1 Introduction

Summary

This chapter introduces the notion of requirements engineering and requirements engineering process improvement. To simplify the presentation, we have organised it as a set of questions and answers about requirements engineering. The questions which we answer are as follows:

Contents

The development of computer-based systems has been plagued with problems since the 1960s. Systems may be delivered late, over budget, they do not do what users really want and they are often never used to their full effectiveness by the people who have paid for them. There is rarely a single reason (or a single solution) for these problems but we know that a major contributory factor is problems with system and software requirements.

System requirements define what services the system should provide and set out constraints on the system's operation. Common problems which arise with system requirements are that:

1. the requirements do not reflect the real needs of the customer for the system.
2. requirements are inconsistent and/or incomplete.
3. it is expensive to make changes to requirements after they have been agreed.
4. there are misunderstandings between customers, those developing the system requirements and software engineers developing or maintaining the system.

We are convinced that the best way to reduce these problems is to improve the processes of discovering, understanding, negotiating, describing, validating and managing system requirements. We believe that the best way to do this is in a *gradual* way where you introduce new or improved procedures over a period of time. We do not suggest rapid process change. No-one knows enough about requirements engineering processes to assess if some radically different process will be effective. We think that the business risks of major process re-engineering are simply too high to be acceptable.

The guidelines in this book are therefore intended to support a gradual approach to process improvement. They are based on good requirements engineering practice. They range from very simple guidelines which might be thought of as common sense (but which are easy to overlook) through to suggestions for introducing new

methods and techniques to discover and analyse system requirements.

In this introductory chapter, we discuss requirements engineering and the requirements engineering process. To help you understand what the book is about, we have structured this chapter as a list of related questions and answers. This is rather like the increasingly widely used FAQ (frequently asked questions) lists which are used to introduce newcomers to information on the Internet.

We then go on, in Chapter 2, to discuss how to use the book to improve your requirements engineering process. We also introduce the notion of requirements engineering process maturity and suggest how you can assess your maturity level. Detailed good practice guidelines are presented in Chapters 3 to 10 and Chapters 11 to 13 give more detailed information on requirements engineering techniques.

1.1 How Will This Book Help Me?

The book is designed to help you improve your requirements engineering processes. We suggest that you should improve your process by identifying weaknesses then introducing new practices to address these weaknesses. We describe a set of good requirements engineering practices and suggest how you can implement these in your organisation.

To make the description of these good practices easy to read, we have organised them as a set of guidelines (presented in Chapters 3 to 10). Each guideline is presented in a standard way which will help you assess if it is appropriate for your organisation

1. The name of the guideline and a brief guideline description.
2. A set of bullet points setting out the benefits of implementing the guideline.
3. An implementation guide which provides some advice on how to implement the guideline. This is advice *not* instruction. You may have a better way to implement a guideline.

4 A brief assessment of the costs of introducing the guideline in to an organisation, the costs of applying the guideline in requirements engineering processes and problems which you may encounter when applying the guideline. Costs are given as low medium or high. Low-cost guidelines should involve less than 10 days of effort to introduce or apply, medium-cost involves 10–100 days of effort and high-cost involves more than 100 effort-days.

We suggest that you read this chapter and Chapter 2 to get an overall understanding of the requirements engineering process and our approach to process improvement. You should then dip into the guideline chapters to read the specific guidelines which are most relevant to your problems.

1.2 What are Requirements?

Requirements are defined during the early stages of a system development as a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system. Therefore a requirement might describe:

- a user-level facility (e.g. 'the word processor must include a spell checking and correction command'),
- a very general system property (e.g. 'the system must ensure that personal information is never made available without authorisation'),
- a specific constraint on the system (e.g. 'the sensor must be polled 10 times per second'),
- a constraint on the development of the system (e.g. 'the system must be developed using Ada').

Some people suggest that requirements should always be statements of *what* a system should do rather than a statement of *how* it should do it. This is an attractive idea but it is too simplistic in practice.

