



International
Requirements
Engineering
Board

rockynook

> COMPUTING



2nd Edition

Klaus Pohl · Chris Rupp

Requirements Engineering Fundamentals

A Study Guide for the
Certified Professional for Requirements Engineering Exam
Foundation Level / IREB compliant

About the Authors



Klaus Pohl holds a full professorship for Software Systems Engineering at the Institute for Computer Science and Business Information Systems (ICB) at University of Duisburg-Essen, Germany. He was the scientific funding director of Lero, the Irish Software Engineering Research Centre. Currently he is the acting director of paluno—The Ruhr Institute for Software Technology—at the University of Duisburg-Essen. He received his Ph.D. and his habilitation in computer science from RWTH Aachen, Germany.

Klaus is (co-)author of more than 250 peer-reviewed publications and several text books. He served as Program and General Chair for many international and national conferences including the 35th ACM/IEEE Conference on Software Engineering (ICSE 2013). As consultant, assessor, and expert he supports small and multi-national companies, research institutes, and public funded research programs. Klaus is co-founder of the IREB e.V. (International Requirements Engineering Board). You can find more information on <https://sse.uni-due.de>.



Chris Rupp—SOPHIST-in-chief (formally: founder and executive partner of the SOPHIST GmbH), chief consultant, coach and trainer. Looking back over 25 years of professional experience, a lot has come up: a company . . . 6 books . . . 55 employees . . . countless articles and presentations . . . and a whole lot of experience. My passion for project consultation might account for the fact that, until now, I do not “only” manage, but I am still directly involved in projects and close to customers. What drives me is the vision to implement good ideas so that developers, contractual partners and users—both direct and indirect—face an intelligent, sophisticated and beneficial product. In doing so, I work with a range of methods and approaches in agile and non-agile environments.

In order to standardize qualification for requirements engineers / business analysts, I founded the IREB e.V.

(International Requirements Engineering Board). You can find further information on www.sophist.de.

Klaus Pohl · Chris Rupp

Requirements Engineering Fundamentals

A Study Guide for the Certified Professional
for Requirements Engineering Exam

Foundation Level – IREB compliant

2nd Edition

rockynook

Klaus Pohl (klaus.pohl@sse.uni-due.de)

Chris Rupp (chris.rupp@sophist.de)

Translated from German by Thorsten Weyer, Bastian Tenbergen, and Marta Tayeh.

Editor: Michael Barabas

Project Manager: Matthias Rossmanith

Copyeditor: Judy Flynn

Proofreader: James Johnson

Layout and Type: Josef Hegele

Cover design: Helmut Kraus, www.exclam.de

Printer: Courier

Printed in USA

ISBN 978-1-937538-77-4

2nd Edition 2015

© 2015 by Klaus Pohl and Chris Rupp

Rocky Nook Inc.

802 East Cota St., 3rd Floor

Santa Barbara, CA 93103

www.rockynook.com

Library of Congress Cataloging-in-Publication Data

Pohl, Klaus.

Requirements engineering fundamentals : a study guide for the certified professional for requirements engineering exam, foundation level, IREB compliant / Klaus Pohl, Chris Rupp. -- 2nd edition.

pages cm

ISBN 978-1-937538-77-4 (softcover : alk. paper)

1. Software engineering--Examinations--Study guides. 2. System design--Examinations--Study guides. 3. Requirements engineering--Examinations--Study guides. 4. Electronic data processing documentation--Examinations--Study guides. I. Rupp, Chris. II. Title.

QA76.758.P6413 2015

005.1076--dc23

2015009245

Many of the designations in this book used by manufacturers and sellers to distinguish their products are claimed as trademarks of their respective companies. Where those designations appear in this book, and Rocky Nook was aware of a trademark claim, the designations have been printed in caps or initial caps. They are used in editorial fashion only and for the benefit of such companies, they are not intended to convey endorsement or other affiliation with this book.

No part of the material protected by this copyright notice may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission of the copyright owner. While reasonable care has been exercised in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

This book is printed on acid-free paper.

Foreword

Dear reader,

With *Requirements Engineering Fundamentals*, you are holding the official text book of the *Certified Professional for Requirements Engineering (CPRE) – Foundation Level* certification in your hands.

The 2nd edition of this book is aligned with the curriculum (version 2.2) of the International Requirements Engineering Board e.V. (IREB) and the IREB glossary. In addition, some minor defects of the 1st edition have been corrected. A short introduction to the IREB and the certification process can be found in the previous section “The Certified Professional for Requirements Engineering (CPRE) Exam”.

The aim of this book is to aid you in your preparation for the certification examination of the Certified Professional for Requirements Engineering. The book is suited for your individual preparation for the examination as well as for companion literature to training courses offered by training providers.

In addition to the book, you should consider the information about the preparation for the certification examination published on the IREB website (<http://www.ireb.org/en>). That additional information reflects updates of the curriculum (after version 2.2) and potentially amends this book with respect to some areas of interest. Errata to this book are published on the IREB website.

Our decision to author this book collaboratively was not unjustified. The book at hand is meant to integrate long-lasting practical experiences with educational and research knowledge concerning the topic of requirements engineering, in particular for the Foundation Level of the Certified Professional for Requirements Engineering. As a consequence, this book is based on the two best-selling books in the German language about requirements engineering by the two main authors:

Klaus Pohl: *Requirements Engineering – Grundlagen, Prinzipien, Techniken*. Published at dpunkt.verlag, Heidelberg, 2008. This book was written from a perspective of research and education and offers a structured discussion of the fundamentals, principles, and techniques of requirements engineering. (Also available in English: *Requirements Engineering – Fundamentals, Principles, and Techniques*. Springer, New York, 2010)

Chris Rupp: *Requirements-Engineering und -Management – Aus der Praxis von klassisch bis agil*. Published at Hanser Fachbuchverlag, Munich, 2014. This book contains application-oriented knowledge about requirements engineering, which supports the requirements engineer in his or her daily practice. (Individual chapters also available in English on the SOPHIST website: <http://www.sophist.de>)

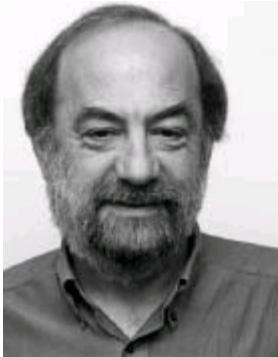
We have chosen not to reference the two books listed above in the individual chapters of this book. You can find detailed additional information on the topics of this book in both of the books mentioned above.

This book was made possible with the help of a number of people. Our special thanks go to Dirk Schüpferling and Thorsten Weyer for their contributions to this book and their outstanding commitment, without which this book would not have been possible. Many reviews and consistent support by other board members increased the quality of this book. We particularly thank all board members of the IREB for their active support. In addition, Urte Pautz of the Siemens AG; Christian Pikalek and Rainer Joppich of the SOPHIST GmbH (www.sophist.de); and Dr. Kim Lauenroth and Nelufar Ulfat-Bunyadi from “paluno – The Ruhr Institute for Software Technology” at the University of Duisburg-Essen (www.paluno.de) have contributed to individual sections of the book. Furthermore, we want to thank Thorsten Weyer and Bastian Tenbergen (paluno) as well as Marta Tayeh (SOPHIST GmbH) for their commitment towards translating this book from German into English. Thanks also to Philipp Schmidt and Dirk Schüpferling for their support in aligning this book to the IREB syllabus version 2.2.

We also want to thank Christa Preisendanz, Dr. Michael Barabas, and Judy Flynn for their support in publishing this book.

*Klaus Pohl and Chris Rupp
Essen and Nuremberg, February 2015*

With Contributions from



Karol Frühauf
INFOGEM AG, SAQ

Karol Frühauf studied in Bratislava and at RWTH Aachen, gaining his degree in computer engineering in 1975. He then spent 12 years at Brown, Boveri & Cie working as a programmer, head of quality and finally as a manager in network control technology. In 1987, Frühauf founded INFOGEM AG with Helmut Sandmayr, and the company has since gained a reputation as one of the leading system engineering consulting and training addresses in Switzerland. He is an honorary member of SAQ, the Swiss Association for Quality and was instrumental in the launch of the “Brückenwächter” (“Bridge Guard”) residence for artists and scientists in Štúrovo, Slovakia.



Emmerich Fuchs
FUCHS-INFORMATIK AG

Emmerich Fuchs has over 30 years of experience in application development. Since 1985, he has been working as a lecturer at schools of higher education and as a seminar instructor as well as a co-author of many books and an examination expert. In 1989, he founded the FUCHS-INFORMATIK AG and is now working as a consulting business manager for renowned companies in the areas of business process modeling, requirements engineering, and quality assurance.



Prof. Dr. Martin Glinz

University of Zurich

Martin Glinz is a full professor of computer science and leads the research unit Requirements Engineering at the University of Zurich. He is mainly interested in methods, languages and tools for requirement modeling. His additional fields of interest include software engineering, software quality, and modeling. He obtained his doctoral degree from RWTH Aachen in computer science. Before he accepted the call to Zurich, he worked for over 10 years in the industry as a researcher, developer, consultant, and lecturer in the field of software engineering. He is a member of the board of publishers of *Requirements Engineering* and a member of the International Requirements Engineering Board (IREB). He was chairman of the steering committee for the International Requirements Engineering Conference from 2007–2009.



Rainer Grau

Digitec Galaxus

Rainer Grau is Head of Business Development at Digitec/Galaxus, one of Switzerland's top eCommerce companies. He and his team are responsible for innovation and portfolio management as well as the implementation of all the company's strategic projects. Before joining Digitec/Galaxus he was a director and partner at Zühlke Engineering, where he was in charge of agility, lean management, requirements engineering and product management.

Rainer Grau holds various teaching posts at Swiss universities and is actively involved in SAQ, the Swiss Association for Quality. He is a founder member of the Swiss Agile Leaders Circle where he supports community members in their requirements engineering, enterprise agility and lean management activities.

Rainer Grau likes to spend his free time with his family, on his bicycle, windsurfing, rock climbing or reading the latest novels by T.C. Boyle and Haruki Murakami.

**Colin Hood**

Colin Hood Systems Engineering Ltd.

Starting out in 1977, Colin Hood has accompanied the evolution of control systems from their beginnings in relay-based systems through programmable logic controllers (PLCs) to modern software-controlled safety-critical systems. His various jobs have included analysis, design, implementation, testing and delivery of complex software systems. Requirements engineering has always been the foundation of his success at companies such as Alcatel, BMW, DaimlerChrysler, Hella and Miele. As well as continually improving the processes involved, he specializes in introducing new methods and tools that support the process of change.

**Dr. Frank Houdek**

Daimler AG

Frank Houdek graduated in Computer Science at the University of Ulm and joined the Daimler Research Centre in 1995. After completing his PhD in empirical software engineering in 1999 he began working in requirements engineering and has headed various research and technology transfer projects within the Daimler passenger car and commercial vehicles business units. Since 2013 he has been responsible for coordinating the requirements engineering activities for all electric/electronic specifications in Mercedes-Benz passenger car development.

Dr. Houdek is a member of GI (German Interest Group on Computer Science) and IEEE CS, and belongs to the steering committee of the GI Group 2.1.6 (Requirements Engineering). He is also involved in the organizational and program committees for requirements engineering events such as RE, REFSQ, and ICSE.

He is responsible for the *Requirements Engineering* module of the *Software Engineering for Embedded Systems* course at the Technical University at Kaiserslautern.



Dr. Peter Hruschka
Atlantic Systems Guild

Peter Hruschka has been working as an independent IT and management consultant since 1994. His mission is the practical implementation of new ideas in software engineering. This comprises the entire spectrum from analysis of the initial situation via the creation of strategic plans to introductory training for every (structured or object-oriented) method and process to guarantee success. Dr. Hruschka is principal of the Atlantic Systems Guild, an internationally renowned group of experts on software technology, and founder of the German network of agile developers.



Prof. Dr. Barbara Paech
University of Heidelberg

Barbara Paech is a professor with the Institute for Computer Science of the University of Heidelberg. Until October 2003, she was a department leader with the Fraunhofer Institute for Experimental Software Engineering. Her area of research is software engineering, especially the methods and processes necessary to improve quality with appropriate effort. For many years, she has been active in the area of requirements engineering and usability engineering. Paech and her group have implemented many national, international and industrial research and technology transfer projects. She is a member of the International Requirements Engineering Board (IREB).



Dirk Schüpferling

SOPHIST GmbH

I am a SOPHIST since 2001 and the past years have led me to the conclusion that, in most cases, communication is the key to (customer) satisfaction. What surprised me was that features like laziness or being a know-it-all can—applied correctly—lead to something positive. The specialist calls this “reuse” or “identifying potential for improvement”. I transmit this knowledge as a classic Requirements-Engineer, as well as in agile contexts (e.g., as Product Owner) in various projects. My job is to support the project team in the conception or application of new methods.



Thorsten Weyer

University of Duisburg-Essen

Thorsten Weyer is a research group leader at the University of Duisburg-Essen and Head of Requirements Engineering and Conceptual Design at “paluno – The Ruhr Institute for Software Technology” at the University of Duisburg-Essen. He has worked for more than a decade as a researcher and consultant in requirements engineering, systems analysis, variability management, and model-based software engineering. He is a member of the organizational and program committees for various scientific conferences and also contributes his expertise to research funding projects and international trade publications. Thorsten Weyer is a member of the International Requirements Engineering Board (IREB) and co-publisher of the *Requirements Engineering Magazine*.

Contents

Foreword

With Contributions from

1 Introduction and Foundations

1.1 Introduction

 1.1.1 Figures and Facts from Ordinary Projects

 1.1.2 Requirements Engineering – What Is It?

 1.1.3 Embedding Requirements Engineering into Process Models

1.2 Fundamentals of Communication Theory

1.3 Characteristics of a Requirements Engineer

1.4 Requirement Types

1.5 Importance and Categorization of Quality Requirements

1.6 Summary

2 System and Context Boundaries

2.1 System Context

2.2 Defining System and Context Boundaries

 2.2.1 Defining the System Boundary

 2.2.2 Defining the Context Boundary

2.3 Documenting the System Context

2.4 Summary

3 Eliciting Requirements

3.1 Requirements Sources

 3.1.1 Stakeholders and Their Significance

 3.1.2 Handling Stakeholders in the Project

3.2 Requirements Categorization According to the Kano Model

3.3 Elicitation Techniques

 3.3.1 Types of Elicitation Techniques

 3.3.2 Survey Techniques

 3.3.3 Creativity Techniques

 3.3.4 Document-centric Techniques

 3.3.5 Observation Techniques

 3.3.6 Support Techniques

3.4 Summary

4 Documenting Requirements

4.1 Document Design

4.2 Types of Documentation

 4.2.1 The Three Perspectives of Requirements

 4.2.2 Requirements Documentation using Natural Language

 4.2.3 Requirements Documentation using Conceptual Models

 4.2.4 Hybrid Requirements Documents

4.3 Document Structures

 4.3.1 Standardized Document Structures

 4.3.2 Customized Standard Contents

4.4 Using Requirements Documents

4.5 Quality Criteria for Requirements Documents

 4.5.1 Unambiguity and Consistency

 4.5.2 Clear Structure

 4.5.3 Modifiability and Extendibility

 4.5.4 Completeness

 4.5.5 Traceability

4.6 Quality Criteria for Requirements

4.7 Glossary

4.8 Summary

5 Documenting Requirements in Natural Language

5.1 Effects of Natural Language

5.1.1 Nominalization

5.1.2 Nouns without Reference Index

5.1.3 Universal Quantifiers

5.1.4 Incompletely Specified Conditions

5.1.5 Incompletely Specified Process Verbs

5.2 Requirement Construction using Templates

5.3 Summary

6 Model-Based Requirements Documentation

6.1 The Term Model

6.1.1 Properties of Models

6.1.2 Modeling Languages

6.1.3 Requirements Models

6.1.4 Advantages of Requirements Models

6.1.5 Combined Use of Models and Natural Language

6.2 Goal Models

6.2.1 Goal Documentation Using AND/OR Trees

6.2.2 Example of AND/OR Trees

6.3 Use Cases

6.3.1 UML Use Case Diagrams

6.3.2 Use Case Specifications

6.4 Three Perspectives on the Requirements

6.5 Requirements Modeling in the Data Perspective

6.5.1 Entity-Relationship Diagrams

6.5.2 UML Class Diagrams

6.6 Requirements Modeling in the Functional Perspective

- 6.6.1 Data Flow Diagrams
 - 6.6.2 Models of the Functional Perspective and Control Flow
 - 6.6.3 UML Activity Diagrams
 - 6.7 Requirements Modeling in the Behavioral Perspective
 - 6.7.1 Statecharts
 - 6.7.2 UML State Diagrams
 - 6.8 Summary
- 7 Requirements Validation and Negotiation**
- 7.1 Fundamentals of Requirements Validation
 - 7.2 Fundamentals of Requirements Negotiation
 - 7.3 Quality Aspects of Requirements
 - 7.3.1 Quality Aspect “Content”
 - 7.3.2 Quality Aspect “Documentation”
 - 7.3.3 Quality Aspect “Agreement”
 - 7.4 Principles of Requirements Validation
 - 7.4.1 Principle 1: Involvement of the Correct Stakeholders
 - 7.4.2 Principle 2: Separating the Identification and the Correction of Errors
 - 7.4.3 Principle 3: Validation from Different Views
 - 7.4.4 Principle 4: Adequate Change of Documentation Type
 - 7.4.5 Principle 5: Construction of Development Artifacts
 - 7.4.6 Principle 6: Repeated Validation
 - 7.5 Requirements Validation Techniques
 - 7.5.1 Commenting
 - 7.5.2 Inspection
 - 7.5.3 Walk-Through
 - 7.5.4 Perspective-Based Reading
 - 7.5.5 Validation through Prototypes
 - 7.5.6 Using Checklists for Validation

- 7.6 Requirements Negotiation
 - 7.6.1 Conflict Identification
 - 7.6.2 Conflict Analysis
 - 7.6.3 Conflict Resolution
 - 7.6.4 Documentation of the Conflict Resolution

- 7.7 Summary

8 Requirements Management

- 8.1 Assigning Attributes to Requirements
 - 8.1.1 Attributes for Natural Language Requirements and Models
 - 8.1.2 Attribute Scheme
 - 8.1.3 Attribute Types of Requirements
- 8.2 Views on Requirements
 - 8.2.1 Selective Views on the Requirements
 - 8.2.2 Condensed Views on the Requirements
- 8.3 Prioritizing Requirements
 - 8.3.1 Method for Requirements Prioritization
 - 8.3.2 Techniques for Requirements Prioritization
- 8.4 Traceability of Requirements
 - 8.4.1 Advantages of Traceable Requirements
 - 8.4.2 Purpose-Driven Definition of Traceability
 - 8.4.3 Classification of Traceability Relations
 - 8.4.4 Representation of Requirements Traceability
- 8.5 Versioning of Requirements
 - 8.5.1 Requirements Versions
 - 8.5.2 Requirements Configurations
 - 8.5.3 Requirements Baselines
- 8.6 Management of Requirements Changes
 - 8.6.1 Requirements Changes

- 8.6.2 The Change Control Board
 - 8.6.3 The Change Request
 - 8.6.4 Classification of Incoming Change Requests
 - 8.6.5 Basic Method for Corrective and Adaptive Changes
- 8.7 Measurement of Requirements
- 8.7.1 Product vs. Process Metric
 - 8.7.2 Examples of Product and Process Metrics
- 8.8 Summary
- 9 Tool Support**
- 9.1 General Tool Support
- 9.2 Modeling Tools
- 9.3 Requirements Management Tools
- 9.3.1 Specialized Tools for Requirements Management
 - 9.3.2 Standard Office Applications
- 9.4 Introducing Tools
- 9.5 Evaluating Tools
- 9.5.1 Project View
 - 9.5.2 User View
 - 9.5.3 Product View
 - 9.5.4 Process View
 - 9.5.5 Provider View
 - 9.5.6 Technical View
 - 9.5.7 Economic View
- 9.6 Summary

References

The glossary of those terms used in this book (IREB-Glossary) can be found on the website of the “International Requirements Engineering Board e.V.”

www.ireb.org/en

1 Introduction and Foundations

The impact of requirements engineering (RE) on successful and customer-oriented systems development can no longer be ignored. It has become common practice to provide resources for requirements engineering. In addition, there is a growing understanding that the role of the requirements engineer is essentially self-contained and comprises a series of demanding activities.

1.1 Introduction

Why perform requirements engineering?

According to the figures reported in the Standish Group's Chaos Report of 2006, much has improved in the execution of software projects in the twelve years between 1994 and 2006. While about 30 percent of the software projects investigated in 1994 failed, it was a mere 20 percent in 2006. The number of projects that exceeded time or budget constraints significantly and/or did not meet customer satisfaction dropped from 53 percent to 46 percent [Chaos 2006]. Jim Johnson, chairperson of the Standish Group, names three reasons for the positive development of the figures since 1994. One is that the communication of requirements has much improved since ten years ago. These figures are of importance since how the requirements of a system are handled is a significant cause for project failures and/or time and budget overruns.

1.1.1 Figures and Facts from Ordinary Projects

Requirements engineering as a cause of errors

According to past studies, approximately 60 percent of all errors in system development projects originate during the phase of requirements engineering [Boehm 1981]. These errors, however, are often discovered only in later project phases or once the system has been deployed because incorrect or incomplete requirements can be interpreted by developers in such a fashion that they are subjectively sound or (subconsciously) complete. Missing requirements often remain undetected during design and realization because developers trust the requirements engineers to deliver

high-quality work. Developers implement whatever the requirements document says or what they believe it to be saying. Unclear, incomplete, or wrong requirements inevitably lead to the development of a system that does not possess critical properties or possesses properties that were not requested.

Costs of errors during requirements engineering

The later in the development project a defect in the requirements is corrected, the higher are the costs associated with fixing it. For instance, the effort to fix a requirements defect is up to 20 times higher if the correction is done during programming as opposed to fixing the same defect during requirements engineering. If the defect is fixed during acceptance testing, the effort involved may be up to a 100 times higher [Boehm 1981].

Symptoms and causes of deficient requirements engineering

Symptoms for inadequate requirements engineering are as numerous as their causes. Frequently, requirements are missing or not clearly formulated. For instance, if the requirements do not reflect customer wishes precisely or if the requirements are described in an imprecise way and thus allow for several interpretations, the result is often a system that does not meet the expectations of the client or the users.

The most common reason for deficient requirements is the misconception of the stakeholders that much is self-evident and does not need to be stated explicitly. This results in problems in communication among the involved parties that arise from differences in experience and knowledge. To make matters worse, it is often the case that especially the client wishes for quick integration of recent results into a productive system.

The significance of good requirements engineering

The increasing importance of software-intensive systems in industrial projects as well as the need to bring more innovative, more individual, and more comprehensive systems to market and the need to do so quicker, better, and with a higher level of quality calls for efficient requirements engineering. Complete requirements free from defects are the basis for successful system development. Potential risks have to be identified during requirements engineering and must be reduced as early as possible to allow for successful project progress. Faults and gaps in requirement documents must be discovered early on to avoid tedious change processes.

1.1.2 Requirements Engineering – What Is It?

In order to make a development project succeed, it is necessary to know the requirements for the system and to document them in a suitable manner.

Definition 1-1: *Requirement*

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

[IEEE 610.12-1990]

Stakeholders

The term *stakeholder* is essential in requirements engineering. Among other things, stakeholders are the most important sources of requirements. Not considering a stakeholder often results in fragmentally elicited requirements, i.e., incomplete requirements [Macaulay 1993]. Stakeholders are those people or organizations that have some impact on the requirements. This could be people that are going to interact with the system (e.g., users or administrators), people that have a mere interest in the system but are not likely to use it (e.g., the management, a hacker from which the system must be protected, stakeholders of competing systems), but also legal entities, institutions, etc., because these are embodied by living people who may choose to influence or define the requirements of the system.

Definition 1-2: *Stakeholder*

A stakeholder of a system is a person or an organization that has an (direct or indirect) influence on the requirements of the system.

Goal of requirements engineering

During the development process, requirements engineering must elicit the stakeholders' requirements, document the requirements in a suitable manner, validate and verify the requirements, and manage the requirements over the course of the entire life cycle of the system [Pohl 1996].

Definition 1-3: *Requirements Engineering*

- (1) Requirements engineering is a systematic and disciplined approach to the specification and management of requirements with the following goals:
 - (1.1) Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically
 - (1.2) Understanding and documenting the stakeholders' desires and needs, specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs

Four core activities of requirements engineering

The four core activities to meet these ends are as follows:

- *Elicitation*: During requirements elicitation, different techniques are used to obtain requirements from stakeholders and other sources and to refine the requirements in greater detail.
- *Documentation*: During documentation, the elicited requirements are described adequately. Different techniques are used to document the requirements by using natural language or conceptual models (see [chapters 4, 5, and 6](#)).
- *Validation and negotiation*: In order to guarantee that the predefined quality criteria are met, documented requirements must be validated and negotiated early on (see [chapter 7](#)).
- *Management*: Requirements management is orthogonal to all other activities and comprises any measures that are necessary to structure requirements, to prepare them so that they can be used by different roles, to maintain consistency after changes, and to ensure their implementation (see [chapter 8](#)).

These core activities can be applied for different levels of requirements abstraction, like stakeholder requirements, system requirements, and software requirements. Their execution can follow different processes, such as the processes recommended in [ISO/IEC/IEEE 29148:2011].

Constraints

Different project constraints influence requirements engineering. For instance, people, domain factors, or organizational constraints (e.g., spatial distribution or temporal availability of project members) have a large impact on the choice of suitable techniques.

1.1.3 Embedding Requirements Engineering into Process Models

Requirements engineering as a self-contained phase

Ponderous process models (e.g., the Waterfall model [Royce 1987] or the V-Model [V-Modell 2004]) aim at completely eliciting and documenting all requirements in an early project phase before any design or realization decisions are made. The goal of such models is to elicit all requirements prior to the actual development. As a result, in these process models, requirements engineering is understood to be a finite, time-restricted initial phase of system development.

Requirements engineering as a continuous, collateral process

Lightweight process models (e.g., eXtreme Programming [Beck 1999]), on the other hand, only elicit necessary requirements once they are supposed to be implemented as “foretelling” future functionalities is difficult and requirements change over the course of the project. In these process models, requirements engineering is treated as a continuous, comprehensive process that comprises and integrates all phases of system development.

1.2 Fundamentals of Communication Theory

Language as a medium for requirement communication

Requirements must be communicated. In most cases, one uses a rule-driven medium that is accessible to the communication partner—natural language.

For the transmission of information from one individual to another to work properly, a common code is needed. The sender encodes her message and the receiver has to decode it. Such a common code is intrinsic to any two people that speak the same language (e.g., German), have the same cultural background, and have similar experiences. The more similar the cultural and educational background, the area of expertise, and the everyday work life, the better the exchange of information works. However, such ideal conditions most often do not exist between stakeholders. It is therefore sensible to agree upon a common language and how this common language is to be used. This can, for instance, be achieved by means of glossaries (see [chapter 4](#)), in which all important terms are explained. Alternatively, this can be done by agreeing upon a formal descriptive language, e.g., OMG’s Unified Modeling Language, UML (see [chapter 6](#)).

Type of communication medium

Another important factor is the type of communication medium. In verbal communication, the success of the communication relies heavily on redundancy (e.g., language and gestures or language and intonation) and feedback. In written technical communication, for example, information is transmitted with a minimum of redundancy and feedback.

Language comfort

In addition to the problems arising from differing domain vocabularies and different communication media, it can often be observed that information is not adequately transmitted or not transmitted at all. This can be traced back to natural transformations that occur during human perception. These transformational effects are, in particular, *focusing* and *simplification* and can impact the communication more or less harshly.

Implicit background knowledge

Communication—i.e., the language-based expression of knowledge—is necessarily simplifying in nature. The author expects the reader to have some kind of implicit background knowledge. It is the simplifications that arise from language-based knowledge expression that become problematic with regard to requirements, as requirements can become interpretable in different ways. In [chapter 5](#), natural language-based requirement documentation is discussed in further detail.

1.3 Characteristics of a Requirements Engineer

Central role

The requirements engineer as a project role is often at the center of attention. She is usually the only one who has direct contact with the stakeholders and has both the ability and the responsibility to become as familiar as possible with the domain and to understand it as well as possible. She is the one that identifies the needs underlying the stakeholders' statements and amends them in a way that architects and developers—usually laymen where the domain in question is concerned—can understand and implement them. The requirements engineer is, in a manner of speaking, a translator that understands the domain as well as its particular language well enough and also possesses enough IT know-how to be aware of the problems the developers face and to be able to communicate with them on the same level. The requirements engineer therefore has a central role in the project.

Seven necessary capabilities of a requirements engineer

To be able to fulfill all of her tasks, the requirements engineer needs much more than process knowledge. Many of the capabilities required must be based on practical experience.

- *Analytic thinking*: The requirements engineer must be able to become familiar with domains that are unknown to her and must understand and analyze complicated problems and relationships. Since stakeholders often discuss problematic requirements by means of concrete examples and (suboptimal) solutions, the requirements engineer must be able to abstract from the concrete statements of the stakeholder.
- *Empathy*: The requirements engineer has the challenging task of identifying the actual needs of a stakeholder. A core requirement to be able to achieve this is to have good intuition and empathy for people. In addition, she must identify problems that might arise in a group of stakeholders and act accordingly.
- *Communication skills*: To elicit the requirements from stakeholders and to interpret them correctly and communicate them in a suitable manner, a requirements engineer must have good communication skills. She must be able to listen, ask the right questions at the right time, notice when a statement does not contain the desired information, and make further inquiries when necessary.
- *Conflict resolution skills*: Different opinions of different stakeholders can be the cause of conflicts during requirements engineering. The requirements engineer must identify conflicts, mediate between the parties involved, and apply techniques suitable to resolving the conflict.
- *Moderation skills*: The requirements engineer must be able to mediate between different opinions and lead discussions. This holds true for individual conversations as well as group conversations and workshops.
- *Self-confidence*: Since the requirements engineer is frequently at the center of attention, she occasionally is exposed to criticism as well. As a result, she needs a high level of self-confidence and the ability to defend herself should strong objections to her opinions arise. She should never take criticism personally.
- *Persuasiveness*: Among other things, the requirements engineer is, in a matter of speaking, a kind of attorney for the requirements of the stakeholders. She must be able to represent the requirements in team meetings and presentations. In addition, she must consolidate differing opinions, facilitate a decision in case of a disagreement, and create consensus among the stakeholders.

1.4 Requirement Types

Generally, one can distinguish between three types of requirements:

- *Functional requirements* define the functionality that the system to be developed offers. Usually, these requirements are divided into functional requirements, behavioral requirements, and data requirements (see [chapter 4](#)).

Definition 1-4: Functional Requirement

A functional requirement is a requirement concerning a result of behavior that shall be provided by a function of the system.

- *Quality requirements* define desired qualities of the system to be developed and often influence the system architecture more than functional requirements do. Typically, quality requirements are about the performance, availability, dependability, scalability, or portability of a system. Requirements of this type are frequently classified as non-functional requirements.

Definition 1-5: Quality Requirement

A quality requirement is a requirement that pertains to a quality concern that is not covered by functional requirements.

- *Constraints* cannot be influenced by the team members. Requirements of this type can constrain the system itself (e.g., “The system shall be implemented using web services”) or the development process (“The system shall be available on the market no later than the second quarter of 2012”). In contrast to functional and quality requirements, constraints are not implemented, they are adhered to because they merely limit the solution space available during the development process.

Definition 1-6: Constraint

A constraint is a requirement that limits the solution space beyond what is necessary for meeting the given functional requirements and quality requirements.

In addition to the classification into functional requirements, quality requirements, and constraints, a number of different classifications of requirements are used in practice. For example, there are a number of classifications suggested by several standards, e.g., CMMI [SEI 2006] or SPICE [ISO/IEC 15504-5]. Other classification schemes describe requirement attributes, such as the level of detail of a requirement, the priority, or the degree of legal obligation of requirements (see [chapters 4 and 8](#)).

1.5 Importance and Categorization of Quality Requirements

In daily practice, quality requirements of a system are often not documented, inadequately documented, or improperly negotiated. Such circumstances can threaten the project's success or the subsequent acceptance of the system under development. Therefore, the requirements engineer should place special emphasis on the elicitation, documentation, and negotiation of quality requirements during the development process.

Typically, many different kinds of desired qualities of the system are assigned to the requirement type *quality requirement*. In order to be able to deal with quality requirements in a structured manner, many different classification schemes for quality requirements have been proposed. The ISO/IEC 25010:2011 standard [ISO/IEC 25010:2011], for example, suggests a classification scheme for quality requirements that can also be used as a standard structure for requirements documentation and as a checklist for requirements elicitation and validation. Among others, the following categories are typical for quality requirements (see [ISO/IEC 25010:2011]):

- Requirements that define the performance of the system, in particular response time behavior and resource utilization
- Requirements that define the security of the system, in particular with regard to accountability, authenticity, confidentiality, and integrity
- Requirements that define the reliability of functionalities, in particular with regard to availability, fault tolerance, and recoverability
- Requirements that define the usability of a system, in particular with regard to accessibility, learnability, and ease of use
- Requirements that define the maintainability of a system, in particular with regard to reusability, analyzability, changeability, and testability
- Requirements that define the portability of a system, in particular with regard to adaptability, installability, and replaceability

Currently, quality requirements are often specified using natural language. However, numerous approaches to document quality requirements by means of models have been suggested over the past couple of years.

The requirements engineer is responsible for making sure the quality requirements are as objective and verifiable as possible. Typically, this necessitates that the quality requirements are quantified. For example, a quality requirement with regard to system performance could specify that a system shall process 95 percent of all queries within 1.5 seconds and that it must not take longer than 4 seconds to process queries at any

given time. This can cause quality requirements to be refined by means of additional functional requirements. This could be the case for a quality requirement that is concerned with system security if a functional requirement specifies the exact encryption algorithm to satisfy the need for encryption as demanded by some quality requirement.

Quality requirements are often related to different functional requirements. As a result, quality requirements should always be kept separated from functional requirements. In other words, quality requirements should not be mixed with functional requirements and should be documented separately, with explicit documentation of their relation to functional requirements.

1.6 Summary

Requirements engineering can hardly be avoided, especially when systems are to be developed that satisfy customers and meet budget constraints and schedules. The goal of requirements engineering is to document customer requirements as completely as possible in good quality and to identify and resolve problems in the requirements as early as possible. Successful requirements engineering is based on including the right stakeholders as well as embedding the four core activities of requirements engineering (*elicitation, documentation, validation and negotiation, and management*) into the system development process. At the center of attention is the requirements engineer, who is the primary contact point in requirements engineering and possesses a great deal of domain knowledge and process knowledge as well as a multitude of soft skills.

2 System and Context Boundaries

The requirements for a system to be developed do not simply exist, they have to be elicited. The purpose of defining the system and context boundaries in requirements engineering is to identify the part of the environment that influences the requirements for the system to be developed.

2.1 System Context

Anticipate the system in operation

In the development process, requirements engineering fulfils the task of identifying all those material and immaterial aspects that have a relationship to the system. In order to do that, it is anticipated what the system will be like once it becomes real. By doing so, those parts of the real world which will potentially influence the requirements of the system can be identified. To be able to specify the requirements for a system correctly and completely, it is necessary to identify the relationships between individual material and immaterial aspects as precisely as possible. The part of reality that is relevant for the requirements of a system is called the system context.

Definition 2-1: System Context

The system context is the part of the system environment that is relevant for the definition as well as the understanding of the requirements of a system to be developed.

Context aspects in the system context

Among others, the following possible aspects of reality influence the context of a system:

- People (stakeholders or groups of stakeholders)
- Systems in operation (other technical systems or hardware)
- Processes (technical or physical processes, business processes)
- Events (technical or physical)
- Documents (e.g., laws, standards, system documentation)

Consequence of erroneous or incomplete context consideration

If the system context is incorrectly or incompletely considered during requirements engineering, it may result in incomplete or erroneous requirements. This leads to the system operating on the basis of incomplete or erroneous requirements, which is often the reason for system failure during operation. Such errors often remain undetected during the validation procedures, which determine if the system meets the specified requirements, and occur only during operation, sometimes entailing catastrophic consequences.

System context and requirement context

The origin of the system's requirements lies within the context of the system to be developed. For example, stakeholders, pertinent standards, and legal guidelines demand particular functional properties that the system to be developed must possess at its interfaces. A requirement is therefore defined for a specific context and can only be interpreted correctly in regard to this specific context. The better the context of a requirement is understood (e.g., why is the technical system "X" in the system context the origin of some requirement), the lower the likelihood of incorrect interpretation of the requirement. Therefore, a purpose-driven documentation of the system context or information about the system context is of particular importance.

2.2 Defining System and Context Boundaries

It is within the responsibility of the requirements engineer to define the system context properly. In order to do so, it is necessary to separate the system context from the system to be developed as well as from the parts of reality that are irrelevant for the system (see [figure 2-1](#)):

- *Defining the system boundary:* When defining the system boundary, a decision has to be made: Which aspects pertain to the system to be developed and which aspects belong in the system context?
- *Defining the context boundary:* When defining the context boundary, the question to be answered is: Which aspects pertain to the system context (i.e., have a relation to the system to be developed) and which aspects are part of the irrelevant environment?

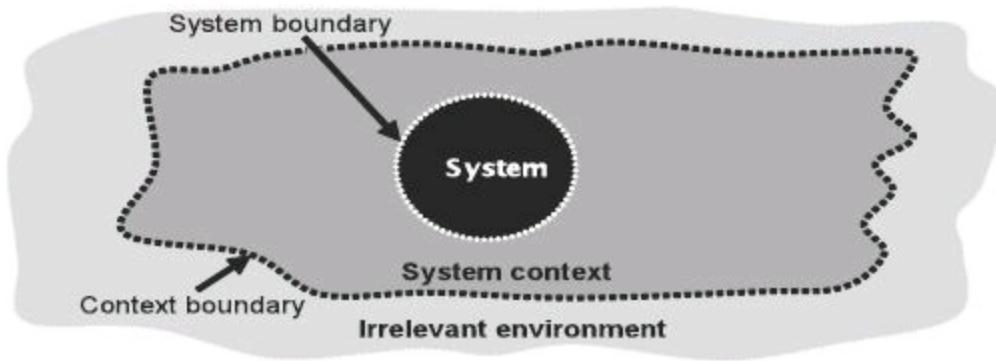


Figure 2-1 System and context boundary of a system

System and context boundaries define the system context.

Thus, system and context boundaries define the system context. The system context comprises all aspects that are relevant with regard to the requirements for the system to be developed. These aspects cannot be altered or modified by the system development process.

2.2.1 Defining the System Boundary

The system boundary separates the object of concern (i.e., the system) from its environment. When the system boundary is defined, the scope of the development (i.e., the aspects that are covered by the system to be developed) as well as the aspects that are not part of the system are determined. We therefore define the system boundary as follows:

Definition 2-2: System Boundary

The system boundary separates the system to be developed from its environment; i.e., it separates the part of the reality that can be modified or altered by the development process from aspects of the environment that cannot be changed or modified by the development process.

All aspects that are within the system boundary can thus be altered during system development. For instance, an existing system that consists of hardware and software components and is supposed to be replaced by the new system can be within the system boundary. Aspects within the system context can be business processes, technical processes, people and roles, organizational structures, and components of the IT infrastructure. [Figure 2-2](#) schematically shows the system context of a system. The system context consists of other systems, groups of stakeholders that in some way use the interfaces of the system to be developed, and additional requirements sources and their interrelations.

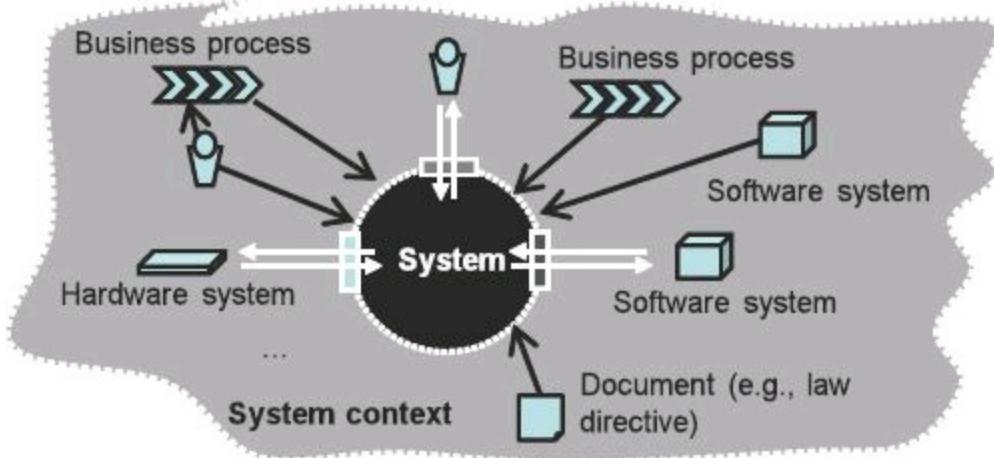


Figure 2-2 Types of aspects within the system context

Sources and sinks as the starting point

Among other things, sources and sinks (see, e.g., [DeMarco 1978]) can be used to identify the interfaces the system has with its environment. Sources provide inputs for the system. Sinks receive outputs from the system. Possible sources and sinks of a system are as follows:

- (Groups of) stakeholders
- Existing systems (both technical and nontechnical systems)

Interfaces: interaction between system and environment

Sources and sinks interact with the system to be developed via system interfaces. Using these interfaces, the system provides its functionality to the environment, monitors the environment, influences parameters of the environment, and controls operations of the environment. Depending on the type of the respective source or sink, the system needs different interface types (e.g., human–machine interface, hardware interface, or software interface). The interface type in turn may also impose specific constraints or additional sources of requirements on the system to be developed.

Gray zone between system and system context

Frequently, the system boundary is not precisely defined until the end of the requirements engineering process. Before that, some or several interfaces as well as desired functions and qualities of the system to be developed are only partially known or not known at all. We refer to this initially vague separation of the system and its context as the gray zone between the system and the context (see [figure 2-3](#)). At the beginning of the requirements engineering process, it may, for example, not be clear whether the system should implement a certain function (e.g., “pay by credit card”) or

whether there is another system in the system context providing such a function that should be used (e.g., “payment processing”).

Adjusting the gray zone

The system boundary may not only shift within the gray zone (① in [figure 2-3](#)) but also the gray zone itself may shift during the requirements engineering process (② in [figure 2-3](#)). This kind of shifting is caused by the fact that aspects, pertaining at first to the system context, now will be modified during system development. Such a situation occurs during requirements engineering, for example, if it is not clear in the system context whether certain activities of a business process should be implemented or supported by the system to be developed or not. In this situation, it is not clear which aspects belong to the system and can thus be changed or modified and which aspects belong to the system context. This causes a corresponding shift of the gray zone between system and system context (see [figure 2-3](#)).

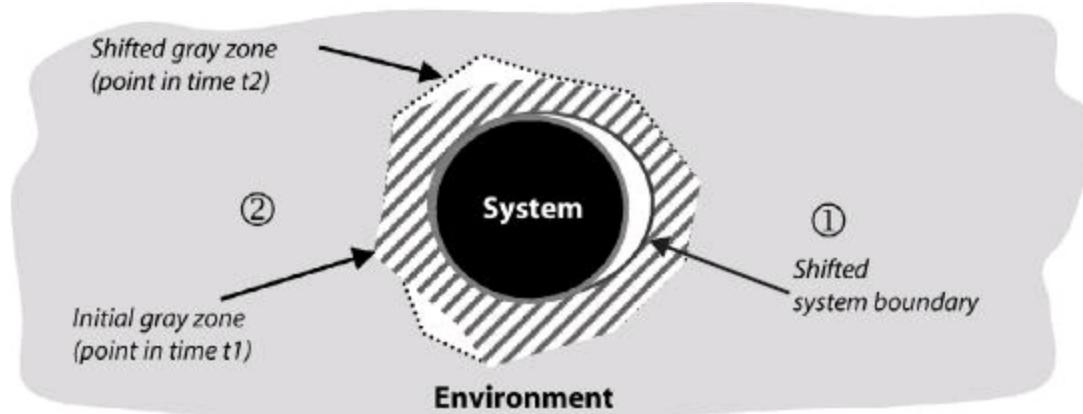


Figure 2-3 Gray zone of the system boundary

The gray zone shifts, for instance, when interfaces are attributed to the system boundary and the gray zone is extended to comprise aspects of the environment that concern these interfaces.

2.2.2 Defining the Context Boundary

The context boundary distinguishes between context aspects, i.e., those aspects of the environment that need to be taken into account during requirements engineering (e.g., as requirements sources) and those aspects that are irrelevant for the system. The context boundary can be defined as follows:

Definition 2-3: Context Boundary

The context boundary separates the relevant part of the environment of a system to be developed from the

irrelevant part, i.e., the part that does not influence the system to be developed and, thus, does not have to be considered during requirements engineering.

Concretion and shift of the context boundary

At the beginning of the requirements engineering process, frequently only part of the environment as well as single specific relationships between the environment and the system to be developed are known. In the course of requirements engineering, it is necessary to concretize the boundary between system context and irrelevant environment by analyzing relevant aspects within the environment with regard to their relationships to the system. Besides the system boundary, the context boundary typically also shifts during requirements engineering. For instance, it may be possible that a law directive that was considered to be relevant for the system to be developed no longer impacts the system or is no longer considered relevant. The system context is therefore reduced (① in [figure 2-4](#)). If a new law directive is identified that influences the system, the system context is extended accordingly (② in [figure 2-4](#)).

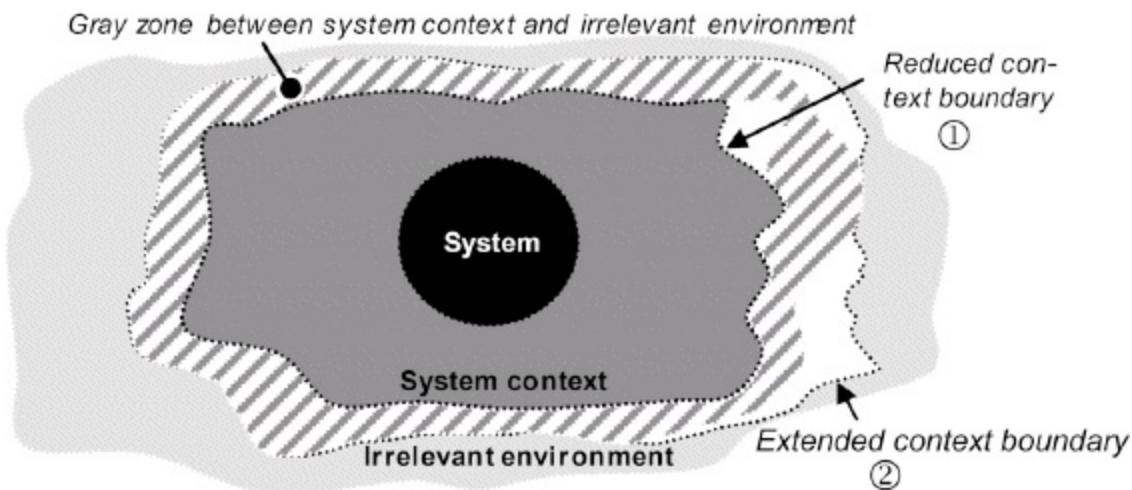


Figure 2-4 Gray zone between system context and irrelevant environment

Gray zone between system context and irrelevant environment

Since the context boundary separates the system context from those parts of reality that are irrelevant to the system, a complete and precise definition of the context boundary for complex systems is virtually impossible. In addition, it may not be possible to clarify for single aspects of the environment whether they influence the system to be developed or are influenced by it or not. These two observations are the reason for the existence of a gray zone with regard to the context boundary (see [figure 2-4](#)).

Resolving and shifting of the gray zone

This gray zone therefore comprises identified aspects of the environment for which it is unclear whether they have a relation to the system or not. In contrast to the gray

zone between the system and the system context that must be resolved in the course of requirements engineering, it is not necessary to resolve the gray zone between the system context and the irrelevant environment entirely.

2.3 Documenting the System Context

In order to document the system context (especially the system and context boundaries), “use case” diagrams [Jacobson et al. 1992] (see sections 4.2.3 and 6.3.1) or “data flow” diagrams [DeMarco 1978] (see section 6.6.1) are often used. When the context is modeled with data flow diagrams, sources and sinks in the environment of the system that represent the source or destination of data flows (or flows of material, energy, money, etc.) are modeled. In use case diagrams, actors (such as people or other systems) in the system environment and their usage relationships to the system are modeled. To model the system context, UML class diagrams [OMG 2007] (see section 6.5.2) may also be used. In order to document the system context of a system as thoroughly as possible, typically several documentation forms are used.

2.4 Summary

The system context is the part of the reality that influences the system to be developed and thus also influences the requirements for the system. In order to be able to elicit the requirements for the system to be developed, it is necessary to define the boundary of the system to the system context and the boundary of the system context to the irrelevant environment first. When the system boundaries are defined, the scope of the system is determined. The scope comprises those aspects that can be changed and designed during system development. At the same time, it is also defined which aspects belong to the environment and thus cannot be altered during development and may provide constraints for the system to be developed.

The context boundary separates the part of the environment that influences the requirements for the system to be developed from that part that does not influence the requirements. Typical aspects within the system context are stakeholders (e.g., the users of the system) and documents (e.g., standards that have to be considered) as well as other systems that, for instance, interact with the system to be developed. Defining the system and context boundaries successfully is the foundation for a systematic elicitation of requirements for the system to be developed.

3 Eliciting Requirements

A core activity of requirements engineering is the elicitation of requirements for the system to be developed. The basis for requirements elicitation is the knowledge that has been gained during requirements engineering about the system context of the system to be developed, which comprises the requirements sources that are to be analyzed and queried.

3.1 Requirements Sources

Three types of requirements sources

There are three different kinds of requirements sources:

- *Stakeholders* (see section 1.1.2) are people or organizations that (directly or indirectly) influence the requirements of a system. Examples of stakeholders are users of the system, operators of the system, developers, architects, customers, and testers.
- *Documents* often contain important information that can provide requirements. Examples of documents are universal documents, such as standards and legal documents, as well as domain- or organization-specific documents, such as requirements documents and error reports of legacy systems.
- *Systems in operation* can be legacy or predecessor systems as well as competing systems. By giving the stakeholders a chance to try the system out, they can gain an impression of the current system and can request extensions or changes based on their impressions.

3.1.1 Stakeholders and Their Significance

Significance of stakeholders

Identifying the relevant stakeholders is a central task of requirements engineering [Glinz and Wieringa 2007]. For the requirements engineer, stakeholders are important

sources of requirements for the system (see section 1.1.2). It is the task of the requirements engineer to gather, document, and consolidate the partially conflicting goals and requirements of different stakeholders [Potts et al. 1994] (see [chapter 8](#)).

Consequences of unconsidered stakeholders

If stakeholders are not identified or not considered, it may result in significant negative repercussions for the project progress because requirements may remain undetected. At the latest, these overlooked requirements will enter the picture in the form of change requests during system operation. Fixing these issues retroactively causes high additional costs. Therefore, it is essential to identify all stakeholders and integrate them into the elicitation procedures.

Stakeholder lists provide overview.

An auxiliary technique for stakeholder identification is maintaining checklists. This allows for systematic and targeted elicitation of relevant stakeholders. If the stakeholder list is updated too late or incompletely, the result may be that important aspects of the system remain undetected, that the project goal is missed, or that significant additional costs arise from fixing issues. The starting point for stakeholder elicitation is often suggestions of relevant stakeholders that are made by management or by domain experts, for example. On the basis of these suggestions, relevant stakeholders can be identified.

3.1.2 Handling Stakeholders in the Project

Managing stakeholders

It can often be observed in practice that a lot of stakeholders are involved in complex and “difficult” projects. Due to limited resources, the stakeholders that are the most suitable for requirements elicitation must be carefully selected. To document the stakeholders in the development process, it makes sense to use tables and spreadsheets that contain (at least) the following data: name, function (role), additional personal and contact data, temporal and spatial availability during the project progress, relevance of the stakeholder, area and extent of expertise of the stakeholder, and the stakeholder’s goals and interests regarding the project.

Making collaborators out of the affected

Handling stakeholders also means continuously exchanging information: Periodic status updates and continuous involvement of the stakeholders assist the requirements engineer in turning people previously simply affected by the project (i.e., principally

affected stakeholders) into collaborators (i.e., well-integrated, jointly responsible stakeholders).

Individual “contracts” with the stakeholders

Stakeholders that are not given enough attention by the requirements engineer might be overly critical toward the project. In addition, some stakeholders may show a lack of motivation because they are sufficiently satisfied with the legacy system, are afraid of change, or are negatively biased due to previous projects. It's the requirements engineer's task to support the project manager in convincing all stakeholders of the benefit of the project. To avoid misunderstandings and disputes regarding competence, it is useful to formally agree on the tasks, responsibilities, and managerial authority as well as to determine individual goals, communication paths, and feedback loops that can be used by the stakeholders. Depending on the culture of the organization, this agreement and determination can be done verbally (i.e., by “shaking hands”) or, more formally, by means of written documentation. The individual agreements should be signed off by the managers.

Obligations and privileges of the stakeholders

A number of obligations and privileges result from the agreement with the stakeholders.

The requirements engineer

- speaks the language of the stakeholders,
- becomes thoroughly familiar with the application domain,
- creates a requirements document,
- is able to get work results across (e.g., by means of diagrams and graphs),
- maintains a respectful relationship with any stakeholder,
- presents her ideas and alternatives as well as their realizations,
- allows stakeholders to demand properties that make the system user-friendly and simple,
- ensures that the system satisfies the functional and qualitative demands of the stakeholders.

The stakeholders

- introduce the requirements engineer to the application domain,
- supply the requirements engineer with requirements,
- document requirements assiduously,

- make timely decisions,
- respect the requirements engineer's estimates of costs and feasibility,
- prioritize requirements,
- inspect the requirements that the requirements engineer documents, such as prototypes, etc.,
- communicate changes in requirements immediately,
- adhere to the predetermined change process,
- respect the requirements engineering process that has been instated.

Elicitation techniques determine communication and process.

In addition, the requirements engineer plans and organizes the communication paths as well as drafts a structured schedule for the requirements engineering activities that are to be performed in collaboration with the stakeholders. This organization and the type of communication are significantly influenced by the elicitation techniques that can be used during requirements engineering.

3.2 Requirements Categorization According to the Kano Model

Influence of the requirements on satisfaction

Knowing the importance of a requirement for the satisfaction of the stakeholders is very helpful for requirements elicitation. Along with the respective properties of a product that determine the satisfaction, the satisfaction is classified into the following three categories [Kano et al. 1984]:

- *Dissatisfiers* are properties of the system that are self-evident and taken for granted (subconscious knowledge).
- *Satisfiers* are explicitly demanded system properties (conscious knowledge).
- *Delighters* are system properties that the stakeholder does not know or expect and discovers only while using the system—a pleasant and useful surprise (unconscious knowledge).

As time goes by, delighters turn into satisfiers and dissatisfiers as the user becomes accustomed to the properties of the system. When eliciting requirements, all three categories must be considered.

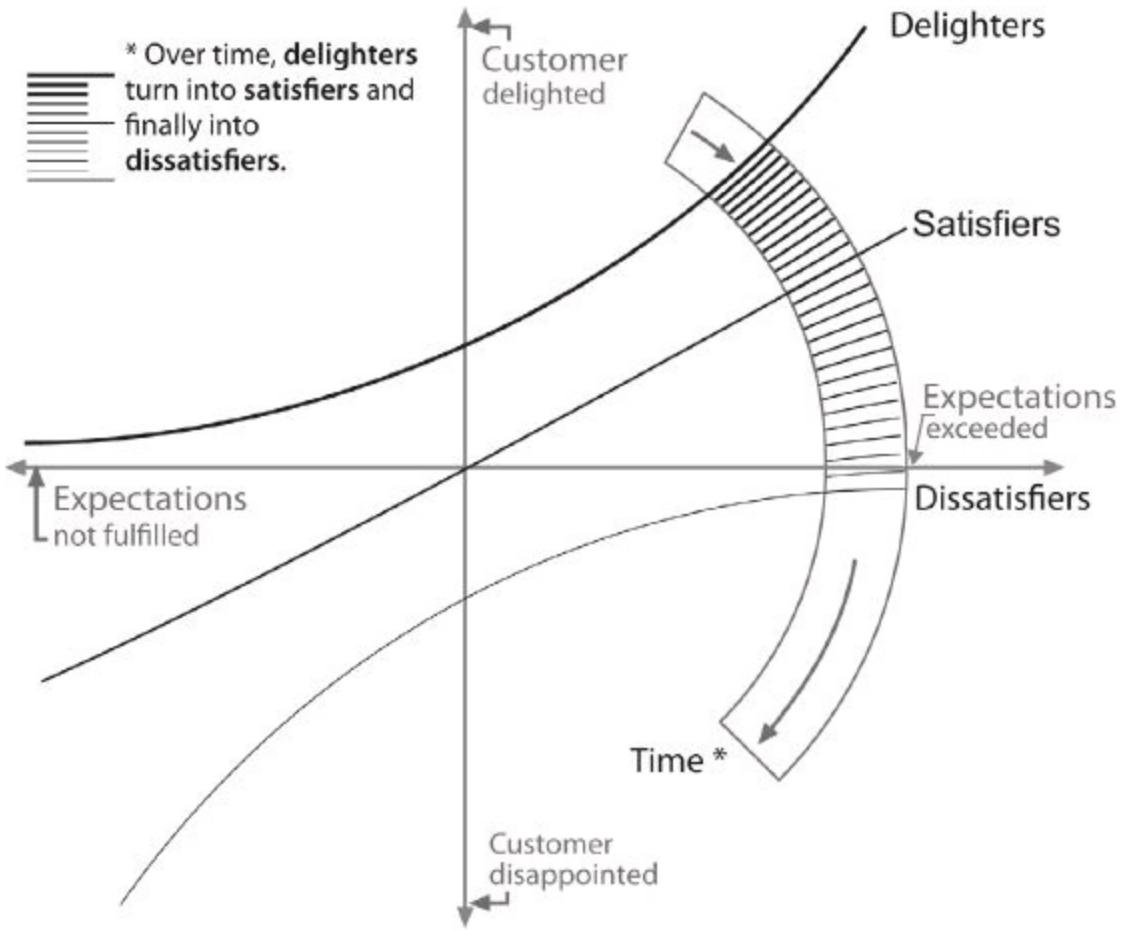


Figure 3-1 Graphical representation of the Kano model

Dissatisfiers

Dissatisfiers (subconscious requirements) must be fulfilled by the system in any case. Otherwise, stakeholders will be disappointed and dissatisfied. Completely fulfilled dissatisfiers do not generate a positive disposition but merely help to avoid massive discontent. Dissatisfiers are dominantly influenced by existing systems. Therefore, observation and document-centric techniques are especially well suited for the elicitation of these factors.

Satisfiers

Satisfiers (conscious requirements) are properties that are consciously known to the stakeholders and explicitly demanded. When these properties are fulfilled, stakeholders are content and satisfied, which is desirable. If some demanded properties are missing, the stakeholders probably will not accept the product. Their satisfaction decreases with each missing satisfier. Satisfiers can be elicited well using survey techniques.

Delighters

Delighters (unconscious requirements) are properties of a system whose value is recognized only when the stakeholder can try out the system for herself or the requirements engineer proposes them. Creativity techniques are best suited to elicit delighters.

3.3 Elicitation Techniques

Requirements elicitation: no universal method

The main goal of all elicitation techniques is in supporting the requirements engineer in ascertaining the knowledge and requirements of the stakeholders. How and when a technique can be applied depends on the given conditions. Applying the technique consciously and in a fashion appropriate to the situation at hand allows for tailoring the requirements elicitation process which takes into account project constraints so that requirements may be elicited as completely and comprehensibly as possible.

3.3.1 Types of Elicitation Techniques

Influencing factors regarding the choice of elicitation techniques

Elicitation techniques serve the purpose of identifying the conscious, unconscious, and subconscious stakeholder requirements. However, there is no universal method to elicit these requirements [Hickey and Davis 2003]. Every project has individual constraints and individual characteristics and is by and large unique, but there are always elicitation techniques that are compatible with the project. The most important influencing factors when choosing the appropriate elicitation techniques are as follows:

- the distinction between conscious, unconscious, and subconscious requirements that are to be elicited
- the time and budget constraints, as well as the availability of the stakeholders
- the experience of the requirements engineer with a particular elicitation technique
- the chances and risks of the project

Risk factors

The first important step when choosing a suitable elicitation technique is to perform an analysis of constraints critical to the project, i.e., identifying so-called risk factors.

Mostly, these result from human, organizational, and professional influences, as illustrated in the following passages.

Human influences

During the requirements elicitation phase, which is heavily influenced by the stakeholders, good communication is essential. In order to assure high-quality communication between the requirements engineer and stakeholders, it is important to determine the type of requirement, the desired level of detail, and the experience of the requirements engineer and the interviewees with different elicitation techniques.

Social, group-dynamic, and cognitive capabilities of the stakeholders also influence the choice of suitable elicitation techniques significantly. Another influence factor is whether the elicited knowledge is explicit (consciously known) by each individual stakeholder or if it is implicit or unconscious (i.e., covert).

Organizational influences

Organizational risk factors the project faces need to be investigated as well. Among other things, this comprises the distinction between fixed price contracts and service contracts, whether the system to be built is a new development or an extension of a legacy system, and spatial and temporal availability of the stakeholders.

Operational influences of the content

In addition, it is necessary to consider the operational content of the requirements. If the system is very complex, it is advisable to employ a structuring approach during elicitation in order to deconstruct the operational contents into understandable parts.

Combine techniques with regard to your particular situation to lower risks.

Another influencing factor on the choice of elicitation techniques is the desired level of detail of the requirements. Abstract requirements can be elicited rather well using creativity techniques. With the stakeholders, a vision of the system or its important properties can be created or collected. Inquisitive (survey) techniques or observational techniques can aid in eliciting requirements of a medium level of detail [Robertson 2002]. Finely detailed requirements can be elicited well by making use of document-centric techniques, i.e., techniques that use existing documents because information up to an arbitrary level of detail can be extracted from these.

It is advisable to combine different techniques because this minimizes many of the risks inherent to the project. Weaknesses and pitfalls of a particular technique can be balanced out through the use of another technique whose strong points lie where the

first technique may have deficits.

3.3.2 Survey Techniques

Eliciting explicit knowledge

Survey techniques aim at eliciting as precise and unbiased statements as possible from stakeholders regarding their requirements. All survey techniques assume that the respondent is capable of explicitly expressing his or her knowledge and that he or she is committed to investing time and effort for the elicitation. Survey techniques are usually driven by the requirements engineer because she asks the questions. This, however, might result in the fact that stakeholder concerns are forgotten, superseded, or disregarded.

Interview

- During an *interview*, the requirements engineer asks predetermined questions to one or more stakeholders and documents the answers. Questions that arise during the conversation can be discussed immediately, and the requirements engineer may uncover subconscious requirements through clever questions. Interviews can be employed during the entire development phase of the system. An experienced interviewer individually controls the course of the conversation, completely commits herself to each stakeholder, inquires about specific aspects, and thus ensures the completeness of the answers. The most prominent disadvantage of this elicitation technique is that it is very time-consuming.

Questionnaire

- *Questionnaire*: Making use of open and/or closed questions (e.g., multiple choice questions) is another way of eliciting requirements from stakeholders. If there are a large number of participants that must be surveyed, an online questionnaire is a viable option. Questionnaires can elicit a magnitude of information in a short amount of time and at low costs. As long as answers are predetermined, even stakeholders that are not able to explicitly express their knowledge can deliver an assessment. A disadvantage of using a questionnaire is that it can be only employed to gather requirements the requirements engineer already knows or conjectures. Creating a proper questionnaire is often tricky and time-consuming and requires thorough knowledge of the domain in question and the psychological guidelines for creating questionnaires. In addition, as opposed to interviews, questionnaires do not provide immediate feedback between the

surveyor and the surveyed, so it becomes apparent that questions were forgotten or badly formulated only once the questionnaires have been evaluated.

3.3.3 Creativity Techniques

Establishing innovations

Creativity techniques serve the purpose of developing innovative requirements, delineating an initial vision of the system, and eliciting excitement factors. Creativity techniques are usually not well suited for establishing fine-grained requirements about the system behavior. The following creativity techniques are commonly used [Maiden and Gizikis 2001]:

Brainstorming

- During *brainstorming*, ideas are collected within a certain time frame, usually in groups of 5 to 10 people. The ideas are documented by a moderator without discussing, judging, or commenting on them at first. Participants use ideas of other participants to develop new original ideas or to modify existing ideas. After that, the collected ideas are subjected to a thorough analysis. This technique is especially effective when a large number of people of different stakeholder groups are involved. Among the advantages of this technique is that a large number of ideas can be collected in a short amount of time and multiple people can expand on these ideas collaboratively. The unbiased collection of these ideas allows new solutions to pop up. Brainstorming is usually less effective when the dynamics of the group are muddled or when participants with very varied levels of dominance are involved. For such situations, other creativity techniques may be better suited, e.g., the 6-3-5 method (six participants, three ideas each, fivefold hand-off of the ideas) [Rohrbach 1969] or the brainwriting method.

Brainstorming paradox

- *Brainstorming paradox* is a modification of regular brainstorming in that events that must not occur are collected. Afterward, the group develops measures to prevent the events collected earlier from happening. Through this process, participants often realize which actions may entail negative results. With this method, risks can be identified early on and countermeasures can be developed. Advantages and disadvantages of this technique are identical to those of classic brainstorming.

Change of perspective

- *Change of perspective*: Among the techniques that employ a change of perspective (adopting different extreme standpoints), the most common technique is the so-called Six Thinking Hats [DeBono 2006]. Each of the six hats represents a particular perspective that is in turn adopted by each of the participants. The resulting solutions approach the problem from different standpoints. That way, even stakeholders that are very convinced of their own opinion are persuaded to adopt a different standpoint. This technique is extraordinarily beneficial when stakeholders can only express their knowledge in a biased manner or are harshly constricted to their opinions. On the other hand, this technique cannot be applied if the requirements require a fine-grained level of detail because this would render the technique very laborious.

Analogy technique

- *Analogy techniques (bionics/bisociations)*: In bionics, problems that the project faces are mapped to an analogous situation occurring in nature, and the solutions nature provides are sought and then mapped back to the project. In bisociation, the analogies need not originate in nature. These techniques assume that each participant is capable of analogous thinking, that a lot of time is available, and that the participants have an in-depth knowledge of the domain with which an analogy will be drawn. Analogy techniques can be applied covertly or in the open. When this technique is applied covertly, the participants are only told the analogy. The requirements engineer is then responsible for mapping the results onto the real problem space. When this technique is applied in the open, the stakeholders know the real problem space as well as the analogy.

3.3.4 Document-centric Techniques

Document-centric techniques reuse solutions and experiences made with existing systems. When a legacy system is replaced, this technique ensures that the entire functionality of the legacy system can be identified. Document-centric techniques should be combined with other elicitation techniques so that the validity of the elicited requirements can be determined and new requirements for the new system can be identified.

System archaeology

- *System archaeology* is a technique that extracts information required to build a new system from the documentation or implementation (code) of a legacy system or a competitor's system. The technique is often applied when explicit knowledge about the system logic has been lost partially or entirely. By analyzing existing code, the requirements engineer ensures that none of the functionalities of the legacy system will be overlooked and the system logic of the legacy system is elicited anew. This method leads to a large amount of very detailed requirements and is very laborious. However, system archaeology is the only technique that can ensure that all functionalities of the legacy system will be implemented in the new system. When it becomes obviously apparent that the legacy system and the new system differ in functionality, additional elicitation techniques, e.g., creativity techniques, must be applied early on.

Perspective-based reading

- *Perspective-based reading* (see section 7.5.4) is applied when documents need to be read with a particular perspective in mind, e.g., the perspective of the implementer or the tester. Aspects that are contained in the document but do not pertain to the current perspective are ignored. This allows for an analysis that is strictly focused on particular parts of the existing documentation. This way, detailed, technology-related or implementation-related aspects can be separated from essential operational aspects that are relevant for the successor system.

Reuse

- *Reuse*: Requirements that have been previously compiled and brought up to a certain quality standard can be reused. In order to do that, the requirements are stored in a database, for instance, and kept available at the required level of detail for reuse. Through reuse, the costs involved with the elicitation procedures can be significantly reduced.

3.3.5 Observation Techniques

Question observations and optimize processes.

When domain specialists are unable to spend the time needed to share their expertise with the requirements engineer, or are unable to express and denote their knowledge, observation techniques are helpful. During observation, the requirements engineer observes the stakeholders while they go about their work. The requirements engineer

documents all steps and thus elicits the processes the system must support as well as potential mistakes, risks, and open questions. All those are potential requirements that need to be formulated. The stakeholders can actively demonstrate their knowledge in using the application or can remain passive, with the requirements engineer merely observing. The requirements engineer ought to question the observed processes so that the situation as it should be can be established. Otherwise, she is at risk of documenting outdated technological decisions and suboptimal processes (i.e., the situation as is and not as it should be). As the requirements engineer is an external observer, her chances of identifying inefficient processes are good and she can then suggest better solutions. She is farther removed from the processes than the stakeholders, who frequently repeat work steps without questioning them critically. Observation techniques are well suited to elicit detailed requirements and dissatisfiers because the requirements engineer can recognize dissatisfiers thought of as self-evident or only subconsciously known by the stakeholders. In addition, the requirements engineer becomes very familiar with the domain language, which simplifies further elicitation. Satisfiers can only be observed if they have been implemented in the legacy system or are actively employed in the current processes. As a result, this technique is not suited for the development of new processes. During system development, field observations and apprenticing are especially well suited as elicitation techniques.

Field observation

- *Field observation:* The requirements engineer is on location with the specialist or the users of the system and observes and documents the processes and operational procedures that they carry out. Using these observations, she formulates the requirements. Often, this can be further aided by audio and video recordings. This technique is well suited for operational procedures that are difficult to express verbally, but it can only be applied if the procedures are visible physically.

Apprenticing

- With *apprenticing*, the requirements engineer must actively learn and perform the procedures of the stakeholders. Just like an apprentice, the requirements engineer is encouraged to question unclear and complex operational procedures so that she may gather domain experience. Thereby, she can experience requirements that the stakeholders take for granted and therefore cannot elucidate. Another advantage is that the typical balance of power between the requirements engineer and the respective specialist is reversed because the stakeholder now adopts the role of

the “master” that has the knowledge the apprentice is yet lacking.

3.3.6 Support Techniques

Support techniques serve as an addition to the elicitation techniques and try to balance out the weaknesses and pitfalls of the chosen elicitation technique.

Mind mapping

- In *mind mapping*, a graphical representation of the refined relationships and interdependencies between terms is created. Mind mapping is often used as a supporting technique for brainstorming or brainstorming paradox.

Workshops

- During a joint meeting, the requirements engineer and the stakeholders elaborate the goals (or details of a certain functionality) of the system. For example, the necessary user interfaces of the system can be designed in a *workshop* [Gottesdiener 2002].

CRC cards

- With the CRC technique (CRC stands for *Class Responsibility Collaboration*), context aspects and their respective attributes and properties are denoted on index cards. Requirements are then formulated using these cards.

Audio and video recordings

- *Audio and video recordings* are very well suited to elicit requirements when stakeholders are not always available, when budget is tight, or when the system is highly critical. Especially during field observations, audio and video recordings can help capture fast-paced processes. The disadvantage of this technique is that stakeholders often feel supervised when they are being recorded and as a result might deliver biased statements or, in extreme cases, might even refuse to cooperate.

Modeling action sequences

- *Use case modeling*: Use cases document the external view of the system to be

developed. A use case has a trigger event, which triggers the use case and an expected result, or outcome of the use case. Every use case is a functionality that must be supported by the system to be developed (see section 6.3).

Prototypes for illustration

- *Prototypes* are well suited to question established requirements and to elicit requirements in situations where stakeholders have only a vague understanding of what is to be developed. Potential consequences of new or changed requirements can be identified easier. For example, user interface prototypes are frequently used in practice to find additional functional requirements.

3.4 Summary

Requirements elicitation is a core activity in requirements engineering. Aside from documents and legacy systems, stakeholders are the main sources for requirements. It is important to initially agree upon mutual rights and responsibilities of the stakeholders and the requirements engineer in order to facilitate communication and cooperation and to successfully integrate the stakeholders into the elicitation process. The choice of the right elicitation technique for the respective project is made by the requirements engineer based on the given cultural, organizational, and domain-specific constraints.

4 Documenting Requirements

In requirements engineering, information that has been established or worked out during different activities must be documented. Among this information are, for example, protocols of interviews and reports of validation or agreement activities, but also change requests. The main and most important documentation task in requirements engineering, though, is to document the requirements for the system in a suitable manner.

4.1 Document Design

A documentation technique is any kind of more or less formal depiction that eases communication between stakeholders and increases the quality of the documented requirements. In principle, any kind of documentation technique can be used to document the requirements, let it be natural language-based documentation by means of prose, more structured natural language-based text, or more formal techniques such as state diagrams.

Definition 4-1: *Requirements Document / Requirements Specification*

A requirements specification is a systematically represented collection of requirements, typically for a system or component, that satisfies given criteria.

Reasons for the documentation

During the life cycle of a requirements document, many people are trusted with the documentation. During communication, the documentation has a goal-oriented and supporting role. The main reasons for documenting requirements are as follows:

Central role of requirements

- *Requirements are the basis of the system development.* Requirements of any kind influence the analysis, design, implementation, and test phases directly and indirectly. The quality of a requirement or of a requirements document has a strong impact on the progress of the project and therefore on its success.

Legal relevance

- *Requirements have a legal relevance.* Requirements are legally binding for the contractor and the client, and the client can sue for their fulfillment. Documenting the requirements can help to quickly overcome legal conflicts between two or more parties.

Complexity

- *Requirements documents are complex.* Systems that possess thousands of requirements that in turn have complex interdependencies on multiple layers are not unheard of in practice. Without suitable documentation, keeping on top of things can become very difficult for anyone involved.

Accessibility

- *Requirements must be accessible to all involved parties.* Projects undergo certain “development” as time goes by—with regard to the subject as well as the staff. When requirements can be permanently accessed, uncertainty and obscurities can be avoided and staff that has recently joined the project can quickly get up to speed.

Another argument for a good documentation, supportive of the project, is that employees almost never share the same understanding of a subject matter. Therefore, requirements should be documented in a way that they meet the quality demands of all involved.

4.2 Types of Documentation

Requirements for a system can be documented in three different perspectives. In practice, natural language as well as conceptual models are used to this end, or oftentimes, an advantageous combination of both is employed.

4.2.1 The Three Perspectives of Requirements

Requirements for a system can be documented in three different perspectives onto the system to be developed:

Data perspective

- *Data perspective*: In the data perspective, a static-structural perspective on the requirements of the system is adopted. For example, the structure of input and output data as well as static-structural aspects of usage and dependency relations of the system and the system context can be documented (e.g., the services of an external system).

Functional perspective

- *Functional perspective*: The functional perspective documents which information (data) is received from the system context and manipulated by the system or one of its functions. This perspective also documents which data flows back into the system context. The order in which functions processing the input data are executed is also documented.

Behavioral perspective:

- *Behavioral perspective*: In the behavioral perspective, information about the system and how it is embedded into the system context is documented in a state-oriented manner. This is done by documenting the reactions of the system upon events in the system context, the conditions that warrant a state transition, and the effects that the system shall have on its environment (e.g., effects of the system analyzed that represent events in the system context of a different system).

4.2.2 Requirements Documentation using Natural Language

Advantages of using natural language

Natural language, particularly prose, is the most commonly applied documentation form for requirements in practice. In contrast to other documentation forms, prose has a striking advantage: No stakeholder has to learn a new notation. In addition, language can be used for miscellaneous purposes—the requirements engineer can use natural language to express any kind of requirement.

Disadvantages of using natural language

Natural-language-based documentation is well suited to document requirements in any of the three perspectives. However, natural language can allow requirements to be ambiguous, and requirements of different types and perspectives are in danger of being

unintentionally mixed up during documentation. In that case, it is difficult to isolate information pertaining to a certain perspective amidst all of the requirements in natural language.

4.2.3 Requirements Documentation using Conceptual Models

In contrast to natural language, the different types of conceptual models cannot be used universally. When documenting requirements by means of models, special modeling languages must be used that pertain to the appropriate perspective. Assuming the modeling language selected for a documentation task is applied correctly, its use constructively guarantees that the models created depict information pertaining to the respective perspective only. The models depict the documented requirements much more compactly and they therefore are easier for a trained reader to understand than is natural language. In addition, conceptual models offer a decreased degree of ambiguity (i.e., fewer ways to be interpreted) than natural language due to their higher degree of formality. However, using conceptual modeling languages for requirements documentation requires specific knowledge of modeling. The following list includes short descriptions of the most important diagrams discussed in [chapter 6](#).

Overview of system functions

- *Use case diagram:* A use case diagram allows you to gain a quick overview of the functionalities of the specified system. A use case describes which functions are offered to the user by the system and how these functions relate to other external interacting entities. However, use cases do not describe the responsibilities that the functions have in detail (see section 6.3).

Structural data modeling and structuring of terms

- *Class diagram:* Among other things, class diagrams are used in requirements engineering to document requirements with regard to the static structure of data, to document static-structural dependencies between the system and the system context, or to document complex domain terms in a structured manner (see section 6.5.2).

Sequence modeling

- *Activity diagram:* Using activity diagrams, business processes, or sequence-oriented dependencies of the system in regard to processes within the system

context can be documented. Activity diagrams are also well suited to model the sequential character of use cases or to model a detailed specification of the interaction of functions that process data (see section 6.6.3).

Event-driven behavior

- *State diagram:* State diagrams are used in requirements engineering to document event-driven behavior of a system. The focus of this type of model is on the individual states the system can be in, events and their respective conditions that trigger a state transition, and effects of the system in its environment.

4.2.4 Hybrid Requirements Documents

Combined use of documentation types

Requirements documents first and foremost contain requirements. In addition, in many situations it is sensible to document decisions, important explanations, and other relevant information as well. Depending on the target audience of the document, the perspective on the system, and the documented knowledge, suitable documentation types are selected. Typically, documents contain a combination of natural language and conceptual models. The combination allows the disadvantages of both documentation types to be decreased by means of the strengths of the other documentation type, and combining documentation types exploits the advantages of both. For instance, models can be amended or complemented by natural language comments and natural language requirements and natural language glossaries can be summarized and their dependencies can be depicted clearly by making use of models.

4.3 Document Structures

Influence of the requirements on satisfaction

Requirements documents contain a magnitude of different information. These must be well structured for the reader. In order to do that, one can make use of standardized document structures or individually define a custom document structure.

4.3.1 Standardized Document Structures

Adaptation of existing standard outlines

Standard outlines offer a predefined structure, i.e., predefined stereotypes according to which the information can be classified. By using standard outlines, a rough structure along with a short description of the content of the main sections is predetermined. Using standard outlines has the following advantages:

- Standard outlines simplify incorporating new staff members.
- Standard outlines allow for quickly finding desired contents.
- Standard outlines allow for selective reading and validation of requirements documents.
- Standard outlines allow for automatic verification of requirements documents (e.g., with regard to completeness).
- Standard outlines allow for simplified reuse of the contents of requirements documents.

It must be noted that these structures must be tailored with regard to the specific project properties to meet the respective constraints. In the following paragraphs, three of the most widely used standardized document structures are introduced.

Rational Unified Process

The *Rational Unified Process (RUP)* [Kruchten 2001] is usually used for software systems that are developed using object-oriented methods. The client creates a *business model* that contains different artifacts from the business environment (e.g., business rules, business use cases, business goals), which serve as the basis for requirements of the system over the course of development. The contractor uses the structures of the *software requirements specification (SRS)* to document all software requirements. These structures are closely related to the ISO/IEC/IEEE standard 29148:2011, as described next.

ISO/IEC/IEEE standard 29148:2011

The ISO/IEC/IEEE standard 29148:2011 [ISO/IEC/IEEE 29148:2011] contains an outline designed for the documentation of software requirements (software requirements specification). The standard structure suggests dividing the requirements document into five parts with regard to their subject matter:

- A chapter with introductory information (e.g., system goal, system bounding) and a general description of the software (e.g., perspective of the system, properties of future users)

- A chapter with a listing of all documents that are referenced in the specification
- A chapter for specific requirements (e.g., functional requirements, performance, interfaces)
- A chapter with all planned measures for verification
- Appendices (e.g., information about assumptions that were made, identified dependencies)

V-Model

The *V-Model* [V-Modell 2004] of the German Federal Ministry of the Interior (BMI) defines different structures, depending on the creator of the requirements document:

- The *Customer Requirements Specification*, known in the German original as Lastenheft, is created by the customer and describes all of the demands to the contractor regarding the subject of the contract, i.e., deliveries and services. In addition, in many cases, demands of the users, including all constraints to the system and the development process, are documented. Therefore, the Customer Requirements Specification usually describes *what is made for what*.
- The *System Requirements Specification*, known in the German original as Pflichtenheft, is based on the Customer Requirements Specification and contains the implementation suggestions that the contractor has elaborated. Therefore, the System Requirements Specification is a refinement of the requirements and constraints of the Customer Requirements Specification.

4.3.2 Customized Standard Contents

The minimum content

As described in section 4.3.1, standardized document structures are adapted with regard to the specific project conditions. The following issues should be addressed by any chosen structure.

Introduction

The introduction contains information about the entire document. This information allows gaining a quick overview of the system.

- *Purpose*: This section discusses why the document was created and who the target audience for the requirements document is.

- *System coverage*: This part consists of the system to be developed. It indicates system name and the principle goals and advantages that arise from introducing the system.
- *Stakeholder*: This section contains a list of stakeholders and their relevant information (see section 3.1.1).
- *Definitions, Acronyms, and Abbreviations*:¹ In this section, the terms used in the document are defined so that they can be used consistently throughout the document.
- *References*:² All documents that are referenced by the requirements document are listed herein.
- *Overview*: At the end of the introductory chapter, the content and structure of the following sections of the requirements document should be explained briefly.

General Overview

In this section, additional information is documented that increases the understandability of the requirements. In contrast to the introduction, this is merely operational information that does not pertain to administration, management, or organizational aspects of the requirements document.

- *System environment*: The embedding of the system into the environment is of key concern in this paragraph. The results of your definition of the system boundary and context boundary can be found herein.
- *Architecture description*: In this section, the operational interfaces of the system (e.g., user interfaces, hardware and software interfaces, and communication interfaces) are documented. In addition, further information, e.g., regarding storage limitations, is also discussed.
- *System functionality*: This section contains the coarse functionalities and tasks of the system. This can be documented, for example, using a use case diagram.
- *User and target audience*: The different users of the system that make up the target audience are listed.
- *Constraints*: In this section, all conditions ought to be listed that have not been documented thus far and might hinder the requirements engineering.
- *Assumptions*: Decisions, such as not implementing certain aspects of the system due to budgeting reasons, or other general assumptions about the system context that the requirements are based upon are documented here.

Requirements

This part contains functional requirements as well as quality requirements.

Appendices

In the appendices, additional information that completes the document can be documented. For example, the appendices can include additional documents regarding the user characteristics, standards, conventions, or background information regarding the requirements document.

Index

The index typically contains a table of contents (i.e., a structure of the chapters) and an index directory. In highly dynamic requirements documents, this may be a highly critical section that must be kept up-to-date.

4.4 Using Requirements Documents

Requirements documents as the basis for development

Over the course of the project, requirements documents serve as the basis for different tasks:

- *Planning*: Based on the requirements document, concrete work packages and milestones for the implementation of the system can be defined.
- *Architectural design*: The detailed documented requirements (along with constraints) serve as the basis for the design of the system architecture.
- *Implementation*: Based on the architectural design, the system is implemented by making use of the requirements.
- *Test*: On the basis of requirements that have been documented in the requirements document, test cases can be developed that can be used for system validation later on.
- *Change management*: When requirements change, the requirements document can serve as the basis to analyze the extent to which other parts of the system are influenced. The change effort can thus be estimated.
- *System usage and system maintenance*: After the system is developed, the

requirements document is used for maintenance and support. This way, the requirements document can be used to analyze concrete defects and shortcomings that surface during system use. For example, one can deduct if a defect is a result of using the system incorrectly, a result of an error in requirements, or a result of an error in implementation.

- *Contract management*: The requirements document is the prime subject of a contract between a client and a contractor in many cases.

4.5 Quality Criteria for Requirements Documents

To become a basis for the subsequent processes, the requirements document must meet certain quality criteria. According to the ISO/IEC/IEEE standard 29148:2011 [ISO/IEC/IEEE 29148:2011], a requirements document shall be complete and consistent. Moreover, a requirements document shall support readability by offering a clear structure, reasonable scope, and traceability. Overall, a requirements document shall fulfil the following quality criteria:

- Unambiguity and consistency
- Clear structure
- Modifiability and extendibility
- Completeness
- Traceability

4.5.1 Unambiguity and Consistency

Quality of individual requirements is a prerequisite.

Requirements documents can be consistent and unambiguous only when the individual requirements are consistent and unambiguous. In addition, it must be guaranteed that individual requirements do not contradict one another. To achieve this, it is advisable to make use of conceptual models (see [chapter 6](#)). Another aspect of unambiguity pertains to the unique identification of a requirements document or a requirement among the set of all requirements documents or requirements in a development project (see section 8.5).

4.5.2 Clear Structure

Allows for selective reading

In order to guarantee that the requirements document is readable by any stakeholder, it should be appropriately comprehensive and clearly structured. Unfortunately, no clear-cut suggestions can be made regarding the appropriate comprehensiveness of a requirements document. A very comprehensive requirements document with a good structure can be just as appropriate as a less comprehensive document because a clear structure will allow the reader to skip parts that are not relevant to him. An unstructured or badly structured requirements document of the same high level of comprehensiveness would not be appropriate because the document must be read in its entirety in order for a stakeholder to be able to identify parts that are relevant to her. A good starting point is the standard structures described in section 4.3.1.

4.5.3 Modifiability and Extendibility

Content and structure should support changeability.

Requirements documents must be easy to extend. There are always requirements that are changed, altered, added, or removed as a project progresses. As a result, the structure of requirements documents should be easy to modify and extend. The requirements documents of a project should be subject to the project's version control management.

4.5.4 Completeness

Two types of completeness in requirements documents

Requirements documents must be complete, i.e., they must contain all relevant requirements (and required additional information), and each requirement must be documented completely.³ All possible inputs, influential factors, and required reactions of the system must be described for each desired system function. This comprises describing error and exception cases in particular. Also, quality requirements, such as requirements pertaining to reaction times or availability and usability of the system, must be noted.

Evidence, reference, and sources are formal necessities.

Formal factors also contribute to completeness. Graphs, diagrams, and tables should be appropriately labeled. Another important aspect is that consistent reference

and index directories must exist. Also, definitions and norm reference that denote specific terms must be included in any requirements document. The comprehensiveness of a requirements document is a challenge during requirements engineering. Often, a compromise must be found between the time resources available and the completeness of the requirements documents.

4.5.5 Traceability

Relationship to other development documents

An important quality criterion is traceability of relationships between requirements documents and other documents (e.g., business process model, test plans, or design plans). These documents could have been created in previous development phases, in subsequent development phases, or concurrently with the requirements documents. Among other things, traceability supports change management (see section 8.4).

4.6 Quality Criteria for Requirements

Quality criteria for single document requirements

Each documented requirement should fulfil the following quality criteria:

- *Agreed*: A requirement is agreed upon if it is correct and necessary in the opinion of all stakeholders.
- *Unambiguous*: [ISO/IEC/IEEE 29148:2011] A requirement that is unambiguously documented can be understood in only one way. It must not be possible to interpret the requirement in a different way. All readers of the requirement must arrive at the same understanding of the requirement.
- *Necessary*: [ISO/IEC/IEEE 29148:2011] A documented requirement must represent the facts and conditions of the system context in a way that it is valid with regard to the actualities of the system context. These actualities may be the different stakeholders' ideas, relevant standards, or interfaces to external systems.
- *Consistent*: [ISO/IEC/IEEE 29148:2011] Requirements must be consistent with regard to all other requirements, i.e., the requirements must not contradict one another, regardless of their level of detail or documentation type. In addition, a requirement must be formulated in a way that allows for consistency with itself, i.e., the requirement may not contradict itself.

- *Verifiable*: [ISO/IEC/IEEE 29148:2011] A requirement must be described in a way that allows for verification. That means that tests or measurements can be carried out that provide evidence of the functionality demanded by the requirement.
- *Feasible*: [ISO/IEC/IEEE 29148:2011] It must be possible to implement each requirement given the organizational, legal, technical, or financial constraints. This means that a member of the development team ought to be involved in rating the goals and requirements so that he can show the technical limits of the implementation of a particular requirement. In addition, the costs for the implementation must be incorporated into the rating. Occasionally, stakeholders withdraw a requirement if the costs for its realization become apparent.
- *Traceable*: [ISO/IEC/IEEE 29148:2011] A requirement is traceable if its origin as well as its realization and its relation to other documents can be retraced. This can be done by means of unique requirement identifiers. Using these unique identifiers, requirements that are derived from other requirements on a different level of the specification can be connected. For example, a system goal can be traced through all levels of abstraction, from design to implementation and test. Details can be found in section 8.4.
- *Complete*: [ISO/IEC/IEEE 29148:2011] Each individual requirement must completely describe the functionality it specifies. Requirements that are yet incomplete must be specially marked, for example by inserting “tbd” (“to be determined”) into the respective text field or by setting a corresponding status. These markings can then be systematically searched for and missing information can be amended accordingly.
- *Understandable*: Requirements must be comprehensible to each stakeholder. Therefore, the type of requirements documentation (see section 4.2) can vary significantly, depending on the development phase (and therefore, depending on the involved staff). In requirements engineering, it is important to strictly define the terms used.

Fundamental principles of understandability

Along with quality criteria for requirements, there are two fundamental rules that enhance the readability of requirements:

- *Short sentences and short paragraphs*: As human short-term memory is very limited, circumstances that belong together should be described in no more than seven sentences.
- *Formulate only one requirement per sentence*: Formulate requirements using

active voice and use only one process verb. Long, complicated interlaced sentences must be avoided.

4.7 Glossary

A frequent cause for conflicts in requirements engineering is that the people that are involved in the development process have different interpretations of terms. In order to avoid these conflicts, it is necessary that everyone who is involved in the development process shares the same understanding of the terminology used. Therefore, all relevant terms must be defined in a common glossary. A glossary is a collection of term definitions and contains the following elements:

- Context-specific technical terms
- Abbreviations and acronyms
- Everyday concepts that have a special meaning in the given context
- Synonyms, i.e., different terms with the same meaning
- Homonyms, i.e., identical terms with different meanings

Consistent definitions

By defining the meaning of terms, you can increase the understandability of requirements considerably. Misunderstandings and different interpretations of terms that might lead to conflicts can be avoided from the beginning.

Reuse of glossary entries

Often, in different projects, terms are used that are similar to one another or in fact identical. This may be the case, for example, when one system is developed for different customers but within the same domain. In this case, already existing glossary entries should be reused. It may even be feasible to define such terms in a universal, inter-project glossary. The additional effort of creating such a glossary will pay off in future projects. For certain domains, collections of term definitions already exist and are publicly accessible. These may serve as the foundation for the definition of specific glossaries. For example, in [IEEE 610.12-1990], typical terms of software engineering are defined.

Rules for Using a Glossary

Basic rules for using a glossary

Since creating a glossary is absolutely mandatory, the following must be noted:

- *The glossary must be centrally managed:* At any time, there must be only one valid glossary, which must also be centrally accessible. There must not be multiple valid glossaries.
- *Responsibility must be assigned:* One particular individual must be assigned with the task of maintaining the glossary and ensuring consistency and up-to-dateness. The necessary resources to accomplish this task must be included in the project plan.
- *The glossary must be maintained over the course of the project:* In order to ensure that the glossary is consistent and up-to-date, it must be maintained over the course of the entire project by the person that was assigned this responsibility.
- *The glossary must be commonly accessible:* The term definitions must be available for all involved personnel. This is the only way a common understanding of the terms can be ensured.
- *Using the glossary must be obligatory:* All involved personnel must be obliged to exclusively use the terms and term definitions as they have been defined in the glossary.
- *The glossary should contain the sources of the terms:* In order to be able to resolve questions and problems at any time during the course of the project, it must be possible to determine the source of a term.
- *The stakeholders should agree upon the glossary:* Only stakeholders can reliably validate the operational definitions for their respective project context. Each definition should be validated by the stakeholders or their representatives. In addition, the individual term definitions in the glossary should be explicitly approved. This approval signals that the respective term is correct and its use is obligatory.
- *The entries in the glossary should have a consistent structure:* All entries in the glossary must be structured in the same way. In order to support a consistent documentation, it is advisable to use a template for glossary definitions. In addition to the definition and the meaning of a term, the template should specify possible synonyms and homonyms.

To reduce the effort of aligning terms with one another, it is advisable to start with the creation of the glossary early on in the project.

4.8 Summary

The documentation of requirements plays a central role in requirements engineering. As the amount of requirements is often vast, it is very important to clearly structure the requirements so that personnel not involved with the project also understand them. Also, looking up and changing requirements is simplified and accelerated in this way. This makes meeting the quality criteria for requirements documents much easier. Using customized documentation structures has proven to be suitable for that purpose. These are completed by inserting project-specific requirements written in natural language in conjunction with conceptual requirements models.

1. This section can also be treated as an appendix to the document.
2. This section can also be treated as an appendix to the document.
3. Strictly speaking, this statement holds true only for the requirements document of the next system release (see section 8.5.3).

5 Documenting Requirements in Natural Language

Elicited requirements for the system to be developed are frequently documented using natural language. Natural language has the advantage that it (allegedly) does not require preparation time in order to be read and understood by stakeholders [Robertson and Robertson 2006]. In addition, natural language is universal in the sense that it can be used to describe any circumstances. However, there are some problems associated with the use of natural language for requirements documentation.

5.1 Effects of Natural Language

Subjective perception

As natural language is inherently ambiguous and statements in natural language can often be interpreted in multiple ways, it is necessary to place special emphasis on potential ambiguities in such statements to satisfy the criterion of unambiguousness. Requirements are defined and read by people with different knowledge, different social backgrounds, and different experiences. The diversity among the people involved in the development processes may lead to misunderstanding as humans interpret information differently (they form a so-called “deep structure” in their mind) and thus construe it differently as well (e.g., as a requirement). During such a process (i.e., perception and representation of information), so-called “transformational effects” occur that show different characteristics with every human but may occur in all humans [Bandler and Grinder 1975, Bandler 1994].

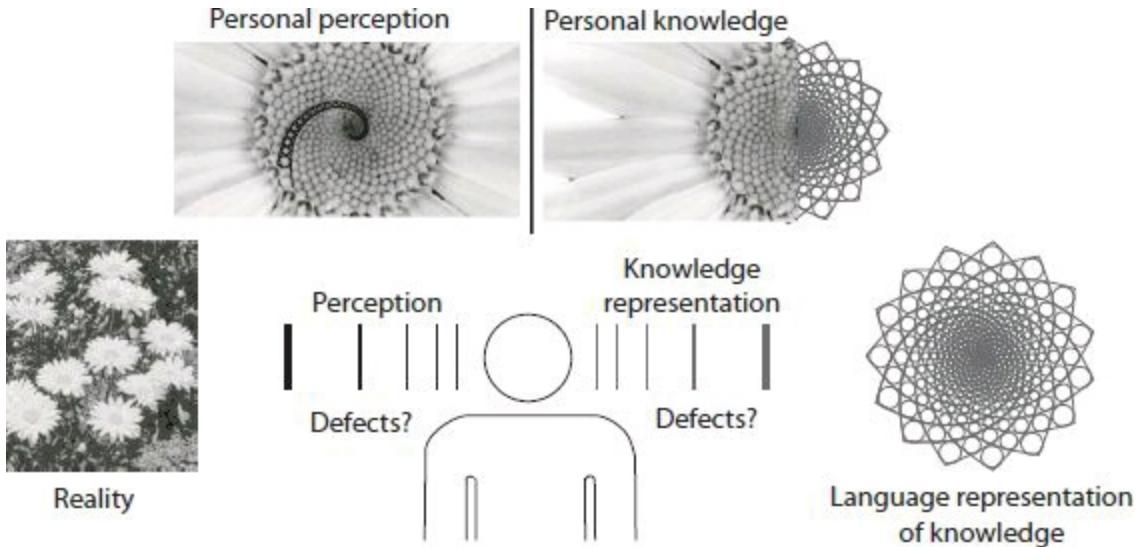


Figure 5-1 Transformational effects in perception and representation of knowledge

Transformational effects

The fact that transformational effects adhere to certain rules can be exploited by the requirements engineer to elicit the deep structure (i.e., what the author of a requirement really meant) from its surface structure (i.e., the requirements). The following list includes the five transformational processes that are most relevant for requirements engineering:

- Nominalization
- Nouns without reference index
- Universal quantifiers
- Incompletely specified conditions
- Incompletely specified process verbs

5.1.1 Nominalization

Reduction of processes

By means of nominalization, a (sometimes long-lasting) process is converted into a (singular) event. All information necessary to accurately describe the process is thereby lost. The process word *transmit* turns into the noun *transmission*. Other typical examples of nominalization are the terms *input*, *booking*, and *acceptance*.

Example 5-1: Nominalization

“In case of a system crash, a restart of the system shall be performed.”

The terms *system crash* and *restart* each describe a process that ought to be analyzed more precisely.

Define processes completely.

Per se, there are no arguments against the use of nominalized terms to describe complex processes. However, the process should be explicitly defined by the term used. The definition of a nominalized term must not allow for any leeway in the interpretation of the processes and must precisely depict the process, including any exceptions that may occur as well as all input and output parameters. It is therefore not necessary to avoid nominalizations, but they should only be used if the underlying process is completely defined. During the linguistic analysis of a text, all nominalizations ought to be examined to determine whether they have been defined in sufficient detail in another part of the requirements document and whether they are clear for all stakeholders. If this is not the case, another requirement or a glossary entry must be created.

5.1.2 Nouns without Reference Index

Nouns with missing reference

As with process verbs, nouns are frequently incompletely specified. Linguists call this a missing or inadequate index of reference. Examples of terms that contain incompletely specified nouns are *the user*, *the controller*, *the system*, *the message*, *the data*, or *the function*.

Example 5-2: Nouns without reference indices

The data shall be displayed to the user on the terminal.

The following questions arise: What data exactly? Which user exactly? Which terminal exactly? If this information is amended, the requirement might thus read as follows:

Example 5-3: Nouns with added reference indices

The system shall display the billing data to the registered user on the terminal she is logged in to.

5.1.3 Universal Quantifiers

Specify amounts and frequencies.

Universal quantifiers specify amounts or frequencies. They group a set of objects and make a statement about the behavior of this set. When using universal quantifiers, there is the risk that the specified behavior or property does not apply to all objects within the specified set. Stakeholders tend to group objects together, even though some of these objects might be special cases or exceptions, where the behavior specified does not apply to all the objects of a group.

Identify universal quantifiers.

It must be verified whether the specified behavior really applies to all objects summarized through the quantifiers. Universal quantifiers can be easily identified through trigger words such as *never*, *always*, *no*, *none*, *every*, *all*, *some*, or *nothing*.

Example 5-4: Universal quantifiers

The system shall show all data sets in every submenu.

In this case, the following question must be asked: Really in every submenu? Really all data sets?

5.1.4 Incompletely Specified Conditions

Identify and clarify condition structures.

Incompletely specified conditions are another indicator of a potential loss of information. Requirements that contain conditions specify the behavior that must occur when the condition is met. In addition, they must specify what behavior must occur if the condition is not met (the part that is often missing). Especially in complex conditional structures, decision tables can be invaluable tools to find unspecified variants of conditions or actions. Trigger words are, for instance, *if . . . then*, *in case*, *whether*, and *depending on*.

Example 5-5: Incompletely specified condition

The restaurant system shall offer all beverages to a registered guest over the age of 20 years.

At least one aspect remains unspecified in the example above: Which beverages shall be offered to a guest that is 20 years or younger? Clarifying this question may lead to extending the requirement as follows:

Example 5-6: More completely specified condition

The restaurant system shall offer

All alcohol-free beverages to any registered user younger than 21 years
All beverages including all alcoholic beverages to any user over the age of 20

5.1.5 Incompletely Specified Process Verbs

Completing process words

Some process verbs require more than one noun to be considered completely specified. The verb *transmit*, for instance, requires at least three supplements to be considered complete: *what* is being transmitted, *from where* it is being transmitted, and *to where* it is being transmitted. The feel for language (also referred to as “Sprachgefühl”) is a valuable tool to help gauge which process word must be supplemented in order to be considered complete. Similarly, adjectives and adverbs may need to be supplemented as well. While the effect is much less frequent with these types of words than with verbs, it is often hard to recognize.

Avoid passive voice.

The use of incompletely specified process words can mostly be avoided or kept to a minimum if requirements are formulated using the active voice rather than the passive voice.

Example 5-7: Requirement using the passive voice

To log a user in, the login data is entered.

Use active voice.

In this requirement using passive voice, it is unclear who enters the login data. It is also unclear where and how this is done. If this requirement is reformulated using the active voice, at least the agent or person responsible must be included.

The same requirement using active voice might be as follows:

Example 5-8: Requirement using active voice

The system must allow the user to enter his user name and password using the keyboard of the terminal.

5.2 Requirement Construction using Templates

Quality by means of requirements templates and glossaries

Requirements templates provide a simple and easily understandable approach to reduce language effects when documenting requirements. Templates support the author in achieving high quality and syntactic unambiguousness in optimal time and at low costs.

Definition 5-1: Requirements Template

A requirements template is a blueprint for the syntactic structure of individual requirements.

In order to achieve lexical clearness in the documentation as well, it is wise to use requirements templates in conjunction with project glossaries (see section 4.7).

The following is a step-by-step description of the correct application of requirements templates.

Step 1: Determine the Legal Obligation

How legally binding is a requirement?

In the beginning, you should determine the degree of legal obligation for a requirement. Usually, one distinguishes between legally obligatory requirements, urgently recommended requirements, future requirements, and desirable requirements. To achieve this within a requirement, you can use the modal verbs *shall*, *should*, *will*, and *may*. Alternatively, the legal obligation of a requirement can be documented by a specific requirements attribute.

Step 2: The Requirement Core

Determine the required process.

The core of each requirement is the functionality that it specifies (e.g., print, save, paste, or calculate). This functionality is referred to as the *process*. Processes are activities and may only be described using verbs. The process that depicts the system behavior by means of a requirement is to be described in step 2.

Since process words determine semantics, they must be defined as clearly as possible and be used as consistently as possible (see section 4.7).

Step 3: Characterize the Activity of a System

For functional requirements, the system activity can be classified as one of three

relevant types:

- *Autonomous system activity*: The system performs the process autonomously.
- *User interaction*: The system provides the process as a service for the user.
- *Interface requirement*: The system performs a process depending on a third party (e.g., another system). The system is passive and waits for an external event.

In step 3, any kind of system activity that is specified by a requirement of the system is documented using exactly one of three requirements templates. These requirements templates are described in more detail in the following sections.

After performing steps 1 through 3, the structure of the requirement has been developed (see [figure 5-2](#)). The words that are written in angle brackets must be replaced accordingly.

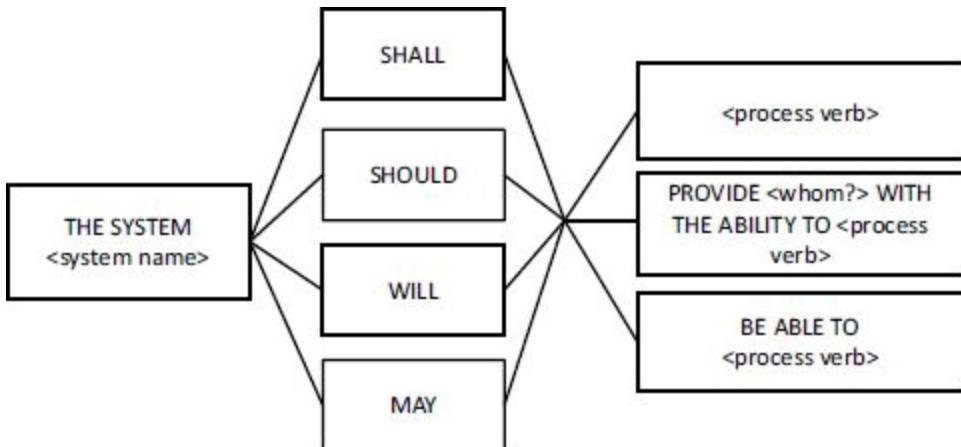


Figure 5-2 The core of a requirement and its legal obligation

Type 1: *Autonomous system activity*

The first template type is used when requirements are constructed that depict system activities that are performed autonomously. The user does not interact with the activity. We define the following requirements template:

THE SYSTEM SHALL/SHOULD/WILL/MAY <process verb>

<Process verb> depicts a process verb as described in step 2, e.g., *print* for print functionality or *calculate* for some calculation that is performed by the system.

Type 2: *User interaction*

If the system provides a functionality to a user (for example, by means of an input interface), or the system directly interacts with a user, requirements are constructed using template type 2:

THE SYSTEM SHALL/SHOULD/WILL/MAY provide <whom?> with the ability to
<process verb>

The user that interacts with the system is integrated into the requirement through <whom?>.

