Elizabeth Hull
Ken Jackson
Jeremy Dick

Requirements Engineering

Third Edition

Elizabeth Hull • Ken Jackson
Jeremy Dick

Requirements Engineering

Springer

Elizabeth Hull, BSc, PhD, CEng, FBCS
School of Computing and Mathematics
University of Ulster
Newtownabbey, Co Antrim
UK
mec.hull@ulst.ac.uk

Jeremy Dick, BSc (Eng), ACGI,
DPhil, DIC, MA
Integrate Systems Engineering Ltd
Bath
UK
jeremy.dick@integrate.biz

Ken Jackson, BSc, MSc, MBCS
University of Ulster
Newtownabbey, Co Antrim
UK
kenjackson@fastmail.fm

ISBN 978-1-84996-404-3 e-ISBN 978-1-84996-405-0
DOI 10.1007/978-1-84996-405-0
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010937427

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

We would like to dedicate this book as follows:

To my late parents John and Edna Hull

Elizabeth Hull

To my wife Chris,
To my children and their spouses Kate, Stef, Andy, Amy and Pete
and to my grand children Lizzie, Alice, Emily and Annabel

Ken Jackson

To my wife
Yvonne and to my children
Sebastian, Timothy, Angus, Robin and Felicity

Jeremy Dick

Preface to the Third Edition

In our desire to keep the material in this book current, the main driver in creating a new edition has been to adapt to the latest release of DOORS. Since the publication of Edition 2, Telelogic – the developer of DOORS – has been acquired by IBM, and the tool has become part of the IBM/Rational stable. While the basic functions of the tool remain unchanged, the look-and-feel has advanced considerably. Therefore, Chapter 9 has been updated for DOORS version 9.2.

At the same time, we felt the need to provide a more explicit definition of Requirements Engineering. In searching the literature, we could not find a satisfactory definition, and we have addressed this in Chapter 1.

Apart from this, there is an expanded description of Product Family Management in Chapter 8, and a variety of small corrections throughout.

We hope our readers – students and practitioners – continue to find this a valuable text in advancing their understanding of the topic.

April 2010

Elizabeth Hull
Ken Jackson
Jeremy Dick

Preface to the Second Edition

This second edition follows quickly on the first edition and is an indication of how fast the subject is changing and developing. In the past 2 years there have been significant advances and these are reflected in this new edition.

Essentially, this is an update that places more emphasis on modelling by describing a greater range of approaches to system modelling. It introduces the UML2, which is the recent standard approved by the OMG. There is also an enhanced discussion on the relationship between requirements management and modelling, which relates well to the concept of rich traceability.

The chapter on the requirements management tool DOORS has been revised to use Version 7 of the tool and this is complemented with examples taken from the DOORS/Analyst tool which demonstrates how the concepts of modelling can be captured and created within DOORS.

The text is still aimed at students and practitioners of systems engineering who are keen to gain knowledge of using requirements engineering for system development.

As before, a website supporting additional material is available at:
http://www.requirementsengineering.info

June 2004

Elizabeth Hull
Ken Jackson
Jeremy Dick

Preface to the First Edition

Requirements Engineering is common sense, but it is perceived to be difficult and is not well understood. For these reasons it is generally not very well done. The ever-increasing pressures on an organisation are often given as the main reasons for not introducing a more disciplined approach to requirements engineering, but its aim will be to do the job properly, so the task of the requirements engineer is to work out how best to help the organisation achieve its goal.

Systems engineering is critical in today's industry and requirements engineering is an important stage of that overall process. A good process is key to requirements engineering – it determines how efficiently and rapidly products can be generated. This is particularly important in a global competitive market where the 'time to market' and meeting stakeholder requirements are the key success factors.

Requirements engineering is also about management and hence issues in relation to requirements and management blend to show how requirements can be used to manage systems development.

The book is concerned with engineering requirements and how systems engineers may be helped to create better requirements. A generic process is presented which assists the reader in gaining a good understanding of the essence of requirements engineering. The process is then instantiated for the problem and solution domains of development. The book also addresses the concept of system modelling and presents various techniques and methods which are widely used. An important feature of the book is the presentation of approaches to traceability, the way in which it is captured and discusses metrics which can be derived from traceability. Finally the book presents an overview of DOORS which is a tool for requirements management. A case study is used to illustrate the process presented in the book and the features of the tool.