1 The readers of a document are often practical engineers who can relate to implementation descriptions much

better than they can understand very abstract problem statements. You have to write requirements which are understandable to the likely readers of the document.

2 In almost all cases, the system being specified is only one of several systems in an environment. To be compatible with its environment, and to conform to standards and with organisational concerns, you may have to specify implementation policies which constrain the options of the system designers.

Requirements therefore invariably contain a mixture of problem information, statements of system behaviour and properties and design and manufacturing constraints.

1.3 What is Requirements Engineering?

Requirements engineering is a relatively new term which has been invented to cover all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system. The use of the term 'engineering' implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, relevant, etc. The term 'requirements engineering' has come from a systems engineering background; if you are from a commercial systems background, you can think of requirements engineering as more or less the same thing as systems analysis.

Most of the guidelines in the book apply to system requirements engineering. That is, they apply to systems implemented as software, hardware, or by the people involved with the system. However, in some cases, the guidelines only really make sense when applied to software requirements. This is normally obvious from the guideline text.

1.4 What is a Requirements Document?

The requirements document is an official statement of the system requirements for customers, end-users and

software developers. Depending on the organisation, the requirements document may have different names such as the 'functional specification', 'the requirements definition', 'the software requirements specification (SRS)', 'the safety/reliability plan', etc. We use the term 'requirements document' in this book to cover all of these.

1.5 What is the Best Way to Write Requirements?

There is no best way to write requirements. It depends on normal organisational practice and the notations which are used by writers and readers of the requirements. Requirements may be stated in a language which reflects the background of the requirement source. If the source is an engineer, it may be written in engineering terms; if the source is a manager, it will be written in natural language. Most requirements are written as natural language sentences supplemented with diagrams and tables of detailed information. We make no assumptions in this book about how you write requirements nor are we trying to sell any particular notation for requirements specification.

1.6 How Detailed Should Requirements Be?

There is no simple answer to this question. Different organisations think of requirements in different ways and write them at different levels of detail. A requirement for controlling a motor may include details of the specific control signals to be used; a requirement for managing customer information might simply state the information to be provided. The system designer decides how to organise and present this information.

The level of detail that you need largely depends on the normal practice in your organisation, whether or not the requirements document will be the basis of a contract for software development and the type of system which is being developed. If you are developing a product which you both specify and implement, you may produce a

fairly general specification. You will add details to this as the system development proceeds. On the other hand, if you are contracting the system development to another company, you need a much more detailed specification which defines what they must implement.

In some organisations, requirements are first expressed as an informal high-level description which is then developed into a more detailed specification. The abstract requirements are the requirements of the stakeholders of the system and the detailed requirements are a system specification.

1 *Stakeholder requirements* (sometimes called user requirements). These are requirements which are written from the point-of-view of system stakeholders. They are not usually expressed in great detail. They are described in natural language, informal diagrams or using some notation which is appropriate to the problem being solved (e.g. mathematical equations for a control system).

2 *System requirements.* These are more detailed specifications of requirements which may be expressed as an abstract model of the system. This may be a mathematical model or may be based on graphical notations such as data-flow diagrams, object class hierarchies, etc. These models are always annotated with natural language descriptions.
It is generally true that system requirements are more detailed than stakeholder requirements. However, there are exceptions to this and, in both cases, requirements may be stated in a detailed or in an abstract way or anywhere in between.

1.7 What is the Difference Between Functional and Non-functional Requirements?

Very roughly, functional requirements describe what the system should do and non-functional requirements place constraints on how these functional requirements are implemented. A functional requirement might state that a system must provide some facility for authenticating the

identity of a system user; a non-functional requirement might state that the authentication process should be completed in four seconds or less.

However, it is not always as simple as this. The same functional authentication requirement could have a supplementary requirement which stated that a specific signature verification system (which was known to work in less than four seconds) should be used for authentication. This could be interpreted as either a functional requirement or a non-functional requirement according to the above definition. High-level non-functional requirements are often decomposed into functional system requirements.