*Type 3:
Interface requirement*

If the system performs an activity and is dependent on neighboring systems, the third template type is to be used. Whenever messages or data are received from a neighboring system, the system must react by executing specific behavior. The following template has proven itself as well suited:

THE SYSTEM SHALL/SHOULD/WILL/MAY be able to <process verb>

Step 4: Insert Objects

Complete process verbs.

Some process verbs require one or more additional objects to be considered complete (see section 5.1.5). In step 4, potentially missing objects and supplements of objects (adverbials) are identified and added to the requirement. For instance, the requirements template for the process verb *print* is amended by the information of *what* is being printed and *where* it is printed. The amendment can be seen in [figure 5-3](#).

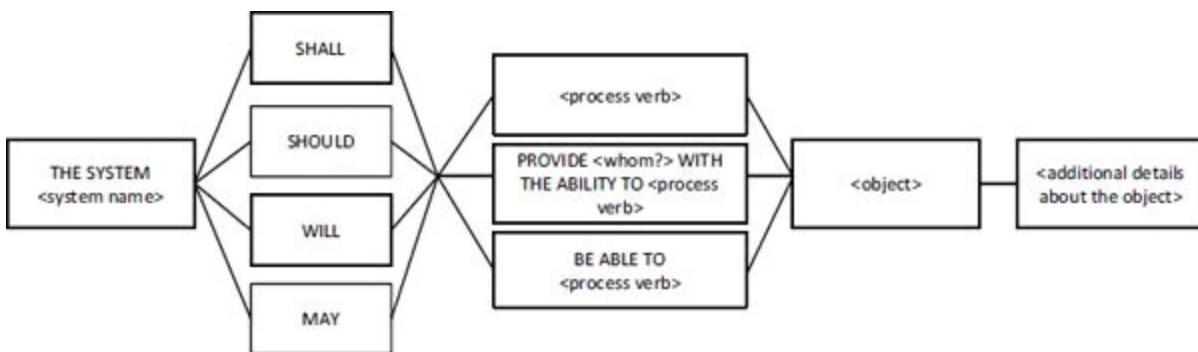


Figure 5-3 Principle of a complete requirements template without conditions

Step 5: Determine Logical and Temporal Conditions

Add conditions.

Typically, requirements do not document continuous functionalities, but functionalities that are performed or provided only under certain logical or temporal constraints. In order to easily differentiate between logical and temporal conditions, we choose the temporal conjunction *as soon as* for temporal conditions and the conditional conjunction *if* for logical conditions. The conjunction *when* makes not clear whether a temporal or a logical condition is described and should therefore be avoided. In step 5, quality requirements that describe the conditions under which a requirement is fulfilled are added to the beginning of a requirement as a subordinate clause.

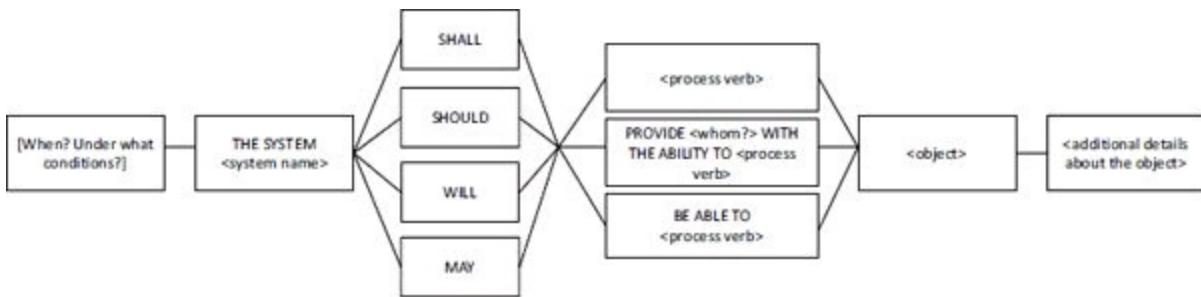


Figure 5-4 The complete requirements template with conditions

Requirements templates should be used when project members show interest in a formal development process. Style and creativity are harshly limited when requirements templates are used. Experience shows it is best not to make the use of requirements templates compulsory, but to offer training on the method and treat it as a supplemental tool.

5.3 Summary

System requirements are frequently documented using natural language. Typical advantages that arise from natural language requirements are good readability of requirements, the fact that natural language can be universally applied to document any circumstance, and the fact that no prior knowledge is necessary regarding the notation. On the other hand, there are a number of disadvantages that arise from the fact that natural language requirements are not formalized, e.g., ambiguity. Since project members interpret requirements differently due to differences in their respective knowledge, social background, and experiences, using natural language for requirements documentation often leads to misunderstanding in practice. These disadvantages can be minimized during requirements documentation—for example, by making use of requirements templates and by checking the requirements against linguistic effects.

6 Model-Based Requirements Documentation

During model-based documentation of requirements in requirements engineering, three types of requirements are documented independently and used in conjunction:

- *Goals* describe intentions of stakeholders or groups of stakeholders. Goals can potentially conflict with one another.
- *Use cases* and *scenarios* document exemplary sequences of system usage. Scenarios are grouped together in use cases.
- *System requirements* (generally referred to as requirements) describe detailed functions and qualities that the system to be developed shall implement or possess. In addition, system requirements provide complete and precise information to serve as input for subsequent development steps.

In practice, requirements are often documented using natural language. However, it can be observed that requirements are increasingly often documented using models. Requirement models are used in addition to natural language requirements documentation and partly replace requirements that would have been documented using natural language.

6.1 The Term *Model*

Models as abstracting images from reality

A model is an image that abstracts from reality or that serves as a abstracted representation of reality that is to be created. Modeling may be applied to material or immaterial objects of an existing reality or a reality to be developed. Similarly to [Stachowiak 1973], we define the term *model* as follows:

Definition 6-1: *Model*

A model is an abstract representation of an existing reality or a reality to be created.

6.1.1 Properties of Models

Every model possesses three important properties that are also the prevalent advantages of models:

- *Mapping of reality*: Every model maps certain aspects of the observed reality onto its modeling elements. Model creation can be descriptive and prescriptive in nature. In the case of descriptive model construction, the resulting model documents the existing reality. In the case of prescriptive model construction, the resulting model serves as a prototype for a fictitious reality. Depending on the perspective, models themselves can be both descriptive and prescriptive at the same time. For example, a model is descriptive with regard to the conception of the stakeholder who is constructing it and prescriptive with regard to the system to be developed.
- *Reduction of reality*: Models reduce the mapped reality. It is common to differentiate between selection and compression. During selection, only particular aspects that are part of the universe of discourse of the system are modeled. In contrast, aspects of the subject-matter of the system are summarized during compression.
- *Pragmatic property*: A model is always constructed for a special purpose and within a special context. The purpose of the model may affect the construction and the purpose-driven reduction of reality within the models. Ideally, a model contains only the information necessary for the respective purpose.

Typically, graphical models are used in requirements engineering. Their modeling elements are conceptualizations of material or immaterial objects, or people, in reality.

6.1.2 Modeling Languages

Syntax and semantics

In order to construct conceptual models, specific modeling languages are used. A modeling language is defined by its syntax and semantics:

- Syntax: The syntax of a modeling language defines the modeling elements to be used and specifies the valid combinations thereof.
- Semantics: The semantics defines the meaning of the individual modeling elements and serves therefore as a foundation for the interpretation of the models

of the respective modeling language.

Different degrees of formalization

Conceptual modeling languages can be classified as formal, informal, and semiformal, depending on the degree of formalization. The degree of formalization of a modeling language depends on the magnitude of formal definitions (e.g., mathematical calculus) that define the syntax and semantics.

6.1.3 Requirements Models

Conceptual models that document the requirements of a system are called requirements models. The Unified Modeling Language (UML) is frequently used to construct requirements models [OMG 2007]. UML has developed into the quasi-standard for model-based construction of software systems. It consists of a set of partially complementary modeling languages that are particularly used in requirements engineering to model the requirements of a system from different perspectives. Extensive examples of modeling using UML can be found in [Rupp et al. 2007], for instance.

Requirements models vs. design models

A considerable difference between the conventional use of conceptual models in system development and model usage for requirements documentation is that conventional models document solutions chosen during system development. Requirements models, on the other hand, depict specific aspects of the underlying problem.

6.1.4 Advantages of Requirements Models

Increased understandability

Research on human cognition has shown that information can be perceived and memorized faster and better when depicted graphically as opposed to making use of natural language. (e.g., [Glass and Holyoak 1986], [Kosslyn 1988], and [Mietzel 1998]. These findings can be applied to the use of requirements models in particular.

Support perspectives of documentation.

An additional advantage when using requirements models is that in contrast to

natural language, the modeling languages used have a strictly defined focus. An example is the different kinds of maps that can be drawn for a city. Depending on what aspect of the city is being mapped (modeled), different types of abstraction can be used to construct the map. For instance, a subway map will show underground subway stations and subway lines. However, the length of each connection on the map may not accurately depict the distance between the stations but may estimate the transit time instead. In contrast to a subway map of a city, a road map of a city accurately depicts the streets, paths, and locations of sights. Both models depict the same reality, but with a different focus that defines purpose-driven abstractions.

Requirements models also have the advantage that the different types of modeling elements within the same modeling language dictate the method of abstraction as well as what is being abstracted and what is not.

6.1.5 Combined Use of Models and Natural Language

Using both natural language and requirements models in combination allows the advantages of both documentation techniques to be exploited while minimizing their disadvantages. For example, natural language requirements can be summarized and their interrelations depicted using models. On the other hand, natural language can help enrich requirements models and modeling elements with additional information.

6.2 Goal Models

Many methods in requirements engineering are based on the explicit consideration of stakeholders' intentions by means of goals (e.g., [van Lamsweerde et al. 1991] and [Yu 1997]). Ordinarily, the effort required to explicitly consider goals during requirements engineering is minimal. However, the positive impact on requirements engineering—if goals are modeled—and on the quality and comprehensiveness of the requirements is very high. Goals are a stakeholder's (e.g., a person's or an organization's) description of a characteristic property of the system to be developed or the development project.

Natural-language-based and model-based documentation

Goals are very well suited to refine the vision of the system. Refining a goal is known as goal decomposition. Goals can be documented using natural language (e.g., by means of predesigned templates) or using goal models. A widely known and very common goal modeling technique is the use of AND/OR trees. By means of AND/OR

trees, hierarchical decompositions can be documented. The type of refinement relation is depicted by graphic representations of the branches. The direction of the goal decomposition is not represented through branches but through the top-down structure of the tree.

6.2.1 Goal Documentation Using AND/OR Trees

Using AND/OR trees, two types of decomposition relationships can be distinguished. [Figure 6-1](#) schematically shows these types of decomposition as well as their modeling elements.

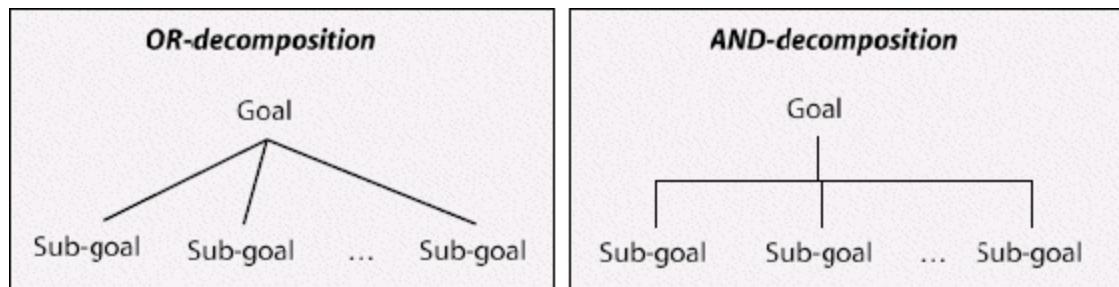


Figure 6-1 Modeling of goal decomposition using AND/OR trees

AND-decomposition vs. OR-decomposition

With regard to decomposition relations, one can differentiate between AND-decomposition and OR-decomposition. In case of AND-decomposition, every sub-goal must be fulfilled so that the super-goal (the root) is fulfilled. In contrast, in OR-decomposition, it suffices if at least one sub-goal is fulfilled so that the super-goal is met.

6.2.2 Example of AND/OR Trees

[Figure 6-2](#) shows an AND/OR tree that documents the hierarchical decomposition of the goal “Comfortable navigation to destination”.

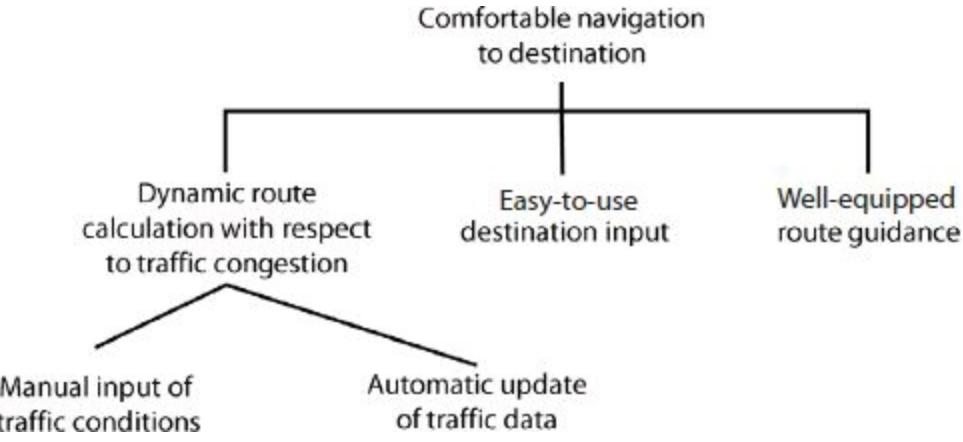


Figure 6-2 Goal model in the form of an AND/OR tree

Modeling goals with AND/OR trees

As the goal model in figure 6-2 shows, the goal “comfortable navigation to destination” is refined into the three sub-goals “dynamic route calculation with respect to traffic congestion”, “comfortable destination input”, and “comfortable route guidance” via AND-decomposition. This depicts that all three sub-goals must be met to consider the super-goal fulfilled. The sub-goal “dynamic route calculation with respect to traffic congestion” in turn is refined by the two sub-goals “manual input of traffic conditions” and “automatic update of traffic data”. The type of decomposition relation depicts that only one of the two sub-goals must be met to consider the super-goal met.

6.3 Use Cases

Use cases were first proposed in [Jacobson et al. 1992] as a method to document the functionalities of a planned or existing system on the basis of simple models. The use case approach is based on two concepts that are used in conjunction with one another:

- Use case diagrams
- Use case specifications

6.3.1 UML Use Case Diagrams

Relations between use cases

Use case diagrams in the UML [OMG 2007] (see section 4.2.3) are simple models to schematically document the functions of a system from a user’s perspective and to

document the interrelations of the functions of a system and the relations between these functions and their environment.

Modeling Elements of UML Use Case Diagrams

Figure 6-3 shows the most essential modeling elements of use case diagrams, as defined in the Unified Modeling Language (UML) [OMG 2007].

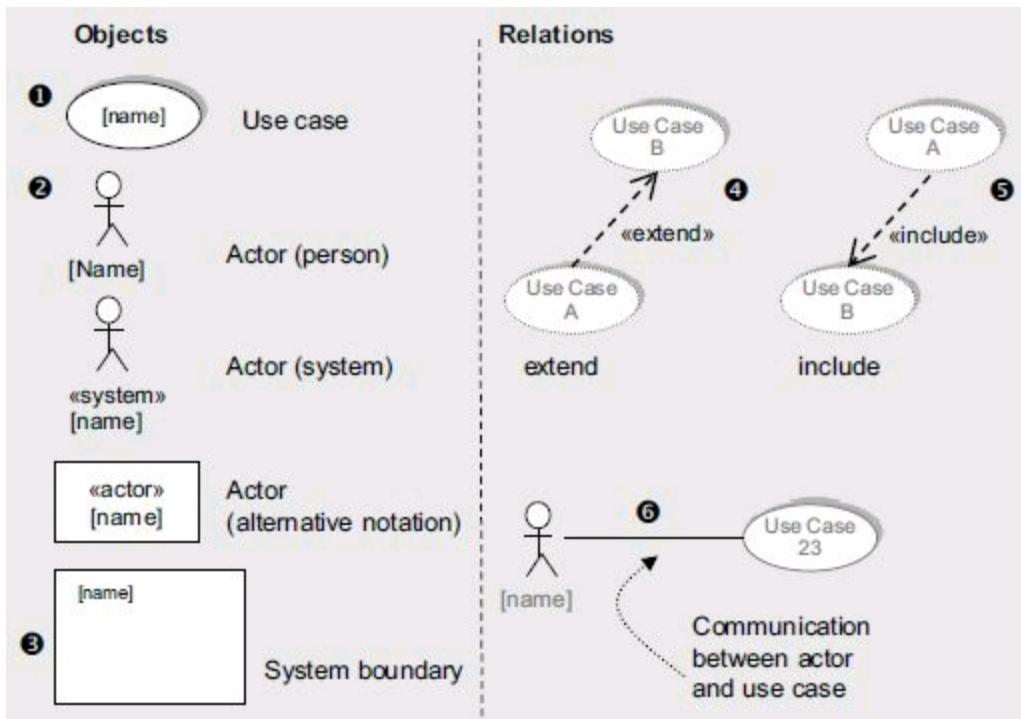


Figure 6-3 Essential modeling elements of use case diagrams

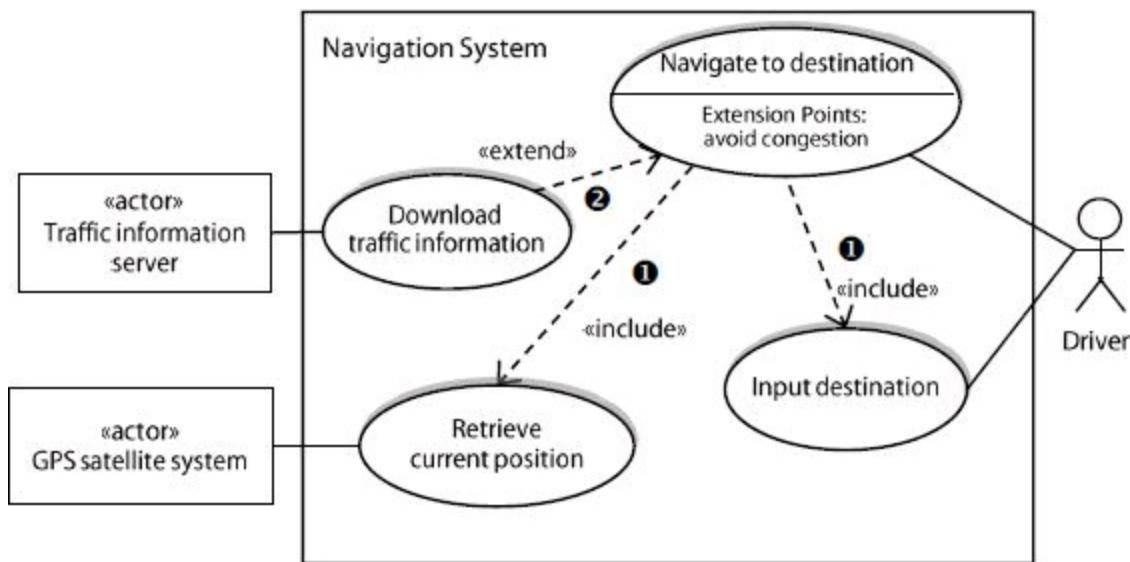
1. *Use cases*: Use cases that are defined for the system are depicted using oval shapes. These shapes contain the name of the use case. Alternatively, the name can be written beneath the use case.
2. *Actors*: Actors are outside the system boundary and represent people or systems that interact with the system modeled. Actors are depicted by a rectangle that receives the name of the actor and is tagged with the stereotype “actor”. If the actor is a person, a stick figure may be used. If the actor is a system, either a rectangle or a stick figure may be used in conjunction with the stereotype “system”.
3. *System boundaries*: System boundaries within a use case diagram separate the parts of the use case that are part of the system from the parts (people or systems) that are outside the system boundary. Optionally, the name of the system may be denoted at the system boundary in the diagram.
4. *Extend relation*: An extend relation depicts that an interaction sequence that

belongs to use case A extends some interaction sequence in use case B at a specified point. This is known as the extension point. The extension is triggered by the condition defined.

5. *Include relation*: An include relation from one use case to another use case depicts that the interaction sequence of the first use case includes the interaction sequence of the other use case.
6. *Relation between actors and use cases*: If communication between a use case and one or more actors takes place during the execution of the use case, the communication must be annotated by means of a communication relation between the respective actors and the use case.

Example of UML Use Case Diagrams

[Figure 6-4](#) shows an example of a use case diagram.



[Figure 6-4](#) An example using modeling elements of use case diagrams

The model comprises the use cases “download traffic information”, “retrieve current position”, and “input navigate to destination” elements. The relations in [figure 6-4](#) that are labeled by numbers are explained in further detail below:

Include

1. The use case “navigate to destination” is related to the use cases “input destination” and “retrieve current position” via an include relation. The relationship depicts that the interaction steps defined in the use cases “input destination” and “retrieve current position” are contained in the use case “navigate to destination”.

Extend

2. The extend relation between the use cases “download traffic information” and “navigate to destination” defines that the interaction steps defined in the use case “download traffic information” are included in the interaction steps of the use case “navigate to destination” if a certain condition, such as “avoid congestion”, is met. The extension point “avoid congestion” depicts the step in the use case “navigate to destination” at which the additional interaction steps are being executed.

Generalization

UML also provides a generalization relation between use cases or actors. In this case, the specializing use cases or actors inherit the properties of the generalizing use case or actor (e.g., [Rumbaugh et al. 2005]). For instance, the actors “service mechanic” and “customer service representative” can be generalized as the actor “employee”. The generalizing actor would carry all aspects that the actors “service mechanic” and “customer service representative” have in common (e.g., employee ID).

6.3.2 Use Case Specifications

Use case diagrams show the system’s relevant functions from a user’s perspective and specific relationships between the functions of the system or between functions of the system and aspects in the system’s context. With the exception of a use case’s name and its relationships, use cases diagrams do not document any information about the individual use cases such as the systematic interaction between a use case and an actor. This information is documented textually by means of adequate templates in conjunction with use case diagrams.

Reference templates for the documentation of use cases

Pertinent literature proposes different templates for textual specification of use cases (e.g., [Cockburn 2001]). These templates define types of information that should be documented for a use case and suggest an appropriate structure for the information. The template references therefore document experience-based knowledge regarding structured textual documentation of use cases. In order to textually specify use cases, the template in [table 6-1](#) is suitable.

Template for Textual Use Case Documentation		
Nr.	Section	Content / Explanation
1	Designation	Unique designation of the use case.
2	Name	Unique name of the use case.
3	Authors	Names of the authors that were involved in this use case description.
4	Priority	Importance of the use case according to the applied prioritization technique.
5	Criticality	Criticality of the use case, e.g., with respect to how much damage a failure of the use case may cause.
6	Source	Designation of the source from which the use case was elicited ([stakeholder document system]).
7	Person responsible	The stakeholder who is responsible for this use case.
8	Description	Brief description of the use case.
9	Trigger event	Name of the event that triggers this use case.
10	Actors	List of all actors that are involved in this use case.
11	Pre-conditions	List with all necessary constraints that must be met before the use case can begin execution.
12	Post-conditions	List of all states the system can be in immediately after the execution of the main scenario.
13	Result	Description of the results that are produced during use case execution.
14	Main scenario	Description of the main scenario of the use case.
15	Alternative scenarios	Description of the alternative scenarios of the use case or list of the trigger events of alternative scenarios. Often, post-conditions different than those described in (12) may hold.
16	Exception scenarios	Description of the exception scenarios of the use case or list of the trigger events of exception scenarios. Often, post-conditions different than those described in (12) may hold.
17	Qualities	Cross references to quality requirements.

Table 6-1 Template for textual use case documentation

Rows of a use case template

The template for the specification of use cases contains the following attributes:

- Attributes for unique identification of use cases (rows 1 and 2)
- Management attributes (rows 3 through 7)
- Attribute for the description of the use case (row 8)
- Specific use case attributes, e.g., the trigger event (row 9), actors (row 10), pre-

and post-conditions (rows 11 and 12), the result of the use case (row 13), the main scenario (row 14), alternative and exception scenarios (rows 15 and 16), and cross references to quality requirements (row 17)

Table 6-2 shows the specification of the use case “navigate to destination” by means of the reference template suggested in [table 6-1](#).

Section	Content
Designation	UC-12-37
Name	Navigate to destination
Authors	John Smith, Sandra Miller
Priority	Importance for system success: high Technological risk: high
Criticality	High
Source	C. Warner (domain expert for navigation systems)
Responsible	J. Smith
Description	The driver of the vehicle types the name of the destination. The navigation system guides the driver to the desired destination.
Trigger event	The driver wishes to navigate to his destination.
Actors	Driver, traffic information server, GPS satellite system
Pre-condition	The navigation system is activated.
Post-condition	The driver has reached his destination.
Result	Route guidance to the destination

Main scenario	<ol style="list-style-type: none"> 1. The navigation system asks for the desired destination. 2. The driver enters the desired destination. 3. The navigation system pinpoints the destination in its maps. 4. On the basis of the current position and the desired destination, the navigation system calculates a suitable route. 5. The navigation system compiles a list of waypoints. 6. The navigation system shows a map of the current position and shows the route to the next waypoint. 7. When the last waypoint is reached, the navigation system shows "destination reached" on the screen.
Alternative scenario	<p>4a. Calculation of the route must honor traffic information and avoid traffic congestions.</p> <p>4a1. The navigation system queries the server for updated traffic information.</p> <p>4a2. The navigation system calculates a route that does not contain any traffic congestions.</p>
Exception scenarios	Trigger event: The navigation system does not receive a GPS signal from the GPS satellite system.
Qualities	<p>→ QR.04 (reaction time upon user input)</p> <p>→ QR.15 (operating comfort)</p> <p>(QR = quality requirements)</p>

Table 6-2 Example of template-based documentation of a use case

6.4 Three Perspectives on the Requirements

Separately documenting the perspectives

When documenting requirements on the basis of models, one typically distinguishes three types of perspectives: data, function, and behavior (see section 4.2.1). Each perspective is documented separately, using suitable conceptual modeling languages [Davis 1993], [Pohl et al. 2005]:

- *Data perspective*: In this perspective, the structures of input and output data as well as static-structural aspects of the usage and dependency relationships of the system in the system context are documented.
- *Functional perspective*: This perspective documents which information of the system context is being manipulated by the system to be developed and which data is being transmitted to the system context by the system.
- *Behavioral perspective*: The embedding of the system in the system context is documented on the basis of states in this perspective. This is done, for instance, by documenting the reaction of the system to events within the system context,

documenting the conditions that trigger a state change, or documenting the effects that the system has on its environment.

Examples of the three perspectives

Figure 6-5 illustrates the three perspectives on functional requirements and gives an example of a suitable modeling language for each perspective that can be used to document the requirements. This way, requirements aspects that pertain to the static structure can be modeled using UML class diagrams, for instance. Requirements in the functional perspective can be modeled using UML activity diagrams and requirements in the behavioral perspective can be modeled using statecharts (see sections 6.6 and 6.7).

Perspectives are not disjoint.

Certain aspects of the models of a particular perspective can also be found in other perspectives. The three perspectives are therefore not disjoint. For example, the data, whose static structure is defined in a UML class diagram can potentially also be found in the functional perspective because it depicts the inputs and outputs of actions in a UML activity diagram. As the three perspectives are not disjoint, the models can be reciprocally checked for completeness and consistency with regard to the information that is modeled at the intersections.

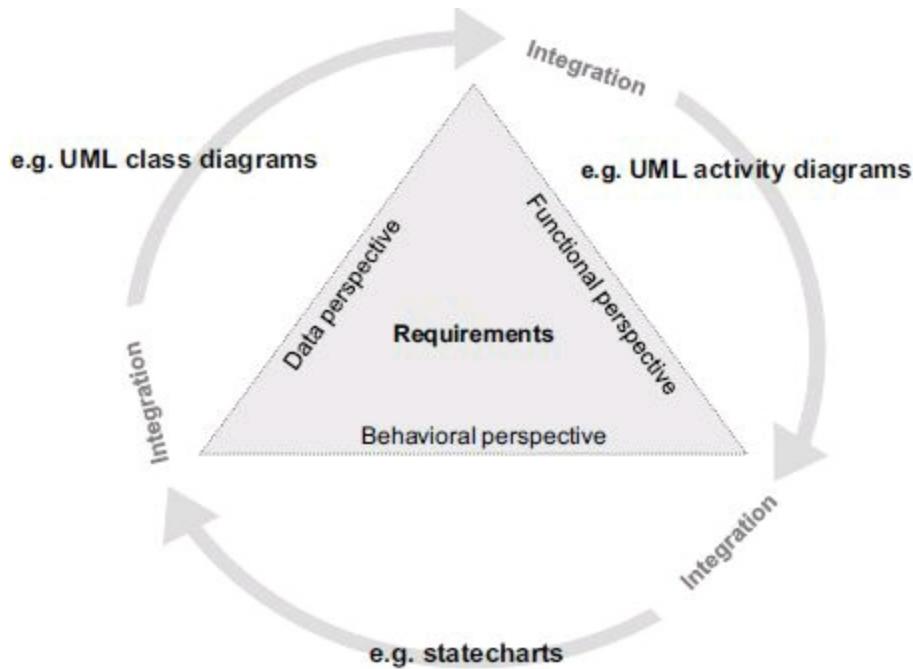


Figure 6-5 Three perspectives on requirements

6.5 Requirements Modeling in the Data Perspective

Several different modeling languages are well suited to modeling structural aspects of requirements in the data perspective. Commonly, entity-relationship models, extensions of the traditional entity-relationship model according to Chen [Chen 1976.], and, increasingly, class diagrams of the UML (e.g., [Rumbaugh et al. 2005]) are used as requirements models of the data perspective.

6.5.1 Entity-Relationship Diagrams

The traditional entity-relationship model

Traditionally, entity-relationship diagrams are used for modeling the data perspective because they display the structure of an object of an universe of discourse by means of entity types and relation types [Chen 1976].

Extensions of the entity-relationship model

A number of extensions for the entity-relationship model have been suggested. These extensions mainly concern the generalization/specialization relations, inheritance mechanisms, and roles of entities and extend the model by a (min, max)-notation for cardinalities of relations.

Modeling Elements of Entity-Relationship Diagrams

According to [Chen 1976], the modeling language used to construct entity-relationship diagrams includes the modeling elements depicted in [figure 6-6](#).

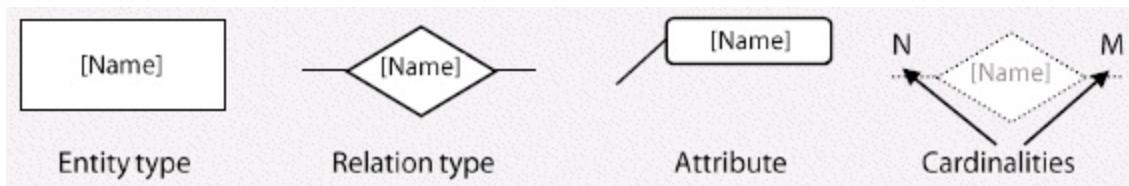


Figure 6-6 Important modeling elements of entity-relationship diagrams according to Chen

Classification: abstraction from concrete objects

Entity types define a set of entities within the universe of discourse (that is, objects with the same properties, such as people or items). An entity type (often mistakenly referred to as an entity) abstracts from the concrete characteristics of these entities and therefore classifies a set in the sense of the classification of uniform entities. For instance, the entity type “pilot” classifies all people within the universe of discourse that have the characteristic of holding a piloting license.

Abstraction from concrete relationships

A *relation type* abstracts from a concrete characteristic of a relationship and of entities that are in relation to one another. A relation type classifies the set of uniform relations between entity types within the universe of discourse. For example the relation type “executes” can be defined between the two entity types “pilot” and “flight” to represent concrete “executes”-relations between concrete pilots and concrete flights. If a concrete “is_passenger” relation is defined between a concrete passenger “John Locke”⁴ and a concrete flight with the flight number “OA 815”⁵, then this relation depicts that “John Locke” is a passenger of the flight with the flight number “OA 815”.

Properties of entity types and relation types

An *attribute* can be defined for entity types as well as relation types. An attribute defines the properties of an entity type or a relation type. Possible attributes for the entity type “passenger” could be “family name”, “given name”, “passport number”, and “reserved seat”, for instance.

Sketch level vs. instance level

An entity-relationship model documents the structure of a universe of discourse by means of entity types (i.e., classes of uniform entities) and relations (i.e., classes of uniform relationships). An entity-relationship model is defined on the modeling level and defines the set of all valid instances on the instance level.

Number of relation instances

The *cardinality* of a (binary) relation defines the number of relation instances that an entity may participate in [Elmasri and Navathe 2006]. If no cardinalities are annotated for a specific relation type, it is assumed that an arbitrary number of entities (in other words, at least zero entities) may participate in such a relation. Using cardinalities for relations therefore limits the number of instances that are principally possible in an entity-relationship diagram.

Example of an Entity-Relationship Diagram

The entity-relationship model shown in [figure 6-7](#) shows four entity types (i.e., classes of entities) and three relation types (i.e., classes of relationships). The individual entity types possess attributes that describe specific properties of the associated entities. For example, the entity type “traffic jam information” has the attributes “road”, “start”, and “length”, which depict the road on which a traffic jam is currently

present, the GPS coordinates of the starting point of the jam, and the length of the jam. The relation type “queries” between the entity types “navigation device” and “traffic jam information” means that on the instance level, a relationship between a concrete navigation device and the information on zero or more concrete traffic jams exists. The cardinalities of the entity types with regard to the relation type “queries” means that a concrete navigation device can query information on an arbitrary amount (“N”) of traffic jams. In the other direction, any traffic jam information can be queried by an arbitrary number (“M”) of navigation devices.

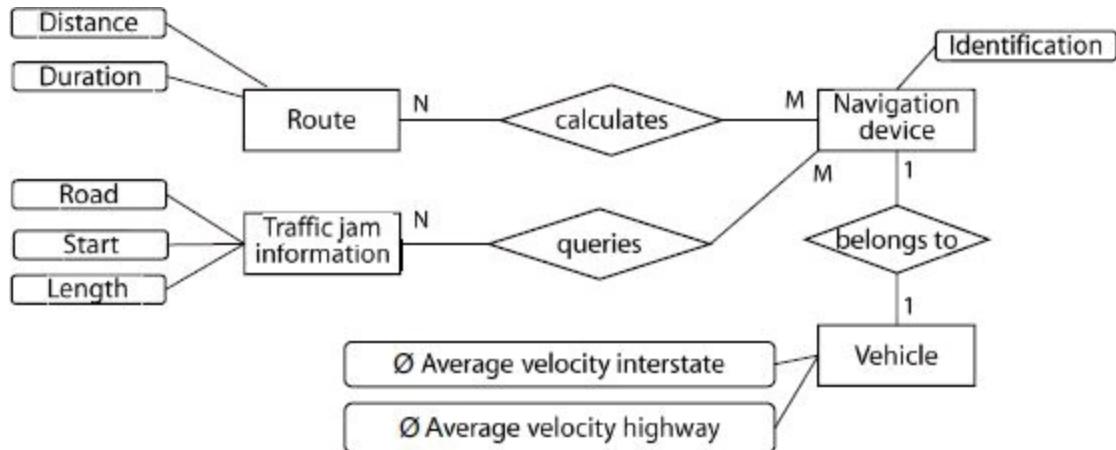


Figure 6-7 Entity-relationship diagram (data model) according to Chen

6.5.2 UML Class Diagrams

Static perspective: data/structure

Class diagrams of the UML can be used to model the data perspective of requirements of a system to be developed as well. A class diagram consists of a set of classes and associations between classes. Classes and associations in UML class diagrams are similar to entity types and relation types in entity-relationship diagrams. Class models possess additional modeling elements (e.g., that allow for the specification of valid operations on the instances of a class) and thus have a greater power of description.⁶

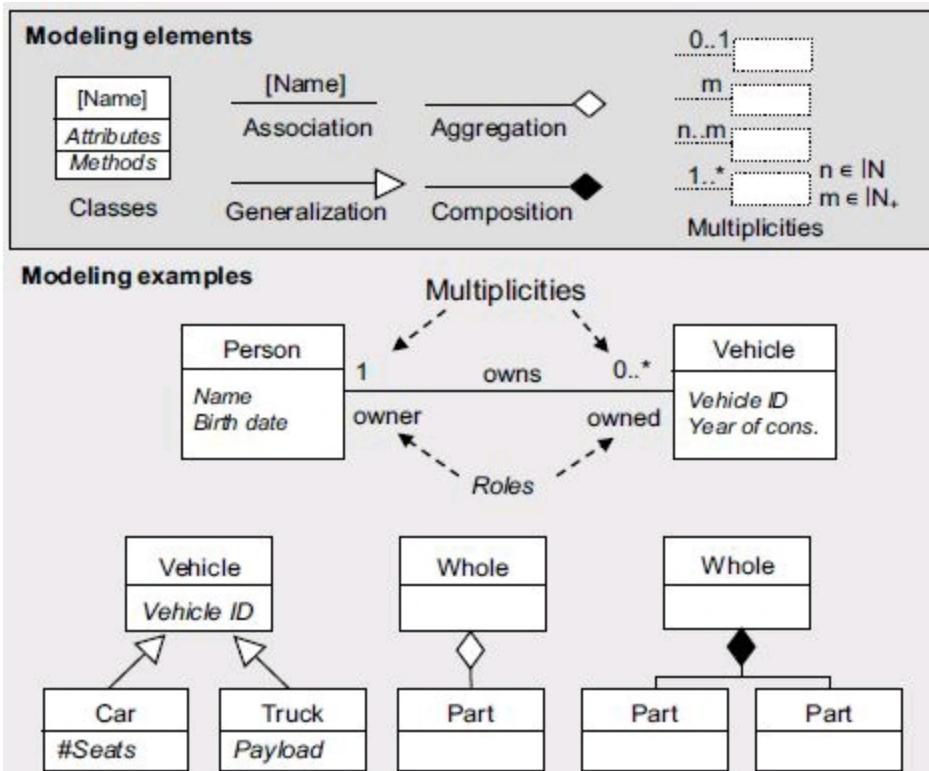


Figure 6-8 Important modeling elements of class diagrams of UML

Modeling Elements of Class Diagrams

Figure 6-8 shows important modeling elements of class diagrams of the UML as well as a number of modeling examples.

Classes

A *class* is depicted by a rectangle that is separated into sections (also called *compartments*). In the upper section, attributes are depicted that are described in more detail by the instances of the class. In the lower sections, all operations that can be performed on the instances of the class are listed. Depending on the modeling goal, i.e., depending on the purpose of the model, the compartments for attributes and/or methods can also be hidden or left out entirely.

Associations, multiplicities, and roles

Associations between classes are depicted by edges. Associations can optionally be given a name. In addition, *multiplicities* can be annotated at each end of an association. Multiplicities are statements about the instance level of a class and depict how many instances of a class may be associated in a particular way with a defined number of instances of another class. By annotating optional *roles* at one or both ends of an association, the meaning of the instances of a class with regard to the association

can be further refined.

Aggregation and composition

Aggregations and *compositions* are specific types of associations. Both describe a relationship between a whole and its constituents. A composition documents a stronger binding than an aggregation in that a constituent in a composition cannot exist without its whole. In class models of the UML, an aggregation is depicted as an empty diamond and a composition is depicted as a filled diamond.

Generalization

Moreover, in class diagrams, *generalizations* between classes can be documented. A generalization between classes of the UML is a relationship between a more specific class (the sub-type) and a more general class (the super-type). The sub-type in a generalization relation inherits all properties of the super-type and can adapt and/or extend these.

Example of a UML Class Diagram

The class diagram in [figure 6-9](#) comprises six classes that all have respective attributes. The associations between the classes are depicted by means of edges. For example, an association with the name “calculates” exists between the class “navigation device” and the class “route”. Taking into account the multiplicities, this association depicts that a navigation system can calculate an arbitrary amount (at least zero, as depicted by an asterisk *) of routes. In return, every route can be calculated by an arbitrary amount (*) of navigation devices. A route is an aggregation of at least one, but arbitrarily many (1..*) road segments, and every road segment belongs to an arbitrary amount (*) of routes. A road segment is defined by a road name as well as start and end points. [Figure 6-9](#) also shows that “navigation device w/ congestion avoiding” is a specialization of the generic type “navigation device”. The sub-type “navigation device w/ congestion avoiding” inherits the properties (in this case, the attribute “identification”) from its super-type “navigation device” and extends the set of attributes by an attribute that specifies the threshold length of a traffic congestion, which triggers a route recalculation.

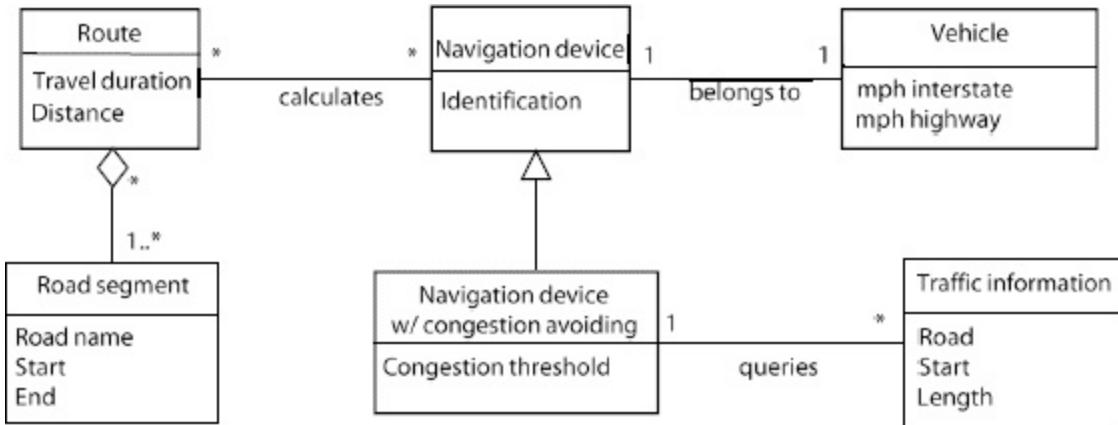


Figure 6-9 Class diagram in UML notation

6.6 Requirements Modeling in the Functional Perspective

The functional perspective of requirements deals with the transformation of input data received from the environment into output data released into the environment of the system. There are a number of different model-based approaches that can be used to model the functional perspective of requirements. The majority of these techniques is based on the structured system analysis approaches of the 1970s and 1980s, such as the structured analysis [DeMarco 1978, Weinberg 1978] or the essential system analysis [McMenamin and Palmer 1988].

6.6.1 Data Flow Diagrams

At the center of attention of modeling requirements from a functional perspective are diagrams that model the functionality of the respective system by means of processes (functions), data stores, sources, and sinks in the system environment as well as data flow. A commonly used type of functional models are data flow diagrams, as suggested in the structured analysis according to [DeMarco 1978]. Data flow models allow modeling the system on different levels of abstraction.

Modeling Elements of Data Flow Diagrams

Figure 6-10 shows the modeling elements in data flow diagrams in the notation suggested by [DeMarco 1978].

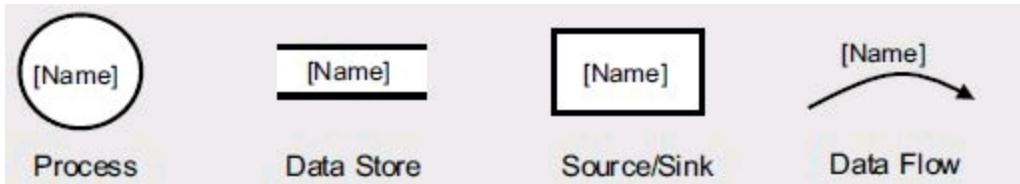


Figure 6-10 Important modeling elements of data flow diagrams according to DeMarco

Data manipulation

Processes depict the functions of a particular system necessary to transform the data that flows into the system (information flow).⁷ A process consumes the input data, processes this data, and outputs the result of the processing in the form of output data. How the data is transformed is not depicted in data flow diagrams.

Resting data

Data stores are abstract concepts designed to depict persistent data. Processes can access data in a data store in a read and write manner so that the processes may access necessary input data or persistently store output data.

Objects in the system environment

Sources/sinks describe objects (like people, groups of people, departments, organizations, or systems) in the environment of the system that exchange data with the system. Sources/sinks are aspects of the system environment and cannot be altered during system development (see section 2.1). Sources are aspects of the system environment that deliver data to the system, while sinks receive data from the system.

Flowing data

A *data flow* describes data that is transported between processes, data stores, and sources/sinks [Yourdon 1989]. In requirements models, a data flow can model both the transport of material and immaterial objects, e.g., information flow or material flow. Typically, only the most important data flows are modeled in data flow diagrams. Data flows that are not relevant for the requirements of the system can be neglected.

Example of a Data Flow Diagram

System interfaces

Figure 6-11 shows a simplified data flow diagram of a navigation system in the notation suggested by DeMarco. The interfaces of the system to the context are defined by the data flows to the sources “GPS satellite system” and “traffic information

server” as well as to the sink “driver”.

Process “calculate route”

The functionality of the navigation system is separated into three distinct processes. Process one, named “calculate route”, receives up-to-date traffic information via its interface to the source “traffic information server” as well as data about the current location via its interface to the source “GPS satellite system”. In addition, the process “calculate route” is provided with the desired destination by the driver of the vehicle. The calculated route is stored in the data store “route data”.

Process “determine next waypoint”

Process two—“determine next waypoint”—accesses the data store and retrieves data concerning the current route. The process determines the next waypoint and outputs this information.

Process “recalculate route”

Process three—“recalculate route”—plots a new course to the destination. In order to do so, it gathers traffic information from the source “traffic information server” and, potentially, information about the current location. The newly calculated course is stored in the data store “route data”.

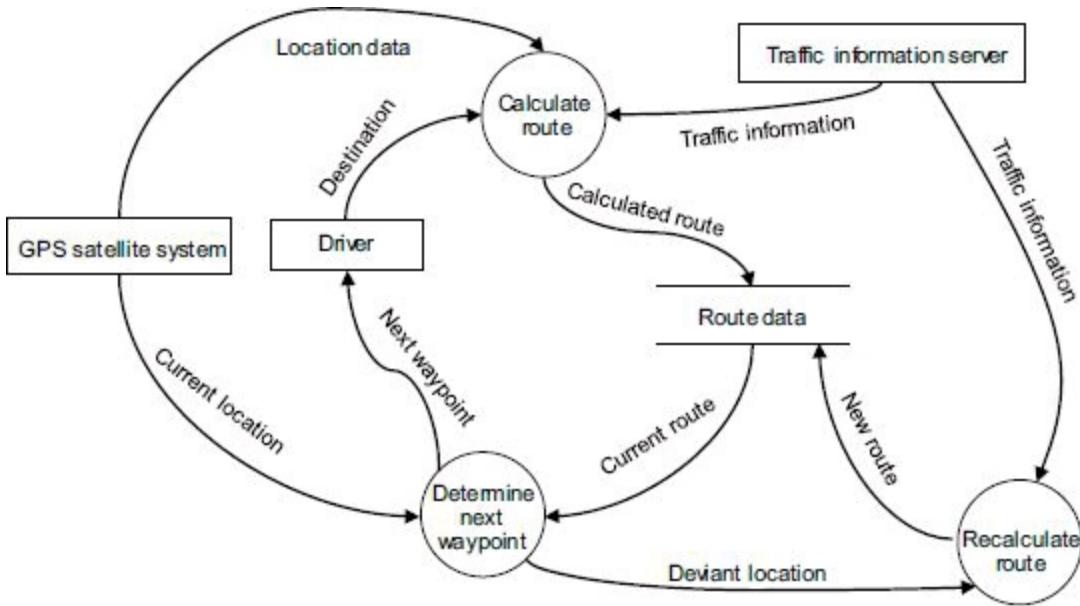


Figure 6-11 Data flow diagram in the notation suggested by DeMarco

6.6.2 Models of the Functional Perspective and Control Flow

In data flow diagrams, it cannot be seen which conditions trigger which processes.

Data flow diagrams merely depict data dependencies of the processes in a system and document necessary input and generated output data. Approaches used in structured system analysis, however, often offer complementary behavioral descriptions and control flow descriptions. This is achieved either by using distinct documentation forms, such as mini-specifications in structured analysis, or by means of implicit language extensions of data flow models. Language extensions offer the ability to model additional aspects, e.g., the control flow between functions, as done in SA/RT [Ward and Mellor 1985, Hatley and Pirbhai 1988].

6.6.3 UML Activity Diagrams

UML activity diagrams are well suited to model action sequences [OMG 2007]. Along with activity diagrams in UML, event-driven process chains (EPC) can be used to model sequences of activities [Keller et al. 1992], especially in information system development. UML activity diagrams depict the control flow between activities or actions. In case of a sequential progression of actions, a subsequent action is executed once every precedent action terminates. [Figure 6-12](#) shows important modeling elements of activity diagrams in UML [OMG 2007].

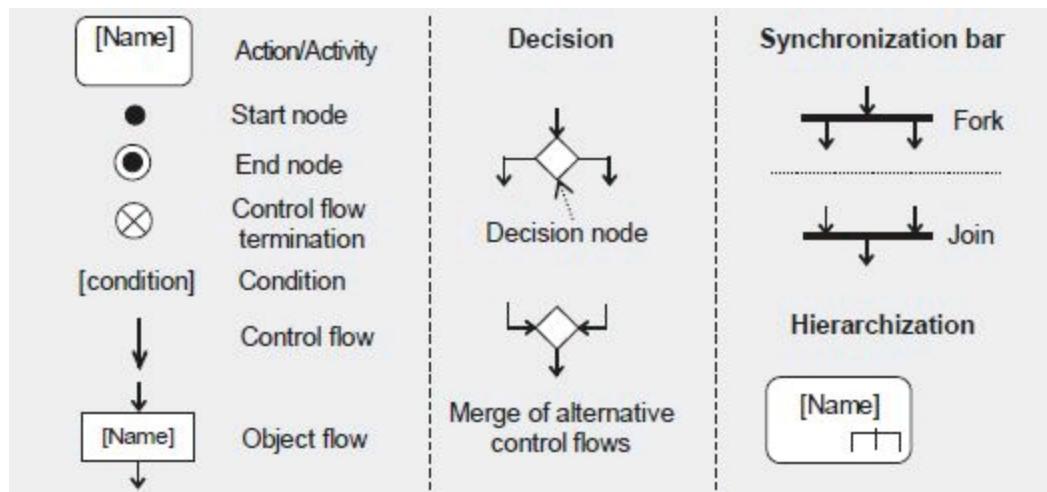


Figure 6-12 Modeling elements of activity diagrams of the UML

Action nodes

Activity diagrams are control flow graphs that consist of action nodes and the control flow between these action nodes (i.e., the arrows in the control flow graph depicting transitions). Action nodes execute an action. The start and end nodes in activity diagrams have defined semantics. The start node represents an event that initiates the execution of the activity diagram. End nodes are special nodes that represent the termination of the activity diagram.

Control flows, object flows, responsibilities

Depicting alternative control flows in activity diagrams can be achieved through the use of decision nodes. At decision nodes, conditions that trigger alternative control flows are annotated. In addition, synchronization bars allow for concurrent execution of control flows. A special type of control flows are object flows. By making use of activity partitions (swimlanes), different activities can be documented as the responsibility of specific actors.

Sequence Modeling using UML Activity Diagrams

The activity diagram in [figure 6-13](#) documents the process “navigate to destination”. Input and output data can be documented by modeling additional object flows along the edges. The data and object flows are special types of control flows of the activity diagram. Every action is executed if and only if previous actions have been carried out and all incoming object flows are available. The action diagram in [figure 6-13](#) also shows object flows that are documented in addition to the actions and control flows.

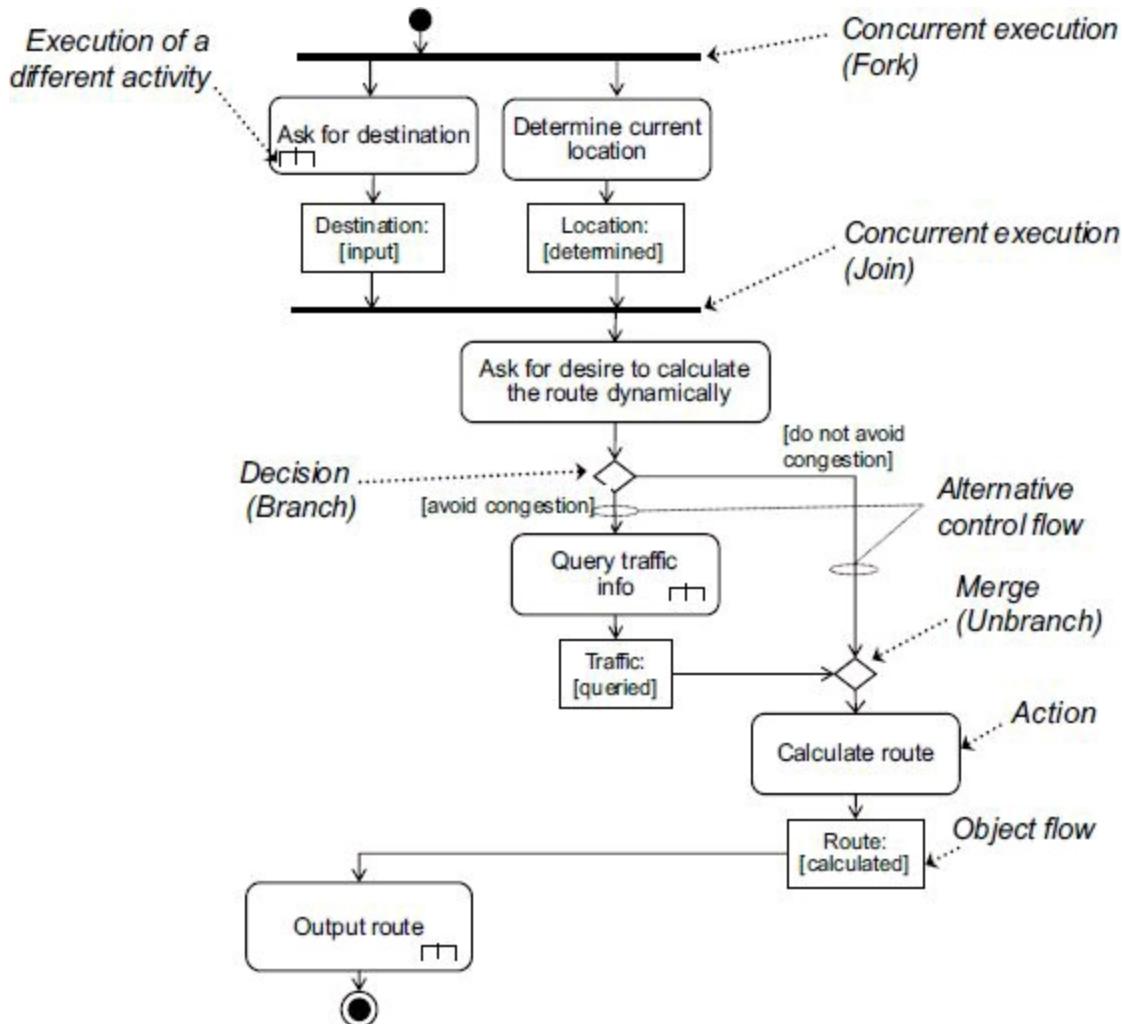


Figure 6-13 Activity diagram in UML notation

The activity diagram above documents the sequence of actions necessary for a navigation device to calculate a route. The model documents that initially the desired destination is asked for and that the current location is determined. These two actions happen concurrently, independent from one another. The input destination (object flow: object → destination; state → input) and the determined location (object flow: object → location; state → determined) are relayed. If the driver has opted to automatically circumvent traffic congestions, the system queries for up-to-date traffic information. Once the updated traffic information is received or if the driver has not selected to circumvent traffic jams, the system calculates a route to the destination. The calculated route is output to the driver.

Modeling sequences of a use case

Activity diagrams are well suited to document the relationships and execution conditions of main, alternative, and exception scenarios. Decision nodes represent branches in the control flow between the main scenario and alternative and exception scenarios.

Control Flow of Main and Alternative Scenarios

The activity diagram in [figure 6-14](#) shows the control flow of the main and alternative scenario of the use case “navigate to destination” as documented in [table 6-2](#). Alternative control flow branches begin at the decision nodes that document the respective alternative- and exception scenarios to a particular main scenario.

Main and alternative scenarios

The activity diagram shows that initially, the action “start navigation” is executed. After that, the actions “input destination” and “determine GPS coordinates” are executed concurrently and independent from one another. Once both actions have been executed, the system asks the driver if he wishes the route to be calculated dynamically (action “ask for desire to calculate the route dynamically”). If the driver does not request the route to be calculated dynamically (selection “do not avoid congestions”), no specific action is executed (see [table 6-1](#) → main scenario). If the driver selects dynamic route calculation (selection “avoid congestions”), updated traffic information is determined (action “query traffic info”, see [table 6-1](#) → alternative scenario). After that, the route is calculated (action “calculate route”) and output to the driver (action “output route”).

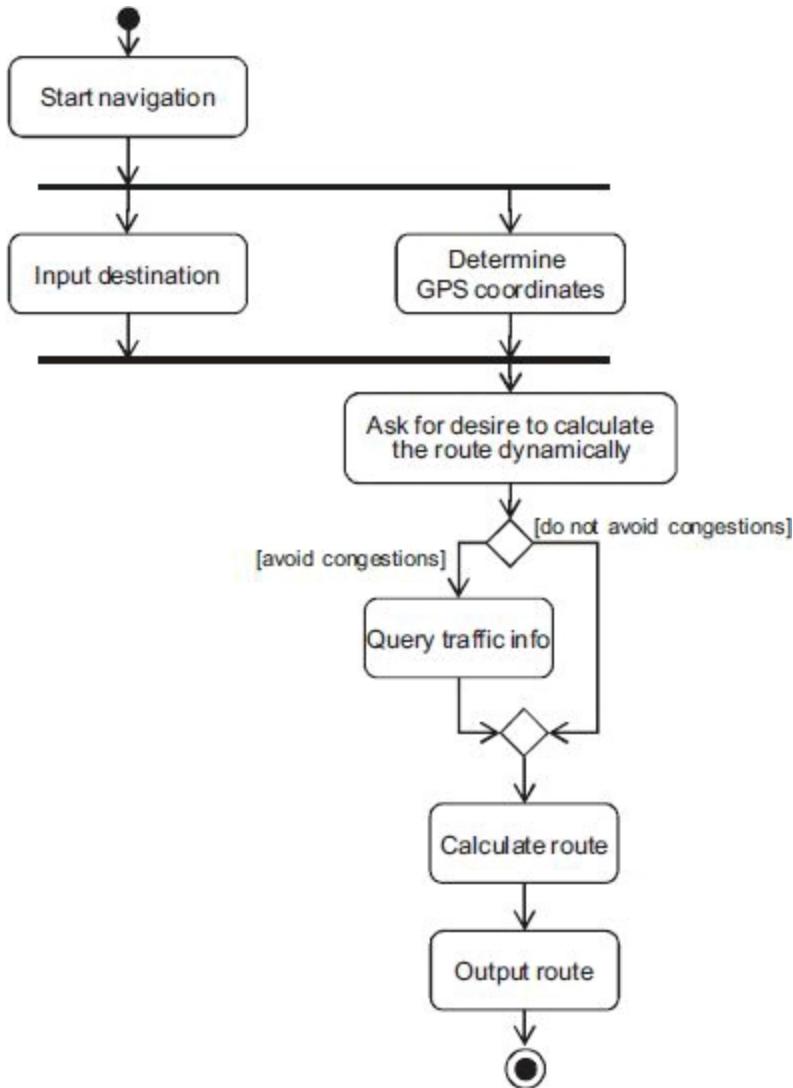


Figure 6-14 Documentation of the control flow of scenarios using UML activity diagrams

6.7 Requirements Modeling in the Behavioral Perspective

Finite-state automata

To model the dynamic behavior of a system, modeling approaches based on automata theory are typically employed. The definition of a finite-state automaton comprises a set of states and a set of transitions that, dependent on the current state of the automaton, are performed given some event.

Mealy and Moore automata

In the scope of system modeling, extensions of finite-state automata are frequently used that are based on the concepts of so-called Mealy [Mealy 1955] and Moore automata [Moore 1956], respectively. In Mealy automata, the output of an automaton

depends on the current state of the automaton as well as on the input. In contrast, in Moore automata, the output merely depends on the current state.

6.7.1 Statecharts

Statecharts = state machines + hierarchization + conditions + concurrency

Due to challenges that arise when using finite state automata in practice (such as missing support for abstraction), the automata concept has been developed into a technique of modeling the reactive behavior of a system. A widely applied technique to model the behavior of a system is the use of statecharts [Harel 1987]. Statecharts are a type of automata that is based on finite-state automata but are extended to support hierarchization of states to document conditions of state transitions and to model concurrent behavior. [Figure 6-15](#) shows the modeling elements of statecharts in the notation suggested by Harel [Harel 1987].

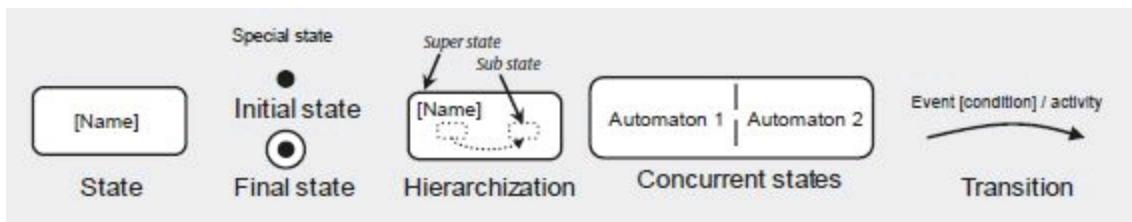


Figure 6-15 Modeling elements of statecharts

State

A state defines a period of time in which the system shows a specific behavior and waits for a particular event to occur in order to perform a defined transition.

Transition with condition and activity

A transition is triggered by a particular event once it occurs in a specific state. A transition describes the change from one state to the next. The change of states can additionally be dependent on some condition. The system can perform particular activities if it is in a particular state (typical for Moore automata) or if it performs a transition to another state (typical for Mealy automata). These activities can be directed toward the system itself or the environment of the system.

Hierarchization and abstraction

Statecharts allow for the hierarchical refinement of states that in turn represent automata. The initial state is referred to as super state and is defined by a number of

refining states. Hierarchization allows abstracting from the irrelevant details of a state by—depending on the purpose of the model—only regarding and/or modeling the super state rather than the entire sub automaton that defines the super state. The detailed behavior of the system can, if necessary, be refined by defining the respective partial automata.

Concurrency

Along with hierarchical decomposition of a state into refining automata, a state can be decomposed into several concurrent automata. The concurrent automata can be synchronized by means of transition conditions (e.g., “if automaton A is in state 4”). [Figure 6-16](#) shows a behavior model for a navigation device of a vehicle by means of a statechart. The navigation device is initially in the state “navigation device inactive”.

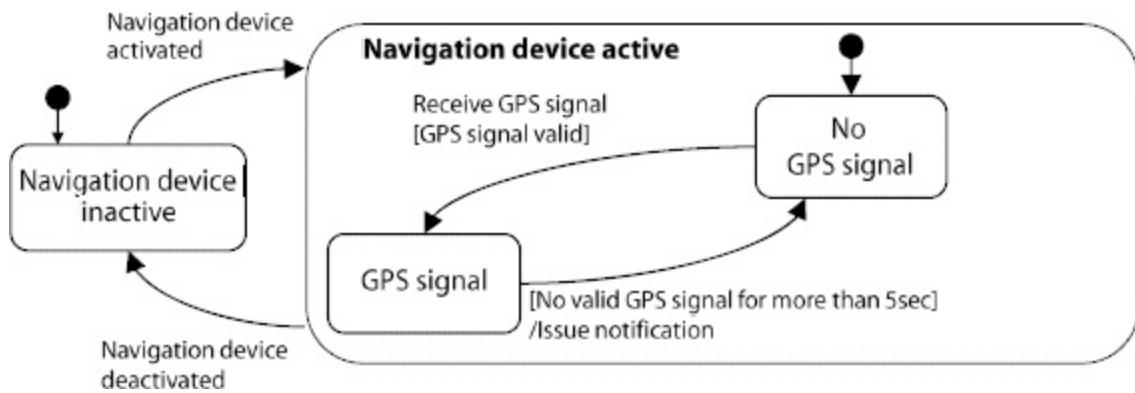


Figure 6-16 Simplified statechart of a vehicular navigation device

Transition into super state

By turning on the navigation device (event: “navigation device activated”), the system transitions into the super state “navigation device active” (more precisely, the system transitions into the initial state “no GPS signal” of the super state “navigation device active”). The super state “navigation device active” is refined by a partial automaton that consists of two states. For example, if a GPS signal is received in the state “navigation device active: no GPS signal”⁸, the system transitions into the state “navigation device active: GPS signal” and issues a notification. If the device is deactivated while in the state “navigation device active” (event: “navigation device deactivated”), the system transitions into the state “navigation device inactive”.

6.7.2 UML State Diagrams

Modeling reactive behavior of a system using UML

In order to model reactive system behavior, Unified Modeling Language (UML) [OMG 2007] offers so-called state machines that are essentially based on statecharts. [Figure 6-17](#) shows the most important modeling elements of UML state diagrams. The notation of the modeling elements of UML state diagrams has largely been adopted from statecharts. However, UML 2 extends the modeling elements of statecharts, e.g., by the ability to define explicit entry and exit points of hierarchical states [OMG 2007].

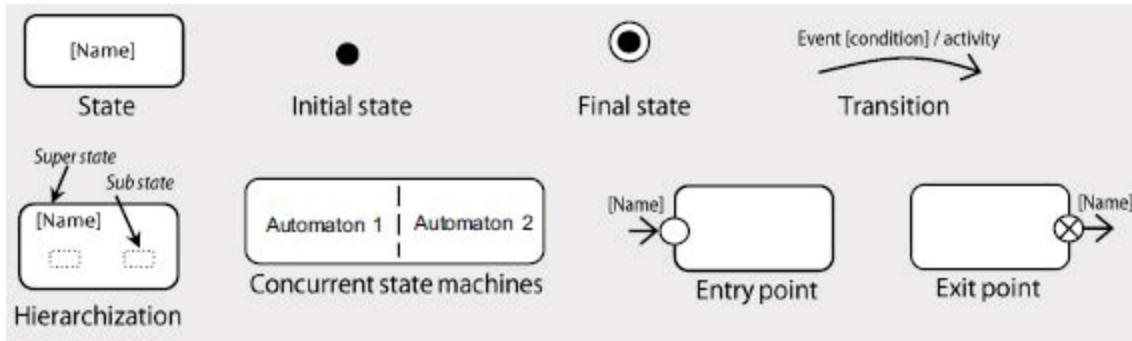


Figure 6-17 Modeling elements of state machines as defined by the UML 2

States and transitions

Just as in statecharts, a state defines a period of time in which a system shows a particular behavior and waits for a particular event to occur. A transition is triggered by an event that occurs in a particular state and describes the change from one state to the next. A transition can be dependent on a condition. In addition, the system can perform actions that are directed toward the system or its environment.

Hierarchization and concurrency

Depending on the purpose of the model, state machines allow hierarchically combining states into super states, thereby abstracting from the potentially very complex behavior of these states. Aside from hierarchically decomposing states by means of partial automata, a state can be decomposed into several concurrent state machines. Just as in statecharts, synchronization of concurrent state machines can be achieved using conditions.

Encapsulation of internal states using entry and exit points

UML 2 defines entry points and exit points as an extension of statecharts that allow for additional hierarchization of states. An exit point is an externally visible pseudo-state that is immediately associated with an internal state. An exit point is an externally visible pseudo-state that has its origin in an internal state. A super state within a state machine can have arbitrarily many entry and exit points that can be identified by a name [Rumbaugh et al. 2005].

Figure 6-18 shows a state diagram of UML that possesses two explicitly defined entry points (“enter new destination” and “last destination”) as well as one exit point (“navigation successful”) along with the modeling elements introduced in section 6.7.1.

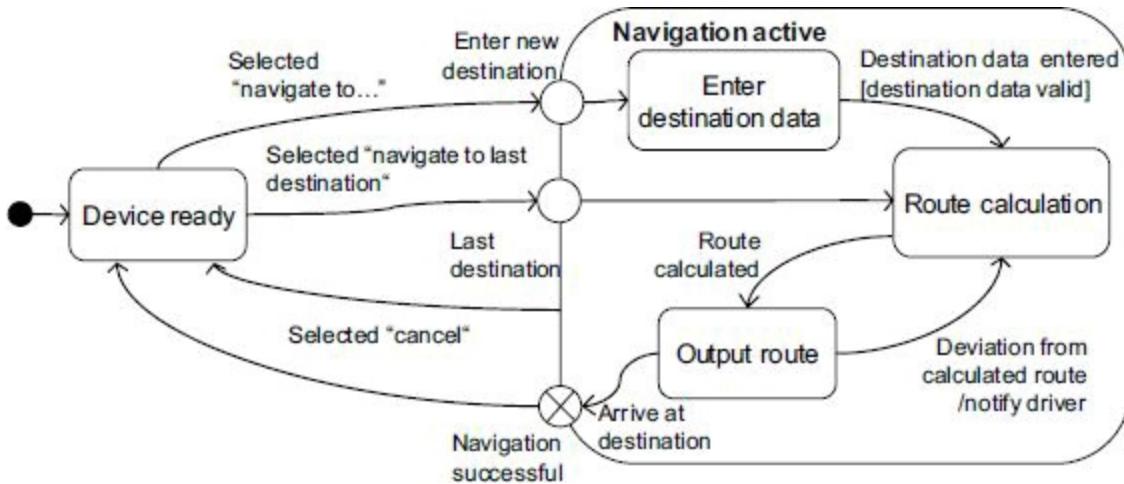


Figure 6-18 State diagram in UML 2 notation

The state diagram in figure 6-18 documents the reactive behavior of a navigation device. Initially, the system is in the state “device ready”. By selecting “navigate to . . .”, the system changes into the super state “navigation active” and, within the super state, into the sub state “enter destination data” by making use of the entry point “enter new destination”. Alternatively, the system changes from the state “device ready” into the internal state “route calculation” of the super state “navigation active” by making use of the entry point “last destination” as soon as the event “navigate to last destination” occurs. Once the system is in the state “navigation active: enter destination data”, the system transitions into the state “navigation active: route calculation” if the condition that the destination data is valid has been met.

Once the route is calculated in the state “navigation active: route calculation”, the system transitions into the state “navigation active: output route”. If a deviation from the route is detected (event: “deviation from calculated route”) in the state “navigation active: route calculation”, the driver is notified (activity: “notify driver”). From the state “navigation active”, the system transitions into the state “device ready” once the event “cancel” occurs. If the system is in the state “navigation active: route calculation” and the destination is reached, the system exits the super state “navigation active” via the exit point “navigation successful” to transition into the state “device ready”.

6.8 Summary

Along with using natural language to document requirements, requirements can be documented by means of models. Typically, natural language requirements and requirements models are frequently employed in conjunction so that the advantages of both forms of documentation can be exploited.

Model-based documentation of requirements has, among other things, the advantage that graphical (imagelike) descriptions of circumstances can be understood faster and better than natural-language descriptions. Among the models that are frequently used in requirements engineering are goal models (e.g., in the form of AND/OR trees) and use case diagrams as well as conceptual models to document requirements from three perspectives: data, functional, and behavioral. For each of these three perspectives, there are suitable conceptual modeling languages that provide purpose-specific means to document the information depicted in each respective perspective.

4. More precisely, there is an entity that is an instance of the entity type “passenger” that possesses a unique identity and has the attribute value “John Locke” for the attribute “name”.
5. More precisely, there is an entity that is an instance of the entity type “flight” that possesses a unique identity and has the attribute value “OA 815” for the attribute “flight number”.
6. A comprehensive overview of the different modelling elements of the UML can be found e.g., in [OMG 2007].
7. In the structured analysis, the flow of data, information, documents, or material is considered a data flow.
8. For unique identification, a state that is part of a super state is referenced by “super state: state”. The state “no GPS signal” in the super state “navigation device active” is therefore referenced as “navigation device active: no GPS signal”.

7 Requirements Validation and Negotiation

Validation and negotiation during requirements engineering is meant to ensure that the documented requirements meet the predetermined quality criteria, such as correctness and agreement (see section 4.6). The introduced principles and techniques can be used to validate and negotiate individual requirements or entire requirements documents.

7.1 Fundamentals of Requirements Validation

During the requirements engineering activity, it is necessary to review the quality of the requirements developed. Among others, the requirements are presented to the stakeholders with the goal to identify deviations between the requirements defined and the stakeholders' actual wishes and needs.

Approving requirements

During requirements validation, the decision of whether a requirement possesses the necessary level of quality is made (see [chapter 4](#)) and whether the requirement can be approved to be used for further development activities (such as design, implementation, and testing). This decision should be made on the basis of predefined acceptance criteria.

Goal of validation

The goal of requirements validation is therefore to discover errors in the documented requirements. Typical examples of errors in requirements are ambiguity, incompleteness, and contradictions (see section 7.3).

Error proliferation

Requirements documents are reference documents for all further development activities. Therefore, errors negatively affect all further development activities. A requirements error that is discovered when the system is already deployed and operating requires all artifacts affected by the error to be revised, such as source code, test artifacts, and architectural descriptions. Correcting errors in requirements

once the system is in operation therefore entails significant costs.

Legal risks

A contract between client and contractor is often based on requirements documents. Critical errors in requirements can lead to the fact that contractual agreements cannot be met, e.g., scope of supply and services, expected quality, or completion deadlines.

7.2 Fundamentals of Requirements Negotiation

Contradictory requirements cause conflicts.

If there is no consent among the stakeholders regarding the requirements and thus the requirements cannot be implemented collectively in the system, a conflict arises between the contradictory requirements as well as between the stakeholders that demand contradictory requirements. For example, one stakeholder could demand the system to shut down in case of a failure, whereas another stakeholder could require the system to restart.

Risks and opportunities of conflicts

The acceptance of a system is threatened by unresolved conflicts because unresolved conflicts cause the requirements of at least one group of stakeholders to not be implemented. In the worst case, a conflict causes stakeholder support to cease, causing the development project to fail (cf. [Easterbrook 1994]). Other than posing risks, conflicts can also be an opportunity for requirements engineering because conflicts between stakeholders require a solution that can potentially help discover new ideas for development and can illustrate different options (cf. [Gause and Weinberg 1989]). Therefore, treating and resolving conflicts openly during requirements engineering can increase acceptance.

Goal of requirements negotiation

The goal of negotiation is to gain a common and agreed-upon understanding of the requirements of the system to be developed among all relevant stakeholders.

Reducing costs and risks in late phases

Requirements validation and negotiation is an activity that must be performed (to a varying degree of intensity) throughout the entirety of requirements engineering. The validation and negotiation of requirements therefore causes additional effort and

therefore additional costs. However, the advantages gained by performing requirements validation and negotiation as described in the previous sections (reduction of overall cost, increase in acceptance, supporting creativity and innovations) is usually significantly higher than the costs that arise due to the increased effort.

7.3 Quality Aspects of Requirements

A major aim of using quality criteria (e.g., completeness, understandability, agreement) in requirements validation is to be able to check requirements systematically (see section 1.1.2). In order to assure an objective and consistent validation, it is necessary that each quality criterion is concretized and refined. In correspondence with the overall goals of the requirements engineering process, the validation is carried out with the following goals:

- *Content*: Have all relevant requirements been elicited and documented with the appropriate level of detail?
- *Documentation*: Are all requirements documented with respect to the predetermined guidelines for documentation and specification?
- *Agreement*: Do all stakeholders concur with the documented requirements and have all known conflicts been resolved?

Three quality aspects

Each of the three goals implies an individual approach that focuses on specific aspects of the quality of the requirements. Therefore, the following three quality aspects have been defined:

- Quality aspect “content”
- Quality aspect “documentation”
- Quality aspect “agreement”

A requirement should be approved for further development activities only if all three quality aspects have been checked. The quality aspects are described in detail in the following sections and made concrete through different fine-grained quality criteria (with no claim of completeness).

7.3.1 Quality Aspect “Content”

The quality aspect “content” refers to the validation of requirements with respect to errors in the content. Errors in requirements with regard to content negatively influence the subsequent development activities and cause these activities to be based upon erroneous information.

Test criteria of the quality aspect “content”

Errors in requirements with regard to content are present when specific quality criteria for requirements (see section 4.6) or for requirements documents (see section 4.5) are violated. The validation of requirements with regard to the quality aspect “content” is successful once requirements validation has been applied to the following error types and no significant shortcomings have been detected:

- *Completeness (set of all requirements)*: Have all relevant requirements for the system to be developed (for the next system release) been documented?
- *Completeness (individual requirements)*: Does each requirement contain all necessary information?
- *Traceability*: Have all relevant traceability relations been defined (e.g., to relevant requirements sources)?
- *Correctness/adequacy*: Do the requirements accurately reflect the wishes and needs of the stakeholders?
- *Consistency*: Is it possible to implement all defined requirements for the system to be developed jointly? Are there no contradictions?
- *No premature design decisions*: Are there any forestalled design decisions present in the requirements not induced by constraints (e.g., constraints that specify a specific client-server architecture to be used)?
- *Verifiability*: Is it possible to define acceptance and test criteria based on the requirements? Have the criteria been defined?
- *Necessity*: Does every requirement contribute to the fulfillment of the goals defined?

7.3.2 Quality Aspect “Documentation”

The quality aspect “documentation” deals with checking requirements with respect to flaws in their documentation or violations of the documentation guidelines that are in effect, such as understandability of the documentation formats and the consideration of organizational or project-specific guidelines regarding the documentation of requirements but also the structure of the requirements documents.

Implications of the violation of documentation guidelines

Ignoring the documentation guidelines can, among other things, lead to the following risks:

- *Impairment of development activities*: It may be impossible to carry out development activities that are based upon a specific documentation format.
- *Misunderstandings*: Requirements may not be understandable or may be misunderstood by the people that need to comprehend them. As a result, the requirement may be unusable.
- *Incompleteness*: Relevant information is not documented in the requirements.
- *Overlooking requirements*: If requirements are not documented at the position that they are supposed to in the requirements document, these requirements may be overlooked in subsequent activities.

Test criteria of the quality aspect “documentation”

Requirements validation with regard to the quality aspect “documentation” is successful when requirements validation has been applied to the following error types and no significant shortcomings have been detected:

Four test criteria of the quality aspect “documentation”

- *Conformity to documentation format and to documentation structures*: Are the requirements documented in the predetermined documentation format? For instance, has a specific requirements template or a specific modeling language been used to document the requirements? Has the structure of the requirements document been maintained? For instance, have all requirements been documented at the position defined by the document structure?
- *Understandability*: Can all documented requirements be understood in the context given? For instance, have all terms used been defined in a glossary (see section 4.7)?
- *Unambiguity*: Does the documentation of the requirements allow for only one interpretation or are multiple different interpretations possible? For instance, does a text-based requirement not possess any kind of ambiguity?
- *Conformity to documentation rules*: Have the predetermined documentation rules and documentation guidelines been met? For instance, has the syntax of the modeling language been used properly?

7.3.3 Quality Aspect “Agreement”

The quality aspect “agreement” deals with checking requirements for flaws in the agreement of requirements between stakeholders.

Last opportunity for changes

During the course of requirements engineering, stakeholders gain novel knowledge about the system to be developed. Due to this additional knowledge, the opinion of the stakeholders regarding a requirement that has already been agreed upon can change. During requirements validation, stakeholders have the opportunity to request changes without impairing the subsequent development activities.

Requirements validation with regard to the quality aspect “agreement” is successful when requirements validation has been applied to the following error types and no significant shortcomings have been detected:

Three test criteria of the quality aspect “agreement”

- *Agreed*: Is every requirement agreed upon with all relevant stakeholders?
- *Agreed after changes*: Is every requirement agreed upon with all relevant stakeholders after it has been changed?
- *Conflicts resolved*: Have all known conflicts with regard to the requirements been resolved?

7.4 Principles of Requirements Validation

Considering the following six principles of requirements validation increases the quality of the validation results:

- *Principle 1*: Involvement of the correct stakeholders
- *Principle 2*: Separating the identification and the correction of errors
- *Principle 3*: Validation from different views
- *Principle 4*: Adequate change of documentation type
- *Principle 5*: Construction of development artifacts
- *Principle 6*: Repeated validation

The individual principles are explained in the following sections.

7.4.1 Principle 1: Involvement of the Correct Stakeholders

The choice of stakeholders for requirements validation depends on the goals of the validation as well as the requirements that are to be audited.

When assembling the auditing team, at least the following two aspects ought to be considered.

Independence of the auditor

Generally, it should be avoided that the author of a requirement is also the person to validate it. The author will make use of his or her prior knowledge when reading or reviewing the requirement. This prior knowledge can negatively influence the identification of errors because potential erroneous passages of the requirements documentation or the requirements are implicitly and subconsciously amended by the author's own knowledge and can thus easily be overlooked.

Internal vs. external auditors

Suitable auditors can be identified within or outside of the developing organization. Internal audits are performed by stakeholders that are members of the developing organization and can be used to validate intermediate results or preliminary requirements. An internal validation is easy to coordinate and organize because the stakeholders are available from within the organization. An external audit requires a higher degree of effort because it identifies auditors and (potentially) hires them for payment. In addition, external auditors have to become familiar with the context of the system to be developed. Due to the high effort, an external audit should be performed only on requirements that exhibit a high level of quality.

7.4.2 Principle 2: Separating the Identification and the Correction of Errors

Basic principle

Separation between identifying errors and actually fixing them has proven itself in the domain of software quality assurance. The same principle can be applied to requirements validation. During validation, the flaws identified are documented immediately. After that, each flaw identified is double-checked to determine whether it really is an error.

Concentrating on error identification

Separating error identification and error correction allows auditors to concentrate on the identification. Measures to correct the errors are taken only after identification measures have been completed. This has the advantages that the resources available for error correction can be used purposefully, that premature error identification does not create additional errors, and that no alleged error is fixed that did not need fixing because further investigation of the error may result in the fact that an alleged error is in fact no error at all. That way, potentially present significant errors are less likely to be overlooked because the auditor is concentrating on fixing a previous error instead of identifying new ones.

7.4.3 Principle 3: Validation from Different Views

Perspective-based validation

Validating requirements from different views is another principle that has proven itself in practice. In this principle, requirements are validated and agreed upon from different perspectives (e.g., by different people, see section 7.5.4). Comparable methods are used in other disciplines as well. For instance, in a legal trial, circumstances are often reported from the perspective of different people so that a sound overall picture can be gained.

7.4.4 Principle 4: Adequate Change of Documentation Type

Strengths and weaknesses of documentation types

Changing the documentation type during requirements validation uses the strengths of one documentation type to balance out the weaknesses of other documentation types. For instance, good understandability and expressiveness are strengths of natural language texts. However, their weakness is potential ambiguity and difficulty in expressing complex circumstances. Graphic models are able to depict complex circumstances rather well, but the individual modeling constructs are restricted in expressiveness.

Simpler identification of errors

Transcribing a requirement that is already documented in another form of documentation simplifies finding errors. For instance, ambiguities in natural language requirements can be identified much easier by transcribing them into a model-based representation.

7.4.5 Principle 5: Construction of Development Artifacts

Suitability of the requirements for design, test, and manual creation

Constructing development artifacts aims at validating the quality of requirements that are meant to be the basis of creating design artifacts, test artifacts, or the user manual. During the course of the validation, the activities usually carried out during subsequent phases to construct respective development artifacts are carried out for small samples. For instance, the auditor intensively deals with a requirement by creating a test case. This way, errors (e.g., ambiguity) can be identified in the requirement. This kind of validation, however, demands a lot of resources because subsequent development activities must be executed at least in part.

7.4.6 Principle 6: Repeated Validation

Validation occurs at a distinct point in time during the development process and relies on the level of knowledge of the auditors at that point in time. During requirements engineering, the stakeholders gain additional knowledge about the planned system. Therefore, a positive validation of requirements does not guarantee that requirements are still valid at a later point in time. Requirements validation should occur multiple times in the following cases (among others):

- Lots of innovative ideas and technology used in the system
- Significant gain of knowledge during requirements engineering
- Long-lasting projects
- Very early requirements validation
- Unknown domain
- Requirements that are to be reused

7.5 Requirements Validation Techniques

In the following sections, techniques for requirements validation are introduced. Often, manual validation techniques, which are also known by the general term *review*, are used for requirements validation. Three major types of reviews can be differentiated:

- Commenting

- Inspections
- Walk-throughs

Along with reviews, the following three techniques have proven themselves to be useful for requirements validation:

- Perspective-based reading
- Validation through prototypes
- Using checklists for validation

In the following, these six techniques are described. Prior to applying any of these techniques, preparatory steps need to be taken as needed, such as identifying and inviting the right stakeholders or organizing suitable rooms and supplies.

7.5.1 Commenting

Individual validation of requirements

During commenting, the author hands his or her requirements over to another person (e.g., a co-worker). The goal is to receive the co-worker's expert opinion with regard to the quality of a requirement. The co-worker reviews the requirement with the goal to identify issues that impair requirement quality (e.g., ambiguity or errors) with respect to predetermined quality criteria. The identified flaws are marked in the requirements document and briefly explained.

7.5.2 Inspection

Typical phases of an inspection

Inspections of software or any other type of product are done to systematically check development artifacts for errors by applying a strict process [Laitenberger and DeBaud 2000].

An inspection is typically separated into various phases [Gilb and Graham 1993]: planning, overview, defect detection, defect correction, follow-up, and reflection. For requirements validation, the planning, overview, error detection, and error collection phases are relevant (see principle 2, separating the identification and correction of errors in section 7.4.2). Individual preparation is an obligatory part of inspections. An inspection session usually serves the purpose of collecting and evaluating error

indications. Occasionally, performing dedicated inspection sessions is omitted when performing inspections.

Planning

Among other things, the goal of the inspection, the work results that are to be inspected, and the roles and participants are determined during the planning phase.

Overview

In the overview phase, the author explains the requirements to be inspected to all team members so that there is a common understanding about the requirement among all inspectors.

Error detection

In the error detection phase, the inspectors search through the requirement for errors. Error detection can be performed individually by each inspector or collaboratively in a team. Individual inspection has the advantage that each inspector can concentrate on the requirements. On the other hand, team inspections have the advantage that communication between the inspectors creates synergy effects during error detection. During the course of error detection, any errors that are found are purposefully documented.

Error collection and consolidation

In the error collection phase, all identified errors are collected, consolidated, and documented. During consolidation, errors that have been identified multiple times or errors that aren't really errors are identified. The latter can be the case if, for instance, an inspector makes wrong assumptions about a requirement or interprets some constraint the wrong way. Along with consolidation, the identified errors and correcting measures are documented in an error list. Inspections are also known as *technical reviews*.

Roles during inspection

For an inspection to be performed, the following roles must be staffed with suitable personnel:

- *Organizer*: The organizer plans and supervises the inspection process.
- *Moderator*: The moderator leads the session and ensures that the predetermined inspection process is followed. It is advisable to select a neutral moderator because the moderator could potentially balance out opposing opinions of authors and inspectors.

- *Author*: The author explains the requirements that he created to the inspectors in the overview phase and later on is responsible for correcting the errors identified.
- *Reader*: The reader introduces the requirements to be inspected successively and guides the inspectors through them. The role of the reader should be given to a neutral stakeholder so that the inspectors can center their attention on the requirements instead of on the interpretation of the author. Often, the moderator is also the reader.
- *Inspectors*: The inspectors are responsible for finding errors and communicating their findings to the other members of the project team.
- *Minute-taker*: This person takes minutes of the results of the inspection.

7.5.3 Walk-Through

Lightweight inspection

In requirements validation, a walk-through is a lightweight version of an inspection. A walk-through is less strict than an inspection and the involved roles are differentiated to a lesser degree. During a walk-through, at least the roles of the reviewer (comparable to the inspector), author, and minute-taker, and potentially the moderator, are staffed.

Discussion of the identified flaws in quality during a group session

The goal of a walk-through of requirements is to identify quality flaws within requirements by means of a shared process and to gain a shared understanding of the requirements between all the people involved. To prepare for a walk-through, the requirements to be validated are handed out to all participants and inspected. During the walk-through session, the participants discuss the requirements to be validated step-by-step, under guidance of the moderator/reader. Usually, the author of a requirement is the one who introduces the requirement to all other participants. This way, the authors have the opportunity to give additional information to the group along with the actual requirement (e.g., alternative requirements, decisions, and rationale for decisions). A minute-taker documents the flaws in quality that have been identified during the session.

7.5.4 Perspective-Based Reading

Check requirements from a defined perspective.

Perspective-based reading is a technique for requirements validation in which requirements are checked by adopting different perspectives [Basili et al. 1996]. Typically, perspective-based reading is applied in conjunction with other review techniques (e.g., during inspections or walk-throughs). Focusing on particular perspectives when reading a document verifiably leads to improved results during requirements validation. Possible perspectives for validation, for instance, emerge from the different addressees of a requirement [Shull et al. 2000]:

- *User/customer perspective:* The requirements are checked from the perspective of the customer or the user to determine whether they describe the desired functions and qualities of the system.
- *Software architect perspective:* The requirements are checked from the perspective of the software architect to ascertain if they contain all necessary information for architectural design (e.g., if all relevant performance properties have been described).
- *Tester perspective:* The requirements are checked from the perspective of the tester to establish whether they contain the information necessary to derive test cases from the requirements.

Perspective quality aspects

The three quality aspects (see section 7.3) also describe three possible perspectives for requirements validation:

- *Content perspective:* With the content perspective, the auditor verifies the content of requirements and focuses on the quality of the content of the documented requirement.
- *Documentation perspective:* With the documentation perspective, the auditor ensures that all documentation guidelines for requirements and requirements documents have been met.
- *Agreement perspective:* With the agreement perspective, the auditor checks if all stakeholders agree on a requirement, i.e., if the requirements are agreed upon and conflicts have been resolved.

In addition, further perspectives that emerge from the individual context of the development project can be created as need be.

Define validation directives for each perspective.

During perspective-based validation, each auditor is assigned a perspective (at the proper point in time) from which she reads and validates the requirement. For each perspective defined, detailed instructions for performing the validation should be laid down because the auditor might not be familiar with all relevant details of her assigned perspective. It is advisable to associate questions with each validation instruction that must be answered by the content of the requirements or by the auditor after she has read the requirement, respectively. In addition, validation instructions can be amended with a checklist that summarizes the most important content aspects that ought to be addressed by a requirement with regard to the appropriate perspective.

Follow-up

During the course of the follow-up to a perspective-based reading session, the results of the chosen perspective are analyzed and consolidated. On the one hand, the results of the perspective-based reading contain answers to the predefined questions, and on the other hand, open issues that the auditors noticed while reading may be present. The consolidation can be done as a group effort, similarly to a review.

Support of other techniques

Perspective-based reading can be both an independent technique for requirements validation and a support technique for other validation techniques, such as inspections or reviews of requirements documents by means of perspective-based reading.

7.5.5 Validation through Prototypes

Requirements validation by means of prototypes allows auditors to experience the requirements and to try them out. Experiencing requirements directly through prototypes [Jones 1998] is the most effective method to identify errors in requirements. Stakeholders can try out the prototype and compare their own idea of how the system ought to be implemented with the prototype at hand and thereby find discrepancies between their ideas and the current implementation.

Evolutionary vs. throw-away prototypes

Depending on the further use of the prototype, one can distinguish between throw-away prototypes and evolutionary prototypes [Sommerville 2007]. Throw-away prototypes are not maintained once they have been used. Evolutionary prototypes are developed with the goal to be developed further and improved in later steps. In contrast to throw-away prototypes, implementation plays a much more significant role

here. Therefore, the effort to create evolutionary prototypes is much higher.

Selection of relevant requirements

Before a prototype can be implemented, the requirements that shall be validated through the prototype must be selected. The set of requirements to be validated is limited by development resources (e.g., time, money, etc.) that can be allocated for validation. For example, a selection criterion can be the criticality of a requirement.

Preparation of the validation

The following preparations have to be made in order to validate requirements by means of prototypes:

- *Manual/instructions*: The users of the prototype must be supplied with the necessary information so that they can use or apply the prototype. This can be done by means of a manual or by means of proper instruction.
- *Validation scenarios*: Validation scenarios that the users of the prototype can perform with the prototype should be prepared. A validation scenario defines, for example, all relevant data sets or user interactions.
- *Checklist with validation criteria*: For requirements validation, a checklist of validation criteria should be created according to which the prototype (and by proxy, the requirements) can be validated.

Performing the validation

The auditor should validate the prototype without being influenced; i.e., the auditor should execute the validation scenarios independently and by herself. This ensures that the validation results are created without bias.

During validation, the auditors can and ought to execute alternative and deviant scenarios and should use the prototype exploratively and experimentally once the required validation scenarios have been covered. For example, error cases that have remained hidden until then can be identified. For experimental validation of the prototype, the auditor needs to know the scope of the prototype, i.e., the set of requirements that have been considered when the prototype was created. Without knowledge of the implemented requirements, an auditor cannot decide whether an identified error can be traced back to a missing requirement or if the requirement has been consciously omitted in the prototype.

Documentation of the validation results

Requirements validation through prototypes therefore permits two types of result

documentation:

- *Protocol of the auditor*: The auditor documents the results and experiences made during the validation of the prototype, e.g., by means of validation scenarios as well as a checklist that he has been supplied with.
- *Observation protocol*: The auditor can be observed by a second person. The second person creates a so-called observation protocol. This protocol can disclose additional important symptoms for errors in requirements. For example, when the auditor hesitates at a certain step in the validation scenario while using the prototype and the observer documents this, it may be an indication for missing apparentness and as such an indication for impaired understandability of the prototype. Under certain circumstances, it may be advisable to record the validation on video because the validation situation can be analyzed in detail during the follow-up. For example, a video recording can show the realization of requirements pertaining to anthropometric properties (such as ergonomics) or intuitive use and can be investigated in detail.

Analysis

The results of the validation are analyzed after validation is complete. Change suggestions for the requirements are consolidated. If significant changes to the requirements are necessary, it may be advisable to revise the prototype and validate anew.

7.5.6 Using Checklists for Validation

A checklist comprises a set of questions and/or statements about a certain circumstance. Checklists can be applied whenever many aspects must be considered in a complex environment and no aspect must be omitted. A checklist for requirements validation contains questions that ease the detection of errors [Boehm 1984]. Using checklists for requirements validation is very common in practice. Checklists can be used in all previously introduced techniques for requirements validation.

Creating checklists

Before a checklist can be used, every single question or statement must be defined. The sources for questions and statements in the following list can be used to create checklists to support requirements validation:

- The three quality aspects of requirements (see section 7.3)

- Principles of requirements validation (see section 7.4)
- Quality criteria for requirements documents (see section 4.5)
- Quality criteria for individual requirements (see section 4.6)
- Experiences of the auditors from prior projects
- Error statistics [Chernak 1996]

Improving checklists

Checklists are not necessarily complete. When using a checklist, one should always look for opportunities to improve the checklist for future use. For example, if a question was forgotten, the checklist should be amended to contain the extra question. Ambiguous questions or questions that are not understandable must be marked and revised. Outdated or no longer valid questions should be removed.

Checklists as a guideline

Checklists can support requirements validation in different ways. They can serve as a guideline for the auditor, who can follow the checklists at her own discretion (e.g., during a review).

Checklists as a means of structuring

The checklist can define a list of questions that must be strictly adhered to. These questions must be answered by the auditor to validate the requirements. In this case, the checklist serves as a measure to approach the validation in a structured manner. For example, the checklist may detail the exact process that the auditors are asked to apply, which guarantees that every auditor validates the requirements in the same way. This makes the results more comparable.

Hybrid forms of checklist application are also possible. For example, a checklist can contain obligatory questions for perspective-based reading and can contain suggestions that the auditor may or may not follow.

Successfully applying checklists

Applying checklists for requirements validation successfully depends on the manageability and complexity of the checklist. A large amount of questions can make it more difficult to use the checklist because the auditor does not have a steady overview of the questions and is thus forced to consult the checklist frequently. It is therefore advisable to design the checklist to not be longer than a single page [Gilb and Graham 1993]. In addition, questions that are formulated altogether too generically or abstractly can make it more difficult to use the checklist. For example, the question “Is the requirement formulated appropriately?” can lead to a multitude of different

answers, depending on what the auditor considers an appropriately formulated requirement. The questions therefore ought to be as precise as possible.

7.6 Requirements Negotiation

To negotiate the requirements of a system to be developed, it is necessary to identify conflicts and to resolve those conflicts. This is done by means of systematic conflict management. The conflict management in requirements engineering comprises the following four tasks:

Four tasks of conflict management

- Conflict identification
- Conflict analysis
- Conflict resolution
- Documentation of the conflict resolution

These four tasks are explained in the following sections.

7.6.1 Conflict Identification

Conflicts can arise during all requirements engineering activities. For example, different stakeholders can utter contradicting requirements during elicitation.

Conflict identification in all requirements engineering activities

Conflicts between requirements and conflicts between stakeholders are often not obvious due to different reasons. During the entire requirements engineering process, the requirements engineer should pay attention to potential conflicts so that they can be identified, analyzed, and resolved early on.

7.6.2 Conflict Analysis

Determining the conflict type

During conflict analysis, the reason for an identified conflict must be determined. According to [Moore 2003], different types of conflicts exist.

Data conflict

A *data conflict* between two or more stakeholders is characterized by a deficit of information, by false information, or by different interpretations of some information. For example, take the following requirement: “R131: The reaction time of the planned system shall not exceed one second”. A data conflict between two stakeholders with regard to this requirement can arise from the fact that one stakeholder considers a reaction time of 1 second to be too slow while another stakeholder does not believe that a reaction time of 1 second is feasibly implementable (i.e., it is too short).

Conflict of interest

A *conflict of interest* between two or more stakeholders is characterized by subjectively or objectively different interests or goals of stakeholders. A conflict of interest between two or more stakeholders can arise, for instance, when one stakeholder primarily focuses on keeping the costs of the planned system at a minimum while another stakeholder primarily desires a high level of quality. A conflict of interest between these two stakeholders arises when the first stakeholder rejects a requirement due to estimated costs and the second stakeholder insists on implementing it due to quality reasons.

Conflict of value

A *conflict of value* is characterized by differing underlying values stakeholders have regarding some circumstance (e.g., cultural differences, personal ideals). For instance, a conflict of value arises when one stakeholder favors open source technologies while another stakeholder favors closed sources technologies.

Relationship conflict

A *relationship conflict* is characterized by strong emotions, stereotypical relationship concepts, deficient communication, or negative interpersonal behavior between stakeholders (e.g., insults, disrespect). For instance, a relationship conflict arises when two stakeholders of equal rank or position (e.g., department leaders) reject each other's requirements and try to distinguish themselves by forcing their requirements onto the project.

Structural conflict

A *structural conflict* is characterized by unequal levels of authority or power. For instance, a structural conflict can arise between an employee and his superior if the superior invariably rejects requirements that the employee has defined because he does not recognize the employee's competence to delineate requirements.

Mixed reasons for conflicts

Often, it is difficult to unambiguously classify emerging conflicts. For example, a conflict can be a relationship conflict with clear structural components. Similarly, a conflict of interest can be a conflict of values as well. Therefore, it is advisable to analyze an identified conflict with respect to all types so that all possible reasons for the conflict can be determined and suitable resolution strategies can be selected.

7.6.3 Conflict Resolution

Good conflict resolution is a success factor.

Conflict resolution is very important for requirements negotiation because the strategy of conflict resolution has a big influence on the willingness of the people involved (e.g., customers, consultants, or developers) to continue working together. For example, a conflict resolution considered unfair by at least one party of the conflict can lead to a decreased engagement and willingness to collaborate in the project. On the other hand, a resolution that is considered fair by all parties can increase the willingness to cooperate because this signals that everyone's ideas about the planned system are being considered.

Involvement of the relevant stakeholders

Independently from the selected resolution strategy, it is essential to involve all relevant stakeholders. If not all relevant stakeholders are considered, some opinions and viewpoints will remain unconsidered. The conflict will therefore only be resolved in part or incompletely. In the following paragraphs, different conflict resolution techniques are introduced.

Agreement

With the conflict resolution technique *agreement*, all conflict parties negotiate a solution to the conflict. The parties exchange information, arguments, and opinions and try to convince one another of each other's viewpoints in order to come to an agreeable solution.

Compromise

With the conflict resolution technique *compromise*, all conflict parties try to find a compromise between alternative solutions. In contrast to an agreement, a compromise consists of an amalgamation of different parts of the alternative solutions. Also, a compromise can mean that all alternative solutions as proposed thus far are discarded

and entirely new solutions are creatively developed.

Voting

In the conflict resolution technique *voting*, all conflict parties vote on solution alternatives. The alternatives that are up for voting are presented to all relevant stakeholders. Each stakeholder casts her vote for an alternative and the alternative with the most votes is accepted as the resolution for the conflict.

Definition of variants

In the conflict resolution technique *definition of variants*, the system is developed in a way that permits the definition of variants by deriving variants, by selecting parameters that define system variants, or by selecting variable system properties. This way, the system can satisfy the different interests of the stakeholders.

Overruling

In the conflict resolution technique *overruling*, a conflict is resolved by means of the hierarchical organization. This means that a conflict party with higher organizational rank or position wins the conflict by overruling objections of organizationally lower parties. If both parties have the same organizational rank, the conflict is resolved by a superior stakeholder or some third-party decider. This conflict resolution technique is only advisable if other resolution techniques have failed (e.g., no compromise could be found) or are not applicable due to limitations of resources (e.g., time).

Consider-all-facts

In the conflict resolution technique *consider-all-facts (CAF)*, all influencing factors of a conflict are being investigated so that as much information about the conflict can be collected as possible. This information is used during resolution. By prioritizing the influence factors, the relevance is determined. Based on the results of this technique, the plus-minus-interesting conflict resolution technique can be applied.

Plus-minus-interesting

In the conflict resolution technique *plus-minus-interesting (PMI)*, all positive and negative repercussions of a solution alternative are investigated so that positive and negative repercussions can be evaluated. Positive repercussions are placed in the category “plus” and negative repercussions are placed in the category “minus”. Repercussions that are neither positive nor negative are placed in the category “interesting”. Repercussions in the category “interesting” cannot be evaluated yet and must be investigated further to determine if their influence is positive or negative.

Decision matrix

In the conflict resolution technique *decision matrix*, a table is created that contains solution alternatives in the columns and all relevant decision criteria in the rows. The decision criteria can be identified by means of the technique “consider-all-facts”. For each combination of criterion and solution alternative, an assessment is made, for instance by means of a pointscale ranging from irrelevant (0 points) to relevant (10 points). [Table 7-1](#) shows a decision matrix.

	Solution alternative 1	Solution alternative 2	Solution alternative 3
Criterion 1	3	6	2
Criterion 2	5	4	10
Criterion 3	10	3	5
Sum	18	13	17

Table 7-1 Decision matrix

In order to find a solution, the sums of the columns are calculated; i.e., the assessments of the criteria of each solution alternative are summed up. The solution alternative with the highest score is accepted as the decision. In the example shown in [table 7-1](#), this would be solution alternative 1.

7.6.4 Documentation of the Conflict Resolution

Risks of missing conflict documentation

Conflicts cannot be avoided during requirements engineering. A resolution to a conflict must always be traceably documented. If a conflict resolution is not properly documented, the following threats (among others) to the project may arise:

- *Handling conflicts repeatedly*: A certain conflict can arise a second time during the requirements engineering process. Without proper documentation of the conflict resolution, the conflict must be analyzed and resolved anew. This causes additional effort and can potentially lead to additional conflicts or abrogate previous resolutions.
- *Inappropriate conflict resolution*: During the requirements engineering process, the resolution of a conflict can turn out to be wrong or unsuitable. In this case, the conflict must be investigated and resolved anew. Without proper documentation, relevant information that has been considered during the initial analysis and

resolution can be overlooked and the new conflict resolution can once again lead to false results.

In both cases, proper documentation of the conflict and its resolution supports the requirements engineering process and ensures that relevant information already known can be considered.

7.7 Summary

The quality of the elicited and documented requirements must be assured during requirements engineering so that it can be guaranteed that the requirements meet the desires and ideas of the stakeholders adequately. Therefore, it is necessary to validate the requirements with regard to the quality of their content, their documentation, and their agreement with respect to the different stakeholders. There are different techniques that can be selected and purposively combined for requirements validation, depending on the project peculiarities and project goals. Among the most common validation techniques for requirements are the different types of requirements reviews (e.g., commenting, inspection, walk-through) as well as perspective-based reading and validation through prototypes and checklists.

For requirements negotiation, it is necessary to identify conflicts between stakeholders, analyze them, and resolve them in a suitable manner. A systematic conflict management supports analysis and resolution of the conflicts that have been identified over the course of requirements validation or other requirements engineering activities.

8 Requirements Management

Requirements management comprises purposefully assigning attributes to requirements, defining views on requirements, prioritizing requirements, and tracing requirements as well as versioning requirements, managing requirements changes, and measuring requirements. Requirements management includes the management of individual requirements as well as the management of requirements documents.

8.1 Assigning Attributes to Requirements

Information about the requirements must be documented throughout the entire life cycle of a system. This includes, for example, unique identifiers of a requirement, the name of the requirement, the author and sources of the requirement, and the person responsible for the requirement.