This book should be read by those systems engineers (requirements engineers) in industry, who, being practitioners are keen to gain knowledge of using requirements engineering for system development. The book will also be of interest to final year undergraduate students in Computer Science, Software Engineering and Systems Engineering studying a course in Requirements Engineering and also to postgraduate research students in Computer Science or more generally in Engineering.

The approach taken in the book is based on current research in Requirements Engineering, however it has not only taken the academic view but has also built substantially on current experience of working in industry to enable system engineers to manage requirements (and projects) more successfully. It provides a snapshot, in this rapidly evolving subject, of what we see as best practice in Requirements Engineering today.

A web site supporting additional material for the book can be found at: http://www.requirementsengineering.info/

May 2002

Elizabeth Hull
Ken Jackson
Jeremy Dick

Acknowledgements

Thanks are due to a number of individuals and organisations who helped in various ways:

Richard Stevens, who inspired us with his work on requirements management and who laid the foundation for the work in this book. He was a founder of Requirements Engineering Ltd. (later Quality Systems and Software Ltd.), which developed the DOORS tool.

Les Oliver (who worked for Astrium at the time) for assistance in the development of statecharts for agreement, qualification and satisfaction.

Praxis Critical Systems (now Altran Praxis) for the initial concept of design justification which become *Rich Traceability*.

Keith Collyer, Jill Burnett and other colleagues of Telelogic Ltd. for contributions to ideas presented in this book and for reviews, comments, suggestions and encouragement.

Contents

Chapter 1
Introduction

There is no fair wind for one who knows not whither he is bound.

Lucius Annaeus Seneca, philosopher, 3–65 AD

1.1 Introduction to Requirements

If ever systems development projects needed a "fair wind", they certainly do so today. Fast-changing technology and increased competition are placing ever-increasing pressure on the development process. Effective requirements engineering lies at the heart of an organisation's ability to guide the ship and to keep pace with the rising tide of complexity.

Software is currently the dominant force of change of new products. The trend is driven by three key factors:

Arbitrary complexity. The most complex systems tend to be those with software, often integrated deep inside the system's components. The complexity of such products is limited only by the imagination of those who *conceive them.*

Instant distribution. Today a company can think of a new product, implement it in software, and rapidly distribute it around the world. For example, a car manufacturer can improve the software in its diagnostic system, and then transmit it electronically around the world to tens of thousands of car showrooms in a day.

"Off-the-shelf" components. Systems are now constructed from bought-in technology and ready-made components with a corresponding reduction in the product development cycle.

The net impact of these trends is a sudden intensity of competition, and the ability to monopolise the rewards from the new technology without needing large factories. The result is pressure to reduce the development cycle and the time to deploy technology. But 'time to market' is not sufficient. The real goal is 'time to market with the right product'. Establishing the requirements enables us to agree on and visualise the 'right product'. A vital part of the systems engineering process, requirements engineering first

E. Hull et al., *Requirements Engineering*, DOI 10.1007/978-1-84996-405-0_1,

defines the problem scope and then links all subsequent development information to it. Only in this way can one expect to control and direct project activity; managing the development of a solution that is both appropriate and cost-effective.

Requirements are the basis for every project, defining what the stakeholders – users, customers, suppliers, developers, businesses – in a potential new system need from it, and also what the system must do in order to satisfy that need. To be well understood by everybody they are generally expressed in natural language and herein lies the challenge: to capture the need or problem completely and unambiguously without resorting to specialist jargon or conventions. Once communicated and agreed, requirements drive the project activity. However the needs of the stakeholders may be many and varied, and may indeed conflict. These needs may not be clearly defined at the start, may be constrained by factors outside their control or may be influenced by other goals which themselves change in the course of time. Without a relatively stable requirements base a development project can only flounder. It is like setting off on a sea journey without any idea of the destination and with no navigation chart. Requirements provide both the "navigation chart" and the means of steering towards the selected destination.

Agreed requirements provide the basis for planning the development of a system and accepting it on completion. They are essential when sensible and informed tradeoffs have to be made and they are also vital when, as inevitably happens, changes are called for during the development process. How can the impact of a change be assessed without an adequately detailed model of the prior system? Otherwise what is there to revert to if the change needs to be unwound?