1.8 What are System Stakeholders?

System stakeholders are people who will be affected by the system and who have a direct or indirect influence on the system requirements. This includes end-users of the system, managers and others involved in the organisational processes influenced by the system, engineers responsible for the system development and maintenance, customers of the organisation who will use the system to provide some services, external bodies such as regulators or certification authorities, etc.

For example, say an automated railway signalling system is to be developed. Possible stakeholders are:

- operators responsible for running the signalling system
- train crew
- railway managers
- passengers
- equipment installation and maintenance engineers
- safety certification authorities

We recommend that you draw up an explicit list of stakeholders at an early stage in your requirements engineering process.

1.9 Does System Size Make a Difference?

System size makes a tremendous difference. The problems of requirements engineering increase exponentially with the size of the system. Some military systems are gigantic; for such a case a requirements document is organised into several volumes with thousands of individual requirements. Almost everything (CASE tools, databases, organisational processes) starts to break down when it has to deal with the amount of information which is involved in something like this.

Requirements engineering for these huge systems often requires special-purpose notations, tools and techniques. These are too specialised to discuss here. The guidelines suggested in this book are intended to be helpful for all types of system requirements engineering but we know that some of them break down when applied to very large systems. In particular, techniques which involve cross-comparisons of requirements simply do not work when there are too many requirements. The information management problem is so great that the guidelines cannot be applied.

1.10 What is a Requirements Engineering Process?

A requirements engineering process is a structured set of activities which are followed to derive, validate and maintain a systems requirements document. A complete process description should include what activities are carried out, the structuring or schedule of these activities, who is responsible for each activity, the inputs and outputs to/from the activity and the tools used to support requirements engineering.

Very few organisations have an explicitly-defined and standardized requirements engineering process. They simply define the result of the process, namely the requirements document. The people involved in the process are responsible for deciding what to do and when

to do it, what information they need, what tools they should use, etc.

We are convinced that you will benefit by defining a requirements engineering process which is appropriate to your organisation. A good process description will provide guidance to the people involved and will reduce the probability that activities will be forgotten about or carried out in a perfunctory way.

1.11 How Do I Recognise Requirements Engineering Process Problems?

You can recognise requirements engineering process problems through either information about the process or information about the product. If you answer 'yes' to some or all of the following questions, there is almost certainly scope for requirements engineering process improvements.

1. Is your requirements engineering process usually over-budget and/or does it take longer than predicted?
2. Do people involved in requirements engineering complain that they do not have enough time or resources to do their job properly?
3. Are there complaints about the understandability or the completeness of the requirements documents which you produce?
4. Do system designers complain of rework resulting from requirements errors?
5. Do customers for your systems fail to use all of the system capabilities?
6. Is there a very high volume of change requests immediately after a system is delivered to customers?
7. Does it take a very long time to agree on system changes resulting from new requirements?

If you have answered 'no' to all these questions, you can stop reading now. You do not need the advice in this book.

1.12 Can You Suggest a Good Requirements Engineering Process?

We cannot do this without knowing about your organisation, your systems engineering and software development processes and the type of software which you develop. There are many possible ways to organise requirements engineering processes and they do not transfer well from one organisation to another. To define a good requirements engineering process, you need to involve people from your organisation who are actually involved in requirements engineering. You may have to get outside help from consultants, as they can take a more objective perspective than those involved in the process.

Most of the standards which have been developed for requirements engineering are concerned with process outputs such as the structure of requirements documents. Within general software engineering standards such as the US DoD standard 2167A, there is some mention of requirements engineering activities but nothing like a standard process description.

However, we would normally expect a good requirements engineering process to include the following activities.

1 *Requirements elicitation.* The system requirements are discovered through consultation with stakeholders, from system documents, domain knowledge and market studies.