8.1.1 Attributes for Natural Language Requirements and Models

To document information about requirements, it has proven useful to delineate this information in a structured manner: as attributes. Attributes of a requirement are defined by a unique name, a short description of the contents, and the set of possible values that can be assigned to the attribute.

Template-based assignment of attributes to requirements

The simplest way to define requirement attributes is by means of a table structure (template). The template defines the relevant information that is to be documented. This information, i.e., the defined attributes (attribute types), can be different for each type of requirement. For example, the template for functional requirements can be different from the template for quality requirements with respect to the defined attribute types and/or the allowed attribute values.

8.1.2 Attribute Scheme

The set of all defined attributes for a class of requirements (e.g., functional requirements, quality requirements) is called an attribute scheme. Attribute schemes are usually tailored to meet the individual project's needs.

Assignment of requirement attributes

During the course of the project, the attributes of the requirements are assigned with fitting attribute values. [Figure 8-1](#) shows an exemplary assignment of attributes for a requirement including the attributes “identifier”, “name”, and “requirement description” as well as attributes that allow for documenting the stability of the requirements and its source as *well as its author*.

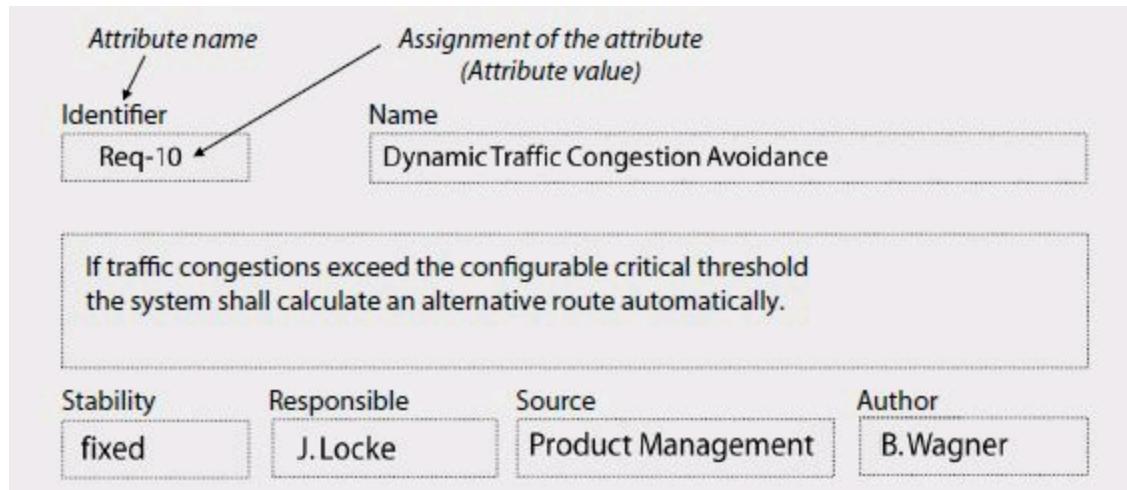


Figure 8-1 Example of requirement attribute assignment

The requirement that is documented on the basis of the simple template shown in [figure 8-1](#) has the code “Req-10” as its unique identifier. It bears the name “Dynamic Traffic Congestion Avoidance” and a description that specifies the subject of this requirement. The stability of this requirement is classified as “fixed”, “J. Locke” is the person responsible for this requirement, and the requirement stems from the source “Product Management”. “B. Wagner” is the author.

The reader of the requirement (e.g., the contractor, product manager, developer, project manager) has a significant advantage when template-based documentation is used, namely that information of the same type can always be found in the same position (e.g., the requirement stability is always in the template section “stability”). In addition, template-based assignment of attributes has the advantage for the requirements engineer that it is harder for her to overlook important information and that this information, supported by the structure of the template and the predetermined attribute values, can be documented purposefully and correctly.

8.1.3 Attribute Types of Requirements

Frequently used attribute types

The various standards in requirements engineering and the most pertinent tools for requirements documentation and management often offer a set of predefined attributes. [Table 8-1](#) lists attribute types that are frequently used in practice during requirements management.

Attribute Type	Meaning
Identifier	Short, unique identifier of a requirement artifact from the set of all regarded requirements.
Name	Unique, characterizing name.
Description	Briefly describes the content of the requirement.
Version	Current version of the requirement.
Author	Specifies the author of the requirement.
Source	Specifies the source or sources of the requirement.
Stability	Specifies the approximate stability of the requirement. The stability is the amount of changes that are to be expected with regard to the requirement. Possible values can be "fixed", "established", and "volatile".
Risk	Specifies the risk based on an estimate of the amount of damage and loss and the probability of occurrence.
Priority	Specifies the priority of the requirement regarding the chosen prioritization properties, e.g., "importance for market acceptance", "order of implementation", "loss/opportunity cost if not implemented".

Table 8-1 Frequently used attribute types

Additional attribute types for requirements

Along with the requirements attributes listed in [table 8-1](#), many additional attribute types exist to document important information of a requirement. [Table 8-2](#) shows a selection of additional attribute types for requirements.

Attribute Type	Meaning
Person responsible	Specifies the person, group of stakeholders, organization, or organizational unit that is responsible for the content of the requirement.
Requirement type	Specifies the type of requirement (e.g., "functional requirement", "quality requirement", or "constraint") depending on the applied classification scheme.
Status regarding the content	Specifies the current status of the content of the requirement, e.g., "idea", "concept", "detailed content".
Status regarding the validation	Specifies the current status of the validation, e.g., "unvalidated", "erroneous", "in correction".
Status regarding the agreement	Specifies the current status of the negotiation, e.g., "not negotiated", "negotiated", "conflicting".
Effort	Estimated/actual effort to implement the requirement
Release	The designation of the release in which the requirement shall be implemented.
Legal obligation	Specifies the degree of legal obligation of the requirement.
Cross references	Specifies relations to other requirements, for example, if it is known that the implementation of the requirement requires prior implementation of another requirement.
General information	In this attribute, arbitrary information that is considered relevant can be documented, for example, if the negotiation of this requirement is scheduled for discussion during the next meeting with the stakeholders.

Table 8-2 Additional types of requirements attribute

Project-specific tailoring of the attribute scheme

The attribute types suggested are the basis for defining an attribute scheme in the development project. To define an attribute scheme, at least the following specific aspects must be considered:

- Specific properties of the project, e.g., project size, local or distributed development, or project risk
- Constraints of the organization, e.g., organizational standards and regulations
- Properties and regulations of the application domain, e.g., reference models, modeling guidelines, standards
- Constraints and restrictions of the development process, e.g., liability law, process standards

Definition of attributes by means of information models

When employing tools for requirements management, defining the attribute structure of requirements is often not done by means of tables, but is model based, by means of information models. A model-based definition of an attribute scheme determines the attribute types as well as limitations in attribute values, similar to template-based definitions. In addition, model-based attribute scheme definition allows for determining relations between attribute types of different attribute schemes.

Advantages of model-based attributing

Along with the advantages of table-based definition, model-based assignment of requirement attributes additionally allows consideration of requirement dependencies when selectively accessing the requirements. This aids in maintaining consistency in the attributes of the requirements. Furthermore, the information model of a model-based assignment of requirement attributes can serve as the foundation for the definition of an attribute structure to be used in a requirements management tool. (see section 9.3). Also, templates for the assignment of requirement attributes can be generated on the basis of the information model.

8.2 Views on Requirements

Structuring requirements by means of information models allows for generating specific views on requirements. It can be seen in practice that the amount of requirements and the amount of dependencies among requirements are evermore increasing. In order to keep the complexity of the requirements manageable for the project staff, it is necessary to selectively access and thereby filter the requirements depending on the current task.

Role-specific definition of views

Views on requirements are often defined for different roles in the development process. Examples include views for the architect, the programmer, the project manager, and the tester. It is common to define multiple views for a role in order to support the sub-activities of each role. One particular view can also be applied to multiple roles.

8.2.1 Selective Views on the Requirements

A view contains a part of all available requirement information. A view can do the

following:

- Select particular requirements; i.e., not every requirement is contained in a view.
- Mask certain attributes of requirements; i.e., not every attribute of a requirement is contained in a view.
- Arbitrarily combine both these selection principles; i.e., only a subset of all available requirements and only a subset of all available attributes are contained in a view.

Generating selective views

Figure 8-2 illustrates the generation of three views, represented by a table that is defined on the basis of the structure of the attributes. In all three cases, the views are created by selecting attribute types as well as by determining the attributes that must be available. The definition of the first view (1), for example, determines that only those requirements are selected that “J. Locke” is responsible for and that have a stability of “fixed”. Of all selected requirements, only the attributes “identifier”, “name”, “description”, and “author” are being considered.

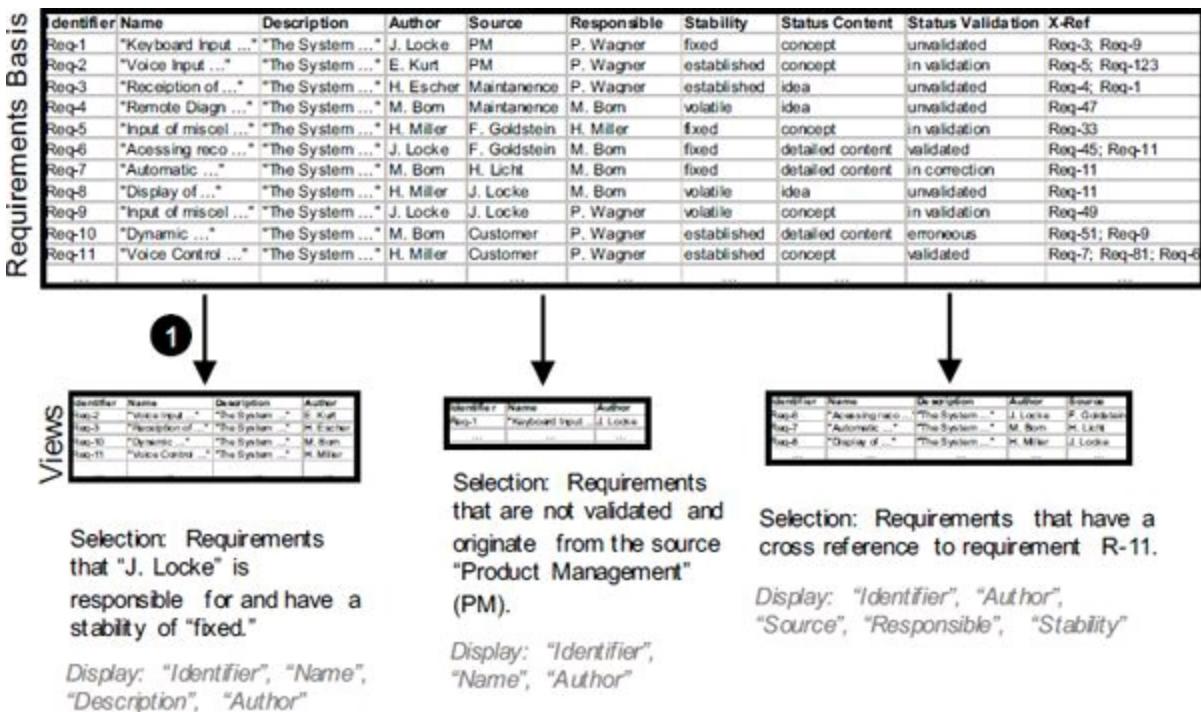


Figure 8-2 Selective views on the requirements

8.2.2 Condensed Views on the Requirements

Along with selecting existing information from the requirements basis, views can

contain generated or condensed data that is not immediately contained in the requirements. Views that contain only generated or condensed data are called condensed views.

Generating condensed views

Condensed views can be defined by aggregating the data contained in the requirements basis. A condensed view can, for example, contain information on the percentage of requirements that stem from a particular source.

Combination of selecting and condensing

A single view may also consist of a combination of generated, condensed, and selected data.

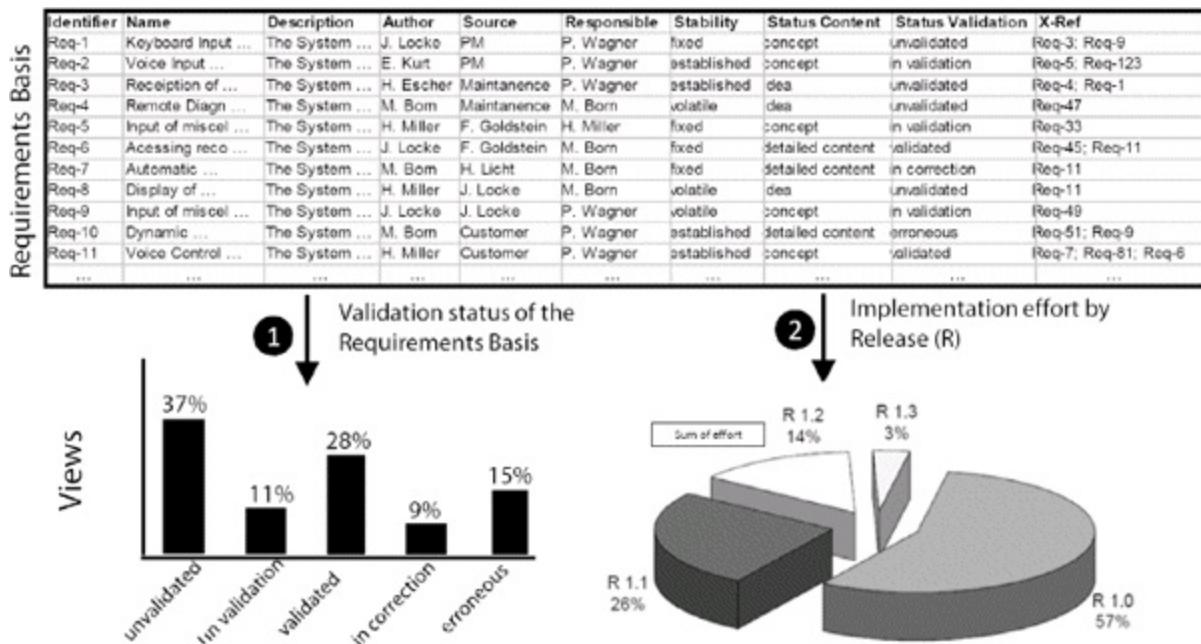


Figure 8-3 Condensed view generated from a requirements basis

Figure 8-3 shows two condensed views of the requirements. The view “Validation status of the Requirements Basis” (1) groups requirements according to the current status of validation and calculates the percentage value of the requirements with regard to the status “unvalidated”, “in validation”, “validated”, “in correction”, and “erroneous”. The result is depicted as a bar chart in the figure above. In view (2), “Implementation effort by Release”, the estimated and actual effort involved with the implementation of the requirements of a particular release is depicted. In order to calculate this aggregated data, the requirements are grouped by their respective release and their implementation effort is summed up. The result is depicted as a pie chart in figure 8-3.

8.3 Prioritizing Requirements

Requirements are prioritized during requirements engineering using different prioritization criteria in all sub-activities. Requirements can be prioritized by their order of implementation, for example. Due to the different prioritizations in the various sub-activities, the priority of a requirement can be determined by one or more attributes (e.g., priority of the contractor, priority due to urgency of implementation).

8.3.1 Method for Requirements Prioritization

Determining goal and constraints of prioritization

In order to prioritize a set of requirements, a goal (i.e., purpose) of prioritization must be defined first. In addition, the constraints of prioritization are documented, such as the availability of different stakeholders and groups thereof or the resources available for prioritization.

Determining prioritization criteria

Depending on the goal of prioritization, the criterion for prioritizing the requirements (or the combination of two or more criteria) is chosen. The following are typical examples of prioritization criteria:

- Cost of implementation
- Risk
- Damage due to unsuccessful implementation
- Volatility
- Importance
- Duration of implementation (i.e., how long it takes to be implemented)

Determining Stakeholders

Depending on the goal of prioritization and the selected prioritization criteria, it is usually necessary to involve different stakeholders in the prioritization process. By choosing appropriate stakeholders, it can be guaranteed that the required expert knowledge is available during the prioritization process. The stakeholders that ought to be involved are, depending on the goal and prioritization criteria, developers, project managers, customers, or users, for example.

Selection of artifacts

In addition, the requirements to be prioritized must be selected. When selecting requirements, one must make sure that the selected requirements stem from the same level of abstraction. Prioritizing requirements from considerably differing levels of detail will lead to inconsistent and erroneous results because stakeholders tend to assign a higher priority to requirements at higher levels of abstraction than to more refined and concrete requirements.

Selection of prioritization techniques

On the basis of the determined properties of the prioritization (e.g., constraints, criteria of prioritization, etc.), a suitable prioritization technique or a combination of multiple techniques is selected.

8.3.2 Techniques for Requirements Prioritization

For prioritization, multiple techniques exist. The techniques mainly differ with regard to the time and effort needed but also with regard to the suitability of the different prioritization criteria and project properties.

Ad hoc techniques and analytical techniques

The spectrum of prioritization techniques spans from simple, single-criterion classification to elaborate analytic prioritization approaches, such as AHP (Analytical Hierarchy Process) [Saaty 1980], Cost-Value-Analysis [Karlsson and Ryan 1997], or QFD (Quality Function Deployment) [Akao 1990].

In many projects, simple ad hoc prioritization techniques such as ranking or requirements classification are well suited. Especially with regard to the resources available, using ad hoc techniques is often advisable.

If the decision process is considered too incomprehensible, or if the results are too erroneous, analytical approaches for prioritization should be used (additionally). In practice, multiple prioritization techniques are used in combination in order to prioritize the requirements [Lehtola and Kauppinen 2006].

Ranking and Top-Ten Technique

Two well-established techniques for requirement prioritization are, for example, the following [Lauesen 2002]:

- *Ranking*: In this technique, a number of selected stakeholders arrange the

requirements to be prioritized with respect to a specific criterion.

- *Top-Ten Technique*: In this technique, the n most important requirements for a defined criterion are selected. For these requirements, a ranking order is determined afterward. This ranking order represents the importance of the selected requirements with regard to the defined criterion.

Single-Criterion Classification

Another prioritization technique that is often used in practice is based on the classification of requirements with respect to the importance of the realization of the requirements for the system's success. This type of prioritization is based on assigning each requirement to one of the following priority classes [IEEE 830-1998]:

- *Mandatory*: A mandatory requirement is a requirement that must be implemented at all costs or else the success of the system is threatened.
- *Optional*: An optional requirement is a requirement that does not necessarily need to be implemented. Neglecting a few requirements of this class does not threaten the success of the system.
- *Nice-to-have*: Nice-to-have requirements are requirements that do not influence the system's success if they are not implemented.

In practice, differentiating between “optional” and “nice-to-have” requirements can be very difficult. Therefore, requirements classification demands classification criteria that are as objectively verifiable as possible.

Kano Classification

The Kano approach introduced in section 3.2 also supports the prioritization of requirements. By making use of the Kano approach, one can classify and prioritize requirements with respect to their acceptance on the market. In order to do so, the following three property classes (see also [figure 3-1](#)) are classified:

The three properties in the Kano approach

- *Dissatisfiers*: A requirement specifies a dissatisfier the system must possess in order to be successfully introduced to the market.
- *Satisfiers*: A requirement specifies a satisfier if the customers consciously demand the associated property. Satisfiers of the system specify the degree of satisfaction of the customer. An increase in the number of satisfiers usually leads

to increased customer satisfaction.

- *Delighters*: A requirement specifies a delighter if the customers do not consciously demand the defined system property or the customers do not expect the implementation of the property. The customer satisfaction increases exponentially by implementing delighters.

On the basis of requirements classified according to Kano, a prioritization of the requirements can be performed in order to plan the system releases, for example.

Prioritization Matrix According to Wieggers

Computing requirement priorities

The prioritization matrix according to Wieggers [Wieggers 1999] is an analytical prioritization approach for requirements. The core of the approach is a prioritization matrix according to which the priorities of the regarded requirements can be determined systematically. [Figure 8-4](#) shows the structure of a prioritization matrix according to Wieggers as well as the method according to which priorities are calculated.

Relative weight 1 → 2 (WeightBenefit)					→ 1 (WeightDetri- ment)	→ 1 (WeightCost)	→ 0.5 (WeightRisk)			
	Requirement 2	Relative Benefit	Relative Detriment	Total				Priority	Rank	
R ₁	5	3	13	16.8	2	13,3	1	9,1	0.941	1
R ₂	9	7	25	32.5	5	33,3	3	27,2	0.692	3
R ₃	5	7	17	22.1	3	20,0	2	18,2	0.759	2
R ₄	2	1	5	6.5	1	6,7	1	9,1	0.577	4
↓ R ₅	4	9	17	22.1	4	26,7	4	36,4	0.489	5
Total	25	27	77	100	15	100	11	100	—	
	3	4	5	6	7	8	9			

Figure 8-4 Calculation of priorities in a prioritization matrix according to Wieggers

Systematic method to determine the requirement priorities

In the following, the calculation of priorities in a prioritization matrix according to Wieggers is only briefly sketched. More detailed information can be found in [Wieggers 1999].

The calculation of priorities in a prioritization matrix according to Wieggers can be done as follows:

1. Determine the relative weights for benefit, detriment, cost, and risk.
2. Determine the requirements to be prioritized.
3. Estimate the relative benefit.
4. Estimate the relative detriment.
5. Calculate the total values and percentage values for each requirement:

$$Value\%(R_i) =$$

$$Benefit(R_i) \times WeightBenefit + Detriment(R_i) \times WeightDetriment$$

6. Estimate the relative cost and calculate the cost percentage for each requirement.
7. Estimate the relative risks and calculate the risk percentage for each requirement.
8. Calculate the individual requirement priorities:

$$Priority(R_i) =$$

$$Value\%(R_i) / (Cost\%(R_i) \times WeightCost + Risk\%(R_i) \times WeightRisk)$$

9. Assert the rank of the individual requirements.

It became apparent in practice that analytical prioritization approaches such as the prioritization matrix according to Wiegers as sketched above demand considerably more time and effort than ad hoc approaches, so these ad hoc approaches are to be favored in many cases. However, analytical approaches have the advantage that the degree of subjectivity in the prioritization results can be significantly reduced so that they lead to more objective and comprehensible results in complex and critical prioritization situations.

8.4 Traceability of Requirements

An important aspect of requirements management is ensuring the traceability of requirements. The traceability of a requirement is the ability to trace the requirements over the course of the entire life cycle of the system (see section 4.5.5).

8.4.1 Advantages of Traceable Requirements

Advantages of requirements traceability

The use of traceability information supports system development in many aspects and is often the precondition for establishing and using certain techniques during the developmental process [Pohl 1996; Ramesh 1998]:

- *Verifiability*: Traceability of requirements allows verifying whether a requirement has been implemented in the system, i.e., if the requirement has been implemented through a system property.
- *Identification of gold-plated solutions in the system*: Traceability of requirements allows for the identification of so-called gold-plated solutions of the developed system and thereby allows identifying unneeded properties. In order to do that, for each system property (functional or qualitative), a check is performed to determine whether it contributes to the implementation of a requirement of the system.
- *Identification of gold-plated solutions in the requirements*: Tracing requirements back to their origin allows identifying requirements that do not contribute to any system goal and are not associated with any source. Usually, there is no reason for these requirements to exist and hence these requirements do not have to be implemented.
- *Impact analysis*: Traceability of requirements allows for the analysis of effects during change management. For example, traceability of requirements allows identifying the requirements artifacts that must be changed when their underlying requirements undergo a change.
- *Reuse*: Traceability of requirements allows for the reuse of requirements artifacts in other projects. By comparing the requirements of a previous project to the requirements of a new project by means of trace links, development artifacts (e.g., components, test cases) can be identified that may be adapted and/or reused in the new development project.
- *Accountability*: Traceability of requirements allows for retroactive assignment of development efforts to a requirement. After the requirement is implemented, for example, all partial efforts for the associated development artifact can be summed up and associated with the requirement.
- *Maintenance*: Traceability of requirements allows for simplified system maintenance. For example, the cause and effect of failures can be identified, the system components that are affected by the failure can be determined, and the effort for removing the underlying error can be estimated.

8.4.2 Purpose-Driven Definition of Traceability

As resources are usually severely restricted during development projects, capturing all conceivable information that supports the traceability of requirements over the course of the system life cycle is almost never possible.

Purpose of traceability information

In order to establish requirements traceability effectively and efficiently, the information to be recorded should be chosen with respect to the purpose that it will serve. In other words, only the information which has a clear purpose for system development or system evolution [Dömges and Pohl 1998; Ramesh and Jarke 2001] ought to be recorded. Recording of traceability information that is not purpose driven often results in the fact that the recorded information cannot be profitably used in the development project. Traceability information that is recorded in this fashion is often sketchy and incomplete, unstructured, and erroneous with regard to its further use.

8.4.3 Classification of Traceability Relations

Pre-RS traceability and post-RS traceability

The pertinent literature on the topic of requirements traceability suggests different kinds of traceability of requirements. A common differentiation is distinguishing between pre-requirements-specification (pre-RS) traceability and post-requirements-specification (post-RS) traceability of requirements [Gotel and Finkelstein 1994]. We thus distinguish between three kinds of traceability:

- *Pre-RS traceability*: Pre-RS traceability are traceability links between requirements and those artifacts that are the basis for the requirements, e.g., artifacts like the source or origin of a requirement (previous artifacts).
- *Post-RS traceability*: Post-RS traceability comprises traceability information between requirements and artifacts of subsequent development activities. For example, such artifacts could be components, implementation, or test cases that belong to a requirement (posterior artifacts).
- *Traceability between requirements*: The traceability between requirements is about mapping dependencies between requirements. An example of this kind of traceability is the information that a requirement refines another requirement, generalizes it, or replaces it.

Figure 8-5 shows the three types of traceability of requirements in requirements engineering.

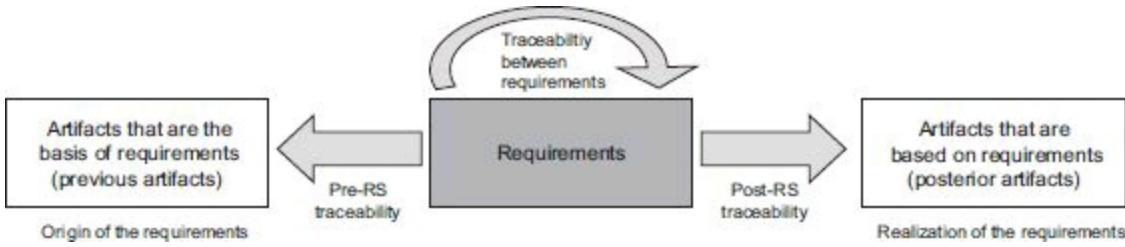


Figure 8-5 Types of requirements traceability

Figure 8-6 shows the three types of requirements traceability by means of requirement “R-14” in an example. The pre-RS traceability comprises the relations of requirement “R-14” to its origin. The origin of this requirement are the artifacts in the system context that influence the requirement. The post-RS traceability of requirement “R-14” consists of the relations to the components in the rough design, the refined design, and the respective implementation as well as test cases that are used during system testing and verify the implementation of the requirement in the developed system.

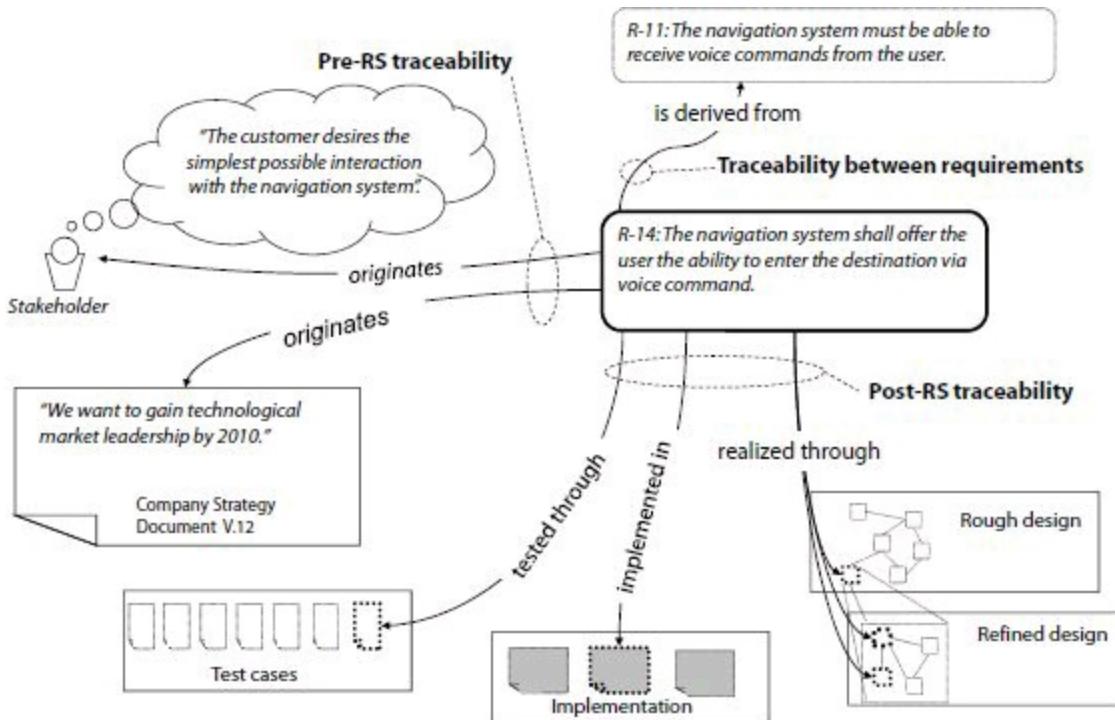


Figure 8-6 Example of the three types of requirements traceability

In addition, **figure 8-6** shows the traceability between requirements. The traceability link between requirement “R-14” and “R-11” documents that requirement “R-14” was derived from requirement “R-11”.

8.4.4 Representation of Requirements Traceability

Requirements traceability information can be represented in different ways. The most

common approaches to representing traceability are simple textual references, hyperlinks, and trace matrices and trace graphs.

Text-Based References and Hyperlinks

This simple way to represent traceability information of a requirement consists of annotating the target artifact as a textual reference in the requirement (initial artifact) or to establish a hyperlink between the initial artifact and the target artifact. When linking artifacts, different types of hyperlinks with specific link semantics can be used.

Trace Matrices

Another common technique for representing and documenting traceability information between requirements as well as between requirements and previous and posterior artifacts in the development process are trace matrices. The rows in a trace matrix contain the initial artifacts (requirements). In the columns, the target artifacts (e.g., sources of requirements, development artifacts, requirements) are represented. If a trace link exists between an initial artifact in row n and a target artifact in column m , cell (n, m) is marked in the trace matrix.

Interpretation of a trace matrix

[Figure 8-7](#) shows a simple trace matrix for the trace relation “derived” that exists between two requirements. An entry in the matrix specifies that a trace link of type “derived” exists from a requirement “Req- n ” to another requirement “Req- m ” such that “Req- n ” was derived from “Req- m ”.

		Target artifacts					
		derived	Req-1	Req-2	Req-3	Req-4	Req-5
Initial artifacts	Req-1		X				
	Req-2			X			
	Req-3						X
	Req-4			X			
	Req-5						

Figure 8-7 Representation of traceability information in a trace matrix

Maintainability of trace matrices

In practice, it became apparent that trace matrices are difficult to maintain as the number of requirements increases. A trace matrix that, for example, documents the

refinement relations between merely 2,000 requirements contains over four million cells. In addition, many trace matrices must be created in order to be able to represent the available information cleanly (e.g., with regard to different types of traceability links).

Trace Graphs

A trace graph is a graph in which all nodes represent artifacts and all edges represent relationships between artifacts. The distinction between different artifacts and types of traceability can be realized by means of assigning different attributes to the nodes and edges of the graph.

Trace graph over different development artifacts.

Figure 8-8 shows the representation of traceability information in a simple example. In the trace graph, a node type is defined for each type of artifact (context information “C”, requirements “Req-n”, components “Comp-n”). In addition, three types of edges are defined to represent three types of traceability relations (“realized through”, “is origin”, “refines”).

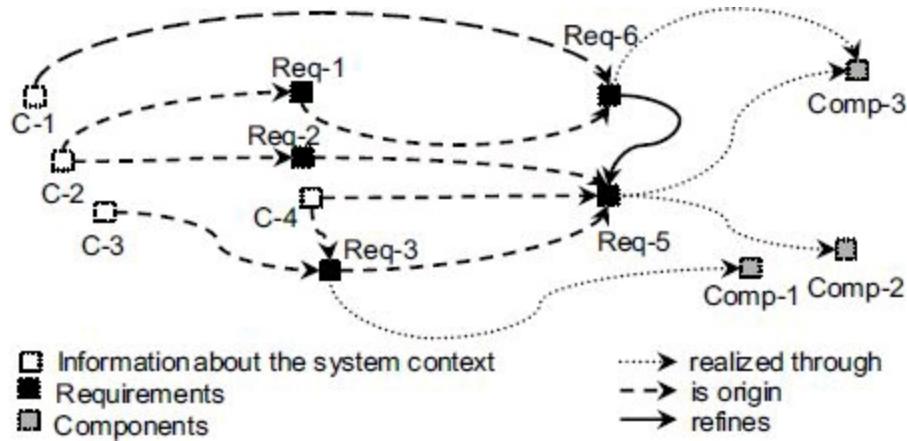


Figure 8-8 Representation of traceability in a trace graph (extract)

Traceability chains

If traceability information about previous artifacts (e.g., stakeholders and interview protocols) as well as posterior artifacts (e.g., test cases and components) must be managed, traceability chains for the respective requirement can be created at different levels, up to a trace of the requirement over the entire life cycle of the system. Common tools to maintain requirements allow for the definition of representation levels when creating traceability chains so that, depending on the selected level, only immediate relations of a requirement or entire traceability chains for the requirement can be generated and displayed. The traceability chains are the foundation for a

comprehensive impact analysis during requirements change management.

8.5 Versioning of Requirements

During the life cycle of a system, the requirements of the system change as new requirements are added and existing requirements are removed or altered. The reasons for changes in requirements are diverse. One possible reason is, for instance, the fact that stakeholders learn more and more about the system as requirements engineering progresses. As a result, new and altered requirements come to their mind. Due to these changes, a suitable versioning of requirements is strongly advisable.

Subject of version control

Versioning of requirements aims at providing access to the specific change states of individual requirements over the course of the life cycle of the system. The version of a requirement is defined by its specific content of the change state and is marked by a unique version number. The information that is subject to version management can be single text-based requirements, sentences, sections of requirements documents, or entire requirements documents, but also requirements models and partial requirements models.

8.5.1 Requirements Versions

When versioning requirements, one can distinguish between the version and the increment of the version number. For example, the version number 1.2 references a requirement with version 1 and the increment 2.

[Figure 8-9](#) illustrates the method of assigning version numbers. As shown in the figure, with smaller changes regarding the content, the increment is increased by one. If larger changes are performed, the version number is incremented. If the version number is increased, the increment is set to the initial value (0). A v can be added in front of the version number to make it more understandable and easier to identify as such.

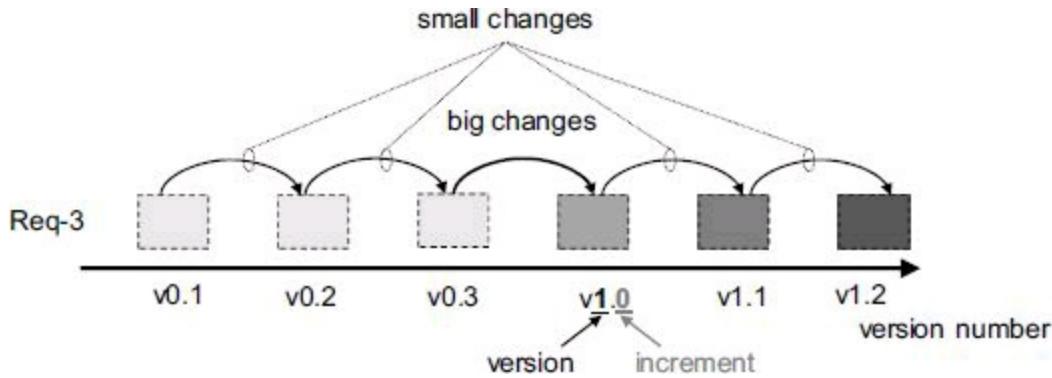


Figure 8-9 Requirements versions

Along with the rather simple structuring by means of version numbers, and the proposed method of versioning requirements, other methods of assigning version numbers are widely used. For example, it is possible to distinguish between the version identifier, the increment identifier, and the sub-increment identifier (v1.2.12).

8.5.2 Requirements Configurations

A requirements configuration consists of a set of requirements with the additional condition that each selected requirement is present in the requirements configuration with exactly one version, identified by the version number.

Dimensions of configuration management of requirements

Managing configurations of requirements can be described in two dimensions [Conradi and Westfechtel 1998]: In the product dimension, configuration management deals with individual requirements within the requirements base (foundation). In the version dimension, configuration management considers the various change states as part of version management within the product dimension. [Figure 8-10](#) illustrates both dimensions of configuration management of requirements. On the requirements axis, requirements are represented. On the version axis, the different versions of the requirements are depicted.

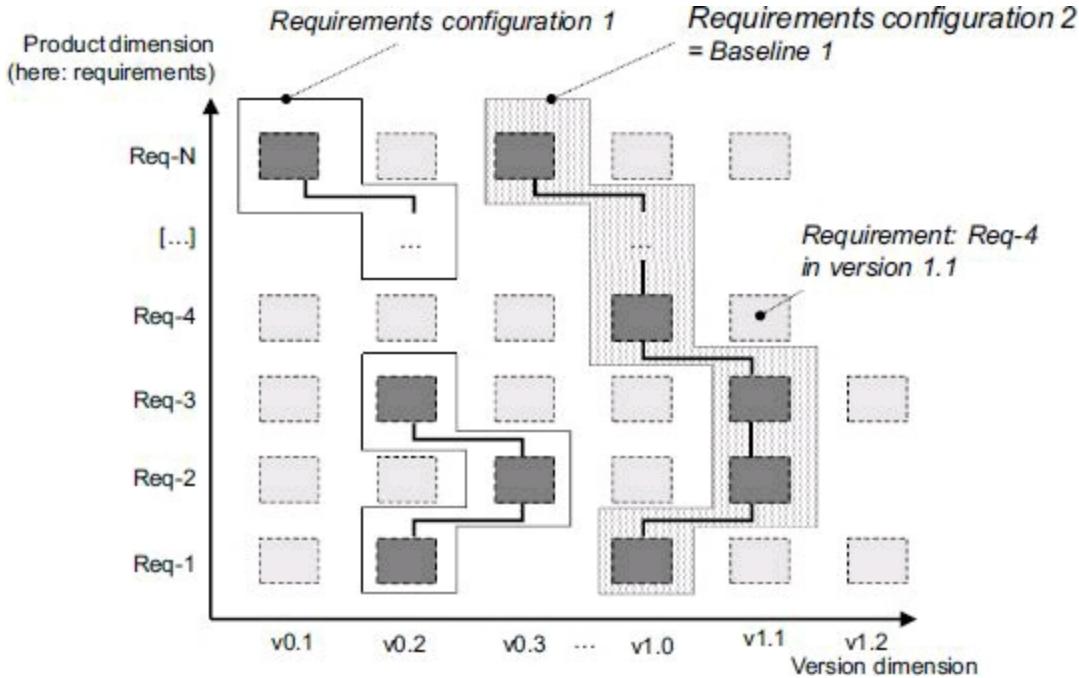


Figure 8-10 Dimensions of configuration management of requirements (based on [Conradi and Westfechtel 1998])

Properties of requirements configurations

A configuration of requirements subsumes a defined set of logically connected requirements (more precisely, versions of requirements), where each requirement of the requirements base may occur at most once in the requirements configuration. A requirements configuration does not need to contain a version of every requirement that is considered in the product dimension (see figure 8-10, requirements configuration 1). A configuration of requirements has the following properties:

- *Logical connection*: The requirements contained in a configuration are directly logically connected to one another, i.e., a goal-oriented grouping of the requirements to a common configuration has been performed.
- *Consistency*: The requirements contained in a configuration do not contradict one another, i.e., the configuration contains requirements that are contradiction free in their respective version.
- *Unique identification*: A configuration has a unique identifier (ID) which can be used to uniquely identify the configuration.
- *Immutable*: A configuration defines a certain, immutable state of the specification. If requirements of a configuration are changed, a new version of the requirements and potentially of the configuration is the result.
- *Basis for rollbacks*: If changes of requirements must be undone, configurations offer the ability to roll back requirements to a specific version within a

configuration. Therefore, a consistent state of the specification can be maintained.

8.5.3 Requirements Baselines

Configuration vs. baseline

Requirements baselines are specific configurations of requirements that typically comprise stable versions of requirements and, also, often define a release of a system. Due to that property, requirements baselines are usually visible externally (e.g., to the contractor). When requirements baselines are used, a number of important activities in the development process are supported:

- *Basis for release planning:* Requirements baselines are configurations of “stable” requirements, specially marked for the contractor. Baselines therefore serve as the basis of communication for the planning of system releases as well as their definition.
- *Estimation of the effort involved with implementation:* As baselines of requirements can be used for the definition of system releases, they can also be used to estimate the effort needed to realize a system release. This can be done by estimating the partial effort involved with implementing a requirement of the baseline and summing up the total effort for the remaining baseline.
- *Comparison to competing products:* Requirements baselines can be used to compare the planned system to competing systems.

8.6 Management of Requirements Changes

Requirements change over the course of the entire development and life cycle of a system. This means that new requirements are added and existing requirements are changed or removed.

8.6.1 Requirements Changes

Reasons for changes

The reasons for changes in requirements can be multifarious. Along with changes that stem immediately from errors or incomplete requirements, the evolution of the context can make it necessary to change the requirements. For example, changes in the

stakeholders' desired application of the system, amendments to a law, new technologies, or additional competition in the market can influence the requirements and make changes necessary. Changes in requirements, however, can also stem from system failure after the system was deployed if an error in the requirements can be held responsible for the failure.

Changes per se are not negative.

Changes in requirements per se are not negative. They are merely an indication that stakeholders deal closely with the system and learn more and more about its functions, qualities, and restrictions. If change requests only occur infrequently during development of the system, it may be a sign of low stakeholder interest in the system to be developed.

Change frequency as an indicator of process quality

However, if requirements changes occur very frequently, the development of a system that is in agreement with all involved stakeholders becomes nearly impossible. A high change frequency is, among other things, an indicator for inadequately performed requirements engineering activities, such as elicitation and negotiation techniques. In addition, a high change frequency takes up a lot of resources in the development project.

8.6.2 The Change Control Board

Over the course of the system life cycle, it is necessary to channel change requests for requirements and define a structured process that will lead to a justified decision about whether a change request is approved and how it is approved. Changes can pertain to individual requirements (e.g., redefining a requirement) or the entire requirements document. The evaluation of requirements changes, as well as the decision about performing the change, is usually the responsibility of a change control board. The change control board (CCB) typically has the following tasks:

Tasks of the change control board

- Estimate the effort for performing the change (potentially commission a third party with an effort analysis).
- Evaluate change requests, e.g., with respect to the effort/benefit ratio.
- Define requirement changes or define new requirements on the basis of change requests.

- Decide about acceptance or rejection of change requests.
- Classify incoming change requests.
- Prioritize accepted change requests.
- Assign accepted change requests to change projects.

Representatives in the change control board

In some cases, the CCB may want to delegate these tasks to another party. Decisions about changes have to be negotiated and agreed upon with the contractor and all involved stakeholders in the development project. Therefore, the change control board should consist of, among others, the following stakeholders, depending on the properties of the system to be developed and the development process:

- Change manager
- Contractor
- Architect
- Developer
- Configuration manager
- Customer representative
- Product manager
- Project manager
- Quality assurance representative
- Requirements engineer

The role of the change manager

The chairperson of the change control board is the change manager. The change manager has the task, among other things, of mediating between parties in case of conflicts and to negotiate decisions with the respective parties. In addition, the change manager is responsible for communicating and documenting decisions.

8.6.3 The Change Request

Template for change requests

In order to be able to manage changes of requirements during requirements engineering, they have to be documented in a purpose-oriented manner. A change request documents the desired change and contains additional information for the

management of the change request.

A change request should contain the following information:

Change information

- *Identifier*: The identifier makes it possible to uniquely identify a change request at any point during the life cycle of the system.
- *Title*: The title summarizes the key concern of the change request in one brief statement.
- *Description*: The description documents the requirement change as precisely as possible. It can contain information on the effect of the changes as well.
- *Justification*: The most important reasons as to why the change is necessary are listed here.
- *Date filed*: The date at which the change request was filed.
- *Applicant*: The name of the person that issued the change request.
- *Priority (in the applicant's opinion)*: The importance of the change request according to the applicant's opinion.

Management information for the change request

In addition to the preceding change information, the following information for requirements change management is helpful:

- *Change validator*: The person that verifies if the change has been performed correctly.
- *Impact analysis status*: Flags whether an impact analysis has already been performed on the change request.
- *CCB decision status*: Flags whether the change control board has already decided upon the change request.
- *CCB priority*: Documents the priority of the change request assigned by the change control board.
- *Responsible*: Documents the person that is in charge of performing the change request.
- *System release*: Documents in which system release the changed requirement shall be implemented.

8.6.4 Classification of Incoming Change Requests

Corrective, adaptive, and exceptional changes

After it has been filed, the change request is classified by the change manager and the change control board. Typically, the change manager pre-classifies incoming change requests that will be introduced, adapted (if necessary), and finally approved (or rejected) during the next change control board meeting. A change request can be classified according to the following three categories:

- *Corrective requirement change*: A change request is classified thusly if the reason for the change request is a failure of the system during its operation that can be attributed to an error in the requirements.
- *Adaptive requirement change*: A change request is thusly classified if a requested change requires the system to be amended. A possible reason for an adaptive requirement change can be a change in the system context, e.g., a new technology is available or the system boundary was altered (see section 2.2).
- *Exceptional change (hotfix)*: A change request is classified as an exceptional change if the change must absolutely immediately be done at all costs. Exceptional changes can be either corrective or adaptive.

Different processing methods

The method for processing requirements changes depends on their classification. For example, exceptional changes must be analyzed, evaluated, decided, and potentially implemented right away. Contrastingly, adaptive requirement changes are often processed in batches at a later point in time, typically as soon as the next (or some subsequent) system release is imminent. On the other hand, corrective requirement changes are usually analyzed, evaluated, and if necessary implemented rather promptly after the change request has been filed.

8.6.5 Basic Method for Corrective and Adaptive Changes

Figure 8-11 illustrates the principal method of handling change requests. This method can be tailored depending on organizational and project-specific particularities.

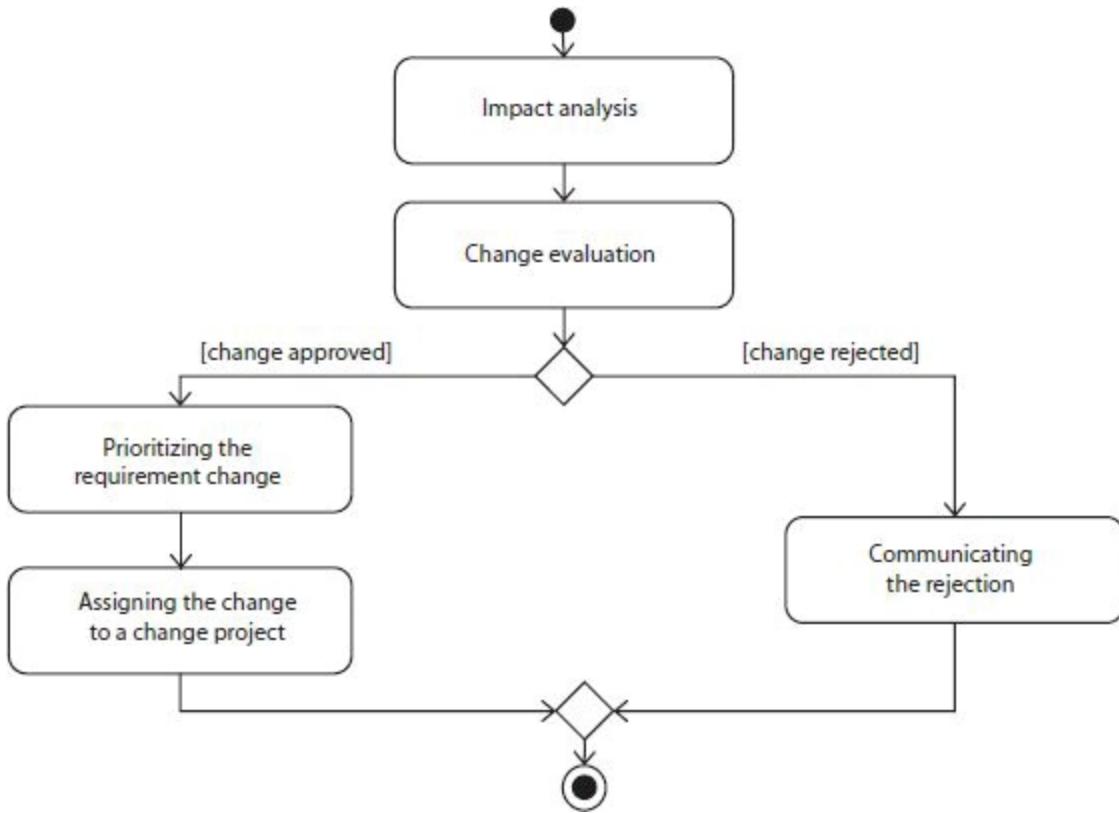


Figure 8-11 Method for handling change requests

Impact analysis

During impact analysis, the effort for performing the change is estimated. In order to do so, all requirements affected by the change are sought out, including any newly defined requirements. Afterward, the posterior development artifacts that potentially will have to be changed or redeveloped are identified (e.g., test cases or components). For each affected artifact, the effort for implementing the change is determined and the total effort for the change is computed by summing up all partial efforts.

The consistent integration of the changes into the requirements basis often only negligibly influence the total effort. The most significant portion of the total effort is usually generated by the necessary adaptations of the posterior development artifacts.

Using traceability information

Identifying those requirements and posterior development artifacts that are affected by a requirements change can be automated or at least supported by means of traceability information. If no or not all necessary traceability information is available, domain experts or experts of the development team should be questioned with respect to the consequences of the change request filed.

Evaluating a change

After the impact analysis has been completed, the change control board evaluates the change filed. In order to do that, cost and benefit are compared and evaluated with regard to the available resources. For example, the benefit of the change can be the avoided loss in prestige, improved market position, or avoided contract penalties.

Implementing approved changes

In the next step, approved changes are prioritized by the change control board. Afterward, the requirements changes are assigned to a change project or the next (or any subsequent) system release for implementation.

Validating the requirement changes

Planning, control of the implementation, and validation of the successfully applied changes are typically the responsibility of the change manager or of the change control board and may be delegated, of course.

8.7 Measurement of Requirements

Metrics can be used to assess the quality of requirements and the requirements engineering process. A metric can be used to measure one or more properties of requirements or of the requirements engineering process. The measurement results obtained by using metrics are indicators of the product and process quality.

8.7.1 Product vs. Process Metric

We thus differentiate between two types of metrics:

- Product metrics, used to obtain insights regarding the amount and quality of the documented requirements and requirements documents
- Process metrics, used to obtain insights regarding the progress and quality of the requirements engineering process

8.7.2 Examples of Product and Process Metrics

A typical example of a process metric used in requirements engineering is a metric used to measure the “requirements changes” over a period of time (e.g., already agreed-upon requirements that have been changed within one month or week).

A typical example of a product metric used in requirements engineering is a metric used to measure the “number of requirements errors” identified in a requirements specification at a given point in time. Typically, the error rate is calculated as a relative value, for example, per 100 pages of the specification or per 1000 requirements.

The rate of requirements errors is primarily an indicator of the quality of the requirements documents produced. Moreover, it is also an indicator of the quality of the requirements engineering process.

8.8 Summary

Requirements management is a core activity of requirements engineering. It’s the aim of this activity to maintain persistent availability of the documented requirements as well as other relevant information over the course of the entire system or product life cycle, to structure this information in a sensible manner (e.g., by means of requirements attributes), and to ensure selective access to this information. The management of requirements comprises techniques of the following categories:

- *Assigning attributes to requirements*: In order to allow for requirements management, properties of requirements are documented by means of requirements attributes.
- *Prioritizing requirements*: Requirements are prioritized at different points in time, during different activities, and according to different criteria. Depending on the goal of prioritization and the subject of prioritization, different prioritization techniques are to be used.
- *Traceability of requirements*: During requirements management, traceability information of requirements is recorded, organized, and maintained so that information about cross references and dependencies between requirements or between requirements and other development artifacts can be used.
- *Versioning of requirements*: Versioning and configuring requirements makes it possible to keep information about specific developmental states of requirements and requirements documents available over the course of the life cycle of the system or the product.
- *Management of requirements changes*: Usually, the change control board is responsible for processing change requests. The change control board decides if a change request is approved or rejected and prioritizes it. The board also performs an impact analysis to estimate the impact of the change on all requirements and development artifacts as well as the resources necessary for

implementing the change.

- *Measurement of requirements*: Product and process metrics can be used to measure the quality of the requirements and the requirements engineering process.

9 Tool Support

The different activities of requirements engineering should be supported by adequate tools that ideally integrate and continue processing the already existing information. This information could have been generated during requirements engineering (e.g., natural language or model-based requirements) or could have been used as the basis for requirements (e.g., conversation minutes, goal documents, lists of stakeholders). In practice, the most commonly known tools for requirements engineering are tools that support the management of requirements (see [chapter 8](#)). This chapter primarily considers requirements management tools (RM tools, for short). Along with RM tools, there are also tools in requirements engineering that support the elicitation, documentation, negotiation, and validation of requirements.

9.1 General Tool Support

Tools during system development

A great number of tools that are being used during system development can also be used during requirements engineering. In that sense, test management, bug tracking, or configuration management tools often offer the ability to manage requirements or have the ability to be extended to do so. One advantage of using such tools for requirements management is that requirements can be well integrated with the artifacts the tools were originally designed to create, like test cases or change requests. For example, if requirements are managed using a test management tool and not a distinct RM tool, an interface between two tools can be omitted and tracing test cases and their respective requirements becomes much simpler.

Support through wiki technologies

Wiki technologies are nowadays also used to support requirements engineering. For instance, glossaries can be authored collaboratively or system requirements can be worked on in cooperation using wiki technologies. Especially in case of systems with a large number of stakeholders, wikis have proven themselves as exceedingly useful in practice.

Tools to structure, present, visualize, and simulate

Tools of other tool categories can help increase the effectiveness and efficiency of requirements engineering. Mind maps that have been developed during brainstorming sessions can serve as a structuring aid, and presentation tools can help in designing a rough analysis concept. If prototypes are used, simulation tools or test environments can help to simulate the operation of the system. Tools to design prototypical user interfaces (GUI prototypes) or development environments can illustrate user interfaces and functions and serve as a basis for discussion. Flow charting tools and visualization programs can be used to generate different diagrams and graphics.

Communication, office, and project management tools

Also, tools that are commonplace in everyday work scenarios, such as office suites, can be used gainfully in requirements engineering. Mail clients, chat software, address books, calendar applications, and group-ware platforms as well as tools for project management, planning, and project controlling are everyday work tools that can aid requirements engineering. These tools support stakeholders in the communication, planning, and coordination of their tasks.

9.2 Modeling Tools

Along with natural-language-based information, in requirements engineering information is also documented based on models, which can be generated using modeling tools (see [chapter 6](#)). These tools do not only offer the ability to create the models, they often also allow analyzing the models for syntactic correctness.

When choosing modeling tools, it is important to adhere to criteria similar to those for specialized requirements management tools (see section 9.5). The modeling tool must provide a unique ID to each model element to support traceability between the different models and allow for multi-user manipulation. In addition, modeling tools should offer some kind of version control functionality with regard to the models and the model elements.

Traceability between multiple tools

An important aspect related to the application of different tools is the integration and traceability between artifacts of the different tools (e.g., use cases, behavior models, and test cases). The choice of the modeling tool or the RM tool ought to be made with regard to the interface between both tools. That means that an interface should either be already present or be easy to create. Such an interface should allow for tracing changes in models and/or requirements and for managing the traces

between models and requirements (see chapter 8). If requirements change, it is indispensable to make the necessary changes in the associated model elements as well. Similarly, if a model changes, the necessary changes must be integrated into the natural language requirements as well.

9.3 Requirements Management Tools

Necessary properties of RM tools

To support requirements management techniques (as described in chapter 8) most optimally, a RM tool should have the following basic properties:

- Manage different information (e.g., natural language requirements, conceptual models, sketches, test plans, change requests)
- Manage logical relationships between information (traceability, e.g., between requirements or between requirements and their implementation)
- Allow for unique identification (e.g., a unique ID for every managed artifact)
- Edit the managed information (multi-user accessibility, access control, configuration and version management)
- Allow for different views on the managed information, depending on the purpose
- Organize the managed information (grouping, hierarchically structuring, assigning attributes, and annotation of additional information)
- Generate reports or summaries regarding the managed information (e.g., reports of change requests for requirements)
- Generate different kinds of output documents based on the managed information (e.g., generate requirements documents for a specific system release)

Depending on the amount of functions and depending on what the basic functions cover, requirements management tools can be categorized in two ways:

- Specialized tools
- Standard office applications

9.3.1 Specialized Tools for Requirements Management

Tools of this category have been developed specifically to support requirements management techniques and govern any tasks associated therewith. Characteristic

properties of such tools are as follows (see [chapter 8](#)):

Characteristic RM tool properties

- Management of requirements and attributes on the basis of information models
- Organization of requirements (by means of hierarchy levels)
- Configuration and version management on requirement level
- Definition of requirement baselines
- Multi-user accessibility and management (e.g., access control)
- Traceability management
- Consolidation of elicited requirements (e.g., generation of views)
- Change management support (change control)

Architecture of RM tools

The different RM tools that are available on the market possess a similar structure. The most common tools have a user interface that the user can use to access all functions necessary to carry out the requirements management tasks. The managed data is stored in a database and can be edited using an integrated editor. Different import and export functions for documents ensure that imported data from external systems can be read by the RM tool and exported data can be read by external systems.

Suitability of RM tools

Such requirements management tools thus cover most of the basic functions. They are very well suited to managing the relevant information for requirements engineering. An overview of the products that support requirements engineering and that are available on the market can be found, for example, on the website of the INCOSE and of the Volere process.

9.3.2 Standard Office Applications

In many projects, standard office applications are still used to manage requirements (e.g., word processors and spreadsheet calculators). The main reasons for this are that on one hand, such applications are very widely distributed, and on the other hand, no additional effort must be spent to become familiar with them. In conjunction with using templates – like, for instance, templates for requirements documentation (see section 5.2) – these applications are suited for documenting and, to some extent, for managing requirements (e.g., traceability relations can be established by means of hyperlinks).

Office applications give only little support.

However, such tools support the basic functions of requirements management only to a limited extent. They do not offer a version control mechanism on the level of requirements, nor do they have supporting features for specific techniques for requirements management (e.g., the ability to maintain traceability links between individual artifacts in an automated way). Some of the basic functions can be emulated using other tools. For instance, an office application that is used in conjunction with some version control tool may fulfill the requirement of active version control or managed multi-user access. Nevertheless, the productivity and performance with regard to requirements management that can be achieved with specialized tools cannot be achieved using standard office applications.

9.4 Introducing Tools

Assign responsibility.

Before any effort can be spent on finding a tool that supports requirements management in the best possible manner, responsibilities regarding requirements engineering should already have been delineated in the organization or in the project. In addition to the parties responsible, the techniques and processes that are necessary to achieve the goal of requirements engineering and requirements management (see [chapter 8](#)) must be defined. After all, even the most sophisticated requirements management tool is but an aid for the requirements engineer and requirements engineering.

The tool follows the method.

Only when every process and every technique has been defined and all involved people are able to follow these constraints can an evaluation of the available tools be performed. The following considerations have to be factored in when choosing and introducing tools for requirements engineering:

Consider necessary resources.

- The choice and introduction of tools takes up resources in the organization. This holds not only for personnel entrusted with the introduction of a tool, but also for the future users of a tool. These efforts have to be considered during evaluation.

Pilot project

- In practice, it has proven problematic to introduce a tool while a development

project is already in progress. While additional effort for instruction of the employees can be estimated rather well, the risks that are associated with introducing a new tool while a project is in progress are easily underestimated. Employee resistance or deficiencies of the tool that become apparent when the tool is deployed can influence the project negatively. Such risks can be avoided by introducing new tools in pilot projects. In this pilot project, additional resources for tool introduction, employee instruction, and process tailoring should be factored in.

Evaluation

- A suitable tool should be determined in the context of a tool evaluation. When manufacturers are surveyed and critical “must-have” criteria are defined, potential candidates for introduction can be selected and investigated in further detail. In order to do that, a catalogue of criteria must be created that describes which requirements a tool for requirements engineering must fulfill. The tools that remain to be evaluated can then be rated according to these requirements.

Costs

- Costs for a tool usually exceed licensing cost alone. Typically, costs for employee instruction as well as potential tool customization and costs for support must be taken into account as well.

Instruct employees.

- It is necessary for the future users of the tool to know, actively shape, and master the processes and activities that they encounter during requirements engineering. The users must be instructed with regard to processes, techniques, and the respective tool support.

9.5 Evaluating Tools

Due to the many different kinds of tools that are available, evaluating tools with regard to their adequacy to support requirements engineering is very tedious and challenging in practice.

Views on tools in requirements engineering

To evaluate the tools as objectively as possible, different views on the tools in

requirements engineering should be adopted. By defining different tool views, it is possible to analyze the adequacy of a tool systematically and to prioritize the tool requirements individually. [Figure 9-1](#) shows views that could be used to evaluate tool adequacy in requirements engineering.

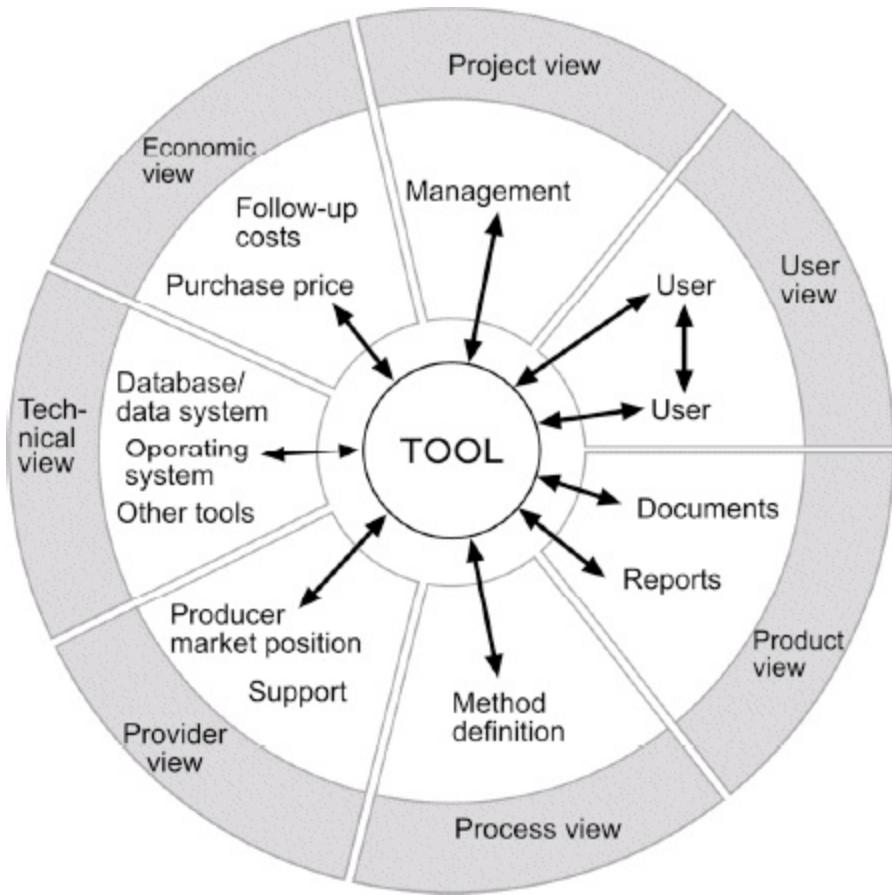


Figure 9-1 Views on a requirements engineering tool

For each view, criteria should be defined that are tailored toward the core aspects of the respective perspective.

9.5.1 Project View

Project support

The project view shows the extent to which the tool can support the project. Relevant criteria are support during project preparation, project planning, and project execution. With regard to project preparation, criteria can be considered that pertain to the definition of project-specific information types and documents. With regard to project planning, the scope of defined milestones as well as how information and documents that are created by means of the tool pertain to the milestones. Project execution comprises criteria that pertain to the scope of project control and project

lead on the basis of information and documents that are created with the tool.

9.5.2 User View

Perspective of the future users

The user view considers the requirements for the tool that emerge out of the perspective of the users (e.g., multi-user capability). The evaluation from the perspective of the user is focused on tool usage, mapping of roles, and support of group work. In detail, this means that the different stakeholders that are involved in a development project must be adequately mapped by appropriate user management and access rights management. This enables the users to gain the appropriate access to the tool functions and the stored information, depending on their respective role.

9.5.3 Product View

Tool functions

The product view contains the functionalities that the tool possesses (e.g., different documentation types for requirements). Among other things, the supported document types, views, and reports that can be generated, as well as traceability between the selected products, are considered in this view.

9.5.4 Process View

Method support offered by the tool

The process view focuses on the method support offered by the tool (e.g., possible guidance, maintenance of traceability relations). Considerations of the process view comprise the ability to document activities within the tool as well as the extent to which the tool offers method guidance. With regard to method guidance, different degrees of obligation can be distinguished. Method guidance can be strict and restrictive or offer more lenient suggestions and hints. Along with the degree of method support that is offered by the tool, the degree to which a project-specific process model can be defined can also be considered in this view.

9.5.5 Provider View

Market position of the manufacturer and support offered

The provider view considers the market position as well as the different services that are offered by a manufacturer. When choosing a tool, not only the functional aspects but also constraints that must be fulfilled for the tool to be applicable are pertinent. The degree of brand awareness, for instance, and the reputation of the provider are therefore often used as decision criteria. Due to the relatively high acquisition cost and the long-term subscriptions to support services, a close commitment toward the provider is made.

9.5.6 Technical View

The tool's ability to perform and to integrate

The technical view involves technical context conditions that the system is expected to meet. Important aspects in the technical view are, for instance, the ability to integrate the tool, the performance of the used repository, the necessary hardware and software, and scalability of the tool. The ability of the tool to integrate can be determined, for instance, by investigating to what extend the functionalities of the tool are accessible via an API and to what degree the process, data, and control integration is possible. The scalability of the tool can be determined, for instance, by determining the maximum number of users that can be maintained or the maximum number of objects (e.g., content packages or documents). The performance of the repository used can be measured by determining the degree to which importing and exporting data can be done as well as by determining the performance of the query interfaces or the available security concepts.

9.5.7 Economic View

Introduction and follow-up costs

The economic view regards the possible costs that arise due to the acquisition, introduction, and maintenance of a tool (e.g., licensing costs, employee instruction costs, and support costs). The amount of the relevant costs can consist of the integration costs, costs of operation, maintenance and infrastructure, costs for method tailoring, and acquisition costs.

9.6 Summary

When managing requirements during requirements engineering, it is necessary to store the information in a way that the quality criteria for requirements management are met. Tools support the requirements engineer in doing so. These tools can be differentiated into professional RM tools, modeling tools, and standard office applications and differ from one another in the functionalities that are offered to the requirements engineer. This is the reason an evaluation must be done before a tool is selected, so as to not inhibit the introduction process unnecessarily.

References

- [Akao 1990] Y. Akao: Quality Function Deployment – Integrating Customer Requirements into Product Design. Productivity Press, Portland, 1990.
- [Bandler 1994] R. Bandler: Metasprache und Psychotherapie: Die Struktur der Magie I. Junfermann, Paderborn, 1994.
- [Bandler and Grinder 1975] R. Bandler, J. Grinder: The Structure of Magic II. Science and Behaviour Books, Palo Alto CA, 1975.
- [Basili et al. 1996] V. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sörumsgard, M. Zelkowitz: The Empirical Investigation of Perspective-Based Reading. Empirical Software Engineering, Vol. 1, No. 12, Springer-Verlag, Berlin, Heidelberg, 1996, pp. 133–144.
- [Beck 1999] K. Beck: Extreme Programming Explained – Embrace Change. Addison-Wesley, Reading MA, 1999.
- [Boehm 1981] B. Boehm: Software Engineering Economics. Prentice Hall, Englewood Cliffs, 1981.
- [Boehm 1984] B. Boehm: Verifying and Validating Software Requirements and Design Specifications. IEEE Software, Vol. 1, No. 1, IEEE Press, Los Alamitos, 1984, pp. 75–88.
- [Chaos 2006] Standish Group: Chaos Report, 2006.
- [Chen 1976] P. Chen: The Entity-Relationship Specification – Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1, 1976, pp. 9–38.
- [Chernak 1996] Y. Chernak: A Statistical Approach to the Inspection Checklist Formal Synthesis and Improvement. IEEE Transactions on Software Engineering, Vol. 22, No. 12, 1996, pp. 866–874.
- [Cockburn 2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, Reading, MA, 2001.
- [Conradi and Westfechtel 1998] R. Conradi, B. Westfechtel: Version Models for Software Configuration Management. ACM Computing Surveys, Vol. 30, No. 2, 1998, pp. 232–282.
- [Davis 1993] A. M. Davis: Software Requirements – Objects, Functions, and States. Prentice Hall, Englewood Cliffs, 1993.
- [DeBono 2006] E. DeBono: Edward DeBono’s Thinking Course: Powerful Tools to Transform Your Thinking. BBC Active, Harlow, 2006.
- [DeMarco 1978] T. DeMarco: Structured Analysis and System Specification. Yourdon Press, New York, 1978.
- [Dömges and Pohl 1998] R. Dömges, K. Pohl: Adapting Traceability Environments to Project-Specific Needs. Communications of the ACM, Vol. 41, No. 12, 1998, pp. 55–62.
- [Easterbrook 1994] S. Easterbrook: Resolving Requirements Conflicts with Computer-Supported Negotiation. In: M. Jirotka, J. Goguen (eds.): Requirements Engineering – Social and Technical Issues, Academic Press, London, 1994, pp. 41–65.
- [Elmasri and Navathe 2006] R. Elmasri, S. B. Navathe: Fundamentals of Database Systems. 5th Edition, Addison-Wesley, Reading MA, 2006.
- [Gause and Weinberg 1989] D. C. Gause, M. Weinberg: Exploring Requirements – Quality before Design. Dorset House, New York, 1989.

- [Gilb and Graham 1993] T. Gilb, D. Graham: Software Inspection. Addison-Wesley, Reading MA, 1993.
- [Glass and Holyoak 1986] A. L. Glass, K. J. Holyoak: Cognition. Random House, New York, 1986.
- [Glinz and Wieringa 2007] M. Glinz, R. Wieringa: Stakeholders in Requirements Engineering. IEEE Software 24, 2, 2007, pp. 18–20.
- [Gotel and Finkelstein 1994] O. Gotel, A. Finkelstein: An Analysis of the Requirements Traceability Problem. In: Proceedings of the IEEE International Conference on Requirements Engineering (ICRE'94), 1994, pp. 94–102.
- [Gottesdiener 2002] E. Gottesdiener: Requirements by Collaboration: Workshops for Defining Needs. Addison-Wesley Longman, Amsterdam, 2002.
- [Harel 1987] D. Harel: Statecharts – A Visual Formalism for Complex Systems. Science of Computer Programming, Vol. 8, No. 3, 1987, pp. 231–274.
- [Hatley and Pirbhai 1988] D. J. Hatley, I. A. Pirbhai: Strategies for Real Time System Specification. Dorset House, New York, 1988.
- [Hickey and Davis 2003] A. M. Hickey, A. M. Davis: Elicitation Technique Selection: How Do Experts Do It? Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03), Monterey Bay, USA, 2003, pp. 169–178.
- [IEEE 610.12-1990] Institute of Electrical and Electronics Engineers: IEEE Standard Glossary of Software Engineering Terminology (IEEE Std. 610.12-1990). IEEE Computer Society, New York, 1990.
- [IEEE 830-1998] Institute of Electrical and Electronics Engineers: IEEE Recommended Practice for Software Requirements Specifications (IEEE Std. 830-1998). IEEE Computer Society, New York, 1998.
- [ISO/IEC 9126] International Organisation for Standardization: Software Engineering – Product Quality – Part 1: Quality Model. Geneva, 2001.
- [ISO/IEC 15504-5] International Organisation for Standardization: An Exemplar Process Assessment Model. Geneva, 2007.
- [ISO/IEC 25010:2011] International Organization for Standardization: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, Geneva 2011.
- [ISO/IEC/IEEE 29148:2011] International Organization for Standardization: Systems and software engineering – Life cycle processes – Requirements engineering, Geneva, 2011.
- [Jacobson et al. 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Oevergaard: Object Oriented Software Engineering – A Use Case Driven Approach. Addison-Wesley, Reading MA, 1992.
- [Jones 1998] T. C. Jones: Estimating Software Costs. McGraw-Hill, New York, 1998.
- [Kano et al. 1984] N. Kano, S. Tsuji, N. Seraku, F. Takahashi: Attractive Quality and Must-be Quality. Quality – The Journal of the Japanese Society for Quality Control, Vol. 14, No. 2, 1984, pp. 39–44.
- [Karlsson and Ryan 1997] J. Karlsson, K. Ryan: A Cost-Value Approach for Prioritizing Requirements. IEEE Software, Vol. 14, No. 5, IEEE Press, Los Alamitos, 1997, pp. 67–74.
- [Keller et al. 1992] G. Keller, M. Nüttgens, A.-W. Scheer: Semantische Prozeßmodellierung auf der Grundlage »Ereignisgesteuerter Prozessketten (EPK)«. Publications of the institute for business informatics (IWi), Saarland University, Issue 89, Saarbrücken, 1992.
- [Kosslyn 1988] S. M. Kosslyn: Imagery in Learning. In: M. Gazzaniga (ed.): Perspectives in Memory Research, The MIT Press, Cambridge, 1988.
- [Kruchten 2001] P. Kruchten: The Rational Unified Process: An Introduction, Addison-Wesley, 2001.
- [Laitenberger and DeBaud 2000] O. Laitenberger, J.-M. DeBaud: An Encompassing Life Cycle Centric Survey of Software Inspection. Journal of Systems and Software, Vol. 50, No. 1, 2000, pp. 5–31.

- [Lauesen 2002] S. Lauesen: Software Requirements – Styles and Techniques, Addison-Wesley, London, 2002.
- [Lehtola and Kauppinen 2006] L. Lehtola, M. Kauppinen: Suitability of Requirements Prioritization Methods for Market-driven Software Product Development. *Software Process – Improvement and Practice*, Vol. 11, No. 1, 2006, pp. 7–19.
- [Macaulay 1993] L. Macaulay: Requirements Capture as a Cooperative Activity. In: Proceedings of the 1st IEEE International Symposium on Requirements Engineering, 1993, pp. 174–181.
- [Maiden and Gizikis 2001] N. Maiden, A. Gizikis: Where Do Requirements Come From? *IEEE Software* 18, 5, 2001, pp. 10–12.
- [McMenamin and Palmer 1988] S. M. McMenamin, J. F. Palmer: Essential Systems Analysis. Prentice Hall, London, 1984.
- [Mealy 1955] G. H. Mealy: A Method for Synthesizing Sequential Circuits. *Bell System Technical Journal*, Vol. 34, No. 5, 1955, pp. 1045–1079.
- [Mietzel 1998] G. Mietzel: Pädagogische Psychologie des Lernens und Lehrens. 5th Edition, Hogrefe-Verlag, Göttingen, 1998.
- [Moore 1956] E. F. Moore: Gedanken-Experiments on Sequential Machines. In: C. Shannon, J. McCarthy (eds.): *Automata Studies*, Princeton University Press, Princeton, 1956, pp. 129–153.
- [Moore 2003] C. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts. 3rd Edition, Jossey-Bass, San Francisco, 2003.
- [OMG 2007] OMG: Unified Modeling Language: Superstructure, Version 2.1.1. OMG document formal/2007-02-05.
- [Pohl 1996] K. Pohl: Process-Centered Requirements Engineering. Research Study Press, Advanced Software Development, Taunton, Somerset, 1996.
- [Pohl 2008] K. Pohl: Requirements Engineering – Grundlagen, Prinzipien, Techniken. dpunkt.verlag, Heidelberg, 2008.
- [Pohl 2010] K. Pohl: Requirements Engineering – Fundamentals, Principles, and Techniques. Springer, New York, 2010.
- [Pohl et al. 2005] K. Pohl, G. Böckle, F. van der Linden: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer-Verlag, Berlin, Heidelberg, New York, 2005.
- [Potts et al. 1994] C. Potts, K. Takahashi, A. Antón: Inquiry-Based Requirements Analysis. *IEEE Software* 11, 2, 1994, pp. 21–32.
- [Ramesh 1998] B. Ramesh: Factors Influencing Requirements Traceability Practice. *Communications of the ACM*, Vol. 41, No. 12, ACM Press, 1998, pp. 37–44.
- [Ramesh and Jarke 2001] B. Ramesh, M. Jarke: Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering* 27, 1, 2001, pp. 58–92.
- [Robertson 2002] J. Robertson: Eureka! Why Analysts Should Invent Requirements. *IEEE Software* 19, 4, 2002, pp. 20–22.
- [Robertson and Robertson 2006] S. Robertson, J. Robertson: Mastering the Requirements Process. 2nd Edition, Addison-Wesley, Upper Saddle River, 2006.
- [Rohrbach 1969] B. Rohrbach: Kreativ nach Regeln – Methode 635, eine neue Technik zum Lösen von Problemen. *Absatzwirtschaft* 12, Issue 19, 1969, pp. 73–75.
- [Royce 1987] W. W. Royce: Managing the Development of Large Software Systems. In: Proceedings of the 9th International Conference on Software Engineering (ICSE'87), IEEE Computer Society Press, Los Alamitos, 1987, pp. 328–338.

- [Rumbaugh et al. 2005] J. Rumbaugh, I. Jacobson, G. Booch: The Unified Modeling Language Reference Manual. 2nd Edition, Addison-Wesley, Boston, 2005.
- [Rupp 2014] C. Rupp: Requirements-Engineering und -Management – Aus der Praxis von klassisch bis agil. Hanser-Verlag, Munich, 2014. (Individual chapters also available in English on the SOPHIST website: <http://www.sophist.de>)
- [Rupp et al. 2007] C. Rupp, S. Queins, B. Zengler: UML 2 glasklar – Praxiswissen für die UML-Modellierung. Hanser-Verlag, Munich, 2007.
- [Saaty 1980] T. L. Saaty: The Analytical Hierarchy Process. McGraw-Hill, New York, 1980.
- [SEI 2006] *Software Engineering Institute*: CMMI for Development (CMMI-Dev), V1.2, Technical Report CMU/SEI-2006-TR-008 – ESC-TR-2006-008. Carnegie Mellon, Software Engineering Institute, Pittsburgh, PA 2006.
- [Shull et al. 2000] F. Shull, I. Rus, V. Basili: How Perspective-Based Reading Can Improve Requirements Inspections. IEEE Computer, Vol. 33, No. 7, 2000, pp. 73–79.
- [Sommerville 2007] I. Sommerville: Software Engineering. 8th Edition, Pearson Studium, Boston, 2007.
- [Stachowiak 1973] H. Stachowiak: Allgemeine Modelltheorie. Springer-Verlag, Vienna, 1973.
- [van Lamsweerde et al. 1991] A. van Lamsweerde, A. Dardenne, B. Delcourt, F. Dubisy: The KAOS Project – Knowledge Acquisition in Automated Specification of Software. In: Proceedings of AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence, 1991, pp. 69–82.
- [V-Modell 2004] *V-Modell*: V-Modell® XT, 2004, Entwicklungsstandard für IT-Systeme des Bundes, Bundesrepublik Deutschland, Vorgehensmodell. www.kbst.bund.de
- [Ward and Mellor 1985] P. Ward, S. Mellor: Structured Development of Real-Time Systems – Introduction and Tools. Vol. 1. Prentice Hall, Upper Saddle River, 1985.
- [Weinberg 1978] V. Weinberg: Structured Analysis. Yourdon Press, New York, 1978.
- [Wiegers 1999] K. E. Wiegers: Software Requirements. Microsoft Press, Redmond, 1999.
- [Yourdon 1989] E. Yourdon: Modern Structured Analysis. Prentice Hall, Englewood Cliffs, 1989.
- [Yu 1997] E. Yu: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97), IEEE Computer Society, Los Alamitos, 1997, pp. 226–235.

**„Grey is all theory – important is
what happens on the pitch“**

Adi Preißler

Exactly! That is why teams have to ensure the best possible preparation for the pitch. Why not let the professional coaches – the SOPHISTs – support you?

We coach you in:

- eliciting and documenting system requirements.
- detecting inconsistencies and redundancies in models.
- applying notations properly and ensuring perfect modeling.
- choosing an architecture that is still valid after the acceptance of the product.

Presentations, trainings, consulting and coaching – we provide you with different tactics.

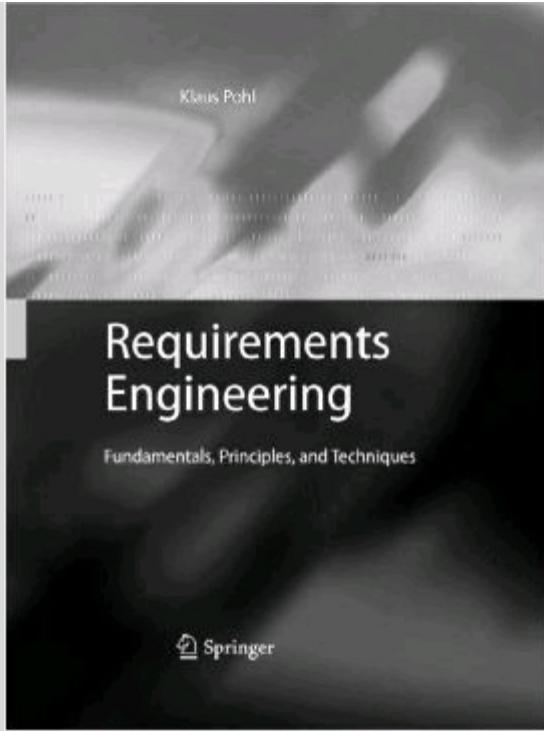
Let us keep you from running into the offside trap.

How about discussing your personal game plan?

Just call +49 911 - 40 9000.

You can also write us an e-mail to heureka@sophist.de

www.sophist.de



Klaus Pohl

Requirements Engineering

Fundamentals, Principles, and Techniques

Springer-Verlag 2010

Hardcover

814 pages

ISBN 978-3-642-12577-5

www.requirements-book.com

In this textbook, Klaus Pohl provides a comprehensive and well-structured introduction to the fundamentals, principles, and techniques of requirements engineering. He presents approved techniques for eliciting, negotiating and documenting as well as validating, and managing requirements for software-intensive systems. The various aspects of the process and the techniques are illustrated using numerous examples.

The book aims at professionals, students, and lecturers in systems and software engineering or business applications development. Professionals such as project managers, software architects, systems analysts, and software engineers will benefit in their daily work from the didactically well-presented combination of validated procedures and industrial experience.

Students and lecturers will appreciate the comprehensive description of sound fundamentals, principles, and techniques, complemented by a commented list of references for further reading. Lecturers will find additional teaching material on www.requirements-book.com.



www.paluno.uni-due.de/en



International
Requirements
Engineering
Board

Martin Glinz

Hans van Loenhoud

Stefan Staal

Stan Bühne

Handbook for the CPRE Foundation Level according to the IREB Standard

Education and Training for

Certified Professional for Requirements Engineering (CPRE)

Foundation Level

Version 1.0.0

November 2020

Terms of Use

All contents of this document, especially texts, photographs, graphics, diagrams, tables, definitions and templates, are protected by copyright. Copyright © 2020 for this handbook is with the authors. All (co-)authors of this document have transferred the exclusive right of use to IREB e.V.

Any use of the handbook or its components, in particular copying, distribution (publication), translation, or reproduction, requires the prior consent of IREB e.V.

Any individual is entitled to use the contents of the handbook within the scope of the acts of use permitted by copyright law, in particular to quote these correctly in accordance with recognized academic rules.

Educational institutions are entitled to use the contents of the handbook for teaching purposes under correct reference to the work.

Use for advertising purposes is only permitted with the prior consent of IREB e.V.

Acknowledgements

The content of this handbook was reviewed by Rainer Grau, Karol Frühauf, and Camille Salinesi. Tracey Duffy performed an English review. Stan Bühne and Stefan Sturm did the final editing.

Approved for release on November 11, 2020 by the IREB Council upon recommendation of Xavier Franch and Frank Houdek.

We thank everybody for their involvement.

Table of Contents

Table of Contents.....	3
Foreword	6
Version History	7
1. Introduction and Overview	8
1.1 Requirements Engineering: What	8
1.2 Requirements Engineering: Why	9
1.3 Requirements Engineering: Where.....	10
1.4 Requirements Engineering: How.....	11
1.5 The Role and Tasks of a Requirements Engineer.....	11
1.6 What to Learn about Requirements Engineering	11
1.7 Further Reading	12
2. Fundamental Principles of Requirements Engineering.....	13
2.1 Overview of Principles	13
2.2 The Principles Explained.....	13
2.3 Further Reading	23
3. Work Products and Documentation Practices	25
3.1 Work Products in Requirements Engineering.....	25
3.1.1 Characteristics of Work Products	25
3.1.2 Abstraction Levels.....	27
3.1.3 Level of Detail.....	28
3.1.4 Aspects to be Considered	28
3.1.5 General Documentation Guidelines	31
3.1.6 Work Product Planning.....	31
3.2 Natural-Language-Based Work Products.....	32
3.3 Template-Based Work Products	34
3.3.1 Phrase Templates	34
3.3.2 Form Templates	36
3.3.3 Document Templates.....	37
3.3.4 Advantages and Disadvantages	37
3.4 Model-Based Work Products	38
3.4.1 The Role of Models in Requirements Engineering	39

3.4.2	Modeling System Context.....	45
3.4.3	Modeling Structure and Data.....	48
3.4.4	Modeling Function and Flow	51
3.4.5	Modeling State and Behavior.....	53
3.4.6	Modeling Goals	57
3.5	Glossaries.....	58
3.6	Requirements Documentation Structures	59
3.7	Prototypes in Requirements Engineering.....	60
3.8	Quality Criteria for Work Products and Requirements	61
3.9	Further Reading	62
4.	Practices for Requirements Elaboration.....	63
4.1	Sources for Requirements.....	64
4.1.1	Stakeholders	66
4.1.2	Documents.....	70
4.1.3	Other Systems	71
4.2	Elicitation of Requirements.....	72
4.2.1	The Kano Model.....	74
4.2.2	Gathering Techniques.....	77
4.2.3	Design and Idea-Generating Techniques.....	80
4.3	Resolving Conflicts regarding Requirements	84
4.3.1	How Do You Resolve a Requirements Conflict?.....	85
4.3.2	Conflict Types.....	87
4.3.3	Conflict Resolution Techniques.....	89
4.4	Validation of Requirements.....	92
4.4.1	Important Aspects for Validation.....	93
4.4.2	Validation Techniques	95
4.5	Further Reading	99
5.	Process and Working Structure.....	100
5.1	Influencing Factors	100
5.2	Requirements Engineering Process Facets.....	102
5.2.1	Time Facet: Linear versus Iterative	103
5.2.2	Purpose Facet: Prescriptive versus Explorative.....	103
5.2.3	Target Facet: Customer-Specific versus Market-Oriented	104
5.2.4	Hints and Caveats	105

5.2.5	Further Considerations	105
5.3	Configuring a Requirements Engineering Process	106
5.3.1	Typical Combinations of Facets.....	106
5.3.2	Other RE Processes.....	109
5.3.3	How to Configure RE Processes.....	109
5.4	Further Reading	110
6.	Management Practices for Requirements	111
6.1	What is Requirements Management?	112
6.2	Life Cycle Management	113
6.3	Version Control.....	114
6.4	Configurations and Baselines	116
6.5	Attributes and Views.....	118
6.6	Traceability	120
6.7	Handling Change	122
6.8	Prioritization	124
6.9	Further Reading	127
7.	Tool Support.....	128
7.1	Tools in Requirements Engineering	128
7.2	Introducing Tools	130
7.2.1	Consider All Life Cycle Costs beyond License Costs.....	130
7.2.2	Consider Necessary Resources	130
7.2.3	Avoid Risks by Running Pilot Projects.....	130
7.2.4	Evaluate the Tool according to Defined Criteria.....	131
7.2.5	Instruct Employees on the Use of the Tool	132
7.3	Further Reading	132
8.	References	133

Foreword

This handbook provides an introduction to Requirements Engineering based on the syllabus version 3.0 for the Certified Professional for Requirements Engineering (CPRE)—Foundation Level according to the IREB standard. It complements the syllabus and addresses three groups of readers:

- ▶ *Students and practitioners* who want to learn about Requirements Engineering and take the certification exam can use this handbook as a companion book to training courses offered by training providers, as well as for self-study and individual preparation for the certification exam. This handbook may also be used to refresh existing knowledge about Requirements Engineering, for example, when preparing for a CPRE Advanced Level course and exam.
- ▶ *Training providers* who offer trainings on the CPRE Foundation Level can use this handbook as a complement to the syllabus for developing their training materials or as a study text for the participants in their trainings.
- ▶ *Professionals* in industry who want to apply proven RE concepts and knowledge in their practical work will find a wealth of useful information in this handbook.

This handbook also provides a link between the syllabus, which lists and explains the learning objectives, and the literature on Requirements Engineering. Every chapter comes with references to the literature and hints for further reading. The structure of the handbook matches the structure of the syllabus.

The terminology used in this handbook is based on the CPRE Glossary of Requirements Engineering Terminology [Glin2020]. We recommend downloading this glossary from the IREB website and use it as a terminology reference.

You find more information about the CPRE certification program, including the syllabi, glossary, examination regulations and sample exam questions on the IREB website at <https://www.ireb.org>.

Both the authors and IREB have invested a significant amount of time and effort into preparing, reviewing and publishing this handbook. We hope that you will enjoy studying this handbook. If you detect any errors or have suggestions for improvement, please contact us at info@ireb.org.

We would like to thank all people who contributed to the creation and publication of this handbook. Karol Fröhlauf, Rainer Grau and Camille Salinesi carefully reviewed the manuscript and provided valuable suggestions for improvement. Tracey Duffy did an English review. We also thank the IREB Council Shepherds of this handbook, Xavier Franch and Frank Houdek, for their feedback and support. Stefan Sturm provided encouragement and logistic support. We also thank our spouses and families for their patience and support.

Martin Glinz, Hans van Loenhoud, Stefan Staal, and Stan Bühne

November 2020

Version History

Version	Date	Comment	Authors
1.0.0	November 11, 2020	First release	Martin Glinz Hans van Loenhoud Stefan Staal Stan Bühne

1. Introduction and Overview

In this chapter, you will learn what Requirements Engineering (RE) is all about and the value that RE brings.

1.1 Requirements Engineering: What

Since the beginning of human evolution, humans have been building technical and organizational systems to *support* them in completing tasks or achieving objectives. With the rise of engineering, humans have also started to build systems that *automate* human tasks.

Whenever humans decide to build a system to support or automate human tasks, they have to figure out *what to build*. This means that they have to learn about the desires and needs of the persons or organizations who will use the system, benefit from it, or be impacted by it. In other words, they need to know about the *requirements* for that system. Requirements form the basis for any development or evolution of systems or parts thereof. Requirements always exist, even when they are not explicitly captured and documented.

The need for requirements

The term *requirement* denotes three concepts [Glin2020]:

DEFINITION 1.1. REQUIREMENT: 1. A need perceived by a *stakeholder*. 2. A capability or property that a *system* shall have. 3. A documented representation of a need, capability, or property.

Requirement

A systematically represented collection of requirements—typically for a system—that satisfies given criteria is called a *requirements specification*.

We distinguish between three kinds of requirements:

Kinds of requirements

- ▶ *Functional requirements* concern a result or behavior that shall be provided by a function of a system. This includes requirements for data or the interaction of a system with its environment.
- ▶ *Quality requirements* pertain to quality concerns that are not covered by functional requirements — for example, performance, availability, security, or reliability.
- ▶ *Constraints* are requirements that limit the solution space beyond what is necessary to meet the given functional requirements and quality requirements.

Note that dealing with requirements for projects or development processes is outside the scope of this handbook.

Distinguishing between functional requirements, quality requirements, and constraints is not always straightforward. One proven way to differentiate between them is to ask for the *concern* that a requirement addresses: if the concern is about required results, behavior, or interactions, we have a functional requirement. If it is a quality concern that is not covered by the functional requirements, we have a quality requirement. If the concern is about restricting the solution space but is neither a functional nor a quality requirement, we have a constraint. The popular rule “*What* the system shall do → functional requirement vs. *how* the system shall do it → quality requirement” frequently leads to misclassifications, particularly when requirements are specified in great detail or when quality requirements are very important.

How to distinguish between functional requirements, quality requirements, and constraints

For example, the requirement “The customer entry form shall contain fields for the customer’s name and first name, taking up to 32 characters per field, displaying at least 24 characters, left-bound, with a 12 pt. sanserif font” is a functional requirement even though it contains a lot of information about *how*. As another example, consider a system that processes the measurement data produced by the detector of a high-energy particle accelerator. Such detectors produce enormous quantities of data in real time. If you ask a physicist “What shall the system do?”, one of the first answers would probably be that the system must be able to cope with the volume of data produced. However, requirements concerning data volume or processing speed are quality requirements [Glin2007] and not functional requirements.

When people take a systematic and disciplined approach to the specification and management of requirements, we call this *Requirements Engineering (RE)*. The following definition of Requirements Engineering also reflects why we perform RE.

DEFINITION 1.2. REQUIREMENTS ENGINEERING (RE): The systematic and disciplined approach to the specification and management of requirements with the goal of *understanding the stakeholders’ desires and needs* and *minimizing the risk* of delivering a system that does not meet these desires and needs.

*Requirements
Engineering*

The concept of *stakeholders* [GIWi2007] is a fundamental principle of Requirements Engineering (see Chapter 2).

DEFINITION 1.3. STAKEHOLDER: A person or organization who influences a system’s requirements or who is impacted by that system.

Stakeholder

Note that influence can also be indirect. For example, some stakeholders may have to follow instructions issued by their managers or organizations.

Following the definition in the CPRE RE glossary [Glin2020], we use the term *system* in a broad sense in this handbook:

DEFINITION 1.4. SYSTEM: 1. In general: a principle for ordering and structuring. 2. In engineering: a coherent, delimitable set of elements that—by coordinated action—achieve some purpose.

System

Note that a system may comprise other systems or *components* as subsystems. The purposes achieved by a system may be delivered by:

- ▶ Deploying the system at the place(s) where it is used
- ▶ Selling/providing the system to its users as a *product*
- ▶ Having providers who offer the system’s capabilities to users as *services*

We therefore use the term *system* as an umbrella term which includes products, services, apps, or devices.

1.2 Requirements Engineering: Why

Developing systems (building new ones as well as evolving existing ones) is an expensive endeavor and constitutes a high risk for all participants. At the same time, systems that have practical relevance are too large for a single person to grasp intellectually. Therefore, engineers have developed various principles and practices for handling the risk when developing a system and for mastering the intellectual complexity. Requirements Engineering provides the principles and practices for the requirements perspective.

Adequate Requirements Engineering (RE) adds *value* [Glin2016], [Glin2008] to the process of developing a system:

Value of adequate RE

- ▶ RE minimizes the risk of failure or costly modifications in later development stages. The early detection and correction of wrong or missing requirements is much cheaper than the correction of errors and rework caused by missing or wrong requirements in later development stages or even after deployment of a system.
- ▶ RE eases the intellectual complexity involved in understanding the problem that a system is supposed to solve and reflecting on potential solutions.
- ▶ RE provides a proper basis for estimating development effort and cost.
- ▶ RE is a prerequisite for testing the system properly.

Typical symptoms of inadequate RE are missing, unclear, or wrong requirements due to:

Symptoms of inadequate RE

- ▶ Development teams rushing right into implementing a system due to schedule pressure
- ▶ Communication problems between parties involved—in particular, between stakeholders and system developers and among the stakeholders themselves
- ▶ The assumption that the requirements are self-evident, which is wrong in most cases
- ▶ People conducting RE activities without having adequate education and skills

1.3 Requirements Engineering: Where

Application of RE

Requirements Engineering can be applied to requirements for any kind of system. However, the dominant application case for RE today involves systems in which software plays a major role. Such systems consist of software components, physical elements (technical products, computing hardware, devices, sensors, etc.), and organizational elements (persons, positions, business processes, legal and compliance issues, etc.).

Systems that contain both software and physical components are called *cyber-physical systems*.

Cyber-physical systems

Systems that span software, hardware, people, and organizational aspects are called *socio-technical systems*.

Socio-technical systems

Depending on the perspective taken, there are different sorts of requirements.

System requirements

System requirements describe how a system shall work and behave—as observed at the interface between the system and its environment—so that the system satisfies its stakeholders' desires and needs. In the case of pure software systems, we speak of software requirements.

Stakeholder requirements

Stakeholder requirements express stakeholders' desires and needs that shall be satisfied by building a system, seen from the stakeholders' perspective.

User requirements

User requirements are a subset of the stakeholder requirements. They cover the desires and needs of the users of a system.

Domain requirements specify required domain properties of a socio-technical or cyber-physical system.

Domain requirements

Business requirements focus on the business goals, objectives, and needs of an organization that shall be achieved by employing a system (or a collection of systems).

Business requirements

The sorts of requirements as defined above match those defined in the standard [ISO29148], with the exception of domain requirements. Due to their importance, we treat domain requirements as a sort of their own. The role and importance of domain requirements are discussed in Section 2.2, Principle 4.

1.4 Requirements Engineering: How

The major tasks in RE are the elicitation (Chapter 4), documentation (Chapter 3), validation (Section 4.4), and management (Chapter 6) of requirements. Tool support (Chapter 7) can help perform these tasks. Requirements analysis and requirements conflict resolution are considered to be part of elicitation.

Major tasks of RE

However, there is no universal process that describes when and how RE should be performed when developing a system. For every system development that needs RE activities, a suitable RE process must be tailored from a broad range of possibilities. Factors that influence this tailoring include, for example:

- The overall system development process—in particular, linear and plan-driven vs. iterative and agile
- The development context—in particular, the relationship between the supplier, the customer(s), and the users of a system
- The availability and capability of the stakeholders

No universal process

There is also a mutual dependency between the requirements work products to be produced (see Chapter 3.1) and the RE process to be chosen. More details are given in Chapter 5.

1.5 The Role and Tasks of a Requirements Engineer

In practice, very few people have the job title *Requirements Engineer*. We consider people to act in the *role* of a Requirements Engineer when they:

Requirements Engineer is a role

- Elicit, document, validate, and/or manage requirements as part of their duties
- Have in-depth knowledge of RE, which enables them to define RE processes, select appropriate RE practices, and apply these practices properly
- Are able to bridge the gap between the problem and potential solutions

The role of Requirements Engineer is part of several job functions defined by organizations. For example, business analysts, application specialists, product owners, systems engineers, and even developers may act in the role of a Requirements Engineer. Having RE knowledge and skills is also useful for many other professionals—for example, designers, testers, system architects, or CTOs.

1.6 What to Learn about Requirements Engineering

The set of skills that a Requirements Engineer must learn consists of various elements. The foundational elements are covered in the subsequent chapters of this handbook.

What you will learn in this handbook

Beyond technical and analytical skills, a Requirements Engineer also needs what are referred to as soft skills: the ability to listen, moderate, negotiate, and mediate, as well as empathy for stakeholders and openness to the needs and ideas of others.

RE is governed by a set of fundamental principles that apply to all tasks, activities, and practices of RE. These principles are presented in Chapter 2.

Requirements can be documented in various forms. Various work products can be created at different levels of maturity and detail, from rather informal and temporary ones to very detailed and structured work products that adhere to strict representation rules. It is important to select work products and forms of documentation that are adequate for the situation at hand and to create the chosen work products properly. Work products and documentation practices are presented in Chapter 3.

Requirements can be elaborated (i.e., elicited and validated) with various practices. A Requirements Engineer must be able to select the practices that are best suited in a given situation and apply these practices properly. Elaboration practices are presented in Chapter 4.

Understanding possible processes and working structures enables Requirements Engineers to define a way of working that fits with the specific needs of the system development situation at hand. Processes and working structures are presented in Chapter 5.

Existing requirements can be managed with various practices. Requirements Engineers should be able to understand which requirements management practices support them for which tasks. Management practices are presented in Chapter 6.

Tools make RE more efficient. Requirements Engineers need to know how RE tools can support them and how to select a suitable tool for their situation. Tool support is discussed briefly in Chapter 7.

1.7 Further Reading

The RE terminology used in this handbook is defined in the CPRE Glossary of Requirements Engineering Terminology [Glin2020]. Glinz and Wieringa [GlWi2007] explain the notion of stakeholders. Lawrence, Wiegers, and Ebert [LaWE2001] briefly discuss the risks and pitfalls of RE.

Gause and Weinberg [GaWe1989] wrote one of the first textbooks on RE, which is still worth looking at. Pohl [Pohl2010], Robertson and Robertson [RoRo2012] and Wiegers and Beatty [WiBe2013] are popular textbooks on RE. The course notes of Glinz [Glin2019] provide a slide-based introduction to RE. The textbook by van Lamsweerde [vLam2009] presents a goal-oriented approach to RE. Jackson [Jack1995] contributes an insightful collection of essays about software requirements.

Please be aware that the official textbook for the IREB CPRE Foundation Level version 2.2 [PoRu2015] is no longer fully aligned with version 3.0 of the CPRE Foundation Level Syllabus, on which this handbook is based. However, this textbook still provides a concise introduction to RE and will be updated soon.

There are also textbooks in languages other than English. For example, Badreau and Boulanger [BaBo2014] have written an RE textbook in French. The books by Ebert [Eber2014] and Rupp [Rupp2014] are popular RE textbooks written in German.

2. Fundamental Principles of Requirements Engineering

In this chapter, you will learn about nine basic principles of Requirements Engineering (RE).

2.1 Overview of Principles

RE is governed by a set of fundamental principles that apply to all tasks, activities, and practices in RE. A *task* is a coherent chunk of work to be done (for example, eliciting requirements). An *activity* is an action or a set of actions that a person or group performs to accomplish a task (for example, identifying stakeholders when eliciting requirements). A *practice* is a proven way of how to carry out certain types of tasks or activities (for example, using interviews to elicit requirements from stakeholders).

The principles listed in Table 2.1 form the basis for the practices presented in the subsequent chapters of this handbook.

Table 2.1 Nine fundamental principles of Requirements Engineering

-
- | | |
|---|--|
| 1 | <i>Value orientation:</i> Requirements are a means to an end, not an end in itself |
| 2 | <i>Stakeholders:</i> RE is about satisfying the stakeholders' desires and needs |
| 3 | <i>Shared understanding:</i> Successful systems development is impossible without a common basis |
| 4 | <i>Context:</i> Systems cannot be understood in isolation |
| 5 | <i>Problem, requirement, solution:</i> An inevitably intertwined triple |
| 6 | <i>Validation:</i> Non-validated requirements are useless |
| 7 | <i>Evolution:</i> Changing requirements are no accident, but the normal case |
| 8 | <i>Innovation:</i> More of the same is not enough |
| 9 | <i>Systematic and disciplined work:</i> We can't do without in RE |
-

2.2 The Principles Explained

Principle 1 – Value orientation: Requirements are a means to an end, not an end in itself

The act of writing requirements is not a goal by itself. Requirements are useful—and the effort invested in Requirements Engineering is justified—only if they add *value* [Glin2016], [Glin2008], cf. Section 1.2. We define the value of a requirement as being its *benefit* minus its *cost*. The benefit of a requirement is the degree to which it contributes to building successful systems (that is, systems that satisfy the desires and needs of their stakeholders) and to reducing the risk of failure and costly rework in system development. The cost of a requirement amounts to the cost for eliciting, validating, documenting, and managing it.

Value of requirements: benefit minus costs

Reducing the risk of rework during development is a constituent part of the benefit of a well-crafted requirement. Detecting and fixing a missed or wrong requirement during implementation or when the system is already in operation can easily cost one or two orders of magnitude more than specifying that requirement properly right from the beginning. Consequently, a significant amount of the benefit of requirements comes from costs saved during the implementation and operation of a system.

Benefit from reduced rework

In other words, the benefits of RE are often long-term benefits, whereas the costs are immediate. This must be kept in mind when setting up a new project. Reducing costs in the short term by spending less for RE has a price: it considerably increases the risk of expensive rework in later stages of the project.

The *value of Requirements Engineering* can be considered to be the cumulative value of the requirements specified. As customers typically pay for systems to be implemented, but not for the requirements needed to do that, the economic value of RE is mostly an indirect one. This effect is reinforced by the fact that the benefit of requirements that stem from reduced rework costs is an indirect one: it saves costs during implementation and operation.

The economic effects of Requirements Engineering are mostly indirect ones; RE as such just costs.