Even as the problem to be solved and potential solutions are defined one must assess the risks of failing to provide a satisfactory solution. Few sponsors or stakeholders will support product or systems development without a convincing risk management strategy. Requirements enable the management of risks from the earliest possible point in development. Risks raised against requirements can be tracked, their impact assessed, and the effects of mitigation and fallback plans understood, long before substantial development costs have been incurred.

Requirements therefore form the basis for:

- Project planning
- Risk management
- Acceptance testing
- Trade off
- Change control

The most common reasons for project failures are not technical and Table 1.1 identifies the main reasons why projects fail. The data is drawn from surveys conducted by the Standish Group in 1995 and 1996, and shows the percentage of projects that stated various reasons for project failure. Those marked with an asterisk are directly related to requirements.

The problems fall into three main categories:

Requirements – either poorly organised, poorly expressed, weakly related to stakeholders, changing too rapidly, or unnecessary; unrealistic expectations

Table 1.1 Reasons for project failure

*	Incomplete requirements	13.1%
*	Lack of user involvement	12.4%
	Lack of resources	10.6%
*	Unrealistic expectations	9.9%
	Lack of executive support	9.3%
*	Changing requirements/specifications	8.7%
	Lack of planning	8.1%
*	Didn't need it any longer	7.5%

Standish Group 1995 & 1996
Scientific American, Sept. 1994

Table 1.2 Project success factors

*	User involvement	15.9%
	Management support	13.9%
*	Clear statement of requirements	13.0%
	Proper planning	9.6%
*	Realistic expectations	8.2%
	Smaller milestones	7.7%
	Competent staff	7.2%
*	Ownership	5.3%

Standish Group 1995 & 1996
Scientific American, Sept. 1994

Management problems of resources – failure to have enough money, and lack of support, or failure to impose proper discipline and planning, many of these arise from poor requirements control

Politics – which contributes to the first two problems

All these factors can be addressed at fairly low cost.

Project success factors are not quite the inverse of the failure factors, but as can be seen in Table 1.2, Management support and proper planning are clearly seen as important here – the larger the project, and the longer its schedule, the higher the chance of failure (Scientific American, Sept. 1994).

This book considers an engineering approach to requirements in general and requirements management in particular. It explains the differences between stakeholder requirements and system requirements and indicates how requirements can be used to manage system development. It also shows how traceability from stakeholder requirements through system requirements to design can be used to measure progress, manage change and assess risks. It exposes the reader to those qualities of requirements that make them suitable for validating and verifying designs and solutions, and stresses the need to produce designs that can be integrated and tested easily.

Requirements management has important interfaces to project management, which is recognised in the book through the presence of Chapter 8, "Management Aspects of Requirements Engineering".

1.2 Introduction to Systems Engineering

This book is not just about requirements for software. The principles and practice of requirements engineering apply to complete systems in which software may only play a small part.

For example, consider a railway system such as the West Coast Mainline from London to Glasgow. A high-level requirement on the system may be to achieve a journey time from Euston Station in London to Glasgow in Scotland in less than 250 min. Satisfaction of this single requirement arises from the coordinated interaction of every major component of the system:

- The trains, and their speed
- The tracks, and their ability to support high-speed trains
- The stations and station staff, and the waiting time they impose on the trains
- The drivers, and their ability to control the trains
- The signalling subsystems
- The train control and detection subsystems
- The power delivery subsystems

While the software in the signalling and control subsystems plays a vital part in achieving this requirement, it cannot deliver alone. The complete solution involves the whole system. In fact, most requirements are satisfied by the properties that emerge from the way the system as a whole behaves.

What then is meant by a "system"?

> *System*: a collection of components – machine, software and human – which co-operate in an organised way to achieve some desired result – the requirements.

Thus systems include people. In the West Coast Mainline, the drivers and station staff – the training they receive and the procedures they use – are just as important as the software and machine components.

Since components must co-operate, interfaces between components are a vital consideration in system (and requirements) engineering – interfaces between people and machine components, between machine components, and between software components. An example of a machine-to-machine interface in a railway system is the way train wheels interface with the track. Apart from the physical arrangements (which are designed to allow the train to be guided along the track without sliding off), electrical currents across the rails may be used to detect the presence of the train as part of the train control subsystem.

At the heart of the concept of a "system", lies the idea of "emergent properties". This refers to the fact that the usefulness of a system does not depend on any particular part of the system, but emerges from the way in which its components interact. Emergent properties may be desirable, in that they have been anticipated and designed into the system so as to make the system useful; or they may be