addison-Wesley, Harlow.

Robertson, S. and Robertson, J. (2012) *Mastering the Requirements Process*, 3rd edn. Addison Wesley, Harlow.