To optimize the value of a requirement, Requirements Engineers have to strike a proper balance between the benefit and the cost of a requirement. For example, eliciting and documenting a stakeholder's need as a requirement eases the communication of this need among all parties involved. This increases the probability that the system to be built will eventually satisfy this need, which constitutes a benefit. The less ambiguously and the more precisely the requirement is stated, the higher its benefit, because this reduces the risk of costly rework due to misinterpretation of the requirements by the system architects and development teams. On the other hand, increasing the degree of unambiguity and precision of a requirement also increases the cost involved in eliciting and documenting the requirement.

Actually, the amount of RE required to achieve requirements with optimal value depends on numerous factors given by the specific situation in which requirements are being created and used. Obviously, the risk of building a system that eventually does not satisfy the desires and needs of its stakeholders, which may result in failure or costly rework, is *the driving force* that determines the amount of RE required. First and foremost, the criticality of every requirement should be assessed in terms of the importance of the stakeholder(s) who state the requirement (see Principle 2) and the impact of missing the requirement (Figure 2.1).

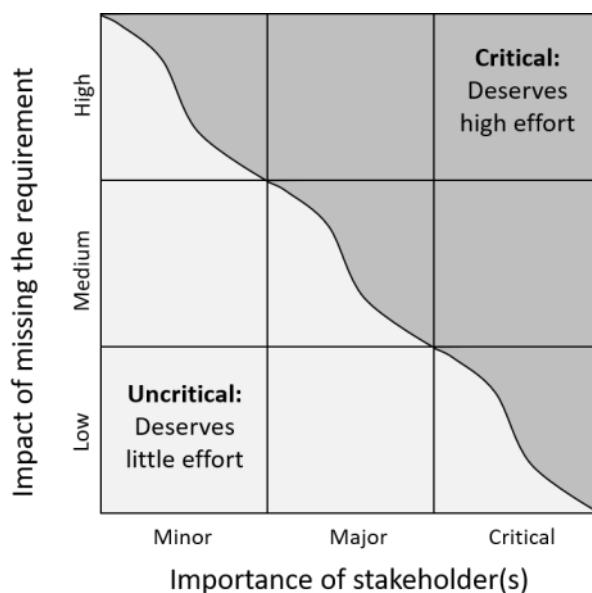


Figure 2.1 Assessing the criticality of a requirement [Glin2008]

Value of Requirements Engineering

Optimizing the value of requirements

Influencing factors

In addition, the following influencing factors should be considered:

- ▶ Effort needed to specify the requirement
- ▶ Distinctiveness of the requirement (how much it contributes to the success of the overall system)
- ▶ Degree of shared understanding between stakeholders and developers and among stakeholders
- ▶ Existence of reference systems (that can serve as a specification by example)
- ▶ Length of feedback cycle (the time between getting a requirement wrong and detecting the error)
- ▶ Kind of customer-supplier relationship
- ▶ Regulatory compliance required

We summarize this issue in two rules of thumb:

- ▶ The optimal amount of RE to be invested depends on the specific situation and is determined by many influencing factors.
- ▶ The effort invested into RE should be inversely proportional to the risk you are willing to take.

Principle 2 – Stakeholders: RE is about satisfying the stakeholders' desires and needs

The eventual goal of building a system is that the system, when it is used, solves problems that its users need to solve and satisfies the expectations of further people—for example, those who have ordered and paid for the system, or those who are responsible for security in the organization that uses the system. Therefore, we have to figure out the needs and expectations of the people who have a stake in the system, the system's *stakeholders* [GIWi2007]. The core goals of RE are *understanding the stakeholders' desires and needs* and *minimizing the risk* of delivering a system that does not meet these desires and needs; see Definition 1.2 in Section 1.2.

Every stakeholder has a *role* in the context of the system to be built—for example, user, client, customer, operator, or regulator. Depending on the RE process used, the developers of a system can also be stakeholders. This is frequently the case in agile and in market-oriented development. A stakeholder may also have more than one role at the same time. For every relevant stakeholder role, suitable people acting in this role must be selected as representatives.

Stakeholder roles

For stakeholder roles with too many individuals or when individuals are unknown, *personas* (fictitious characters that represent a group of users with similar characteristics) can be defined as a substitute. For systems that are already in use, users who provide feedback about the system or ask for new features should also be considered as stakeholders.

Personas

It makes sense to classify the stakeholders into three categories with respect to the degree of influence that a stakeholder has on the success of the system:

- ▶ *Critical*: not considering these stakeholders will result in severe problems and probably make the system fail or render it useless.
- ▶ *Major*: not considering these stakeholders will have an adverse impact on the success of the system but not make it fail.
- ▶ *Minor*: not considering these stakeholders will have no or minor influence on the success of the system.

Classifying stakeholders

This classification is helpful when assessing the criticality of a requirement (see Figure 2.1) and when negotiating conflicts between stakeholders (see below).

It is not sufficient to consider only the requirements of end users and customers. Doing this would mean that we might miss critical requirements from other stakeholders, which can easily lead to development projects that fail or overrun their budgets and deadlines.

Involving the right people in the relevant stakeholder roles is crucial for successful RE.

Practices for identifying, prioritizing, and working with stakeholders are discussed in Chapter 4.

Stakeholders in different roles naturally have different viewpoints [NuKF2003] of a system to be developed. For example, users typically want a system to support their tasks in an optimal way, the managers who order the system want to get it at a reasonable cost, and the organization's chief security officer cares primarily about the security of the system. Even stakeholders in the same role may have different needs. For example, in the group of end users, casual users have user interface requirements that may differ strongly from those of professional users.

As a consequence, it is not sufficient to just collect requirements from stakeholders. It is vital to identify inconsistencies and conflicts between the requirements of different stakeholders and to resolve these, be it by finding a consensus, by overruling, or by specifying system variants for stakeholders who factually have different needs; see Section 4.3.

Principle 3 – Shared understanding: Successful systems development is impossible without a common basis

System development, including RE, is a multi-person endeavor. To make such an endeavor a success, the people involved need a *shared understanding* of the problem and the requirements that stem from it [GIFr2015].

RE creates, fosters, and secures shared understanding between and among the parties involved: stakeholders, Requirements Engineers, and developers. We distinguish between two forms of shared understanding:

- *Explicit shared* understanding is achieved through carefully elicited, documented, and agreed requirements. This is the primary goal of RE in plan-driven processes.
- *Implicit shared* understanding is based on shared knowledge about needs, visions, context, etc. In agile RE, when requirements are not fully specified in writing, reliance on implicit shared understanding is key.

Both implicit and explicit shared understanding may be *false*, meaning that people believe that they have a shared understanding of an issue but in fact interpret this issue in different ways. Therefore, we can never rely blindly on shared understanding. Instead, the task of RE is to create and foster shared understanding and also secure it—that is, assess whether there is a true shared understanding. To limit the effort involved, it is vital to concentrate on shared understanding about *relevant* things—that is, those aspects that lie within the context boundary of a system (cf. Principle 4). Even with a perfect shared understanding, important requirements may still be missed because nobody considered them. Figure 2.2 illustrates different situations of shared understanding with a simple example of a couple that wants to install a swing in their garden for their children [Glin2019]. The sticky note in the middle symbolizes a written specification.

Considering only end users and customers does not suffice

Stakeholders have different viewpoints

Managing inconsistencies and conflicts

The need for shared understanding

Explicit and implicit shared understanding

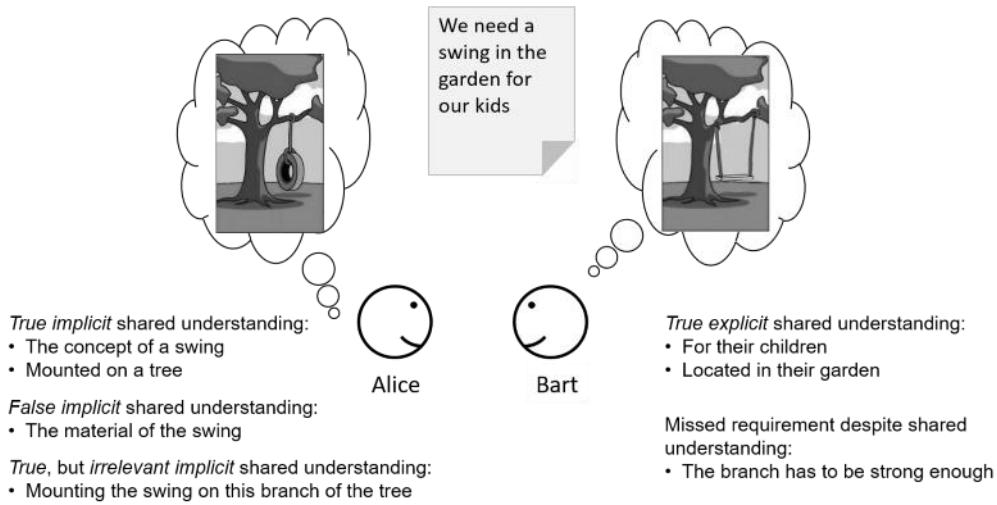


Figure 2.2 Different situations of shared understanding—illustrated with an example of a couple that wants to install a swing for their children

Proven practices for *achieving* shared understanding include working with glossaries (Section 3.5), creating prototypes (Section 3.7), or using an existing system as a reference point.

Achieving shared understanding in RE

The main means for *assessing* true explicit shared understanding in RE is thoroughly validating all specified requirements (cf. Principle 6 and Section 4.4). Practices for assessing implicit shared understanding include providing examples of expected outcomes, building prototypes, or estimating the cost of implementing a requirement. The most important practice for reducing the impact of false shared understanding is using a process with short feedback loops (Chapter 5).

Assessing shared understanding in RE

There are factors that constitute enablers or obstacles of shared understanding. For example, enablers are:

Enablers and obstacles

- ▶ Domain knowledge
- ▶ Domain-specific standards
- ▶ Previous successful collaboration
- ▶ Existence of reference systems known by all people involved
- ▶ Shared culture and values
- ▶ Informed (not blind!) mutual trust

Obstacles are:

- ▶ Geographic distance
- ▶ Supplier-customer relationship guided by mutual distrust
- ▶ Outsourcing
- ▶ Regulatory constraints
- ▶ Large and diverse teams
- ▶ High turnover among the people involved

The lower the probability and impact of false shared understanding and the better the ratio between enablers and obstacles, the more RE can rely on implicit shared understanding. Conversely, the fewer enablers and the more obstacles to shared understanding we have and the higher the risk and impact of false shared understanding for a requirement, the more such requirements have to be specified and validated explicitly.

Relying on shared understanding

Principle 4 – Context: Systems cannot be understood in isolation

Requirements never come in isolation. They refer to *systems* that are embedded in a *context*. While the term *context* in general denotes the network of thoughts and meanings needed for understanding phenomena or utterances, it has a special meaning in RE.

DEFINITION 2.1. CONTEXT (IN RE): The part of a system's environment being relevant for understanding the system and its requirements.

Context

The context of a system is delimited by the system boundary and the context boundary [Pohl2010] (see Figure 2.3).

DEFINITION 2.2. CONTEXT BOUNDARY: The boundary between the context of a system and those parts of the application domain that are irrelevant for the system and its requirements.

Context boundary

The *context boundary* separates the relevant part of the environment of a system to be developed from the irrelevant part—that is, the part that does not influence the system to be developed and, thus, does not have to be considered during Requirements Engineering.

DEFINITION 2.3. SYSTEM BOUNDARY: The boundary between a system and its surrounding context.

System boundary

The *system boundary* delimits the system as it shall be after its implementation and deployment. The system boundary is often not clear initially and it may change over time. Clarifying the system boundary and defining the external interfaces between a system and the elements in its context are genuine RE tasks.

The system boundary frequently coincides with the *scope* of a system development.

DEFINITION 2.4. SCOPE: The range of things that can be shaped and designed when developing a system.

Scope

Sometimes, however, the system boundary and its scope do not match (see Figure 2.3). There may be components within the system boundary that have to be reused as they are (i.e., they cannot be shaped or designed), which means that they are out of scope. On the other hand, there may be things in the system context that can be redesigned when the system is developed, which means that they are in scope.

It is not sufficient to consider just the requirements within the system boundary.

First, when the scope includes parts of the system context, as shown in Figure 2.3, context changes within the scope may impact the system's requirements. For example, when a business process shall be partially automated by a system, it may be useful to adapt the process in order to simplify its automation. Obviously, such adaptation impacts the requirements of the system.

Considering context changes in RE

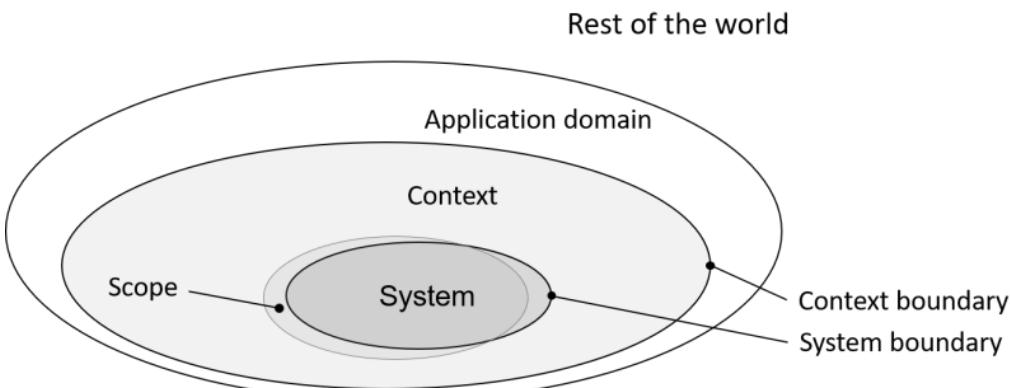


Figure 2.3 System, context, and scope

Second, there may be real-world phenomena in the system context that a system shall monitor or control. Requirements for such phenomena must be stated as domain requirements and must be adequately mapped to system requirements. For example, in a car equipped with an automatic gearbox, there is a requirement that the parking position can be engaged only when the car is not moving. In the context of a software system that controls the gearbox, this is a domain requirement. In order to satisfy this requirement, the controller needs to know whether or not the car is moving. However, the controller cannot sense this phenomenon directly. Hence, the real-world phenomenon “car is not moving” must be mapped to a phenomenon that the control system can sense—for example, input from a sensor that creates pulses when a wheel of the car is spinning. The domain requirement concerning engaging the parking position is then mapped to a system requirement such as “The gearbox control system shall enable the engagement of the parking position only if no pulses are received from the wheel spinning sensors.”

Mapping real-world phenomena

Third, there may be requirements that cannot be satisfied by any system implementation unless certain *domain requirements* and *domain assumptions* in the context of the system hold. Domain assumptions are assumptions about real-world phenomena in the context of a system. For example, consider an air traffic control system (ATS). The requirement “R1: The ATS shall maintain accurate positions for all aircraft controlled by the system” is an important system requirement. However, this requirement can be met only if the radar in the context of the ATS satisfies the requirements of correctly identifying all aircraft in the airspace controlled by the radar and correctly determining their position. In turn, these requirements can be satisfied only if all aircraft spotted by the radar respond properly to the interrogation signals sent by the radar.

Domain requirements and domain assumptions

Furthermore, requirement R1 can be met only if certain domain assumptions in the context of the ATS hold—for example, that the radar is not jammed by a malicious attacker and that no aircraft are flying at an altitude that is lower than the radar can detect.

RE goes beyond considering the requirements within the system boundary and defining the external interfaces at the system boundary. RE must also deal with phenomena in the system context.

Consequently, RE must also consider issues in the system context:

- If changes in the context may occur, how do they impact the requirements for the system?
- Which requirements in the real-world context are relevant for the system to be developed?

RE has to consider the context

- ▶ How can such real-world requirements be mapped adequately to requirements for the system?
- ▶ Which assumptions about the context must hold such that the system will work properly and the requirements in the real world will be met?

Principle 5 – Problem, requirement, solution: An inevitably intertwined triple

Problems, their solutions, and requirements are closely and inevitably intertwined [SwBa1982]. Every situation in which people are not satisfied with the way they are doing things can be considered as the occurrence of a *problem*. In order to eliminate that problem, a socio-technical system may be developed and deployed. *Requirements* for that system must be captured in order to make the system an effective *solution* to the problem. Specifying requirements does not make sense if there is no problem to solve or if no solution will be developed. Neither does it make sense to develop a solution that is searching for a problem to solve or for requirements to satisfy.

Why problems, requirements, and solutions are intertwined

It is important to note that problems, requirements, and solutions do not necessarily occur in this order. For example, when designing an innovative system, solution ideas create user needs that have to be worked out as requirements and implemented in an actual solution.

Problems, requirements, and solutions can be intertwined in many ways:

Forms of intertwinement

- ▶ Hierarchical intertwinement: when developing large systems with a multi-level hierarchy of subsystems and components, high-level requirements lead to high-level design decisions, which in turn inform lower-level requirements that lead to lower-level design decisions, etc.
- ▶ Technical feasibility: specifying non-feasible requirements is a waste of effort; however, it may only be possible to assess the feasibility of a requirement when exploring technical solutions.
- ▶ Validation: prototypes, which are a powerful means for validating requirements, constitute partial solutions of the problem.
- ▶ Solution bias: different stakeholders may envisage different solutions for a given problem, with the consequence that they specify different, conflicting requirements for that problem.

The intertwinement of problems, requirements, and solutions also has consequences for the development process for a system:

Consequences of intertwinement

- ▶ Strictly separating RE from system design and implementation activities is rarely possible. Therefore, strict waterfall development processes do not work well.
- ▶ Nevertheless, Requirements Engineers aim to separate problems, requirements, and solutions from each other as far as possible when thinking, communicating, and documenting. This *separation of concerns* makes RE tasks easier to handle.

Despite the inevitable intertwinement of problems, requirements, and solutions, Requirements Engineers strive to separate requirements concerns from solution concerns when thinking, communicating, and documenting.

Principle 6 – Validation: Non-validated requirements are useless

When a system is developed, the final system deployed shall satisfy the stakeholders' desires and needs. However, performing this check at the very end of development is very risky. In order to control the risk of unsatisfied stakeholders from the beginning, validation of requirements must start during RE (see Figure 2.4).

Why we need early validation

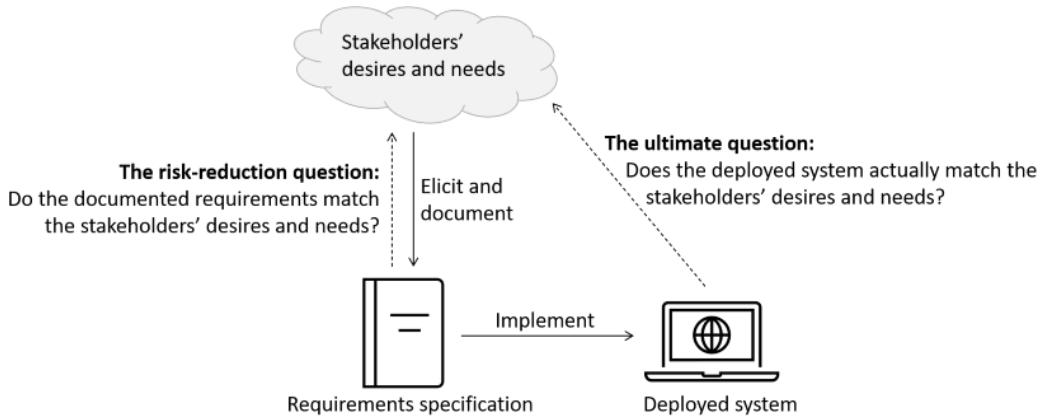


Figure 2.4 Validation [Glin2019]

DEFINITION 2.5. VALIDATION: The process of confirming that an item (a system, a work product, or a part thereof) matches its stakeholders' needs.

Validation

In RE, *validation* is the process of confirming that the documented requirements match the stakeholders' needs; in other words, confirming whether the right requirements have been specified.

Validation is a core activity in RE: there is no specification without validation.

When validating requirements, we have to check whether:

Things to validate

- Agreement about the requirements has been achieved among the stakeholders (conflicts resolved, priorities set)
- The stakeholders' desires and needs are adequately covered by the requirements
- The domain assumptions (see Principle 4 above) are reasonable—that is, we can expect that these assumptions can be met when the system is deployed and operated

Practices for validating requirements are discussed in Section 4.4.

Principle 7 – Evolution: Changing requirements are no accident, but the normal case

Evolution is inevitable

Every technical system is subject to evolution. Needs, businesses, and capabilities change continuously. As a natural consequence, the requirements for systems that are expected to satisfy needs, support businesses, and use technical capabilities will also change. Otherwise, such systems and their requirements progressively lose their value and eventually become useless.

A requirement may change while Requirements Engineers are still eliciting other requirements, when the system is under implementation, or when it is deployed and being used.

There are many reasons that lead to requests to change a requirement or a set of requirements for a system, for example:

Reasons for changing requirements

- Changed business processes
- Competitors launching new products or services
- Clients changing their priorities or opinions
- Changes in technology
- Feedback from system users asking for new or changed features
- Detection of errors in requirements or detection of faulty domain assumptions

Requirements may also change due to feedback from stakeholders when validating requirements, due to the detection of faults in previously elicited requirements, or due to changed needs.

As a consequence, Requirements Engineers must pursue two seemingly contradictory goals:

- Permit requirements to change, because trying to ignore the evolution of requirements would be futile.
- Keep requirements stable, because without some stability in the requirements, the cost for change can become prohibitively high. Also, development teams cannot develop systematically if requirements change on a daily basis.

Enabling change while preserving stability

Requirements Engineers need to manage the evolution of requirements. Otherwise, the evolution will manage them.

Change processes for requirements that address both goals are discussed in Section 6.7.

Principle 8 – Innovation: More of the same is not enough

While RE is concerned with satisfying the stakeholders' desires and needs, Requirements Engineers who just play the role of the stakeholders' voice recorder, specifying exactly what the stakeholders tell them, are doing the wrong job. Giving stakeholders exactly what they want means missing out on the opportunity of doing things better than before.

Requirements Engineers are not the stakeholders' voice recorders

For example, imagine the following scenario. An insurance company wants to renew the reporting system for its agents. The most frequently used report is a table with 18 columns, which is about twice as wide as the screen when displayed on the agents' laptop computers. Viewing this report thus requires a lot of scrolling. The stakeholders therefore want to be able to zoom in the report, using plus and minus buttons on the screen. In this situation, good Requirements Engineers will not just record this as a requirement. Instead, they will start to ask questions. It turns out that the company is going to replace the agents' laptops with tablets. Hence, implementing two-finger gestures instead of the required buttons will make zooming much easier. Furthermore, it turns out that three columns in the report can be eliminated with a slight change to the reporting rules, which the company agrees to make. Also, only six columns of the report are always needed; the remaining columns are used only in special cases.

Example of how it could work

Taking this into account, the Requirements Engineers would suggest that the stakeholders require that (1) the report shall show the same information as in the current system, minus the content of the three eliminated columns; (2) when the report is opened, only the six important columns are displayed in full width, while the other columns are collapsed to minimal width; and (3) that agents can expand a collapsed column by tapping its header (and collapse it again with another tap).

This way, the agents will get a system that does not simply add a workaround for viewing an oversized report. Instead, the system will solve the agents' problem with an innovative feature for filtering information and will also feature an intuitive means of zooming.

This is how innovation emerges. Good Requirements Engineers are innovation-aware: they strive not just to satisfy stakeholders but to make them happy, excited, or feel safe [KSTT1984]. At the same time, they avoid the trap of believing that they know everything better than the stakeholders do.

Good Requirements Engineers go beyond what their stakeholders tell them.

On a small scale, RE shapes innovative systems by striving for exciting new features and ease of use. Beyond that, Requirements Engineers also need to look for the big picture, exploring with the stakeholders whether there are any disruptive ways of doing things, leading to large-scale innovation [MaGR2004].

Section 4.2 discusses several techniques for fostering innovation in RE.

Innovation awareness

Shaping innovative systems in RE

Principle 9 – Systematic and disciplined work: We can't do without in RE

RE is not an art but a discipline, which calls for RE to be performed in a systematic and disciplined way. Regardless of the process(es) used to develop a system, we need to employ suitable RE processes and practices for systematically eliciting, documenting, validating, and managing requirements. Even when a system is developed in an ad hoc fashion, a systematic and disciplined approach to RE (for example, by systematically fostering shared understanding, see Principle 3) will improve the quality of the resulting system.

As a discipline, RE calls for systematic and disciplined work

Agility and flexibility are not valid excuses for an unsystematic, ad hoc style of work in RE.

However, there is neither a universal RE process nor a universal set of RE practices that work well in every given situation or at least in most situations: there is no "one size fits all" in RE.

No "one size fits all"

Systematic and disciplined work means that Requirements Engineers:

- Configure an RE process that is well suited for the problem at hand and fits well with the process used for developing the system (see Chapter 5).
- From the set of RE practices and work products available, select those that are best suited for the given problem, context, and working environment (see Chapters 3, 4 and 6).
- Do not always use the same process, practices, and work products.
- Do not reuse processes and practices from past successful RE work without reflection.

What Requirements Engineers need to do

2.3 Further Reading

Glinz [Glin2008] discusses the value of quality requirements and of requirements in general [Glin2016].

Glinz and Wieringa [GlWi2007] explain the notion and importance of stakeholders.

Glinz and Fricker [GlFr2015] discuss the role and importance of shared understanding.

The papers by Jackson [Jack1995b] and Gunter et al. [GGJZ2000] are fundamental for the problem of requirements in context. The role of context in RE is also discussed by Pohl [Pohl2010].

Gause and Weinberg [GaWe1989] discuss the interdependence of problems and solutions. Swartout and Balzer [SwBa1982] were the first to point out that creating a complete specification before starting implementation is rarely possible.

Validation is covered in any RE textbook. Grünbacher and Seyff [GrSe2005] discuss how to achieve agreement by negotiating requirements.

Kano et al. [KSTT1984] were among the first to stress the role of innovation. Maalej, Nayebi, Johann, and Ruhe [MNJR2016] discuss the use of explicit and implicit user feedback for RE. Maiden, Gitzikis, and Robertson [MaGR2004] discuss how creativity can foster innovation in RE. Gorschek et al. [GFPK2010] outline a systematic innovation process.

3. Work Products and Documentation Practices

Traditional Requirements Engineering (RE) calls for the writing of a comprehensive, complete, and unambiguous requirements specification [IEEE830], [Glin2016]. While it is still appropriate to create fully-fledged requirements specifications in many cases, there are also many other cases where the cost of writing such specifications exceeds their benefit. For example, fully-fledged requirements specifications are useful or even necessary when tendering or outsourcing the design and implementation of a system or when a system is safety-critical and regulatory compliance is required. On the other hand, where stakeholders and developers join forces to define and develop a system iteratively, writing a comprehensive requirements specification does not make sense. It is therefore vital in RE to adapt the documentation to the project context and to select work products for documenting requirements and requirements-related information that yield optimal value for the project.

In this chapter, you will learn about the typical RE work products and how to create them.

3.1 Work Products in Requirements Engineering

There are a variety of work products that are used in RE.

DEFINITION 3.1. WORK PRODUCT: A recorded intermediate or final result generated in a work process.

Work product

We consider the term *artifact* as a synonym for work product. We prefer the term work product over artifact to express the connotation that a work product is the result of work performed in a work process.

Artifact

According to this definition, an RE work product can be anything that expresses requirements, from a single sentence or diagram to a system requirements specification that covers hundreds of pages. It is also important to note that a work product may contain other work products.

3.1.1 Characteristics of Work Products

Work products can be characterized by the following facets: *purpose*, *size*, *representation*, *lifespan*, and *storage*.

Characterization of work products

Table 3.1 gives an overview of typical work products used in RE along with their respective purpose (that is, what the work product specifies or provides) and typical size. The table is structured into four groups: work products for single requirements, sets of requirements, documentation structures, and other work products.

Representation

There are many different ways to represent a work product. In RE, representations based on *natural language*, *templates*, and *models* are of particular importance. These are discussed in Sections 3.2, 3.3, and 3.4, respectively. There are further representations, such as drawings or prototypes, which are covered in Section 3.7.

Every work product has a *lifespan*. This is the period of time from the creation of the work product until the point where the work product is discarded or becomes irrelevant. We distinguish between three categories of work products with respect to lifespan: *temporary*, *evolving*, and *durable* work products.

Lifespan

Temporary work products are created to support communication and create shared understanding (for example, a sketch of a user-system interaction created in a workshop). Temporary work products are discarded after use; no metadata is kept about these work products.

Temporary work products

Evolving work products grow in several iterations over time (for example, a collection of user stories that grows in both the number of stories and the story content). Some metadata (at least the owner, status, and revision history) should be kept for every evolving work product. Depending on the importance and status of a work product, change control procedures need to be applied when modifying an evolving work product.

Evolving work products

Durable work products have been baselined or released (for example, a requirements specification that is part of a contract or a sprint backlog that is implemented in a given iteration). A full set of metadata must be kept to manage the work product properly and an elaborate change process must be followed to change a durable work product (Chapter 6).

Durable work products

A temporary work product may become an evolving one when Requirements Engineers decide to keep a work product and develop it further. In this case, some metadata should be added in order to keep the evolution of the work product under control. When an evolving work product is baselined or released, it changes its lifespan status from evolving to durable.

*Temporary → Evolving
→ Durable*

Table 3.1 Overview of RE work products

Work product	Purpose: The work product specifies /provides	Size	<i>Typical RE work products</i>
Single requirement			
Individual requirement	A single requirement, typically in textual form	S	
User story	A function or behavior from a stakeholder's perspective	S	
Set of requirements			
Use case	A system function from an actor's or user's perspective	S-M	
Graphic model	Various aspects, for example, context, function, behavior (see Section 3.4)	M	
Task description	A task that a system shall perform	S-M	
External interface description	The information exchanged between a system and an actor in the system context	M	
Epic	A high-level view of a stakeholder need	M	
Feature	A distinguishing characteristic of a system	S-M	
Documents or documentation structures			
System, business, stakeholder, or user requirements specification	A comprehensive requirements document	L-XL	
Product and sprint backlog	A list of work items, including requirements	M-L	
Story map	A visual arrangement of user stories	M	
Vision	A conceptual imagination of a future system	M	

Work product	Purpose: The work product specifies /provides	Size
Other work products		
Glossary	Unambiguous and agreed common terminology	M
Textual note or graphic sketch	A memo for communication and understanding	S
Prototype	A specification by example, particularly for understanding, negotiating, and validating requirements	S-L

S: Small, M: Medium, L: Large, XL: Very large

Nowadays, most work products are stored electronically as files, in databases, or in RE tools. Informal, temporary work products may also be stored on other media—for example, paper or sticky notes on a Kanban board.

Storing work products

3.1.2 Abstraction Levels

Requirements and their corresponding work products occur at various abstraction levels—from, for example, high-level requirements for a new business process, down to requirements at a very detailed level, such as the reaction of a specific software component to an exceptional event.

Requirements occur at various abstraction levels

Business requirements, domain requirements, and stakeholder/user requirements typically occur at a higher level of abstraction than system requirements. When a system consists of a hierarchy of subsystems and components, we have system requirements at the corresponding abstraction levels for subsystems and components. As a consequence, requirements are frequently organized in three layers of abstraction: the business, system, and component levels.

Typical layers: business, system, components

When business requirements and stakeholder requirements are expressed in durable work products—such as business requirements specifications, stakeholder requirements specifications, or vision documents—they precede the specification of system requirements. For example, in contractual situations, where a customer orders the development of a system from a supplier, the customer frequently creates and releases a stakeholder requirements specification. The supplier then uses this as the basis for producing a system requirements specification. In other projects, business requirements, stakeholder requirements, and system requirements may co-evolve.

Dependencies

Some work products, such as individual requirements, sketches, or process models, occur at all levels. Other work products are specifically associated with certain levels. For example, a system requirements specification is associated with the system level. Note that an individual requirement at a high abstraction level may be refined into several detailed requirements at more concrete levels.

Choosing a proper abstraction level

The choice of the proper abstraction level depends on what is to be specified. It is important, however, not to mix requirements that are at different abstraction levels. For example, in the specification of a healthcare information system, when writing a detailed requirement about photos on client ID cards, the subsequent paragraph should not state a general system goal such as reducing healthcare cost while maintaining the current service level for clients. In small and medium-sized work products, requirements should be at more or less the same abstraction level. In large work products such as a system requirements specification, requirements at different levels of abstraction should be kept separate by structuring the specification accordingly (Section 0).

Requirements naturally occur at different levels of abstraction. Selecting work products that are adequate for a given level of abstraction and properly structuring work products that contain requirements at multiple abstraction levels is helpful.

3.1.3 Level of Detail

When specifying requirements, Requirements Engineers have to decide on the level of detail in which the requirements shall be specified. However, deciding which level of detail is appropriate or even optimal for a given requirement is a challenging task.

For example, in a situation where the customer and the supplier of a system collaborate closely, it might be sufficient to state a requirement about a data entry form as follows: “The system shall provide a form for entering the personal data of the customer.” In contrast, in a situation where the design and implementation of the system are outsourced to a supplier with little or no domain knowledge, a detailed specification of the customer entry form will be necessary.

The level of detail to which requirements should be specified depends on several factors, in particular:

- ▶ The problem and project context: the harder the problem and the less familiar the Requirements Engineers and developers are with the project context, the more detail is necessary.
- ▶ The degree of shared understanding of the problem: when there is low implicit shared understanding (see Principle 3 in Chapter 2), explicit, detailed specifications are required to create the necessary degree of shared understanding.
- ▶ The degree of freedom left to designers and programmers: less detailed requirements give the developers more freedom.
- ▶ Availability of rapid stakeholder feedback during design and implementation: when rapid feedback is available, less detailed specifications suffice to control the risk of developing the wrong system.
- ▶ Cost vs. value of a detailed specification: the higher the benefit of a requirement, the more we can afford to specify it in detail.
- ▶ Standards and regulations: Standards imposed and regulatory constraints may mean that requirements have to be specified in more detail than would otherwise be necessary.

How much detail?

Factors affecting the level of detail needed

There is no universally “right” level of detail for requirements. For every requirement, the adequate level of detail depends on many factors. The greater the level of detail in the requirements specified, the lower the risk of eventually getting something that has unexpected or missing features or properties. However, the cost for the specification increases as the level of detail increases.

3.1.4 Aspects to be Considered

Regardless of the RE work products being used, several aspects need to be considered when specifying requirements [Glin2019].

First, as there are functional requirements, quality requirements, and constraints (see Section 1.1), Requirements Engineers have to make sure that they cover all three kinds of requirements when documenting requirements. In practice, stakeholders tend to omit quality requirements because they take them for granted.

Considering multiple aspects

They also tend to specify constraints as functional requirements. It is therefore important that the Requirements Engineers get this right.

When looking at functional requirements, we observe that they pertain to different aspects, as, for example, a required data structure, a required order of actions, or the required reaction to some external event. We distinguish between three major aspects: *structure and data*, *function and flow*, and *state and behavior*.

Aspects within the functional requirements

The *structure and data* aspect focuses on requirements concerning the static structure of a system and the (persistent) data that a system must know in order to perform the required functions and deliver the required results.

Structure and data

The *function and flow* aspect deals with the functions that a system shall provide and the flow of control and data within and between functions for creating the required results from given inputs.

Function and flow

The *state and behavior* aspect concentrates on specifying the state-dependent behavior of a system—in particular, how a system shall react to which external event depending on the system's current state.

State and behavior

When dealing with *quality requirements*, such as usability, reliability, or availability, a quality model—for example, the model provided by ISO/IEC 25010 [ISO25010]—can be used as a checklist.

Quality requirements

Within the quality requirements, *performance requirements* are of particular importance. Performance requirements deal with:

- ▶ Time (e.g., for performing a task or reacting to external events)
- ▶ Volume (e.g., required database size)
- ▶ Frequency (e.g., of computing a function or receiving stimuli from sensors)
- ▶ Throughput (e.g., data transmission or transaction rates)
- ▶ Resource consumption (e.g., CPU, storage, bandwidth, battery)

Some people also consider the required accuracy of a computation as a performance requirement.

Whenever possible, measurable values should be specified. When values follow a probability distribution, specifying just the average does not suffice. If the distribution function and its parameters cannot be specified, Requirements Engineers should strive to specify minimum and maximum values or 95 percent values in addition to the averages.

Difficulty of documenting quality requirements

Documenting quality requirements beyond performance requirements is notoriously difficult.

Qualitative representations, such as “The system shall be secure and easy to use,” are ambiguous and thus difficult to achieve and validate.

Quantitative representations are measurable, which is a big asset in terms of systematically achieving and validating a quality requirement. However, they raise principal difficulties (for example, how can we state security in quantitative terms?) and can be quite expensive to specify.

Operationalized representations state a quality requirement in terms of functional requirements for achieving the desired quality. For example, a data security requirement may be expressed in terms of a login function that restricts the access to the data and a function that encrypts the stored data. Operationalized representations make quality requirements testable but may also imply premature design decisions.

The often-heard rule “Only a quantified quality requirement is a good quality requirement” is outdated and may lead to quality requirements having low or even negative value due to the high effort involved in the quantification. Instead, a risk-based approach should be used [Glin2008].

Qualitative representations of quality requirements suffice in the following situations:

- ▶ There is sufficient implicit shared understanding between stakeholders, Requirements Engineers, and developers.
- ▶ Stakeholders, Requirements Engineers, and developers agree on a known solution that satisfies the requirements.
- ▶ Stakeholders only want to give general quality directions and trust the developers to get the details right.
- ▶ Short feedback loops are in place such that problems can be detected early.

When developers are *able to generalize from examples*, specifying quality requirements in terms of quantified examples or comparisons to an existing system is a cheap and effective way of documenting quality requirements.

Only in cases where there is a *high risk* of not meeting the stakeholders’ needs, particularly when quality requirements are *safety-critical*, should a fully quantified representation or an operationalization in terms of functional requirements be considered.

When specifying *constraints*, the following categories of constraints should be considered:

- ▶ *Technical*: given interfaces or protocols, components, or frameworks that have to be used, etc.
- ▶ *Legal*: restrictions imposed by laws, contracts, standards, or regulations
- ▶ *Organizational*: there may be constraints in terms of organizational structures, processes, or policies that must not be changed by the system.
- ▶ *Cultural*: user habits and expectations are to some extent shaped by the culture the users live in. This is a particularly important aspect to consider when the users of a system come from different cultures or when Requirements Engineers and developers are rooted in a different culture to the system’s users.
- ▶ *Environmental*: when specifying cyber-physical systems, environmental conditions such as temperature, humidity, radiation, or vibration may have to be considered as constraints; energy consumption and heat dissipation may constitute further constraints.
- ▶ *Physical*: when a system comprises physical components or interacts with them, the system becomes constrained by the laws of physics and the properties of materials used for the physical components.
- ▶ Furthermore, *particular solutions or restrictions demanded by important stakeholders* also constitute constraints.

Constraints

Constraint categories to be considered

Finally, requirements can only be understood in *context* (see Principle 4 in Chapter 2). Consequently, a further aspect has to be considered, which we call *context and boundary*.

Context and boundary

The *context and boundary* aspect covers domain requirements and domain assumptions in the context of the system, as well as the external interfaces between the system and its environment at the system boundary.

There are many interrelationships and dependencies between the aspects mentioned above. For example, a request issued by a user (context) may be received by the system via an external interface (boundary), trigger a state transition of the system (state and behavior), which initiates an action (function) followed by another action (flow) that requires data with some given structure (structure and data) to provide a result to the user (context) within a given time interval (quality).

Interrelationships and dependencies between aspects

Some work products focus on a specific aspect and abstract from the other aspects. This is particularly the case for requirements models (Section 3.4). Other work products, such as a system requirements specification, cover all these aspects. When different aspects are documented in separate work products or in separate chapters of the same work product, these work products or chapters must be kept consistent with each other.

Many different aspects need to be considered when documenting requirements, in particular, functionality (structure and data, function and flow, state and behavior), quality, constraints, and surrounding context (context and boundary).

3.1.5 General Documentation Guidelines

Independently of the techniques used, there are some general guidelines that should be followed when creating RE work products:

- ▶ Select a work product type that fits the intended purpose.
- ▶ Avoid redundancy by referencing content instead of repeating the same content again.
- ▶ Avoid inconsistencies between work products, particularly when they cover different aspects.
- ▶ Use terms consistently, as defined in the glossary.
- ▶ Structure work products appropriately—for example, by using standard structures.

General guidelines

3.1.6 Work Product Planning

Each project setting and each domain is different, so the set of resulting work products must be defined for each endeavor. The parties involved, particularly the Requirements Engineers, stakeholders, and project/product owners or managers need to agree upon the following issues:

- ▶ In which work products shall the requirements be recorded and for what purpose (see Table 3.1)?
- ▶ Which abstraction levels need to be considered (Section 3.1.2)?
- ▶ Up to which level of detail must requirements be documented at each abstraction level (Section 3.1.3)?
- ▶ How shall the requirements be represented in these work products (for example, natural-language-based or model-based, see below) and which notation(s) shall be used?

What to consider

Requirements Engineers should define the RE work products to be used at an early stage in a project. Such early definition:

Define RE work products early

- ▶ Helps in the planning of efforts and resources
- ▶ Ensures that appropriate notations are used
- ▶ Ensures that all results are recorded in the right work products
- ▶ Ensures that no major reshuffling of information and “final editing” is needed
- ▶ Helps to avoid redundancy, resulting in less work and better maintainability

3.2 Natural-Language-Based Work Products

Natural language, in both spoken and written form, has always been a core means for communicating requirements for systems. Using natural language to write RE work products has many advantages. In particular, natural language is extremely expressive and flexible, which means that almost any conceivable requirement in any aspect can be expressed in natural language. Furthermore, natural language is used in everyday life and is taught at school, so no specific training is required to read and understand requirements written in natural language.

Advantages

Human evolution has shaped natural language as a means for *spoken communication between directly interacting people*, where misunderstandings and missing information can be detected and corrected rapidly. Hence, natural language is *not* optimized for precise, unambiguous, and comprehensive communication by means of written documents. This constitutes a major problem when writing technical documentation (such as requirements) in natural language. In contrast to communication in *spoken* natural language, where the communication is contextualized and interactive with immediate feedback, there is no natural means for rapidly detecting and correcting ambiguities, omissions, and inconsistencies in texts *written* in natural language. On the contrary, finding such ambiguities, omissions, and inconsistencies in written texts is difficult and expensive, particularly for work products that contain a large amount of natural language text.

Problems

The problem can be mitigated to some extent by writing technical documentation consciously, following proven rules and avoiding known pitfalls.

Writing rules

When writing requirements in natural language, Requirements Engineers can avoid many potential misunderstandings by applying some simple rules:

- ▶ Write short and well-structured sentences. The rule of thumb is to express a single requirement in one sentence in natural language. To achieve a good structure, Requirements Engineers should use phrase templates (Section 3.3).
- ▶ Create well-structured work products. Besides writing well-structured sentences (see above), work products written in natural language should also be well-structured as a whole. A proven way to do this is by using a hierarchical structure of parts, chapters, sections, and subsections, as is usually done in technical books. Document templates (Section 3.3) help you to achieve a good structure.
- ▶ Define and consistently use a uniform terminology. Creating and using a glossary (Section 3.5) is the core means for avoiding misunderstandings and inconsistencies about terminology.

Short sentences

Structured work products

Uniform terminology

- Avoid using vague or ambiguous terms and phrases.
- Know and avoid the pitfalls of technical writing (see below).

Vagueness, ambiguity
Knowing the pitfalls

When writing technical documents in natural language, there are some well-known pitfalls that should be avoided or things that need to be used with care (see, for example, [GoRu2003]).

Requirements Engineers should *avoid* writing requirements that contain the following:

- *Incomplete descriptions.* Verbs in natural language typically come with a set of placeholders for nouns or pronouns. For example, the verb “give” has three placeholders for *who* gives *what* to *whom*. When writing a requirement in natural language, all placeholders of the verb used should be filled.
- *Unspecific nouns.* Using nouns such as “the data” or “the user” leaves too much room for different interpretations by different stakeholders or developers. They should be replaced by more specific nouns or be made more specific by adding adjectives or assigning them a well-defined type.
- *Incomplete conditions.* When describing what shall be done, many people focus on the normal case, omitting exceptional cases. In technical writing, this is a trap to avoid: when something happens only if certain conditions are true, such conditions shall be stated, providing both *then* and *else* clauses.
- *Incomplete comparisons.* In spoken communication, people tend to use comparatives (for example, “the new video app is much better”) without saying what they are comparing to, typically assuming that this is clear from the context. In technical writing, comparisons should include a reference object, for example, “faster than 0.1 ms”.

Things to avoid

Incomplete descriptions

Unspecific nouns

Incomplete conditions

Incomplete comparisons

There are some further things that Requirements Engineers need to use with care, as they constitute potential pitfalls:

- *Passive voice.* Sentences in passive voice have no subject. If a requirement is stated in the passive voice, this may hide who is responsible for the action described in the requirement, leading to an incomplete description.
- *Universal quantifiers.* Universal quantifiers are words such as *all*, *always*, or *never*, which are used to make statements that are universally true. In technical systems, however, such universal properties are rare. Whenever Requirements Engineers use a universal quantifier, they need to reflect on whether they are stating a truly universal property or whether they are instead specifying a general rule that has exceptions (which they also need to specify). They should apply the same caution when using “either-or” clauses, which, by their semantics, exclude any further exceptional cases.
- *Nominalizations.* When a noun is derived from a verb (for example, “authentication” from “to authenticate”), linguists call this a nominalization. When specifying requirements, Requirements Engineers need to handle nominalizations with care because a nominalization may hide unspecified requirements. For example, the requirement “Only after successful authentication, the system shall provide a user access to (...)" implies that a procedure for authenticating users exists.

Things to handle with care

Passive voice

Universal quantifiers

Nominalizations

When writing such a requirement, therefore, the Requirements Engineer must check whether there are also requirements about the procedure for authenticating legitimate users.

Natural language is a very powerful means for writing requirements. To mitigate the inherent disadvantages of using natural language for technical documentation, Requirements Engineers should follow proven writing rules and avoid well-known pitfalls.

3.3 Template-Based Work Products

As mentioned in Section 3.2 above, using templates is a proven means for writing good, well-structured work products in natural language and thus mitigating some of the weaknesses of natural language for technical writing. A template is a kind of ready-made blueprint for the syntactic structure of a work product. When using natural language in RE, we distinguish between three classes of templates: phrase templates, form templates, and document templates.

3.3.1 Phrase Templates

DEFINITION 3.2. PHRASE TEMPLATE: A template for the syntactic structure of a phrase that expresses an individual requirement or a user story in natural language.

Phrase template

A phrase template provides a skeleton structure with placeholders, in which Requirements Engineers fill in the placeholders in order to get well-structured, uniform sentences that express the requirements.

Using phrase templates is a best practice when writing individual requirements in natural language and when writing user stories.

3.3.1.1 Phrase Templates for Individual Requirements

Various phrase templates for writing individual requirements have been defined, for example, in [ISO29148], [MWHN2009], and [Rupp2014]. The standard ISO/IEC/IEEE 29148 [ISO29148] provides a single, uniform template for individual requirements as follows:

[<Condition>] <Subject> <Action> <Objects> [<Restriction>].

*ISO/IEC/IEEE 29148
phrase template*

Example: When a valid card is sensed, the system shall display the “Enter your PIN” message on the dialog screen within 200 ms.

When formulating an action with this template, the following conventions about the use of auxiliary verbs are frequently used in practice:

Using auxiliary verbs

- *Shall* denotes a mandatory requirement.
- *Should* denotes a requirement that is not mandatory but strongly desired.
- *May* denotes a suggestion.
- *Will* (or using a *verb in the present tense* without one of the auxiliary verbs mentioned above) denotes a factual statement that is not considered as a requirement.

When there are no agreed meanings for auxiliary verbs in a project, or when in doubt, definitions such as the ones given above should be made part of a requirements specification.

EARS (Easy Approach to Requirements Syntax) [MWHN2009] provides a set of phrase templates that are adapted to different situations as described below.

EARS templates

Ubiquitous requirements (must always hold): The <system name> shall <system response>.

Event-driven requirements (triggered by an external event):

WHEN <optional preconditions> <trigger> the <system name> shall <system response>.

Unwanted behavior (describing situations to be avoided):

IF <optional preconditions> <trigger>, THEN the <system name> shall <system response>.

Note: Although the unwanted behavior template is similar to the event-driven one, Mavin et al. provide a separate template for the latter, arguing that unwanted behavior (primarily due to unexpected events in the context, such as failures, attacks, or things that nobody has thought of), is a major source of omissions in RE.

State-driven requirements (apply only in certain states):

WHILE <in a specific state> the <system name> shall <system response>.

Optional features (applicable only if some feature is included in the system):

WHERE <feature is included> the <system name> shall <system response>.

In practice, sentences that combine the keywords WHEN, WHILE, and WHERE may be needed to express complex requirements.

EARS has been designed primarily for the specification of cyber-physical systems. However, it can also be adapted for other types of systems.

3.3.1.2 Phrase Templates for User Stories

The classic phrase template for writing user stories was introduced by Cohn [Cohn2004]:

Cohn's user story template

As a <role> I want <requirement> so that <benefit>.

Example: “As a line manager, I want to make ad hoc inquiries to the accounting system so that I can do financial planning for my department.”

While Cohn has designated the <benefit> part of the template as optional, it is standard practice nowadays to specify a benefit for every user story.

Every user story should be accompanied by a set of *acceptance criteria*—that is, criteria that the implementation of the user story must satisfy in order to be accepted by the stakeholders. Acceptance criteria make a user story more concrete and less ambiguous. This helps to avoid implementation errors due to misunderstandings.

Acceptance criteria

3.3.2 Form Templates

DEFINITION 3.3. FORM TEMPLATE: A template providing a form with predefined fields to be filled in.

Form template

Form templates are used to structure work products of medium size such as use cases. Cockburn [Cock2001] introduced a popular form template for use cases. [Laue2002] proposed a template for task descriptions. Table 3.2 shows a simple form template for use cases. Each flow step may be subdivided into an action by an actor and the response by the system.

Table 3.2 A simple form template for writing use cases

Name	< A short active verb phrase>	<i>Use case template</i>
Precondition	<Condition(s) that must hold when the execution of the use case is triggered>	
Success end condition	<State upon successful completion of use case>	
Failed end condition	<State upon failed execution of use case>	
Primary actor	<Actor name>	
Other actors	<List of other actors involved, if any>	
Trigger	<Event that initiates the execution of the use case>	
Normal flow	<p><Description of the main success scenario in a sequence of steps:</p> <p style="padding-left: 20px;"><step 1> <action 1> <step 2> <action 2> ... <step n> <action n> ... ></p>	
Alternate flows	<Description of alternative or exceptional steps, with references to the corresponding steps in the normal flow>	
Extensions	<Extensions to the normal flow (if there are any), with references to the extended steps in the normal flow>	
Related information	<Optional field for further information, such as performance, frequency, relationship to other use cases, etc.>	

Form templates are also useful for writing quality requirements in a measurable form [Gilb1988]. Table 3.2 provides a simple form template for measurable quality requirements, along with an example.

Table 3.3 A form template for specifying measurable quality requirements

Template	Example	Measurable quality requirement template
ID	<Number of requirement>	R137.2
Goal	<Qualitatively stated goal>	Confirm room reservations immediately
Scale	<Scale for measuring the requirement>	Elapsed time in seconds (ratio scale)
Meter	<Procedure for measuring the requirement>	Timestamping the moments when the user hits the “Reserve” button and when the app has displayed the confirmation. Measuring the time difference.
Minimum	<Minimum acceptable quality to be achieved>	Less than 5 s in at least 95% of all cases
OK range	<Value range that is OK and is aimed at>	Between 0.5 and 3 s in more than 98% of all cases
Desired	<Quality achieved in the best possible case>	Less than 0.5 s in 100% of all cases

3.3.3 Document Templates

DEFINITION 3.4. DOCUMENT TEMPLATE: A template providing a predefined skeleton structure for a document.

Document template

Document templates help to systematically structure requirements documents—for example, a system requirements specification. RE document templates may be found in standards, for example in [ISO29148]. The Volere template by Robertson and Robertson [RoRo2012], [Vole2020] is also popular in practice. When a requirements specification is included in the set of work products that a customer has ordered and will pay for, that customer may prescribe the use of document templates supplied by the customer. In

Figure 3.1, we show an example of a simple document template for a system requirements specification.

3.3.4 Advantages and Disadvantages

Using templates when writing RE work products in natural language has major advantages. Templates provide a clear, re-usable structure for work products, make them look uniform, and thus improve the readability of the work products. Templates also help you to capture the most relevant information and make fewer errors of omission. On the other hand, there is a potential pitfall when Requirements Engineers use templates mechanically, focusing on the syntactic structure rather than on content, neglecting everything that does not fit the template.

<p>Part I: Introduction</p> <ol style="list-style-type: none"> 1. System purpose 2. Scope of system development 3. Stakeholders <p>Part II: System overview</p> <ol style="list-style-type: none"> 4. System vision and goals 5. System context and boundary 6. Overall system structure 7. User characteristics <p>Part III: System requirements</p> <p>Organized hierarchically according to system structure, using a hierarchical numbering scheme for requirements</p> <p>Per subsystem/component:</p> <ul style="list-style-type: none"> • Functional requirements (structure and data, function and flow, state and behavior) • Quality requirements • Constraints • Interfaces <p>References</p> <p>Appendices</p> <p>Glossary (if not managed as a work product of its own)</p> <p>Assumptions and dependencies</p>	<i>Sample document template</i>
--	---------------------------------

Figure 3.1 A simple system requirements specification template

Using templates when writing RE work products in natural language improves the quality of the work products provided that the templates are not misused as just a syntactic exercise.

3.4 Model-Based Work Products

Requirements formulated in natural language can easily be read by people provided they can speak the language. Natural language suffers from ambiguity due to the imprecision of semantics of words, phrases, and sentences [Davi1993]. This imprecision may lead to confusion and omissions in requirements. When you read textual requirements, you will try to interpret them in your own way. We often try to imagine these requirements in our mind. When the number of requirements is manageable, it is possible to maintain insight and an overview of the textual requirements. When the number of textual requirements becomes “too big,” we lose the overview. That limit is different for each person. The number of textual requirements is not the only reason for losing insight and overview. The complexity of the requirements, the relationship between the requirements, and abstraction of the requirements also contribute to this. You may have to read the requirements formulated in natural language several times before you get a correct and complete picture that the system must comply with. We have a limited ability to process requirements in natural language.

Requirements in natural language have their limitations

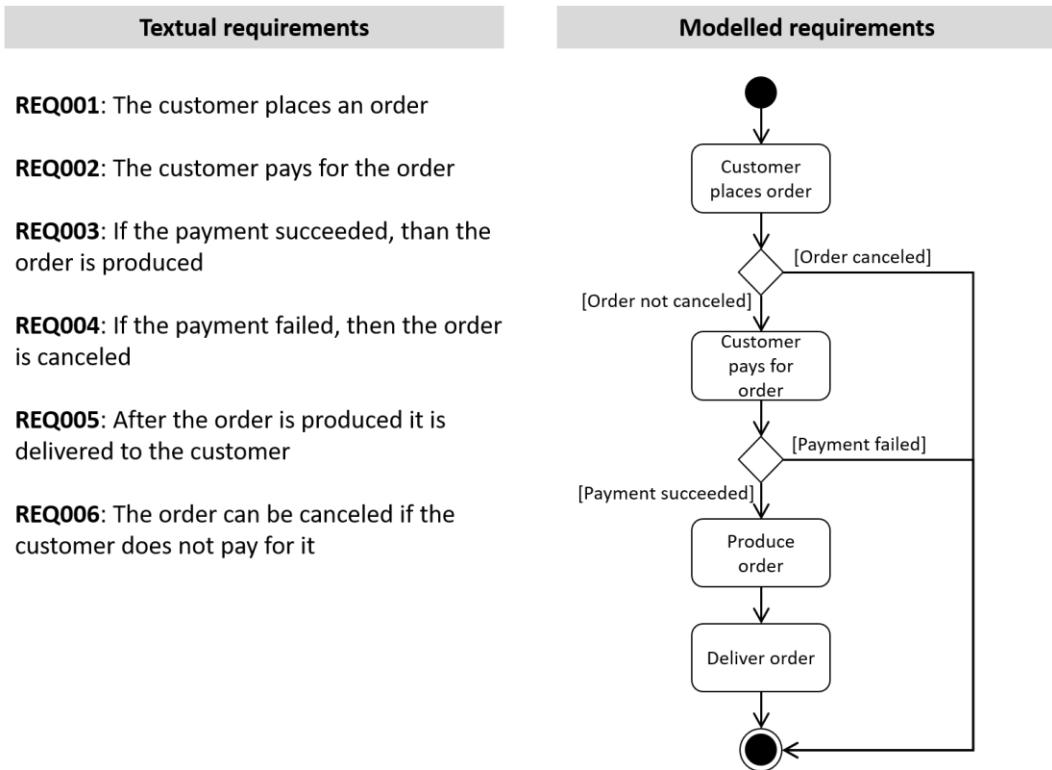


Figure 3.2 Textual requirements versus modeled requirements

A model is an abstract representation of an existing part of reality or a part of reality to be created. Displaying the requirements (also) with a model (or picture) will contribute to readers grasping the requirements. Such diagrammatic representation of a model is called a diagram.

A model is an abstract representation of reality

The diagram in Figure 3.2 shows at a glance what the system must provide, but only if you have mastered the modeling language. It is evident that if you do not understand the diagram, in this case a UML activity diagram, the picture will not contribute to a better understanding of the requirements.

Modeling requirements contributes to maintaining an overview of and insight into the requirements

In the next section (3.4.1), the concept of a requirements model is explained. Modeling of business requirements and goals is explained in Section 3.4.5.1. An important method for describing the demarcation of a system is the context model. Examples of the context are depicted in Section 3.4.2. Sections 3.4.3 to 3.4.5 give a number of examples of modeling languages that are often used in systems engineering practice.

3.4.1 The Role of Models in Requirements Engineering

Like any language, a modeling language consists of grammatical rules and a description of the meaning of the language constructs, see Section 3.4.1.1. Although a model is a visual representation of reality, the language rules are important in order to understand the model and the nuances in the model.

It is not always efficient or effective to summarize the requirements in a model. By understanding the properties of a model, we can better determine when we can apply which model, see Section 3.4.1.2.

Just as natural language has advantages and disadvantages for expressing the requirements, so do models. If we observe these facts in applying a model, we can better determine the added value of applying the "correct" model. This is discussed in Section 3.4.1.3.

Many models have already been standardized and are used in various fields of application, see Section 3.4.1.4. Consider, for example, the construction of a house, where an architect uses a standardized model to describe the house. Another example is electronics, where the drawing of electronic diagrams is standardized so that professionals can understand, calculate, and realize the electronics.

To determine whether a diagram is applied correctly, we can validate the quality criteria of a diagram. These criteria are described in Section 3.4.1.5.

3.4.1.1 Syntax and Semantics

If you think about a natural language, for example your native language, it is defined by its grammar and semantics.

The grammar describes the elements (words and sentences) and the rules that the language must obey. In a modeling language, this is called the syntax, see Figure 3.3. The syntax describes which notation elements (symbols) are used in the language. It also describes how these notation elements can be used in combination.

A modeling language consists of syntax and semantics

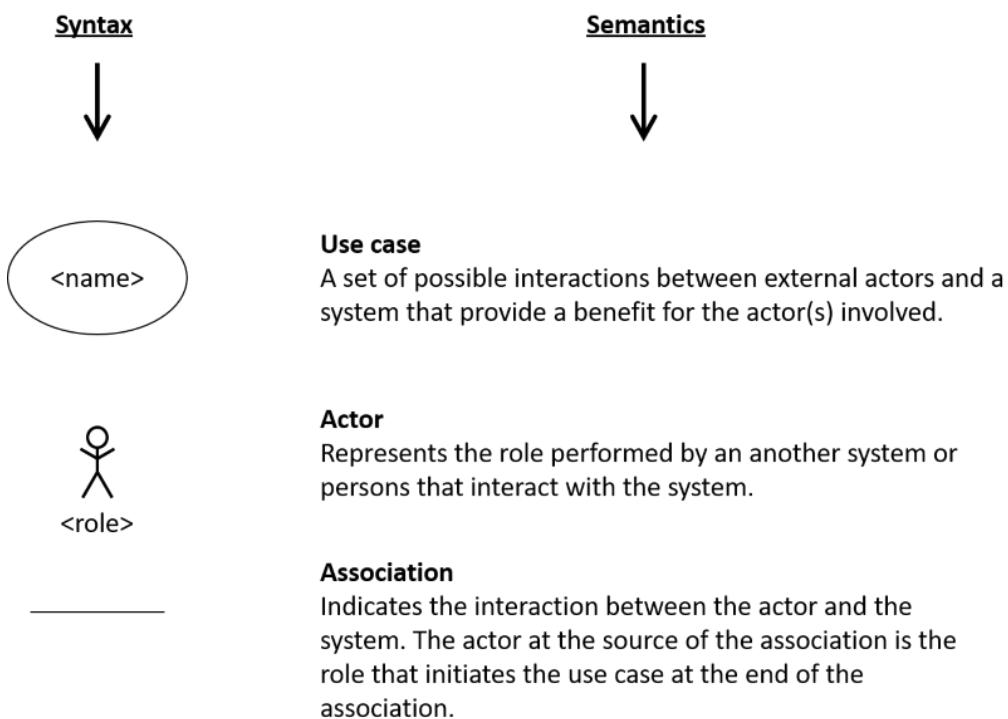


Figure 3.3 Modeling language syntax and semantics

The semantics defines the meaning of the notation elements and defines the meaning of the combination of elements. Understanding the meaning of the notation elements is fundamental for preventing the risk of the model being misinterpreted.

3.4.1.2 Properties of a Model

A requirements model is a conceptual model that depicts the requirements for the system to be developed. A model is also used to represent the current situation to understand, analyze, and explore the present problems. In this context, conceptual means that reality is reduced to its essence. A model has a high level of abstraction and reduces reality to what is relevant at this generic level.

A conceptual modeling language can be standardized (internationally) and is then referred to as a formal modeling language. An example of this is the widespread and frequently applied modeling language UML (Unified Modeling Language).

A model has a number of properties that are explored further in the following sections:

- A model is made for a specific purpose.
- A model gives a representation of reality.
- A model is used to reduce information so that we can better understand reality or focus on part of the reality.

A model is an abstract representation of an existing part of reality or a part of reality to be created. Reality can be presented from two angles, the descriptive and the prescriptive.

A model is an abstract representation of reality

A descriptive model shows the current reality and reflects the requirements that are met. If no descriptive model is available yet, such a model is the result of the analysis of the current situation. Such a model is also called the original.

A model represents the reality in a descriptive or prescriptive way

A prescriptive model indicates what future reality is expected or required. If a descriptive model exists for the given situation, then the prescriptive model can be derived from the original by indicating which requirements will be new, changed, or are no longer needed. It describes the ultimate situation desired.

A model reduces information

Reality can be complex. If we apply "too many" details, a model can be hard to grasp. This complex reality can be simplified by reducing the amount of information in the model. In a model, we can omit irrelevant information. Reducing the amount of information can give us a better understanding of reality and allow us to understand the essence of this reality more easily. Based on the intended purpose (first property) for which the model is applied, only the relevant information is displayed in the model.

Please note, if "too much" information is reduced, a clouded or incorrect image of reality may arise. Thus, careful consideration should be given to how much of the information can be reduced without distorting reality.

There are several ways to reduce information:

- By compression or aggregation
Aggregating information is a way to make information more abstract. The information is stripped of irrelevant details and is therefore more compact.
The information is, as it were, condensed.
- By selection
By selecting only the relevant information, and not everything, it is possible to indicate what the subject under consideration is. The focus is on a specific part or number of parts of the total.

Both ways of reducing information can also be applied together.

A model suits a specific purpose

A model is a representation of reality and each model represents certain aspects of reality. For example, a construction drawing shows the breakdown of the space in a building and an electrical diagram shows the wiring of the electrical circuit. Both models represent the building for a specific purpose. A model is made for a specific purpose in a specific context. In the example above, the context is the design and/or realization of a building. The various construction drawings represent information about a specific aspect of the building. This makes it immediately clear that a specific model can be used only if it fits the purpose for which the model was made.

3.4.1.3 Advantages and Disadvantages of Modeling Requirements

Compared with natural languages, models have the following advantages, among others:

Advantages of models

- The elements and their connections are easier to understand and to remember.
A picture tells more than a thousand words. A picture, and also a model, can be easier to grasp and to remember. Note that a model is not self-explanatory and needs extra information—i.e., a legend, examples, scenarios, etc.
- The focus on a single aspect reduces the cognitive load needed to understand the requirements modeled.
Because a model has a specific purpose and a reduced amount of information, understanding the reality modeled can require less effort.
- Requirements modeling languages have a restricted syntax that reduces possible ambiguities and omissions.
Because the modeling language (syntax and semantics) is simpler—i.e., limited number of notation elements and stricter language rules compared with natural language—the risk of confusion and omissions is smaller.
- Higher potential for automated analysis and processing of requirements.
Because a modeling language is more formal (limited number of notation elements and stricter language rules) than a natural language, it lends itself better to automating the analysis or processing of requirements.

Despite the great advantages for visualizing requirements with models, models also have their limitations.

Disadvantages of models

- Keeping models that focus on different aspects consistent with each other is challenging.
If multiple models are used to describe the requirements, it is important to keep these models consistent with each other. This requires a lot of discipline and coordination between the models.
- Information from different models needs to be integrated for causal understanding.
If multiple models are used, all models must be understood to enable a good understanding of the requirements.
- Models focus primarily on functional requirements.
The models for describing quality requirements and constraints are limited if not lacking in specific context. These types of requirements should then be supplied in natural language together with the models—for example, as a separate work product.
- The restricted syntax of a graphic modeling language implies that not every relevant item of information can be expressed in a model.
Because a model is made for a specific purpose and context, it is not always possible to record all requirements in the model or in multiple models.
Requirements that cannot be expressed in models are added to the model as natural language requirements or as a separate work product.

Therefore, requirements models should always be accompanied by natural language [Davi1995].

3.4.1.4 Application of Requirements Models

As indicated in the previous sections, there are common models for various contexts. For example, in architecture, you have construction drawings, piping diagrams, electrical diagrams, etc. to express the specifications of a building. In other contexts—for example, software development—there are modeling languages that are useful in these types of context. An important aspect in applying models is to use models that are common in the context or that have been specially developed for a specific context.

Many modeling languages—for example, UML [OMG2017] or BPMN [OMG2013]—have been standardized. When requirements are specified in a non-standard modeling language, the syntax and semantics of the language should be explained to the reader—for example, via a legend.

Models are used to describe the requirements from a certain perspective. In system development, functional requirements are categorized in the following perspectives (see also Section 3.1.4):

- ▶ **Structure and data**
Models that focus on the static structural properties of a system or a domain
- ▶ **Function and flow**
Models that focus on the sequence of actions required to produce the required results from given inputs or the actions required to execute a (business) process, including the flow of control and data between the actions and who is responsible for which action
- ▶ **State and behavior**
Models that focus on the behavior of a system or the life cycle of business objects in terms of state-dependent reactions to events or the dynamics of component interaction

The nature of the system being modified or built gives direction to the models to be used. For example, if the nature of the system is to process information and relationships, then it is expected that there are quite a lot of functional requirements that describe this information and these relationships. As a result, we use a matching modeling language that lends itself to modeling data and its structure.

The nature of a system helps in the selection of the appropriate model

Naturally, a system will consist of a combination of the above perspectives. It follows that a system needs to be modeled from multiple perspectives. Sections 3.4.3 to 3.4.5 elaborate the different models for each perspective in more detail.

Before the requirements are elicited and documented—for example with models—an inventory is taken of goals and context. These can also be modeled, see Sections 3.4.5.1 respectively 3.4.2.

Applying models helps us mainly in the following ways:

- ▶ *Specifying* (primarily functional) requirements in part or even completely, as a means of replacing textually represented requirements
- ▶ *Decomposing* a complex reality into well-defined and complementing aspects; each aspect being represented by a specific model, helping us to grasp the complexity of the reality

- ▶ *Paraphrasing* textually represented requirements in order to improve their comprehensibility, in particular with respect to relationships between them
- ▶ *Validating* textually represented requirements with the goal of uncovering omissions, ambiguities, and inconsistencies

Modeling the requirements also helps with structuring and analyzing knowledge. You can use diagrams to structure your own thoughts to get a better understanding of the system and its context.

3.4.1.5 Quality Aspects of a Requirements Model

This is a supplementary section for which there will be no questions in the CPRE Foundation level exam.

A substantial part of the requirements models are diagrams or graphical representations. The quality of the requirements model is determined by the quality of the individual diagrams and their mutual relationships. In turn, the quality of the individual diagrams is determined by the quality of the model elements within the diagrams. The quality of the requirements models and model elements can be assessed against three criteria [LiSS1994]:

1. Syntactic quality
2. Semantic quality
3. Pragmatic quality

The syntactic quality expresses the extent to which a single model element (graphical or textual), requirements diagram, or requirements model complies with the syntactic specifications. If, for example, a model that describes the requirements as a class model contains modeling elements that are not part of the syntax, or model elements are misused, then this will decrease the syntactic quality of the model. A stakeholder of this model—for example, a tester—might misinterpret the information that is represented by the model. This might eventually lead to inappropriate test cases.

The quality of a model is determined by three criteria

Syntactic quality

Requirements modeling tools provide facilities for checking the syntactic quality of the models.

The semantic quality expresses the extent to which a single model element (graphical or textual), the requirements diagram, or the requirements model correctly and completely represents the facts.

Semantic quality

Just like in natural language, semantics gives meaning to the words. If a term can have different meanings or there are several terms that mean the same thing, this can lead to miscommunication. The same applies to the semantics of modeling elements. If the modeling elements are misinterpreted or applied incorrectly, the model may be misinterpreted.

The pragmatic quality expresses the extent to which a single model element (graphical or textual), the requirements diagram, or the requirements model is suitable for the intended use—that is, whether the degree of detail and abstraction level is appropriate for the intended use and whether the appropriate model is selected with respect to the domain or context. This can be assessed if the purpose and the stakeholders of the diagram are known. Intermediate versions of the model can be submitted to the stakeholders interested to validate whether the diagrams fit their purpose.

Pragmatic quality

During validation of the requirements, the quality of the modeling diagrams used is assessed to make sure that these diagrams fit their intended purpose and usefulness.

3.4.1.6 Best of Both Worlds

As explained in the previous section, requirements that are expressed in textual or visual/graphical form (i.e., via requirements models) have their advantages and disadvantages. By using both textual and graphic representations of the requirements, we can harness the power and benefits of both forms of representation.

Amending a model with textual requirements adds more meaning to the model. Another useful combination is that we can link quality requirements and constraints to a model or specific modeling element. This provides a more complete picture of the specific requirements.

Using models can also support the textual requirements. Adding models and images to the textual requirements supports these models for a better understanding and overview.

Document requirements in natural language and models to benefit from the strengths of both approaches

3.4.2 Modeling System Context

Chapter 2, Principle 4 introduces the notion that requirements never come in isolation and that the system context, such as existing systems, processes, and users need to be considered when defining the requirements for the new or changed system.

Context models specify the structural embedding of the system in its environment, with its interactions to the users of the system as well as to other new or existing systems within the relevant context. A context model is not a graphical description of the requirements but is used to reveal some of the sources of the requirements. Figure 3.4 provides an abstract example of a system and its environment, with its interfaces to the users of the system and its interfaces to other systems. Thus, context diagrams help to identify user interfaces as well as system interfaces. If the system interacts with users, the user interfaces must be specified in a later step during RE. If the system interacts with other systems, the interfaces to these systems must be defined in more detail in a later step. Interfaces to other systems may already exist or may need to be developed or modified.

Context models help to understand the context and boundaries of a system

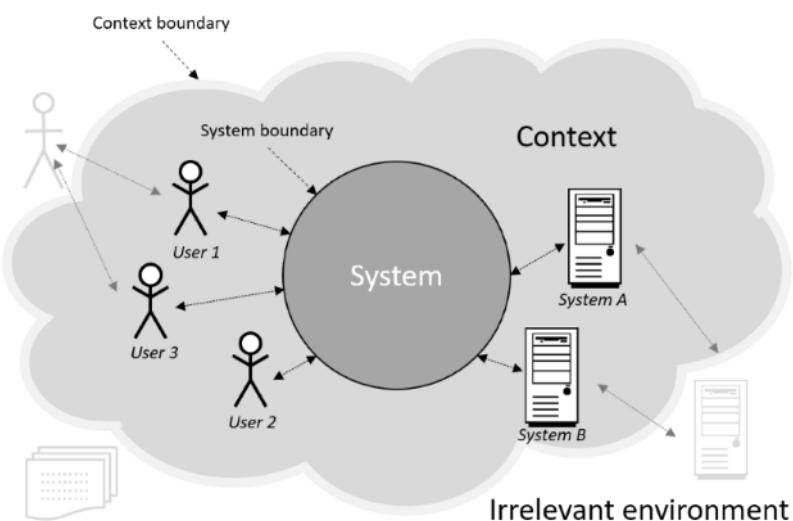


Figure 3.4 A system in its context

Even if there is no standardized modeling language for context models, context models are frequently represented by:

- Data flow diagrams from structured analysis [DeMa1978]
- UML use case diagrams [OMG2017]

Note: the UML use case model consists of two elements; the use case diagram (see Figure 3.6) and the use case specification (Section 3.4.2.2). This chapter focuses on modeling with the use case diagrams.

- Tailored box-and-line diagrams [Glin2019]

In the systems engineering domain, SysML block definition diagrams [OMG2018] can be adapted to express context models by using stereotyped blocks for the system and the actors.

In the next two subsections, we introduce the notation of data flow diagrams (DFD) and UML use case diagrams to model the context of a system. These two examples do not describe the complete context but emphasize the context from a specific viewpoint.

3.4.2.1 Modeling the System Context with a Data Flow Diagram (DFD)

The system context can be viewed from different perspectives. The structured analysis of systems [DeMa1978] talks about the context diagram. This diagram is a special data flow diagram (DFD) where the system is represented by one process (the system). Figure 3.5 shows an example of a context diagram.

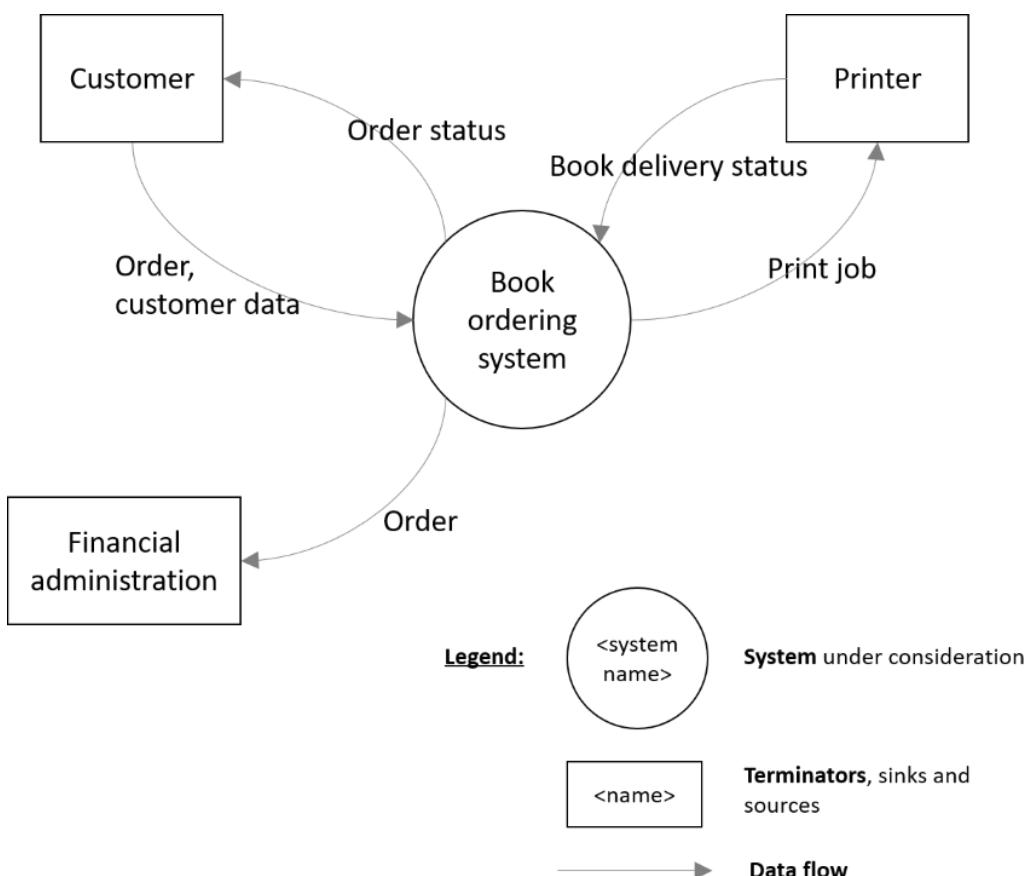


Figure 3.5 Example of a context diagram using a DFD

The system is placed centrally in the model. It has a clear name so that the readers know which system is being considered.

The rectangles around the system are terminators: customer, printer, and financial administration. A terminator that provides information or services to the system is called a *source*. A terminator that takes information or services from the system is called a *sink*. A terminator can take either role depending on the data provided or retrieved, such as the customer in the example above.

The arrows in the example show how the information from the terminators flows into the system (source) and from the system to the terminators (sinks). The arrows are given a logical name that describes what information is transferred. Irrelevant details are omitted at the context diagram level. The information flow between the customer and the system contains, for example, *customer data*. What information (name, date of birth, email address, telephone number, delivery address, billing address, etc.) makes up the *customer data* does not have to be relevant yet for this level of abstraction.

A DFD gives insights into the interface of the system with its context

The flow of information can consist of tangible (materials) and intangible (information) objects. Also, at this conceptual level, there is no reference (yet) to *how*—email, website, form, etc.—the information is provided.

Adding extra details to the context diagram can make it clearer to the stakeholders involved and may help to improve the shared understanding. These details need to be worked out for each individual situation.

Using a data flow diagram to model the context of a system provides some insights into the interactions of the system with its environment, for example:

- ▶ The interfaces to people, departments, organizations, and other systems in the environment
- ▶ The (tangible and intangible) objects that the system receives from the environment
- ▶ The (tangible and intangible) objects that are produced by the system and are delivered to the environment

A data flow diagram indicates a clear boundary between the system and its environment. The relevant users and systems of the environment are identified during elicitation of requirements (Section 4.1). DFD context diagrams can help to structure the context to reach a shared understanding of the system context and the system boundary.

3.4.2.2 Modeling the System Context with a UML Use Case Diagram

Another view of the context of a system can be reached from a functional perspective. The UML use case diagram is a common approach for modeling the functional aspects of a system and the system boundaries, along with the system's interactions with users and other systems. Use cases provide an easy way to systematically describe the various functions within the defined scope from a user perspective. This is different to DFD context diagrams, where the system is represented as a big black box.

Use cases were first proposed as a method for documenting the functions of a system in [Jaco1992]. The UML use cases consists of use case diagrams with associated textual use case specifications (see Section 3.3.2). A use case specification specifies each use case in detail by, for example, describing the possible activities of the use case, its processing logic, and preconditions and postconditions of the execution of the use case. The specification of use cases is essentially textual—for example, via use case templates as recommended in [Cock2001].

As mentioned, a UML use case diagram shows the functions (use cases) from the point of view of the direct users and other systems that interact with the system under consideration. The name of the use case is often composed of a verb and a noun. This gives a brief description of the function offered by the system, as shown by the example in Figure 3.6.

A use case diagram gives insights into the functionality provided to the direct users in the context

The actors are the direct users or systems that interact with the system under consideration. The actor (user or system) that starts the use case receives the benefit that the use case delivers (e.g., showing the status of an order to the customer). The association connects the actor with the relevant use case but it does not document any direction or data flow (as is done in DFDs); it expresses only that the actor receives the benefit from the use case.

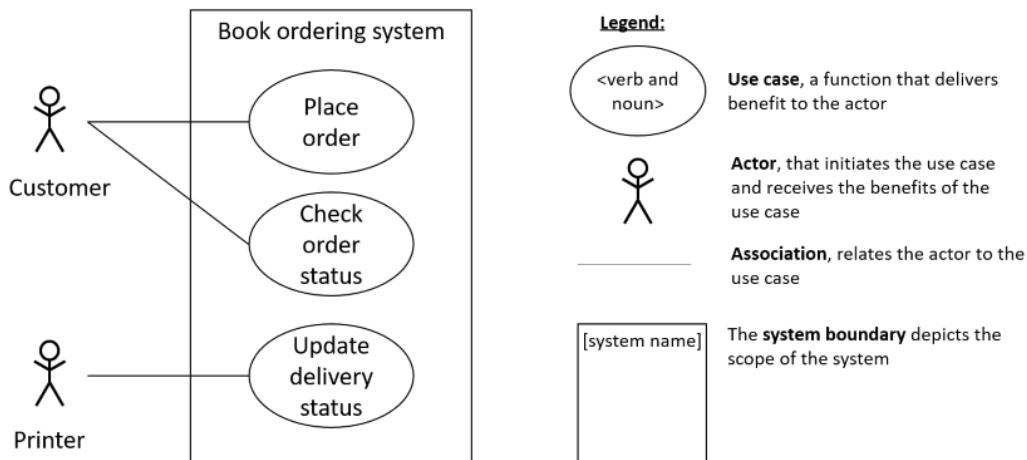


Figure 3.6 Example of a context diagram using a use case diagram

A UML use case diagram describes the functionality that the system offers to its environment. The separation between the functionality in the system and the actors in the context is visualized with the system boundary (rectangle around the use cases, e.g., "book ordering system"). Use case diagrams support sharpening of the system boundary and checking whether the functional scope of the system at a high level is covered.

Each use case also includes a detailed use case specification, documenting the preconditions, trigger, actions, postconditions, actors, and so forth. Use cases are usually described using a template (Section 3.3). If the scenarios of a use case become complex or large, the recommendation is to visualize the scenarios with UML activity diagrams, see Section 3.4.4.1. The detailed specification of use cases is not part of context modeling and can be elaborated at a later time, when this information becomes relevant.

3.4.3 Modeling Structure and Data

For functional requirements from the perspective of business objects (see Section 3.1.4), different data models are available. A (business) object can be a tangible or intangible object, such as a bicycle, pedal, bicycle bell, but also a training request, a shopping basket with digital products, and so on. A (business) object is "something" in the real world. Some (or maybe all) of these (business) objects are used by the system under consideration. The system uses these objects as input to process, to persist, and/or to deliver output. Data models are used to describe the (business) objects that must be known by the system. These kinds of diagrams model the object, attributes of the object, and the relationships between objects. For the sake of simplicity, we refer to modeling structure and data—these, however, represents information structures between (business) objects in the real world.

A number of common models for depicting structure and data are:

- ▶ Entity relationship diagrams (ERD) [Chen1976]
- ▶ UML class diagrams [OMG2017]
- ▶ SysML Block Definition Diagrams [OMG2018]
- ▶ Building information models (BIM) [ISO19650]: these model the elements required to plan, build, and manage buildings and other construction elements. The details of models outside the system domain are not covered by this handbook.

Common models for depicting structure and data

To explain the concept of modeling structure and data, this chapter uses the UML class diagram as an example. UML, short for Unified Modeling Language, consists of an integrated set of diagrams. This set of diagrams is a collection of best engineering practices and has proven successful in modeling complex and large systems. UML was designed by Grady Booch, James Rumbaugh, and Ivar Jacobson in the 1990s and it has been a standardized modeling language since 1997.

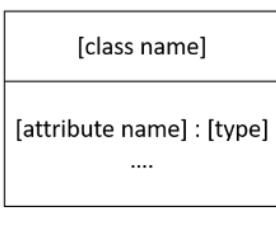
If more depth or a different model is desired, read the literature referred to and practice with the desired modeling language.

3.4.3.1 Modeling Structure and Data with UML Class Diagrams

UML is a collection of different models that can be used to describe a system. One of these models is the class diagram. A class diagram depicts a set of classes and associations between them. We discuss only the common and simple notation elements of this model. If more depth is desired, we refer to the literature or the CPRE Advanced Level Requirements Modeling.

In the overview below you will find the most common notation elements.

Most common notation elements of UML class diagrams

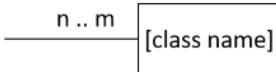


A **class** describes a set of (material or immaterial) objects that have a similar structure, behaviour and relationships.

An **attribute** describes a property of the class. The attribute is expressed as a certain **type** that restricts the values that can be assigned to the attribute.



An **association** relates two classes to each other.



Multiplicities define how many instances of the class at the corresponding association end can participate in the association to a given instance of the class at the other end of the association.

Where $n \in \mathbb{N}$ and $m \in \mathbb{N}$. Some examples are:

- 0 .. 1 (zero or one time)
- 0 .. * (zero or more times)
- 1 .. * (one or more times)
- 7 (exactly seven times)
- 1 (default value, exactly one time)



The **role** name defines which role the class plays in the association

Figure 3.7 Subset of the modeling elements of UML class diagrams

In a class model, you will find the concepts and terms that are relevant in the domain. These concepts include a clear definition that is included in the glossary. With the use of data models, the glossary is extended with information about the structure and coherence of the terms and concepts. A clear definition and coherence of the terms used prevents miscommunication about the matter under consideration.

Figure 3.8 shows a simplified model of the book ordering system (see examples of the context in Section 3.4.2). The static information that the system needs to perform its functionality—ordering a book—is modeled.

A customer orders a book and hence information is persisted for the classes *Customer*, *Order*, and *Book*. A customer can place an order and therefore a relationship (association) exists between the *Customer* and the *Order*. A customer can place multiple orders over time and he/she only becomes a customer if he/she places an order. This information determines the multiplicity: 1 customer places 1 or more orders.

The fact that a customer can order a book means that there is also a relationship between the classes *Order* and *Book*. To keep the example simple, here, the customer can order only one book at a time. Also, an order must contain at least one book. An order that has no book is not an order.

In the class *Book*, the attribute *inStock* is also maintained. Information such as "if the stock is not sufficient to fulfill the order, then a print job is sent to the printer" cannot be modeled. This is a type of information that cannot be modeled in a class diagram because it describes a certain functionality of the system. This information is part of the requirements and should be documented in another work product. It can be added as a textual requirement that accompanies the class diagram, or be modeled with another diagram—for example, a UML activity diagram (see Section 3.4.4.1).

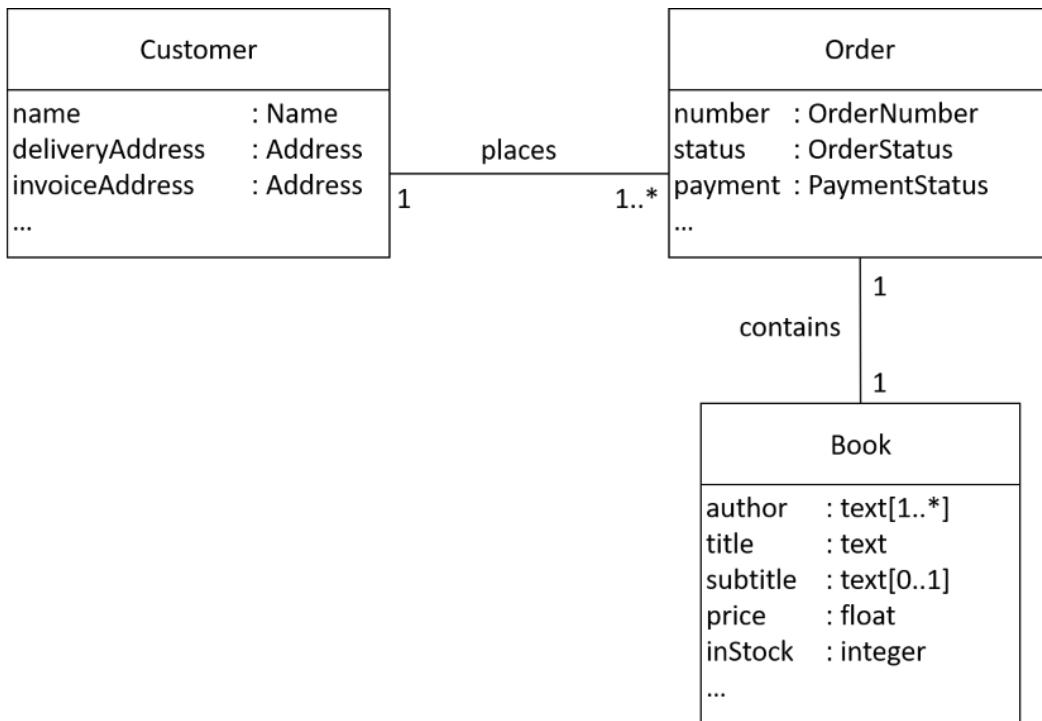


Figure 3.8 Example of a simple UML class diagram

3.4.4 Modeling Function and Flow

Function and flow describe how the (sub)system shall transform input into output. We can visualize this type of requirement with models that depict function and flow.

Unlike modeling data, which essentially needs only one diagram type, function and flow can be viewed from different angles. Depending on the needs of the stakeholders to take the next step in the development process, more than one model might be needed to document the requirements about function and flow.

Some common models for depicting function and flow are:

- Business Process Modeling Notation (BPMN) [OMG2013]

These process models are used to describe business processes or technical processes. BPMN is frequently used to express business process models.

- UML use case diagram [OMG2017]

See Section 3.4.2.2.

- UML activity diagram [OMG2017]

See Section 3.4.4.1.

- UML sequence diagram [OMG2017]

See Section 3.4.5.1.

- Data flow diagram [DeMa1978]

See Section 3.4.2.1.

- Domain story models [HoSch2020]

These models specify visual stories about how actors interact with devices, artifacts, and other items in a domain, typically using domain-specific symbols. They are a means for understanding the application domain in which a system will operate.

Common models for depicting function and flow

To explain the concept of modeling function and flow, we limit this section to a few examples of UML diagrams. If more depth or a different model is desired, read the literature referred to and practice with the relevant modeling language.

3.4.4.1 UML Activity Diagram

UML activity models are used to specify system functions. They provide elements for modeling actions and the control flow between actions. Activity diagrams can also express who is responsible for which action. Advanced modeling elements (not covered by this handbook) provide the means for modeling data flow.

A UML activity diagram expresses the control flow of activities of a (sub)system. Flow thinking comes from visualizing program code with flow charts (according to [DIN66001], [ISO5807]). This helped programmers to conceive and understand complex structures and flows in programs. With the introduction of UML [OMG2017], a model has been introduced for visualizing activities and actions from a functional perspective.

In the overview below you will find the basic notation elements.

Basic notation elements
of UML activity
diagrams



The **start node** indicates the starting point of the activity diagram.



The **end node** indicates the end point of the activity flow. There can be multiple end nodes.



The **control flow termination** ends the activity flow.

[name]

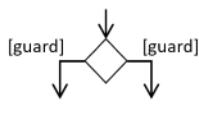
An **action/activity** represents the non-interruptible action of objects. An executable computation that results in a change in state of the system or the return of a result.



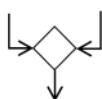
The **action or control flow** illustrates the transitions from one action state to another (also called edges and paths).

Figure 3.9 Basic notation elements of the UML activity diagram

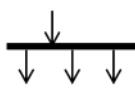
With this set of basic notation elements, you can set up a simple sequential activity diagram. If more control is required, the model can be extended with decisions and parallel flows of activities using the notation elements below.



The **decision node** branches the activity flow based on a guard. The outgoing alternatives should be labeled with a condition or guard expression. The guard must be true before the move to the appropriate flow.

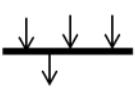


The **merge node** brings together multiple flows that are not concurrent.



Synchronization bar

A **fork node** is used to split a single incoming flow into multiple concurrent flows.



A **join node** joins multiple concurrent flows back into a single outgoing flow. The action following the join node is executed after all parallel actions have been executed.

A fork and join mode are used together and are therefore often referred to as synchronization.

Figure 3.10 Decisions and parallel flows in a UML activity diagram

Activity diagrams can be used to specify the processing logic of use case scenarios in detail (see Section 3.3.2). Activity diagrams are created to visualize the scenarios, which are processes with activities and processing logic. As long as the diagram remains understandable, the main scenario can be modeled jointly with the alternative scenarios and the exception scenarios as part of the same diagram.

Figure 3.11 gives a simple example of the book ordering system. This simplified flow of action starts when the customer sends in their order. First, the *Order* and the *Customer* information are validated to determine whether all (necessary) information is supplied. If either the *Order* or the *Customer* information is invalid (incorrect or insufficient), then a notification is sent to the customer and the order process is canceled. The basic scenario is that the *Order* and *Customer* information are valid. The scenario that the *Order* or *Customer* information is invalid is called an exceptional flow and handles a functional faulty condition in the process.

If both *Order* and *Customer* information are correct, then the stock is checked. If there is a sufficient number of products in stock, the *Order* is picked and sent to the *Customer*. An alternative flow is started if there are insufficient products in stock. A print job request is sent to the *Printer* and a notification for a redelivery is sent to the *Customer*.

Within the book ordering system, there are also other flows that are separated from the order and delivery process. For example, the payment, redelivery, and invoice processes have separate flows to allow a clear separation of concerns. If, for example, the decision is taken to no longer keep any products in stock, then the order and delivery process still applies. If changes are needed in this flow, these changes may not affect the other flows. This decomposition of functionality helps to keep things simple and clear.

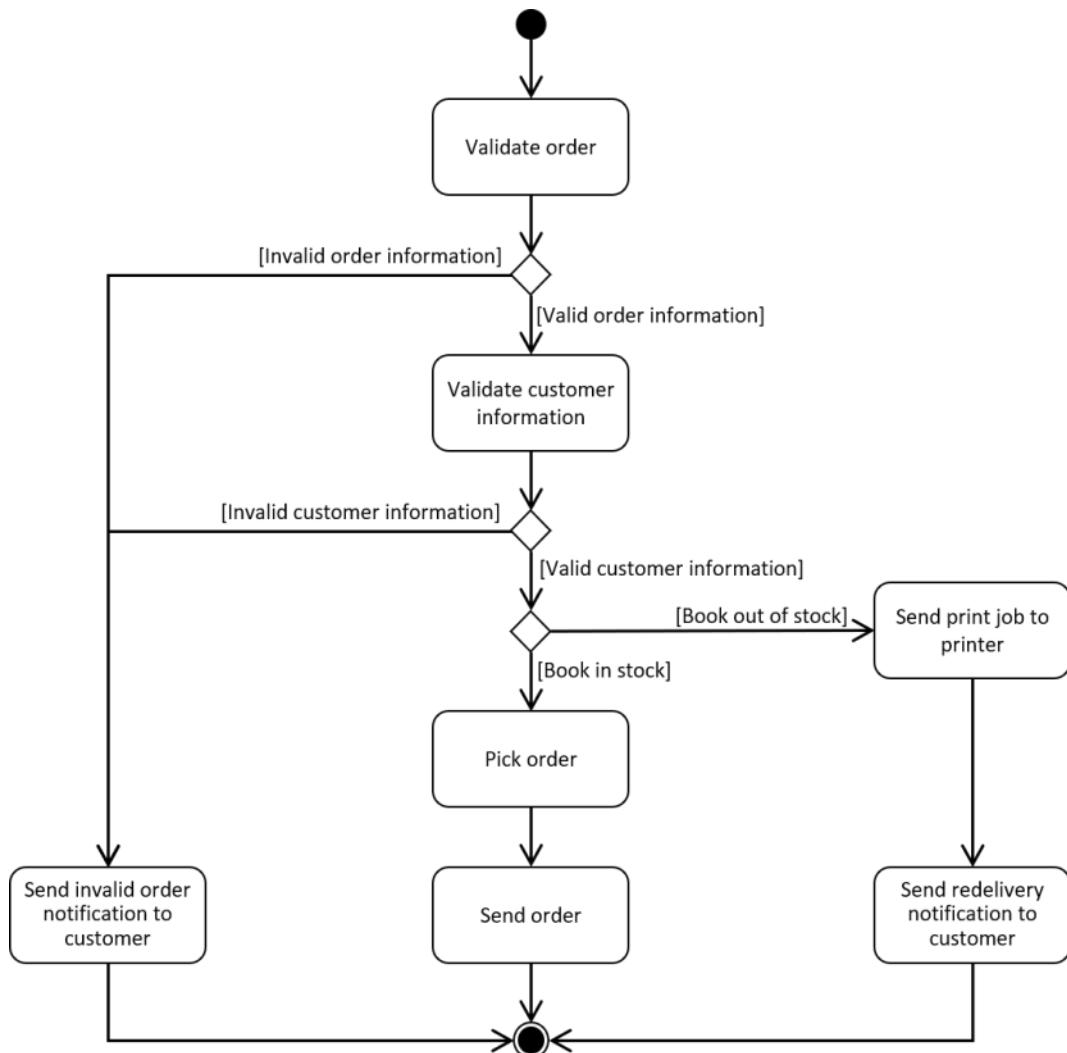


Figure 3.11 Example of a UML activity diagram

3.4.5 Modeling State and Behavior

Functional requirements that describe the behavior, states, and transitions of a (sub)system or that of a business object are requirements in the behavioral perspective. An example of a system state is *On*, *Standby*, or *Off*. A business object can have a life cycle that goes through a number of prescribed states. For example, a business object *Order* can be in the following states: *Placed*, *Validated*, *Paid*, *Shipped*, and *Completed*.

A technique widely used to describe the behavior of a system is statecharts [Hare1988]. Statecharts are state machines with states that are decomposed hierarchically and/or orthogonally. State machines, including statecharts, can be expressed in the UML modeling language [OMG2017] with state machine diagrams (also called state diagrams).

State diagrams describe state machines that are finite. This means that these systems eventually reach a final state. A state diagram shows the states that the system or an object can take. It also indicates how to switch state—that is, the state transition. A system does little by itself. Switching the state requires a trigger from the system or from the environment of the system.

Common models for representing behavior and states include:

- ▶ Statecharts [Hare1988]
- ▶ UML state diagram [OMG2017]

To explain the concept of modeling behavior and states, this chapter uses the UML state diagram as an example. If more depth or a different model is desired, read the literature referred to and practice with the relevant modeling language.

In the overview below you will find the basic notation elements.

*Basic notation elements
of UML state diagrams*

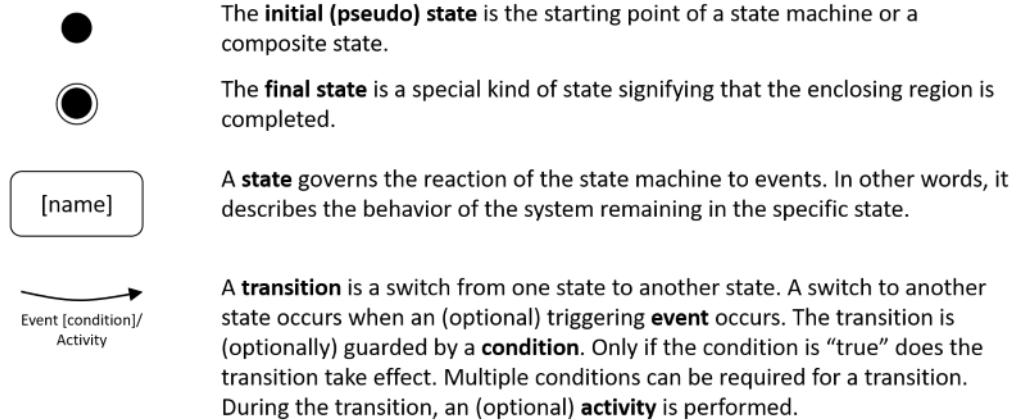


Figure 3.12 Basic notation elements of the UML state diagram

As discussed at the beginning of the section, a state diagram can clarify the states an object can take. We see here an opportunity to visualize additional (and partly redundant) information of an object. Imagine that you order a book on a website and you want to track the status of your order. An order is used in the real world and is modeled as a business object in a class diagram (see Figure 3.8) with, most likely, an attribute *status*. The class diagram indicates that the attribute *status* can assume a limited number of values, such as *Validated*, *Paid*, *Delivered*, *Canceled*, and so on. The class diagram does not describe the order of possible status changes. A class diagram does not describe the behavior of the system in a certain "status" either. This can be made clear with a UML state diagram—for example, that an offered order cannot go directly to the status *Delivered* without the customer having paid for the order.

Figure 3.13 gives an example of a state diagram of the book ordering system. In the class diagram (Figure 3.8) of the book ordering system, an object *Order* is modeled. This object has an attribute *status* that can have a limited number of values. These values are listed and explained in the class diagram. What a class diagram does not describe is the sequence in which the order is processed. A state diagram visualizes the states and transitions between the states, making it clear what the sequence of the order status is. The state diagram shows, for example, that the order cannot be sent before it is completely picked (transition between the states *Picked* and *Sent*).

Also, if the order is in the state *Sent*, the next state can only be *Paid*. A transition from *Sent* to *Handled* is not possible. This diagram also makes clear that payment happens after the book is sent. You can ask the stakeholders whether this is what they need or have requested.

A transition may direct to the same status. This situation is visible in the state *Picked*. Each time the order is not picked to completion, it stays in the same state to prevent it from sending an incomplete order. Only when the order is completely picked is it then sent to the customer.

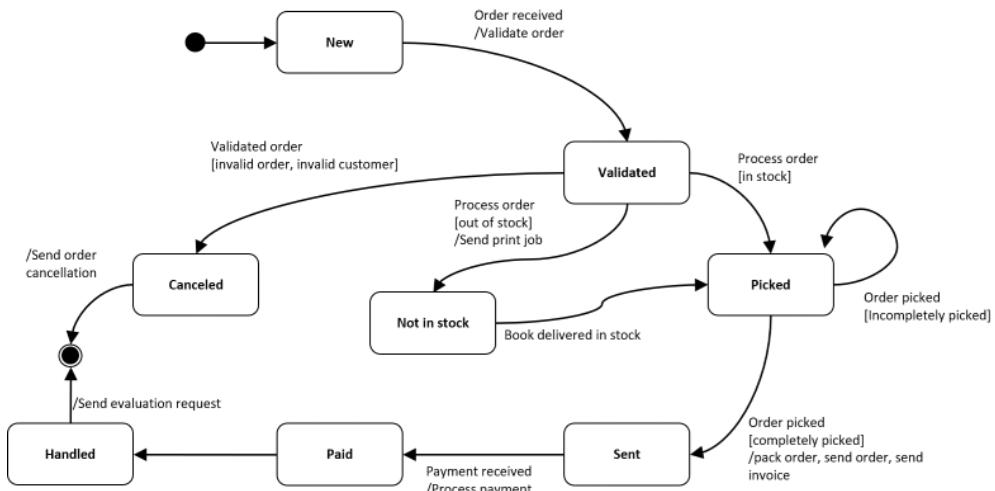


Figure 3.13 Example of a UML state diagram

A few months after the release of the book ordering system, customers complained that they did not have the ability to cancel an order. It was agreed that a customer could cancel the order in each state of the order process. Modeling this new requirement means that a transition to *Canceled* is needed from each state. This might make the diagram difficult to read. Adding a textual requirement to describe this behavior might be a way to keep the model simple for the audience.

3.4.5.1 UML Sequence Diagram

The following UML diagram is supplementary and will not be questioned in the CPRE Foundation level exam.

The UML sequence diagram is used to depict the interaction between communication partners and to model the dynamic aspect of systems. The communication partners are actors, systems, components, and/or objects within a system.

The interaction displays the sequence of messages (a scenario) between these communication partners. The interaction that takes place between the communication partners realizes the purpose of a scenario, respectively (a part) of a use case.

In the overview below you will find the basic notation elements.

*Basic notation elements
of UML sequence
diagrams*

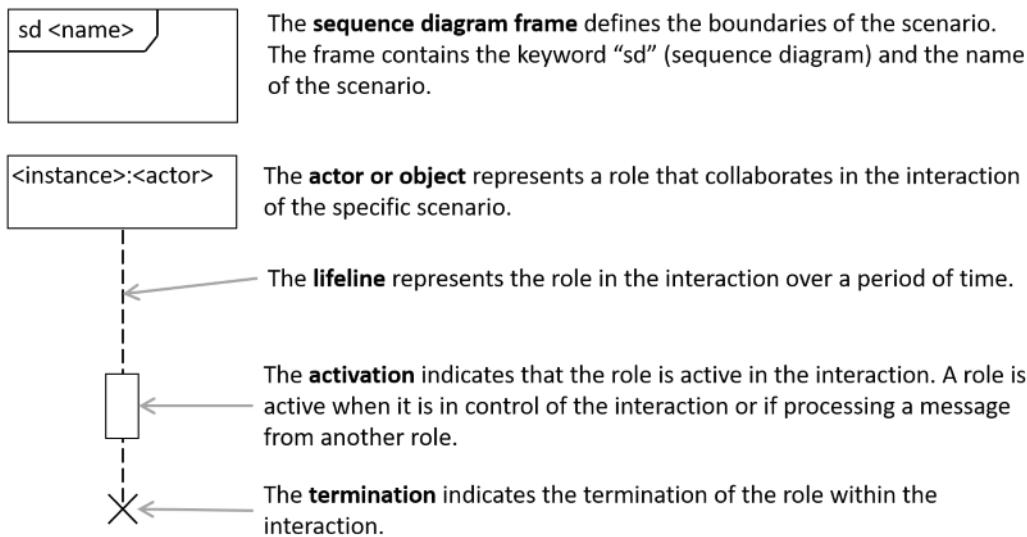


Figure 3.14 Basic notation elements of the UML sequence diagram

A lifeline in a scenario depicts the role in the scenario, meaning the instance of an actor. When sequence diagrams are modeled, the instance name of an actor or object is often omitted. The roles that participate in the communication interact with each other by sending messages. There are two types of messages that are used in the interaction.

