Chapter five

5.2 What are requirements?

A requirement is a statement about an intended product that specifies what it should do or how it should perform. One of the aims of the requirements activity is to make the requirements as specific, unambiguous, and clear as possible.

5.2.1 Different kinds of requirements

Two different kinds of requirements have traditionally been identified:

1. ***functional requirements,*** which say what the system should do, ***For example functional requirement*** word processor may be that it should support a variety of formatting styles This requirement might then be decomposed into more specific requirements detailing the kind of formatting required such as formatting by paragraph, by character, and by document, down to a very specific level such as that character formatting must include 20 typefaces, each with bold, italic, and standard options

2. ***Non-functional requirements,*** which say what constraints there are on the system and its development. ***For example A non-functional requirement*** for a word processor might be that it must be able to run on a variety of platforms such as PCs, Macs and Unix machines. Another might be that it must be able to function on a computer with 64 MB RAM. A different kind of non-functional requirement would be that it must be delivered in six months' time.

5.3 Data gathering

So how do we go about determining requirements? Data gathering is an important part of the requirements activity and also of evaluation.

The purpose of data gathering is to collect sufficient, relevant, and appropriate data so that a set of stable requirements can be produced. Even if a set of initial requirements exists, data gathering will be required to expand, clarify, and confirm those initial requirements. There is essentially a small number of basic techniques for data gathering:

1) *Questionnaires:* Most of us are familiar with questionnaires. They are a series I of questions designed to elicit specific information from us. The questions may require different kinds of answers: some require a simple YES\NO, others ask us to choose from a set of pre-supplied answers, and others ask for a longer response or comment.

2) *Interviews.* involve asking someone a set of questions. Often interviews are face-to-face, but they don't have to be.

3) *Focus groups and workshops*. Interviews tend to be one on one, and elicit only one person's perspective. As an alternative or as corroboration, it can be very revealing to get a group of stakeholders together to discuss issues and requirements.

4) *Naturalistic observation.* observation provides a richer view. Observation involves spending some time with the stakeholders as they go about their day-to-day tasks, observing work as it happens, in its natural setting. A member of the design team shadows a stakeholder, making notes, asking questions (but not too many), and observing what is being done in the natural context of the activity.

5) *Studying documentation.* Procedures and rules are often written down in manuals and these are a good source of data about the steps involved in an activity and only source. Other documentation that might be studied includes diaries or job logs that are written by the stakeholders during the course of their work.

5.4 Data interpretation and analysis

Once the first data-gathering session has been conducted, interpretation and analysis can begin.

The aim of the interpretation is to begin structuring and recording descriptions of requirements. Using a template such as the one suggested in Volere (Figure 7.5) highlights the kinds of information you should be looking for and guides the data interpretation and analysis.

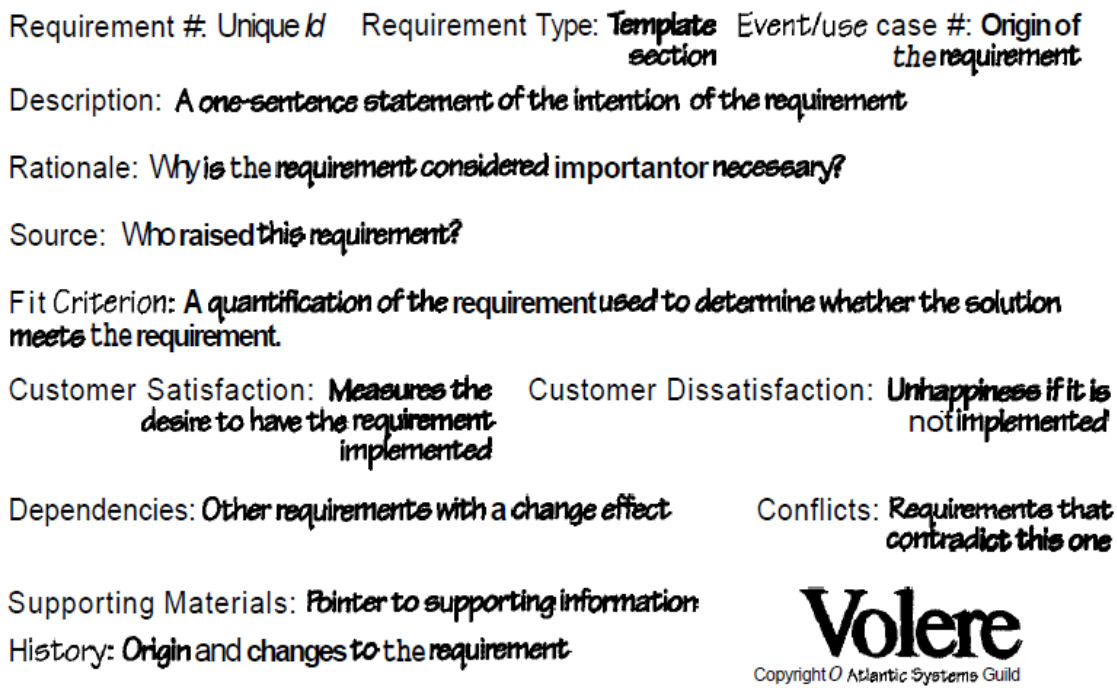

Figure 7.5 The Volere shell for requirements.

Chapter six

6.1 Prototyping

It is often said that users can't tell you what they want, but when they see something and get to use it, they soon know what they don't want. Having collected information about work practices and views about what a system should and shouldn't do, we then need to try out our ideas by building prototypes and iterating through several versions. And the more iterations, the better the final product will be. (هذه الفقرة للتوضيح وليس للحفظ)

What is a prototype

So a prototype is a limited representation of a design that allows users to interact with it and to explore its suitability. The term prototype, you may imagine something like a scale model of a building or a bridge. But In fact, a prototype can be anything from a paper-based storyboard through to a complex piece of software, and from a cardboard mockup to a molded or pressed piece of metal. A prototype allows stakeholders to interact with an envisioned product, to gain some experience of using it in a realistic setting, and to explore imagined uses.