2 *Requirements analysis and negotiation.* The requirements are analysed in detail, and should be some formal negotiation process involving different stakeholders to decide on which requirements are to be accepted.

3 *Requirements validation.* These should be a careful check of the requirements for consistency and completeness.

To support these activities, a requirements management process should be introduced to manage changes to the requirements.

1.13 Where Does ISO 9000 Fit In?

ISO 9000 is a standard for quality management processes and variants of this standard have been developed for software engineering. Many organisations have invested a lot of effort in improving their quality management and are now 'ISO 9000 certified'. This has led to significant improvements in quality management across many business sectors. However, ISO 9000 says nothing about requirements engineering. Therefore, there has been a natural tendency in business to focus on ISO 9000 activities at the expense of other processes even if these other processes are known to be a problem.

Approaches to process improvement which are embodied in standards such as ISO 9000 concentrate on process management and process improvement through the development and application of management procedures. They emphasise that you should define processes and put procedures in place to ensure that people follow these defined processes. They do not specify what these processes should be.

We take a complementary approach and discuss process activities rather than process management. Our view is that processes are improved by including particular good practices in your defined processes. All our suggestions are conformant with existing standards. They can be introduced without compromising any ISO 9000 certification which you have.

1.14 Where Can I Find Out More About Requirements Engineering?

There are a number of books on requirements engineering which may help you understand requirements engineering and how to improve your requirements engineering process. However, you should be warned that these books seem mostly to talk about problems rather than solutions.

1 *Software Requirements: Objects, Functions and States* (A. Davis, Prentice-Hall, 1993). This is probably the best

known book in this area. Its orientation is towards the use of structured methods for requirements engineering. It is quite long (500+ pages) and of most interest to practitioners who already have well-developed requirements engineering practices. It is very good on system modelling but weak on areas such as requirements validation and management.

2 *System Requirements Engineering* (P. Loucopoulos and V. Karakostas, McGraw-Hill, 1995). A short textbook which presents an overview of requirements engineering intended for students taking courses in this topic. It touches on lots of topics but provides little detail about any of them. It mostly describes problems and current research in this area rather than practical requirements engineering solutions.

3 *Exploring Requirements: Quality before Design* (D. C. Gause and G. M. Weinberg, Dorset House, 1989). This is an anecdotal book about requirements. It is easy to read and includes a lot of good practical advice. Its focus is on non-technical issues.

4 *Requirements and Specifications: A Lexicon of Software Practice, Principles and Prejudices* (M. Jackson, Addison Wesley, 1995). An interesting selection of short pieces on requirements and specifications. Readable and wise but not intended as a process improvement guide. It covers both technical and non-technical issues.

5 *Requirements Engineering: Frameworks for Understanding* (R. J. Wieringa, Wiley, 1996). A book for students which is mostly concerned with structured methods for modelling information systems. It covers modelling techniques such as data-flow diagrams and entity-relation modelling. It does not cover more general issues of requirements engineering and management.

6 *System and Software Requirements Engineering* (R. H. Thayer and M. Dorfman, IEEE Computer Society Press, 1990). An excellent tutorial volume of research papers in requirements engineering. A new edition is planned for 1997.

7 *Standards, Guidelines and Examples on System and Soft-*

ware Requirements Engineering (M. Dorfman and R. H. Thayer, IEEE Computer Society Press, 1990). This is a comprehensive set of standards which are relevant to requirements engineering.

8 A very comprehensive list of CASE tools, including tools for supporting the requirements engineering process, is available through the World-Wide-Web at URL
http://www.qucis.queensu.ca:1999/Software-Engineering/tools.html

Another good site for CASE tool information is at URL
http://stfc.comp.polyu.edu.hk/STFC/SoftFactory/database/tools.html

9 Techniques which are designed to help with requirements engineering process improvement have been developed in the REAIMS project and are available through the World-Wide-Web at URL:

http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/