*Basic notation elements
of messaging in UML
sequence diagrams*

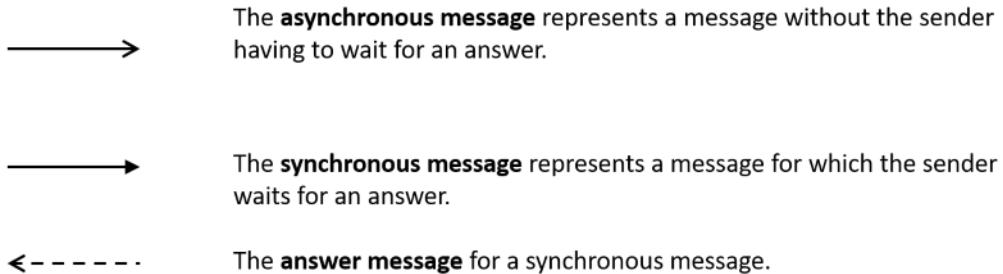


Figure 3.15 Basic notation elements of messaging in the UML sequence diagram

A message can also be sent from or to objects outside the scenario. This is represented as a filled-in circle. The sender or receiver of these kinds of messages may be unknown.

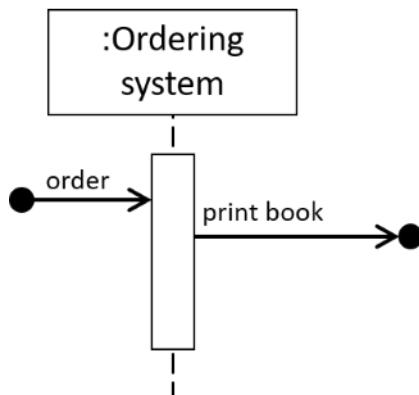


Figure 3.16 Messages from and to an object outside the scenario

Figure 3.17 shows a model of the scenario in which a customer orders a book and that specific book is out of stock. The *Customer* asks to place an *Order*. If the *Order* is invalid, a notification that the *Order* is canceled is returned. If the *Order* is valid, the stock is checked and if a book is out of stock, a print job is sent to the *Printer*. This is an asynchronous message because it might take some time to print the book. A notification is sent to the *Customer* that the book is out of stock and will be redelivered. The *Order* is deactivated until the book is delivered by the *Printer*.

When the book is received from the *Printer*, the *Order* object is activated again. The order is picked and sent to the *Customer*. This completes the *Order* and a last notification of the status is sent to the *Customer*.

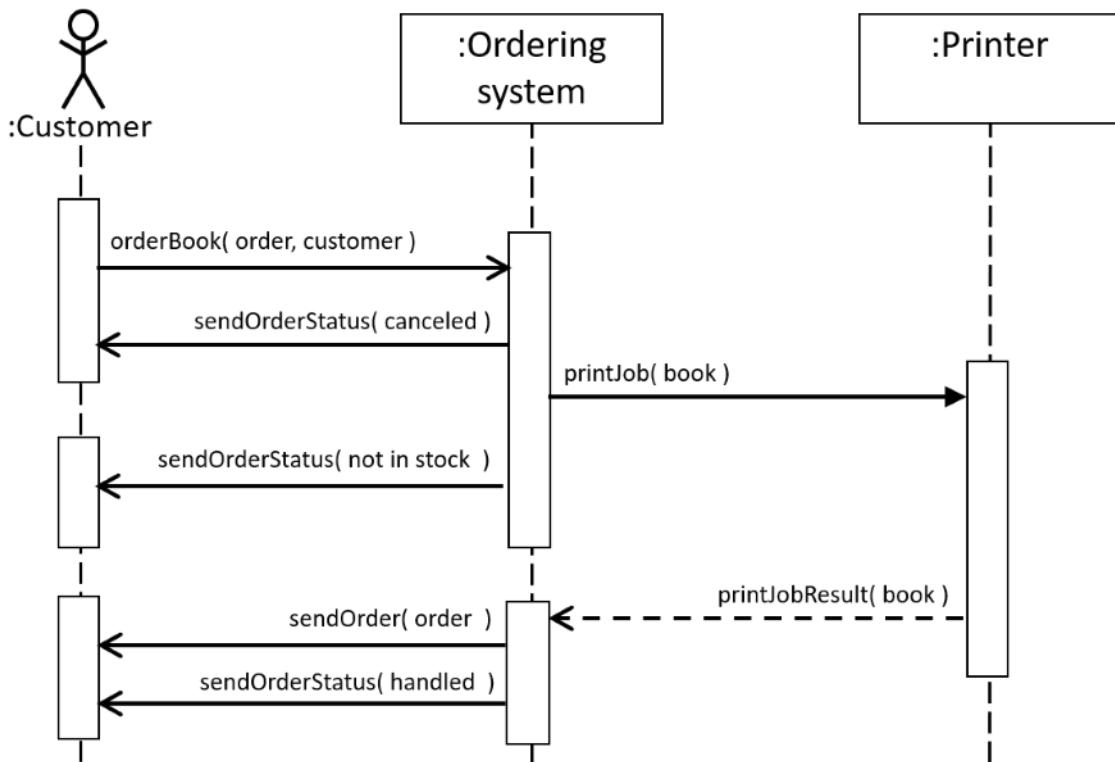


Figure 3.17 Example of a UML sequence diagram

3.4.6 Modeling Goals

Business requirements describe a business goal or need. They describe the end result that the solution must meet and with which the (business) problem is solved, see Chapter 2, Principle 5. To ensure that the focus is on solving the problem and that the effort focuses on adding value, goals are carefully described. In Requirements Engineering, there are several ways to document goals. The most common one is the use of natural language (Section 3.2) or templates (Section 3.3). Template-based documentation forms can be found, for instance, in [Pich2010], [Pohl2010], or [RoRo2012].

There are also some model-based notations for documenting goals. The easiest and most common notation is an AND/OR tree [AnPC1994]. AND/OR trees allow us to document goals at different levels of detail and to link subgoals with goals using AND and OR relationships. An AND relationship means that all subgoals need to be fulfilled to fulfill the goal. An OR relationship is used to express that at least one of the subgoals needs to be fulfilled to fulfill the goal.

More academic modeling approaches for goals can be found in:

- ▶ Goal-oriented requirements language (GRL) [GRL2020]
This is a language that supports goal-oriented modeling and reasoning of requirements, especially for dealing with non-functional requirements.
- ▶ Knowledge acquisition in automated specification (KAOS) [vLam2009]
KAOS is a methodology that contains goal modeling. This enables analysts to build requirements models and to derive requirements documents from KAOS goal models.

Documenting goals (in textual or graphical form) is an important starting point for eliciting requirements, referring the requirements to their rationale, and identifying sources—like stakeholders—of the requirements, etc.

3.5 Glossaries

Glossaries are a core means of establishing shared understanding of the terminology used when developing a system: they help avoid people involved as stakeholders or developers using and interpreting the same terms in different ways.

Why glossaries

A good glossary contains definitions for all terms that are relevant for the system, be they context-specific terms or everyday terms that are used with a special meaning in the context of the system to be developed. A glossary should also define all abbreviations and acronyms used. If there are synonyms (that is, different terms denoting the same thing), they should be marked as such. Homonyms (that is, identical terms that denote different things) should be avoided or at least marked as such in the glossary.

Glossary content

There are a couple of rules that guide the creation, use, and maintenance of the glossary in a system development project.

Glossary rules

1. *Creation and maintenance.* To ensure that the terminology defined in the glossary is consistent and always up to date, it is vital that the glossary is managed and maintained centrally over the entire course of a project, with one person or a small group being responsible for the glossary. When defining terms, it is important that the stakeholders are involved and agree on the terminology.
2. *Usage.* In order to get the full benefit of a glossary, its use must be mandatory. Work products should be checked for proper glossary usage. Obviously, this means that everybody involved in a project must have read access to the glossary.

When an organization develops related systems in multiple projects, it makes sense to create a glossary at the enterprise level in order to achieve consistent terminology across projects.

Creating, maintaining, and using a glossary consistently avoids errors and misunderstandings concerning the terminology used. Working with glossaries is a standard best practice in RE.

3.6 Requirements Documentation Structures

It is not sufficient to work with requirements at the level of individual requirements. Requirements must be collated and grouped in suitable work products, be they explicit requirements documents or other RE-related documentation structures (such as a product backlog).

Documents and documentation structures

Document templates (see Section 3.3.3) may be used to organize such documents with a well-defined structure in order to create a consistent and maintainable collection of requirements. Document templates are available in literature [Vole2020], [RoRo2012] and in standards [ISO29148]. Templates may also be reused from previous, similar projects or may be imposed by a customer. An organization may also decide to create a document template as an internal standard.

Document templates

A requirements document may also contain additional information and explanations—for example, a glossary, acceptance criteria, project information, or characteristics of the technical implementation.

Frequently used requirements documents are:

Frequently used documents

- ▶ *Stakeholder requirements specification*: the stakeholders' desires and needs that shall be satisfied by building a system, seen from the stakeholders' perspective. When a customer writes a stakeholder requirements specification, it is called a *customer requirements specification*.
- ▶ *User requirements specification*: a subset of a stakeholder requirements specification, covering only requirements of stakeholders who are prospective users of a system.
- ▶ *System requirements specification*: the requirements for a system to be built and its context so that it satisfies its stakeholders' desires and needs.
- ▶ *Business requirements specification*: the business goals, objectives, and needs of an organization that shall be achieved by employing a system (or a collection of systems).
- ▶ *Vision document*: a conceptual imagination of a future system, describing its key characteristics and how it will create value for its users.

Frequently used alternative documentation structures are:

Frequently used documentation structures

- ▶ *Product backlog*: a prioritized list of work items, covering all requirements that are needed and known for the product
- ▶ *Sprint backlog*: a selected subset of a product backlog with work items that will be realized in the next iteration
- ▶ *Story map*: a visual two-dimensional organization of user stories in a product backlog with respect to time and content

There is no standard or universal requirements document or documentation structure. Accordingly, documents or documentation structures should not be reused from previous projects without reflection. The actual choice depends on several factors, for example:

Choosing a proper documentation form

- ▶ The development process chosen
- ▶ The project type and domain (for example, tailor-made solution, product development, or standard product customizing)

- The contract (a customer may prescribe the use of a given documentation structure)
- The size of the document (the larger the document, the more structure is needed)

3.7 Prototypes in Requirements Engineering

Prototypes play an important role both in engineering and design.

DEFINITION 3.5 PROTOTYPE: 1. In manufacturing: A piece which is built prior to the start of mass production. 2. In software and systems engineering: A preliminary, partial realization of certain characteristics of a system. 3. In design: A preliminary, partial instance of a design solution.

Prototype

Prototypes in software and systems engineering are used for three major purposes [LiSZ1994]:

Exploratory prototypes are used to create shared understanding, clarify requirements, or validate requirements at different levels of fidelity. Such prototypes constitute temporary work products that are discarded after use. Exploratory prototypes may also be used as a means of specification by example. Such prototypes must be treated as evolving or durable work products.

Exploratory prototype

Experimental prototypes (also called breadboards) are used to explore technical design solution concepts, in particular with respect to their technical feasibility. They are discarded after use. Experimental prototypes are not used in RE.

Experimental prototype

Evolutionary prototypes are pilot systems that form the core of a system to be developed. The final system evolves by incrementally extending and improving the pilot system in several iterations. Agile system development frequently employs an evolutionary prototyping approach.

Evolutionary prototype

Requirements Engineers primarily use exploratory prototypes as a means for requirements elicitation and validation. In elicitation, prototypes serve as a means of specification by example. In particular, when stakeholders cannot express what they want clearly, a prototype can demonstrate what they would get, which helps them shape their requirements. In validation, prototypes are a powerful means for validating the *adequacy* (see Section 3.8) of requirements.

Prototypes in RE

Exploratory prototypes can be built and used with different degrees of fidelity. We distinguish between wireframes, mock-ups, and native prototypes.

Wireframe

Wireframes (also called paper prototypes) are low-fidelity prototypes built with paper or other simple materials that serve primarily for discussing and validating design ideas and user interface concepts. When prototyping digital systems, wireframes may also be built with digital sketching tools or dedicated wireframing tools. However, when using a digital tool for wireframing, it is important to retain the essential properties of a wireframe: it can be built quickly, modified easily, and does not look polished nor resemble a final product.

Mock-up

Mock-ups are medium-fidelity prototypes. When specifying digital systems, they use real screens and click flows but without real functionality. They serve primarily for specifying and validating user interfaces. Mock-ups give users a realistic experience of how to interact with a system through its user interface. They are typically built with dedicated prototyping tools.

Native prototype

Native prototypes are high-fidelity prototypes that implement critical parts of a system to an extent that stakeholders can use the prototype to see whether the prototyped part of the system will work and behave as expected.

They serve both for specification by example and for thorough validation of critical requirements. Native prototypes may also be used to explore and decide about requirements *variants* for some aspect—for example, two different possible ways of supporting a given business process.

Depending on the degree of fidelity, exploratory prototypes can be an expensive work product. Requirements Engineers have to consider the trade-off between the cost of building and using prototypes and the value gained in terms of easier elicitation and reduced risk of inadequate or even wrong requirements.

3.8 Quality Criteria for Work Products and Requirements

Obviously, Requirements Engineers should strive to write good requirements that meet given quality criteria. RE literature and standards provide a rich set of such quality criteria. However, there is no general consensus about which quality criteria shall be applied for requirements. The set of criteria presented in this subsection aims to provide a proven practice at foundation level.

Many approaches

Modern RE follows a value-oriented approach to requirements (see Principle 1 in Chapter 2). Consequently, the degree to which a requirement fulfills the given quality criteria shall correspond to the value created by this requirement. This has two important consequences:

- ▶ Requirements do not have to fully adhere to all quality criteria.
- ▶ Some quality criteria are more important than others.

No universal fulfillment

We distinguish between quality criteria for single requirements and quality criteria for RE work products such as RE documents or documentation structures.

For single requirements, we recommend using the following quality criteria:

- ▶ *Adequate*: the requirement describes true and agreed stakeholder needs.
- ▶ *Necessary*: the requirement is part of the relevant system scope, meaning that it will contribute to the achievement of at least one stakeholder goal or need.
- ▶ *Unambiguous*: there is a true shared understanding of the requirement, meaning that everybody involved interprets it in the same way.
- ▶ *Complete*: the requirement is self-contained, meaning that no parts necessary for understanding it are missing.
- ▶ *Understandable*: the requirement is comprehensible to the target audience, meaning that the target audience can fully understand the requirement.
- ▶ *Verifiable*: the fulfillment of the requirement by an implemented system can be checked indisputably (so that stakeholders or customers can decide whether or not a requirement is fulfilled by the implemented system).

Quality criteria for single requirements

Adequacy and understandability are the most important quality criteria. Without them, a requirement is useless or even detrimental, regardless of the fulfillment of all other criteria. Verifiability is important when the system implemented must undergo a formal acceptance procedure.

Some people use *correctness* instead of *adequacy*. However, the notion of correctness implies that there is a formal procedure for deciding whether something is correct or not. As there is no formal procedure for validating a documented requirement against the desires and needs that stakeholders have in mind, we prefer the term adequacy over correctness.

For work products covering multiple requirements, we recommend applying the following quality criteria:

Quality criteria for RE work products

- ▶ *Consistent*: no two requirements, recorded in a single work product or in different work products, contradict each other.
- ▶ *Non-redundant*: each requirement is documented only once and does not overlap with another requirement.
- ▶ *Complete*: the work product contains all relevant requirements (functional requirements, quality requirements, and constraints) that are known at this point in time and that are related to this work product.
- ▶ *Modifiable*: the work product is set up in such a way that it can be modified without degrading its quality.
- ▶ *Traceable*: the requirements in the work product can be traced back to their origins, forward to their implementation (in design, code, and test), and to other requirements they depend on.
- ▶ *Conformant*: if there are mandatory structuring or formatting instructions, the work product must conform to them.

3.9 Further Reading

Mavin et al. [MWHN2009] introduce and describe the EARS template. Robertson and Robertson [RoRo2012] describe the Volere templates. Goetz and Rupp [GoRu2003], [Rupp2014] discuss rules and pitfalls for writing requirements in natural language. Cockburn [Cock2001] has written an entire book about how to write use cases. Lauesen [Laue2002] discusses task descriptions and also provides some examples of real-world RE work products.

The ISO/IEC/IEEE standard 29148 [ISO29148] provides many resources concerning RE work products: phrase templates, quality criteria for requirements, and detailed descriptions of the content of various RE work products, including a document template for every work product. Cohn [Cohn2010] has a chapter on how to frame requirements in a product backlog.

Gregory [Greg2016] and Glinz [Glin2016] discuss the problem of how detailed requirements should be specified and to what extent complete and unambiguous requirements specifications are possible.

Numerous publications deal with using models to specify requirements. The UML specification [OMG2017], as well as textbooks about UML, describe the models available in UML. Hofer and Schwentner [HoSch2020] introduce domain modeling with domain storytelling. [OMG2013] and [OMG2018] describe the modeling languages BPMN for modeling business processes and SysML for modeling systems, respectively. The books by Booch, Rumbaugh, and Jacobson [BoRJ2005], [JaSB2011], [RuJB2004] give more depth and (practical) applications of UML. Furthermore, the following books and articles are recommended for more thorough knowledge and patterns in modeling requirements: [DaTW2012], [Davi1993], [Fowl1996], [GHJV1994]. [LiSS1994] and [Pohl2010] provide a better understanding of the quality aspects of models.

4. Practices for Requirements Elaboration

In the previous chapters, we learned about the nature of requirements as the representation of the wishes and needs of people and organizations for something new (e.g., a system to be developed or adapted), about the principles that underlie the production of the requirements, and about ways to capture the requirements in documentation. We establish requirements before we build or modify a (part of a) system to ensure that the system is useful for—and accepted by—the people or the organization that requested it. These requirements then serve as input for a development team that will build and implement the system.

This is Requirements Engineering in a nutshell; it happens, explicitly or often implicitly, whenever and wherever people try to develop something. In principle, the quality of the requirements determines the quality of the output of the subsequent development. Without proper requirements, it is unlikely that the resulting system will be useful. Therefore, it is important to elaborate the requirements in a professional way. This necessitates an explicit definition of the *how to*: the practices to be used for high-quality elaboration.

That is what this chapter is about: it gives an overview of the tasks, activities, and practices that are relevant for anyone involved in Requirements Engineering. It starts with the search for potential sources of requirements and it ends with the delivery of a single, consistent, understandable, and agreed set of requirements that can serve as input for the efficient development, maintenance, and operation of an effective system.

Tasks in RE

The first task in every Requirements Engineering effort will be identifying and analyzing potential *sources for requirements*. This may seem like a simple and obvious task, but as we will see in Section 4.1, there are quite a few aspects that need to be considered and analyzed. Overlooking a source will inevitably lead to poor or even missing requirements and therefore degrade the quality of the resulting system.

The next step is eliciting the requirements from these sources. It is like drawing water from a well: you never know what is in the bucket until you have brought it to the surface. In Requirements Engineering, this task is called *elicitation*; it is explained in Section 4.2. In elicitation, we turn implicit desires, wishes, needs, demands, expectations, and whatever else into explicit requirements that can be recognized and understood by all parties involved.

However, when you ask two people about their requirements for a certain system, you will rarely get exactly the same answers. In a whole series of requirements elicited from different sources, it is almost certain that some of them will be conflicting. As it is impossible to implement conflicting requirements in one and the same system, *conflict resolution* will always be an important task in Requirements Engineering, as described in Section 4.3.

Section 4.4 is devoted to the final task in Requirements Engineering: the *validation of requirements*. The purpose of this step is to ensure that the quality of the set of requirements elicited and the individual requirements within this set is good enough to enable subsequent system development.

From the above description of Requirements Engineering tasks, you could get the impression that they are performed as a linear process with a strict sequence of steps. However, this is certainly not the intention of this description and rarely the case in practice.

Figure 4.1 shows some process steps that are common in Requirements Engineering. They might be performed in parallel, in loops, or sequentially—whatever is suitable in the given situation.

Not a linear process

The starting point is often a limited set of obvious sources. During elicitation from these sources, new sources are identified, triggering additional elicitation tasks. When conflicts are encountered, more detailed elicitation may be required to find a way out. In validation, it may appear that a source has been overlooked, a requirement is faulty, or a conflict has remained uncovered, resulting in a new series of source analysis, elicitation, and/or conflict resolution and validation activities. Even during the subsequent system development, circumstances may necessitate additional Requirements Engineering.

In agile projects, iterative and incremental Requirements Engineering and system development go hand in hand, with requirements being elaborated just before the development of a new system increment. In such projects, you will often see that a project starts with a limited product backlog of high-level requirements that are refined and detailed only when they are candidates for the next iteration.

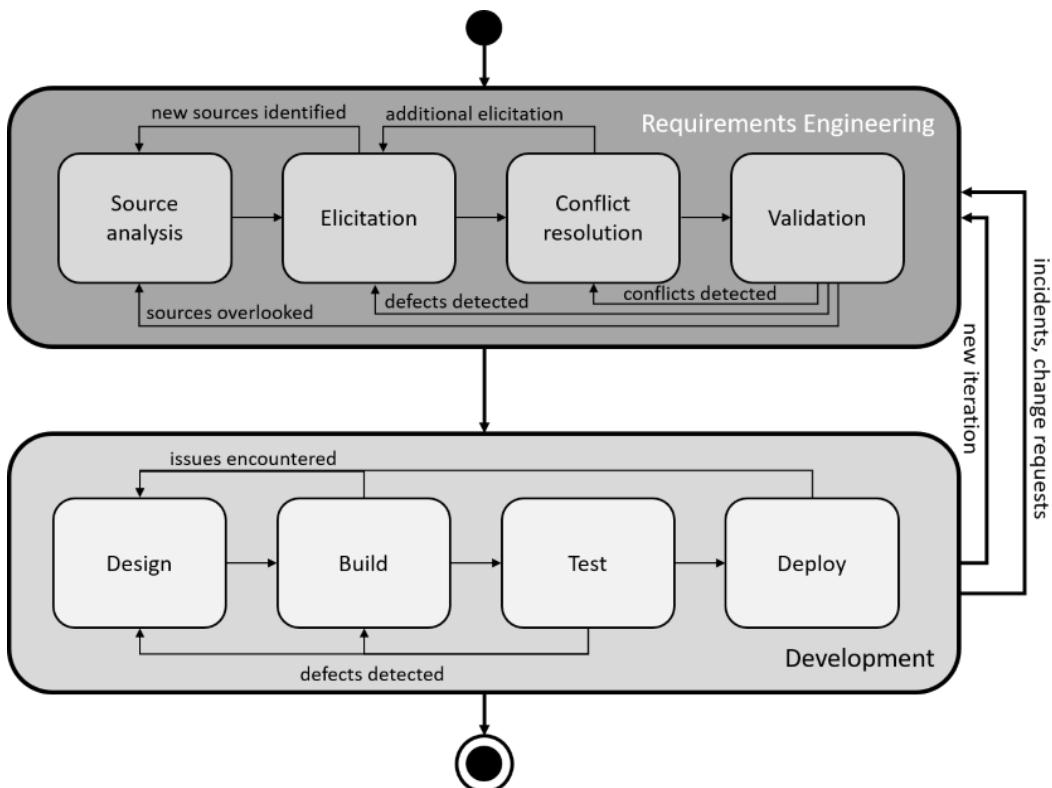


Figure 4.1 Requirements Engineering is not a linear process

4.1 Sources for Requirements

Requirements are not like candy bars, lying on the shelf for everyone to pick them as they please. In the introduction to this chapter, we compared requirements with water to be drawn from a well: it is quite an effort to bring them to the surface. Therefore, the first problem that a Requirements Engineer will face is “Where are the wells?” As no requirement comes without a source, one of the first activities in requirements elicitation is to identify the potential sources. It is not enough to identify these sources only at the beginning of a project or product development; this is a process that will be repeated over and over again.

Right from the start of requirements elaboration, the Requirements Engineer should be engaged in identifying, analyzing, and involving all relevant requirements sources, as missing a relevant source will inevitably lead to an incomplete understanding of relevant requirements. And this will continue until the end: the identification of requirements sources is a process that requires constant reconsideration.

Continuous effort

Chapter 2, Principle 3 emphasizes the necessity for (explicit and implicit) *shared understanding* between and among all parties involved: stakeholders, Requirements Engineers, developers. Understanding the context of the system to be developed in a certain application domain is a prerequisite to being able to identify the relevant requirements sources. Domain knowledge, previous successful collaboration, common culture and values, and mutual trust are enablers for shared understanding, while geographic distance, outsourcing, or large teams with high turnover are obstacles.

In Chapter 2, Principle 4, we introduced the context as a concept that is essential for understanding and specifying a system and its requirements. We defined the context as that part of reality that lies between the *system boundary* and the *context boundary*. Entities in this context will somehow influence the system or even interact with it but will not be contained in the system itself.

This would make the search for requirements sources quite simple: just look around in the context! But it is not that easy. At the start of a development process, the context has not been defined yet; the system boundary and the context boundary still have to be determined. Therefore, the search for requirements sources is an iterative, recursive process.

Recursiveness

Potential sources are analyzed for their relationship with the future system. If you find no relationship when analyzing a potential source, this means that it is part of the irrelevant environment and will not be analyzed for requirements. Potential sources that appear to be part of the future system are of no interest to the Requirements Engineer either; they *belong* to the developers. Only those entities for which analysis reveals an interaction with, an interface to, or an influence on the future system, but that remain (relatively) unchanged during the next development deserve attention as sources for requirements. In this analysis, the system boundary and the context boundary are delineated, vague at first and becoming sharper as more and more sources are identified. As the context thus becomes clearer, it becomes easier to identify new sources, which in turn sharpen the boundaries further.

The search for requirements sources usually starts with a few obvious sources, often identified by the client at the start of a development effort. Initial elicitation from these sources will uncover other potential sources, which are then analyzed to decide whether or not they are relevant for the system. During this analysis, new potential sources may again pop up. In fact, in every elicitation effort, the Requirements Engineer will remain keen on detecting new sources. This may continue until the very end of the development effort. However, we try to identify the major, most relevant sources early, because all other Requirements Engineering activities depend on this early identification.

In Requirements Engineering, we discern three major categories of sources:

- ▶ Stakeholders
- ▶ Documents
- ▶ (Other) systems

These categories are discussed in more detail in the following sections.

4.1.1 Stakeholders

In Chapter 2, Principle 2, you learned about the stakeholder as a person or organization that *influences* a system's requirements or *is impacted* by that system.

The stakeholders of a system are the main sources for requirements. Even more than with other sources, failure to include a relevant stakeholder will have a major negative impact on the quality of the final set of requirements; discovering such stakeholders late (or missing them altogether) may lead to expensive changes or, at the end, a useless system. To create a system that fulfills the needs of all of its stakeholders, the systematic identification of stakeholders should start at the beginning of any development effort and the results should be managed throughout development. Stakeholders can be found in a broad area around the system, ranging from direct and indirect users of the system, (business) managers, IT staff such as developers and operators, to opponents and competitors, governmental and regulatory institutions, and many others. The prime question for identifying a person or an organization as a stakeholder is: "Does a relevant relationship exist between the person/organization and the system?"

It helps to see stakeholders as human beings made of flesh and blood. If you identify an organization as stakeholder, ask yourself questions such as the following: "Can I identify a person who is responsible for this organization? Who can be seen as the prime contact of this organization? Who represents this organization within our company?" For instance, if the government is the stakeholder because a certain law is involved, look for someone who represents the government as the source to be approached for requirements. In this case, it is not very useful to identify the Prime Minister as this person; the head of the internal legal department would be a better choice.

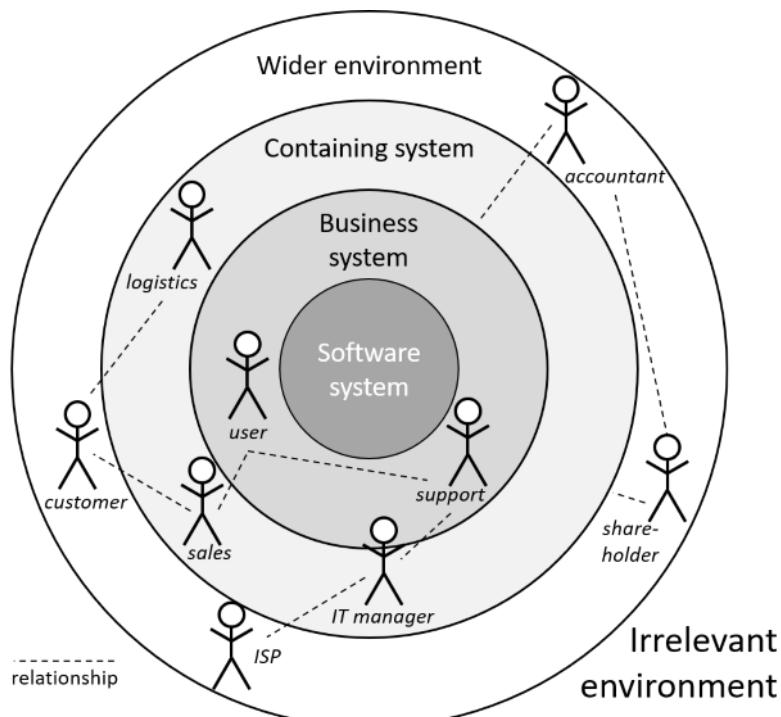


Figure 4.2 Alexander's onion model

There is no standard technique for identifying stakeholders but Ian Alexander's onion model [Alex2005] can be a good start, see Figure 4.2. This model shows how a (software) system is surrounded by several layers of higher-level socio-technical¹ and social systems, each having its own stakeholders. At the start of a requirements development effort, a few of these stakeholders will be evident—for instance, end users or customers. They can be used as a starting point in the search for other stakeholders. After identifying them as relevant sources, the Requirements Engineer will analyze their relationships, both in inner and outer surrounding systems. In this analysis, new stakeholders will be found, who in turn may have other (and more) relationships to be analyzed. You could call this the *snowball* principle: the more stakeholders you have found, the easier it becomes to find new ones. However, when arriving at stakeholders in Alexander's *wider environment*, any outer relationships will end up in the irrelevant environment, which means that they will no longer reveal new sources.

Onion model

Apart from stakeholders referring to other stakeholders, documents can often reveal new stakeholders. Good examples are organizational charts, process descriptions, marketing reports, and regulatory documents. For more information about documentation as a source for requirements, see Section 4.1.2. Checklists of typical stakeholder groups and roles can be a useful tool to avoid overlooking certain inconspicuous potential stakeholders. Also, analyzing stakeholders of legacy or similar systems can help.

As a Requirements Engineer, you will collect a lot of data about your stakeholders and maintain this data until your work is done. You must know who the stakeholders are, how you can reach them, when and where they are available, what their expertise is, as well as their relevance as a source, what their attitude towards the project is and their influence on it, what their roles are in the company and in the project, etc. Usually, this information is kept in a stakeholders list, and it must be kept up to date, as during all steps, you will remain in contact with all stakeholders—some intensely and very closely, others infrequently and superficially. See Table 4.1 below for a simplified example.

Stakeholders list

Table 4.1 Example of a stakeholders list

Name	Dept	Phone	Availability	Influence	Interest
Marlene	Owner	242263	Mondays only	++	o
Peter	Sales	481225	Permanent	++	+
Eva	Legal	481237	Not in June	+	-
Hassan	Logistics	552651	Permanent	o	++
Mira	Service desk	242424	After 4pm	-	+

Maintaining a good, open relationship with the stakeholders is key to getting relevant information from them. This relies primarily on behavioral characteristics such as integrity, honesty, and respect.

Respect

¹ A socio-technical system is a system that considers requirements spanning hardware, software, personal, and community aspects, while recognizing the interaction between society's complex infrastructures and human behavior.

Open and frequent communication about your plans, your activities, and your results is essential. You may have to turn stakeholders from initial opponents into collaborators. As a Requirements Engineer, you must understand what the stakeholders expect from you. You must also *sell* your work by showing them the benefits of your solution and by removing the impediments that stakeholders experience or expect on their way to that solution. Unfortunately, it is not uncommon that certain (mostly internal) stakeholders foresee or in fact experience negative consequences from the changes that result from your project. In such cases, it will be hard to get their cooperation, even though you will certainly need it. Escalation to higher levels in the organization may then be the only way to proceed, even though the resulting relationship will be far from optimal. Stakeholder relationship management [Bour2009] is an effective way to counter problems with stakeholders.

This implies a continuous process of gaining and maintaining the support and commitment of stakeholders by engaging the right stakeholders at the right time and understanding and managing their expectations.

In order to engage stakeholders in the elicitation process, you must ensure that they know what the project is about and what their role within the project is. You have to understand their needs and try to address these needs as far possible within your competencies in the project. While stakeholders deserve to be treated with respect, you may ask the same from the stakeholders, at least from those who are actively engaged in the project. This means that they should have time for you when you need them. They should give you the information that you ask for, as well as other information that they know to be relevant. Their feedback on your work products should be given timely and they should refrain from gossip about the project, etc.

Rights and obligations

Problems with stakeholders typically arise if the rights and obligations of the Requirements Engineer and the stakeholders with respect to the proposed system or the current project are not clear or if the respective needs are not sufficiently addressed. If problems are encountered, a kind of *stakeholder agreement* or *stakeholder contract* can help to provide all parties with the desired clarity. When this occurs within an organization, endorsement by senior management may add to the success of such an approach.

4.1.1.1 A Special Stakeholder: The User

Every system that we develop will have some interaction with certain users; why else would you develop it? Of course, when you are working on the requirements for an embedded technical subsystem, hidden inside some kind of complicated machinery, users will only interact with it indirectly through several layers of surrounding systems. In such cases, these users will not be your most important sources of requirements. However, in many systems, specific human beings will have a direct interface with the system: the users. Their acceptance of the system is vital to the success of the project, so they are your prime interest and will receive special attention during all Requirements Engineering.

There are two major categories of users:

- *Internal users* are directly related to the organization for which the system is being developed, such as staff, management, subcontractors. They are mostly limited in number, more or less known individually, and somehow involved in the project. It is relatively easy to contact them and they can be reached, influenced, and motivated through formal, existing channels.

- ▶ *External users* are outside the company, such as customers, business partners, civilians. Their number may be (very) large, in many cases they are not known individually, and they could be completely unaware of or indifferent to the project. You cannot influence them through formal channels. To get in contact with them, you may need to do special things to motivate them to participate, such as advertising, promising some reward, or giving them free access to a beta version. Forming a user panel or addressing the crowd (sometimes with payment) can be useful ways of involving these users.

Be aware that in addition to these regular categories, it can also be relevant to pay attention to *misusers*: people who intentionally try to interact with the system in a harmful way, such as hackers or competitors. It is rarely possible to contact or to influence them, but you can try to develop measures to discourage them, to keep them out, or to detect foreseeable attempts of misuse.

This categorization should not be considered very strictly. We can imagine projects in which users outside the company are small in number and can be reached easily, so they can be seen as *internal*. On the other hand, in big companies, the distance to the users can be so large that they should be treated more or less as *external*.

If you have a good insight into your user base, you should make a distinction between user roles. Separate roles will usually have different information needs, will use the system in their own way, and may have distinct access rights to functions and data—for instance, users who input data versus supervisors who check this input. In such cases, make sure that you include representatives of all relevant roles in the elicitation.

User roles

Often, especially at the beginning of elicitation efforts, such insight will be missing. Then, it is even more important to realize that there is no such thing as *The User*. *The User* is not an amorphous mass of identical humans but rather a collection of individuals, each of them with their own habits, wishes, and needs. When a system has thousands of users or more, of course you will not be able to fine tune the requirements to their individual needs. On the other hand, a *one size fits all* approach aiming for the *average user* might not work either.

In such cases, it helps to discern a number of user types or user groups that show certain, often behavioral similarities within the group as distinct from other groups. In practice, having some three to seven groups often works best. Then, as a Requirements Engineer, you will treat every group as a distinct source for requirements. A good technique is the use of *personas* [Hump2017]. Personas are fictitious individuals that describe typical user groups of the system with similar needs, goals, behaviors, or attitudes.

User groups

4.1.1.2 Personas

There are two major approaches to creating personas:

► Data-driven

In this approach, personas are developed with marketing techniques, such as surveys, focus groups, and other ethnographic data collection techniques. Usually, they are called *quantitative personas*.

► Imagination

As a cheaper and quicker alternative, personas may be developed by imagination—for instance, in a brainstorming session with the project team. We call them *ad hoc personas*. As a Requirements Engineer, you must be aware that ad hoc personas are based on imagination and assumptions. These assumptions must be checked and confirmed throughout the Requirements Engineering process.

Basically, persona descriptions contain information that is relevant in view of the development of the system at hand. Usually, this information will be *enriched* with additional data, such as name, address, hobbies, and a drawing or portrait picture.

Personas

Such personas are called *qualitative personas*. They give a human face to the abstract concept of persona. This may help you to understand that your work as a Requirements Engineer not only relates to facts but also to emotions. Figure 4.3 gives an example of such a qualitative persona description.

If you use personas in your project, it may be useful to look for a few individuals that fit the persona descriptions and treat them as representatives of each group. In that case, you have real stakeholders with whom you can communicate. However, always remember that the group that such a stakeholder represents is an artificial one that does not exist in the real world but only in the context of the system or project.

4.1.2 Documents

Documents can be a major source for requirements too. As a Requirements Engineer, you often have to do a lot of reading, especially at the beginning of a project. All kinds of documents may be relevant: company-, domain- and project-related documents, product and process descriptions, legal and regulatory documentation, etc. As with stakeholders, you can make a distinction between *internal* and *external* documents. Internal documents can be easy to get but may be confidential and cannot be shared without explicit consent.

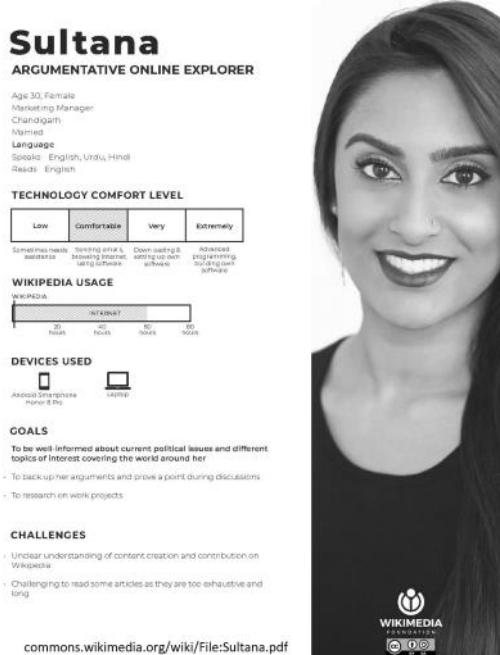


Figure 4.3 Persona example

Often, you will need to sign a non-disclosure agreement before you receive access to them. External documents may be difficult to find but are usually public; if not, make sure that you are allowed to access and use them.

Documents can be a great way to find other sources. For instance, an internal process description may mention certain roles as being involved in that process, which in turn can lead you to new potential stakeholders. However, documents can also be direct sources for requirements, especially those that are easily overlooked or not regularly mentioned by stakeholders: constraints in standards, company guidelines, and other legal or regulatory documents; detailed requirements in procedures and work instructions; bright new ideas in marketing material from competitors. Studying documentation can help you to understand the context of the system to be developed, even by reading emails between people who took the initiative for the project. Reading about analogous solutions for problems and goals in other companies and domains can spark your imagination and show a feasible direction for your current project.

As a Requirements Engineer, you should be aware that a document is always related to some people: the author(s), the (target) audience, a manager responsible for or an auditor checking adherence to it, someone who pointed out its existence to you, etc. All those people may have a role as a stakeholder; it is up to you to find out whether or not this is the case. You should always check the validity and relevance of a document and you often need stakeholders to confirm this to you. If you were to derive requirements from an invalid or outdated document, the system developed from these requirements would probably fail.

Relationship with stakeholders

Just like stakeholders, documents used as requirements sources must be managed. You can use a document list, comparable to the stakeholder list discussed above. All documents should be kept in some kind of common, indexed library with a unique identification to allow them to be referenced. Dates and version numbers are important to guard against working with outdated versions; you could check at regular intervals whether a newer version has been published and whether this influences the requirements. You should preferably work with final versions but in practice, you often have to deal with drafts, so you also have to record the status of documents. Keep old versions in an archive, because they may be important to understanding how a system and its requirements evolved. Setting up suitable and accurate management of the documents involved right from the start of a project will save you a lot of work later on, in Requirements Engineering, development, and deployment. It is a good starting point for establishing backward traceability, as discussed in Section 6.6.

4.1.3 Other Systems

You can also consider other systems as sources for requirements of the system you are interested in. Here, you can make a distinction between *internal* and *external* systems, just as in documentation and with the same considerations about access and confidentiality. Another distinction is that of *similar* systems versus *interfacing* systems.

Similar systems have certain functionalities in common with the system to be developed. They may be predecessor or legacy systems, competitor systems, comparable systems used in other organizations, etc. You often study them through their documentation but sometimes you can observe them in action or try them out as if they were a kind of prototype. You may be able to contact their users to learn more about the functionalities and solutions of such systems. Predecessor or legacy systems are often a good source for identifying detailed (functional and quality) requirements and constraints.

Similar systems

However, be aware that (especially technical) constraints from the past may not be relevant anymore and may no longer restrict your current solution space.

Sometimes, a new system and a legacy system will coexist during a certain period, leading to additional requirements—for instance, with regard to data sharing. Competitor and comparable systems may be studied for their solution characteristics and can be a good source for identifying *delighters* (see Section 4.2.1).

Interfacing systems will have a direct relationship with the system for which you are developing the requirements. They will exchange data with your system as a source and/or a sink though some (synchronous or asynchronous, in real time or in batch) interface (see also Section 3.4.2 on system interfaces in context modeling). The correct configuration, content, and behavior of such an interface is often essential for ensuring that your system works, and you will therefore have to understand the system in detail. You can study interfacing systems through their documentation, but as every (technical) detail is important here, simulation or testing may be necessary. With regard to older or legacy systems in particular, you cannot trust their documentation to be up to date so you will need some proof. To understand an interface, you will also have to understand the context, functionality, and behavior of the interfacing system. It will be helpful if you can contact application managers, architects, or designers of such systems, especially if the interfacing system itself has to be updated to allow for the new interface. Also be aware that an interfacing system will itself have users; it may be interesting to consider these users as stakeholders in Alexander's *wider environment* of your own system.

Interfacing systems

4.2 Elicitation of Requirements

If we continue the analogy of water being drawn from a well, we are now at the point that we have found the well and we start pulling the rope to get the bucket full of the required water (or in this case requirements) to the surface. That is what we call *elicitation*: the effort expended by the Requirements Engineer to turn implicit desires, demands, wishes, needs, expectations—which until now were hidden in their sources—into explicit, understandable, recognizable, and verifiable requirements. Of course, we will have to use all wells to be complete and pull the rope in the right way to make sure that we get all the water to the surface. In Requirements Engineering terminology, we say that we should apply the right *elicitation techniques*.

A common categorization of elicitation techniques is the distinction between:

- ▶ Gathering techniques
- ▶ Design and idea-generating techniques

From these categories, you can select a wide range of elicitation techniques, each with their own characteristics. Figure 4.4 gives an overview of elicitation techniques in their categories and subcategories.

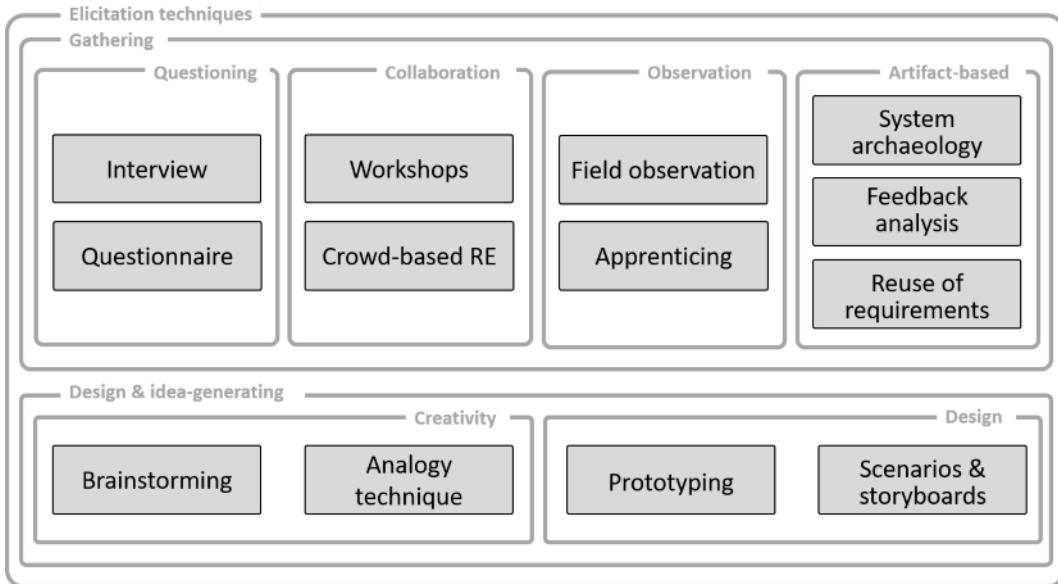


Figure 4.4 An overview of elicitation techniques

A critical key competence of the Requirements Engineer is the ability to choose the right (mix of) techniques under the given circumstances. Picking the right ones may depend on many factors, such as:

▶ Type of system

A completely new innovative system will benefit more from creativity techniques, while a replacement system in a safety-critical environment may need questioning techniques and system archaeology.

Selecting the right technique

▶ Software development life cycle model

In a waterfall project, you may have planned for extensive techniques such as apprenticeship or analogies, while in an agile environment, brainstorming, storyboarding, and prototyping may prevail.

▶ People involved

For instance, field observation will probably not be appreciated in highly confidential businesses; a comprehensive survey may be preferred over a high number of individual interviews.

▶ Organizational setup

A solid government organization needs a totally different approach to a young startup; a dispersed, highly decentralized company needs a different approach to a compact company with a single location.

The best results are usually achieved with a combination of different elicitation techniques. For a systematic approach to selecting them, see [CaDJ2014].

Elicitation techniques are—or at least, should be—able to detect all kinds of requirements. In Requirements Engineering practice, however, explicit *functional requirements* are often overrated, and the more implicit *quality requirements* and *constraints* get less attention.

Quality requirements and constraints

This may result in a system that—with all functional requirements being fulfilled—does not perform, has poor usability, does not comply with architectural guidelines, or fails to meet certain other quality requirements or constraints, and consequently will not be accepted.

Stakeholders can be sources, but you will often find more information in documents. For the elicitation of *quality requirements*, applying a checklist based on the ISO 25010 standard [ISO25010] can help to detect and quantify them—for example, in preparing for an interview. *Constraints* can be found by considering possible restrictions of the solution space—for example, technical, architectural, legal, organizational, cultural, or environmental issues. Relevant documentation can often be identified through staff members.

4.2.1 The Kano Model

One of the major circumstances to consider in selecting an elicitation technique is the nature and the importance of a requirement that we are trying to uncover. To gain more insight into the nature of certain requirements, the Kano model [Verd2014] comes in handy. This model, shown in Figure 4.5, classifies features of a system into three categories:

Kano model

- **Delighters** (synonyms: excitement factors, unconscious requirements)
A delighter is a feature that customers are not aware of; that is why we call them *unconscious*. The customers do not ask for the feature because they do not know that it is possible in the system—for instance, a smartphone that can be turned into a beamer. At first, when the feature is new on the market, most customers will have their doubts about it, but when some early adopters have tried it out and start spreading the word, more and more people want to have it. If a delighter is absent, no one will complain; but when present, this can be a differentiating feature that attracts lots of customers.
- **Satisfiers** (synonyms: performance factors, conscious requirements)
A satisfier is something that the customers explicitly ask for (hence *conscious* requirements). The more satisfiers you can put into your system, the higher the satisfaction of the customers will be. An example could be the number of lenses and video options in a modern smartphone. Because adding satisfying features usually also means higher costs, you will often need a kind of cost/benefit analysis to decide how many of them will be incorporated in the system.
- **Dissatisfiers** (synonyms: basic factors, subconscious requirements)
A dissatisfier is also a feature that the customers do not ask for. Here, however, the reason for not asking for it is that the feature is so obvious (*subconscious*) that the customers cannot imagine it not being part of the system; these features are tacitly considered as must-haves. Imagine a smartphone without GPS. If a dissatisfier is included as a feature of a system, customers will not notice it because they think the system cannot exist without it. However, if you overlook such a requirement and leave it out of the system, customers will be very upset and will refuse to use the system.

The Kano model looks at requirements from the perspective of the customer. It focuses on differentiating features, as opposed to expressed needs. With the Kano model in mind, you may find more requirements than when focusing only on the explicitly formulated needs from the stakeholders. As we will see later in this chapter, all categories can be linked to distinct elicitation techniques.

Perspective of the customer

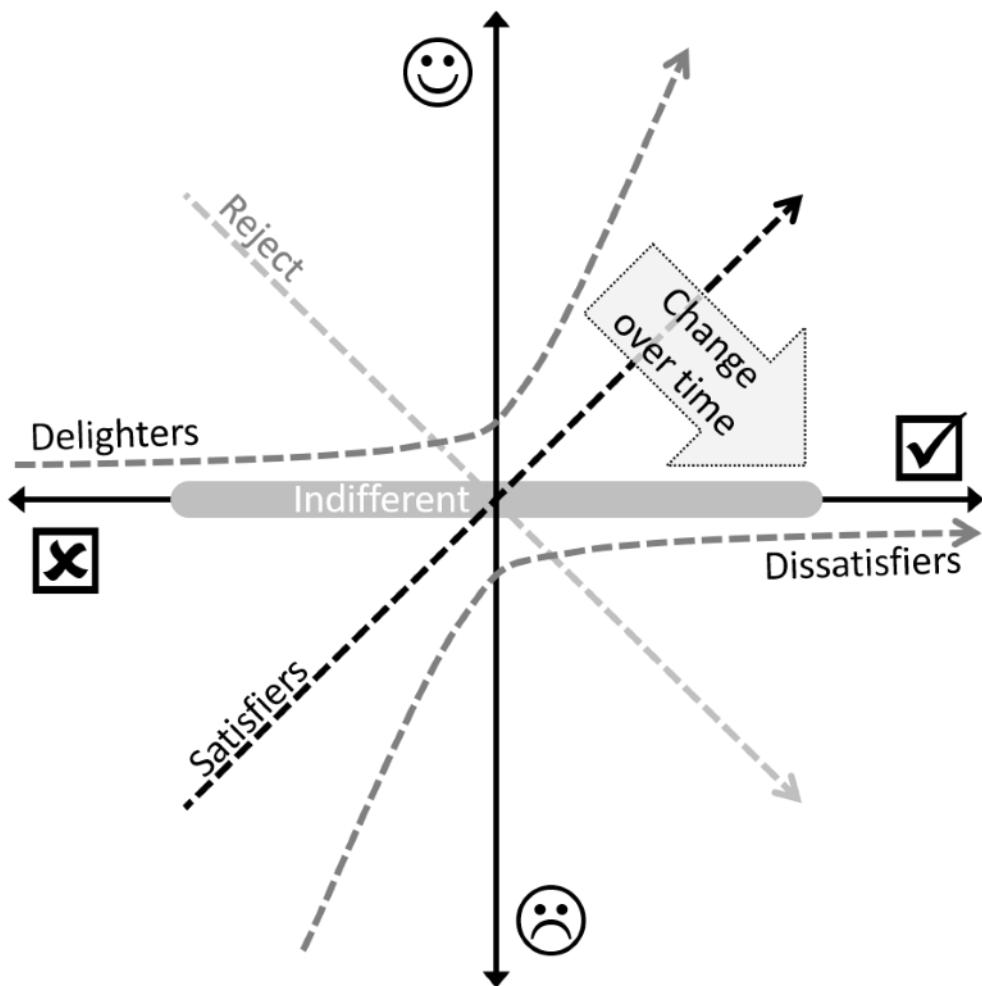


Figure 4.5 The Kano model

In fact, the original Kano model contains two more categories, the *indifferent* (or *I don't care*) and the *reject* (or *I hate*) requirements. These categories do not get much attention in most Requirements Engineering handbooks but can still be useful for you as a Requirements Engineer. Suppose, for instance, that developers want to add a certain feature to the system for technical reasons. If, after analysis, you find that the customers are indifferent to this feature, it is safe to include it in the system. However, if it turns out to be a reject requirement, you should tell the developers to look for a less harmful alternative, as implementing this requirement can turn out to be a costly mistake.

One interesting observation when working with the Kano model is that requirements tend to change over time. If someone introduces a new feature, there is no certainty about how the market will react to that feature. Sometimes, customers will be indifferent to it, and the feature will survive only if it does not increase the price of the product.

If customers reject it, the feature will probably be removed from the product as soon as possible. However, when (maybe initially a vanguard of) the customers like the feature, it will become a delighter, a unique selling point for which the customers are prepared to pay the price. As more and more customers discover, experience, and like this new feature, it will become a satisfier that is explicitly asked for. Gradually, when similar systems start to implement the same feature, customers may forget that systems did not originally include such a feature and will take it for granted, turning it into a dissatisfier. That is why many systems contain features that the users consider as indispensable without knowing why and thus without explicitly asking for them.

A good example is the camera function on cell phones, for which this process took less than 20 years. The first time a camera was introduced as part of a cell phone, most customers were puzzled: no one had asked for this feature and most customers thought "If I want to take a picture, I need a camera." However, some early adopters tried it out and discovered the convenience of taking pictures without a dedicated camera and being able to share them instantly with other people without making a print. They liked the camera feature as a delighter and all brands started to implement it in their phones, turning it into a satisfier: the better the pictures were, the more satisfied the user was. Nowadays, when buying a new cell phone, everybody takes for granted that it will have a camera function so it has become a dissatisfier: "If I can't take a picture with this cell phone, it is useless."

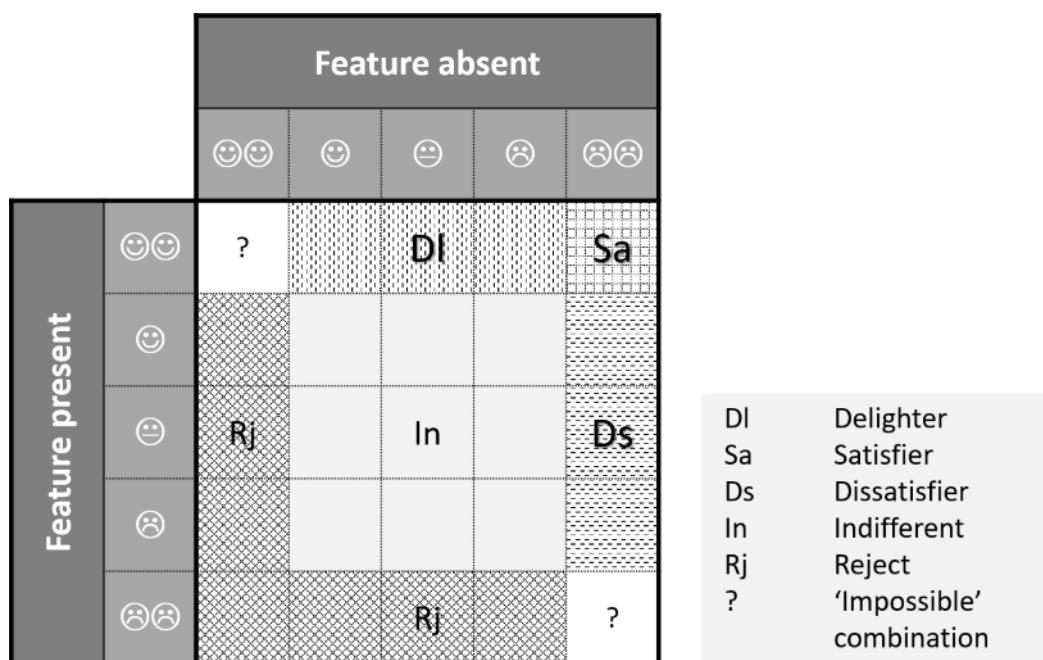


Figure 4.6 Kano analysis matrix

How can you categorize a specific feature? You use the technique of Kano analysis. For a specific feature, you ask two questions to a representative group of potential users: (1) "What would you feel if this feature were present in the system?" and (2) "What would you feel if this feature were absent from the system?" You let them score the answers on a 5-point scale between "I love it" and "I hate it" and then plot the average answer on the Kano analysis matrix as shown in Figure 4.6. The cell that comes up gives you the Kano classification for the feature.

[Kano analysis matrix](#)

The next question is: why bother with Kano analysis in requirements elicitation?

As we explain in the following sections, you will need different techniques to find these different categories of features. By themselves, stakeholders will mainly talk about their satisfiers—their conscious requirements that they explicitly ask for. It is much more difficult to detect the other categories but fortunately, there are several useful techniques for doing so.

4.2.2 Gathering Techniques

With *gathering techniques*, you examine the different sources that you have identified and elicit the requirements from there. These established techniques have been commonly used throughout Requirements Engineering and predominantly yield satisfiers and dissatisfiers.

Gathering techniques can be further subdivided into four categories:

- ▶ Questioning techniques
- ▶ Collaboration techniques
- ▶ Observation techniques
- ▶ Artifact-based techniques

Questioning techniques are always used in an interaction with stakeholders. The Requirements Engineer poses appropriate questions to the stakeholders in order to let the stakeholder do the thinking and to receive answers from which requirements can be derived. Examples of questioning techniques are:

- ▶ *Interview*: Due to their flexibility, *interviews* are probably one of the most frequently used elicitation techniques. They do not require specific tools and can be used to elicit high-level requirements as well as very specific ones. Usually, an interview is a one-to-one session between a Requirements Engineer (interviewer) and an individual stakeholder (interviewee), but a small group of interviewees is also an option. Typically, requirements elicited with an interview are satisfiers, as the interviewee voices conscious information. The interview technique is not overly complicated and most people have a good understanding of what it is. However, you need clear goals and good preparation to obtain useful results. Interviews can reveal detailed information and offer flexibility based on the answers given. They are rather time-consuming, so this technique is less appropriate when you want to reach large numbers of stakeholders.
- ▶ *Questionnaire*: With a *questionnaire*, a larger group of stakeholders is asked to answer—orally, in writing, or on a web page—the same set of questions, which are presented in a structured way. Quantitative questionnaires are used to confirm hypotheses or previously elicited requirements. They use closed-ended questions (only predefined answers allowed) and can therefore be evaluated quickly and deliver statistical information. On the other hand, qualitative questionnaires use open-ended questions and can find new requirements. They tend to deliver complex results and are thus usually more time-consuming to prepare and to evaluate. In general, questionnaires are a preferred technique for large groups. Be aware, however, that designing a good questionnaire involves quite a lot of effort. A questionnaire is often the next step after obtaining a preliminary idea based on a series of interviews in order to validate these ideas within a larger group.

Questioning techniques

In the category of *collaboration techniques*, we find all kinds of collaboration between the Requirements Engineer and other people (stakeholders, experts, users, customers, etc.). Some examples are:

Collaboration techniques

➤ Workshops

Workshop is an umbrella term for group-oriented techniques, ranging from small informal meetings to organized events with several dozen or even hundreds of stakeholders. A nice definition is as follows: "A requirements workshop is a structured meeting in which a carefully selected group of stakeholders and content experts work together to define, create, refine, and reach a closure on deliverables (such as models and documents) that represent user requirements" [Gott2002]. With a workshop, you can get a good global insight in a short time because you use the interaction between the participants. If you need more detail, follow-up interviews or questionnaires are appropriate. Workshops can serve as a gathering technique but they can also be used in creativity techniques (see Section 4.2.3).

➤ Crowd-based Requirements Engineering

In *crowd-based* (also known as platform-based) Requirements Engineering (see [GreA2017]), elicitation is turned into a participatory effort with a crowd of stakeholders, in particular the users, leading to more accurate requirements and ultimately better software. The power of the crowd lies in the diversity of talents and expertise available within the crowd. As the amount of data obtained from the crowd will be large, an automated platform for processing this data is essential. This platform should offer community-oriented features that support collaboration and knowledge sharing and foster the engagement of larger groups of stakeholders in the collection, analysis, and development of software requirements, as well as validation and prioritization of these requirements in a dynamic, user-driven way.

Observation techniques are also applied in relation to stakeholders. The stakeholders are observed while they are engaged in their normal (business) processes in their usual context without direct interference from the Requirements Engineer. Observation techniques are particularly useful for identifying dissatisfiers. You may observe peculiar activities, sequences, data, etc. that are so common to the stakeholders that they do not mention them, and these aspects thus do not easily come to light in gathering techniques.

Observation techniques

Common forms of observation techniques are:

➤ Field observation

During *field observation*, the Requirements Engineer watches (mostly) end users in their environment while these users perform the activities for which a system is to be developed. Field observation is typically used in situations where interaction would distract the users or would interfere with the process itself and potentially falsify results. It can even be applied without informing the subjects observed, e.g. by sitting with other patients in a dentist's waiting room to observe their behavior. With field observation, you will be able to detect (often detailed) requirements that would not easily be found with other techniques—for instance, because actions and behaviors are too complicated to put into words.

Be aware that field observation requires a lot of preparation, a sharp eye, and lots of time. Video is quite helpful for capturing stakeholder behavior. It can be used in conjunction with direct field observation and may even replace it in situations where the actual presence of the Requirements Engineer is not allowed or desired. Video offers the possibility of postprocessing to allow for detailed investigation of acts and proceedings that are difficult to observe.

➤ Apprenticing

Apprenticing differs from field observation in that it is participatory. In apprenticeship, the Requirements Engineer (*apprentice*) does a kind of internship in the environment in which the system at hand will be used (or is already in use) and experienced users (*masters*) teach the apprentice how things work. The apprentice participates but does not interfere; they act like a novice in the field and are allowed to make mistakes and ask “dumb” questions. The aim is to create a deep understanding of the domain, the business, and the processes before the actual elicitation of the requirements starts. A follow-up with interviews and questionnaires will often be required to verify the initial ideas. The resulting requirements can subsequently be documented and validated. An optimal duration for such an internship depends on many different factors (e.g., complexity of the process, repetitiveness, time availability of master and apprentice) but usually varies between one day and several weeks. Be aware that apprenticeship may be difficult or impossible to organize in certain domains, such as medicine, aviation, or the military.

Artifact-based techniques do not use stakeholders (directly) but rather work products such as documents and systems, or even images, audio and video files, as sources for requirements. These techniques can find (sometimes very detailed) satisfiers and dissatisfiers. It is usually a time-consuming task to examine (often poorly structured, outdated, or partly irrelevant) work products in detail. Nonetheless, artifact-based techniques can be useful, particularly when stakeholders are not readily available.

A few examples of artifact-based techniques are:

➤ System archaeology

In *system archaeology*, requirements are extracted from existing systems—such as legacy systems, competitor systems, or even analogous systems—by analyzing their documentation (designs, manuals) or implementation (code, comments, scripts, user stories, test cases). This technique is mainly used if an existing system has been used for many years and is now to be replaced by a new system for whatever reason; the new system has to cover the same functionality as the old one, or at least certain parts of it. System archaeology often takes a lot of time but may reveal detailed requirements and constraints that are not easily detected otherwise. However, you will need extra time to check, through other channels, whether or not these requirements are still valid and relevant.

Artifact-based techniques

Examples for artifact-based techniques

► Feedback analysis

There are many ways to collect *feedback* from (potential) users and customers, be it on an existing system or on a prototype. Feedback data may be structured (e.g., a 5-star rating in an app store) or unstructured (like review comments). It may be gathered via web surveys and contact forms, during beta or A/B testing, on social media, or even as customer remarks received in a call center. Often, the amount of data is quite large, and analysis will be time-consuming. However, the feedback can be very useful for gaining insight into the user's *pains and gains*. Negative scores and critical remarks will help you to detect unnoticed dissatisfiers. Positive scores and compliments will give you additional information about satisfiers. Occasionally, comments may even contain innovative ideas that can be turned into delighters. Feedback analysis can thus result in adjustment of existing requirements but also to the discovery of new ones.

► Reuse of requirements

Many organizations already have a large collection of requirements that have been elicited and elaborated in the past for previous systems. Many of these requirements may be applicable for a new system too, especially requirements that have been derived from an overarching domain model. Therefore, *reuse* of existing requirements can save lots of time and money because you can skip their elicitation. However, this works only if this collection of existing requirements is up to date, managed effectively, easily available, and documented extensively, which unfortunately is not often the case. Even if reuse is feasible, be aware that you still need to validate with the stakeholders whether these reusable requirements are relevant and valid in the new situation, be it directly or with some adjustments.

4.2.3 Design and Idea-Generating Techniques

In the past, Requirements Engineering has focused on gathering and documenting the necessary requirements from all relevant stakeholders by applying gathering techniques as introduced in the previous section. The growing influence of software as an innovation driver in many businesses is now increasingly demanding a new positioning of Requirements Engineering as a creative, problem-solving activity. This involves the application of other techniques that no longer consider stakeholders (and their documents and systems) the one and only source of requirements. Innovative systems need new, maybe disruptive features that the current stakeholders cannot imagine (yet).

Design and idea-generating techniques

Design and idea-generating techniques have emerged to fulfill this need. These techniques promote creativity, mostly within teams, for the generation of ideas and may provide additional ways to elaborate a given idea. These techniques can find new requirements that are often delighters. Many diverse techniques exist within this broad category, some remarkably simple, others quite elaborate. We will look at a few examples from two subcategories:

- Creativity techniques
- Design techniques

In addition, we will look at the emerging field of *design thinking*.

Creativity techniques stimulate creativity in order to find or to create new requirements that cannot be gathered directly from the stakeholders because the stakeholders are not aware of the feasibility of certain new features or (technical) innovations. These techniques are usually applied within diverse, multi-disciplinary teams of IT staff such as analysts, Requirements Engineers, developers, testers, product owners, application managers, etc., with or without business representatives, users, clients, and other stakeholders. The techniques stimulate out-of-the-box and borderless thinking and elaboration of each other's ideas. Unfortunately, none of them guarantee success in generating creative results as several mechanisms in our brain have to come together to enable creative ideas.

Creativity techniques

An obvious example where creativity techniques are important is the games industry. You can of course ask gamers for their requirements with a gathering technique and you will learn what gamers like or dislike about the current games. However, to develop a successful game, you need to surprise the gamers with something new; you have to discover their delights. That is exactly where creativity techniques fit in.

Several preconditions have been identified as important factors for creativity to emerge:

Preconditions

- ▶ Chance—and therefore time—for an idea to come up
- ▶ Knowledge of the subject matter, which raises the odds for an idea that makes the difference
- ▶ Motivation, as our brain can only be creative if there is a direct benefit for its owner
- ▶ Safety and security, as useless ideas must not have negative consequences

Two examples of creativity techniques are presented here:

Examples of creativity techniques

- ▶ Brainstorming

Brainstorming (see [Osbo1948]) supports the development of new ideas for a given question or problem. As with most creativity techniques, the crucial point of brainstorming is to defer judgment by separating the finding of ideas from the analysis of ideas. Some general guidelines for brainstorming include:

- Quantity prevails over quality.
- Free association and visionary thinking are explicitly desired.
- Taking on and combining expressed ideas is allowed and desired.
- Criticizing other participants' ideas is forbidden even if an idea seems to be absurd.

After a brainstorming session, the ideas that have emerged are categorized, assessed, and prioritized. Selected ideas then serve as input for further elicitation.

➤ Analogy technique

The *analogy technique* (see [Robe2001]) helps with the development of ideas for critical and complex topics. It uses analogies to support thinking and the generation of ideas. Its success or failure is influenced mainly by the selection of a proper analogy for the given problem. The selected analogy can be close to (e.g., the same problem in another business) or distant from (e.g., comparing an organization with a living organism) the original problem. The application of the analogy technique consists of two steps:

- Elaborate the aspects of the selected analogy in detail without referring to the original problem.
- Transfer all identified aspects of the analogy back to the original problem.

The resulting concepts and ideas will then be a starting point for additional elicitation.

Design techniques can be seen as a special category of creativity techniques that provide additional, explorative, or combinatorial techniques to elaborate ideas and gain further insights for a given idea. Many of these techniques start from market research or bottleneck analysis and rely heavily on visualization, team cooperation, and customer feedback.

Design techniques

Popular techniques in this category include:

➤ Prototyping

By *prototype* (in relation to elicitation; see also Section 3.7 for more information), we mean a kind of intermediate work product that is created or released to generate feedback. Prototypes can range from simple paper sketches to working pre-release versions of a system. They allow future users to experiment with the system in a more or less tangible way and to investigate certain, as yet unclear, characteristics during Requirements Engineering and before the actual implementation. As we will see in the section on validation (4.4.2), prototypes are primarily used for checking that previously defined requirements have been implemented correctly. However, with proper guidance of the users and analysis of their feedback, this technique can also be used to derive new requirements. It may be particularly useful for detecting non-functional requirements, dissatisfiers and constraints, or whatever other characteristics that cannot easily be understood or defined up front in models and documentation.

➤ Scenarios and storyboards

The word *scenario* stems from the theater, where it is used to refer to an outline of a play, opera, or similar, indicating a sequence of scenes with their characters. In IT, we use this term to describe a flow of actions for a system, including the users involved (who we usually call actors here). Through scenarios, you can explore alternative ways of realizing a process in a system. Because of their lightweight structure, they are easy to develop and can be changed rapidly. In the same way as for prototypes, scenarios and storyboards can be applied in both (early) elicitation and (later) validation of requirements.

Scenarios can be documented in a written or a visual form. The visual form of a scenario is called a *Storyboard*.

A storyboard is typically a kind of comic strip with a series of panels that show the interaction of certain personas with the system. See Figure 4.7 for an example. Scenarios and storyboards are useful for early elaboration of ideas in terms of processes and activities.

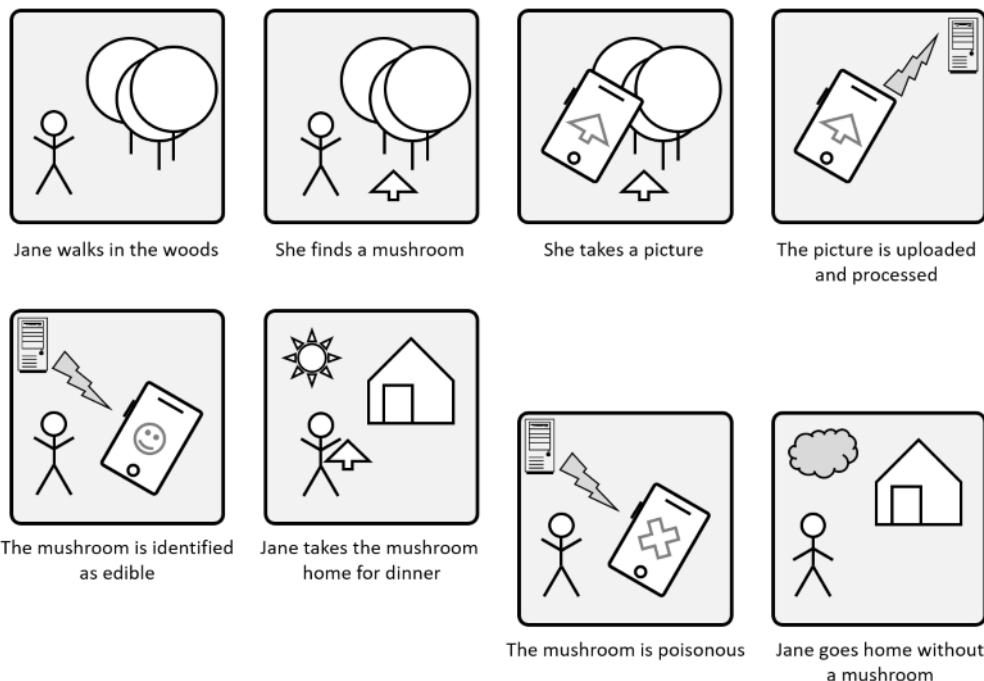


Figure 4.7 Example of a storyboard

Design thinking is not so much a technique but rather a concept, an attitude, a philosophy, a family of processes, and often a toolbox full of techniques. The focus is on innovation and problem solving. Several variants of design thinking exist, mostly using lightweight, visual, and agile techniques. Two basic principles can be found in all variants:

Design thinking

➤ Empathy

The first step for design thinkers is to find the real problem behind the given problem. They try to understand what stakeholders really think, feel, and do when they interact with a system. Therefore, we often refer to design thinking as human-centered design. Personas, empathy mapping, and customer co-creation are common techniques to this end.

➤ Creativity

A common characteristic of design thinking is the *diamond*: the alternation of divergent and convergent thinking. Divergent thinking aims at exploring an issue more widely and deeply, generating lots of different ideas, and convergent thinking focuses, selects, prunes, combines these ideas into a single final delivery. A basic pattern, the *double diamond* model, is shown in Figure 4.8 (see [DeCo2007]).

A detailed treatment of design thinking is beyond the scope of this Foundation Level Handbook.

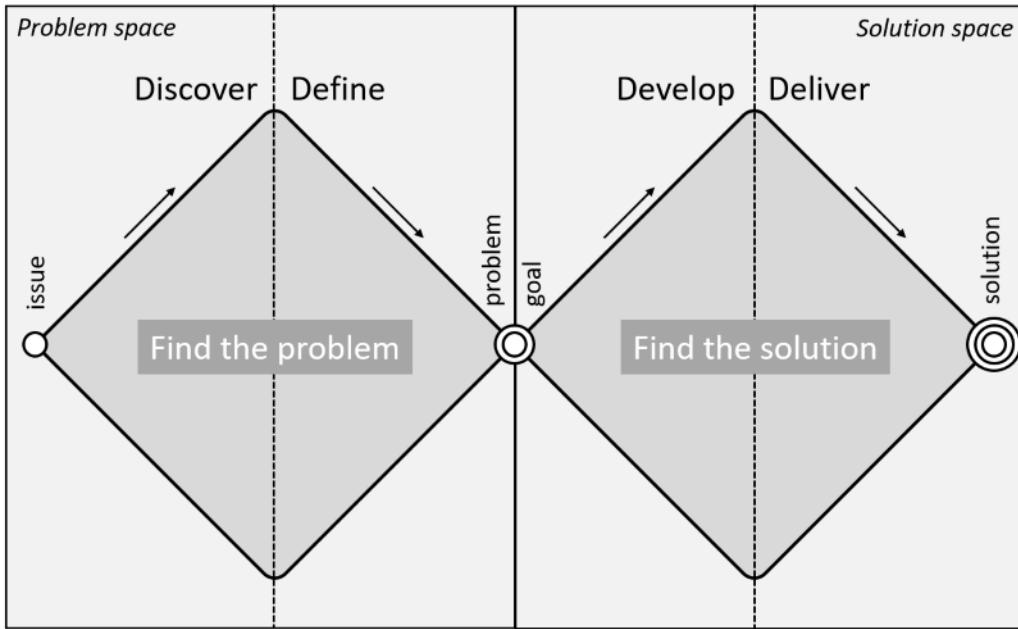


Figure 4.8 The double diamond

4.3 Resolving Conflicts regarding Requirements

During elicitation, you gather a broad collection of requirements from different sources, with different techniques, and at different levels of abstraction and detail. The elicitation techniques that you use do not guarantee by themselves that this collection as a whole forms a single, consistent, agreed upon set of requirements that captures the essence of the system. Both during and after elicitation of a set of requirements for a certain system, you may find out that some of the requirements are conflicting: they may be inconsistent, incompatible, contradictory. It might be that requirements conflict with each other (e.g., “all text must be black on white” versus “all error messages must be red”) or that some stakeholders have a different opinion about the same requirement (e.g., “all error messages must be red” versus “user error messages must be red, all other error messages blue”). As we cannot develop a (specific part of a) system based on conflicting requirements, the conflicts must be resolved before development can start. As a Requirements Engineer, you are the one who should make sure that all stakeholders arrive at a shared understanding (see Chapter 2, Principle 3) of the complete set of requirements as far as they are relevant to them and that they agree on this set.

Resolving Conflicts

But what is a conflict? A conflict is a certain disagreement between people: “An interaction between agents (individuals, groups, organizations, etc.), where at least one agent perceives incompatibilities between her thinking/ideas/perceptions and/or feelings and/or will and that of the other agent (or agents), and feels restricted by the other’s action” [Glas1999]. In a requirements conflict, two or more stakeholders have a different or even contradictory opinion regarding a certain requirement or their requirements cannot be implemented in a certain system at the same time; see Figure 4.9.

Conflict

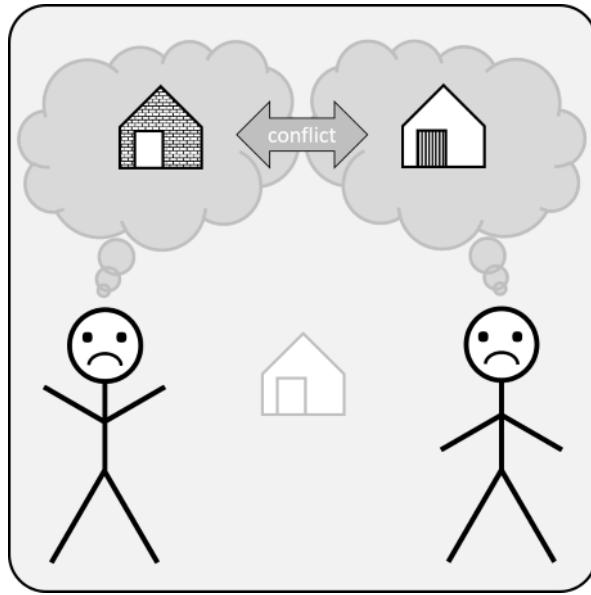


Figure 4.9 A requirements conflict

Dealing with requirements conflicts can be difficult, painful, and time-consuming, especially when personal issues are involved. However, denying or ignoring conflicts is not an option, so the Requirements Engineer must actively search for ways to resolve them. At the end, all stakeholders must understand and agree upon all requirements that are relevant to them. If some stakeholders do not agree, this situation must be recognized as a conflict that must be resolved accordingly.

4.3.1 How Do You Resolve a Requirements Conflict?

To resolve a requirements conflict properly, the following steps should be followed:

➤ Conflict identification

We often have conflicts in our everyday life. They give us an unpleasant feeling, so a common strategy is simply to avoid, ignore, or deny them. That may make conflicts hard to find. Most of them tend to be hidden and can only be detected by careful observation. There are many indicators that you can pay attention to, both in communication and in documentation:

- In communication, you may observe behavior such as denial, indifference, pedantry, continuously asking for more details, deliberately incorrect interpretations, concealment, or delegation.
- In documentation, you may find things such as contradictory statements by stakeholders, conflicting results from analysis of documents or systems, inconsistencies across different levels of detail, and inconsistent use of terms.

If you observe such indicators, this does not necessarily mean that there is a requirements conflict, but you should certainly be suspicious. Thorough discussion with the stakeholders can then bring a hidden conflict to the surface.

Steps to resolve conflicts

► Conflict analysis

Once a conflict has been identified, the Requirements Engineer has to first clarify whether this conflict is a requirements conflict or not. After all, a requirements conflict is the primary responsibility of the Requirements Engineer; other conflicts can be resolved by other participants, such as a department manager or a team lead. The Requirements Engineer should fully understand the nature of the requirements conflict before attempting to resolve it. This means that you will have to collect more information about the conflict itself and the stakeholders involved.

Many aspects deserve attention:

- *Subject matter*: the scope, the problem, or the real issue behind the conflict.
- *Affected requirements*: which specific requirements are affected?
- *Stakeholders involved*: who disagrees with whom about what?
- *Opinions* of the stakeholders: let them make their point as clearly as possible so that all conflicting parties understand the underlying issue.
- The *cause* of the conflict: what is the reason behind the difference in opinions?
- The *history* of the conflict: what has happened before that influences these opinions now?
- *Consequences*: the estimated costs and risks associated both with resolving the conflict or not resolving it.
- *Project constraints*: personal, organizational, content-specific, or domain-specific constraints may determine the solution space.

Analyzing this information will help you to recognize the type of conflict (for more information, see Section 4.3.2) and will indicate ways to resolve it.

► Conflict resolution

Once an in-depth understanding of the nature of the requirements conflict, the attitude of the stakeholders involved, and the project constraints has been reached, the Requirements Engineer will select a suitable resolution technique. Many techniques can be used, as explained in Section 4.3.3. The first step should always be to get the chosen technique accepted by the stakeholders involved before applying it. If some stakeholders do not agree up front with the application of a certain technique, they certainly will not accept the outcome of it, so at the end, the conflict will not be resolved. In principle, the Requirements Engineer is not one of the stakeholders involved, so you can and should apply the selected resolution techniques in an objective, strictly neutral way, and welcome any outcome that results from applying the technique.

► Documentation of conflict resolution

Conflict resolution may influence the requirements in a way that is not obvious for someone who was not involved in the conflict. The resulting set of requirements may seem illogical or inefficient. Therefore, the conflict should be properly documented and communicated with regard to aspects such as the following:

- Assumptions concerning the conflict and its resolution
- Potential alternatives considered
- Constraints influencing the chosen technique and/or resolution
- The way the conflict was resolved, including reasons for the chosen resolution
- Decision-makers and other contributors

If you do not document the resolution, after a while, stakeholders may simply forget or ignore the decisions that have been taken. And later in the project, developers may not understand the rationale behind a particular system design and may implement it in a different way.

You do not need to be afraid of requirements conflicts, as they will always occur. This should not be a surprise to you; in fact, you should be troubled if you do not detect any conflicts. They are quite common, so if you do not find them, you have probably missed some. But never ignore them. If you do not resolve all requirements conflicts that you notice right away, they will pop up later in the development process. And as Barry Boehm [Boeh1981] already found out a long time ago, the later you discover a problem, the more expensive it will be to solve it.

4.3.2 Conflict Types

To achieve a better understanding of the nature of a conflict, it is useful to distinguish between different conflict types. This helps in selecting proper resolution techniques.

We discern six types of conflict:

Conflict types

► Subject matter conflict

A *subject matter* conflict occurs when the conflicting parties really have different factual needs, mostly caused by the intended use of the system in different environments. A good example is a system that is to be used in different countries, each with their own legislation. It may be difficult to resolve such a conflict because the underlying facts cannot be changed. The first thing to do then is to analyze and document these facts in detail and to have the conflicting parties agree on the exact nature of the conflict.

► Data conflict

A *data conflict* is present when some parties refer to inconsistent data from different sources or interpret the same data in a different way. This may be due to poor communication, missing background data, cultural differences, existing prejudices, etc. Estimates in particular, such as future sales, can easily generate a data conflict as they are often based on assumptions. Detecting a data conflict is not easy, because as a Requirements Engineer, you may think that your own sources are right and your own interpretation is self-evident. Due to this bias, you often suspect another conflict type at first.

Understanding how people can come to a different interpretation requires a lot of empathy. Communication—over and over again—is key for both detecting and resolving this type of conflict.

► Interest conflict

An *interest conflict* is based on different positions of the conflicting parties, formed by personal goals, goals related to a group, or goals related to a role. You should understand the concerns and needs of the stakeholders involved before you can resolve this type of conflict. However, keep in mind that in the case of personal interests, stakeholders often do not reveal their true motives and they put forward seemingly factual but essentially artificial arguments. If a discussion is about an interest conflict, you can observe the conflict parties trying to convince each other to follow their arguments and understand the needs of the role or group. Resolution may benefit from identifying and strengthening shared interests. Working on a mutual understanding about the gains and pains of both parties can be a starting point for finding a solution.

► Value conflict

A *value conflict* is based on differences in values and principles of the stakeholders involved. Compared to an interest conflict, a value conflict is more individual and related to global and long-term perspectives. Values are more stable than interests and rarely change in the short term. If a value conflict is the reason for a discussion, the conflict parties will emphasize why their arguments are important from their point of view, revealing their inner values and principles. They tend to insist on their arguments and are unwilling to give up. To resolve such conflicts, look for higher values that unite the parties. Value conflicts are notoriously difficult to resolve and achieving mutual understanding and recognition of each other's principles is the best you can get.

► Relationship conflict

A *relationship conflict* is usually based on negative experiences with another party in the past, or in comparable situations with similar people. Often, emotions and miscommunication are involved, which makes the conflict a lot more difficult to solve. Conflict parties misuse discussions on requirements to express their anger with the behavior of each other, forgetting about facts, figures, and fairness. Bringing the discussion back to requirements will rarely help; sometimes, uniting parties around a higher value is successful. In most cases, you will have to escalate the issue to other stakeholders or a higher level of authority; exchanging people is a potential resolution. Be aware that a relationship conflict often co-occurs with other conflict types—for instance, an interest conflict. Analyzing the root cause and solving the other conflict type may then be the best way to improve the relationship.

► Structural conflict

We call a conflict *structural* when it involves inequality of power, competition over limited resources, or structural dependencies between parties. The resulting imbalance (often perceived by only one of the parties) causes problems in communication and decision making. Another reason for such conflicts may be restrictions on resources or dependencies on work products to be delivered by another party. Parties may use the discussion on requirements to either change or preserve the status quo. Hierarchy may be misused to push through decisions. For structural conflicts too, escalating the issue to other stakeholders or a higher level of authority is often necessary.

Most requirements conflicts can be categorized as either a subject matter, data, interest, or value conflict. Relationship and structural conflicts are often not directly related to requirements and therefore the Requirements Engineer may not be the appropriate party to resolve them. However, in reality, most conflicts fall into more than one category as different causes interact. Therefore, it is advisable to pay attention to all kinds of conflicts, even if the solution is not within your own responsibility. If someone else should resolve the conflict, make sure that it happens; as long as a conflict is not resolved, it will continue to have a negative impact on your work as a Requirements Engineer.

Mixed conflicts

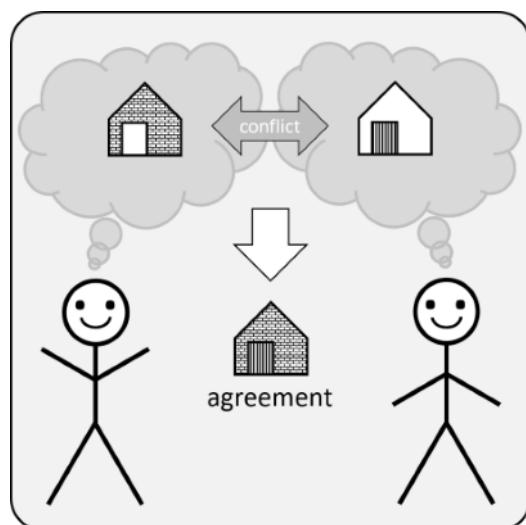
4.3.3 Conflict Resolution Techniques

Depending on the type and the context (stakeholders, constraints, etc.) of a conflict, a proper resolution technique is selected. Commonly used techniques include [PoRu2015]:

► Agreement

An *agreement* results from a discussion between the stakeholders involved, to be continued until they completely understand each other's positions and agree to a certain option preferred by all parties. It can be very time-consuming, especially when multiple parties are involved. If successful, it will provide additional motivation to the stakeholders, so the result has a good chance of being long lasting. Striving to reach an agreement is common in data conflicts. If this technique is unsuccessful within an acceptable timeframe, other techniques can be used thereafter.

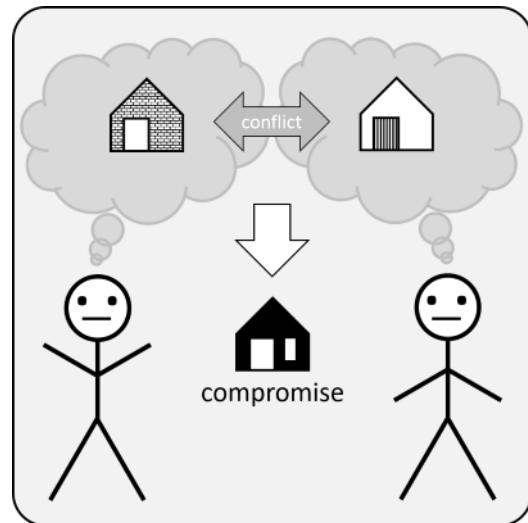
Agreement



► Compromise

Compromise

A *compromise* is quite similar to an agreement. Here, however, stakeholders agree on an option that is not their preference but that they can live with because accepting the compromise is considered better than continuing the conflict. Therefore, a compromise can also be long lasting. The compromise may contain new elements that were not present in the original preferences of the stakeholders and that may have been introduced by the Requirements



Engineer. A good compromise is an alternative in which all parties feel comfortable with the balance of giving up things and getting something else in return. A compromise is often next in line if an agreement cannot be reached in time. It is suitable for subject matter conflicts and may also work for interest and structural conflicts.

► Voting

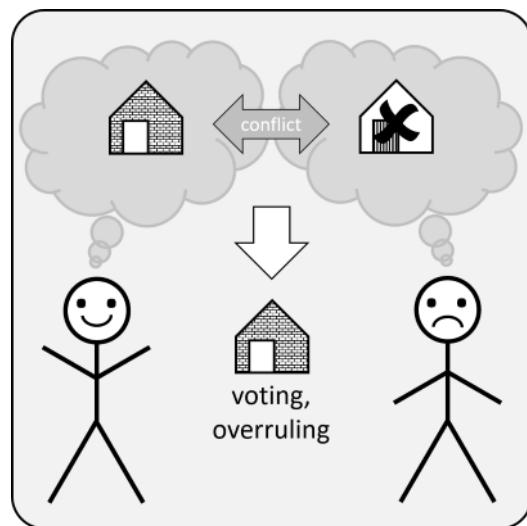
Voting

Voting works best when a relatively simple choice has to be made between a clear set of conflicting requirements. Stakeholders that participate in the voting (usually not only the conflicting parties but all stakeholders involved) should fully understand the alternatives and the consequences of their vote. In order to avoid influences from dependencies or an imbalance of power, voting is best done anonymously and with a neutral moderator. The voting procedure itself should be agreed upon between the stakeholders before the actual voting. Voting is a quick and easy means for conflict resolution but the party that loses the vote will be disappointed and may need attention. Voting can work for most conflict types and may be a good way to solve subject matter and interest conflicts.

➤ Overruling

Overruling

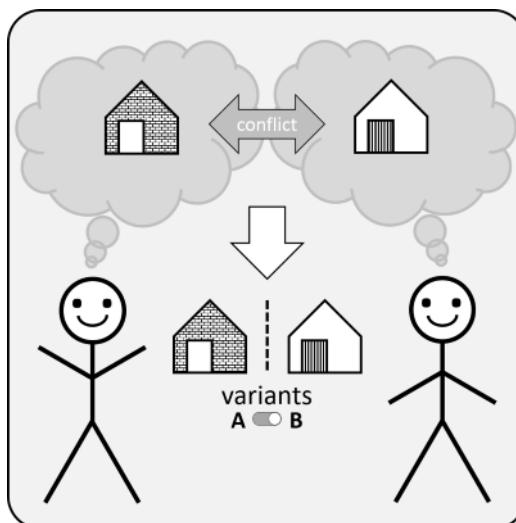
If an agreement or a compromise cannot be reached and at least one of the conflicting parties refuses to participate in voting, *overruling* may be an option. It is often applied under pressure, when there is not enough time to use more convenient techniques. Usually, overruling is done by transferring the choice between conflicting requirements to a decision maker who is higher in authority or hierarchy than all conflicting parties and has enough power to have the decision be implemented. Therefore, it is a good way to solve interest and structural conflicts. In this situation, it is particularly important that the decision maker fully understands the alternatives, the position of the conflicting parties, and the consequences of the decision. A variant of overruling is to outsource the decision to a third party—for instance, an external expert. In that case, it is important to first get an agreement between the stakeholders on the decision maker. As with voting, you may need to pay attention to the *loser*.



➤ Definition of variants

Definition of variants

Definition of variants is often considered for subject matter, interest, and value conflicts. We have seen that we cannot implement conflicting requirements in one and the same system. Definition of variants means that we build separate solutions for all conflicting requirements. This is usually implemented by developing a system that can be configured through parameters to exhibit the desired features. This may seem like a perfect solution but it comes at a price: it takes a lot of time to define the solution and a growing complexity (as well as additional costs) is introduced into the system, both for development and during operations and maintenance. This technique is therefore feasible only if enough time and budget are available.



➤ Auxiliary techniques

Auxiliary techniques

In addition, there are several *auxiliary techniques* that are not usually used on their own but rather to assist the above-mentioned techniques.

CAF

In *Consider-All-Facts* (CAF), you consider alternative solutions for a number of predefined criteria—for example, cost, time, risk, available resources. Weighing these criteria can provide more clarity about the pros and cons of the alternatives and help to identify the *best* alternative.

PMI

Plus-Minus-Interesting (PMI, see [DeBo2005]) is a brainstorming and decision-making tool. It encourages the examination of ideas and concepts from more than one perspective and is therefore valuable for conflict resolution. In PMI, the participants (usually all stakeholders involved) first identify all positive aspects (plus) of the alternatives, then the negatives (minus), and finally the interesting points, things that need further investigation. The alternative with the most pluses and the fewest minuses is the preferred alternative.

Decision matrix

In fact, both CAF and PMI are variants of the *decision matrix*, a *methodical* approach for conflict resolution. The conflicting requirements are assessed based on a (larger) number of criteria, after which, scores on these aspects are used to calculate a (weighted) final score for the alternatives. The *highest* score then wins, like *Alternative 1* in the example of Table 4.2 below. In fact, prioritization (see Section 6.8) is then used as a resolution technique. As stated earlier, these techniques are usually seen as auxiliary: they create more insight into the alternatives and thus help with the chosen resolution technique. They can even be used as a single technique if all stakeholders involved agree to accept the outcome.

Table 4.2 Example of a decision matrix

Criterion	Weight	Alternative 1: iPhone only		Alternative 2: Android & iPhone	
		Score	Weighted	Score	Weighted
Cust. base	2	3	6	4	8
Dev. cost	1	3	3	2	2
T.t. market	3	4	12	2	6
Reputation	2	2	4	4	8
User exp.	1	5	5	3	3
Total			30		27

4.4 Validation of Requirements

In Chapter 2, Principle 6, we emphasized the importance of validating the requirements to avoid unsatisfied stakeholders. Because the requirements form the input for subsequent system development, we must ensure their quality up front to reduce wasted effort downstream, both at the level of the individual requirements and of the complete set (Figure 4.10).

Validation of requirements

We should validate the coverage of stakeholders' needs by our documentation, the degree of agreement among all stakeholders, and the likelihood of our assumptions about the system context before we hand over requirements to the developers or suppliers. Although the level of detail may vary, this applies just as well for iterative as for sequential development approaches.

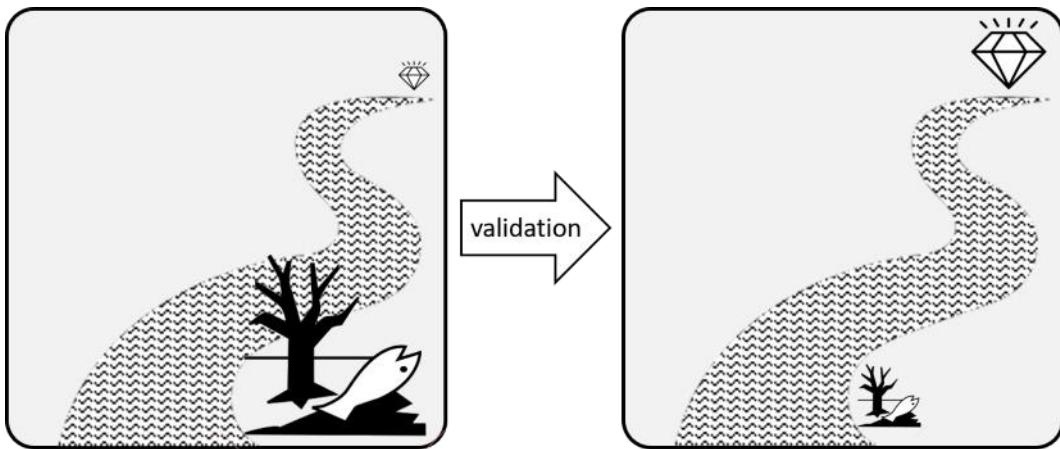


Figure 4.10 Upstream quality reduces downstream waste

Validation adds time and cost to the project, so its efficiency and effectiveness should be a concern of the Requirements Engineer. Therefore, it is important to continuously monitor and analyze defects that occur during development and in operation. If the root cause of such defects appears to be in the requirements, the requirements validation process has somehow failed. Therefore, as a Requirements Engineer, you should continuously and actively look for opportunities to improve it.

Continuous improvement

4.4.1 Important Aspects for Validation

Regarding the concept of validation, certain aspects are important to get the maximum value from it (see also [PoRu2015]):

➤ **Involving the correct stakeholders**

As a Requirements Engineer, you need to decide who you want to invite to participate in the validation. In this respect, one important aspect that you have to consider is the degree of independence between the people involved in the elicitation of the requirements and those validating them. A low level of independence (inviting stakeholders who have already participated in the elicitation) is cheap and easy to organize but may overlook certain defects because of the own focus, blind spots, conflicting interests, or flawed assumptions of these persons. A higher degree of independence (for instance, by inviting external reviewers or auditors) takes more time and effort to organize and perform and brings higher (initial) costs but may in the long run be more effective in finding more and more severe defects. Consequently, higher risk in the project scope and/or the system context asks for a higher degree of independence.

Aspects for validation

➤ Separating the identification and the correction of defects

It may be tempting to fix every defect as soon as it has been detected. However, this usually proves to be neither an efficient nor an effective way of working, as defects may influence each other. A defect found later during validation might invalidate the fixing of an earlier one. A requirement initially marked as defective might prove to be correct when all requirements have been studied. You might decide not to fix some (minor) defects in view of the effort involved related to the total set of defects found. And after all, people involved in validating requirements should concentrate on finding defects and not on developing ideas on how to fix them. Therefore, the recommendation is to first select (a coherent set of) requirements for validation and to decide whether or not to fix certain defects found only after checking the whole set.

➤ Validation from different views

A proper validation is always a group effort, not an activity performed by Requirements Engineers on their own. The best results are achieved when validation is performed by an interdisciplinary team in which selected participants contribute their own expertise. In general, we can say that the input, the output, and peers should be represented. In iterative projects, the current agile team is a reasonable choice, but the degree of independence may be low and additional validators should be invited; in sequential projects, a specific team may be composed for each separate validation effort. Depending on the phase of the project, input from business, users, developers, testers, operators, and application managers is useful; sometimes, subject matter experts or specialists on topics such as performance, security, and usability can be added.

➤ Repeated validation

In sequential projects, most requirements are elicited and documented in the initial phase and validated thoroughly at the end of that phase. However, this should not be the only moment for validation. During the rest of the project, new insights can lead to the original set of requirements being updated, detailed, and expanded. This might threaten the quality, coherence, and consistency of the requirements and thus additional validations may be required. These are often planned at project milestones.

In iterative projects, many of the agile rituals include validation efforts. Sprint planning, backlog refinement, sprint reviews, and even daily standups offer opportunities to validate and improve the requirements. However, these efforts often focus on individual, detailed requirements and the big picture may be neglected. An initial validation of the complete product backlog at the start of a project or increment is a good beginning. Other useful initiatives are repeated hardening sprints and additional overall validation at release times.

4.4.2 Validation Techniques

Validation techniques

As for other techniques, the Requirements Engineer can choose from a large toolbox of validation techniques that differ in formality and effort. Many factors influence the selection of these techniques—for instance, the software development life cycle model, the maturity of the development process, the complexity and risk level of the system, legal or regulatory requirements, and the need for an audit trail.

Often, in the course of a project, the degree of effort and formality increases towards the end, as final decisions about the system and its implementation have to be taken. Also, you will see that the amount, value, and level of detail of feedback from the stakeholders increase as the work products to be validated become more concrete and detailed. This entails the application of different validation techniques in different stages of the project. At the beginning of a project, frequent short, lightweight validation and feedback cycles are preferred, as is usual in agile approaches. This ensures quality right from the start. Later in the project, more formal and time-consuming one-off techniques will prevail.

In general, we discern three categories of validation techniques (see Figure 4.11):

- ▶ Review techniques
- ▶ Exploratory techniques
- ▶ Sample development

Review techniques and sample development are called static, as they concentrate on analyzing the specifications of a system without executing it. In exploratory techniques, the validation focuses on the actual (or simulated) behavior of the system in operation; these techniques are called dynamic.

Static vs dynamic

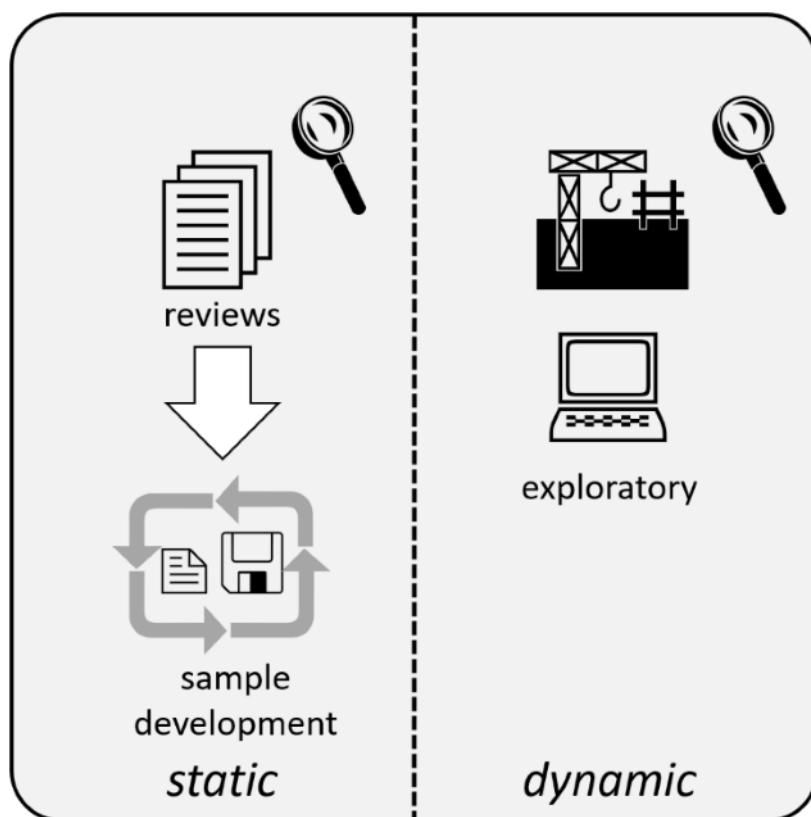


Figure 4.11 Categories of validation techniques

The common feature of *review techniques* is that they rely on visual study of early and intermediate work products. They range from informal to very formal and can be applied from the very beginning of a project until the implementation of the system. In most cases, reviewing the requirements is limited to the earlier phases of a project. Typically, in a review, we check *static* work products that define or describe how the system should work. For more information about reviewing, see [OleA2018].

[Review techniques](#)

Informal reviews usually follow the author-reviewer cycle. An author sends a work product to a group of people with the request to validate it. Usually, this is a small group of team members, peers, and/or users involved in the project. Authors may select the group by themselves or its composition may be prescribed by company regulations. After a short (but often not predefined) period, the author collects all review comments and uses them to update the work product at hand. It is good practice to document the comments in a review register and to keep track of the way in which they are processed. However, due to the informal nature of this type of review, authors are free to decide whether and how to use the comments. Often, the review is repeated over several draft versions until the author is satisfied with the quality.

[Informal reviews](#)

As they are informal, you might expect little benefit from these kinds of reviews for validating and improving the quality of requirements. However, if all participants are committed to quality, and are able and willing to spend enough time on the review process, informal reviews are an easy, cheap, and approachable means of validation. In fact, this approach is common for early drafts. For the final version of a work product, a more formal technique may be a better choice.

Formal reviews follow a prescribed way of working. They are often used for important or milestone work products, for final versions, and in situations where high risks are at stake. While there are many flavors of formal reviews, they can be divided into two main groups:

[Formal reviews](#)

➤ Walkthroughs

The essence of a *walkthrough* is that the author of a work product explains it step by step to an audience in an interactive session. In practice, walkthroughs come in two variants, where (1) reviewers join the meeting without any preparation and listen to the author, asking ad hoc questions; or (2) they obtain the work product before the meeting and will prepare questions for the author. Participants in the audience can make comments, identify flaws, and suggest alternatives. The author gives more explanation if necessary and can discuss solutions for weaknesses identified and weigh alternatives against the original ideas. There are two occasions where walkthroughs are best applied: (a) in an early project phase to discuss the feasibility of a certain system concept or solution outline; and (b) on the transfer of an intermediate work product to another party who will use it as input for subsequent development. In iterative projects, walkthroughs are mostly present in the form of regular refinement sessions prior to an iteration and sprint reviews at the end of it.

➤ Inspections

Inspections are among the most formal review techniques. Here, the responsibility for the review lies not with the author but with an independent review leader, often called *moderator*. An inspection is normally performed in the form of a meeting with the moderator, the author, and a group of *inspectors*.