1. **Low-fidelity prototyping:** is one that does not look very much like the final product.

For example, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal.

2. **High-fidelity prototyping:** uses materials that you would expect to be in the final product and produces a prototype that looks much more like the final thing..

For example, a prototype of a software system developed in Visual Basic is higher fidelity than a paper-based mockup; a molded piece of plastic with a dummy keyboard is a higher-fidelity prototype of the "Palm Pilot" than the lump of wood.

6.2 Construction: from design to implementation

When the design has been around the iteration cycle enough times to feel confident that it fits requirements, everything that has been learned through the iterated steps of prototyping and evaluation must be integrated to produce the final product.

Below discusses two different development philosophies.

- One approach, called evolutionary prototyping, involves evolving a prototype into the final product.
- alternative approach, called throwaway prototyping, uses the prototypes as stepping stones towards the final design. In this case, the prototypes are thrown away and the final product is built from scratch.

Chapter seven

7.1 What, why, and when to evaluate

Users want systems that are easy to learn and to use as well as effective, efficient, safe, and satisfying. Being entertaining, attractive, and challenging, etc. is also essential for some products. So, knowing what to evaluate, why it is important, and when to evaluate are key skills for interaction designers.

Evaluating designs

Evaluation is the process of determining the usability and acceptability of the product or design that is measured in terms of a variety of criteria (Evaluation is needed to check that users can use the product and like it) including the number of errors users make using it, how appealing it is, how well it matches the requirements,

Agencies such as National Institute of Standards and Technology (NIST) in the USA, the international Standards Organization (ISO) and the British Standards Institute (BSI) set standards by which products produced by others are evaluated.

7.1.1 Type of evaluate

The product being developed may be a brand-new product or an upgrade of an existing product.

1. If the product is new,
    - Designers often support this process by developing mockups of the potential product that are used to elicit reactions from potential users.
    - new products do not have previous versions and there may be nothing comparable on the market, so more radical changes are possible if evaluation results indicate a problem.
2. In the case of an upgrade,
    - there is limited scope for change and attention is focused on improving the overall product.
    - This type of design is well suited to usability engineering in which evaluations compare user performance and attitudes with those for previous versions

- ❖ Evaluations done during design to check that the product continues to meet users' needs are known as formative evaluations.

- ❖ Evaluations that are done to assess the success of a finished product, such as those to satisfy a sponsoring agency or to check that a standard is being upheld, are known as summative evaluation.

7.2 Evaluation paradigms and techniques

Before we describe the techniques used in evaluation studies, we shall start by proposing some key terms. We start with the much-used term user studies, defined by Abigail Sellen as follows: "user studies essentially involve looking at how people behave either in their natural environments, or in the laboratory, both with old technologies and with new ones." Any kind of evaluation, whether it is a user study or not, is guided by a set of beliefs. These beliefs and the practices (i.e., the methods or techniques) associated with them are known as an evaluation paradigm, which you

Each paradigm has particular methods and techniques associated with it.

An example of the relationship between a paradigm and the techniques that paradigm can be seen for usability testing,. The techniques associated with usability testing are: user testing in a controlled environment; observation and questionnaires and interviews.

In this part we identify four core evaluation paradigms

7.2.1 Evaluation paradigms

1. A "quick and dirty" evaluation is a common practice in which designers informally get feedback from users or consultants to confirm that their ideas are in line with users' needs and are liked. "Quick and dirty" evaluations can be done at any stage and the emphasis is on fast input rather than carefully documented findings.

2. Usability testing: was the dominant approach in the 1980s, and remains important, although, as you will see, field studies and heuristic evaluations have grown in prominence. Usability testing involves measuring typical users' performance on carefully prepared tasks that are typical of those for which the system was designed. Users' performance is generally measured in terms of number of errors and time to complete the task.

3. Field studies: The distinguishing feature of field studies is that they are done in natural settings with the aim of increasing understanding about what users do naturally and how technology impacts them.

4. Predictive evaluation: In predictive evaluations experts apply their knowledge of typical users, to predict usability problems. The key feature of predictive evaluation is that users need not be present, which makes the process quick, relatively inexpensive, and thus attractive to companies; but it has limitations.

7.2.2 Techniques

There are many evaluation techniques and they can be categorized in various ways,

The brief descriptions below offer an overview of each category, Be aware that some techniques are used in different ways in different evaluation paradigms.

1. Observing users: Observation techniques help to identify needs leading to new types of products and help to evaluate prototypes. Notes, audio, video, and interaction logs are well known ways of recording observations

2. Asking users: what they think of a product-whether it does what they want; whether they had problems using it; whether they want to use it again-is an obvious way of getting feedback. Interviews and questionnaires are the main techniques for doing this.

3. Asking experts: Guided by heuristics, experts step through tasks role-playing typical users and identify problems. Developers like this approach because it is usually relatively inexpensive and quick.

4. User testing: Measuring user performance to compare two or more designs has been the bedrock of usability testing. these tests are usually in controlled settings and involve typical users performing typical well-defined tasks to Data collected and the time taken to complete a task, the number of errors made, Descriptive statistical measures such as means and standard deviations are commonly used to report the results.

5. Modeling users' task performance: various attempts to model human-computer interaction so as to predict the efficiency and problems associated with different designs at an early stage without building prototypes. These techniques are successful for systems with limited functionality such as telephone systems.