The inspectors are selected from peers, business, users, and/or experts. They are asked to check the work product based on their specific expertise, to verify its adherence to applicable standards, norms, and regulations, and to evaluate it against agreed objectives. Often, this check by the inspectors is performed during thorough individual preparation prior to the actual meeting, guided by detailed checklists. In the review meeting, the author participates in the role of a listener, explaining things that are not clear and trying to understand the comments of the inspectors and the consequences for the work product. Typically, an inspection follows a strict and documented process that is managed by the moderator and focuses on finding defects and measuring defined quality aspects and provides a detailed audit trail. In this form, inspections are often used to decide on the release of a work product for a next step in the development process, or even for final implementation. Inspections are mostly applied in (safety-) critical systems and business processes. In agile approaches, this formal way of reviewing is incorporated in the methodology itself—for example, with the Scrum ceremonies (refinement, planning, sprint review).

Exploratory techniques offer a group of stakeholders and prospective users the opportunity to gain hands-on experience with an intermediate version of (part of) the system under development. In contrast to reviews, exploratory techniques are *dynamic*: they look at the (actual or simulated) behavior of the operative system as experienced by the users through the user interfaces. The participants are invited to use the system in a way that is similar to the intended use in production. They are relatively free to do so but sometimes certain guidance is given. After a period of use, the participants report their experiences and their feedback on the current behavior of the system to the Requirements Engineer. This may include defects found and suggestions for improvement.

Exploratory techniques

Exploratory techniques are common in iterative and design thinking development approaches. In fact, the usual incremental development, starting with the release of a *minimum viable product (MVP)*, followed by the addition of more functionality step by step, while carefully measuring market reactions and adjusting the system accordingly, can be seen as an exploratory validation of the requirements in production.

Iterative development

Common exploratory techniques include:

➤ **Prototyping**

In validation with *prototyping*, a specific early version of the system is given to a group of stakeholders for evaluation. This version may be explicitly built for validation purposes, after which it is discarded; we call this an exploratory or throwaway prototype. Of course, evolutionary prototypes, which are continuously updated and extended until they end up in the final product, can also be used for validation during their development. The essence of any prototype is that, from the outside, it looks like the intended system, allowing stakeholders to gain hands-on experience while the internal structure may still be unfinished, inoperative, or even completely missing. When using a prototype for validation, you may have it built to check a specific characteristic, such as user interface, security, or performance.

As we saw in Section 4.2.3, prototyping and storyboarding can also be used as elicitation techniques. In fact, these techniques support both elicitation and validation, going hand in hand: while validating requirements elicited at an earlier point in time, you will almost certainly detect new requirements in the feedback from the participants. Both aspects of prototyping are very prominent in design thinking approaches (see [LiOg2011]).

Elicitation and validation go together

➤ Alpha testing and beta testing

In alpha testing and beta testing, a fully featured, completely working pre-production version of the system is provided to end users for operation with the intended business processes in a realistic environment.

Alpha testing is done at the developer's site in a simulated environment. The group of participants is relatively small, some guidance may be given, and it is possible to observe the interaction of the users with the system—for instance, in a usability lab.

Alpha testing

Beta testing is conducted at the end user's sites in real production (or in whatever environment the end users decide). The system is offered (mostly for free) to a (sometimes selected but usually unknown) group of users, with the implicit request to validate its looks and behavior. In beta testing, it is important to stimulate all participants to give their feedback and to provide an easy way to do so. Analyzing this feedback after a prolonged period of use can give valuable clues to the quality of the requirements. It is particularly useful for checking certain assumptions made during elicitation and development.

Beta testing

➤ A/B testing

A/B testing is often performed with a released version of the system in the fully operational environment but can also be applied with pre-release versions in a protected test environment. The essence of A/B testing is that the system is offered to different (mostly randomly selected) groups of users in two variants that differ in design or functionality and realize the user goals in a different way. The reaction of both groups is measured and compared; this works best when the groups are large enough to allow for statistical analysis. The analysis will then give information on the quality of the underlying requirements and on the correctness of previous assumptions. A/B testing has a prominent role in *The Lean Startup*, one of the design thinking approaches (see [Ries2011]).

In *sample development*, you provide a set of requirements as input for developers; they try to produce some common intermediate work products (e.g., designs, code, test cases, manuals) based on this input. The system itself is not operative (yet), so this kind of validation is *static*, just like in reviewing. During this effort, the developers may detect flaws such as unclarities, omissions, and inconsistencies that prevent them from producing their intended output. Of course, these flaws will be fixed. At the same time, however, the quantity and severity of the flaws detected is an indication of the quality of the requirements. If this quality is not sufficient, more validation is necessary—for instance, additional reviews.

Sample development

A similar validation can be performed by Requirements Engineers themselves. In that case, you try to document a set of requirements in a different form of representation to the original type: commonly, converting a requirements specification created in natural language into a relevant model, or a specific model into a textual description.

Converting documentation types

This exercise is especially useful for detecting omissions. If you encounter serious problems in this conversion, this indicates the need for additional validation.

4.5 Further Reading

Glinz and Wieringa [GIWi2007] explain the notion and importance of stakeholders. Alexander [Alex2005] discusses how to classify stakeholders. Bourne [Bour2009] deals with stakeholder management. Lim, Quercia and Finkelstein [LiQF2010] investigate the use of social networks for stakeholder analysis. Humphrey [Hump2017] discusses user personas.

Zowghi and Coulin [ZoCo2005] present an overview of requirements elicitation techniques. Gottesdiener [Gott2002] has written a classic textbook on workshops in RE. Carrizo, Dieste and Juristo [CaDJ2014] investigate the selection of adequate elicitation techniques.

Maalej, Nayebi, Johann and Ruhe [MNJR2016] discuss the use of explicit and implicit user feedback for eliciting requirements. Maiden, Gitzikis and Robertson [MaGR2004] discuss how creativity can foster innovation in RE.

The book by Moore [Moor2014] is a classic about conflict management. Glasl [Glas1999] discusses how to handle conflicts. Grünbacher and Seyff [GrSe2005] discuss how to achieve agreement by negotiating requirements when validating requirements or resolving conflicts.

Validation is covered in any RE textbook; see [Pohl2010], for example.

5. Process and Working Structure

Whenever work has to be done in a systematic way, a *process* is required to shape and structure the way of working and the creation of work products.

Process

DEFINITION 5.1. PROCESS: A set of interrelated activities performed in a given order to process information or materials.

A Requirements Engineering (RE) process organizes how RE tasks are performed using appropriate practices and producing work products required. However, there is no proven, one-size-fits-all RE process (see Section 1.4). Consequently, Requirements Engineers have to configure a tailored RE process that fits the given situation.

Need for a tailored RE process

The RE process shapes the information flow and the communication model between the participants involved in RE (for example, customers, users, Requirements Engineers, developers, and testers). It also defines the RE work products to be used or produced. A proper RE process provides the framework in which Requirements Engineers elicit, document, validate, and manage requirements.

Shapes information flow and communication

In this chapter, you will learn about the factors that influence the RE process and how to configure an appropriate process from a set of process facets.

5.1 Influencing Factors

There are a variety of influencing factors to consider when configuring an RE process. Before starting with the configuration of an RE process, these factors need to be investigated and analyzed.

Important influencing factors

On the one hand, such analysis provides information about how to configure the RE process. For example, when the analysis indicates that stakeholders have only a vague idea about their requirements, an RE process should be chosen that supports the exploration of requirements. On the other hand, the influencing factors constrain the space of possible process configurations. For example, if the stakeholders are available only at the beginning of a system development project, a process that builds upon continuous stakeholder feedback would not be suitable. Below, we discuss important factors for the RE process.

Overall process fit

Overall process fit. When defining or configuring an RE process, it is vital to know and understand the overall development process chosen for the system to be developed—defining an RE process that does not fit the overall process does not make sense. The overall process may require work products that the RE process must deliver. The terminology used for the RE process should be aligned to the terminology of the overall process. In particular, the terminology for the work products must be aligned. This helps avoid confusion and misunderstandings. It also makes the introduction of the RE process as well as the training and coaching of the people who have to work according to the process easier. For example, if the system is developed using a linear, plan-driven process that relies on the existence of a comprehensive system requirements specification and a system glossary at the end of the requirements phase, the RE process chosen must fit into the requirements phase of the overall process and produce the two work products required.

Development context

Development context. The development context also informs the RE process. Things to consider include the customer-supplier-user relationship, development type, contract issues, and trust. When analyzing the development context, a couple of questions need to be answered:

- ▶ *Customer-supplier-user relationship:* Is there a designated customer who orders the system and pays for it and a supplier who develops the system? Are customer and supplier part of the same organization or do they belong to different organizations? If the former is the case, which people act in the role of customer and which act as supplier? Who are the users of the system? Do the users belong to the customer's organization? If not, do they use the system as a product or service for interacting with the customer (for example, in electronic business) or do they buy the system as a product or service from the customer (for example, a mobile app)?
- ▶ *Development type:* What is the organizational framework for the development of a system? Typical types include:
 - (1) A supplier specifies and develops a system for a specific customer who will use the system.
 - (2) An organization develops a system with the intention to sell it as a product or service to many customers in a certain market segment.
 - (3) A supplier configures a system for a customer from a set of ready-made components.
 - (4) A vendor enhances and evolves an existing product.
- ▶ *Contract:* Is there a contract or similar agreement that formally defines deliverables, costs, deadlines, responsibilities, etc.? Contracts may be classic fixed-price contracts between a customer and a supplier, with fixed functionality, deadlines, and cost, or may just give a financial framework, while the functionality is defined iteratively.
- ▶ *Trust:* Do the parties involved trust each other? If, for example, the customer and the supplier do not trust each other, the requirements have to be specified in more detail than would be necessary in a trust-based relationship.

Stakeholder availability and capability. The availability of stakeholders constrains the configuration options for the RE process. For example, a process requiring continuous close interaction with stakeholders cannot be chosen if core stakeholders are available only for a short period of time at the beginning of the process.

Stakeholder availability and capability

The capability of the stakeholders also influences the process: the less stakeholders are able to express their needs clearly, and the less they know their actual needs, the more the RE process must accommodate the exploration of requirements.

Shared understanding

Shared understanding. Only little Requirements Engineering is needed when there is a high degree of shared understanding (see Chapter 2, Principle 3) between stakeholders, Requirements Engineers, designers, and developers about the problem and the requirements. Consequently, the better the shared understanding, the more lightweight the RE process can be [GIFr2015].

Complexity and criticality. The degree of detail to which requirements need to be specified depends strongly on the complexity and criticality of the system to be developed. When a system is complex and/or critical with respect to safety or security, the RE process chosen must accommodate a detailed specification of the critical requirements, including formal or semi-formal models and strong validation—for example, by verifying models that express prescribed behavior or by building prototypes.

Complexity and criticality

Constraints. Obviously, all influencing factors constrain the space of possible configurations of an RE process. When we talk about constraints, we mean those constraints that are explicitly imposed by, for example, the customer or a regulator. Such constraints may imply the mandatory creation of certain work products and following a mandatory process for producing these work products. Customers or regulators may also demand an RE process that conforms to some given standard.

Constraints

Time and budget available. If schedules and budgets are tight, the time and budget available for RE need to be used wisely, which typically implies choosing a lightweight RE process. Choosing an iterative RE process helps with prioritizing requirements and implementing the most important ones within the given budget and schedule.

Time and budget available

Volatility of requirements. If many requirements are likely to change, it is advisable to choose an iterative, change-friendly RE process.

Volatility of requirements

Experience of Requirements Engineers. The RE process chosen should match the competencies and experience of the Requirements Engineers involved. Otherwise, additional time and budget must be allocated to train and coach the process chosen. It is better to choose a rather simple process that the Requirements Engineers can handle properly than a sophisticated and complicated one that overburdens them.

Experience of Requirements Engineers

5.2 Requirements Engineering Process Facets

Overview of process facets

Defining the RE process from scratch for every RE undertaking is a waste of effort. Whenever the influencing factors allow it, the process should be configured from pre-existing elements. In order to provide guidance on how to configure a proper RE process, we describe three facets with two instances each, together with selection criteria to be considered for each instance [Glin2019]. Later, in Section 5.3, we use these facets to configure RE processes.

Figure 5.1 shows an overview of the facets and instances.

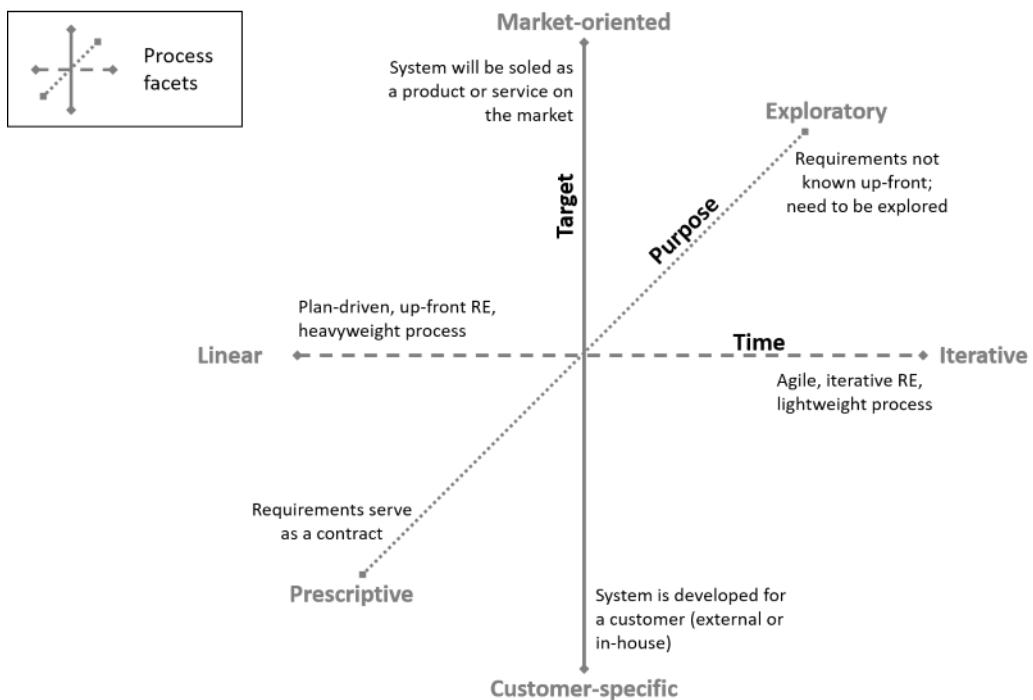


Figure 5.1 RE process facets

The facets can be considered to span a three-dimensional space of process configuration options. Every facet instance comes with criteria for selecting it.

The applicability of these criteria stems from the analysis of the influencing factors discussed in Section 5.1 above. Note that not all criteria need to be fulfilled to choose an instance of a facet.

5.2.1 Time Facet: Linear versus Iterative

The time facet deals with the organization of RE activities on a time scale. We distinguish between linear and iterative processes.

In a *linear RE process*, requirements are specified up front in a single phase of the process. The idea is to produce a comprehensive requirements specification that requires no or only little adaptation or few changes during the design and implementation of the system. Creating a comprehensive requirements specification up front calls for a comprehensive process. Thus, in most cases, linear RE processes are heavyweight processes.

Linear

Criteria for choosing a linear RE process:

- ▶ The development process for the system is plan-driven and mostly linear.
- ▶ The stakeholders are available, know their requirements, and can specify them up front.
- ▶ A comprehensive requirements specification is required as a contractual basis for outsourcing or tendering the design and implementation of the system.
- ▶ Regulatory authorities require a comprehensive, formally released requirements specification at an early stage of the development.

In an *iterative RE process*, requirements are specified incrementally, starting with general goals and some initial requirements and then adding or modifying requirements in every iteration. The idea is to intertwine the specification of requirements with the design and implementation of the system. Due to short feedback loops and the ability to accommodate change or things forgotten in later iterations, iterative RE processes can be lightweight processes.

Iterative

Criteria for choosing an iterative RE process:

- ▶ The development process for the system is iterative and agile.
- ▶ Many requirements are not known up front but will emerge and evolve during the development of the system.
- ▶ Stakeholders are available such that short feedback loops can be established as a means of mitigating the risk of developing the wrong system.
- ▶ The duration of the development allows for more than just one or two iterations.
- ▶ The ability to change requirements easily is important.

5.2.2 Purpose Facet: Prescriptive versus Explorative

The purpose facet deals with the purpose and role of the requirements in the development of a system. We distinguish between prescriptive and explorative RE processes.

In a *prescriptive RE process*, the requirements specification constitutes a contract: all requirements are binding and must be implemented. The idea is to create a requirements specification that can be implemented with no or little further interaction between stakeholders and developers.

Prescriptive

Criteria for choosing a prescriptive RE process:

- ▶ The customer requires a fixed contract for system development, often with fixed functionality, scope, price, and deadline.
- ▶ Functionality and scope take precedence over cost and deadlines.
- ▶ The development of the specified system may be tendered or outsourced.

In an *explorative RE process*, only the goals are known a priori, while the concrete requirements have to be elicited. The idea is that requirements are frequently not known a priori but have to be explored.

Explorative

Criteria for choosing an explorative RE process:

- ▶ Stakeholders initially have only a vague idea about their requirements.
- ▶ Stakeholders are strongly involved and provide continuous feedback.
- ▶ Deadlines and cost take precedence over functionality and scope.
- ▶ The customer is satisfied with a framework contract about goals, resources, and the price to be paid for a given period of time or number of iterations.
- ▶ It is not clear a priori which requirements shall actually be implemented and in which order they will be implemented.

5.2.3 Target Facet: Customer-Specific versus Market-Oriented

The target facet considers the development type: which kind of development do we target with the RE process? On an elementary level, we distinguish between customer-specific and market-oriented RE processes.

Customer-specific

In a *customer-specific RE process*, the system is ordered by a customer and developed by a supplier for this customer. Note that the supplier and the customer may be part of the same organization. The idea is that the RE process reflects the customer-supplier relationship.

Criteria for choosing a customer-specific RE process:

- ▶ The system will be used mainly by the organization that has ordered the system and pays for its development.
- ▶ The important stakeholders are mainly associated with the customer's organization.
- ▶ Individual persons can be identified for the stakeholder roles.
- ▶ The customer wants a requirements specification that can serve as a contract.

In a *market-oriented RE process*, the system is developed as a product or service for a market, targeting specific user segments. The idea is that the organization that develops the system also drives the RE process.

Market-oriented

Criteria for choosing a market-oriented RE process:

- ▶ The developing organization or one of its clients intends to sell the system as a product or service in some market segment.

- ▶ Prospective users are not individually identifiable.
- ▶ The Requirements Engineers have to design the requirements so that they match the envisaged needs of the targeted users.
- ▶ Product owners, marketing people, digital designers, and system architects are primary stakeholders.

5.2.4 Hints and Caveats

It is important to note that the criteria given above are heuristics. They should not be considered as a set fixed rules that always apply. For example, outsourcing the development of the system is done preferably with a prescriptive RE process rather than with an explorative one. This is because the contract between the customer and the supplier is typically based on a comprehensive requirements specification. However, it is also possible to negotiate an outsourcing contract based on an explorative RE process.

Heuristics, no rules

There may be prerequisites for choosing certain instances of process facets or the choice may entail consequences that have to be considered. Here are some examples:

Prerequisites and consequences

- ▶ Linear RE processes work only if a sophisticated process for changing requirements is in place.
- ▶ Linear RE processes imply long feedback loops: it may take months or even years from writing a requirement until its effects are observed in the implemented system. To mitigate the risk of developing the wrong system, requirements must be validated intensively when using a linear RE process.
- ▶ In a market-oriented process, feedback from potential users is the only means of validating whether the product will actually satisfy the needs of the user segment targeted.
- ▶ In an agile setting, an iterative and explorative RE process fits best. Iterations have a fixed length (typically 2-6 weeks). The product owner plays a core role in the RE process, coordinating the stakeholders, organizing the RE work products, and communicating the requirements to the development team.

The three facets mentioned above are not fully independent: the choice made for one facet may influence what can or should be chosen in other ones. Here are some examples:

Mutual influence

- ▶ Linear and prescriptive are frequently chosen together, which means that when Requirements Engineers decide on a linear RE process, they typically decide on a process that is both linear and prescriptive.
- ▶ Explorative RE processes are typically also iterative processes (and vice versa).
- ▶ A market-oriented RE process does not combine well with a linear and prescriptive process.

5.2.5 Further Considerations

The degree to which an RE process must be established and followed, as well as the volume of requirements work products to be produced in this process, depends on the degree of shared understanding and also on the criticality of the system.

Shared understanding and criticality

The better the shared understanding and the lower the criticality, the simpler and more lightweight the RE process can be.

When there is little time and budget available for RE, the resources available must be used carefully. Choosing an iterative and explorative process helps. Furthermore, the process should focus on identifying and dealing with those requirements that are critical for the success of the system.

Time and budget

Finally, the RE process should fit the experience of the Requirements Engineers. The lower their skills and experience, the simpler the RE process should be made—it does not make sense to define a sophisticated process when the people involved cannot enact this process properly.

Experience of Requirements Engineers

5.3 Configuring a Requirements Engineering Process

In a concrete system development context, Requirements Engineers or the person(s) responsible for RE have to choose the RE process to be applied. We recommend analyzing the influencing factors (see Section 5.1) first and then selecting a suitable combination of the process facets described in Section 5.2.

Process configurations

5.3.1 Typical Combinations of Facets

Three combinations of facets (or variants thereof) frequently occur in practice [Glin2019]. In the following, we briefly describe each of them and characterize them in terms of their main application case, typical work products, and typical information flow. Furthermore, we provide an example. Figure 5.2 shows the three typical process configurations in the space of the three facets.

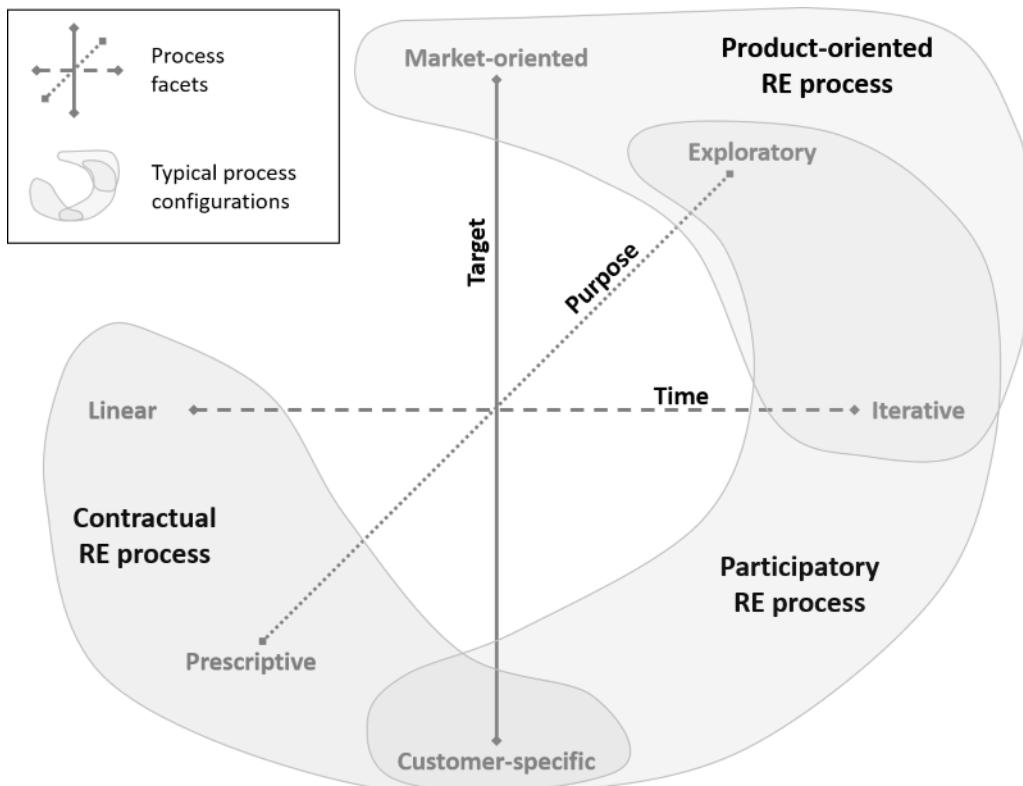


Figure 5.2 Three typical RE process configurations and their relationship to the three facets

Participatory RE Process: Iterative & Explorative & Customer-Specific

A participatory RE process is typically chosen in agile settings when there is a customer who orders a system and a development team that designs and implements it. The focus is on exploring the requirements in a series of iterations in close collaboration between the stakeholders on the customer side, the Requirements Engineers, and the development team.

*Participatory
Mostly agile
For a customer*

Main application case: Supplier and customer collaborate closely; stakeholders are strongly involved in both the RE and the development processes.

Typical work products: Product backlog with user stories and/or task descriptions, vision, prototypes

Typical information flow: Continuous interaction between stakeholders, product owners, Requirements Engineers, and developers

Example: In an insurance company, the business unit that sells corporate insurances to small and medium-sized enterprises has an idea about a new product for insuring customers against the damage incurred by a hacker attack. They contract the corporate IT unit of the company to form a development team with the task of designing and developing a new application that can handle the new insurance product within the existing insurance sales support system. Also, the existing insurance contract management system needs to be adapted accordingly. Beyond some initial requirements, the contracting business unit has no clear idea how the new product should look and how it should be supported by the corporate IT systems. Corporate IT adopted agile development for all their projects some years ago.

Example: Supporting a new insurance product

In this situation, a participatory RE process is appropriate. It fits the overall agile process that corporate IT will employ to develop the new system and adapt the existing ones. Stakeholders from the business unit and Requirements Engineers from corporate IT can jointly elicit the requirements for the new insurance product. As the process is iterative, the development team can develop a prototypical minimum marketable product that helps the management of the business unit to decide whether or not to include the product envisaged in their portfolio or discard the idea. There is a clear customer-supplier relationship between the business unit and corporate IT, so a customer-oriented RE process fits.

Contractual RE Process: Typically Linear & Prescriptive & Customer-Specific

A contractual RE process is typically chosen when the development of a system is tendered and outsourced to a provider with a contract based on a comprehensive requirements specification. It is also a suitable process for RE in large system development projects that apply a waterfall-style development process.

*Contractual
Up-front RE
Waterfall-style
For a customer*

Main application case: The requirements specification constitutes the contractual basis for the development of a system by people not involved in the specification and with little stakeholder interaction after the requirements phase.

Typical work products: Classic system requirements specification, consisting of textual requirements and models

Typical information flow: Primarily from stakeholders to Requirements Engineers

Example: A car manufacturer is developing a new car platform, from which a family of car models will be derived. A major design decision for the new platform is to get rid of the dozens of electronic control units (ECUs) currently used in the cars and replace them with a single control computer that runs a stack of driving control and driving assistance applications. The goal is to save hardware costs, get rid of unwanted interactions between ECUs, and reduce both time and effort for performing updates of the software. Engineers who are responsible for the electronic systems of the new platform have written a customer requirements specification. The company has contracted a large manufacturer of automotive control systems to create a system requirements specification for the new centralized car control system. Later, the car manufacturer will tender the design and implementation of the system based on that specification. The manufacturer will require the implementation to be performed in several iterations in order to ease testing and integration of the system with the new car platform.

Example: Specifying a new control system in the automotive industry

In this situation, a contractual RE process is appropriate. The overall process is linear: the system will be designed and implemented only after the requirements specification has been completed. The fact that the implementation will be iterative does not impact the RE process. Depending on the quality of the existing customer requirements specification and the availability of the stakeholders at the car manufacturer, a linear or an iterative RE process should be chosen.

Obviously, a customer-oriented RE process is needed. The existence of a customer requirements specification and the fact that the system requirements specification will be used to tender the design and implementation of the system call for a prescriptive RE process.

[Product-Oriented RE Process: Iterative & Explorative & Market-Oriented](#)

A product-oriented RE process is typically chosen when an organization is developing a system as a product or service for the market. In most cases, a product-oriented RE process comes together with an agile product development process. The product owner and digital designers play major roles in this process: they strongly influence and shape the product.

*Product-oriented
Mostly agile
For the market*

Main application case: An organization specifies and develops software in order to sell or distribute it as a product or service

Typical work products: Product backlog with user stories and/or task descriptions, vision, prototypes, user feedback

Typical information flow: Interaction between product owner, marketing, Requirements Engineers, digital designers, and developers plus feedback from customers/users

Example: A media company tasks its internal IT with a total renewal of the mobile news app that the company sells to subscribers (with some content being freely accessible). From user feedback, the company maintains a long log of customer criticism and improvement suggestions. In particular, many users criticize the existing app for not being responsive enough, for having bad support for reporting problems and suggestions, and for not supporting two-finger zooming of text or images. The marketing department of the company also perceives the layout of the app to be outdated. They predict that with a fresh layout, more subscribers could be gained. The CEO of the company has decided that the IT department shall collaborate with an external design agency for the visual appearance of the app. The management of the media company wants a minimal product version as a proof of concept, and then new intermediate versions every three weeks that can be reviewed by the marketing department and the company's board of executives.

Example: Total renewal of a mobile news app

In this situation, a product-oriented RE process fits best. Although there is a customer-supplier relationship between the management of the company and its internal IT department, the focus is clearly on creating a renewed product in the segment of mobile news applications. The RE process needs to be explorative, as the requirements beyond the information in the existing log of user feedback are not clear. The overall development process has to be iterative according to the decision of the management of the company. As the requirements need to be explored, an iterative RE process is the best fit here.

5.3.2 Other RE Processes

The three combinations described above cover many of the situations that occur in practice. However, there may be situations where none of the aforementioned process configurations fit. For example, regulatory constraints may impose the use of a process that conforms to a given standard, such as ISO/IEC/IEEE 29148 [ISO29148]. In such a case, the RE process has to be created by process experts from scratch or one of the aforementioned configurations has to be tailored so that it is adapted to the given situation.

Special cases need special RE processes

5.3.3 How to Configure RE Processes

We recommend a five-step procedure for configuring an RE process.

1. *Analyze the influencing factors.* Analyze your situation with respect to the list of influencing factors from Section 5.1.

A five-step configuration procedure

2. *Assess the facet criteria.* Based on the analysis from step 1, go through the list of facet selection criteria given in Section 5.2. You may assign each criterion a value on a five-point scale (--, -, 0, +, ++).

3. *Configure.* If the criteria analysis yields a clear result with respect to the three typical configurations mentioned above, choose that configuration. Otherwise, choose a different process tailoring, guided by the general goal of mitigating the risk of developing the wrong system. For example, imagine a situation where the customer demands a system requirements specification to be created up front, which calls for a linear, prescriptive RE process. However, in your first meetings with the customer, you have noticed that for an important subsystem, the customer has no clear idea what to build, which calls for an explorative RE process. A potential solution could be to choose a contractual RE process as the general RE process framework but create a subproject that stepwise elicits the requirements for that important subsystem, creating prototypes in two or three iterations (guided by a participatory RE subprocess), and then feed the results into the system requirements specification.

4. *Determine work products.* Based on your analysis and process configuration, define the main RE work products that will be produced. Make sure that the RE work products are aligned with the work products of the overall development process.

5. *Select appropriate practices.* For the tasks to be performed—for example, elicitation of requirements—select the practices that fit best in the given situation. Many of these practices, including hints about where and when to apply them, are presented in Chapters 2, 4, and 6 of this handbook.

There is no proven, one-size-fits-all RE process. Based on an analysis of influencing factors, a specific RE process needs to be tailored for every RE undertaking. A simple way of tailoring is configuring an RE process from a set of process facets.

5.4 Further Reading

Armour [Armo2004] and Reinertsen [Rein1997], [Rein2009] provide general thoughts on processes and information flows in processes.

Although the textbook of Robertson and Robertson [RoRo2012] is entitled “Mastering the Requirements Process,” this is a general textbook on all aspects of RE.

Wiegers and Beatty [WiBe2013] provide a chapter about improving RE processes. The book by Sommerville and Sawyer [SoSa1998] contains a collection of good practices to be used in the framework of RE processes.

6. Management Practices for Requirements

Requirements are not carved in stone, eternally present from past to future; they are alive! They are born through elicitation, grow up through documentation, and are shaped through validation. As adults, they go to work through implementation and after a—hopefully—long and prosperous life in operation, they retire in oblivion. Throughout their life cycle, their parents, the Requirements Engineers, take care of them. We nurse them in their infancy, teach them in their youth, escort them in their relationships, and help them find a good job in a healthy system. That is what we call requirements management.

Of course, there are better, more formal, definitions of requirements management. The ISO/IEC/IEEE 29148:2018 [ISO29148] standard defines requirements management as "*activities that identify, document, maintain, communicate, trace and track requirements throughout the life cycle of a system, product or service.*". In the CPRE glossary [Glin2020], requirements management is defined as "*The process of managing existing requirements and requirements related work products, including the storing, changing and tracing of requirements.*". The CPRE glossary also tells us that *requirements management* is an integral part of Requirements Engineering: "*The systematic and disciplined approach to the specification and management of requirements with the goal of ...*".

Requirements management can occur at different levels:

- ▶ The individual requirements
- ▶ The work products that contain these requirements
- ▶ The system related to the work products and the requirements contained therein

Requirements management occurs at different levels

In practice, requirements management is primarily performed at the work product level. Usually a work product contains several individual requirements (e.g. an external interface description), while other work products contain only a single requirement (e.g. a single user story in an agile project) or they represent the whole set of requirements for a system (e.g. software requirements specification). Be aware that all work products of all three levels must be managed, and make sure that you know the relationships between them.

The text above outlines the *what* of requirements management. The rest of this chapter is devoted to the *how*: all kinds of practices that are applicable to make requirements management work. Before we dive into the details of requirements management, let us consider some leading principles for making it work. If you want to manage something, you must be able to recognize it, to store it, and to find it again. Therefore, unique identification, an appropriate degree of standardization, avoidance of redundancy, a central repository, and managed access are a must.

What versus how

In Section 6.1, we take a short look at situations that influence the value, importance, and effort involved in requirements management.

Section 6.2 follows the requirements in their life cycle as part of work products that Requirements Engineers and other IT staff produce and use while developing, implementing, and operating an IT system.

During the lifecycle of a requirement, multiple versions of work products (and the requirements they contain) are created, starting with an early 0.1 draft that, after a series of major and minor changes, evolves into, say, a 3.2 final version. Version control is discussed in Section 6.3.

When developing and using IT systems, it is impractical to deal with all requirements on an individual basis. Therefore, coherent sets of requirements are recognized as configurations and baselines, as explained in Section 6.4.

In order to handle work products and requirements efficiently, we must be able to identify them and collect data about them. That is the topic of Section 6.5.

Section 6.6 looks at requirements traceability. Traceability is an especially important quality characteristic of requirements, as you may have already understood when reading the definitions of requirements management above. Without traceability, it is impossible to link the actual behavior of a system to the original demands of the stakeholders.

Section 6.7 deals with the changes to requirements that occur during their lifetime. In the first phases of their existence, changes can be frequent, but after validation, requirements should be stable. However, changes will still occur. To apply them in an orderly manner, a defined process for handling change should be in place.

By nature, requirements differ in importance and value. Usually, resources to elaborate them are limited, so not every requirement will make it to implementation. This means that stakeholders will have to decide when a certain requirement will be implemented or even whether or not it will be implemented at all. Prioritization, described in Section 6.8, can underpin this decision.

6.1 What is Requirements Management?

In the introduction, we have already seen that requirements management means the management of existing requirements and requirements-related work products, including storing, changing, and tracing the requirements. But why manage them at all?

We manage requirements because they are living things; they are created, used, updated, and deleted again during both their development and operation. And during this whole life cycle, we must make sure that all parties involved have access to the correct versions of all requirements that are relevant to them. If we do not manage requirements properly, we face the risk that some parties may overlook requirements, stick to outdated requirements, work with wrong versions, overlook relationships, and so on. This can seriously hinder the efficiency and effectiveness of system development and usage. In other words: the value of proper requirements management lies in the improved efficiency and effectiveness of a system.

This means that the value of requirements management cannot be separated from the value of the system in question and its context. In practice, we can see huge differences in the importance and level of requirements management and the effort involved [Rupp2014], ranging from an informal subsidiary task of a Requirements Engineer with a spreadsheet, to a full-time function of a dedicated requirements manager with a tool-supported database of requirements.

Value of requirements management

More thorough requirements management is needed with larger numbers of requirements, stakeholders, and developers, with a longer expected lifetime, more changes or higher quality demands on the system, and with a more complex development process, more strict standards, norms, and regulations, including the need for a detailed audit trail.

Often, we see that requirements management is somewhat neglected at the beginning of a project, when a small team is working on an obvious set of high-level requirements. Later on, complexity increases and the team loses the overview, resulting in quality problems and reduced efficiency. Then, a lot of effort has to be spent on catching up with the required level of control. It is more efficient to invest some effort right from the start of a project to set up the requirements management resources and processes with the expected demands at the end in mind.

6.2 Life Cycle Management

As stated in the introduction, requirements and work products that contain requirements have a life. We see them being created, elaborated, validated, consolidated, implemented, used, changed, maintained, reworked, refactored, retired, archived, and/or deleted. That is what we mean by their life cycle: during its life, a requirement can be in a limited number of states and can show a limited number of state transitions based on explicit events in the context. Figure 6.1 shows a simplified statechart as a model for the life cycle of a single requirement (overview only, state transitions are not shown; for instance, the transition from the composite state *Under development* to *In production* may be triggered by a go-live decision from the product owner).

*Life cycle management
of requirements and
work products*

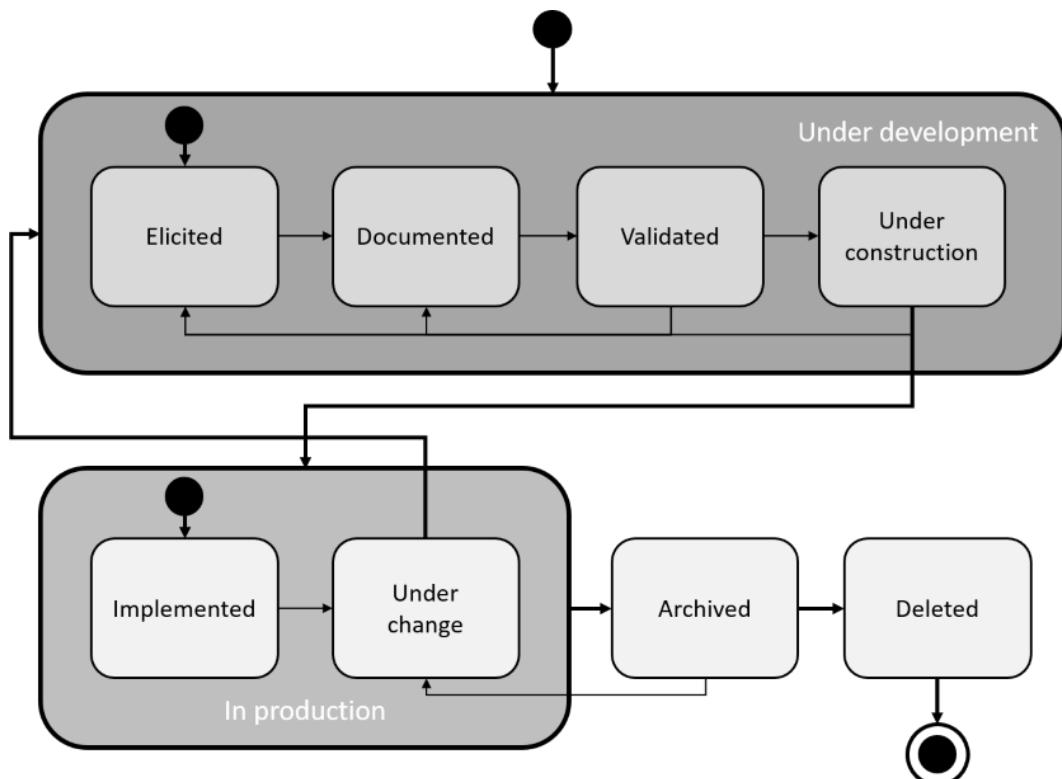


Figure 6.1 Simplified statechart of a requirements life cycle

A complicating factor is that work products and individual requirements have their own different life cycles that only partially overlap. As an example, think of a work product *definition study* in the state *under change*; this does not necessarily mean that all requirements contained in the work product have to be changed. And for the same *definition study*, the state *implemented* makes no sense; only some requirements in it will be implemented—or better: certain code, based on these requirements.

Another complicating factor may be that in practice, the view of the life cycle of requirements is different for different roles. For you as a Requirements Engineer, to trace your work you are interested in different states to the project manager, and other states again compared to the product manager or a change manager: in the diagram above, your interest might end at *validated*, while for the project manager, it only starts at *documented*.

Requirements Engineers actively manage the life cycle of their work products. Life cycle management implies:

- ▶ Defining life cycle models for your work products and the requirements contained in them with
 - The states that a work product or requirement can take
 - The transitions allowed between these states
 - The events that trigger the transition from one state to another
- ▶ Ensuring that only explicitly allowed transitions occur
- ▶ Recording the actual states that the work products and requirements take
- ▶ Recording the actual transitions that occur
- ▶ Reporting on these states and transitions

Life cycle management needs to be actively managed

In simple words: make sure that you know the state that your requirements were in, are in, and will take, how they can change, and why this all happens.

For instance, as a Requirements Engineer, you could be asked to report who approved which version of a requirement to be released as input for the coding phase. Keeping track of requirements states in their life cycle can also be useful for building dashboards and reporting on the progress of a project. It can be a good way to organize work and identify which requirements to work on first.

The state of a work product under life cycle management is often recorded in an attribute (see Section 6.5). It may also be useful to document the beginning and the end date of that state in attributes. In agile projects, the state of a work product (item) can be derived from its position in the product backlog, task backlog, and/or on the task board. Also, meeting the criteria of the *definition of ready* and the *definition of done* can give relevant information, as meeting these criteria actually means attaining a next state.

The thoroughness and level of detail of the life cycle management should be tailored to the needs of the customer, the project, and the system. For instance, the states *under development*, *in production*, and *archived* might be sufficient. In complex or critical projects, you may need a far more detailed model of the states, strict procedures about state transitions, and an audit trail that shows what happened during the project.

6.3 Version Control

It is common for both, work products and individual requirements as part of a work product, to undergo certain changes during their life cycle (see Section 6.7 for more information on handling these changes). After every change, the work product is different to what it was before: it has become a new version.

We want to control the versions of these work products for two reasons:

- ▶ Sometimes changes go wrong. After a while, defects are found, or the intended benefits are not realized. In such a case, we may implement new changes in a next version but we can also decide to go back to a previous version and continue from there. Or maybe, on second thought, we just prefer the earlier version after all.
- ▶ We want to know the history of the work product, understand its evolution right from its origin up to its present situation. This may help us when we have to decide on future changes, or just answer questions on why the current work product is what it is.

Reasons for version control of work products

Version control requires three measures to be in place:

- ▶ An *identification* of each version, to distinguish between the different versions of a work product. This is the version number, often supplemented with a version date.
- ▶ A clear description of each *change*. You must be able to tell—and understand—the difference between a certain version and its predecessor. This change description must be clearly linked to the version number.
- ▶ A strict policy on the *storage* of versions, enabling you to locate and retrieve old versions. Unless storage limitations dictate otherwise, you should preserve all previous versions of all your work products, otherwise you may not be able to restore a version if you need it. On the other hand, unlimited storage will rarely be the case, so it is wise to also have a policy for archiving and cleaning up work products that are no longer used.

Measures for version control

Usually, a work product contains multiple requirements. If a single requirement in that work product changes, both that requirement and the work product should get a new version number, while the unchanged requirements in that work product keep their old version number. This might soon become very confusing. A practical solution may be to do version numbering at the work product level only and let all requirements in it *inherit* the version number and the change history of the work product.

Version numbers are typically composed of (at least) two parts:

- ▶ *Version*. In principle, the version starts at *zero* as long as the work product is under development. When it is formally approved, released, and/or launched, we assign it version *one*. After that, the version is increased only for major, substantive updates.
- ▶ *Increment*. This mostly starts at *one* and is incremented with every (externally visible) change, on the content side or often only textual or editorial. An additional sub-increment may be used for correction of typos only. The increment *nine* is sometimes used to denote a final version just before approval or release.

Version numbers

A new version number is assigned with each formal change.

Often, a change in the life cycle state of a work product is not considered a reason for incrementing the version number, unless it is accompanied by a change in content or text. If, for instance, a requirement receives the state *validated* and the version number 1.0 after approval, there is no need to change this version number if the state changes to *under construction* and subsequently to *implemented*. The state can finally end in *archived* but still keep the same version number 1.0.

6.4 Configurations and Baselines

Suppose you preserve, as advised above, all versions of all requirements that you develop during a project. You will then have an ever-expanding database filled with requirements and you will start to lose the overview. One day, your client comes to your desk and asks: "We have implemented your system at all our branches. Now there seems to be a problem with the calculations in our Barcelona office. Can you tell me what version of the calculation requirements they use there?" If you cannot answer that question, you will wish that you had paid more attention to configuration management.

So, what is a configuration? You will find a definition in the CPRE glossary [Glin2020] but in short, for a Requirements Engineer, a configuration is a consistent set of logically related work products that contain requirements. We select this set with a specific purpose, usually to make clear which requirements are or were valid in a certain situation.

This sets the following properties for a correct configuration:

- ▶ *Logically connected.* The set of requirements in the configuration belongs together in view of a certain goal.
- ▶ *Consistent.* The set of requirements has no internal conflicts and can be integrated in a system.
- ▶ *Unique.* Both the configuration itself and its constituent requirements are clearly and uniquely identified.
- ▶ *Unchangeable.* The configuration is composed of selected requirements, each with a specific version that will never be changed in this configuration.
- ▶ *Basis for reset.* The configuration allows fallback to a previous configuration if any undesired changes appear to have occurred.

Configuration

Properties for Configurations

A configuration is documented as a work product, with a unique identification, a state, and a version number and date, just like any other work product. However, because a configuration is by definition unchangeable, it will always have only one version (e.g., 1.0).

A configuration always has two dimensions [CoWe1998]:

- ▶ The *product* dimension. This indicates which requirements are included in this specific configuration. Sometimes, a configuration will contain all available requirements but usually, it is a certain selection—for instance, all requirements that are implemented in the French release of a system. The British release of the same system might then have a different configuration.

Dimensions of Configurations

- The *version* dimension. In a specific configuration, every selected requirement is present in exactly one, and only one, version. It might be the latest version or an earlier one, depending on the purpose of the configuration itself. As soon as even a single different version of a single requirement is selected, this is a new configuration. Imagine a system for which a new release will be implemented with some requirements in a higher version: this new release will then have a different configuration.

Figure 6.2 gives another example of different configurations consisting of specific sets of versions of requirements.

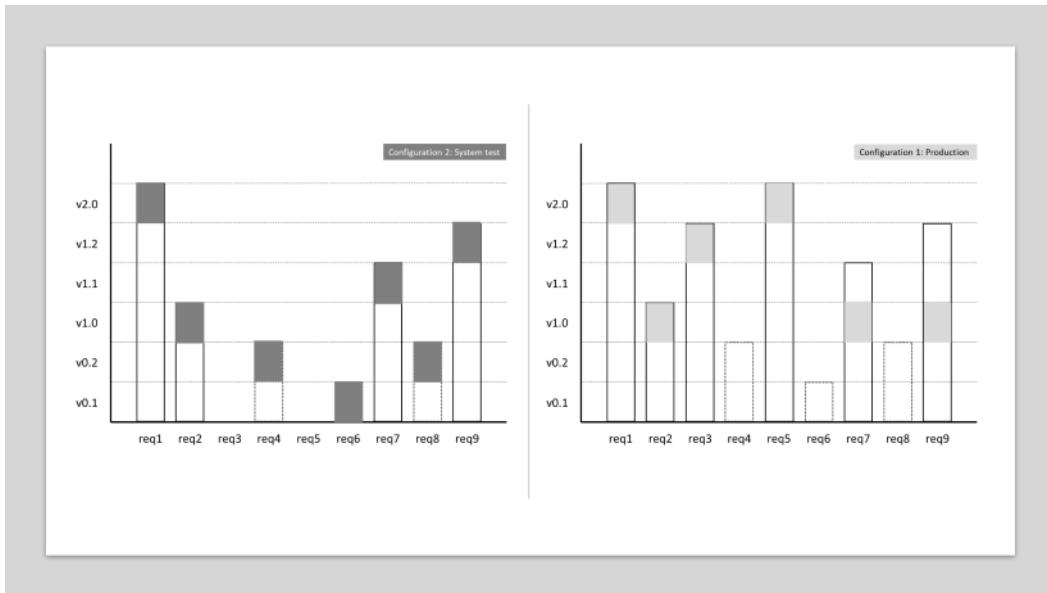


Figure 6.2 Example of configurations

The figure above shows an example of different configurations of a certain system. It shows a collection of nine requirements. Some of them are still in the early stages of development—e.g., requirement 6 with version v0.1. Other requirements have had more versions—for instance, requirement 1, which is finalized and has already had a major update, so is now version v2.0.

The left-hand picture shows the configuration that is currently in production. It consists of R1 v2.0, R2 v1.0, R3 v1.2 (this requirement had two minor updates after implementation), R5 v2.0, R7 v1.0, and R9 v1.0. R4, R6, and R8, being under development, are not present in this configuration, nor are the new versions of R7 and R9.

The right-hand picture shows the configuration that, at the same time, is present in the system test environment. Some requirements (R1, R2) are the same, some are no longer present (R3, R5), the requirements under development (R4, R6, and R8) are included here, and two requirements (R7 and R9) are present in a higher version than in the configuration of the production environment.

In many projects some configurations are treated in a special way: these configurations are called baselines. A *baseline* is a stable, validated, and change-controlled configuration that marks a milestone or another kind of *resting point* in the project. An example can be the configuration at the end of the design phase, just before starting the coding phase, or the configuration that is valid at the go-live of a certain release.

Baseline

The sprint backlog in an agile project serves as the baseline at the start of the next iteration. Baselines are useful for planning purposes as they represent a stable starting point for a next phase. They are often frozen and set aside as an anchor in the hectic life of a project. If something goes terribly wrong in the project, the team can perform a roll-back to the situation of the baseline and restart from there.

For the Requirements Engineer, it is mainly the configuration of work products containing requirements that is important. But in practice, the configuration within a project has a much broader scope, containing selected versions of the work products of all team members, such as requirements, designs, code and test cases. In complex projects, configuration management can be a full-time job, performed with dedicated tooling.

Not only requirements

6.5 Attributes and Views

As a Requirements Engineer, your output consists of all kinds of work products containing requirements. These requirements will have to be managed, otherwise you and your team will quickly lose the overview. To manage the requirements, you have to collect and maintain data about them—metadata, data about data. Metadata makes work products tangible, manageable; through metadata, you can provide and obtain information about the requirements and answer questions that are relevant during and after the project or product life cycle. Think of questions like "*Which requirements are planned for the next release?*" or "*How much effort is this release likely to take?*" or "*How many requirements have a high priority?*"

When considering the requirements as entities about which information is required, the characteristics of these requirements are called *attributes*. In this chapter, we have already seen some common attributes, such as the unique identification, version number, state, several dates. The attributes to be defined for the requirements depend on the information needs of the stakeholders of the project and the system. At the start of a project, an *attribute schema* should be set that enables the Requirements Engineer to fulfill these needs.

Attributes

A good starting point can be found in relevant standards. The ISO standard [ISO29148] mentions:

- ▶ *Identification*. Each requirement should have a unique, immutable identifier, such as a number, name, mnemonic. Without a proper identification, requirements management is impossible.
- ▶ *Stakeholder priority*. The (agreed) priority of the requirement from the viewpoint of the stakeholders. See Section 6.8 for information on how to determine this priority.
- ▶ *Dependency*. Sometimes, there is a dependency between requirements. This may mean that a low-priority requirement should be implemented first because another, high-priority requirement depends on it.
- ▶ *Risk*. This is about the potential that the implementation of the requirement will lead to problems, such as damage, extra costs, delays, legal claims. By nature, this is an estimate, to be based on consensus among stakeholders.
- ▶ *Source*. What is the origin of the requirement, where did it come from? You may need this information for validation, conflict resolution, modification, or deletion.
- ▶ *Rationale*. The rationale gives you the reason why the requirement is needed, the objectives of the stakeholders that should be fulfilled by it.

- ▶ *Difficulty*. This is an estimate of the effort needed to implement the requirement. It is needed for project planning and estimation.
- ▶ *Type*. This attribute indicates whether the requirement is a functional or a quality requirement or a constraint.

There are many ways to store this information. It may be contained in documents or stored in a spreadsheet or database, with the requirements as rows and their attributes as columns. In agile settings, requirements may be recorded on story cards, where the rubrics on the card are the attributes. As discussed in Chapter 7, requirements management tools should offer functionality for storing data about requirements and also reporting on them.

Attributes allow you to provide information about your work products and the requirements contained therein. The simplest way to do so is to produce a report with all the data on all the versions of all requirements. For anything but the simplest system, such a report will be useless as nobody will be able to oversee all the information because it is overwhelmingly complex. Therefore, you should adjust your reports based on the information needs of your target audiences. This is done by using *views* [Glin2020]. A view is an (often predefined) way to filter and sort the data on your work products, resulting in a report that shows precisely what the audience needs, no more, no less. A view is defined with the explicit purpose of delivering relevant information for a specific target group.

Views

We discern three types of views:

Types of views

- ▶ *Selective views*. These views give information on a deliberate selection of the requirements instead of all requirements. For example, a view on only the latest versions of the requirements, or all requirements with the state *validated*, or on the requirements with stakeholder priority *high*; the focus might be on a subsystem, or on the contrary to provide an abstract overview of the system through its high-level requirements only.
- ▶ *Projective views*. A projective view shows a selection from all data (attributes) of the requirements—for example, only the identification, the version number, and the name.
- ▶ *Aggregating views*. In an aggregating view, you will find summaries, totals, or averages, calculated from a set of requirements. An example would be the total number of requirements per department: e.g., 4 from Sales, 5 from Logistics.

Figure 6.3 gives an example of these types of views.

	Projective view (only 4 attributes)					
Selective view (only sales dept)	ID	Version	Name	Type	Source	Difficulty (1..5)
	1	2.0	Calculate	Functional	Sales	3
	2	1.0	Response	Quality	Sales	2
	3	1.2	VAT	Constraint	Sales	1
	4	0.2	Foreign VAT	Constraint	Sales	4
	5	2.0	Delivery date	Functional	Logistics	2
	6	0.1	Track & trace	Functional	Logistics	5
	7	1.1	Courier	Functional	Logistics	1
	8	0.2	Routing	Functional	Logistics	4
	9	1.2	Accessibility	Quality	Logistics	3

Aggregating view	Functional	5	Quality	2	Constraint	2
------------------	------------	---	---------	---	------------	---

Figure 6.3 Different types of views

In most cases, a combination of views is used—for instance, if you want to provide a list with the IDs, version numbers, names, and types (= projective) of all the requirements for the Sales department (= selective).

6.6 Traceability

Throughout this handbook, we have mentioned the topic of traceability [GoFi1994]. Without proper traceability, Requirements Engineering is hardly feasible, as you cannot do the following:

- ▶ Provide evidence that a certain requirement is satisfied
- ▶ Prove that a requirement has been implemented and by what means
- ▶ Show product compliance with applicable laws and standards
- ▶ Look for missing work products (e.g., find out whether test cases exist for all requirements)
- ▶ Analyze the effects of a change to requirements (see Section 6.7)

Traceability

In many cases, especially for safety-critical systems, process standards even explicitly demand the implementation of traceability.

There are three types of questions that can be answered with the aid of traceability (see also Figure 6.4):

Types of traceability

- ▶ *Backward traceability*: What was the origin of a certain requirement? Where was it found? Which sources (stakeholders, documents, other systems) were analyzed during elicitation?

Backward traceability is as well-known as pre-requirements specification traceability.

- ▶ **Forward traceability:** Where is this requirement used? Which deliverables (coded modules, test cases, procedures, manuals) are based on it?

Forward traceability is as well-known as post-requirements specification traceability.

- ▶ **Traceability between requirements:** Do other requirements depend on this requirement or vice versa (e.g., quality requirements related to a functional requirement)? Is the requirement a refinement of a higher-level requirement (e.g., an epic refined in a number of user stories, a user story detailed with a number of acceptance criteria)? How are they related?

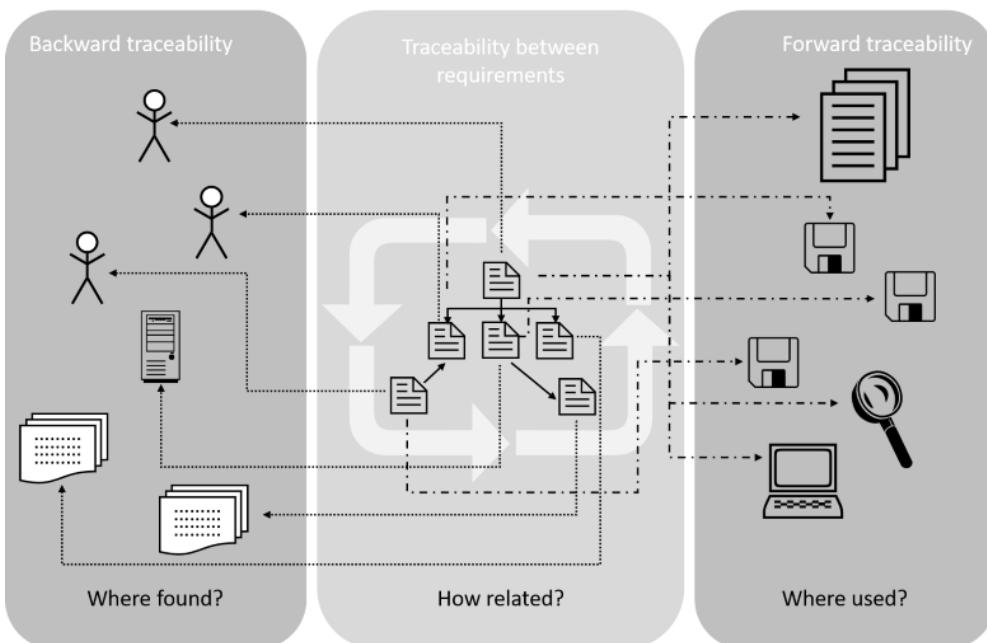


Figure 6.4 Traceability types

There are several ways of documenting traceability. Often, this is done *implicitly*—for instance, by applying document structures, standard templates, or naming conventions. If you identify all your requirements with the code *Req-xxx-nnn*, where *xxx* stands for the department that requested the requirement, everybody will understand that *Req-sal-012* is a requirement for the Sales department (for backward traceability). If you publish a document listing all the requirements that will be implemented in the release of July 1st, you are providing implicit forward traceability information. And if you write a document with a dedicated section on, e.g., price calculations, that could be an example of traceability between requirements. Another example could be a high-level model and a textual description of detailed requirements related to it.

In more complex projects, traceability should (also) be documented *explicitly*. For explicit traceability, you document the relationship between work products based on their unique identification. This can be done in various forms [HuJD2011]:

Documenting the relationship between work products

- Making use of specific attributes such as *Source* suggested by the ISO standard [ISO29148]
- In documents, adding references to predecessor documents, other work products, or individual requirements
- Developing a traceability matrix in a spreadsheet, or a database table (see an example in Table 6.1 below)
- In textual documentation, using Wiki-style hyperlinks
- Visualizing traceability relationships in a *trace graph* (Figure 6.4 is a simplified form of such a graph)
- In many cases, a requirements management or configuration management tool (see Chapter 7) provides functionality to support traceability. Managing traceability in a substantial project can be complicated, especially if you also have to take versioning into account. In such a case, good tooling is indispensable.

Table 6.1 Example of a traceability matrix

Source	R1	R2	R3	R4	R5	R6	R7
<i>Interview Mrs. Smith 06/08</i>	X	X			X		
<i>Summary questionnaire May 12</i>	X			X		X	X
<i>Field observation report 07/03</i>			X	X	X		
<i>Company regulations version 17.a.02</i>			X			X	X
<i>Documentation API HRM system v3.0.2.a</i>	X			X	X		

6.7 Handling Change

“Principle 2: Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.” [BeeA2001]. The founding fathers of the agile movement were crystal clear on this: requirement changes will always occur, whether you like them or not. Many people do not like changes at all, because every change is a risk, a threat to the stability of the project and the system.

Handling Change

However, changing a requirement is not a stand-alone event: it is triggered by changes in the system context, by new insights of the stakeholders, by behavior of competitors, and so on; a law becomes effective, adding a new constraint to the system; due to growing market demand, the performance of the system has to be improved; a competitor system is launched with some *delighter* features that your client wants too. A change should thus be seen as a chance to get a better system, to provide more value to the users.

However, regardless of the situation, every change is also a risk. It can introduce defects, leading to system failure. It can delay the progress of the project. It can take more effort and money than was calculated before. The users may not like it and refuse to work with it. In short, things can go wrong and disturb a previously stable project or system. But that does not mean that changes are bad and should be avoided; it does mean that all changes must be handled carefully to get optimal value at acceptable costs with minimal risk.

In the literature on IT service management (see [Axelos2019]), *change enablement* is described as one of the core practices. This practice ensures that changes are implemented effectively, safely, and in a timely manner in order to meet stakeholders' expectations. The practice balances effectiveness, throughput, compliance, and risk control. It focuses on three aspects:

- ▶ Ensuring that all risks have been accurately assessed
- ▶ Authorizing changes to proceed
- ▶ Managing the change implementation

Change enablement

Change enablement implies that an organization assigns a change authority to decide on the changes and defines a process for handling them. See Figure 6.5 for an outline of this process. These measures are usually tuned to the development approach and the point in time where a change occurs.

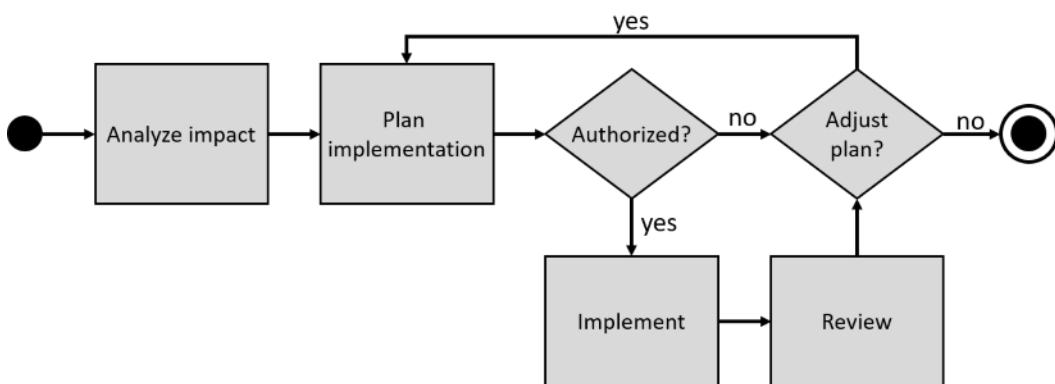


Figure 6.5 Change enablement process

As long as a requirement is in a draft state, the author has the authority to change it and no strict process is followed.

As soon as a requirement is released for further use in the project, the author is no longer free to decide, as every change will have an impact on other work products based on this requirement. Before deciding whether a change should be implemented, an impact analysis should be performed to clarify the efforts and risks of the change. This is where traceability is indispensable. In a *linear* development approach, the change authority will often be assigned to project management, a steering committee, or a *Change Control Board*, and a process is followed, with a formal decision on the change and the planning of its implementation. In an *iterative* development approach, the change authority usually lies with the product owner, who decides on the change and adds an accepted change to the product backlog as a new item (work product). The further implementation is then handled just like any other product backlog item.

Change authority

Once a requirement is implemented in an operational system, an even stricter process should be followed, as every change will now influence users and business processes.

Here, a distinction is often made between *standard* (low-risk, well understood, and pre-authorized, e.g., a change to the VAT percentage), *normal* (based on a formal Request for Change, scheduled, assessed, and authorized, e.g., a change to a price calculation algorithm), and *emergency* changes (to be implemented as soon as possible, e.g., to resolve an incident—but that seldomly involves a change of requirements). Usually, the change authority lies with a *Change Advisory Board* [Math2019]; in an iterative approach like DevOps, a change may be authorized by a release manager.

6.8 Prioritization

Requirements themselves are just concepts in the minds of people. They bring value only when they are implemented in an operational system. This implementation takes effort, time, money, and attention. In most cases, these resources are limited, which means that not all requirements can be implemented, at least not at the same time. This in turn means that the stakeholders have to decide which requirements should come first and which could be implemented later (or not at all). In other words: prioritization [Wieg1999].

Prioritization

The priority of a requirement is defined as the level of importance assigned to it according to certain criteria [Glin2020]. Consequently, you first have to determine what criteria should be used to assess the requirements before you can prioritize them. However, before you can determine the assessment criteria, you must know what the goal of the prioritization is. That goal is usually not your goal as a Requirements Engineer but the goal of certain stakeholders, so you must decide who the stakeholders are for this prioritization. And when you know their goal, it will usually be clear that not all requirements will have to be prioritized but rather only a defined subset.

Summarizing the above, we can outline a sequence of steps to be followed if we want to prioritize requirements:

Steps to be followed for prioritizing requirements

▶ Define major *goals and constraints* for the prioritization

Project and system context largely determine the reasons for prioritization. If, for instance, you prioritize to decide which features will be implemented in the next release, you might focus on business value; if the goal is to select user stories for the next iteration, story points and technical dependencies would be more prominent. Technical or legal constraints might limit the choices to be made.

▶ Define desired *assessment criteria*

In principle, the goals and constraints dictate the criteria to be used. Commonly used criteria are business value for stakeholders, urgency perceived by users, effort to implement, risks for usage, logical and technical dependencies, the legally binding nature of a requirement, or just the (inter-) subjective preference of relevant stakeholders. Sometimes only a single criterion is used but a balanced selection of several relevant criteria may yield a better outcome.

▶ Define the *stakeholders* that have to be involved

Goals and constraints influence which stakeholders you should involve in the prioritization but on the other hand, certain stakeholders themselves set these goals, so you must be aware of the interdependency. As an example, when prioritizing for the launch of a new system, you would probably invite business representatives and a panel of future customers. When prioritizing the product backlog to decide on the next iteration, the scrum team will be involved.

▶ Define the *requirements* that have to be prioritized

It is unlikely that the whole set of requirements has to be prioritized. Once again, this depends primarily on the goals and constraints for prioritizing. For instance, constraints may dictate certain requirements to be must-haves. In fact, it is only useful to prioritize requirements for which you have a choice whether or not to include them in a next step of the development process. This means that the project phase is also an important factor. In an early phase, you might include draft versions in the prioritization; in a late phase, you will often restrict prioritization to requirements that are in a stable version. Be aware that requirements to be prioritized should be at a comparable level of abstraction depending on the prioritization goals. In an early project phase, for instance, you might prioritize themes or features while prioritizing user stories at iteration planning.

▶ Select the prioritization *technique*

A prioritization technique is the way in which you prioritize the requirements. As described below, there are several techniques, which differ in effort, thoroughness, and level of detail. Here too, goals and constraints set the stage, but the most important factor is that the stakeholders involved agree on the technique that you intend to use. If not, they will not accept the outcome and your prioritization effort is in vain.

▶ Perform *prioritization*

When all preparation has been done, you can perform the actual prioritization. You assess all selected requirements on all defined criteria. Together with the stakeholders involved, you apply the selected technique to the criteria assessed. As a result, you get a prioritized list of requirements. However, there might be a problem. Different stakeholders might have different priorities, even if they agree on the criteria assessed. In that case, you typically have a requirements conflict that should be resolved just like any other conflict as described in Section 4.3 on conflict resolution.

Taking a closer look at prioritization techniques, we distinguish between two categories:

Categories of prioritization techniques

► *Ad hoc techniques*

With ad hoc techniques, experts assign priorities to the selected requirements based on their own experience. In principle, this prioritization is based on a single criterion, being the subjective perception of the expert. If this expertise is at a high level and acceptable to the stakeholders, such a technique can be a quick, cheap, and easy way to achieve prioritization. A variant would be to invite several experts and calculate some kind of average priorities. Common ad hoc techniques include Top-10 ranking and MoSCoW (Must have, Should have, Could have, Won't have this time) prioritization. Kano analysis (Section 4.2.1) is also useful: the dissatisfiers are must-haves, the satisfiers should-haves, and the delighters can be could- or won't-haves. For more background, see, for example, [McIn2016].

► *Analytical techniques*

Analytical techniques employ a systematic process for assigning priorities. In such techniques, experts assign weights to multiple assessment criteria (such as benefit, cost, risk, time to implement, etc.) and subsequently, requirements priorities are calculated as weighted outcomes based on these criteria. Such techniques take more effort and time but have the advantage of giving a clear insight into the factors that determine the priorities and into the process by which the priorities are established. This can stimulate the acceptance of the outcome among the stakeholders. However, two aspects must be kept in mind. First, the outcome is heavily influenced by the weight factors that are used in the calculation of the result. Therefore, an agreement among the stakeholders about these weight factors must be established before the actual prioritization. Otherwise, some might try to change the weight factors in order to manipulate the priorities. The second aspect to consider is that the criteria assessed are mostly estimates, not measured facts. And the estimates are often on a simple ordinal scale such as low, medium, high. Thus, the quality of the estimates is decisive for the quality of the resulting prioritization. Nevertheless, analytical techniques are useful for providing a clearly underpinned prioritization that is understood and thus accepted by the stakeholders involved. For a detailed explanation of analytical techniques, see [Olso2014].

It may be tempting to apply detailed, thorough techniques and spend a lot of time producing perfectly accurate estimates in terms of money, hours, expected sales numbers, etc. This could result in requirement A having a calculated priority of 22.76, requirement B of 23.12, and requirement C of 20.29. You would then conclude that evidently, C must be done first and A prior to B. However, you have probably just introduced a pseudo-accuracy with this calculation, and it would be better to conclude that those three requirements are equally important, which might have been your gut feeling right from the start. Always make sure that the effort you spend in prioritizing is justified by the value of a correct prioritization itself. So once again, keep the goals in mind and remember Principle 1: value orientation.

6.9 Further Reading

The textbooks by Pohl [Pohl2010], Davis [Davi2005], Hull, Jackson and Dick [HuJD2011], van Lamsweerde [vLam2009] and Wiegers and Beatty [WiBe2013] provide a comprehensive overview of requirements management. Additional insights to the topic of requirements management is consolidated in the CPRE Advanced Level handbook for Requirements Management by Bühne and Herrmann [BuHe2019].

Cleland-Huang, Gotel and Zisman [ClGZ2012] provide an in-depth treatment of traceability.

Olson [Olso2014] and Wiegers [Wieg1999] deal with prioritization techniques.

7. Tool Support

A Requirements Engineer needs tools to practice his craftsmanship properly—just as a carpenter needs his tools, pencil, a hammer, saw, and drill to design and realize a piece of furniture. Without tools, it is difficult or impossible to record the requirements, work together on the requirements, and be in control of the requirements.

This chapter examines the different types of Requirements Engineering (RE) tools available and the aspects that need to be taken into account to introduce Requirements Engineering tools into an organization.

7.1 Tools in Requirements Engineering

Requirements Engineering is a difficult task without the support of tools. Tools are needed to support Requirements Engineering tasks and activities. Existing tools focus on supporting specific tasks, such as documenting requirements or supporting the RE process, and rarely on all tasks and activities in the Requirements Engineering process. It is therefore not surprising that the Requirements Engineer must have a set of tools at his disposal to support the various components in the Requirements Engineering process—just as the carpenter needs several tools (e.g., computer-aided design (CAD)) to design a piece of furniture and needs tools like a saw, scraper, and sandpaper to realize it.

Necessary support for Requirements Engineering process

Tools are just an aid to the Requirements Engineering process and the Requirements Engineer, and such tools are called CASE (computer-aided software engineering) tools. CASE tools support a specific task in the software production process [Fugg1993].

We differentiate between different types of tools that support the following aspects of Requirements Engineering:

Different types of tools

➤ Management of requirements

Tools in this category have the properties needed to support the activities and topics described in Chapter 6. With these kinds of tools, more control can be established over the Requirements Engineering process.

Requirements are subject to change and in an environment where this happens frequently, a tool with the relevant properties is indispensable.

Tools in this category support:

- Definition and storage of requirements attributes to identify and collect data about work products and requirements as described in Section 6.5
- Facilitation and documentation of the prioritization of requirements (Section 6.8)
- Life cycle management, version control, configurations and baselines as described in Sections 6.2, 6.3, and 6.4
- Tracking and tracing of requirements, as well as defects in the requirements and work products (Section 6.6)
- Change management for requirements; as we learned in Section 6.7, changes are inevitable and have to be carefully managed

► Requirements Engineering process

To support the Requirements Engineering process, information is needed to allow the process to be adjusted or improved. This kind of tool can:

- Measure and report on the Requirements Engineering process and workflow
- This information helps to improve the Requirements Engineering process and reduces waste.
- Measure and report on the product quality
- This information helps to find defects and flaws, which in turn can be used to improve product quality.

► Documentation of knowledge about the requirements

The amount of knowledge (and requirements) built up in a project can be enormous. In addition, a large amount of knowledge is built up about a product during its life cycle. All the relevant information must be carefully documented to enable the following:

- Sharing and creation of a common understanding of the requirements
- Securing the requirements as a legal obligation
- An overview of and insight into the requirements

► Modeling of requirements

As we learned in Section 3.4.1.6, expressing requirements in both diagrams and natural language uses the strengths of both forms of notation. A tool that can model requirements allows you to:

- Structure your own thoughts; it can be used as an aid to thinking
- Specify the requirements in a more formal language than textual requirements, with all benefits that brings

► Collaboration in Requirements Engineering

When several people and disciplines work on the same project, a tool can support and enable this collaboration, especially in the world in which we now live, where more and more activities are performed locally (at home). This kind of tool supports the elicitation, documentation, and management of requirements.

► Testing and/or simulation of the requirements

Tools are becoming more and more sophisticated. More and more options are being developed for testing and/or simulating requirements in advance. This allows a better prediction of whether the proposed requirements will have the intended effect.

The tools available are often a mix of the above. As mentioned before, different tools may need to be combined to adequately support Requirements Engineering. If different tools are used, it is important to pay attention to the integration between them and the interaction with other applications and systems in order to ensure smooth operation.

Sometimes, other kinds of tools (for example, office or issue-tracking tools) are used, or rather, misused, to document or manage requirements. However, these tools have their limitations and should be used only when the Requirements Engineers and stakeholders are in control of the RE process and requirements are aligned. Otherwise, this is a major risk in the RE process, as such tools do not support any requirements management activities.

Misuse of tools can jeopardize the RE process

7.2 Introducing Tools

Selecting an RE tool is no different to selecting a tool for any other purpose. You should describe the objectives, context, and requirements before selecting and implementing the appropriate tool(s).

Introducing tools

Tools are just an aid to the Requirements Engineering process and the Requirements Engineer. They do not solve organizational or human issues. Imagine that, together with your colleagues, you want to document the requirements in a uniform manner. Tools can support this—for instance, with a template in a word processing tool or wiki page. This does not ensure that all your colleagues adopt this working method, neither does it ensure that your colleagues have the discipline to record and manage their requirements in this way. What can help is to make agreements with each other, to check whether the agreements are being fulfilled, and to be able to communicate with each other if agreements are not adhered to. A tool is not going to help you with this. Introducing a Requirements Engineering tool requires clear Requirements Engineering responsibilities and procedures.

A tool can help you to configure your Requirements Engineering process effectively and efficiently. Tools often provide a framework based on best practices experience. These frameworks can then be tailor-made to suit the situation.

As we have learned in the previous chapters, core Requirements Engineering activities are not stand-alone processes. Selecting the appropriate RE tools starts with the definition of the objectives and/or problems you want to solve in the RE process. The next step is to determine the context of the system (in this case, the tool set). Consider the aspects of the context—i.e., stakeholders, processes, events, etc., and apply your Requirements Engineering skills to specify the requirements for the RE tools. Practice what you preach.

The next sections describe some of the aspects that have to be taken into account when introducing a (new) Requirements Engineering tool into your organization.

7.2.1 Consider All Life Cycle Costs beyond License Costs

The most obvious costs, such as purchase costs or licensing costs, are usually factored in. In addition, less visible costs must also be taken into account, such as the use of resources in the implementation, operation, and maintenance of the tool.

Life cycle costs

7.2.2 Consider Necessary Resources

Specifying the requirements and supervising the selection process requires the necessary resources, in addition to the costs mentioned in the previous section. People necessary to guide the selection process, Requirements Engineers, hardware resources, and other resources should not be overlooked. After the tool has been put into use, resources may also be required for maintenance and user support.

Necessary resources

7.2.3 Avoid Risks by Running Pilot Projects

The introduction of a new tool can threaten the control over the current requirements base. A requirements chaos can arise because there is a transition from the old working method and/or tools to the new working method and tools. Introduction of a new tool during an existing project will irrevocably lead to a delay in the delivery of the requirements and even the project.

Running pilot projects

The introduction of a new tool, possibly with a different working method, should be tested on a small scale, where the risks and impact remain manageable. There are several ways to do this:

- ▶ Apply the tool to a non-critical project/system
- ▶ Use the tool redundantly alongside an existing project
- ▶ Apply the tool to a fictional situation/project
- ▶ Import/copy the requirements of a project that has already been completed

When you have reached the point where the tool meets the set goals and requirements, it can be rolled out more widely within the organization or other projects.

7.2.4 Evaluate the Tool according to Defined Criteria

Selecting the appropriate tool can be a difficult task. Extensive verification of whether the objectives and requirements are met is a standard approach in Requirements Engineering. A systematic approach that assesses the tool from different perspectives also contributes to making the right choice. The following perspectives can be considered:

Assessing tools systematically

- ▶ Project perspective

This point of view highlights the project management aspects. Does the tool support the project and the information required in the project?

- ▶ Process perspective

This perspective verifies the support of the Requirements Engineering process. Does the tool sufficiently support the RE process? Can it be sufficiently adapted to the existing RE process and working method?

- ▶ User perspective

This perspective verifies the degree of application by the users of the tool. This is an important view because if users are not satisfied with the tool, the risk of the tool not being accepted increases. Does the tool sufficiently support the authorization of users and groups? Is it sufficiently user-friendly and intuitive?

- ▶ Product perspective

The functionalities offered by the tool are verified from this angle. Are the requirements sufficiently covered by the tool? Where is the data stored? Is there a roadmap with the functional extensions for the tool? Is the tool still supported by the supplier for the time being?

- ▶ Supplier perspective

With this perspective the focus lies on the service and reliability of the supplier. Where is the supplier located? How is the support for this tool arranged?

- ▶ Economic perspective

This perspective looks at the business case: does the tool deliver sufficient benefits in relation to the costs? What are the (management) costs for the purchase and maintenance? What does the tool provide for the RE process? Is a (separate) maintenance contract required?

- ▶ Architecture perspective

This perspective assesses how the tool fits into the (IT) organization. Does the technology applied suit the organization? Can the tool be sufficiently linked with other systems? Does the tool fit into the IT landscape and does it comply with the architectural constraints?

7.2.5 Instruct Employees on the Use of the Tool

Once a tool has been selected, the users should become familiar with the opportunities the tool can add to the Requirements Engineering process. The users—i.e., the Requirements Engineers—should be trained in how to use the tool in the existing Requirements Engineering process. If the users are not sufficiently trained, this may mean that not all the benefits of the tool are used. In fact, it is possible that the tool will be used incorrectly, with all the associated consequences.

Instruct employees on the use of the tools

The Requirements Engineering process can also be changed due to the tool selected. Aspects in the Requirements Engineering process that were not possible before can be made possible with a new tool: for example, adequate version management, modeling of requirements, etc. This can mean that new procedures are agreed, templates are adapted or applied, changes are made to the working method, and so on. The involvement of the Requirements Engineer in this change contributes to the success of the tool's acceptance.

7.3 Further Reading

The following literature can be consulted for an overview of available tools and tool evaluations. Juan M. Carrillo de Gea et. al. provide a comprehensive overview of the role of Requirements Engineering tools [dGeA2011]. The article by Barbara Kitchenham, Stephen Linkman, David Law [KiLL1997] describes and validates a method for systematic tool evaluation. If you are searching for an RE tool, a comprehensive list of tools for Requirements Engineering is provided on the Volere website [Vole2020] or at [BiHe2020].

8. References

- [Alex2005] Ian F. Alexander: A Taxonomy of Stakeholders – Human Roles in System Development. *International Journal of Technology and Human Interaction* 2005, 1(1), 23–59.
- [AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Goal Decomposition and Scenario Analysis in Business Process Reengineering. CAiSE (Conference on Advanced Information Systems Engineering), 1994, 94–104.
- [Armo2004] Philip G. Armour. *The Laws of Software Process: A New Model for the Production and Management of Software*. Boca Raton, Fl.: CRC Press, 2004.
- [Axelos2019] Axelos: ITIL Foundation: ITIL 4 Edition. Axelos Ltd., 2019.
- [BaBo2014] Stéphane Badreau, Jean-Louis Boulanger: *Ingénierie des Exigences*. Paris: Dunod, 2014 (in French).
- [BeeA2001] Kent Beck et al.: Principles behind the Agile Manifesto. <http://agilemanifesto.org/principles.html>, 2001. Last visited August 2020.
- [BiHe2020] Andreas Birk, Gerald Heller: List of Requirements Management Tools. <https://makingofsoftware.com/resources/list-of-rm-tools/>, 2020, Last visited November 2020.
- [Boeh1981] Barry W. Boehm: *Software Engineering Economics*, Englewood Cliffs, New Jersey: Prentice Hall, 1981.
- [BoRJ2005] Grady Booch, James Rumbaugh, Ivar Jacobson: *The Unified Modeling Language User Guide*, 2nd edition. Reading, MA: Addison-Wesley, 2005.
- [Bour2009] Lynda Bourne: *Stakeholder Relationship Management - A Maturity Model for Organisational Implementation*. Farnham: Gower, 2009.
- [BuHe2019] Stan Bühne, Andrea Herrmann: *Handbook Requirements Management according to the IREB Standard – Education and Training for the IREB Certified Professional for Requirements Engineering Qualification Advanced Level Requirements Management*. Karlsruhe: IREB. <https://www.ireb.org/downloads/#cpire-advanced-level-requirements-management-handbook>, 2019. Last visited November 2020.
- [CaDJ2014] Dante Carrizo, Oscar Dieste, Natalia Juristo: Systematizing Requirements Elicitation Technique Selection. *Information and Software Technology* 2014, 56(6), 644–669.
- [Chen1976] Peter P.-S. Chen: The Entity-Relationship Model: Toward a Unified View of Data, *ACM Transactions on Database Systems* 1976, 1(1), 9–36.

- [ClGZ2012] Jane Cleland-Huang, Olly Gotel, Andrea Zisman (eds.): Software and Systems Traceability. London: Springer, 2012.
- [Cock2001] Alistair Cockburn: Writing Effective Use Cases. Boston: Addison-Wesley, 2001.
- [Cohn2004] Mike Cohn: User Stories Applied: For Agile Software Development. Boston: Addison-Wesley, 2004.
- [Cohn2010] Mike Cohn: Succeeding with Agile: Software Development Using Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.
- [CoWe1998] Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management. ACM Computing Surveys 1998, 30(2), 232–282.
- [DaTW2012] Marian Daun, Bastian Tenbergen, Thorsten Weyer: Requirements Viewpoint. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: Model-Based Engineering of Embedded Systems, Heidelberg: Springer, 2012.
- [Davi1993] Alan M. Davis: Software Requirements – Objects, Functions, and States. 2nd Edition, Englewood Cliffs, New Jersey: Prentice Hall, 1993.
- [Davi1995] Alan M. Davis: 201 Principles of Software Development. New York: McGraw-Hill, 1995.
- [Davi2005] Alan M. Davis: Just Enough Requirements Management - Where Software Development Meets Marketing. New York: Dorset House, 2005.
- [DeBo2005] Edward De Bono: De Bono's Thinking Course (Revised Edition), Barnes & Noble Books, 2005.
- [DeCo2007] Design Council: 11 Lessons: A Study of the Design Process. <https://www.designcouncil.org.uk/resources/report/11-lessons-managing-design-global-brands>, 2007. Last visited March 2020.
- [dGeA2011] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández-Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno: Requirements Engineering Tools. IEEE Software 2011, 28(4), 86–91.
- [DeMa1978] Tom DeMarco: Structured Analysis and System Specification. New York: Yourdon Press, 1978.
- [DIN66001] DIN 66001:1983-12: Information processing; graphical symbols and their application. Deutsches Institut für Normung e.V., Berlin, 1983 (in German).
- [Eber2014] Christof Ebert: Systematisches Requirements Engineering, 5. Auflage. Heidelberg: dpunkt 2014 (in German).
- [Fowl1996] Martin Fowler: Analysis Patterns: Reusable Object Models. Reading, MA: Addison-Wesley, 1996.
- [Fugg1993] Alfonso Fuggetta: A Classification of CASE Technology. IEEE Computer 1993, 26(12), 25–38.

- [GaWe1989] Donald C. Gause and Gerald M. Weinberg: Exploring Requirements: Quality before Design. New York: Dorset House, 1989.
- [GFPK2010] Tony Gorschek, Samuel Fricker, Kenneth Palm, and Steven A. Kunsman: A Lightweight Innovation Process for Software-Intensive Product Development. IEEE Software 2010, 27(1), 37–45.
- [GGJZ2000] Carl A. Gunter, Elsa L. Gunter, Michael Jackson, Pamela Zave: A Reference Model for Requirements and Specifications. IEEE Software 2000, 17(3), 37–43.
- [GHJV1994] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Pattern – Elements of Reusable Object-Oriented Software. Reading, Mass.: Addison-Wesley, 1994.
- [Gilb1988] Tom Gilb: Principles of Software Engineering Management. Reading, Mass.: Addison Wesley, 1988.
- [Glas1999] Friedrich Glasl: Confronting Conflict – A First-Aid Kit for Handling Conflict. Stroud, Gloucestershire: Hawthorn Press, 1999.
- [GlFr2015] Martin Glinz and Samuel A. Fricker: On Shared Understanding in Software Engineering: An Essay. Computer Science – Research and Development 2015, 30(3-4), 363–376.
- [Glin2007] Martin Glinz: On Non-Functional Requirements. 15th IEEE International Requirements Engineering Conference, Delhi, India, 2007, 21–26.
- [Glin2008] Martin Glinz: A Risk-Based, Value-Oriented Approach to Quality Requirements. IEEE Software 2008, 25(2), 34–41.
- [Glin2016] Martin Glinz: How Much Requirements Engineering Do We Need? Softwaretechnik-Trends 2016, 36(3), 19–21.
- [Glin2019] Martin Glinz: Requirements Engineering I. Course Notes, University of Zurich, 2019.
<https://www.ifi.uzh.ch/en/rerg/courses/archives/hs19/re-i.html#resources>. Last visited October 2020.
- [Glin2020] Martin Glinz: A Glossary of Requirements Engineering Terminology. Version 2.0. <https://www.ireb.org/downloads/#cpref-glossary>, 2020. Last visited October 2020.
- [GlWi2007] Martin Glinz and Roel Wieringa: Stakeholders in Requirements Engineering (Guest Editors' Introduction). IEEE Software 2007, 24(2), 18–20.
- [GoFi1994] Orlena Gotel, Anthony Finkelstein: An Analysis of the Requirements Traceability Problem. 1st International Conference on Requirements Engineering, Colorado Springs, 1994, 94–101.
- [GoRu2003] Rolf Goetz, Chris Rupp: Psychotherapy for System Requirements. 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), London, 2003, 75–80.

- [Gott2002] Ellen Gottesdiener: Requirements by Collaboration: Workshops for Defining Needs, Boston: Addison-Wesley Professional, 2002.
- [GreA2017] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitzá Guzmán, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, Melanie Stade: The Crowd in Requirements Engineering - The Landscape and Challenges. IEEE Software 2017, 34(2), 44–52.
- [Greg2016] Sarah Gregory: “It Depends”: Heuristics for “Common Enough” Requirements. Keynote speech at REFSQ 2016, Essen, Germany, 2016,
https://refsq.org/fileadmin/sse/external/refsq/refsq2016/user_upload/gregory_keynote.pdf. Last visited May 2020.
- [GRL2020] Goal oriented Requirement Language. University of Toronto, Canada
<https://www.cs.toronto.edu/km/GRL>. Last visited May 2020.
- [GrSe2005] Paul Grünbacher, Norbert Seyff: Requirements Negotiation. In A. Aurum, C. Wohlin (eds.): Engineering and Managing Software Requirements. Berlin: Springer, 2005, 143-162.
- [Hare1988] David Harel. On Visual Formalisms. Communications of the ACM 1988, 31(5), 514–530.
- [HoSch2020] Stefan Hofer, Henning Schwentner: Domain Storytelling — A Collaborative Modeling Method. Available from Leanpub,
<http://leanpub.com/domainstorytelling>. Last visited March 2020.
- [HuJD2011] Elizabeth Hull, Ken Jackson, Jeremy Dick: Requirements Engineering. 3rd ed., Berlin: Springer: 2011.
- [Hump2017] Aaron Humphrey: User Personas and Social Media Profiles. Persona Studies 2017, 3(2), 13–20.
- [IEEE830] IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998, 1998.
- [ISO19650] ISO 19650. Organization and Digitization of Information about Buildings and Civil Engineering Works, including Building Information Modelling (BIM)– Information Management Using Building Information Modelling – Part 1 and 2, 2018.
- [ISO5807] ISO/IEC/IEEE 1985-02: Information processing; Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts. International Organization for Standardization, Geneva, 1985.
- [ISO25010] ISO/IEC/IEEE 25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International Organization for Standardization, Geneva, 2011.
- [ISO29148] ISO/IEC/IEEE 29148: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. International Organization for Standardization, Geneva, 2018.

- [Jack1995] Michael Jackson: Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. New York: ACM Press, 1995.
- [Jack1995b] Michael Jackson: The World and the Machine. 17th International Conference on Software Engineering 1995 (ICSE 1995), 287–292.
- [Jaco1992] Ivar Jacobson: Object-oriented software engineering: a use case driven approach. New York: ACM Press, 1992.
- [JaSB2011] Ivar Jacobson, Ian Spence, Kurt Bittner: Use Case 2.0: The Guide to Succeeding with Use Cases. Ivar Jacobson International SA, 2011.
- [KiLL1997] Barbara Kitchenham, Stephen Linkman, David Law: DESMET: A Methodology for Evaluating Software Engineering Methods and Tools. Computing & Control Engineering Journal 1997, 8(3), 120–126.
- [KSTT1984] Noriaki Kano, Nobuhiko Seraku, Fumio Takahashi, Shinichi Tsuji: Attractive Quality and Must-Be Quality. Hinshitsu (Quality – Journal of the Japanese Society for Quality Control) 1984, 14(2), 39-48 (in Japanese).
- [Laue2002] Søren Lauesen: Software Requirements: Styles and Techniques. London: Addison-Wesley, 2002.
- [LaWE2001] Brian Lawrence, Karl Wieggers, and Christof Ebert: The Top Risks of Requirements Engineering. IEEE Software 2001, 18(6), 62–63.
- [LiOg2011] Jeanne Liedtka, Tim Ogilvie: Designing for Growth: A Design Thinking Tool Kit for Managers. New York: Columbia University Press, 2011.
- [LiSS1994] Odd I. Lindland, Guttorm Sindre, Arne Sølverg: Understanding Quality in Conceptual Modeling. IEEE Software 1994, 11(2), 42–49.
- [LiSZ1994] Horst Licher, Matthias Schneider-Hufschmidt, Heinz Züllighoven: Prototyping in Industrial Software Projects – Bridging the Gap Between Theory and Practice. IEEE Transactions on Software Engineering 1994, 20(11), 825–832.
- [LiQF2010] Soo Ling Lim, Daniele Quercia, Anthony Finkelstein: StakeNet: Using Social Networks to Analyse the Stakeholders of Large-Scale Software Projects. 32nd International Conference on Software Engineering (ICSE 2010), 2010, 295–304.
- [MaGR2004] Neil Maiden, Alexis Gizikis, Suzanne Robertson: Provoking Creativity: Imagine What Your Requirements Could Be Like. IEEE Software 2004, 21(5), 68–75.
- [Math2019] Joseph Mathenge: Change Control Board vs Change Advisory Board: What's the Difference? <https://www.bmc.com/blogs/change-control-board-vs-change-advisory-board>, Nov. 22, 2019. Last visited August 2020.

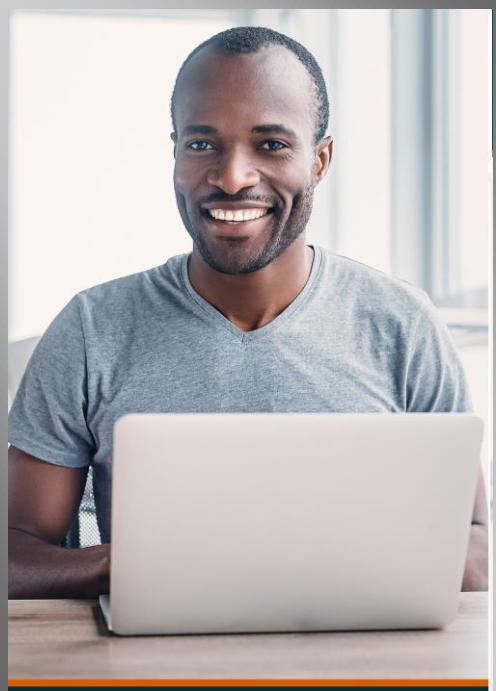
- [McIn2016] John McIntyre: MoSCoW or Kano Models – How Do You Prioritize? <https://www.hotpmo.com/management-models/moscow-kano-prioritize>, Oct. 20, 2016. Last visited August 2020.
- [MNJR2016] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe: Toward Data-Driven Requirements Engineering. IEEE Software 2016, 33(1), 48–54.
- [Moor2014] Christopher W. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts, 4th edition. Hoboken, NJ: John Wiley & Sons, 2014.
- [MWHN2009] Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak: Easy Approach to Requirements Syntax (EARS). 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, Georgia, 2009, 317–322.
- [NuKF2003] Bashar Nuseibeh, Jeff Kramer, Anthony Finkelstein: ViewPoints: Meaningful Relationships are Difficult! 25th International Conference on Software Engineering (ICSE'03), Portland, Oregon, 2003, 676–681.
- [OleA2018] K. Olsen et al.: Certified Tester, Foundation Level Syllabus - Version 2018. International Software Testing Qualifications Board, 2018.
- [Olso2014] David Olson: Matrix Prioritization. <http://www.bawiki.com/wiki/Matrix-Prioritization.html>, 2014. Last visited August 2020.
- [OMG2013] Object Management Group: Business Process Model and Notation (BPMN), version 2.0.2. OMG document, formal/2013-12-09. <https://www.omg.org/spec/BPMN/>. Last visited February 2020.
- [OMG2017] Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5.1. OMG document, formal/2017-12-05. <https://www.omg.org/spec/UML/About-UML/>. Last visited February 2020.
- [OMG2018] Object Management Group: OMG Systems Modeling Language (OMG SysML™), version 1.6. OMG document, ptc/2018-12-08. <https://www.omg.org/spec/SysML/About-SysML/> Last visited October 2020.
- [Osbo1948] Alex F. Osborn: Your Creative Power: How to Use Imagination. C. Scribner's Sons, 1948. (Accessed as digital reprint: Read Books Ltd. (epub eBook), April 2013).
- [Pich2010] Roman Pichler: Agile Product Management with Scrum – Creating Products that Customers Love, Boston: Addison-Wesley, 2010.
- [Pohl2010] Klaus Pohl: Requirements Engineering: Fundamentals, Principles, and Techniques. Berlin-Heidelberg: Springer, 2010.
- [PoRu2015] Klaus Pohl, Chris Rupp: Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam, (2nd ed). Rocky Nook, Santa Barbara, 2015.

- [Rein1997] Donald G. Reinertsen: Managing the Design Factory – A Product Developer’s Toolkit. The Free Press, 1997.
- [Rein2009] Donald G. Reinertsen: The Principles of Product Development Flow: Second Generation Lean Product Development. Redondo Beach, Ca.: Celeritas Publishing, 2009.
- [Ries2011] Eric Ries: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Business, 2011.
- [Robe2001] S. Ian Robertson: Problem Solving. Hove, East Sussex: Psychology Press, 2001.
- [RoRo2012] Suzanne Robertson and James Robertson: Mastering the Requirements Process: Getting Requirements Right. 3rd edition. Boston: Addison-Wesley, 2012.
- [RuJB2004] James Rumbaugh, Ivar Jacobson, Grady Booch: The Unified Modeling Language Reference Manual, 2nd edition. Reading, MA: Addison Wesley, 2004.
- [Rupp2014] Chris Rupp: Requirements-Engineering und Management, 6. Auflage. München: Hanser, 2014 (in German).
- [SoSa1998] Ian Sommerville and Pete Sawyer: Requirements Engineering: A Good Practice Guide. Chichester: John Wiley & Sons, 1997.
- [SwBa1982] William Swartout and Robert Balzer: On the Inevitable Intertwining of Specification and Implementation. Communications of the ACM 1982, 25(7), 438–440.
- [Verd2014] Dave Verduyn: Discovering the Kano Model, in: Kano model, <https://www.kanomodel.com/discovering-the-kano-model>, 2014. Last visited March 2020.
- [vLam2009] Axel van Lamsweerde: Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009.
- [Vole2020] Volere Requirements Resources: <https://www.volere.org>. Last visited June 2020.
- [WiBe2013] Karl Wiegers and Joy Beatty: Software Requirements. 3rd edition. Redmond, Wa.: Microsoft Press, 2013.
- [Wieg1999] Karl E. Wiegers: First Things First: Prioritizing Requirements. <https://www.processimpact.com/articles/prioritizing.pdf>, 1999. Last visited August 2020.
- [ZoCo2005] Didar Zowghi, Chad Coulin: Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum, C. Wohlin (eds.) Engineering and Managing Software Requirements. Berlin: Springer, 2005, 19–46.

PREPARATION GUIDE FOR CBAP®

Based on IIBA BABOK 3.0

CBAP®



“Only book you
ever need to
pass CBAP in 1st
attempt.”

LEARN. LEAD. SUCCEED.

VELLICATE TECHNOLOGIES

CONTENTS

1.	Introduction.....	12
1.1	Purpose of the BABOK® Guide.....	12
1.2	What is Business Analysis?	12
1.3	Who is a Business Analyst?	13
1.4	Structure of the BABOK® Guide	13
	Tasks.....	15
	Underlying Competencies.....	16
	Techniques.....	17
	Perspectives.....	17
1.5	ECBA, CCBA, CBAP Exam Blueprint	19
	ECBA, CCBA, CBAP Eligibility Requirements	20
	ECBA, CCBA, CBAP Application and Exam Fee	20
2.	Business Analysis Key Concepts	22
2.1	The Business Analysis Core Concept Model	22
2.2	Key Terms.....	24
2.3	Requirements Classification Schema	25
2.4	Stakeholders.....	26
2.5	Requirements and Designs	29
3.	Business Analysis Planning and Monitoring	31
	The Core Concept Model in Business Analysis Planning and Monitoring.....	31
3.1	Plan Business Analysis Approach.....	32
3.1.1	Purpose.....	32
3.1.2	Description	32
3.1.3	Inputs.....	32
3.1.4	Elements	33
3.1.5	Guidelines and Tools	36
3.1.5	Techniques	37
3.1.6	Stakeholders.....	38
3.1.7	Outputs	38
3.2	Plan Stakeholder Engagement	38
3.2.1	Purpose	38

3.2.2	Description	38
3.2.3	Inputs	39
3.2.4	Elements	39
3.2.5	Guidelines and Tools	42
3.2.6	Techniques	42
3.2.7	Stakeholders	43
3.2.8	Outputs	43
3.3	Plan Business Analysis Governance	43
3.3.1	Purpose	43
3.3.2	Description	43
3.3.3	Inputs	44
3.3.4	Elements	44
3.3.5	Guidelines and Tools	46
3.3.6	Techniques	47
3.3.7	Stakeholders	47
3.3.8	Outputs	47
3.4	Plan Business Analysis Information Management	48
3.4.1	Purpose	48
3.4.2	Description	48
3.4.3	Inputs	48
3.4.4	Elements	49
3.4.5	Guidelines and Tools	51
3.4.6	Techniques	51
3.4.7	Stakeholders	52
3.4.8	Outputs	52
3.5	Identify Business Analysis Performance Improvements	52
3.5.1	Purpose	52
3.5.2	Description	52
3.5.3	Inputs	53
3.5.4	Elements	53
3.5.5	Guidelines and Tools	54
3.5.6	Techniques	55
3.5.7	Stakeholders	55

3.5.8	Outputs	56
4.	Elicitation and Collaboration	57
	The Core Concept Model in Elicitation and Collaboration	58
4.1	Prepare for Elicitation	58
4.1.1	Purpose	58
4.1.2	Description	58
4.1.3	Inputs	59
4.1.4	Elements	59
4.1.5	Guidelines and Tools	61
4.1.6	Techniques	61
4.1.7	Stakeholders	62
4.1.8	Outputs	62
4.2	Conduct Elicitation	62
4.2.1	Purpose	62
4.2.2	Description	62
4.2.3	Inputs	63
4.2.4	Elements	63
4.2.5	Guidelines and Tools	64
4.2.6	Techniques	64
4.2.7	Stakeholders	65
4.2.8	Outputs	66
4.3	Confirm Elicitation Results	66
4.3.1	Purpose	66
4.3.2	Description	66
4.3.3	Inputs	66
4.3.4	Elements	67
4.3.5	Guidelines and Tools	67
4.3.6	Techniques	67
4.3.7	Stakeholders	68
4.3.8	Outputs	68
4.4	Communicate Business Analysis Information	68
4.4.1	Purpose	68
4.4.2	Description	68

4.4.3	Inputs	69
4.4.4	Elements	69
4.4.5	Guidelines and Tools	70
4.4.6	Techniques	71
4.4.7	Stakeholders.....	71
4.4.8	Outputs	71
4.5	Manage Stakeholder Collaboration.....	71
4.5.1	Purpose	71
4.5.2	Description.....	72
4.5.3	Inputs	72
4.5.4	Elements	73
4.5.5	Guidelines and Tools	74
4.5.6	Techniques	74
4.5.7	Stakeholders.....	75
4.5.8	Outputs	75
5.	Requirements Life Cycle Management.....	76
5.1	Trace Requirements	78
5.1.1	Purpose	78
5.1.2	Description	78
5.1.3	Inputs	79
5.1.4	Elements	80
5.1.5	Guidelines and Tools	81
5.1.6	Techniques	81
5.1.7	Stakeholders.....	81
5.1.8	Outputs	82
5.2	Maintain Requirements	82
5.2.1	Purpose	82
5.2.2	Description	82
5.2.3	Inputs	82
5.2.4	Elements	83
5.2.5	Guidelines and Tools	84
5.2.6	Techniques	84
5.2.7	Stakeholders.....	84

5.2.8	Outputs	85
5.3	Prioritize Requirements.....	85
5.3.1	Purpose	85
5.3.2	Description	85
5.3.3	Inputs	85
5.3.4	Elements	86
5.3.5	Guidelines and Tools	87
5.3.6	Techniques	88
5.3.7	Stakeholders.....	88
5.3.8	Outputs.....	89
5.4	Assess Requirements Changes	89
5.4.1	Purpose	89
5.4.2	Description	89
5.4.3	Inputs	89
5.4.4	Elements	90
5.4.5	Guidelines and Tools	91
5.4.6	Techniques	91
5.4.7	Stakeholders.....	92
5.4.8	Outputs.....	92
5.5	Approve Requirements	93
5.5.1	Purpose	93
5.5.2	Description	93
5.5.3	Inputs	93
5.5.4	Elements	94
5.5.5	Guidelines and Tools	94
5.5.6	Techniques	95
5.5.7	Stakeholders.....	95
5.5.8	Outputs.....	96
6.	Strategy Analysis.....	97
6.1	Analyze Current State	99
6.1.1	Purpose	99
6.1.2	Description	99
6.1.3	Inputs	99

6.1.4	Elements	100
6.1.5	Guidelines and Tools	103
6.1.6	Techniques	103
6.1.7	Stakeholders.....	105
6.1.8	Outputs.....	105
6.2	Define Future State	105
6.2.1	Purpose.....	106
6.2.2	Description.....	106
6.2.3	Inputs.....	107
6.2.4	Elements	107
6.2.5	Guidelines and Tools	111
6.2.6	Techniques	111
6.2.7	Stakeholders.....	112
6.2.8	Outputs.....	113
6.3	Assess Risks	113
6.3.1	Purpose.....	113
6.3.2	Description.....	113
6.3.3	Inputs.....	114
6.3.4	Elements	115
6.3.5	Guidelines and Tools	116
6.3.6	Techniques	117
6.3.7	Stakeholders.....	117
6.3.8	Outputs.....	118
6.4	Define Change Strategy	118
6.4.1	Purpose.....	118
6.4.2	Description.....	118
6.4.3	Inputs.....	119
6.4.4	Elements	119
6.4.5	Guidelines and Tools	122
6.4.6	Techniques	122
6.4.7	Stakeholders.....	123
6.4.8	Outputs.....	124
7.	Requirements Analysis and Design Definition	125

7.1	Specify and Model Requirements	127
7.1.1	Purpose	127
7.1.2	Description	127
7.1.3	Inputs	127
7.1.4	Elements	128
7.1.5	Guidelines and Tools	129
7.1.6	Techniques	130
7.1.7	Stakeholders.....	131
7.1.8	Outputs.....	131
7.2	Verify Requirements	131
7.2.1	Purpose	131
7.2.2	Description	131
7.2.3	Inputs	132
7.2.4	Elements	132
7.2.5	Guidelines and Tools	133
7.2.6	Techniques	133
7.2.7	Stakeholders.....	133
7.2.8	Outputs.....	134
7.3	Validate Requirements.....	134
7.3.1	Purpose	134
7.3.2	Description	134
7.3.3	Inputs	134
7.3.4	Elements	135
7.3.5	Guidelines and Tools	136
7.3.6	Techniques	136
7.3.7	Stakeholders.....	136
7.3.8	Outputs.....	136
7.4	Define Requirements Architecture	137
7.4.1	Purpose	137
7.4.2	Description	137
7.4.3	Inputs	137
7.4.4	Elements	138
7.4.5	Guidelines and Tools	140

7.4.6	Techniques	140
7.4.7	Stakeholders.....	140
7.4.8	Outputs.....	141
7.5	Define Design Options	141
7.5.1	Purpose.....	141
7.5.2	Description.....	141
7.5.3	Inputs.....	141
7.5.4	Elements	142
7.5.5	Guidelines and Tools	143
7.5.6	Techniques	144
7.5.7	Stakeholders.....	144
7.5.8	Outputs.....	144
7.6	Analyze Potential Value and Recommend Solution	145
7.6.1	Purpose.....	145
7.6.2	Description.....	145
7.6.3	Inputs.....	145
7.6.4	Elements	146
7.6.5	Guidelines and Tools	148
7.6.6	Techniques	148
7.6.7	Stakeholders.....	149
7.6.8	Outputs.....	149
8.	Solution Evaluation	150
	The Core Concept Model in Solution Evaluation.....	151
8.1	Measure Solution Performance.....	152
8.1.1	Purpose	152
8.1.2	Description	152
8.1.3	Inputs.....	152
8.1.4	Elements	153
8.1.5	Guidelines and Tools	154
8.1.6	Techniques	154
8.1.7	Stakeholders.....	155
8.1.8	Outputs.....	155
8.2	Analyze Performance Measures	155

8.2.1	Purpose	155
8.2.2	Description	156
8.2.3	Inputs	156
8.2.4	Elements	157
8.2.5	Guidelines and Tools	158
8.2.6	Techniques	158
8.2.7	Stakeholders	159
8.2.8	Outputs	159
8.3	Assess Solution Limitations	159
8.3.1	Purpose	159
8.3.2	Description	159
8.3.3	Inputs	159
8.3.4	Elements	160
8.3.5	Guidelines and Tools	161
8.3.6	Techniques	161
8.3.7	Stakeholders	162
8.3.8	Outputs	162
8.4	Assess Enterprise Limitations	162
8.4.1	Purpose	162
8.4.2	Description	162
8.4.3	Inputs	163
8.4.4	Elements	164
8.4.5	Guidelines and Tools	165
8.4.6	Techniques	166
8.4.7	Stakeholders	167
8.4.8	Outputs	167
8.5	Recommend Actions to Increase Solution Value	167
8.5.1	Purpose	167
8.5.2	Description	167
8.5.3	Inputs	168
8.5.4	Elements	168
8.5.5	Guidelines and Tools	169
8.5.6	Techniques	170

8.5.7	Stakeholders.....	170
8.5.8	Outputs.....	171
9.	Underlying Competencies.....	172
9.1	Analytical Thinking and Problem Solving	172
9.1.1.	Creative Thinking.....	173
9.1.2.	Decision Making	173
9.1.3.	Learning	173
9.1.4.	Problem Solving.....	174
9.1.5.	Systems Thinking	174
9.1.6.	Conceptual Thinking	174
9.1.7.	Visual Thinking.....	175
9.2	Behavioural Characteristics.....	175
9.2.1	Ethics.....	176
9.2.2	Personal Accountability	176
9.2.3	Trustworthiness	177
9.2.4	Organization and Time Management	178
9.2.5	Adaptability	179
9.3	Business Knowledge	180
9.3.1	Business Acumen.....	180
9.3.2	Industry Knowledge.....	181
9.3.3	Organization Knowledge	183
9.3.4	Solution Knowledge	183
9.3.5	Methodology Knowledge	184
9.4	Communication Skills	184
9.4.1	Verbal Communication	185
9.4.2	Non-Verbal Communication.....	186
9.4.3	Written Communication	187
9.4.4	Listening.....	187
9.5	Interaction Skills	188
9.5.1	Facilitation.....	188
9.5.2	Leadership and Influencing	189
9.5.3	Teamwork.....	190
9.5.4	Negotiation and Conflict Resolution	191

9.5.5	Teaching	192
9.6	Tools and Technology	192
9.6.1	Office Productivity Tools and Technology	193
9.6.2	Business Analysis Tools and Technology	193
9.6.3	Communication Tools and Technology.....	193
10.	Techniques	195

1. Introduction

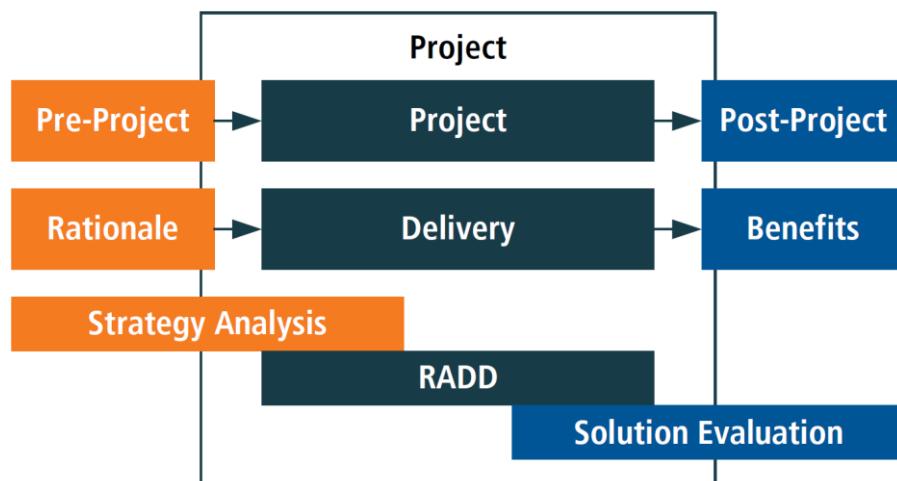
A *Guide to the Business Analysis Body of Knowledge® (BABOK® Guide)* is a globally recognized standard for the practice of business analysis. The *BABOK® Guide* describes business analysis knowledge areas, tasks, underlying competencies, techniques and perspectives on how to approach business analysis.

1.1 PURPOSE OF THE BABOK® GUIDE

The primary purpose of the *BABOK® GUIDE* is to define the profession of business analysis and provide a set of commonly accepted practices. It helps practitioners discuss and define the skills necessary to effectively perform business analysis work. The *BABOK® GUIDE* also helps people who work with and employ business analysts to understand the skills and knowledge they should expect from a skilled practitioner.

The six knowledge areas of the *BABOK® GUIDE* (Business Analysis Planning and Monitoring, Elicitation and Collaboration, Requirements Life Cycle Management, Strategy Analysis, Requirements Analysis and Design Definition, and Solution Evaluation) describe the practice of business analysis as it is applied within the boundaries of a project or throughout enterprise evolution and continuous improvement. The following image shows how three of the knowledge areas support the delivery of business value before, during, and after the life cycle of a project.

Figure 1.1.1: Business Analysis Beyond Projects



1.2 WHAT IS BUSINESS ANALYSIS?

Business analysis is the practice of enabling change in an enterprise by defining needs and recommending solutions that deliver value to stakeholders. Business analysis enables an

enterprise to articulate needs and the rationale for change, and to design and describe solutions that can deliver value.

Business analysis is performed on a variety of initiatives within an enterprise. Initiatives may be strategic, tactical, or operational. Business analysis may be performed within the boundaries of a project or throughout enterprise evolution and continuous improvement. It can be used to understand the current state, to define the future state, and to determine the activities required to move from the current to the future state.

Business analysis can be performed from a diverse array of perspectives. The BABOK®GUIDE describes several of these perspectives: agile, business intelligence, information technology, business architecture, and business process management. A perspective can be thought of as a lens through which the business analysis practitioner views their work activities based on the current context. One or many perspectives may apply to an initiative, and the perspectives outlined in the BABOK® Guide do not represent all the contexts for business analysis or the complete set of business analysis disciplines.

1.3 WHO IS A BUSINESS ANALYST?

A business analyst is any person who performs business analysis tasks described in the BABOK® Guide, no matter their job title or organizational role. Business analysts are responsible for discovering, synthesizing, and analyzing information from a variety of sources within an enterprise, including tools, processes, documentation, and stakeholders. The business analyst is responsible for eliciting the actual needs of stakeholders—which frequently involves investigating and clarifying their expressed desires—in order to determine underlying issues and causes.

1.4 STRUCTURE OF THE BABOK® GUIDE

The core content of the BABOK® GUIDE is composed of business analysis tasks organized into knowledge areas. Knowledge areas are a collection of logically (but not sequentially) related tasks. These tasks describe specific activities that accomplish the purpose of their associated knowledge area.

The Business Analysis Key Concepts, Underlying Competencies, Techniques, and Perspectives sections form the extended content in the BABOK® GUIDE that helps guide business analysts to better perform business analysis tasks. Thus, BABOK contains:

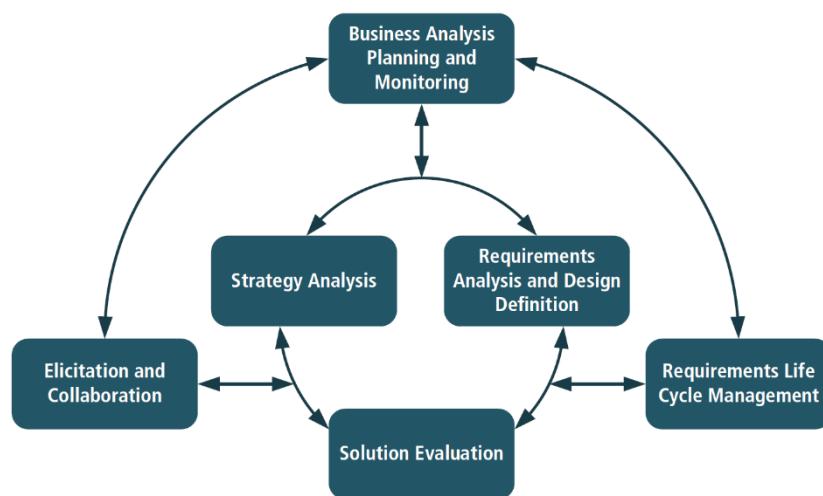
- **Business Analysis Key Concepts:** define the key terms needed to understand all other content, concepts, and ideas within the BABOK® GUIDE. This chapter consists of:
 - Business Analysis Core Concept Model™ (BACCM™)
 - Key Terms
 - Requirements Classification Schema
 - Stakeholders
 - Requirements and Design
- **Business Analysis Planning and Monitoring (BAPM):** describes the tasks that business analysts perform to organize and coordinate the efforts of business analysts and stakeholders. These tasks produce outputs that are used as key inputs and guidelines for the other tasks throughout the BABOK® GUIDE.
- **Elicitation and Collaboration (E&C):**describes the tasks that business analysts perform to prepare for and conduct elicitation activities and confirm the results obtained. It also

describes the communication with stakeholders once the business analysis information is assembled and the ongoing collaboration with them throughout the business analysis activities.

- **Requirements Life Cycle Management (RLCM):** describes the tasks that business analysts perform in order to manage and maintain requirements and design information from inception to retirement. These tasks describe establishing meaningful relationships between related requirements and designs, and assessing, analyzing and gaining consensus on proposed changes to requirements and designs.
- **Strategy Analysis (SA):** describes the business analysis work that must be performed to collaborate with stakeholders in order to identify a need of strategic or tactical importance (the business need), enable the enterprise to address that need, and align the resulting strategy for the change with higher- and lower-level strategies.
- **Requirements Analysis and Design Definition (RADD):** describes the tasks that business analysts perform to structure and organize requirements discovered during elicitation activities, specify and model requirements and designs, validate and verify information, identify solution options that meet business needs, and estimate the potential value that could be realized for each solution option. This knowledge area covers the incremental and iterative activities ranging from the initial concept and exploration of the need through the transformation of those needs into a particular recommended solution.
- **Solution Evaluation(SE):** describes the tasks that business analysts perform to assess the performance of and value delivered by a solution in use by the enterprise, and to recommend removal of barriers or constraints that prevent the full realization of the value.

The six knowledge areas are interrelated with each other and their relationship is depicted in following diagram:

Figure 1.4.1: Relationships Between Knowledge Areas



- **Underlying Competencies:** provide a description of the behaviours, characteristics, knowledge, and personal qualities that support the effective practice of business analysis.
- **Techniques:** provide a means to perform business analysis tasks. The techniques described in the BABOK® GUIDE are intended to cover the most common and widespread techniques practiced within the business analysis community.

- **Perspectives:** describe various views of business analysis. Perspectives help business analysts working from various points of view to better perform business analysis tasks, given the context of the initiative
-

TASKS

A task is a discrete piece of work that may be performed formally or informally as part of business analysis. The BABOK® GUIDE defines a list of business analysis tasks. The definition of a given task is universally applicable to business analysis efforts, independent of the initiative type. A business analyst may perform other activities as assigned by their organization, but these additional activities are not considered to be part of the business analysis profession.

Tasks are grouped into knowledge areas. Business analysts perform tasks from all knowledge areas sequentially, iteratively, or simultaneously. The BABOK® GUIDE does not prescribe a process or an order in which tasks are performed. Tasks may be performed in any order, as long as the necessary inputs to a task are present. A business analysis initiative may start with any task, although likely candidates are Analyze Current State or Measure Solution Performance.

Each task in the BABOK® GUIDE is presented in the following format:

- Purpose
 - Description
 - Inputs
 - Elements
 - Guidelines/Tools
 - Techniques
 - Stakeholders
 - Outputs
-

.1 PURPOSE

The Purpose section provides a short description of the reason for a business analyst to perform the task, and the value created through performing the task.

.2 DESCRIPTION

The Description section explains in greater detail what the task is, why it is performed, and what it should accomplish.

.3 INPUTS

The Inputs section lists the inputs for the task. Inputs are information consumed or transformed to produce an output, and represent the information necessary for a task to begin. They may be explicitly generated outside the scope of business analysis or generated by a business analysis task. Inputs that are generated outside of the business analysis efforts are identified with the qualifier '(external)' in the input list.

There is no assumption that the presence of an input means that the associated deliverable is complete or in its final state. The input only needs to be sufficiently complete to allow successive work to begin. Any number of instances of an input may exist during the life cycle of an initiative.

The Inputs section includes a visual representation of the inputs and outputs, the other tasks that use the outputs, as well as the guidelines and tools listed in the task.

.4 ELEMENTS

The Elements section describes the key concepts that are needed to understand how to perform the task. Elements are not mandatory as part of performing a task, and their usage might depend upon the business analysis approach.

.5 GUIDELINES AND TOOLS

The Guidelines and Tools section lists resources that are required to transform the input into an output. A guideline provides instructions or descriptions on why or how to undertake a task. A tool is something used to undertake a task.

Guidelines and tools can include outputs of other tasks.

.6 TECHNIQUES

The Techniques section lists the techniques that can be used to perform the business analysis task.

.7 STAKEHOLDERS

The Stakeholders section is composed of a generic list of stakeholders who are likely to participate in performing that task or who will be affected by it. The BABOK® GUIDE does not mandate that these roles be filled for any given initiative.

.8 OUTPUTS

The Outputs section describes the results produced by performing the task. Outputs are created, transformed, or changed in state as a result of the successful completion of a task. An output may be a deliverable or be a part of a larger deliverable. The form of an output is dependent on the type of initiative underway, standards adopted by the organization, and best judgment of the business analyst as to an appropriate way to address the information needs of key stakeholders.

As with inputs, an instance of a task may be completed without an output being in its final state. Tasks that use a specific output do not necessarily have to wait for its completion for work within the task to begin.

UNDERLYING COMPETENCIES

Underlying competencies reflect knowledge, skills, behaviours, characteristics, and personal qualities that help one successfully perform the role of the business analyst. These underlying competencies are not unique to the business analysis profession. However, successful execution of tasks and techniques is often dependent on proficiency in one or more underlying competencies.

Underlying competencies have the following structure:

- Purpose
- Definition
- Effectiveness Measures

.1 PURPOSE

The Purpose section describes why it is beneficial for business analysts to have this underlying competency.

.2 DEFINITION

The Definition section describes the skills and expertise involved in the application of this competency.

.3 EFFECTIVENESS MEASURES

The Effectiveness Measures section describes how to determine whether a person is demonstrating skills in this underlying competency.

TECHNIQUES

Techniques provide additional information on ways that a task may be performed.

The list of techniques included in the BABOK® GUIDE is not exhaustive. There are multiple techniques that may be applied alternatively or in conjunction with other techniques to accomplish a task. Business analysts are encouraged to modify existing techniques or engineer new ones to best suit their situation and the goals of the tasks they perform.

Techniques have the following structure:

- Purpose
 - Description
 - Elements
 - Usage Considerations
-

.1 PURPOSE

The Purpose section describes what the technique is used for and the circumstances under which it is most likely to be applicable.

.2 DESCRIPTION

The Description section describes what the technique is and how it is used.

.3 ELEMENTS

The Elements section describes key concepts that are needed to understand how to use the technique.

.4 USAGE CONSIDERATIONS

The Usage Considerations section describes the conditions under which the technique may be more or less effective.

PERSPECTIVES

Perspectives are used within business analysis work to provide focus to tasks and techniques specific to the context of the initiative. Most initiatives are likely to engage one or more perspectives. The perspectives included in the BABOK® GUIDE are:

- Agile
- Business Intelligence
- Information Technology
- Business Architecture
- Business Process Management

These perspectives do not presume to represent all the possible perspectives from which business analysis is practiced. The perspectives discussed in the BABOK® GUIDE represent some of the more common views of business analysis at the time of writing.

Perspectives are not mutually exclusive, in that a given initiative might employ more than one perspective.

Perspectives have the following structure:

- Change Scope
- Business Analysis Scope
- Methodologies, Approaches, and Techniques
- Underlying Competencies
- Impact on Knowledge Areas

.1 CHANGE SCOPE

The Change Scope section describes what parts of the enterprise the change encompasses when viewed from this perspective and to what extent it impacts both the objectives and operations of the enterprise. The change scope also identifies the type of problems solved, the nature of the solutions being sought, and the approach to delivering these solutions and measuring their value.

.2 BUSINESS ANALYSIS SCOPE

The Business Analysis Scope section describes the key stakeholders, including a profile of the likely types of sponsors, the target stakeholders, and the business analyst's role within an initiative. It also defines likely outcomes that would be expected from business analysis work in this perspective.

.3 METHODOLOGIES, APPROACHES, AND TECHNIQUES

The composition of this section is unique to each perspective. In each case it describes the methodologies, approaches, or techniques that are common and specific to the application of business analysis in the perspective. Methodologies and approaches are specialized ways of undertaking the business analysis work. The techniques included in this section are techniques that are not included in the Techniques chapter of the BABOK® GUIDE but are especially relevant to the perspective.

Note: In the Business Architecture perspective, reference models are listed instead of methodologies or approaches. In the Business Process Management perspective, frameworks are listed instead of approaches.

.4 UNDERLYING COMPETENCIES

The Underlying Competencies section describes the competencies that are most prevalent in the perspective.

.5 IMPACT ON KNOWLEDGE AREAS

The Impact on Knowledge Areas section describes how knowledge areas are applied or modified. It also explains how specific activities within a perspective are mapped to tasks in the BABOK® GUIDE.

1.5 ECBAA, CCBA, CBAP EXAM BLUEPRINT

Knowledge Area	ECBA Percent	ECBA # of Questions*	CCBA Percent	CCBA # of Questions	CBAP Percent	CBAP # of Questions
Business Analysis Planning and Monitoring	5%	2-3	12%	15	14%	17
Elicitation and Collaboration	20%	10	20%	26	12%	14
Requirements Life Cycle Management	20%	10	18%	23	15%	18
Strategy Analysis	5%	2-3	12%	16	15%	18
Requirements Analysis & Design Definition	24%	12	32%	42	30%	36
Solution Evaluation	1%	1	6%	8	14%	17
Business Analysis & the BA Profession*	2.5%	1	N/A	N/A	N/A	N/A
Underlying Competencies*	5%	2-3	N/A	N/A	N/A	N/A
Key Concepts*	5%	2-3	N/A	N/A	N/A	N/A
Techniques *	12.5%	6	N/A	N/A	N/A	N/A
TOTAL		50		130		120

1.5.1. ECBA, CCBA, CBAP ELIGIBILITY REQUIREMENTS

Area	ECBA	CCBA	CBAP
BA Work Experience	N/A	3,750 hours in past 7 years	7,500 hours in past 10 years
BABOK Knowledge Area Expertise	N/A	900+ hours in <u>two</u> KAs or 500+ in <u>four</u> KAs	900 + hours in <u>four</u> KAs
Professional Development Hours in last 4 years	21	21	35
Education	Min: High School or equivalent		
References	N/A	2	2
Signed Code of Conduct	Yes	Yes	Yes

For the most recent certification requirements, download ECBA Handbook, CCBA Handbook or CBAP Handbook at www.iiba.org.

1.5.2. ECBA, CCBA, CBAP APPLICATION AND EXAM FEE

	IIBA Member	Non-Member
ECBA™		
Application Fee	\$60	\$60
Exam Fee	\$110	\$235
Retake Fee	\$85	\$195

CCBA® & CBAP®		
Application Fee	\$125	\$125
Exam Fee	\$325	\$450
Retake Fee	\$250	\$375

For the most recent certification exam fee, visit www.iiba.org.

2. Business Analysis Key Concepts

The Business Analysis Key Concepts chapter includes information that provides a foundation for all other content, concepts, and ideas within the *BABOK® Guide*. It provides business analysts with a basic understanding of the central ideas necessary for understanding and employing the *BABOK® Guide* in their daily practice of business analysis.

This chapter consists of:

Concept	Description
Business Analysis Core Concept Model™ (BACCM™)	Defines a conceptual framework for the business analysis profession.
Key Terms	Provides definitions of essential concepts, which are highlighted because of their importance to the <i>BABOK® Guide</i> .
Requirements Classification Schema	Identifies levels or types of requirements that assist the business analyst and other stakeholders in categorizing requirements.
Stakeholders	Defines roles, and characteristics of groups or individuals participating in or affected by the business analysis activities within a change.
Requirements and Designs	Describes the distinction between—and the importance of—requirements and designs as they relate to business analysis.

2.1 THE BUSINESS ANALYSIS CORE CONCEPT MODEL

The BUSINESS ANALYSIS CORE CONCEPT MODEL™ (BACCM™) is a conceptual framework for business analysis. It encompasses what business analysis is and what it means to those performing business analysis tasks regardless of perspective, industry, methodology, or level in the organization. It is composed of six terms that have a common meaning to all business analysts and helps them discuss both business analysis and its relationships with common terminology. Each of these terms is considered to be a core concept.

The six core concepts in the BACCM are: Change, Need, Solution, Stakeholder, Value, and Context. Each core concept is an idea fundamental to the practice of business analysis, and all the concepts are equal and necessary. Each core concept is defined by the other five core concepts and cannot be fully understood until all the concepts are understood. No single concept holds greater importance or significance over any other concept. These concepts are instrumental to understanding the type of information elicited, analyzed, or managed in business analysis tasks.

The BACCM can be used to:

- describe the profession and domain of business analysis,
- communicate about business analysis with a common terminology,
- evaluate the relationships of key concepts in business analysis,
- perform better business analysis by holistically evaluating the relationships among these six concepts, and

- evaluate the impact of these concepts and relationships at any point during a work effort in order to establish both a foundation and a path forward.

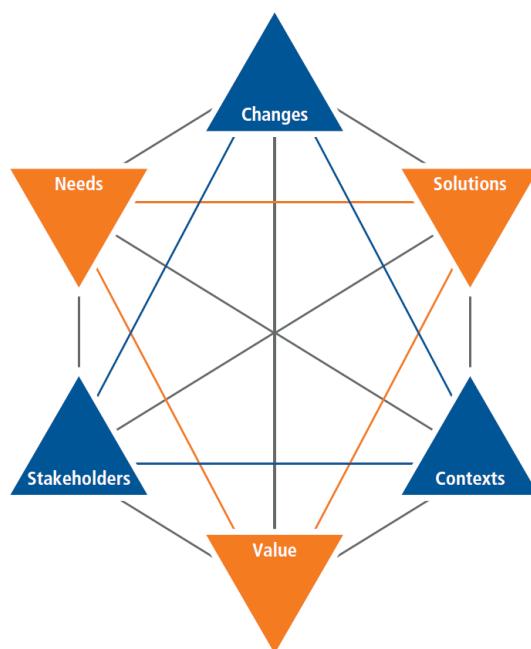
Table 2.1.1: The BACCM

Core Concept	Description
Change	<p>The act of transformation in response to a need.</p> <p>Change works to improve the performance of an enterprise. These improvements are deliberate and controlled through business analysis activities.</p>
Need	<p>A problem or opportunity to be addressed.</p> <p>Needs can cause changes by motivating stakeholders to act. Changes can also cause needs by eroding or enhancing the value delivered by existing solutions.</p>
Solution	<p>A specific way of satisfying one or more needs in a context.</p> <p>A solution satisfies a need by resolving a problem faced by stakeholders or enabling stakeholders to take advantage of an opportunity.</p>
Stakeholder	<p>A group or individual with a relationship to the change, the need, or the solution.</p> <p>Stakeholders are often defined in terms of interest in, impact on, and influence over the change. Stakeholders are grouped based on their relationship to the needs, changes, and solutions.</p>
Value	<p>The worth, importance, or usefulness of something to a stakeholder within a context.</p> <p>Value can be seen as potential or realized returns, gains, and improvements. It is also possible to have a decrease in value in the form of losses, risks, and costs.</p> <p>Value can be tangible or intangible. Tangible value is directly measurable. Tangible value often has a significant monetary component. Intangible value is measured indirectly. Intangible value often has a significant motivational component, such as a company's reputation or employee morale.</p> <p>In some cases, value can be assessed in absolute terms, but in many cases is assessed in relative terms: one solution option is more valuable than another from the perspective of a given set of stakeholders.</p>
Context	<p>The circumstances that influence, are influenced by, and provide understanding of the change.</p> <p>Changes occur within a context. The context is everything relevant to the change that is within the environment. Context may include attitudes, behaviours, beliefs, competitors, culture, demographics, goals, governments, infrastructure, languages, losses, processes, products, projects, sales, seasons, terminology, technology, weather, and any other element meeting the definition.</p>

The core concepts can be used by business analysts to consider the quality and completeness of the work being done. Within each knowledge area description there are examples of how the core concepts may be used and/or applied during the tasks within the knowledge area. While planning or performing a task or technique, business analysts can consider how each core concept is addressed by asking questions such as:

- What are the kinds of CHANGES we are doing?
- What are the NEEDS we are trying to satisfy?
- What are the SOLUTIONS we are creating or changing?
- Who are the STAKEHOLDERS involved?
- What do stakeholders consider to be of VALUE?
- What are the CONTEXTS that we and the solution are in?
- If any of the core concepts experience a change, it should cause us to re-evaluate these core concepts and their relationships to value delivery.

Figure 2.1.1: The BACCM



2.2 KEY TERMS

Business Analysis

The BABOK® GUIDE describes and defines business analysis as the practice of enabling change in an enterprise by defining needs and recommending solutions that deliver value to stakeholders.

Business Analysis Information

Business analysis information refers to the broad and diverse sets of information that business analysts analyze, transform, and report. It is information of any kind—at any level of detail—that is used as an input to, or is an output of, business analysis work. Examples of business analysis information include elicitation results, requirements, designs, solution options, solution scope, and change strategy.

It is essential to expand the object of many business analysis activities from 'requirements' to 'information' to ensure that all inputs and outputs of business analysis are subject to the tasks and activities described in the BABOK® GUIDE. For example, when performing 'Plan Business Analysis Information Management' it includes all the examples listed above. If

the BABOK® Guide described 'Plan Requirements Management', it would exclude important outputs like elicitation results, solution options, and change strategy.

Design

A design is a usable representation of a solution. Design focuses on understanding how value might be realized by a solution if it is built. The nature of the representation may be a document (or set of documents) and can vary widely depending on the circumstances.

Enterprise

An enterprise is a system of one or more organizations and the solutions they use to pursue a shared set of common goals. These solutions (also referred to as organizational capabilities) can be processes, tools or information. For the purpose of business analysis, enterprise boundaries can be defined relative to the change and need not be constrained by the boundaries of a legal entity, organization, or organizational unit. An enterprise may include any number of business, government, or any other type of organization.

Organization

An autonomous group of people under the management of a single individual or board, that works towards common goals and objectives. Organizations often have a clearly defined boundary and operate on a continuous basis, as opposed to an initiative or project team, which may be disbanded once its objectives are achieved.

Plan

A plan is a proposal for doing or achieving something. Plans describe a set of events, the dependencies among the events, the expected sequence, the schedule, the results or outcomes, the materials and resources needed, and the stakeholders involved.

Requirement

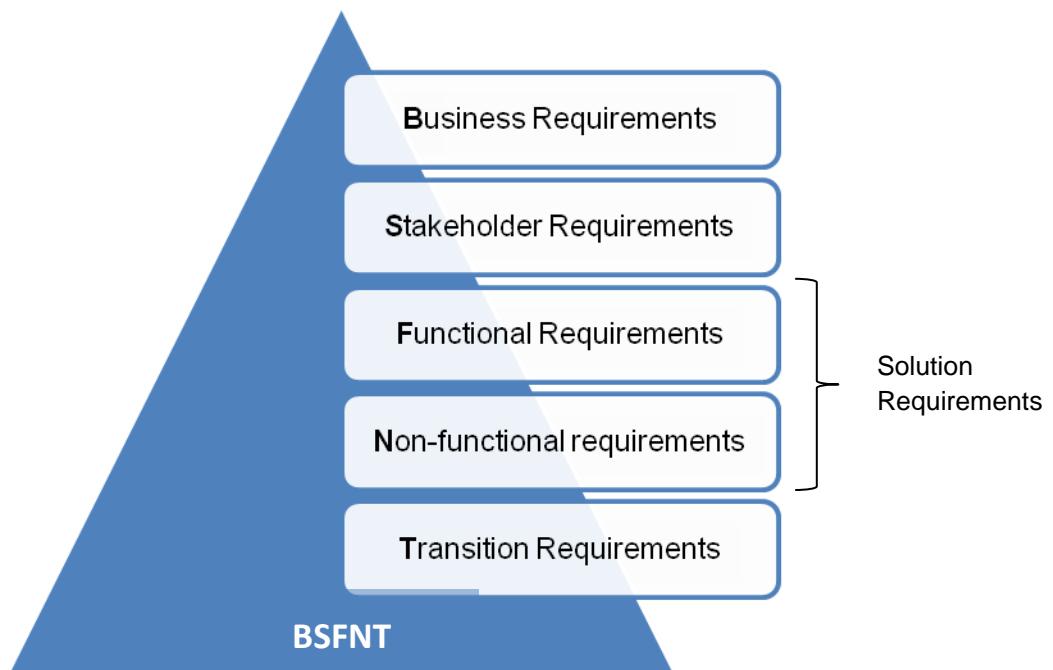
A requirement is a usable representation of a need. Requirements focus on understanding what kind of value could be delivered if a requirement is fulfilled. The nature of the representation may be a document (or set of documents), but can vary widely depending on the circumstances.

Risk

Risk is the effect of uncertainty on the value of a change, a solution, or the enterprise. Business analysts collaborate with other stakeholders to identify, assess, and prioritize risks, and to deal with those risks by altering the likelihood of the conditions or events that lead to the uncertainty: mitigating the consequences, removing the source of the risk, avoiding the risk altogether by deciding not to start or continue with an activity that leads to the risk, sharing the risk with other parties, or accepting or even increasing the risk to deal with an opportunity.

2.3 REQUIREMENTS CLASSIFICATION SCHEMA

For the purposes of the BABOK® GUIDE, the following classification schema describes requirements:



- **Business requirements:** statements of goals, objectives, and outcomes that describe why a change has been initiated. They can apply to the whole of an enterprise, a business area, or a specific initiative.
- **Stakeholder requirements:** describe the needs of stakeholders that must be met in order to achieve the business requirements. They may serve as a bridge between business and solution requirements.
- **Solution requirements:** describe the capabilities and qualities of a solution that meets the stakeholder requirements. They provide the appropriate level of detail to allow for the development and implementation of the solution. Solution requirements can be divided into two sub-categories:
 - **functional requirements:** describe the capabilities that a solution must have in terms of the behaviour and information that the solution will manage, and
 - **non-functional requirements or quality of service requirements:** do not relate directly to the behaviour of functionality of the solution, but rather describe conditions under which a solution must remain effective or qualities that a solution must have.
- **Transition requirements:** describe the capabilities that the solution must have and the conditions the solution must meet to facilitate transition from the current state to the future state, but which are not needed once the change is complete. They are differentiated from other requirements types because they are of a temporary nature. Transition requirements address topics such as data conversion, training, and business continuity.

2.4 STAKEHOLDERS

Each task includes a list of stakeholders who are likely to participate in the execution of that task or who will be affected by it. A stakeholder is an individual or group that a business analyst is likely to interact with directly or indirectly. The BABOK® GUIDE does not mandate that these

roles be filled for any given initiative. Any stakeholder can be a source of requirements, assumptions, or constraints.

This list is not intended to be an exhaustive list of all possible stakeholder classifications. Some additional examples of people who fit into each of these generic roles are listed in the definitions below. In most cases there will be multiple stakeholder roles found within each category. Similarly, a single individual may fill more than one role.

For the purpose of the BABOK® GUIDE, the generic list of stakeholders includes the following roles:

- business analyst,
- customer,
- domain subject matter expert,
- end user,
- implementation subject matter expert,
- operational support,
- project manager,
- regulator,
- sponsor,
- supplier, and
- tester.

.1 BUSINESS ANALYST

The business analyst is inherently a stakeholder in all business analysis activities. The BABOK® GUIDE presumes that the business analyst is responsible and accountable for the execution of these activities. In some cases the business analyst may also be responsible for performing activities that fall under another stakeholder role.

.2 CUSTOMER

A customer uses or may use products or services produced by the enterprise and may have contractual or moral rights that the enterprise is obliged to meet.

.3 DOMAIN SUBJECT MATTER EXPERT

A domain subject matter expert is any individual with in-depth knowledge of a topic relevant to the business need or solution scope. This role is often filled by people who may be end users or people who have in-depth knowledge of the solution such as managers, process owners, legal staff, consultants, and others.

.4 END USER

End users are stakeholders who directly interact with the solution. End users can include all participants in a business process, or who use the product or solution.

.5 IMPLEMENTATION SUBJECT MATTER EXPERT

An implementation subject matter expert is any stakeholder who has specialized knowledge regarding the implementation of one or more solution components.

While it is not possible to define a listing of implementation subject matter expert roles that are appropriate for all initiatives, some of the most common roles are: project librarian, change manager, configuration manager, solution architect, developer, database administrator, information architect, usability analyst, trainer, and organizational change consultant.

.6 OPERATIONAL SUPPORT

Operational support is responsible for the day-to-day management and maintenance of a system or product.

While it is not possible to define a listing of operational support roles that are appropriate for all initiatives, some of the most common roles are: operations analyst, product analyst, help desk, and release manager.

.7 PROJECT MANAGER

Project managers are responsible for managing the work required to deliver a solution that meets a business need, and for ensuring that the project's objectives are met while balancing the project factors including scope, budget, schedule, resources, quality, and risk. While it is not possible to completely define a listing of project management roles that are appropriate for all initiatives, some of the most common roles are: project lead, technical lead, product manager, and team leader.

.8 REGULATOR

Regulators are responsible for the definition and enforcement of standards. Standards can be imposed on the solution by regulators through legislation, corporate governance standards, audit standards, or standards defined by organizational centers of competency. Alternate roles are government, regulatory bodies, and auditor.

.9 SPONSOR

Sponsors are responsible for initiating the effort to define a business need and develop a solution that meets that need. They authorize the work to be performed, and control the budget and scope for the initiative. Alternate roles are executive and project sponsor.

.10 SUPPLIER

A supplier is a stakeholder outside the boundary of a given organization or organizational unit. Suppliers provide products or services to the organization and may have contractual or moral rights and obligations that must be considered. Alternate roles are providers, vendors, and consultants.

.11 TESTER

Testers are responsible for determining how to verify that the solution meets the requirements defined by the business analyst, as well as conducting the verification process. Testers also seek to ensure that the solution meets applicable quality standards, and that the risk of defects or failures is understood and minimized. An alternate role is quality assurance analyst.

2.5 REQUIREMENTS AND DESIGNS

Eliciting, analyzing, validating, and managing requirements have consistently been recognized as key activities of business analysis. However, it is important to recognize that business analysts are also responsible for the definition of design, at some level, in an initiative. The level of responsibility for design varies based on the perspective within which a business analyst is working.

Requirements are focused on the need; designs are focused on the solution. The distinction between requirements and designs is not always clear. The same techniques are used to elicit, model, and analyze both. A requirement leads to a design which in turn may drive the discovery and analysis of more requirements. The shift in focus is often subtle.

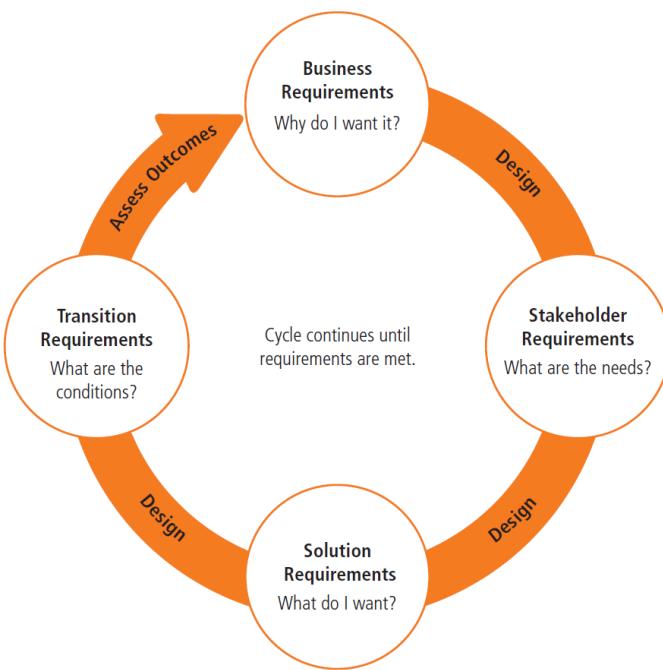
Business analysis can be complex and recursive. A requirement (or set of requirements) may be used to define a design. That design may then be used to elicit additional requirements that are used to define more detailed designs. The business analyst may hand off requirements and designs to other stakeholders who may further elaborate on the designs. Whether it is the business analyst or some other role that completes the designs, the business analyst often reviews the final designs to ensure that they align with the requirements.

The following table provides some basic examples of how information may be viewed as either a requirement or a design.

Table 2.5.1: Requirements and Design

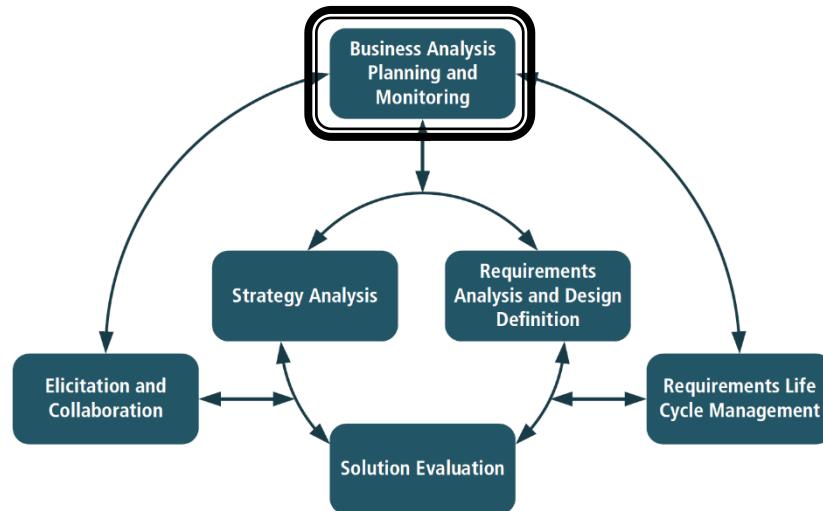
Requirement	Design
View six months sales data across multiple organizational units in a single view.	A sketch of a dashboard.
Reduce amount of time required to pick and pack a customer order.	Process model.
Record and access a medical patient's history.	Screen mock-up showing specific data fields.
Develop business strategy, goals, and objectives for a new business.	Business Capability Model.
Provide information in English and French.	Prototype with text displayed in English and French.

Figure 2.5.1: Requirements and Design Cycle



3. Business Analysis Planning and Monitoring

The Business Analysis Planning and Monitoring knowledge area tasks organize and coordinate the efforts of business analysts and stakeholders. These tasks produce outputs that are used as key guidelines for the other tasks throughout the *BABOK® Guide*.



The Business Analysis Planning and Monitoring knowledge area includes the following tasks:

ASGIP

1. Plan Business Analysis Approach
2. Plan Stakeholder Engagement
3. Plan Business Analysis Governance
4. Plan Business Analysis Information Management
5. Identify Business Analysis Performance Improvements

The Core Concept Model in Business Analysis Planning and Monitoring

The *Business Analysis Core Concept Model™ (BACCM™)* describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Business Analysis Planning and Monitoring.

Core Concept	Usage in Business Analysis Planning and Monitoring
Change	Determine how change to business analysis results will be requested and authorized
Need	Select a business analysis approach that provides a suitable amount of analysis for the desired change

Solution	Evaluate if business analysis performance contributed to successful implementation of a solution
Stakeholder	Perform stakeholder analysis to account for stakeholder characteristics and reflect their needs
Value	Use performance analysis to ensure business activities produce sufficient value for the stakeholders

3.1 PLAN BUSINESS ANALYSIS APPROACH

3.1.1 PURPOSE

The purpose of Plan Business Analysis Approach is to define an appropriate method to conduct business analysis activities.

3.1.2 DESCRIPTION

Business analysis approaches describe the overall method that will be followed when performing business analysis work on a given initiative, how and when tasks will be performed, and the deliverables that will be produced. The business analyst may also identify an initial set of techniques to use. This list may change as the initiative proceeds and the business analyst gains a deeper understanding of the change and its stakeholders.

The business analysis approach may be defined by a methodology or by organizational standards. In some organizations, elements of the business analysis approach may be standardized and formalized into a repeatable business analysis process which can be leveraged for each effort. Even where a standard approach exists, it may be tailored to the needs of a specific initiative. Tailoring may be governed by standards that define which approaches are permitted, which elements of those processes may be tailored, and general guidelines for selecting a process.

If organizational standards do not exist, the business analyst works with the appropriate stakeholders to determine how the work will be completed. For example, if the change is delivered via a project, the standards and approach may be developed during the project planning phase.

The business analysis approach should:

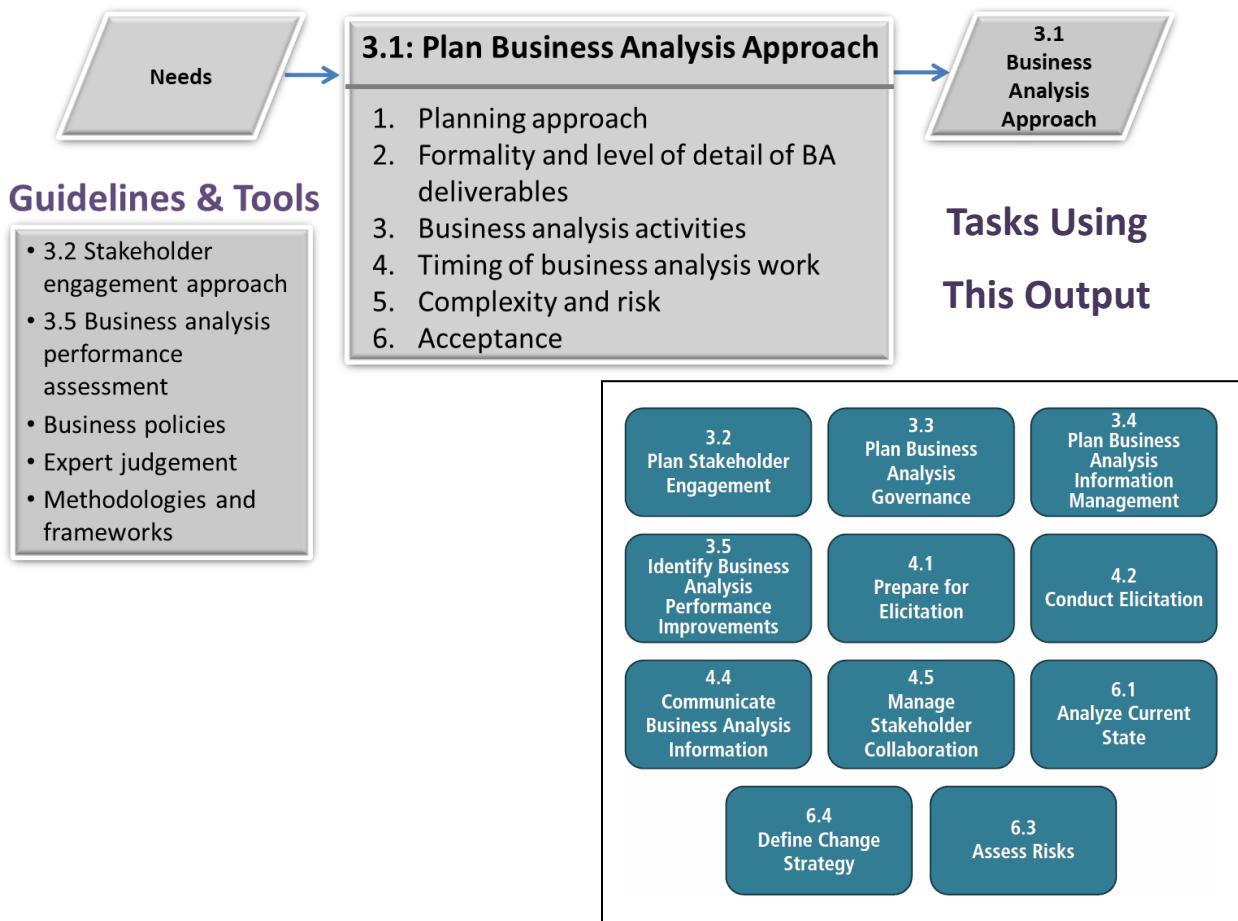
- align to the overall goals of the change,
- coordinate the business analysis tasks with the activities and deliverables of the overall change,
- include tasks to manage any risks that could reduce the quality of business analysis deliverables or impede task efficiency, and
- leverage approaches and select techniques and tools that have historically worked well.

3.1.3 INPUTS

- **Needs:** the business analysis approach is shaped by the problem or opportunity faced by the organization. It is necessary to consider what is known about the need at the time of

planning, while acknowledging that understanding evolves throughout business analysis activities.

Figure 3.1.1: Plan Business Analysis Approach Input/Output Diagram



3.1.4 ELEMENTS

.1 PLANNING APPROACH

There are various planning methods used across perspectives, industries, and enterprises. Many planning methods fit somewhere along a continuum between predictive and adaptive approaches.

Predictive approaches focus on minimizing upfront uncertainty and ensuring that the solution is defined before implementation begins in order to maximize control and minimize risk. These approaches are often preferred in situations where requirements can effectively be defined ahead of implementation, the risk of an incorrect implementation is unacceptably high, or when engaging stakeholders presents significant challenges.

Adaptive approaches focus on rapid delivery of business value in short iterations in return for acceptance of a higher degree of uncertainty regarding the overall delivery of the solution. These

approaches tend to be preferred when taking an exploratory approach to finding the best solution or for incremental improvement of an existing solution.

Different approaches may be used within the same initiative. Among other factors, the business analyst may consider the organization's standards, tolerance for uncertainty, and previous experience with different approaches when planning for business analysis activities.

Regardless of the approach, planning is an essential task to ensure value is delivered to an enterprise. Planning typically occurs more than once on a given initiative as plans are updated to address changing business conditions and newly raised issues. The business analysis approach should describe how plans will be altered if changes are required.

.2 FORMALITY AND LEVEL OF DETAIL OF BUSINESS ANALYSIS DELIVERABLES

When defining the business analysis approach, consider the level of formality that is appropriate for approaching and planning the initiative.

Predictive approaches typically call for formal documentation and representations. Business analysis information may be captured in a formal document or set of representations following standardized templates. Information is captured at various levels of detail. The specific content and format of business analysis information can vary depending on the organizational methodologies, processes, and templates in use.

Adaptive approaches favour defining requirements and designs through team interaction and gathering feedback on a working solution. Mandatory requirements representations are often limited to a prioritized requirements list. Additional business analysis documentation may be created at the discretion of the team, and generally consists of models developed to enhance the team's understanding of a specific problem. Formal documentation is often produced after the solution is implemented to facilitate knowledge transfer.

Other considerations that may affect the approach include:

- the change is complex and high risk,
- the organization is in, or interacts with, heavily regulated industries,
- contracts or agreements necessitate formality,
- stakeholders are geographically distributed,
- resources are outsourced,
- staff turnover is high and/or team members may be inexperienced,
- requirements must be formally signed off, and
- business analysis information must be maintained long-term or handed over for use on future initiatives.

Approach		
	Predictive	Adaptive
Solution Definition	Defined before implementation to maximize control and minimize risk.	Defined in iterations to arrive at best solution or improve an existing solution.
Level of Formality	Formal—information is captured in standardized templates.	Informal—information is gathered through team interaction and feedback.
Activities	Activities required to complete deliverables are identified first and then divided into tasks.	Activities are divided into iterations with deliverables first and then the associated tasks are identified.
Timing	Tasks are performed in specific phases.	Tasks are performed iteratively.

.3 BUSINESS ANALYSIS ACTIVITIES

A business analysis approach provides a description of the types of activities that the business analyst will perform. Frequently the organization's adopted methodologies influence the activities that are selected.

Integrating business analysis activities in the business analysis approach includes:

- identifying the activities required to complete each deliverable and then breaking each activity into tasks,
- dividing the work into iterations, identifying the deliverables for each iteration, and then identifying the associated activities and tasks, or
- using a previous similar initiative as an outline and applying the detailed tasks and activities unique to the current initiative.

.4 TIMING OF BUSINESS ANALYSIS WORK

Business analysts determine when the business analysis tasks need to be performed and if the level of business analysis effort will need to vary over time. This type of planning includes determining whether the business analysis tasks performed within the other knowledge areas will be performed primarily in specific phases or iteratively over the course of the initiative.

The timing of business analysis activities can also be affected by:

- the availability of resources,
- priority and/or urgency of the initiative,
- other concurrent initiatives, or

- constraints such as contract terms or regulatory deadlines.

.5 COMPLEXITY AND RISK

The complexity and size of the change and the overall risk of the effort to the organization are considered when determining the business analysis approach. As complexity and risk increase or decrease, the nature and scope of business analysis work can be altered and reflected in the approach.

The approach may also be altered based on the number of stakeholders or business analysis resources involved in the initiative. As the number of stakeholders increases, the approach may be adjusted to include additional process steps to better manage the business analysis work.

Other factors that can impact complexity include:

- size of the change,
- number of business areas or systems affected,
- geographic and cultural considerations,
- technological complexities, and
- any risks that could impede the business analysis effort.

Factors that can impact the risk level of a business analysis effort include:

- experience level of the business analyst,
- extent of domain knowledge held by the business analyst,
- level of experience stakeholders have in communicating their needs,
- stakeholder attitudes about the change and business analysis in general,
- amount of time allocated by stakeholders to the business analysis activities,
- any pre-selected framework, methodology, tools, and/or techniques imposed by organizational policies and practices, and
- cultural norms of the organization.

.6 ACCEPTANCE

The business analysis approach is reviewed and agreed upon by key stakeholders. In some organizations, the business analysis process may be more structured and require key stakeholders to sign off on the approach to ensure all business analysis activities have been identified, estimates are realistic, and the proposed roles and responsibilities are correct. Any issues raised by stakeholders when reviewing the approach are documented by the business analyst and resolutions are sought. Stakeholders also play a role in reviewing and accepting changes to the approach as alterations are made to accommodate changing conditions across the initiative.

3.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Performance Assessment	Provides results of previous assessments that should be reviewed and incorporated into all planning approaches.
Business Policies	Define the limits within which decisions must be made. They may be described by regulations, contracts, agreements, deals, warranties, certifications, or other legal obligations. These policies can influence the business analysis approach.

Expert Judgment	Used to determine the optimal business analysis approach. Expertise may be provided from a wide range of sources including stakeholders on the initiative, organizational centres of excellence, consultants, or associations and industry groups. Prior experiences of the business analyst and other stakeholders should be considered when selecting or modifying an approach.
Methodologies and Frameworks	Shape the approach that will be used by providing methods, techniques, procedures, working concepts, and rules. They may need to be tailored to better meet the needs of the specific business challenge.
Stakeholder Engagement Approach	Understanding the stakeholders and their concerns and interests may influence decisions made when determining the business analysis approach.

3.1.5 TECHNIQUES

Techniques	Usage
Brainstorming	Used to identify possible business analysis activities, techniques, risks and other relevant items to help build the business analysis approach.
Business Cases	Used to understand whether elements of the problem or opportunity are especially time-sensitive, high-value, or whether there is any particular uncertainty around elements of the possible need or solution.
Document Analysis	Used to review existing organizational assets that might assist in planning the approach.
Estimation	Used to determine how long it may take to perform business analysis activities.
Financial Analysis	Used to assess how different approaches (and the supported delivery options) affect the value delivered.
Functional Decomposition	Used to break down complex business analysis processes or approaches into more feasible components.
Interviews	Used to help build the plan with an individual or small group.
Item Tracking	Used to track any issues raised during planning activities with stakeholders. Can also track risk related items raised during discussions when building the approach.
Lessons Learned	Used to identify an enterprise's previous experience (both successes and challenges) with planning business analysis approach.
Process Modelling	Used to define and document the business analysis approach.
Reviews	Used to validate the selected business analysis approach with stakeholders.
Risk Analysis and Management	Used to assess risks in order to select the proper business analysis approach.

Scope Modelling	Used to determine the boundaries of the solution as an input to planning and to estimating.
Survey or Questionnaire	Used to identify possible business analysis activities, techniques, risks and other relevant items to help build the business analysis approach.
Workshops	Used to help build the plan in a team setting.

3.1.6 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	Can be a source of risk when their involvement is required and availability is lacking. The approach taken may depend on availability and level of their involvement with the initiative.
Project Manager	Determines that the approach is realistic for the overall schedule and timelines. The business analysis approach must be compatible with other activities.
Regulator	May be needed to provide approval for aspects of the business analysis approach or decisions made in tailoring the process, especially in organizations where the business analysis process is audited.
Sponsor	Can provide needs and objectives for the approach and ensures that organizational policies are followed. The selected approach may depend on availability and involvement with the initiative.

3.1.7 OUTPUTS

- **Business Analysis Approach:** identifies the business analysis approach and activities that will be performed across an initiative including who will perform the activities, the timing and sequencing of the work, the deliverables that will be produced and the business analysis techniques that may be utilized. The remaining outputs of the Business Analysis Planning and Monitoring knowledge area may be integrated into an overall approach or be independent based upon methodology, organization, and perspective.

3.2 PLAN STAKEHOLDER ENGAGEMENT

3.2.1 PURPOSE

The purpose of Plan Stakeholder Engagement is to plan an approach for establishing and maintaining effective working relationships with the stakeholders.

3.2.2 DESCRIPTION

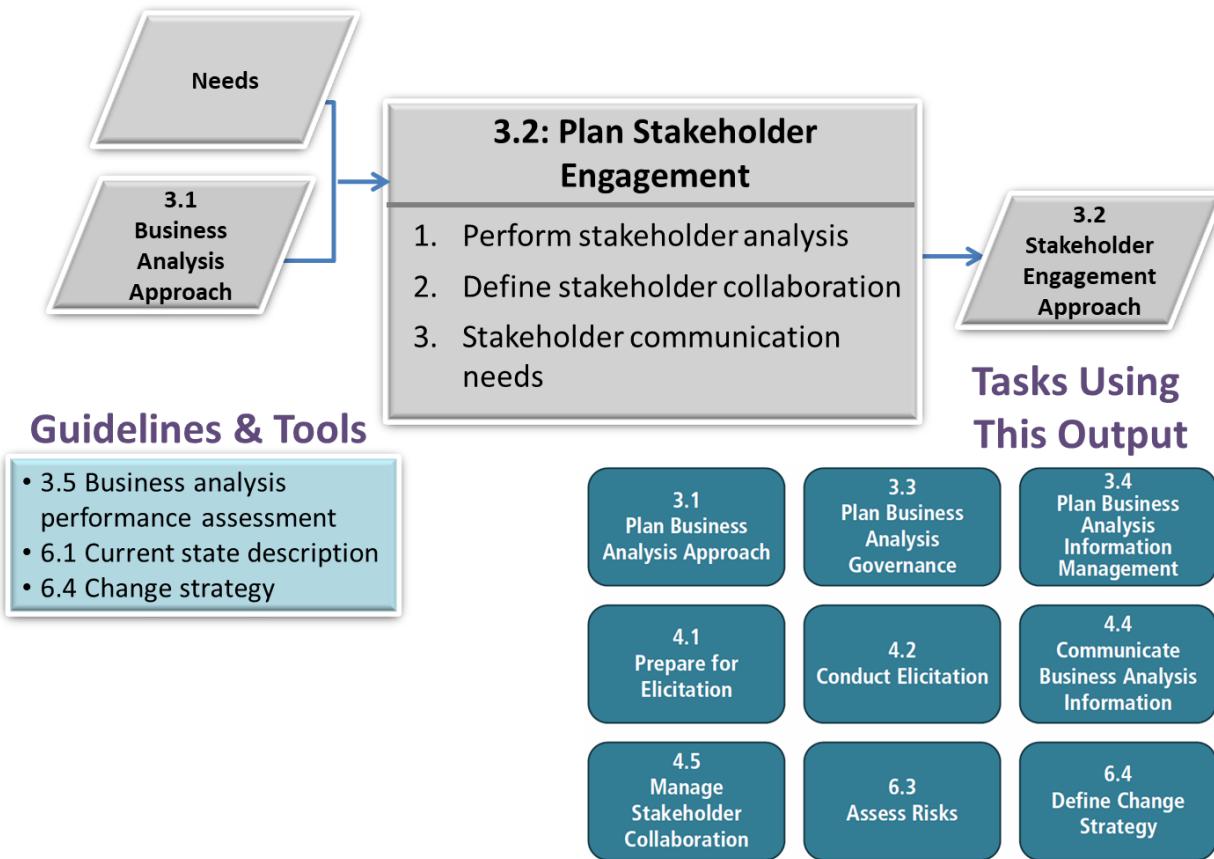
Plan Stakeholder Engagement involves conducting a thorough stakeholder analysis to identify all of the involved stakeholders and analyze their characteristics. The results of the analysis are then utilized to define the best collaboration and communication approaches for the initiative and to appropriately plan for stakeholder risks.

When planning for stakeholder engagement, the degree of complexity can increase disproportionately as the number of stakeholders involved in the business analysis activities increases. This is important because new or different techniques for the management of stakeholders may be required when the engagement moves from collaborating with a few stakeholders into dozens, hundreds, or even thousands of people.

3.2.3 INPUTS

- **Needs:** understanding the business need and the parts of the enterprise that it affects helps in the identification of stakeholders. The need may evolve as stakeholder analysis is performed.
- **Business Analysis Approach:** incorporating the overall business analysis approach into the stakeholder analysis, collaboration, and communication approaches is necessary to ensure consistency across the approaches.

Figure 3.2.1: Plan Stakeholder Engagement Input/Output Diagram



3.2.4 ELEMENTS

.1 PERFORM STAKEHOLDER ANALYSIS

Stakeholder analysis involves identifying the stakeholders (who will be directly or indirectly impacted by the change) and their characteristics, as well as analyzing the information once collected. Stakeholder analysis is performed repeatedly as business analysis activities continue.

A thorough and detailed stakeholder list ensures that stakeholders are not overlooked. Understanding who the stakeholders are, the impact of proposed changes on them, and the influence they may have on the change is vital to understanding what needs, wants, and expectations must be satisfied by a solution. If stakeholders are not identified, the business analyst may miss uncovering critical needs. Stakeholder needs uncovered late will often require a revision to business analysis tasks that are either in progress or are completed. This can result in increased costs and decreased stakeholder satisfaction.

How business analysts perform stakeholder analysis can vary between projects, methodologies, and organizations. A company's organizational chart and business processes can serve as an initial source for identifying internal stakeholders. The sponsor may also identify stakeholders. Stakeholders outside the organization may be identified and can be uncovered by understanding any existing contracts that may be in place, anticipated vendors that may have a role based on existing relationships with the organization, as well as regulatory and governing bodies that may influence the work. Shareholders, customers, and suppliers are also considered when searching for external stakeholders.

Roles

Business analysts identify stakeholder roles in order to understand where and how the stakeholders will contribute to the initiative. It is important that the business analyst is aware of the various roles a stakeholder is responsible for within the organization.

Attitudes

Stakeholder attitudes can positively or negatively impact a change. Business analysts identify stakeholder attitudes in order to fully understand what may impact a stakeholder's actions and behaviours. Knowing how a stakeholder perceives the initiative allows an opportunity for the business analyst to specifically plan their collaboration and engagement with that stakeholder.

Business analysts analyze stakeholder attitudes about:

- business goals, objectives of the initiative, and any proposed solutions,
- business analysis in general,
- the level of interest in the change,
- the sponsor,
- team members and other stakeholders, and
- collaboration and a team-based approach.

Stakeholders with positive attitudes may be strong champions and great contributors. Other stakeholders may not see value in the work, may misunderstand the value being provided, or may be concerned about the effect the change will have on them. Stakeholders who are expected to serve in key roles and participate heavily in business analysis activities, but who view a change negatively, may require collaboration approaches that increase their cooperation.

Decision Making Authority

Business analysts identify the authority level a stakeholder possesses over business analysis activities, deliverables, and changes to business analysis work. Understanding authority levels upfront eliminates confusion during the business analysis effort and ensures the business analyst collaborates with the proper stakeholders when looking for a decision to be made or seeking approvals.

Level of Power or Influence

Understanding the nature of influence and the influence structures and channels within an organization can prove invaluable when seeking to build relationships and trust. Understanding the influence and attitude each stakeholder may have can help develop strategies for obtaining buy-in and collaboration. Business analysts evaluate how much influence is needed to implement a change compared to the amount of influence the key stakeholders can bring. If there is a mismatch between the influence required and the amount of influence the stakeholder has or is perceived to have, business analysts develop risk plans, responses and other strategies that might be needed to obtain the required level of support.

.2 DEFINE STAKEHOLDER COLLABORATION

Ensuring effective collaboration with stakeholders is essential for maintaining their engagement in business analysis activities. Collaboration can be a spontaneous event. However, much collaboration is deliberate and planned, with specific activities and outcomes determined ahead of time during planning activities.

The business analyst may plan different collaboration approaches for internal and external stakeholders, and approaches may differ by business analysis activity. The objective is to select the approaches that work best to meet the needs of each stakeholder group and ensure their interest and involvement is maintained across the initiative. Some considerations when planning collaboration include:

- timing and frequency of collaboration,
 - location,
 - available tools such as wikis and online communities,
 - delivery method such as in-person or virtual, and
 - preferences of the stakeholders.
-
- Planning considerations can be documented in the form of a stakeholder collaboration plan. As factors change, plans can be revisited, and adjustments and adaptations can be made to ensure ongoing engagement of stakeholders.

.3 STAKEHOLDER COMMUNICATION NEEDS

The business analyst evaluates:

- what needs to be communicated,
- what is the appropriate delivery method (written or verbal),
- who the appropriate audience is,
- when communication should occur,
- frequency of communication,
- geographic location of stakeholders who will receive communications,
- level of detail appropriate for the communication and stakeholder, and
- level of formality of communications.

Communication considerations can be documented in the form of a stakeholder communication plan. Business analysts build and review communication plans with stakeholders to ensure their communication requirements and expectations are met.

3.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Performance Assessment	Provides results of previous assessments that should be reviewed and incorporated.
Change Strategy	Used for improved assessment of stakeholder impact and the development of more effective stakeholder engagement strategies.
Current State Description	Provides the context within which the work needs to be completed. This information will lead to more effective stakeholder analysis and better understanding of the impact of the desired change.

3.2.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to produce the stakeholder list and identify stakeholder roles and responsibilities.
Business Rules Analysis	Used to identify stakeholders who were the source of the business rules.
Document Analysis	Used to review existing organizational assets that might assist in planning stakeholder engagement.
Interviews	Used to interact with specific stakeholders to gain more information or knowledge about stakeholder groups.
Lessons Learned	Used to identify an enterprise's previous experience (both successes and challenges) with planning stakeholder engagement.
Mind Mapping	Used to identify potential stakeholders and help understand the relationships between them.
Organizational Modelling	Used to determine if the organizational units or people listed have any unique needs and interests that should be considered. Organizational models describe the roles and functions in the organization and the ways in which stakeholders interact which can help to identify stakeholders who will be affected by a change.
Process Modelling	Used to categorize stakeholders by the systems that support their business processes.
Risk Analysis and Management	Used to identify risks to the initiative resulting from stakeholder attitudes or the inability of key stakeholders to participate in the initiative.
Scope Modelling	Used to develop scope models to show stakeholders that fall outside the scope of the solution but still interact with it in some

	way.
Stakeholder List, Map, or Personas	Used to depict the relationship of stakeholders to the solution and to one another.
Survey or Questionnaire	Used to identify shared characteristics of a stakeholder group.
Workshops	Used to interact with groups of stakeholders to gain more information about stakeholder groups.

3.2.7 STAKEHOLDERS

Stakeholder	Contribution
Customers	A source of external stakeholders.
Domain Subject Matter Expert	May help to identify stakeholders and may themselves be identified to fulfill one or more roles on the initiative.
End User	A source of internal stakeholders.
Project Manager	May be able to identify and recommend stakeholders. Responsibility for stakeholder identification and management may be shared with the business analyst.
Regulator	May require that specific stakeholder representatives or groups be involved in the business analysis activities.
Sponsor	May request that specific stakeholders be involved in the business analysis activities.
Supplier	A source of external stakeholders.

3.2.8 OUTPUTS

- **Stakeholder Engagement Approach:** contains a list of the stakeholders, their characteristics which were analyzed, and a listing of roles and responsibilities for the change. It also identifies the collaboration and communication approaches the business analyst will utilize during the initiative.

3.3 PLAN BUSINESS ANALYSIS GOVERNANCE

3.3.1 PURPOSE

The purpose of Plan Business Analysis Governance is to define how decisions are made about requirements and designs, including reviews, change control, approvals, and prioritization.

3.3.2 DESCRIPTION

Business analysts ensure that a governance process is in place and clarify any ambiguities within it. A governance process identifies the decision makers, process, and information required for decisions to be made. A governance process describes how approvals and prioritization decisions are made for requirements and designs.

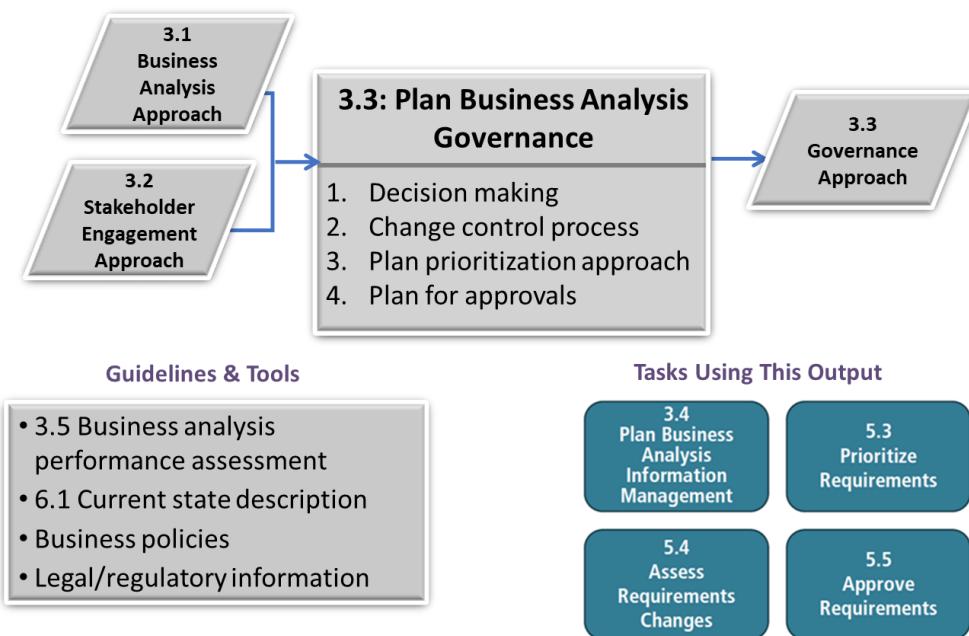
When planning the governance approach, business analysts identify:

- how business analysis work will be approached and prioritized,
- what the process for proposing a change to business analysis information is,
- who has the authority and responsibility to propose changes and who should be involved in the change discussions,
- who has responsibility for analyzing change requests,
- who has the authority to approve changes, and
- how changes will be documented and communicated.

3.3.3 INPUTS

- **Business Analysis Approach:** incorporating the overall business analysis approach into the governance approach is necessary to ensure consistency across the approaches.
- **Stakeholder Engagement Approach:** identifying stakeholders and understanding their communication and collaboration needs is useful in determining their participation in the governance approach. The engagement approach may be updated based on the completion of the governance approach.

Figure 3.3.1: Plan Business Analysis Governance Input/Output Diagram



3.3.4 ELEMENTS

.1 DECISION MAKING

Decisions are made throughout the initiative. A stakeholder may serve in various roles in the decision-making process such as:

- participant in decision-making discussions,
- subject matter expert (SME) lending experience and knowledge to the decision-making process,
- reviewer of information, and
- approver of decisions.

The decision-making process defines what happens when teams cannot reach consensus, by identifying escalation paths and key stakeholders who hold final decision-making authority.

.2 CHANGE CONTROL PROCESS

When business analysts develop a change control process, they:

- **Determine the process for requesting changes:** specify which requirements and designs the change control process covers and determine whether it applies to all changes or only to changes of a specific size, cost, or level of effort. This process details the steps for proposing a change, when changes can be proposed, who can propose changes and how change requests are communicated.
- **Determine the elements of the change request:** identify the information to be included in a proposal to support decision making and implementation if it is approved.

Possible components to consider on a change request are:

- **Cost and time estimates:** for each area affected by the proposed change, the expected cost of change is estimated.
- **Benefits:** an explanation of how the change aligns with the initiative and business objectives to show how the change adds value. Benefits considered include both financial benefits and tactical benefits such as implications to scope, time, cost, quality, and resources.
- **Risks:** an analysis of risks to the initiative, the solution, or business objectives.
- **Priority:** the level of importance of the change relative to other factors such as organizational objectives, regulatory compliance requirements, and stakeholder needs.
- **Course(s) of action:** the course of action for the change includes an assessment of the components of the change request (cost, time, benefits, risks and priority). It is common to identify several alternative courses, including those recommended by the requester and by other stakeholders so decision makers can make a choice that will best serve the needs of the initiative.
- **Determine how changes will be prioritized:** the priority of the proposed change is established relative to other competing interests within the current initiative.
- **Determine how changes will be documented:** configuration management and traceability standards establish product baselines and version control practices that identify which baseline is affected by the change.
- **Determine how changes will be communicated:** how proposed changes, changes under review, and approved, declined, or deferred changes will be communicated to stakeholders.
- **Determine who will perform the impact analysis:** specify who is responsible for performing an analysis of the impacts the proposed change will have across the initiative.
- **Determine who will authorize changes:** include a designation of who can approve changes and what business analysis information their authority covers.

.3 PLAN PRIORITIZATION APPROACH

Timelines, expected value, dependencies, resource constraints, adopted methodologies, and other factors influence how requirements and designs are prioritized.

When planning the prioritization process, business analysts determine the:

- formality and rigour of the prioritization process,
- participants who will be involved in prioritization,
- process for deciding how prioritization will occur, including which prioritization techniques will be utilized, and
- criteria to be used for prioritization. For example, requirements may be prioritized based on cost, risk, and value.

The approach should also determine which stakeholders will have a role in prioritization.

.4 PLAN FOR APPROVALS

An approval formalizes the agreement between all stakeholders that the content and presentation of the requirements and designs are accurate, adequate, and contain sufficient detail to allow for continued progress to be made. The timing and frequency of approvals are dependent on the size and complexity of the change and associated risks of foregoing or delaying an approval.

The business analyst must determine the type of requirements and designs to be approved, the timing for the approvals, the process to follow to gain approval, and who will approve the requirements and designs.

When planning the appropriate approval process, business analysts consider the organizational culture and the type of information being approved. For example, new systems or processes for highly regulated industries such as financial, pharmaceutical, or healthcare are likely to require frequent and rigorous review and approval of very detailed specifications. For other types of initiatives, a less intensive approval process may be more appropriate and result in a faster implementation.

Planning for approvals also includes the schedule of events where approvals will occur and how they will be tracked. Stakeholder availability, attitude, and willingness to engage determine the efficiency of the approval process and may significantly affect delivery timelines.

3.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Performance Assessment	Provides results of previous assessments that should be reviewed and incorporated into all planning approaches.
Business Policies	Define the limits within which decisions must be made. They may be described by regulations, contracts, agreements, warranties, certifications or other legal obligations.
Current State Description	Provides the context within which the work needs to be completed. This information can help drive how to make better decisions.
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed, and can be used to help develop a framework that ensures sound business decision making.

3.3.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to generate an initial list of potential stakeholder names who may need approval roles in the defined governance process.
Document Analysis	Used to evaluate existing governance processes or templates.
Interviews	Used to identify possible decision-making, change control, approval, or prioritization approaches and participants with an individual or small group.
Item Tracking	Used to track any issues that arise when planning a governance approach.
Lessons Learned	Used to find if past initiatives have identified valuable experiences with governance that can be leveraged on current or future initiatives.
Organizational Modelling	Used to understand roles/responsibilities within the organization in an effort to define a governance approach that involves the right stakeholders.
Process Modelling	Used to document the process or method for governing business analysis.
Reviews	Used to review the proposed governance plan with key stakeholders.
Survey or Questionnaire	Used to identify possible decision-making, change control, approval, or prioritization approaches and participants.
Workshops	Used to identify possible decision-making, change control, approval, or prioritization approaches and participants within a team setting.

3.3.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	May be a possible source of a requested change or may be identified as needing to be involved in change discussions.
Project Manager	Works with the business analyst to ensure that overall project governance aligns with the business analysis governance approach.
Regulator	May impose rules or regulations that need to be considered when determining the business analysis governance plan. May also be a possible source of a requested change.
Sponsor	Can impose their own requirements for how business analysis information should be managed. Participates in change discussions and approves proposed changes.

3.3.8 OUTPUTS

- **Governance Approach:** identifies the stakeholders who will have the responsibility and authority to make decisions about business analysis work including who will be responsible for setting priorities and who will approve changes to business analysis information. It also defines the process that will be utilized to manage requirement and design changes across the initiative.

3.4 PLAN BUSINESS ANALYSIS INFORMATION MANAGEMENT

3.4.1 PURPOSE

The purpose of Plan Business Analysis Information Management is to develop an approach for how business analysis information will be stored and accessed.

3.4.2 DESCRIPTION

Business analysis information is comprised of all the information business analysts elicit, create, compile, and disseminate in the course of performing business analysis. Models, scope statements, stakeholder concerns, elicitation results, requirements, designs, and solution options are just a few examples. This includes requirements and designs, from lightweight user stories to formal requirement documents to functioning prototypes.

Information management entails identifying:

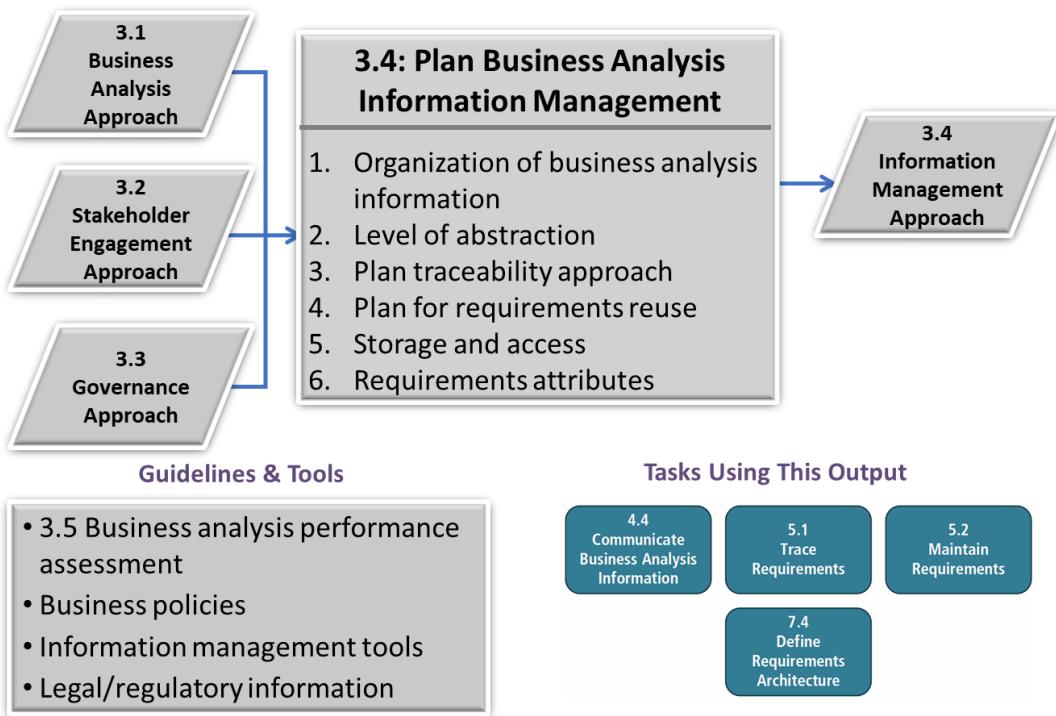
- how information should be organized,
- the level of detail at which information should be captured,
- any relationships between the information,
- how information may be used across multiple initiatives and throughout the enterprise,
- how information should be accessed and stored, and
- characteristics about the information that must be maintained.

Information management helps ensure that business analysis information is organized in a functional and useful manner, is easily accessible to appropriate personnel, and is stored for the necessary length of time.

3.4.3 INPUTS

- **Business Analysis Approach:** incorporating the overall business analysis approach into the information management approach is necessary to ensure consistency across the approaches.
- **Governance Approach:** defines how business analysts manage changes to requirements and designs, how decisions and approvals for business analysis deliverables will be made, and how priorities will be set.
- **Stakeholder Engagement Approach:** identifying stakeholders and understanding their communication and collaboration needs is useful in determining their specific information management needs.

Figure 3.4.1: Plan Business Analysis Information Management Input/Output Diagram



3.4.4 ELEMENTS

.1 ORGANIZATION OF BUSINESS ANALYSIS INFORMATION

Business analysts are responsible for organizing business analysis information in a manner that allows for efficient access and use. Information must be well structured to ensure it is not difficult to locate, conflicts with other information, or is needlessly duplicated.

The business analyst determines how best to structure and organize the business analysis information at the start of an initiative. This involves taking into consideration the type and amount of information to be collected, the stakeholder's access and usage needs, and the size and complexity of the change. Relationships among the types of information must be defined to assist in managing the effect of new or changed information in the future.

.2 LEVEL OF ABSTRACTION

Level of abstraction describes the breadth and depth of the information being provided. Representations of information may range from highly conceptual or summarized to very detailed. In determining how much detail each stakeholder may require as the initiative evolves, consideration is given to the needs of the stakeholders, the complexity of what is being explained, and the importance of the change. Rather than present the same information to all stakeholders, business analysts should present information with appropriate breadth and level of detail based on each stakeholder's role. Business analysis information regarding a topic of significant importance or high level of risk is frequently represented in greater detail.

.3 PLAN TRACEABILITY APPROACH

The traceability approach is based on:

- the complexity of the domain,
- the number of views of requirements that will be produced,
- any requirement-related risks, organizational standards, applicable regulatory requirements, and
- an understanding of the costs and benefits involved with tracing.

Business analysts plan to ensure the approach is at a level of detail to add value without excessive overhead.

.4 PLAN FOR REQUIREMENTS REUSE

Reusing requirements can save an organization time, effort, and cost—provided the requirements are accessible and structured in a manner that supports their reuse. Requirements that are potential candidates for long-term use are those an organization must meet on an ongoing basis such as:

- regulatory requirements,
- contractual obligations,
- quality standards,
- service level agreements,
- business rules,
- business processes, or
- requirements describing products the enterprise produces.

Requirements may also be reused when describing common features or services that are used across multiple systems, processes, or programs.

To make requirements useful beyond the current change, business analysts plan ahead for requirements reuse by identifying how best to structure, store, and access requirements so they are usable and accessible for future business analysis efforts.

In order for requirements to be reused they must be clearly named, defined, and stored in a repository that is available to other business analysts.

.5 STORAGE AND ACCESS

Business analysis information can be stored in many ways. Storage decisions depend on many factors such as who must access the information, how often they need to access it, and what conditions must be present for access. Organizational standards and tool availability also influence storage and access decisions. The business analysis approach defines how various tools will be used on the initiative and how the information will be captured and stored within those tools. Tools may shape the selection of business analysis techniques, notations to be used, and the way that information is organized.

The repository may need to store information other than requirements and designs. It should be able to indicate the status of any stored information, and allow for modification of that information over time.

.6 REQUIREMENTS ATTRIBUTES

Requirements attributes provide information about requirements, and aid in the ongoing management of the requirements throughout the change. They are planned for and determined with the requirements themselves.

Requirements attributes allow business analysts to associate information with individual or related groups of requirements. The information documented by the attributes helps the team efficiently and effectively make trade-offs between requirements, identify stakeholders affected by potential changes, and understand the effect of a proposed change.

Some commonly used requirements attributes include:

- **Absolute reference:** a unique identifier. The reference is not altered or reused if the requirement is moved, changed, or deleted.
- **Author:** provides the name of the person who needs to be consulted should the requirement later be found to be ambiguous, unclear, or in conflict.
- **Complexity:** indicates how difficult the requirement will be to implement.
- **Ownership:** indicates the individual or group that needs the requirement or will be the business owner after the solution is implemented.
- **Priority:** indicates relative importance of requirements. Priority can refer to the relative value of a requirement or to the sequence in which it will be implemented.
- **Risks:** identifies uncertain events that may impact requirements.
- **Source:** identifies the origin of the requirement. The source is often consulted if the requirement changes or if more information regarding the requirement or the need that drove the requirement has to be obtained.
- **Stability:** indicates the maturity of the requirement.
- **Status:** indicates the state of the requirement, whether it is proposed, accepted, verified, postponed, cancelled, or implemented.
- **Urgency:** indicates how soon the requirement is needed. It is usually only necessary to specify this separately from the priority when a deadline exists for implementation.

3.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Performance Assessment	Provides results of previous assessments that should be reviewed and incorporated into all planning approaches.
Business Policies	Define the limits within which decisions must be made. They may be described by regulations, contracts, agreements, warranties, certifications, or other legal obligations.
Information Management Tools	Each organization uses some tools to store, retrieve, and share business analysis information. These may be as simple as a whiteboard, or as complex as a global wiki or robust requirements management tool.
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed, and helps determine how business analysis information will be managed.

3.4.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to help stakeholders uncover their business analysis information management needs.
Interviews	Used to help specific stakeholders uncover their business analysis information management needs.

Item Tracking	Used to track issues with current information management processes.
Lessons Learned	Used to create a source of information for analyzing approaches for efficiently managing business analysis information.
Mind Mapping	Used to identify and categorize the kinds of information that need to be managed.
Process Modelling	Used to document the process or method for managing business analysis information.
Survey or Questionnaire	Used to ask stakeholders to provide input into defining business analysis information management.
Workshops	Used to uncover business analysis information management needs in a group setting.

3.4.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	May need to access and work with business analysis information, and will be interested in a more specific view of business analysis information which relates to their area of expertise.
Regulator	May define rules and processes related to information management.
Sponsor	Reviews, comments on, and approves business analysis information.

3.4.8 OUTPUTS

- **Information Management Approach:** includes the defined approach for how business analysis information will be stored, accessed, and utilized during the change and after the change is complete.

3.5 IDENTIFY BUSINESS ANALYSIS PERFORMANCE IMPROVEMENTS

3.5.1 PURPOSE

The purpose of Identify Business Analysis Performance Improvements is to assess business analysis work and to plan to improve processes where required.

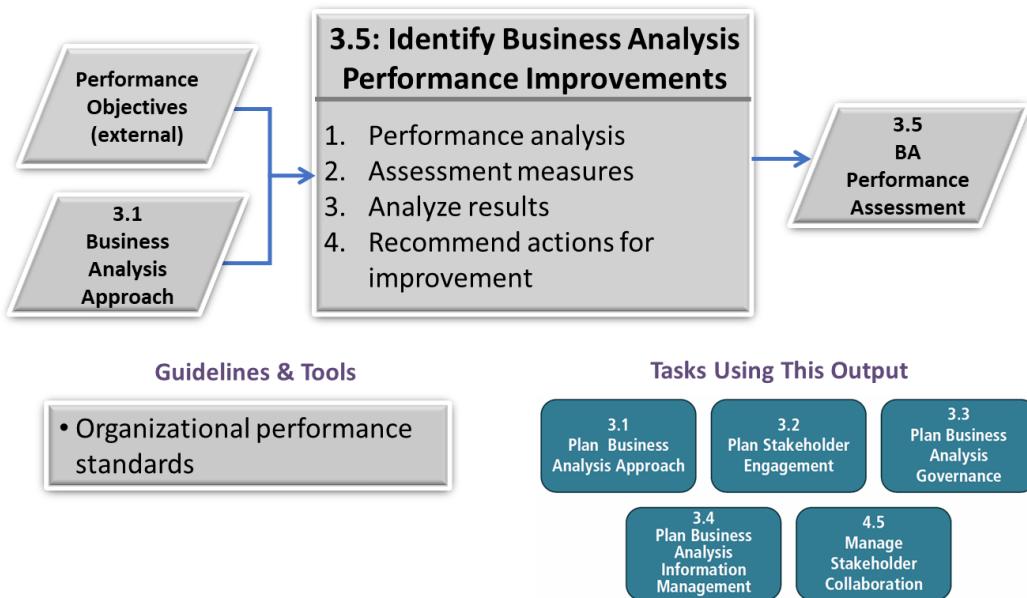
3.5.2 DESCRIPTION

To monitor and improve performance, it is necessary to establish the performance measures, conduct the performance analysis, report on the results of the analysis, and identify any necessary preventive, corrective, or developmental actions. Performance analysis should occur throughout an initiative. Once potential performance improvements are identified, they become guidelines for the next time a task is executed.

3.5.3 INPUTS

- **Business Analysis Approach:** identifies business analysis deliverables that will be produced, activities that will need to be performed (including when they will be performed and who will be performing them), and techniques that will be used.
- **Performance Objectives (external):** describe the desired performance outcomes that an enterprise or organization is hoping to achieve.

Figure 3.5.1: Identify Business Analysis Performance Improvements Input/Output Diagram



3.5.4 ELEMENTS

.1 PERFORMANCE ANALYSIS

What constitutes effective business analysis work depends on the context of a particular organization or initiative. Reports on business analysis performance can be informal and verbal, or they may include formal documentation. Reports on business analysis performance are designed and tailored to meet the needs of the various types of reviewers.

.2 ASSESSMENT MEASURES

If current measures exist, the business analyst may leverage them or determine new measures. The business analyst may also elicit assessment measures from stakeholders.

Performance measures may be based on deliverable due dates as specified in the business analysis plan, metrics such as the frequency of the changes to business analysis work products, the number of review cycles required, task efficiency, or qualitative feedback from stakeholders and peers regarding the business analyst's deliverables. Appropriate performance measures enable the business analyst to determine when problems are occurring that may affect the performance of business analysis or identify opportunities for improvement. Measures may be both quantitative and qualitative. Qualitative measures are subjective and can be heavily influenced by the stakeholder's attitudes, perceptions, and other subjective criteria.

Note: All performance metrics will encourage certain behaviours and discourage others. Poorly chosen metrics may drive behaviour that is detrimental to the enterprise as a whole.

Some possible measures are:

- **Accuracy and Completeness:** determine whether the business analyst work products were correct and relevant when delivered, or whether ongoing revisions were needed to gain acceptance by stakeholders.
- **Knowledge:** assess whether the business analyst had the skills and/or experience to perform the assigned task.
- **Effectiveness:** assess whether the business analyst work products were easy to use as standalone deliverables or whether they required extensive explanation in order to be understood.
- **Organizational Support:** assess whether there were adequate resources available to complete business analysis activities as needed.
- **Significance:** consider the benefit obtained from the work products and assess whether the cost, time, and resource investments expended to produce the work products were justified for the value they delivered.
- **Strategic:** look at whether business objectives were met, problems were solved, and improvements were achieved.
- **Timeliness:** evaluate whether the business analyst delivered the work on time per stakeholder expectations and schedule.

.3 ANALYZE RESULTS

The business analysis process and deliverables are compared against the set of defined measures. The analysis may be performed on the business analysis process, the resources involved, and the deliverables.

Performance may be determined from the point of view of the stakeholders who are the recipients of the business analysis work. Other times a personnel manager or a Centre of Excellence may make this determination and provide assessments. All stakeholders may have input in assessing the value of the business analysis work but organizations may differ in terms of who has the authority to set the targets against which performance is measured.

.4 RECOMMEND ACTIONS FOR IMPROVEMENT

Once the analysis of performance results is complete, the business analyst engages the appropriate stakeholders to identify the following actions:

- **Preventive:** reduces the probability of an event with a negative impact.
- **Corrective:** establishes ways to reduce the negative impact of an event.
- **Improvement:** establishes ways to increase the probability or impact of events with a positive impact.

These actions are likely to result in changes to the business analysis approach, repeatable processes, and tools.

3.5.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Organizational	May include performance metrics or expectations for business

Performance Standards	analysis work mandated by the organization.
------------------------------	---

3.5.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to generate ideas for improvement opportunities.
Interviews	Used to gather assessments of business analysis performance.
Item Tracking	Used to track issues that occur during the performance of business analysis for later resolution.
Lessons Learned	Used to identify recommended changes to business analysis processes, deliverables, templates, and other organizational process assets that can be incorporated into the current initiative and future work.
Metrics and Key Performance Indicators (KPIs)	Used to determine what metrics are appropriate for assessing business analysis performance and how they may be tracked.
Observation	Used to witness business analysis performance.
Process Analysis	Used to analyze existing business analysis processes and identify opportunities for improvement.
Process Modelling	Used to define business analysis processes and understand how to improve those processes to reduce problems from hand-offs, improve cycle times, or alter how business analysis work is performed to support improvements in downstream processes.
Reviews	Used to identify changes to business analysis processes and deliverables that can be incorporated into future work.
Risk Analysis and Management	Used to identify and manage potential conditions or events that may impact business analysis performance.
Root Cause Analysis	Used to help identify the underlying cause of failures or difficulties in accomplishing business analysis work.
Survey or Questionnaire	Used to gather feedback from stakeholders about their satisfaction with business analysis activities and deliverables.
Workshops	Used to gather assessments of business analysis performance and generate ideas for improvement opportunities.

3.5.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	Should be informed about the business analysis activities in order to set expectations regarding their involvement in the work and to elicit their feedback regarding possible improvements to the approach.
Project Manager	Is accountable for the success of a project and must be kept informed of the current status of business analysis work. If potential problems or opportunities for improvement are identified, the project manager must be consulted before changes are implemented to assess whether those changes will

	have an impact on the project. They may also deliver reports on business analysis performance to the sponsor and other stakeholders.
Sponsor	May require reports on business analysis performance to address problems as they are identified. A manager of business analysts may also sponsor initiatives to improve the performance of business analysis activities.

3.5.8 OUTPUTS

- **Business Analysis Performance Assessment:** includes a comparison of planned versus actual performance, identifying the root cause of variances from the expected performance, proposed approaches to address issues, and other findings to help understand the performance of business analysis processes.

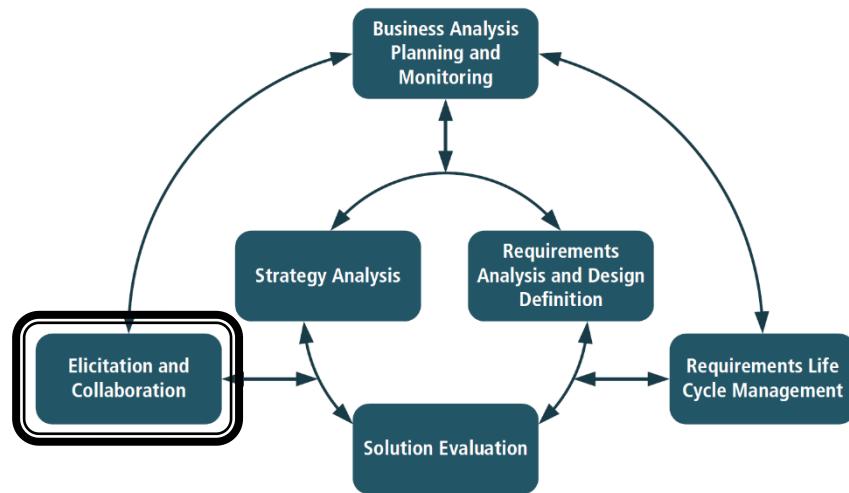
4. Elicitation and Collaboration

The Elicitation and Collaboration knowledge area describes the tasks that business analysts perform to obtain information from stakeholders and confirm the results. It also describes the communication with stakeholders once the business analysis information is assembled.

Elicitation is the drawing forth or receiving of information from stakeholders or other sources. It is the main path to discovering requirements and design information, and might involve talking with stakeholders directly, researching topics, experimenting, or simply being handed information. Collaboration is the act of two or more people working together towards a common goal. The Elicitation and Collaboration knowledge area describes how business analysts identify and reach agreement on the mutual understanding of all types of business analysis information. Elicitation and collaboration work is never a 'phase' in business analysis; rather, it is ongoing as long as business analysis work is occurring.

Elicitation and collaboration can be planned, unplanned, or both. Planned activities such as workshops, experiments, and/or surveys can be structured and organized in advance. Unplanned activities happen in the moment without notice, such as last-minute or 'just in time' collaboration or conversations. Business analysis information derived from an unplanned activity may require deeper exploration through a planned activity.

Eliciting business analysis information is not an isolated activity. Information is elicited while performing any task that includes interaction with stakeholders and while the business analyst is performing independent analytical work. Elicitation may trigger additional elicitation for details to fill in gaps or increase understanding.



The Elicitation and Collaboration knowledge area is composed of the following tasks:

PC⁴

1. **Prepare for Elicitation**
2. **Conduct Elicitation**
3. **Confirm Elicitation Results**

- 4. Communicate Business Analysis Information**
- 5. Manage Stakeholder Collaboration**

The Core Concept Model in Elicitation and Collaboration

The *Business Analysis Core Concept Model™ (BACCM™)* describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Elicitation and Collaboration.

Core Concept	During Elicitation and Collaboration, business analysts...
Change	Use a variety of elicitation techniques to fully identify the characteristics of the change including concerns that stakeholders have about the change. The change itself may determine the appropriate types and extent of elicitation and collaboration.
Need	Elicit, confirm, and communicate needs and supporting business analysis information. As elicitation is iterative and incremental, the understanding of needs may evolve over time.
Solution	Elicit, confirm, and communicate necessary or desired characteristics of proposed solutions.
Stakeholder	Manage the collaboration with the stakeholders who participate in the business analysis work. All stakeholders may participate in different roles and at different times during a change.
Value	Collaborate with stakeholders to assess the relative value of information provided through elicitation, and apply a variety of techniques to confirm and communicate that value.
Context	Apply a variety of elicitation techniques to identify business analysis information about the context that may affect the change.

4.1 PREPARE FOR ELICITATION

4.1.1 PURPOSE

The purpose of Prepare for Elicitation is to understand the scope of the elicitation activity, select appropriate techniques, and plan for (or procure) appropriate supporting materials and resources.

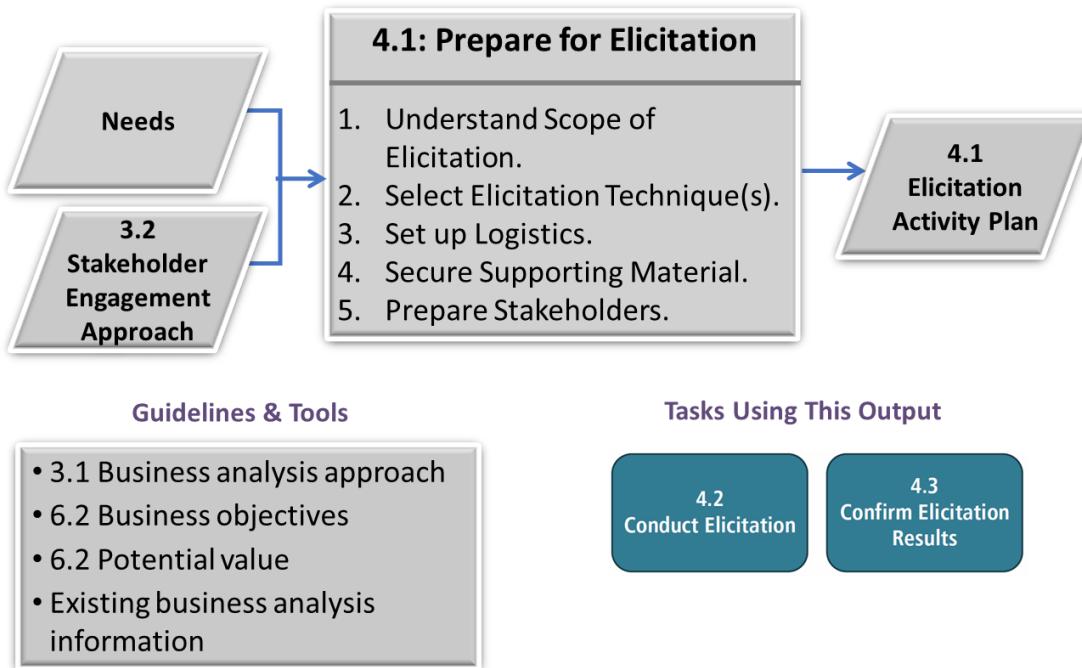
4.1.2 DESCRIPTION

Business analysts prepare for elicitation by defining the desired outcomes of the activity, considering the stakeholders involved and the goals of the initiative. This includes determining which work products will be produced using the elicitation results, deciding which techniques are best suited to produce those results, establishing the elicitation logistics, identifying any supporting materials needed, and understanding circumstances to foster collaboration during an elicitation activity.

4.1.3 INPUTS

- **Needs:** guides the preparation in terms of the scope and purpose of elicitation activities. Elicitation can be used to discover the needs, but in order to get started there must be some need that exists—even if it has not yet been fully elicited or understood.
- **Stakeholder Engagement Approach:** understanding stakeholders' communication and collaboration needs helps plan and prepare appropriate and effective elicitation events.

Figure 4.1.1: Prepare for Elicitation Input/Output Diagram



4.1.4 ELEMENTS

.1 UNDERSTAND THE SCOPE OF ELICITATION

To determine the type of business analysis information to be discovered during the elicitation activity and the techniques that may be used, business analysts consider:

- business domain,
- overall corporate culture and environment,
- stakeholder locations,
- stakeholders who are involved and their group dynamics,
- expected outputs the elicitation activities will feed,
- skills of the business analysis practitioner,
- other elicitation activities planned to complement this one,
- strategy or solution approach,
- scope of future solution, and
- possible sources of the business analysis information that might feed into the specific elicitation activity.

Understanding the scope of the elicitation activity allows business analysts to respond if the activity strays from the intended scope. It also allows them to recognize if people and materials are not available in time, and when the activity is complete.

.2 SELECT ELICITATION TECHNIQUES

In most cases, multiple techniques are used during an elicitation activity. The techniques used depend on cost and time constraints, the types of business analysis information sources and their access, the culture of the organization, and the desired outcomes. The business analyst may also factor in the needs of the stakeholders, their availability, and their location (co-located or dispersed). Choosing the right techniques and ensuring each technique is performed correctly is extremely important to the success of the elicitation activity. When selecting elicitation techniques, business analysts consider:

- techniques commonly used in similar initiatives,
- techniques specifically suited to the situation, and
- the tasks needed to prepare, execute, and complete each technique.

Due to changing dynamics and situations, the business analyst may be required to adjust the initial selections by incorporating more appropriate techniques. A thorough understanding of the variety of techniques available assists the business analyst in adapting to changing circumstances.

.3 SET UP LOGISTICS

Logistics are planned prior to an elicitation activity. The logistics for each elicitation activity include identifying:

- the activity's goals,
- participants and their roles,
- scheduled resources, including people, rooms, and tools,
- locations,
- communication channels,
- techniques, and
- languages used by stakeholders (oral and written).

The logistics may also involve creating an agenda if other stakeholders are involved.

.4 SECURE SUPPORTING MATERIAL

Business analysts identify sources of information that are needed to conduct the elicitation activity. There might be a great deal of information needed to conduct elicitation including people, systems, historical data, materials and documents. Documents could include existing system documents, relevant business rules, organizational policies, regulations, and contracts. Supporting materials might also take the form of outputs of analysis work, such as draft versions of analysis. Business analysts procure or develop the materials and tools needed. Additional planning for experimental elicitation might be required if novel tools, equipment, or techniques are going to be used.

.5 PREPARE STAKEHOLDERS

Business analysts may need to educate stakeholders on how an elicitation technique works or what information is needed. It may be helpful to explain an elicitation technique to stakeholders not involved in the activity to help them understand the validity and relevance of the information elicited. Stakeholders may be unresponsive or challenging during an elicitation activity if they feel that it is

not aligned to their individual objectives, don't understand the purpose, or are confused about the process. In preparing for elicitation, the business analyst should ensure that there is buy-in from all necessary stakeholders.

Business analysts may also prepare stakeholders by requesting that they review supporting materials prior to the elicitation activity in order to make it as effective as possible. An agenda might be provided in advance to support stakeholders in coming prepared to the activity with the necessary frame of mind and information.

Eliciting through research or exploration may be a solo activity for the business analyst and not require preparing other stakeholders.

4.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Sets the general strategy to be used to guide the business analysis work. This includes the general methodology, types of stakeholders and how they should be involved, list of stakeholders, timing of the work, expected format and level of detail of elicitation results, and identified challenges and uncertainties.
Business Objectives	Describe the desired direction needed to achieve the future state. They can be used to plan and prepare elicitation events, and to develop supporting materials.
Existing Business Analysis Information	May provide a better understanding of the goals of the elicitation activity, and aid in preparing for elicitation.
Potential Value	Describes the value to be realized by implementing the proposed future state, and can be used to shape elicitation events.

4.1.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to collaboratively identify and reach consensus about which sources of business analysis information should be consulted and which elicitation techniques might be most effective.
Data Mining	Used to identify information or patterns that require further investigation.
Document Analysis	Used to identify and assess candidate sources of supporting materials.
Estimation	Used to estimate the time and effort required for the elicitation and the associated cost.
Interviews	Used to identify concerns about the planned elicitation, and can be used to seek authority to proceed with specific options.
Mind Mapping	Used to collaboratively identify and reach consensus about which sources of business analysis information should be consulted and which elicitation techniques might be most effective

Risk Analysis and Management	Used to identify, assess, and manage conditions or situations that could disrupt the elicitation, or affect the quality and validity of the elicitation results. The plans for the elicitation should be adjusted to avoid, transfer, or mitigate the most serious risks.
Stakeholder List, Map, or Personas	Used to determine who should be consulted while preparing for the elicitation, who should participate in the event, and the appropriate roles for each stakeholder.

4.1.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	provides supporting materials as well as guidance about which other sources of business analysis information to consult. May also help to arrange research, experiments, and facilitated elicitation.
Project Manager	ensures that the appropriate people and resources are available to conduct the elicitation.
Sponsor	has the authority to approve or deny a planned elicitation event, and to authorize and require the participation of specific stakeholders

4.1.8 OUTPUTS

- **Elicitation Activity Plan:** used for each elicitation activity. It includes logistics, scope of the elicitation activity, selected techniques, and supporting materials.

4.2 CONDUCT ELICITATION

4.2.1 PURPOSE

The purpose of Conduct Elicitation is to draw out, explore, and identify information relevant to the change.

4.2.2 DESCRIPTION

There are three common types of elicitation:

- **Collaborative:** involves direct interaction with stakeholders, and relies on their experiences, expertise, and judgment.
- **Research:** involves systematically discovering and studying information from materials or sources that are not directly known by stakeholders involved in the change. Stakeholders might still participate in the research. Research can include data analysis of historical data to identify trends or past results.
- **Experiments:** involves identifying information that could not be known without some sort of controlled test. Some information cannot be drawn from people or documents—because it is unknown. Experiments can help discover this kind of information. Experiments include observational studies, proofs of concept, and prototypes.

One or more elicitation techniques may be used to produce the desired outcome within the scope of elicitation.

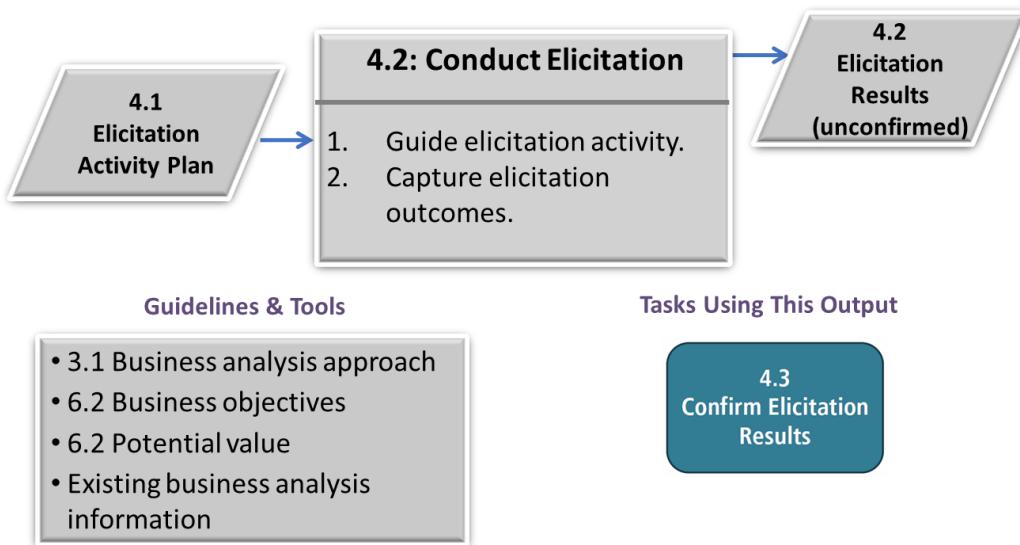
Stakeholders may collaborate in elicitation by:

- participating and interacting during the elicitation activity, and
- researching, studying, and providing feedback on documents, systems, models, and interfaces.

4.2.3 INPUTS

- **Elicitation Activity Plan:** includes the planned elicitation activities and techniques, activity logistics (for example, date, time, location, resources, agenda), scope of the elicitation activity, and available sources of background information.

Figure 4.2.1: Conduct Elicitation Input/Output Diagram



4.2.4 ELEMENTS

.1 GUIDE ELICITATION ACTIVITY

Understanding the proposed representations of business analysis information, which were defined in planning, helps ensure that the elicitation activities are focused on producing the intended information at the desired level of detail. This applies to each instance of an elicitation activity throughout a change and may vary based on the activity. In order to help guide and facilitate towards the expected outcomes, business analysts consider:

- the elicitation activity goals and agenda,
- scope of the change,
- what forms of output the activity will generate,
- what other representations the activity results will support,

- how the output integrates into what is already known,
- who provides the information,
- who will use the information, and
- how the information will be used.

While most of these are considered when planning for the elicitation, they are also all important while performing the elicitation activity in order to keep it on track and achieve its goal. For example, stakeholders might have discussions that are out of scope for the activity or change, and the business analyst needs to recognize that in the moment to determine the next step; either acknowledge it and continue, or guide the conversation differently.

The business analyst also uses this information to determine when there has been sufficient elicitation, in order to stop the activity.

.2 CAPTURE ELICITATION OUTCOMES

Conducting elicitation is frequently iterative and takes place in a series of sessions—in parallel or in sequence—according to the scope of the elicitation activity. If the elicitation activity is unplanned, outcomes are captured and integrated into the appropriate planned outcomes.

Capturing the elicitation outcomes helps to ensure that the information produced during elicitation activities is recorded for later reference and use.

4.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Influences how each elicitation activity is performed, as it identifies the types of outputs that will be needed based on the approach.
Existing Business Analysis Information	May guide the questions posed during elicitation and the approach used to draw out information from various stakeholders.
Stakeholder Engagement Approach	Provides collaboration and communication approaches that might be effective during elicitation.
Supporting Materials	Includes any materials to prepare both the business analyst and participants before elicitation, as well as any information, tools, or equipment to be used during the elicitation.

4.2.6 TECHNIQUES

Techniques	Usage
Benchmarking and Market Analysis	Used as a source of business analysis information by comparing a specific process, system, product, service, or structure with some external baseline, such as a similar organization or baseline provided by an industry association. Market analysis is used to determine what customers want and what competitors provide.
Brainstorming	Used to generate many ideas from a group of stakeholders in a short period, and to organize and prioritize those ideas.
Business Rules Analysis	Used to identify the rules that govern decisions in an organization and that define, constrain, or enable organizational

	operations.
Collaborative Games	Used to develop a better understanding of a problem or to stimulate creative solutions.
Concept Modelling	Used to identify key terms and ideas of importance and define the relationships between them
Data Mining	Used to identify relevant information and patterns
Data Modelling	Used to understand entity relationships during elicitation.
Document Analysis	Used to review existing systems, contracts, business procedures and policies, standards, and regulations.
Focus Groups	Used to identify and understand ideas and attitudes from a group.
Interface Analysis	Used to understand the interaction, and characteristics of that interaction, between two entities, such as two systems, two organizations, or two people or roles.
Interviews	Used to ask questions of stakeholders to uncover needs, identify problems, or discover opportunities.
Mind Mapping	Used to generate many ideas from a group of stakeholders in a short period, and to organize and prioritize those ideas.
Observation	Used to gain insight about how work is currently done, possibly in different locations and in different circumstances.
Process Analysis	Used to understand current processes and to identify opportunities for improvement in those processes.
Process Modelling	Used to elicit processes with stakeholders during elicitation activities.
Prototyping	Used to elicit and validate stakeholders' needs through an iterative process that creates a model of requirements or designs.
Survey or Questionnaire	Used to elicit business analysis information, including information about customers, products, work practices, and attitudes, from a group of people in a structured way and in a relatively short period of time.
Workshops	Used to elicit business analysis information, including information about customers, products, work practices, and attitudes, from a group of people in a collaborative, facilitated way.

4.2.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Will provide valuable business analysis information during elicitation.
Domain Subject Matter Expert	Has expertise in some aspect of the situation and can provide the required business analysis information. Often guides and assists the business analyst in identifying appropriate research sources, and may help to arrange research, experiments, and

	facilitated elicitation.
End User	The user of existing and future solutions, who should participate in elicitation.
Implementation Subject Matter Expert	Designs and implements a solution and provides specialist expertise, and can participate in elicitation by asking clarifying questions and offering alternatives.
Sponsor	Authorizes and ensures that the stakeholders necessary to participate in elicitation are involved.
Any stakeholders	Could have relevant knowledge or experience to participate in elicitation activities.

4.2.8 OUTPUTS

- **Elicitation Results (unconfirmed):** captured information in a format that is specific to the elicitation activity.

4.3 CONFIRM ELICITATION RESULTS

4.3.1 PURPOSE

The purpose of Confirm Elicitation Results is to check the information gathered during an elicitation session for accuracy and consistency with other information.

4.3.2 DESCRIPTION

Elicited information is confirmed to identify any problems and resolve them before resources are committed to using the information. This review may discover errors, omissions, conflicts, and ambiguity.

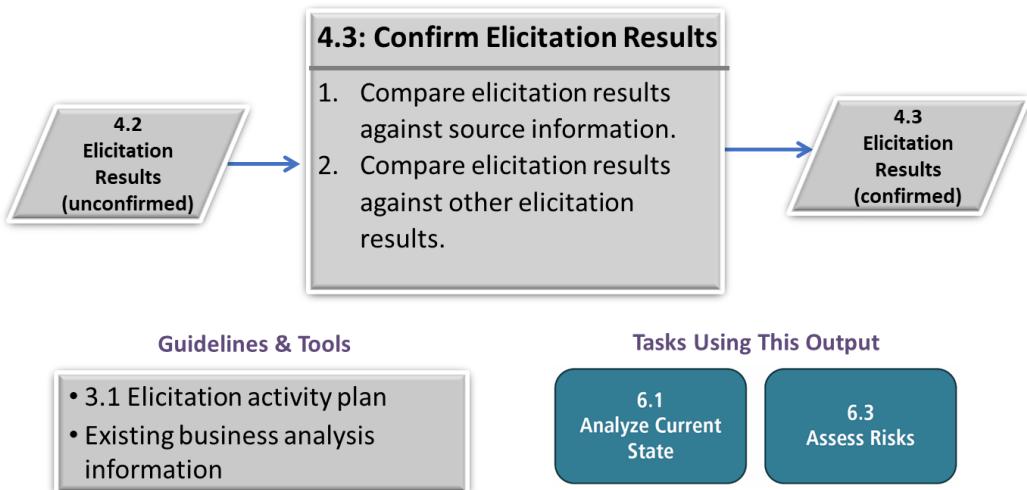
The elicitation results can be compared against their source and other elicitation results to ensure consistency. Collaboration with stakeholders might be necessary to ensure their inputs are correctly captured and that they agree with the results of non-facilitated elicitation. If information is not correct, the business analyst determines what is correct, which can require more elicitation. Committing resources to business analysis activities based on unconfirmed elicitation results may mean stakeholder expectations are not met. If the results are inconsistent, additional elicitation might need to be conducted to resolve the discrepancies.

Confirming the elicitation results is a much less rigorous and formal review than occurs during analysis.

4.3.3 INPUTS

- **Elicitation Results (unconfirmed):** capture information in a format specific to the elicitation activity.

Figure 4.3.1: Confirm Elicitation Results



4.3.4 ELEMENTS

.1 COMPARE ELICITATION RESULTS AGAINST SOURCE INFORMATION

Task Conduct Elicitation describes sources from which elicitation results may be derived, including documents and stakeholder knowledge. The business analyst may lead follow-up meetings where stakeholders correct the elicitation results. Stakeholders may also confirm the elicitation results independently.

.2 COMPARE ELICITATION RESULTS AGAINST OTHER ELICITATION RESULTS

Business analysts compare results collected through multiple elicitation activities to confirm that the information is consistent and accurately represented. As comparisons are drawn, business analysts identify variations in results and resolve them in collaboration with stakeholders. Comparisons may also be made with historical data to confirm more recent elicitation results.

Inconsistencies in elicitation results are often uncovered when business analysts develop specifications and models. These models may be developed during an elicitation activity to improve collaboration.

4.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Elicitation Activity Plan	Used to guide which alternative sources and which elicitation results are to be compared.
Existing Business Analysis Information	Can be used to confirm the results of elicitation activities or to develop additional questions to draw out more detailed information.

4.3.6 TECHNIQUES

Techniques	Usage
------------	-------

Document Analysis	Used to confirm elicitation results against source information or other existing documents.
Interviews	Used to confirm the business analysis information and to confirm that the integration of that information is correct.
Reviews	Used to confirm a set of elicitation results. Such reviews could be informal or formal depending on the risks of not having correct, useful, and relevant information.
Workshops	Used to conduct reviews of the drafted elicitation results using any level of formality. A predetermined agenda, scripts, or scenario test may be used to walk through the elicitation results, and feedback is requested from the participants and recorded.

4.3.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Experts	People with substantial knowledge, experience, or expertise about the business analysis information being elicited, or about the change or the solution, help to confirm that elicitation results are correct, and can help to identify omissions, inconsistencies and conflicts in elicitation results. They can also confirm that the right business analysis information has been elicited.
Any stakeholder	All types of stakeholders may need to participate in confirming elicitation results.

4.3.8 OUTPUTS

- **Elicitation Results (confirmed):** integrated output that the business analyst and other stakeholders agree correctly reflects captured information and confirms that it is relevant and useful as an input to further work.

4.4 COMMUNICATE BUSINESS ANALYSIS INFORMATION

4.4.1 PURPOSE

The purpose of Communicate Business Analysis Information is to ensure stakeholders have a shared understanding of business analysis information.

4.4.2 DESCRIPTION

Business analysts must communicate appropriate information to stakeholders at the right time and in formats that meet their needs. Consideration is given to expressing the information in language, tone, and style that is appropriate to the audience.

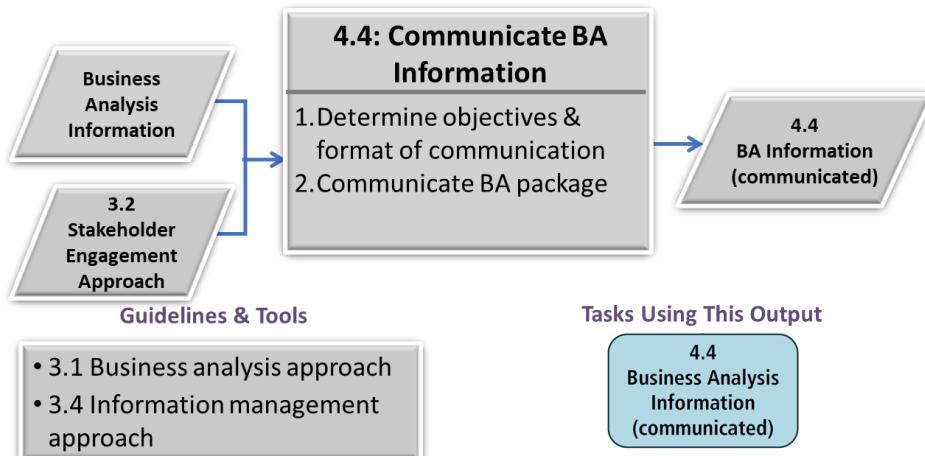
Communication of business analysis information is bi-directional and iterative. It involves determining the recipients, content, purpose, context, and expected outcomes. Task Plan Stakeholder Engagement evaluates communication needs and plans anticipated messages.

Communicating information does not simply involve pushing information out and assuming it was received and understood. Business analysts engage stakeholders to ensure they understand the information and gain agreement. The business analyst acts on any disagreements. The method of delivering the information may need to change if the stakeholders are not receiving or understanding it. Multiple forms of communication might be required for the same information.

4.4.3 INPUTS

- **Business Analysis Information:** any kind of information at any level of detail that is used as an input or output of business analysis work. Business analysis information becomes an input for this task when the need is discovered to communicate the information to additional stakeholders.
- **Stakeholder Engagement Approach:** describes stakeholder groups, roles, and general needs regarding communication of business analysis information.

Figure 4.4.1: Communicate Business Analysis Information Input/Output Diagram



4.4.4 ELEMENTS

.1 DETERMINE OBJECTIVES AND FORMAT OF COMMUNICATION

Business analysis information packages may be prepared for a number of reasons including—but not limited to—the following:

- communication of requirements and designs to stakeholders,
- early assessment of quality and planning,
- evaluation of possible alternatives,
- formal reviews and approvals,
- inputs to solution design,
- conformance to contractual and regulatory obligations, and
- maintenance for reuse.

The primary goal of developing a package is to convey information clearly and in usable format for continuing change activities. To help decide how to present requirements, business analysts ask the following types of questions:

- Who is the audience of the package?
- What will each type of stakeholder understand and need from the communication?
- What is each stakeholder's preferred style of communication or learning?
- What information is important to communicate?
- Are the presentation and format of the package, and the information contained in the package, appropriate for the type of audience?
- How does the package support other activities?
- Are there any regulatory or contractual constraints to conform to?

Possible forms for packages may include:

- **Formal Documentation:** is usually based on a template used by the organization and may include text, matrices, or diagrams. It provides a stable, easy to use, long-term record of the information.
- **Informal Documentation:** may include text, diagrams, or matrices that are used during a change but are not part of a formal organizational process.
- **Presentations:** deliver a high-level overview appropriate for understanding goals of a change, functions of a solution, or information to support decision making.

Consideration is given to the best way to combine and present the materials to convey a cohesive and effective message to one or more stakeholder groups. Packages can be stored in different online or offline repositories, including documents or tools.

.2 COMMUNICATE BUSINESS ANALYSIS PACKAGE

The purpose of communicating the business analysis package is to provide stakeholders with the appropriate level of detail about the change so they can understand the information it contains. Stakeholders are given the opportunity to review the package, ask questions about the information, and raise any concerns they may have.

Selecting the appropriate communication platform is also important. Common communication platforms include:

- **Group collaboration:** used to communicate the package to a group of relevant stakeholders at the same time. It allows immediate discussion about the information and related issues.
- **Individual collaboration:** used to communicate the package to a single stakeholder at a time. It can be used to gain individual understanding of the information when a group setting is not feasible, most productive, or going to yield the best results.
- **E-mail or other non-verbal methods:** used to communicate the package when there is a high maturity level of information that will need little or no verbal explanation to support it.

4.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Describes how the various types of information will be disseminated rather than what will be disseminated. It describes the level of detail and formality required, frequency of the

	communications, and how communications could be affected by the number and geographic dispersion of stakeholders.
Information Management Approach	Helps determine how business analysis information will be packaged and communicated to stakeholders.

4.4.6 TECHNIQUES

Techniques	Usage
Interviews	Used to individually communicate information to stakeholders.
Reviews	Used to provide stakeholders with an opportunity to express feedback, request required adjustments, understand required responses and actions, and agree or provide approvals. Reviews can be used during group or individual collaboration.
Workshops	Used to provide stakeholders with an opportunity to express feedback and to understand required adjustments, responses, and actions. They are also useful for gaining consensus and providing approvals. Typically used during group collaboration.

4.4.7 STAKEHOLDERS

Stakeholder	Contribution
End User	Needs to be communicated with frequently so they are aware of relevant business analysis information.
Customer	Needs to be communicated with frequently so they are aware of relevant business analysis information.
Domain Subject Matter Expert	Needs to understand the business analysis information as part of confirming and validating it throughout the change initiative.
Implementation Subject Matter Expert	Needs to be aware of and understand the business analysis information, particularly requirements and designs, for implementation purposes.
Tester	Needs to be aware of and understand the business analysis information, particularly requirements and designs for testing purposes.
Any stakeholder	All types of stakeholders will likely need to be communicated with at some point during the change initiative.

4.4.8 OUTPUTS

- **Business Analysis Information (communicated):** business analysis information is considered communicated when the target stakeholders have reached an understanding of its content and implications.

4.5 MANAGE STAKEHOLDER COLLABORATION

4.5.1 PURPOSE

The purpose of Manage Stakeholder Collaboration is to encourage stakeholders to work towards a common goal.

4.5.2 DESCRIPTION

Business analysis work lends itself to many collaboration opportunities between groups of stakeholders on the business analysis work products. Stakeholders hold various degrees of influence and authority over the approval of work products, and are also an important source of needs, constraints, and assumptions. As the business analysis work progresses, the business analyst identifies stakeholders, confirms their roles, and communicates with them to ensure that the right stakeholders participate at the right times and in the appropriate roles.

Managing stakeholder collaboration is an ongoing activity. Although managing stakeholder collaboration begins once stakeholders have been identified and analyzed, new stakeholders may be identified at any point during an initiative. As new stakeholders are identified, their role, influence, and relationship to the initiative are analyzed. Each stakeholder's role, responsibility, influence, attitude, and authority may change over time.

The more significant the impact of the change or its visibility within the organization, the more attention is directed to managing stakeholder collaboration. Business analysts manage stakeholder collaboration to capitalize on positive reactions, and mitigate or avoid negative reactions. The business analyst should constantly monitor and assess each stakeholder's attitude to determine if it might affect their involvement in the business analysis activities.

Poor relationships with stakeholders can have many detrimental effects on business analysis, including:

- failure to provide quality information,
- strong negative reactions to setbacks and obstacles,
- resistance to change,
- lack of support for, and participation in, business analysis work, and
- business analysis information being ignored.

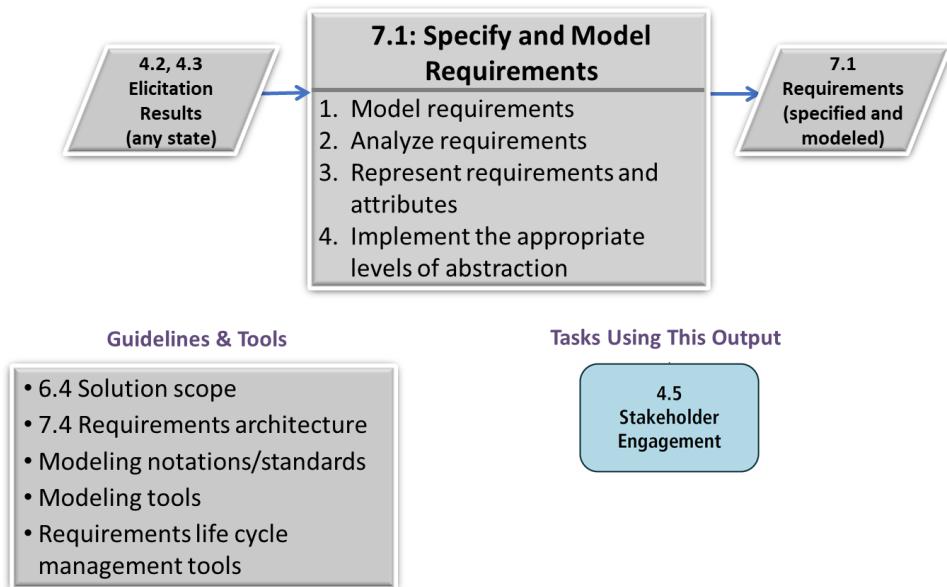
These effects can be modified in part through strong, positive, and trust-based relationships with stakeholders. Business analysts actively manage relationships with stakeholders who:

- provide services to the business analyst, including inputs to business analysis tasks and other support activities,
- depend on services provided by the business analyst, including outputs of business analysis tasks, and
- participate in the execution of business analysis tasks.

4.5.3 INPUTS

- **Stakeholder Engagement Approach:** describes the types of expected engagement with stakeholders and how they might need to be managed.
- **Business Analysis Performance Assessment:** provides key information about the effectiveness of business analysis tasks being executed, including those focused on stakeholder engagement.
-

Figure 4.5.1: Manage Stakeholder Collaboration Input/Output Diagram



4.5.4 ELEMENTS

.1 GAIN AGREEMENT ON COMMITMENTS

Stakeholders participate in business analysis activities that may require time and resource commitments. The business analyst and stakeholders identify and agree upon these commitments as early in the initiative as possible. The specific details of the commitments can be communicated formally or informally, as long as there is explicit understanding of the expectations and desired outcomes of the commitment.

There may be dialogue and negotiation regarding the terms and conditions of the commitments. Effective negotiation, communication, and conflict resolution skills are important to effective stakeholder management.

.2 MONITOR STAKEHOLDER ENGAGEMENT

Business analysts monitor the participation and performance of stakeholders to ensure that:

- the right subject matter experts (SMEs) and other stakeholders are participating effectively,
- stakeholder attitudes and interest are staying constant or improving,
- elicitation results are confirmed in a timely manner, and
- agreements and commitments are maintained.

Business analysts continually monitor for such risks as:

- stakeholders being diverted to other work,
- elicitation activities not providing the quality of business analysis information required, and
- delayed approvals.

.3 COLLABORATION

Stakeholders are more likely to support change if business analysts collaborate with them and encourage the free flow of information, ideas, and innovations. Genuine stakeholder engagement requires that all stakeholders involved feel that they are heard, their opinions matter, and their contributions are recognized. Collaboration involves regular, frequent, and bi-directional communication. Collaborative relationships help maintain the free flow of information when obstacles and setbacks occur, and promote a shared effort to resolve problems and achieve desired outcomes.

4.5.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Describes the nature and level of collaboration required from each stakeholder group to perform planned business analysis activities.
Business Objectives	Describe the desired direction needed to achieve the future state. They can be used to focus diverse stakeholders on a common vision of the desired business outcomes.
Future State Description	Defines the desired future state and the expected value it delivers which can be used to focus diverse stakeholders on the common goal.
Recommended Actions	Communicating what should be done to improve the value of a solution can help to galvanize support and focus stakeholders on a common goal.
Risk Analysis Results	Stakeholder-related risks will need to be addressed to ensure stakeholder collaboration activities are successful.

4.5.6 TECHNIQUES

Techniques	Usage
Collaborative Games	Used to stimulate teamwork and collaboration by temporarily immersing participants in a safe and fun situation in which they can share their knowledge and experience on a given topic, identify hidden assumptions, and explore that knowledge in ways that may not occur during the course of normal interactions.
Lessons Learned	Used to understand stakeholders' satisfaction or dissatisfaction, and offer them an opportunity to help improve the working relationships.
Risk Analysis and Management	Used to identify and manage risks as they relate to stakeholder involvement, participation, and engagement.
Stakeholder List, Map, or Personas	Used to determine who is available to participate in the business analysis work, show the informal relationships between stakeholders, and understand which stakeholders should be consulted about different kinds of business analysis information.

4.5.7 STAKEHOLDERS

Stakeholder	Contribution
All stakeholders	all types of stakeholders who might be involved in collaboration during change.

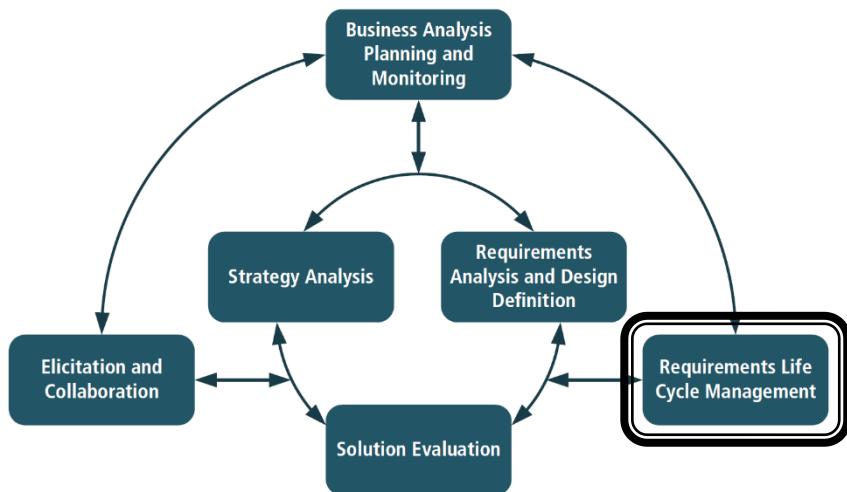
4.5.8 OUTPUTS

- **Stakeholder Engagement:** willingness from stakeholders to engage in business analysis activities and interact with the business analyst when necessary.

5. Requirements Life Cycle Management

The Requirements Life Cycle Management knowledge area describes the tasks that business analysts perform in order to manage and maintain requirements and design information from inception to retirement. These tasks describe establishing meaningful relationships between related requirements and designs, assessing changes to requirements and designs when changes are proposed, and analyzing and gaining consensus on changes.

The purpose of requirements life cycle management is to ensure that business, stakeholder, and solution requirements and designs are aligned to one another and that the solution implements them. It involves a level of control over requirements and over how requirements will be implemented in the actual solution to be constructed and delivered. It also helps to ensure that business analysis information is available for future use.



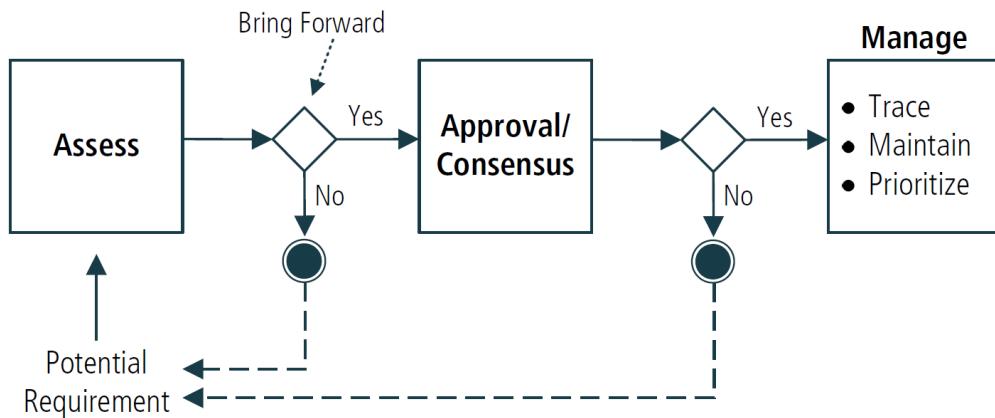
The requirements life cycle:

- begins with the representation of a business need as a requirement,
- continues through the development of a solution, and
- ends when a solution and the requirements that represent it are retired.

The management of requirements does not end once a solution is implemented. Throughout the life of a solution, requirements continue to provide value when they are managed appropriately.

Within the Requirements Life Cycle Management knowledge area, the concept of a life cycle is separate from a methodology or process used to govern business analysis work. Life cycle refers to the existence of various phases or states that requirements pass through as part of any change. Requirements may be in multiple states at one time.

Figure 5.1: Requirements Life Cycle Management



The Requirements Life Cycle Management knowledge area includes the following tasks:

TMPCA

1. **Trace Requirements**
2. **Maintain Requirements**
3. **Prioritize Requirements**
4. **Assess Requirements Changes**
5. **Approve Requirements**

The Core Concept Model in Requirements Life Cycle Management

The BUSINESS ANALYSIS CORE CONCEPT MODEL™ (BACCM™) describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Requirements Life Cycle Management.

Table 5.1: The Core Concept Model in Requirements Life Cycle Management

Core Concept	During Requirements Life Cycle Management, business analysts...
Change	Manage how proposed changes to requirements and designs are evaluated during an initiative.
Need	Trace, prioritize and maintain requirements to ensure that the need is met.
Solution	Trace requirements and designs to solution components to ensure that the solution satisfies the need.
Stakeholder	Work closely with key stakeholders to maintain understanding, agreement, and approval of requirements and designs.
Value	Maintain requirements for reuse to extend value beyond the current initiative.
Context	Analyze the context to support tracing and prioritization activities.

5.1 TRACE REQUIREMENTS

5.1.1 PURPOSE

The purpose of Trace Requirements is to ensure that requirements and designs at different levels are aligned to one another, and to manage the effects of change to one level on related requirements.

5.1.2 DESCRIPTION

Requirements traceability identifies and documents the lineage of each requirement, including its backward traceability, its forward traceability, and its relationship to other requirements.

Traceability is used to help ensure that the solution conforms to requirements and to assist in scope, change, risk, time, cost, and communication management. It is also used to detect missing functionality or to identify if there is implemented functionality that is not supported by any requirement.

Traceability enables:

- faster and simpler impact analysis,
- more reliable discovery of inconsistencies and gaps in requirements,
- deeper insights into the scope and complexity of a change, and
- reliable assessment of which requirements have been addressed and which have not.

It is often difficult to accurately represent needs and solutions without taking into account the relationships that exist between them. While traceability is valuable, the business analyst balances the number of relationship types with the benefit gained by representing them.

Traceability also supports both requirements allocation and release planning by providing a direct line of sight from requirement to expressed need.

The following images show examples of visual representations of traceability for a process and for software requirements.

Figure 5.1.1: Process Traceability

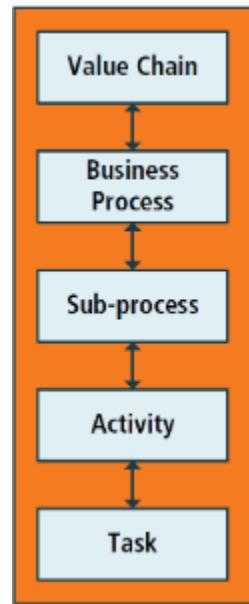
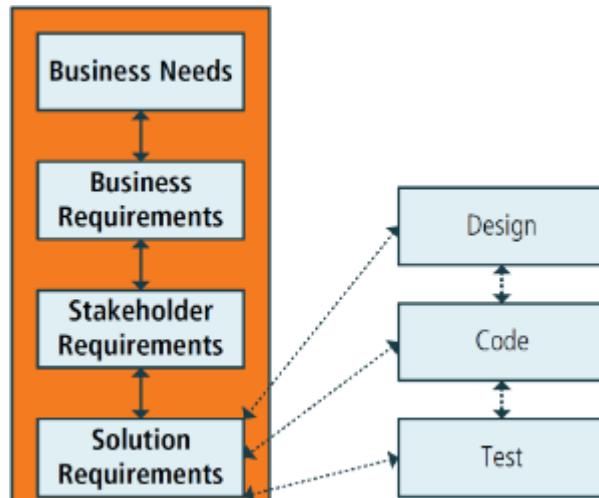


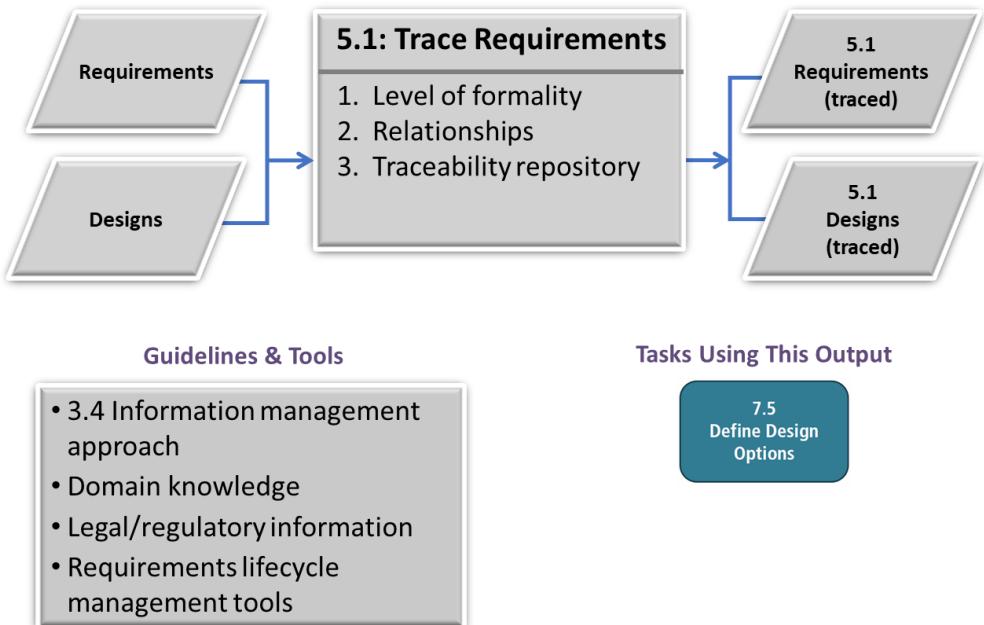
Figure 5.1.2: Software Requirements Traceability



5.1.3 INPUTS

- **Requirements:** may be traced to other requirements (including goals, objectives, business requirements, stakeholder requirements, solution requirements, and transition requirements), solution components, **visuals**, **business rules**, and other work products.
- **Designs:** may be traced to other requirements, solution components, and other work products.

Figure 5.1.3: Trace Requirements Input/Output Diagram



5.1.4 ELEMENTS

.1 LEVEL OF FORMALITY

When tracing requirements, business analysts consider the value that each link is supposed to deliver, as well as the nature and use of the specific relationships that are being created.

The effort to trace requirements grows significantly when the number of requirements or level of formality increases.

.2 RELATIONSHIPS

There are several types of relationships that the business analyst considers when defining the traceability approach:

- **Derive:** relationship between two requirements, used when a requirement is derived from another requirement. This type of relationship is appropriate to link the requirements on different levels of abstraction. For example, a solution requirement derived from a business or a stakeholder requirement.
- **Depends:** relationship between two requirements, used when a requirement depends on another requirement. Types of dependency relationships include:
- **Necessity:** when it only makes sense to implement a particular requirement if a related requirement is also implemented.
- **Effort:** when a requirement is easier to implement if a related requirement is also implemented.

- **Satisfy:** relationship between an implementation element and the requirements it is satisfying. For example, the relationship between a functional requirement and a solution component that is implementing it.
- **Validate:** relationship between a requirement and a test case or other element that can determine whether a solution fulfills the requirement.

.3 TRACEABILITY REPOSITORY

Requirements traceability is documented and maintained in accordance with the methods identified by the business analysis approach. Requirements management tools can provide significant benefits when there is a need to trace a large number of requirements that may be deemed unmanageable with manual approaches.

5.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Domain Knowledge	Knowledge of and expertise in the business domain needed to support traceability.
Information Management Approach	Provides decisions from planning activities concerning the traceability approach.
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed. These may need to be considered when defining traceability rules.
Requirements Management Tools/Repository	Used to store and manage business analysis information. The tool may be as simple as a text document or as complex as a dedicated requirements management tool.

5.1.6 TECHNIQUES

Techniques	Usage
Business Rules Analysis	Used to trace business rules to requirements that they support, or rules that support requirements.
Functional Decomposition	Used to break down solution scope into smaller components for allocation, as well as to trace high-level concepts to low-level concepts.
Process Modelling	Used to visually show the future state process, as well as tracing requirements to the future state process.
Scope Modelling	Used to visually depict scope, as well as trace requirements to the area of scope the requirement supports.

5.1.7 STAKEHOLDERS

Stakeholder	Contribution
Customers	Are affected by how and when requirements are implemented, and may have to be consulted about, or agree to, the traceability relationships.
Domain Subject Matter	May have recommendations regarding the set of requirements

Expert	to be linked to a solution component or to a release.
End User	May require specific dependency relationships that allow certain requirements to be implemented at the same time or in a specific sequence.
Implementation Subject Matter Expert	Traceability ensures that the solution being developed meets the business need and brings awareness of dependencies between solution components during implementation.
Operational Support	Traceability documentation provides another reference source for help desk support.
Project Manager	Traceability supports project change and scope management.
Sponsor	Is required to approve the various relationships.
Suppliers	Are affected by how and when requirements are implemented.
Tester	Needs to understand how and where requirements are implemented when creating test plans and test cases, and may trace test cases to requirements.

5.1.8 OUTPUTS

- **Requirements (traced):** have clearly defined relationships to other requirements, solution components, or releases, phases, or iterations, within a solution scope, such that coverage and the effects of change are clearly identifiable.
- **Designs (traced):** clearly defined relationships to other requirements, solution components, or releases, phases, or iterations, within a solution scope, such that coverage and the effects of change are clearly identifiable.

5.2 MAINTAIN REQUIREMENTS

5.2.1 PURPOSE

The purpose of Maintain Requirements is to retain requirement accuracy and consistency throughout and beyond the change during the entire requirements life cycle, and to support reuse of requirements in other solutions.

5.2.2 DESCRIPTION

A requirement that represents an ongoing need must be maintained to ensure that it remains valid over time.

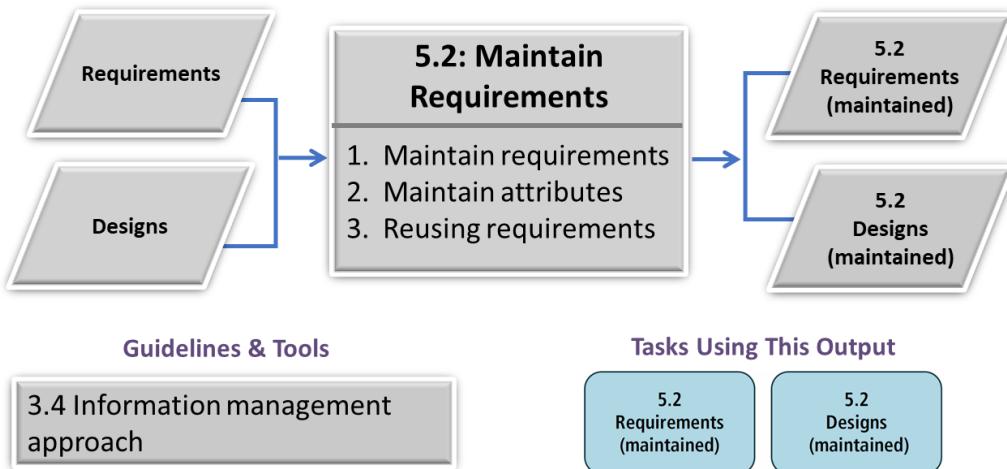
In order to maximize the benefits of maintaining and reusing requirements, the requirements should be:

- consistently represented,
- reviewed and approved for maintenance using a standardized process that defines proper access rights and ensures quality, and
- easily accessible and understandable.

5.2.3 INPUTS

- **Requirements:** include goals, objectives, business requirements, stakeholder requirements, solution requirements, and transition requirements. These should be maintained throughout their life cycle.
- **Designs:** can be maintained throughout their life cycle, as needed.

Figure 5.2.1: Maintain Requirements Input/Output Diagram



5.2.4 ELEMENTS

.1 MAINTAIN REQUIREMENTS

Requirements are maintained so that they remain correct and current after an approved change. Business analysts are responsible for conducting maintenance to ensure this level of accuracy is retained. For requirements to be properly maintained they must be clearly named and defined, and easily available to stakeholders.

Business analysts also maintain the relationships among requirements, sets of requirements, and associated business analysis information to ensure the context and original intent of the requirement is preserved. Repositories with accepted taxonomies assist in establishing and maintaining links between maintained requirements, and facilitate requirements and designs traceability.

.2 MAINTAIN ATTRIBUTES

While eliciting requirements, business analysts elicit requirement attributes. Information such as the requirement's source, priority, and complexity aid in managing each requirement throughout the life cycle. Some attributes change as the business analyst uncovers more information and conducts further analysis. An attribute may change even though the requirement does not.

.3 REUSING REQUIREMENTS

There are situations in which requirements can be reused.

Requirements that are candidates for long-term use by the organization are identified, clearly named, defined, and stored in a manner that makes them easily retrievable by other stakeholders. Depending on the level of abstraction and intended need being addressed, requirements can be reused:

- within the current initiative,
- within similar initiatives,
- within similar departments, and
- throughout the entire organization.

Requirements at high levels of abstraction may be written with limited reference to specific solutions. Requirements that are represented in a general manner, without direct ties to a particular tool or organizational structure, tend to be more reusable. These requirements are also less subject to revision during a change. As requirements are expressed in more detail, they become more tightly associated with a specific solution or solution option. Specific references to applications or departments limit the reuse of requirements and designs across an organization.

Requirements that are intended for reuse reflect the current state of the organization. Stakeholders validate the proposed requirements for reuse before they can be accepted into a change.

5.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Information Management Approach	Indicates how requirements will be managed for reuse.

5.2.6 TECHNIQUES

Techniques	Usage
Business Rules Analysis	Used to identify business rules that may be similar across the enterprise in order to facilitate reuse.
Data Flow Diagrams	Used to identify information flow that may be similar across the enterprise in order to facilitate reuse.
Data Modelling	Used to identify data structure that may be similar across the enterprise in order to facilitate reuse.
Document Analysis	Used to analyze existing documentation about an enterprise that can serve as the basis for maintaining and reusing requirements.
Functional Decomposition	Used to identify requirements associated with the components and available for reuse.
Process Modelling	Used to identify requirements associated with the processes that may be available for reuse.
Use Cases and Scenarios	Used to identify a solution component that may be utilized by more than one solution.
User Stories	Used to identify requirements associated with the story that may be available for reuse.

5.2.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	References maintained requirements on a regular basis to ensure they are accurately reflecting stated needs.

Implementation Subject Matter Expert	Utilizes maintained requirements when developing regression tests and conducting impact analysis for an enhancement.
Operational Support	Maintained requirements are likely to be referenced to confirm the current state.
Regulator	Maintained requirements are likely to be referenced to confirm compliance to standards.
Tester	Maintained requirements are used by testers to aid in test plan and test case creation

5.2.8 OUTPUTS

- **Requirements (maintained)**: defined once and available for long-term usage by the organization. They may become organizational process assets or be used in future initiatives. In some cases, a requirement that was not approved or implemented may be maintained for a possible future initiative.
- **Designs (maintained)**: may be reusable once defined. For example, as a self-contained component that can be made available for possible future use.

5.3 PRIORITY REQUIREMENTS

5.3.1 PURPOSE

The purpose of Prioritize Requirements is to rank requirements in the order of relative importance.

5.3.2 DESCRIPTION

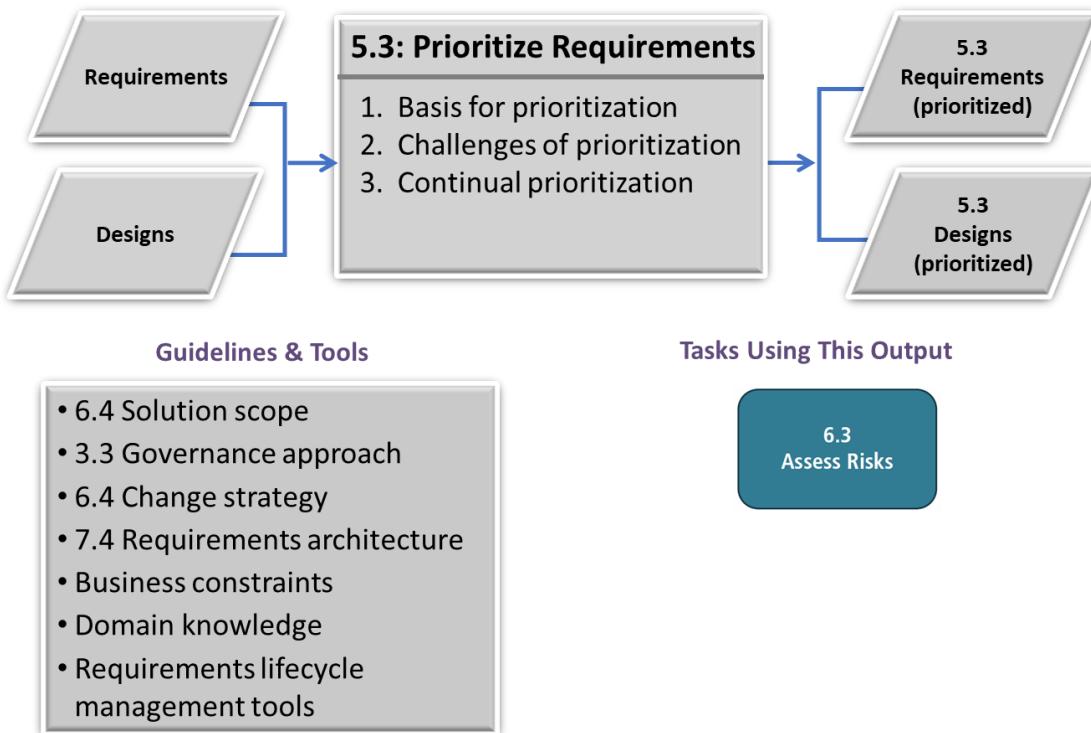
Prioritization is the act of ranking requirements to determine their relative importance to stakeholders. When a requirement is prioritized, it is given greater or lesser priority. Priority can refer to the relative value of a requirement, or to the sequence in which it will be implemented. Prioritization is an ongoing process, with priorities changing as the context changes.

Inter-dependencies between requirements are identified and may be used as the basis for prioritization. Prioritization is a critical exercise that seeks to ensure the maximum value is achieved.

5.3.3 INPUTS

- **Requirements**: any requirements in the form of text, matrices, or diagrams that are ready to prioritize.
- **Designs**: any designs in the form of text, prototypes, or diagrams that are ready to prioritize.

Figure 5.3.1: Prioritize Requirements Input/Output Diagram



5.3.4 ELEMENTS

.1 BASIS FOR PRIORITIZATION

The basis on which requirements are prioritized is agreed upon by relevant stakeholders as defined in the Business Analysis Planning and Monitoring knowledge area.

Typical factors that influence prioritization include:

- **Benefit:** the advantage that accrues to stakeholders as a result of requirement implementation, as measured against the goals and objectives for the change. The benefit provided can refer to a specific functionality, desired quality, or strategic goal or business objective. If there are multiple stakeholders, each group may perceive benefits differently. Conflict resolution and negotiation may be employed to come to consensus on overall benefit.
- **Penalty:** the consequences that result from not implementing a given requirement. This includes prioritizing requirements in order to meet regulatory or policy demands imposed on the organization, which may take precedence over other stakeholder interests. Penalty may also refer to the negative consequence of not implementing a requirement that improves the experience of a customer.
- **Cost:** the effort and resources needed to implement the requirement. Information about cost typically comes from the implementation team or the vendor. Customers may change the priority of a requirement after learning the cost. Cost is often used in conjunction with other criteria, such as cost-benefit analysis.
- **Risk:** the chance that the requirement cannot deliver the potential value, or cannot be met at all. This may include many factors such as the difficulty of implementing a requirement, or

the chance that stakeholders will not accept a solution component. If there is a risk that the solution is not technically feasible, the requirement that is most difficult to implement may be prioritized to the top of the list in order to minimize the resources that are spent before learning that a proposed solution cannot be delivered. A proof of concept may be developed to establish that high risk options are possible.

- **Dependencies:** relationships between requirements where one requirement cannot be fulfilled unless the other requirement is fulfilled. In some situations, it may be possible to achieve efficiencies by implementing related requirements at the same time. Dependencies may also be external to the initiative, including but not limited to other teams' decisions, funding commitments, and resource availability. Dependencies are identified as part of the task Trace Requirements.
- **Time Sensitivity:** the 'best before' date of the requirement, after which the implementation of the requirement loses significant value. This includes time-to-market scenarios, in which the benefit derived will be exponentially greater if the functionality is delivered ahead of the competition. It can also refer to seasonal functionality that only has value at a specific time of year.
- **Stability:** the likelihood that the requirement will change, either because it requires further analysis or because stakeholders have not reached a consensus about it. If a requirement is not stable, it may have a lower priority in order to minimize unanticipated rework and wasted effort.
- **Regulatory or Policy Compliance:** requirements that must be implemented in order to meet regulatory or policy demands imposed on the organization, which may take precedence over other stakeholder interests.

.2 CHALLENGES OF PRIORITIZATION

Prioritization is an assessment of relative value. Each stakeholder may value something different. When this occurs, there may be conflict amongst stakeholders. Stakeholders may also have difficulty characterizing any requirement as a lower priority, and this may impact the ability to make necessary trade-offs. In addition, stakeholders may (intentionally or unintentionally) indicate priority to influence the result to their desired outcome.

Different types of requirements may not all respond to the criteria in the same way and may appear to conflict. There may be a need for stakeholders to make trade-offs in prioritization.

.3 CONTINUAL PRIORITIZATION

Priorities may shift as the context evolves and as more information becomes available. Initially, prioritization is done at a higher level of abstraction. As the requirements are further refined, prioritization is done at a more granular level and will incorporate additional bases for prioritization as they become appropriate. The basis for prioritization may be different at various stages of the change. For example, stakeholders may initially prioritize based on benefits. The implementation team may then re-prioritize the requirements based on the sequence in which they must be implemented due to technical constraints. Once the implementation team has provided the cost of each requirement, the stakeholders may re-prioritize yet again.

5.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Constraints	Regulatory statutes, contractual obligations and business policies that may define priorities.

Change Strategy	Provides information on costs, timelines, and value realization which are used to determine priority of requirements.
Domain Knowledge	Knowledge and expertise of the business domain needed to support prioritization.
Governance Approach	Outlines the approach for prioritizing requirements.
Requirements Architecture	Utilized to understand the relationship with other requirements and work products.
Requirements Management Tools/Repository	Including a requirements attribute for prioritization can help the business analyst to sort and access requirements by priority.
Solution Scope	Considered when prioritizing requirements to ensure scope is managed.

5.3.6 TECHNIQUES

Techniques	Usage
Backlog Management	Used to compare requirements to be prioritized. The backlog can be the location where the prioritization is maintained.
Business Cases	Used to assess requirements against identified business goals and objectives to determine importance.
Decision Analysis	Used to identify high-value requirements
Estimation	Used to produce estimates for the basis of prioritization.
Financial Analysis	Used to assess the financial value of a set of requirements and how the timing of delivery will affect that value.
Interviews	Used to gain an understanding of a single or small group of stakeholders' basis of prioritization or priorities.
Item Tracking	Used to track issues raised by stakeholders during prioritization.
Prioritization	Used to facilitate the process of prioritization.
Risk Analysis and Management	Used to understand the risks for the basis of prioritization.
Workshops	Used to gain an understanding of stakeholders' basis of prioritization or priorities in a facilitated group setting.

5.3.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Verifies that the prioritized requirements will deliver value from a customer or end-user perspective. The customer can also negotiate to have the prioritization changed based on relative value.
End User	Verifies that the prioritized requirements will deliver value from a customer or end-user perspective.
Implementation Subject Matter Expert	Provides input relating to technical dependencies and can negotiate to have the prioritization changed based on technical constraints.

Project Manager	Uses the prioritization as input into the project plan and into the allocation of requirements to releases.
Regulator	Can verify that the prioritization is consistent with legal and regulatory constraints.
Sponsor	Verifies that the prioritized requirements will deliver value from an organizational perspective.

5.3.8 OUTPUTS

- **Requirements (prioritized):** prioritized or ranked requirements are available for additional work, ensuring that the highest valued requirements are addressed first.
- **Designs (prioritized):** prioritized or ranked designs are available for additional work, ensuring that the highest valued designs are addressed first.

5.4 ASSESS REQUIREMENTS CHANGES

5.4.1 PURPOSE

The purpose of Assess Requirements Changes is to evaluate the implications of proposed changes to requirements and designs.

5.4.2 DESCRIPTION

The Assess Requirements Changes task is performed as new needs or possible solutions are identified. These may or may not align to the change strategy and/or solution scope. Assessment must be performed to determine whether a proposed change will increase the value of the solution, and if so, what action should be taken.

Business analysts assess the potential effect of the change to solution value, and whether proposed changes introduce conflicts with other requirements or increase the level of risk. Business analysts also ensure each proposed change can be traced back to a need.

When assessing changes, business analysts consider if each proposed change:

- aligns with the overall strategy,
- affects value delivered to the business or stakeholder groups,
- impacts the time to deliver or the resources required to deliver the value, and
- alters any risks, opportunities, or constraints associated with the overall initiative.

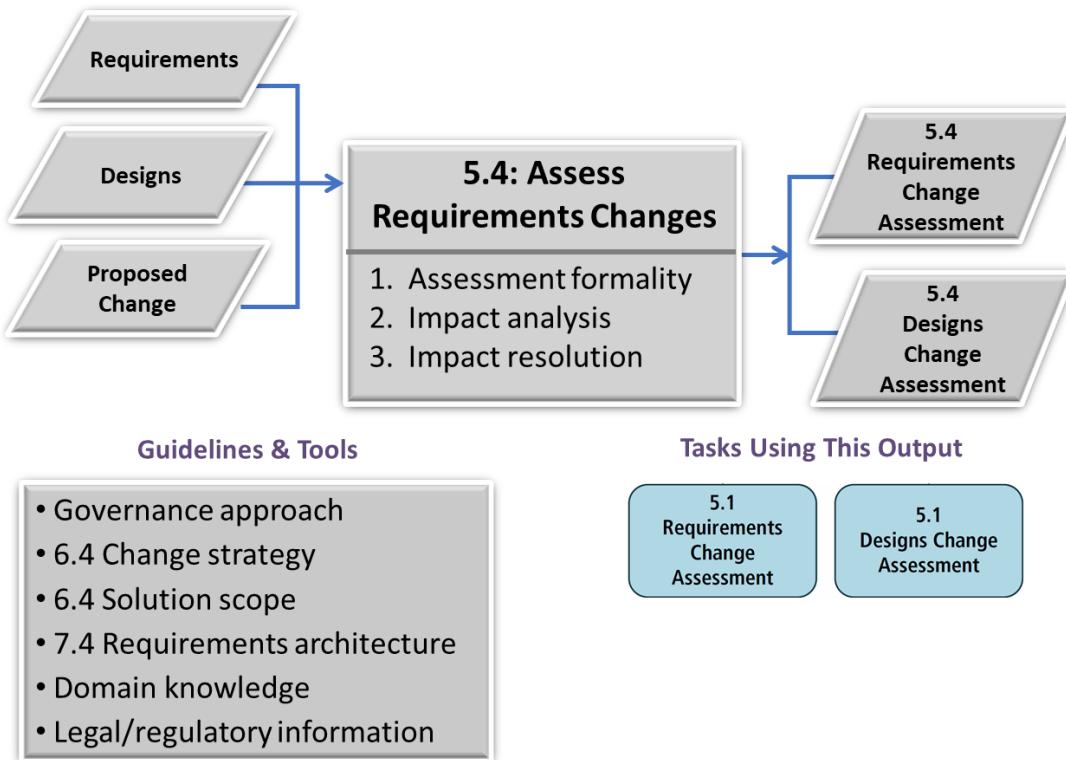
The results of the assessment must support the decision making and change control approaches defined by the task Plan Business Analysis Governance.

5.4.3 INPUTS

- **Proposed Change:** can be identified at any time and impact any aspect of business analysis work or deliverables completed to date. There are many triggers for a proposed change including business strategy changes, stakeholders, legal requirements, or regulatory changes.
- **Requirements:** may need to be assessed to identify the impact of a proposed modification.

- **Designs:** may need to be assessed to identify the impact of a proposed modification.

Figure 5.4.1: Assess Requirements Changes Input/Output Diagram



5.4.4 ELEMENTS

.1 ASSESSMENT FORMALITY

Business analysts will determine the formality of the assessment process based on the information available, the apparent importance of the change, and the governance process. Many proposed changes may be withdrawn from consideration or declined before any formal approval is required. A predictive approach may indicate a more formal assessment of proposed changes. In predictive approaches, the impact of each change can be disruptive; the change can potentially generate a substantial reworking of tasks and activities completed in previous activities. An adaptive approach may require less formality in the assessment of proposed changes. While there may be reworking needed as a result of each change, adaptive approaches try to minimize the impact of changes by utilizing iterative and incremental implementation techniques. This idea of continuous evolution may reduce the need for formal impact assessment.

.2 IMPACT ANALYSIS

Impact analysis is performed to assess or evaluate the effect of a change. Traceability is a useful tool for performing impact analysis. When a requirement changes, its relationships to other requirements or solution components can be reviewed. Each related requirement or component may also require a change to support the new requirement.

When considering changes or additions to existing requirements, business analysts assess the impact of the proposed change by considering:

- **Benefit:** the benefit that will be gained by accepting the change.
- **Cost:** the total cost to implement the change including the cost to make the change, the cost of associated rework, and the opportunity costs such as the number of other features that may need to be sacrificed or deferred if the change is approved.
- **Impact:** the number of customers or business processes affected if the change is accepted.
- **Schedule:** the impact to the existing delivery commitments if the change is approved.
- **Urgency:** the level of importance including the factors which drive necessity such as regulator or safety issues.

.3 IMPACT RESOLUTION

Depending on the planned approach, various stakeholders (including the business analyst) may be authorized to approve, deny, or defer the proposed change. All impacts and resolutions resulting from the change analysis are to be documented and communicated to all stakeholders. How decisions and changes will be made and communicated across an initiative is determined by the task Plan Business Analysis Governance.

5.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Change Strategy	Describes the purpose and direction for changes, establishes the context for the change, and identifies the critical components for change.
Domain Knowledge	Knowledge of and expertise in the business domain is needed to assess proposed requirements changes.
Governance Approach	Provides guidance regarding the change control and decision-making processes, as well as the roles of stakeholders within this process.
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed. These may impact requirements and must be considered when making changes.
Requirements Architecture	Requirements may be related to each other, therefore the business analyst examines and analyzes the requirement relationships to determine which requirements will be impacted by a requested requirements change.
Solution Scope	Must be considered when assessing changes to fully understand the impact of a proposed change.

5.4.6 TECHNIQUES

Techniques	Usage
Business Cases	Used to justify a proposed change.
Business Rules Analysis	Used to assess changes to business policies and business rules, and develop revised guidance.
Decision Analysis	Used to facilitate the change assessment process.

Document Analysis	Used to analyze any existing documents that facilitate an understanding of the impact of the change.
Estimation	Used to determine the size of the change.
Financial Analysis	Used to estimate the financial consequences of a proposed change.
Interface Analysis	Used to help business analysts identify interfaces that can be affected by the change.
Interviews	Used to gain an understanding of the impact on the organization or its assets from a single or small group of stakeholders.
Item Tracking	Used to track any issues or conflicts discovered during impact analysis.
Risk Analysis and Management	Used to determine the level of risk associated with the change.
Workshops	Used to gain an understanding of the impact or to resolve changes in a group setting.

5.4.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Provides feedback concerning the impact the change will have on value.
Domain Subject Matter Expert	Has expertise in some aspect of the situation and can provide insight into how the change will impact the organization and value.
End User	Uses the solution or is a component of the solution, and can offer information about the impact of the change on their activities.
Operational Support	Provides information on both their ability to support the operation of the solution and their need to understand the nature of the change in the solution in order to be able to support it.
Project Manager	Reviews the requirements change assessment to determine if additional project work is required for a successful implementation of the solution.
Regulator	Changes are likely to be referenced by auditors to confirm compliance to standards.
Sponsor	Accountable for the solution scope and can provide insight to be utilized when assessing change.
Tester	Consulted for establishing impact of the proposed changes.

5.4.8 OUTPUTS

- **Requirements Change Assessment:** the recommendation to approve, modify, or deny a proposed change to requirements.
- **Designs Change Assessment:** the recommendation to approve, modify, or deny a proposed change to one or more design components.

5.5 APPROVE REQUIREMENTS

5.5.1 PURPOSE

The purpose of Approve Requirements is to obtain agreement on and approval of requirements and designs for business analysis work to continue and/or solution construction to proceed.

5.5.2 DESCRIPTION

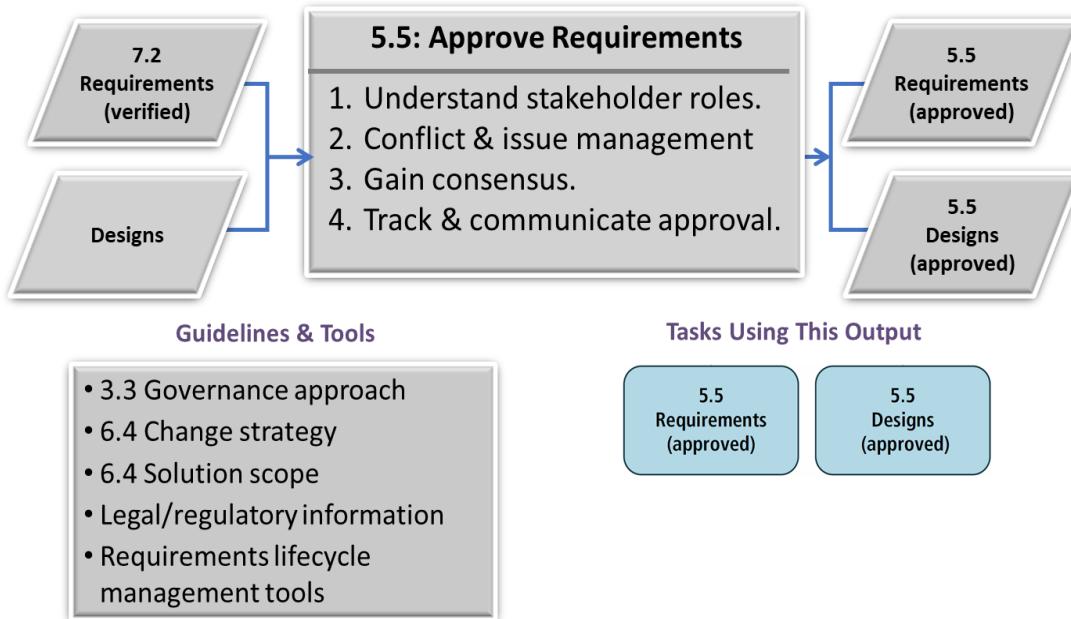
Business analysts are responsible for ensuring clear communication of requirements, designs, and other business analysis information to the key stakeholders responsible for approving that information.

Approval of requirements and designs may be formal or informal. Predictive approaches typically perform approvals at the end of the phase or during planned change control meetings. Adaptive approaches typically approve requirements only when construction and implementation of a solution meeting the requirement can begin. Business analysts work with key stakeholders to gain consensus on new and changed requirements, communicate the outcome of discussions, and track and manage the approval.

5.5.3 INPUTS

- **Requirements (verified):** a set of requirements that have been verified to be of sufficient quality to be used as a reliable body of work for further specification and development.
- **Designs:** a set of designs that have been determined as ready to be used for further specification and development.

Figure 5.5.1: Approve Requirements Input/Output Diagram



5.5.4 ELEMENTS

.1 UNDERSTAND STAKEHOLDER ROLES

The approval process is defined by the task Plan Business Analysis Governance. Part of defining the approval process is understanding stakeholder roles and authority levels. Business analysts are responsible for obtaining stakeholder approvals and are required to understand who holds decision-making responsibility and who possesses authority for sign-off across the initiative. Business analysts also consider any influential stakeholders who should be consulted or informed about the requirements. Few stakeholders may have the authority to approve or deny changes, but many stakeholders may be able to influence these decisions.

.2 CONFLICT AND ISSUE MANAGEMENT

To maintain stakeholder support for the solution, consensus among stakeholders is usually sought prior to requesting approval of requirements. The approach for determining how to secure decisions and resolve conflicts across an initiative is planned for in the task Plan Business Analysis Governance.

Stakeholder groups frequently have varying points of view and conflicting priorities. A conflict may arise among stakeholders as a result of different interpretations of requirements or designs and conflicting values placed on them. The business analyst facilitates communication between stakeholders in areas of conflict so that each group has an improved appreciation for the needs of the others. Conflict resolution and issue management may occur quite often, as the business analyst is reviewing requirements and designs, and aiming to secure sign-off.

.3 GAIN CONSENSUS

Business analysts are responsible for ensuring that the stakeholders with approval authority understand and accept the requirements. Approval may confirm that stakeholders believe that sufficient value will be created for the organization to justify investment in a solution. Business analysts obtain approval by reviewing the requirements or changes to requirements with the accountable individuals or groups and requesting that they approve, indicating their agreement with the solution or designs described.

Using the methods and means established in the tasks Plan Business Analysis Governance and Communicate Business Analysis Information, business analysts present the requirements to stakeholders for approval. Business analysts facilitate this approval process by addressing any questions or providing additional information when requested.

Complete agreement may not be necessary for a successful change, but if there is a lack of agreement, the associated risks are to be identified and managed accordingly.

.4 TRACK AND COMMUNICATE APPROVAL

The business analyst records approval decisions, possibly in requirements maintenance and tracking tools. In order to communicate the status of requirements, it is necessary to keep accurate records of current approval status. Stakeholders must be able to determine what requirements and designs are currently approved and in line for implementation. There may be value in maintaining an audit history of changes to requirements: what was changed, who made the change, the reason for the change, and when it was made.

5.5.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Change Strategy	Provides information which assists in managing stakeholder consensus regarding the needs of all stakeholders.
Governance Approach	Identifies the stakeholders who have the authority and responsibility to approve business analysis information, and explains when such approvals will take place and how they will align to organizational policies
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed. They may impact the requirements and designs approval process.
Requirement Management Tools/Repository	Tool to record requirements approvals
Solution Scope	Must be considered when approving requirements to accurately assess alignment and completeness.

5.5.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to define approval criteria.
Decision Analysis	Used to resolve issues and gain agreement.
Item Tracking	Used to track issues identified during the agreement process.
Reviews	Used to evaluate requirements.
Workshop	Used to facilitate obtaining approval.

5.5.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	May play an active role in reviewing and approving requirements and designs to ensure needs are met.
Domain Subject Matter Expert	May be involved in the review and approval of requirements and designs as defined by stakeholder roles and responsibilities designation.
End User	People who use the solution, or who are a solution component, and may be involved in the review, validation, and prioritization of requirements and designs as defined by the stakeholder roles and responsibilities designation.
Operational Support	Responsible for ensuring that requirements and designs are supportable within the constraints imposed by technology standards and organizational capability plans. Operational support personnel may have a role in reviewing and approving requirements.
Project Manager	Responsible for identifying and managing risks associated with solution design, development, delivery, implementation,

	operation and sustainment. The project manager may manage the project plan activities pertaining to review and/or approval.
Regulator	External or internal party who is responsible for providing opinions on the relationship between stated requirements and specific regulations, either formally in an audit, or informally as inputs to requirements life cycle management tasks.
Sponsor	Responsible to review and approve the business case, solution or product scope, and all requirements and designs.
Tester	Responsible for ensuring quality assurance standards are feasible within the business analysis information. For example, requirements have the testable characteristic.

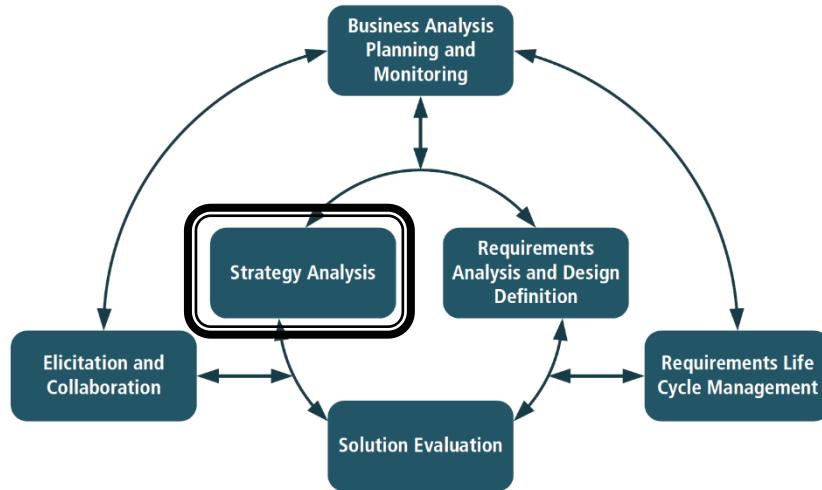
5.5.8 OUTPUTS

- **Requirements (approved):** requirements which are agreed to by stakeholders and are ready for use in subsequent business analysis efforts.
- **Designs (approved):** designs which are agreed to by stakeholders and are ready for use in subsequent business analysis or solution development efforts.

6. Strategy Analysis

Strategy defines the most effective way to apply the capabilities of an enterprise in order to reach a desired set of goals and objectives. Strategies may exist for the entire enterprise, for a division, department or region, and for a product, project, or iteration.

The Strategy Analysis knowledge area describes the business analysis work that must be performed to collaborate with stakeholders in order to identify a need of strategic or tactical importance (the Business Need), enable the enterprise to address that need, and align the resulting strategy for the change with higher- and lower-level strategies.

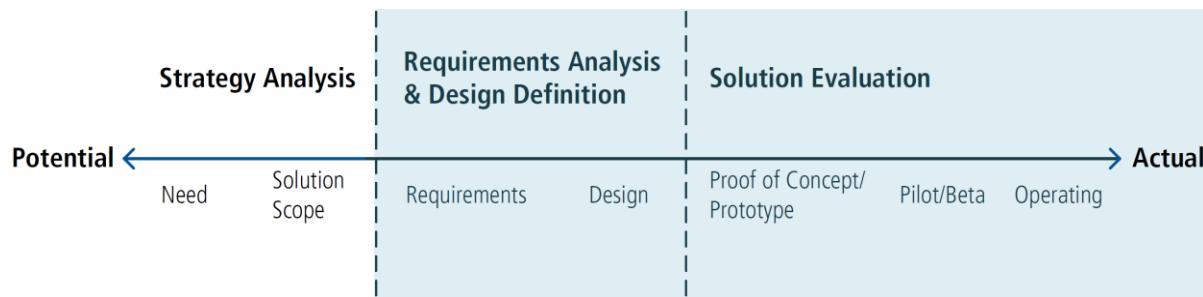


Strategy analysis focuses on defining the future and transition states needed to address the business need, and the work required is defined both by that need and the scope of the solution space. It covers strategic thinking in business analysis, as well as the discovery or imagining of possible solutions that will enable the enterprise to create greater value for stakeholders, and/or capture more value for itself.

Strategy analysis provides context to requirements analysis and design definition for a given change. Strategy analysis should be performed as a business need is identified. This allows stakeholders to make the determination of whether to address that need or not. Strategy analysis is an ongoing activity that assesses any changes in that need, in its context, or any new information that may indicate that an adjustment to the change strategy may be required.

The following image illustrates the spectrum of value as business analysis activities progress from delivering potential value to actual value.

Figure 6.1: Business Analysis Value Spectrum



When performing strategy analysis, business analysts must consider the context in which they are working, and how predictable the range of possible outcomes is. When a change will have a predictable outcome, the future state and possible transition states can typically be clearly defined, and a clear strategy can be planned out. If the outcome of a change is difficult to predict, the strategy may need to focus more on mitigating risk, testing assumptions, and changing course until a strategy that will succeed in reaching the business goals can be identified or until the initiative has ended. These tasks may be performed in any order, though they are often performed concurrently, as strategy must be shaped by what is actually achievable.

A strategy may be captured in a strategic plan, product vision, business case, product roadmap, or other artifacts.

The Strategy Analysis knowledge area includes the following tasks:

CFRS

1. Analyze Current State
2. Define Future State
3. Assess Risks
4. Define Change Strategy

The Core Concept Model in Strategy Analysis

The BUSINESS ANALYSIS CORE CONCEPT MODEL™ (BACCM™) describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Strategy Analysis.

Table 6.1: The Core Concept Model in Strategy Analysis

Core Concept	During Strategy Analysis, business analysts...
Change	Define the future state and develop a change strategy to achieve the future state.
Need	Identify needs within the current state and prioritize needs to determine the desired future state.
Solution	Define the scope of a solution as part of developing a change strategy.
Stakeholder	Collaborate with stakeholders to understand the business need and to develop a change strategy and future state that will meet those needs.
Value	Examine the potential value of the solution to determine if a change is justified.
Context	Consider the context of the enterprise in developing a change strategy.

6.1 ANALYZE CURRENT STATE

6.1.1 PURPOSE

The purpose of Analyze Current State is to understand the reasons why an enterprise needs to change some aspect of how it operates and what would be directly or indirectly affected by the change.

6.1.2 DESCRIPTION

The starting point for any change is an understanding of WHY the change is needed. Potential change is triggered by problems or opportunities that cannot be addressed without altering the current state. Business analysts work to help stakeholders enable change by exploring and articulating the business needs that drive the desire to change. Without clearly understood business needs, it is impossible to develop a coherent strategy, and the resulting change initiative is almost certain to be driven by a mix of conflicting stakeholder demands.

Change always occurs in a context of existing stakeholders, processes, technology, and policies which constitute the current state of the enterprise. Business analysts examine the current state in the context of the business need to understand what may influence proposed changes, and what will be affected by them. The current state is explored in just enough detail to validate the need for a change and/or the change strategy. Understanding the current state of the enterprise prior to the change is necessary to identify what will need to change to achieve a desired future state and how the effect of the change will be assessed.

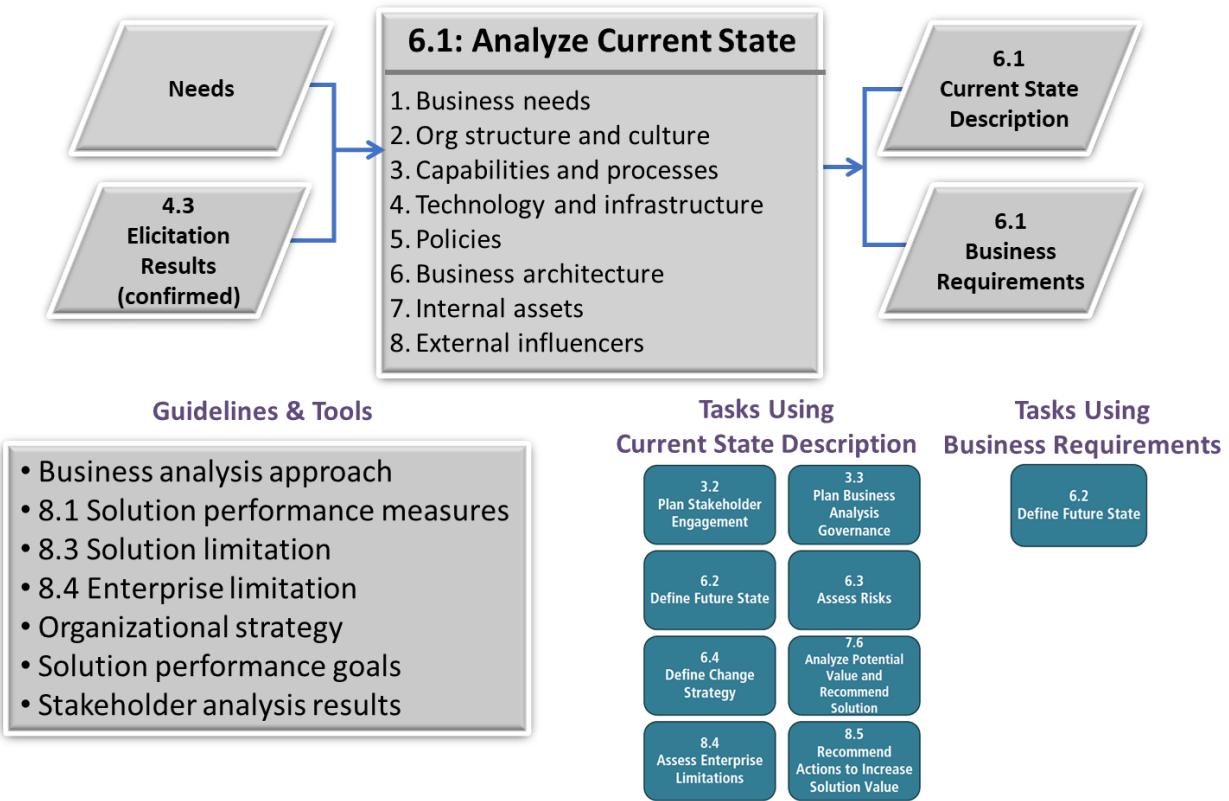
The scope of the current state describes the important existing characteristics of the environment. The boundaries of the current state scope are determined by the components of the enterprise and its environment as they relate to the needs. The current state can be described on different levels, ranging from the entire enterprise to small components of a solution. Creating a model of the current state might require collaboration throughout or outside the enterprise. For small efforts, the scope might be only a small component of an enterprise.

Note: The current state of an enterprise is rarely static while a change is being developed and implemented. Internal and external influencers, as well as other organizational changes, can affect the current state in ways that force alterations in the desired future state, change strategy, or requirements and designs.

6.1.3 INPUTS

- **Elicitation Results:** used to define and understand the current state.
- **Needs:** the problem or opportunity faced by an enterprise or organization often launches business analysis work to better understand these needs.

Figure 6.1.1: Analyze Current State Input/Output Diagram



6.1.4 ELEMENTS

.1 BUSINESS NEEDS

Business needs are the problems and opportunities of strategic importance faced by the enterprise. An issue encountered in the organization, such as a customer complaint, a loss of revenue, or a new market opportunity, usually triggers the evaluation of a business need.

A business need may be identified at many different levels of the enterprise:

- **From the top-down:** a strategic goal that needs to be achieved.
- **From the bottom-up:** a problem with the current state of a process, function or system.
- **From middle management:** a manager needs additional information to make sound decisions or must perform additional functions to meet business objectives.
- **From external drivers:** customer demand or business competition in the marketplace.

The definition of business needs is frequently the most critical step in any business analysis effort. A solution must satisfy the business needs to be considered successful. The way the need is defined determines which alternative solutions will be considered, which stakeholders will be consulted, and

which solution approaches will be evaluated. Business needs are always expressed from the perspective of the enterprise, and not that of any particular stakeholder.

Business needs are often identified or expressed along with a presumed solution. The business analyst should question the assumptions and constraints that are generally buried in the statement of the issue to ensure that the correct problem is being solved and the widest possible range of alternative solutions are considered.

A solution to a set of business needs must have the potential to generate benefits for the enterprise or its stakeholders, or avoid losses that would otherwise occur. Factors the business analyst may consider include:

- adverse impacts the problem is causing within the organization and quantify those impacts (for example, potential lost revenue, inefficiencies, dissatisfied customers, low employee morale),
- expected benefits from any potential solution (for example, increased revenue, reduced costs, increased market share),
- how quickly the problem could potentially be resolved or the opportunity could be taken, and the cost of doing nothing, and
- the underlying source of the problem.

Business needs will drive the overall analysis of the current state. Although it isn't necessary to fully detail all aspects of the current state before further developing the change strategy, this exploration will often uncover deeper underlying causes of the problem or the opportunity that triggered the investigation (which then become additional business needs).

.2 ORGANIZATIONAL STRUCTURE AND CULTURE

Organizational structure defines the formal relationships between people working in the enterprise. While communication channels and relationships are not limited to that structure, they are heavily influenced by it, and the reporting structure may aid or limit a potential change.

Organizational culture is the beliefs, values, and norms shared by the members of an organization. These beliefs drive the actions taken by an organization. Business analysts perform a cultural assessment to:

- identify if cultural changes are required to better achieve the goals,
- identify whether stakeholders understand the rationale for the current state of the enterprise and the value delivered by it, and
- ascertain whether the stakeholders view the current state as satisfactory or if change is needed.

.3 CAPABILITIES AND PROCESSES

Capabilities and processes describe the activities an enterprise performs. They also include the knowledge the enterprise has, the products and services it provides, the functions it supports, and the methods it uses to make decisions. Core capabilities or processes describe the essential functions of the enterprise that differentiate it from others. They are measured by performance indicators that can be used to assess the benefits of a change.

Business analysts may use:

- A capability-centric view of the enterprise when looking for innovative solutions that combine existing capabilities to produce a new outcome. A capability-based view is useful in this situation because capabilities are generally organized in a functional hierarchy with relationships to other capabilities, making it easier to identify any gaps.
 - A process-centric view of the enterprise when looking for ways to improve the performance of current activities. A process-based view is useful in this situation because processes are organized in an end-to-end fashion across the enterprise to deliver value to its customers, making it easier to ensure that a change does in fact increase performance.
-

.4 TECHNOLOGY AND INFRASTRUCTURE

Information systems used by the enterprise support people in executing processes, making decisions, and in interactions with suppliers and customers. The infrastructure describes the enterprise's environment with respect to physical components and capabilities. The infrastructure can include components such as computer hardware, physical plants, and logistics, as well as their operation and upkeep.

.5 POLICIES

Policies define the scope of decision making at different levels of an enterprise. They generally address routine operations rather than strategic change. They ensure that decisions are made correctly, provide guidance to staff on permitted and appropriate behaviour and actions, support governance, and determine when and how new resources can be acquired. Identification of relevant policies may shape the scope of the solution space and may be a constraint on the types of action that can be pursued.

.6 BUSINESS ARCHITECTURE

No part of the current state should be assessed in complete isolation from the rest. Business analysts must understand how all of these elements of the current state fit together and support one another in order to recommend changes that will be effective. The existing business architecture typically meets an assortment of business and stakeholder needs. If those needs are not recognized or do not continue to be met by a proposed transition or future state, changes are likely to result in a loss of value.

.7 INTERNAL ASSETS

Business analysts identify enterprise assets used in the current state. Resources can be tangible or intangible, such as financial resources, patents, reputation, and brand names.

.8 EXTERNAL INFLUENCERS

There are external influences on the enterprise that do not participate in a change but might present constraints, dependencies, or drivers on the current state. Sources of external influence include:

- **Industry Structure:** individual industries have distinct ways in which value is created within that industry. This is a particularly important influencer if a proposed change involves entering a new industry.
- **Competitors:** the nature and intensity of competitors between enterprises within an industry can be significant. The entry of a new competitor may also change the nature of the industry or increase competition.
- **Customers:** the size and nature of existing and potential customer segments can provide influences such as negotiating power and a degree of price sensitivity. Alternatively, the

emergence of new alternative ways that customers can meet their needs may drive the enterprise to deliver greater value.

- **Suppliers:** the variety and diversity of suppliers might be an influencer, as can the power that suppliers have over their customers.
- **Political and Regulatory Environment:** there is often influence from the current and potential impact of laws and regulations upon the industry.
- **Technology:** the productivity enhancing potential of recent and expected technological innovations might influence the need.
- **Macroeconomic Factors:** the constraints and opportunities that exist within the existing and expected macroeconomic environment (for example, trade, unemployment, or inflation) might influence the need.

Some of these sources might use different terminology, based on whether the enterprise is a for-profit corporation, a non-profit enterprise, or a government agency. For example, a country does not have customers; it has citizens.

6.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Guides how the business analyst undertakes an analysis of the current state.
Enterprise Limitation	Used to understand the challenges that exist within the enterprise.
Organizational Strategy	An organization will have a set of goals and objectives which guides operations, establishes direction, and provides a vision for the future state. This can be implicitly or explicitly stated.
Solution Limitation	Used to understand the current state and the challenges of existing solutions.
Solution Performance Goals	Measure the current performance of an enterprise or solution, and serve as a baseline for setting future state goals and measuring improvement.
Solution Performance Measures	Describe the actual performance of existing solutions.
Stakeholder Analysis Results	Stakeholders from across the organization will contribute to an understanding and analysis of the current state.

6.1.6 TECHNIQUES

Techniques	Usage
Benchmarking and Market Analysis	Provides an understanding of where there are opportunities for improvement in the current state. Specific frameworks that may be useful include 5 forces analysis, pest, steep, catwoe, and others.
Business Capability Analysis	Identifies gaps and prioritizes them in relation to value and risk.
Business Model Canvas	Provides an understanding of the value proposition that the enterprise satisfies for its customers, the critical factors in

	delivering that value, and the resulting cost and revenue streams. Helpful for understanding the context for any change and identifying the problems and opportunities that may have the most significant impact.
Business Cases	Used to capture information regarding the business need and opportunity.
Concept Modelling	Used to capture key terms and concepts in the business domain and define the relationships between them.
Data Mining	Used to obtain information on the performance of the enterprise.
Document Analysis	Analyzes any existing documentation about the current state, including (but not limited to) documents created during the implementation of a solution, training manuals, issue reports, competitor information, supplier agreements, published industry benchmarks, published technology trends, and performance metrics.
Financial Analysis	Used to understand the profitability of the current state and the financial capability to deliver change.
Focus Group	Solicits feedback from customers or end users about the current state.
Functional Decomposition	Breaks down complex systems or relationships in the current state.
Interviews	Facilitate dialogue with stakeholders to understand the current state and any needs evolving from the current state.
Item Tracking	Tracks and manages issues discovered about the current state.
Lessons Learned	Enables the assessment of failures and opportunities for improvement in past initiatives, which may drive a business need for process improvement.
Metrics and Key Performance Indicator (KPIs)	Assesses performance of the current state of an enterprise.
Mind Mapping	Used to explore relevant aspects of the current state and better understand relevant factors affecting the business need.
Observation	May provide opportunities for insights into needs within the current state that have not been identified previously by a stakeholder.
Organizational Modelling	Describes the roles, responsibilities, and reporting structures that exist within the current state organization.
Process Analysis	Identifies opportunities to improve the current state.
Process Modelling	Describes how work occurs within the current solution.
Risk Analysis and Management	Identifies risks to the current state.
Root Cause Analysis	Provides an understanding of the underlying causes of any problems in the current state in order to further clarify a need.
Scope Modelling	Helps define the boundaries on the current state description.

Survey or Questionnaire	Helps to gain an understanding of the current state from a large, varied, or disparate group of stakeholders.
SWOT Analysis	Evaluates the strengths, weaknesses, opportunities, and threats to the current state enterprise.
Vendor Assessment	Determines whether any vendors that are part of the current state are adequately meeting commitments, or if any changes are needed.
Workshops	Engage stakeholders to collaboratively describe the current state and their needs.

6.1.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Makes use of the existing solution and might have input about issues with a current solution.
Domain Subject Matter Expert	Has expertise in some aspect of the current state.
End User	Directly uses a solution and might have input about issues with a current solution.
Implementation Subject Matter Expert	Has expertise in some aspect of the current state.
Operational Support	Directly involved in supporting the operations of the organization and provides information on their ability to support the operation of an existing solution, as well as any known issues.
Project Manager	May use information on current state as input to planning.
Regulator	Can inform interpretations of relevant regulations that apply to the current state in the form of business policies, business rules, procedures, or role responsibilities. The regulator might have unique input to the operational assessment, as there might be new laws and regulations with which to comply.
Sponsor	Might have context for performance of existing solutions.
Supplier	Might be an external influencer of the current state.
Tester	Able to provide information about issues with any existing solutions.

6.1.8 OUTPUTS

- **Current State Description:** the context of the enterprise's scope, capabilities, resources, performance, culture, dependencies, infrastructure, external influences, and significant relationships between these elements.
- **Business Requirements:** the problem, opportunity, or constraint which is defined based on an understanding of the current state.

6.2 DEFINE FUTURE STATE

6.2.1 PURPOSE

The purpose of Define Future State is to determine the set of necessary conditions to meet the business need.

6.2.2 DESCRIPTION

All purposeful change must include a definition of success. Business analysts work to ensure that the future state of the enterprise is well defined, that it is achievable with the resources available, and that key stakeholders have a shared consensus vision of the outcome. As with current state analysis, the purpose of future state analysis is not to create a comprehensive description of the outcome at a level of detail that will directly support implementation. The future state will be defined at a level of detail that:

- allows for competing strategies to achieve the future state to be identified and assessed,
- provides a clear definition of the outcomes that will satisfy the business needs,
- details the scope of the solution space,
- allows for value associated with the future state to be assessed, and
- enables consensus to be achieved among key stakeholders.

The future state description can include any context about the proposed future state. It describes the new, removed, and modified components of the enterprise. It can include changes to the boundaries of the organization itself, such as entering a new market or performing a merger or acquisition. The future state can also be simple changes to existing components of an organization, such as changing a step in a process or removing a feature from an existing application. Change may be needed to any component of the enterprise, including (but not limited to):

- business processes,
- functions,
- lines of business,
- organization structures,
- staff competencies,
- knowledge and skills,
- training,
- facilities,
- desktop tools,
- organization locations,
- data and information,
- application systems, and/or
- technology infrastructure.

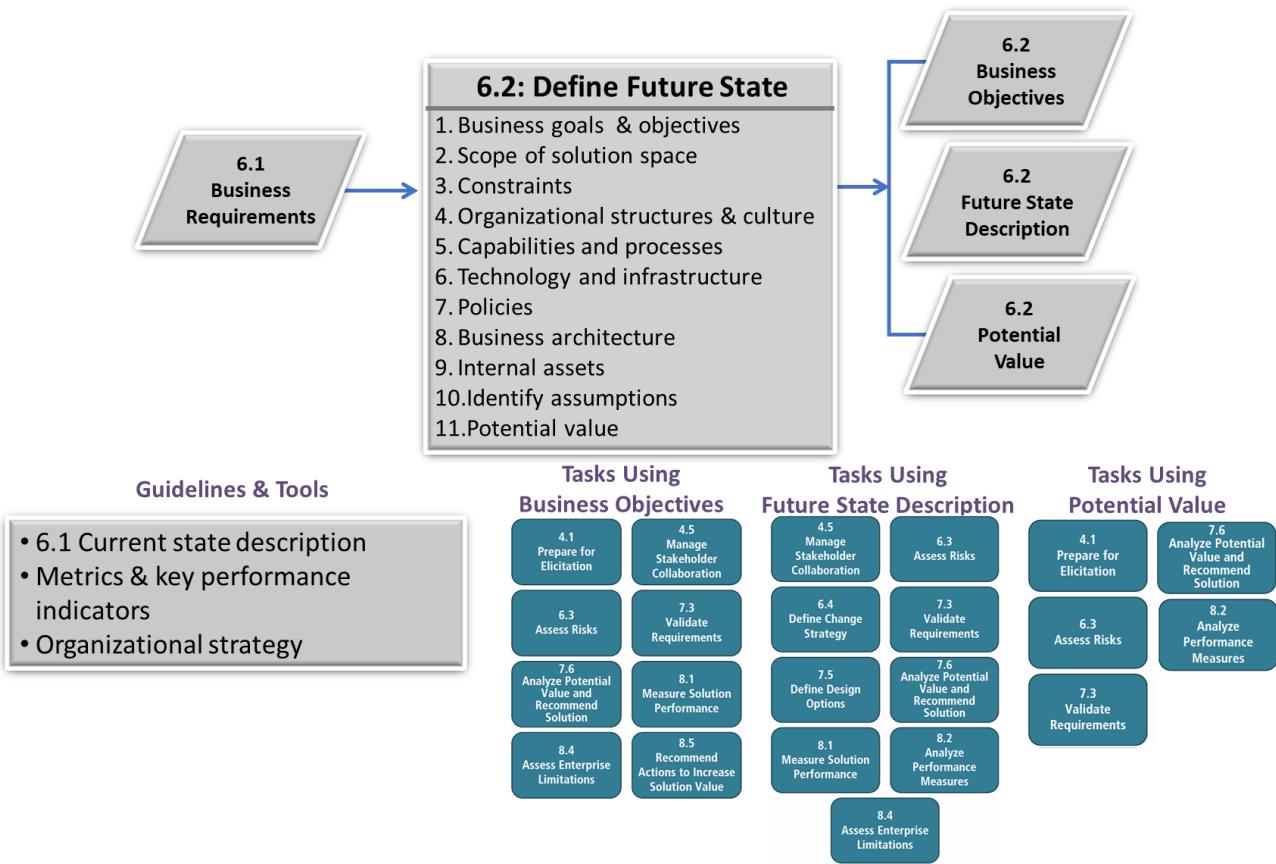
Descriptions may include visual models and text to clearly show the scope boundaries and details. Relevant relationships between entities are identified and described. The effort required to describe the future state varies depending on the nature of the change. The expected outcomes from a change might include specific metrics or loosely defined results. Describing the future state allows stakeholders to understand the potential value that can be realized from a solution, which can be used as part of the decision-making process regarding the change strategy. In environments where changes result in predictable outcomes and predictable delivery of value, and where there are a large number of possible changes that can increase value, the purpose of future state analysis is to gather sufficient information to make the best possible

choices among potential options. In cases where it is difficult to predict the value realized by a change, the future state may be defined by identification of appropriate performance measures (to produce an agreed-upon set of measures for business value), and the change strategy will support exploration of multiple options.

6.2.3 INPUTS

- Business Requirements:** the problems, opportunities, or constraints that the future state will address.

Figure 6.2.1: Define Future State Input/Output Diagram



6.2.4 ELEMENTS

.1 BUSINESS GOALS AND OBJECTIVES

A future state can be described in terms of business objectives or goals in order to guide the development of the change strategy and identify potential value. Business goals and objectives describe the ends that the organization is seeking to achieve. Goals and objectives can relate to changes that the organization wants to accomplish, or current conditions that it wants to maintain.

Goals are longer term, ongoing, and qualitative statements of a state or condition that the organization is seeking to establish and maintain. Examples of business goals include:

- Create a new capability such as a new product or service, address a competitive disadvantage, or create a new competitive advantage.
- Improve revenue by increasing sales or reducing cost.
- Increase customer satisfaction.
- Increase employee satisfaction.
- Comply with new regulations.
- Improve safety.
- Reduce time to deliver a product or service.

High-level goals can be decomposed to break down the general strategy into areas that may lead to desired results, such as increased customer satisfaction, operational excellence, and/or business growth. For example, a goal may be to "increase **number of** high-revenue customers" and then be further refined into a goal to "increase **number of** high revenue customers in the 30-45 age bracket **by 30% within 6 months**".

As goals are analyzed they are converted into more descriptive, granular and specific objectives, and linked to measures that make it possible to objectively assess if the objective has been achieved. Objectives that are measurable enable teams to know if needs were addressed and whether a change was effective. Defining measurable objectives is often critical to justify completing the change and might be a key component to a business case for the change. A common test for assessing objectives is to ensure that they are SMART:

- **S**pecific: describing something that has an observable outcome,
- **M**easurable: tracking and measuring the outcome,
- **A**chievable: testing the feasibility of the effort,
- **R**elevant: aligning with the enterprise's vision, mission, and goals, and
- **T**ime-bounded: defining a time frame that is consistent with the need.

.2 SCOPE OF SOLUTION SPACE

Decisions must be made about the range of solutions that will be considered to meet the business goals and objectives. The scope of the solution space defines which kinds of options will be considered when investigating possible solutions, including changes to the organizational structure or culture, capabilities and processes, technology and infrastructure, policies, products, or services, or even creating or changing relationships with organizations currently outside the scope of the extended enterprise. Solutions in each of these areas generally require specific expertise from both the business analysis and the delivery team. The analysis for this might happen on different levels in the enterprise, and the scope of the solution space is not necessarily related to the size of the change. Even a small change might require looking at the enterprise-level business objectives to ensure alignment.

If multiple future states can meet the business needs, goals and objectives, it will be necessary to determine which ones will be considered. This decision is typically based on the value to be delivered to stakeholders and requires an understanding of possible change strategies. The critical considerations for the decision are dependent on the overall objectives of the enterprise, but will involve an understanding of the quantitative and qualitative value of each option, the time needed to achieve each future state, and the opportunity cost to the enterprise.

.3 CONSTRAINTS

Constraints describe aspects of the current state, aspects of the planned future state that may not be changed by the solution, or mandatory elements of the design. They must be carefully examined to ensure that they are accurate and justified.

Constraints may reflect any of the following:

- budgetary restrictions,
- time restrictions,
- technology,
- infrastructure,
- policies,
- limits on the number of resources available,
- restrictions based on the skills of the team and stakeholders,
- a requirement that certain stakeholders not be affected by the implementation of the solution,
- compliance with regulations, and
- any other restriction.

.4 ORGANIZATIONAL STRUCTURE AND CULTURE

The formal and informal working relationships that exist within the enterprise may need to change to facilitate the desired future state. Changes to reporting lines can encourage teams to work more closely together and facilitate alignment of goals and objectives. Elements of the organizational structure and culture may need to change to support the future state. Describing the components of the future state provides insight into potential conflicts, impact, and limits.

.5 CAPABILITIES AND PROCESSES

Identify new kinds of activities or changes in the way activities will be performed to realize the future state. New or changed capabilities and processes will be needed to deliver new products or services, to comply with new regulations, or to improve the performance of the enterprise.

.6 TECHNOLOGY AND INFRASTRUCTURE

If current technology and infrastructure are insufficient to meet the business need, the business analyst identifies the changes necessary for the desired future state.

The existing technology may impose technical constraints on the design of the solution. These may include development languages, hardware and software platforms, and application software that must be used. Technical constraints may also describe restrictions such as resource utilization, message size and timing, software size, maximum number of and size of files, records, and data elements. Technical constraints include any IT architecture standards that must be followed.

.7 POLICIES

If current policies are insufficient to meet the business need, the business analyst identifies the changes necessary for the desired future state.

Policies are a common source of constraints on a solution or on the solution space. Business policies may mandate what solutions can be implemented given certain levels of approval, the process for obtaining approval, and the necessary criteria a proposed solution must meet in order to receive funding. In some instances, a change to an existing policy may open up alternative solutions that would not otherwise be considered.

.8 BUSINESS ARCHITECTURE

The elements of any future state must effectively support one another and all contribute to meeting the business goals and objectives. In addition, they should be integrated into the overall desired future state of the enterprise as a whole, and support THAT future state.

.9 INTERNAL ASSETS

The analysis of resources might indicate that existing resources need to be increased or require increased capabilities, or that new resources need to be developed. When analyzing resources, business analysts examine the resources needed to maintain the current state and implement the change strategy, and determine what resources can be used as part of a desired future state. The assessment of existing and needed resources is considered when performing a feasibility analysis on possible solution approaches for the change strategy.

.10 IDENTIFY ASSUMPTIONS

Most strategies are predicated on a set of assumptions that will determine whether or not the strategy can succeed, particularly when operating in a highly uncertain environment. It will often be difficult or impossible to prove that the delivery of a new capability will meet a business need, even in cases where it appears reasonable to assume that the new capability will have the desired effect. These assumptions must be identified and clearly understood, so that appropriate decisions can be made if the assumption later proves invalid. Change strategies in uncertain environments can be structured in order to test these assumptions as early as possible to support a redirection or termination of the initiative.

.11 POTENTIAL VALUE

Meeting the business objectives alone does not justify the transition to a future state; the potential value must be evaluated to see if it is sufficient to justify a change.

When defining the future state, business analysts identify the potential value of the solution. The potential value of the future state is the net benefit of the solution after operating costs are accounted for. A change must result in greater value for the enterprise than would be achieved if no action was taken. However, it is possible that the future state will represent a decrease in value from the current state for some stakeholders or even for the enterprise as a whole. New regulations or increased competition, for example, might need to be addressed for the enterprise to remain operating but could still decrease the overall value captured.

While determining the future state, business analysts consider increased or decreased potential value from:

- external opportunities revealed in assessing external influences,
- unknown strengths of new partners,
- new technologies or knowledge,
- potential loss of a competitor in the market, and
- mandated adoption of a change component.

Business analysts identify the specific opportunities for potential alterations in value, as well as the probability of those increases for the individual components of the proposed change. Business analysts estimate a total potential value by aggregating across all opportunities.

The potential value, including the details of the expected benefit and costs and the likely result if no change is made, is a key component to making a business case for the change. Relating descriptions of potential value to measures of actual value currently being achieved enables stakeholders to understand the expected change in value. In most cases, the future state will not

address all of the opportunities for improvement. Any unaddressed opportunities might remain valid after the solution is implemented and should be noted for future analysis in other changes.

In addition to the potential value of the future state, this analysis should consider the acceptable level of investment to reach the future state. While the actual investment will depend on the change strategy, this information guides the selection of possible strategies.

6.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Current State Description	Provides the context within which the work needs to be completed. It is often used as a starting point for the future state.
Metrics and Key Performance Indicators (KPIs)	The key performance indicators and metrics which will be used to determine whether the desired future state has been achieved.
Organizational Strategy	Describes the path, method, or approach an enterprise or organization will take to achieve its desired future state. This can be implicitly or explicitly stated.

6.2.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to identify what may make the future state acceptable and/or how options may be evaluated.
Balanced Scorecard	Used to set targets for measuring the future state
Benchmarking and Market Analysis	Used to make decisions about future state business objectives.
Brainstorming	Used to collaboratively come up with ideas for the future state.
Business Capability Analysis	Used to prioritize capability gaps in relation to value and risk.
Business Cases	Used to capture the desired outcomes of the change initiative.
Business Model Canvas	Used to plan strategy for the enterprise by mapping out the needed infrastructure, target customer base, financial cost structure, and revenue streams required to fulfil the value proposition to customers in the desired future state.
Decision Analysis	Used to compare the different future state options and understand which is the best choice.
Decision Modelling	Used to model complex decisions regarding future state options.
Financial Analysis	Used to estimate the potential financial returns to be delivered by a proposed future state.
Functional Decomposition	Used to break down complex systems within the future state for better understanding.
Interviews	Used to talk to stakeholders to understand their desired future state, which needs they want to address, and what desired

	business objectives they want to meet.
Lessons Learned	Used to determine which opportunities for improvement will be addressed and how the current state can be improved upon.
Metrics and Key Performance Indicators (KPIs)	Used to determine when the organization has succeeded in achieving the business objectives.
Mind Mapping	Used to develop ideas for the future state and understand relationships between them.
Organizational Modelling	Used to describe the roles, responsibilities, and reporting structures that would exist within the future state organization.
Process Modelling	Used to describe how work would occur in the future state.
Prototyping	Used to model future state options and could also help determine potential value.
Scope Modelling	Used to define the boundaries of the enterprise in the future state.
Survey or Questionnaire	Used to understand stakeholders' desired future state, which needs they want to address, and what desired business objectives they want to meet.
SWOT Analysis	Used to evaluate the strengths, weaknesses, opportunities, and threats that may be exploited or mitigated by the future state.
Vendor Assessment	Used to assess potential value provided by vendor solution options.
Workshops	Used to work with stakeholders to collaboratively describe the future state.

6.2.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Might be targeted purchasers or consumers in a future state who might or might not be ready or able to consume a new state.
Domain Subject Matter Expert	Provides insight into current state and potential future states.
End User	Expected to use, or be a component of, a solution that implements the future state.
Implementation Subject Matter Expert	Provides information regarding the feasibility of achieving the future state.
Operational Support	Directly involved in supporting the operations of the enterprise and provides information on their ability to support the operation of a proposed future state.
Project Manager	Might have input on what is a reasonable and manageable desired future state.
Regulator	Ensures that laws, regulations, or rules are adhered to in the desired future state. Interpretations of relevant regulations must be included in the future state description in the form of business policies, business rules, procedures, or role

	responsibilities.
Sponsor	Helps determine which business needs to address and sets the business objectives that a future state will achieve. Authorizes and ensures funding to support moving towards the future state.
Supplier	Might help define the future state if they are supporting delivery of the change or deliver any part of the future state operation.
Tester	Responsible for ensuring an envisioned future state can be sufficiently tested and can help set an appropriate level of quality to target.

6.2.8 OUTPUTS

- **Business Objectives:** the desired direction that the business wishes to pursue in order to achieve the future state.
- **Future State Description:** the future state description includes boundaries of the proposed new, removed, and modified components of the enterprise and the potential value expected from the future state. The description might include the desired future capabilities, policies, resources, dependencies, infrastructure, external influences, and relationships between each element.
- **Potential Value:** the value that may be realized by implementing the proposed future state.

6.3 ASSESS RISKS

6.3.1 PURPOSE

The purpose of Assess Risks is to understand the undesirable consequences of internal and external forces on the enterprise during a transition to, or once in, the future state. An understanding of the potential impact of those forces can be used to make a recommendation about a course of action.

6.3.2 DESCRIPTION

Assessing risks includes analyzing and managing them. Risks might be related to the current state, a desired future state, a change itself, a change strategy, or any tasks being performed by the enterprise.

The risks are analyzed for the:

- possible consequences if the risk occurs,
- impact of those consequences,
- likelihood of the risk, and
- potential time frame when the risk might occur.

The collection of risks is used as an input for selecting or coordinating a change strategy. A risk assessment can include choosing to accept a risk if either the effort required to modify the risk or the level of risk outweighs the probable loss. If the risks are understood and the change proceeds, then the risks can be managed to minimize their overall impact to value.

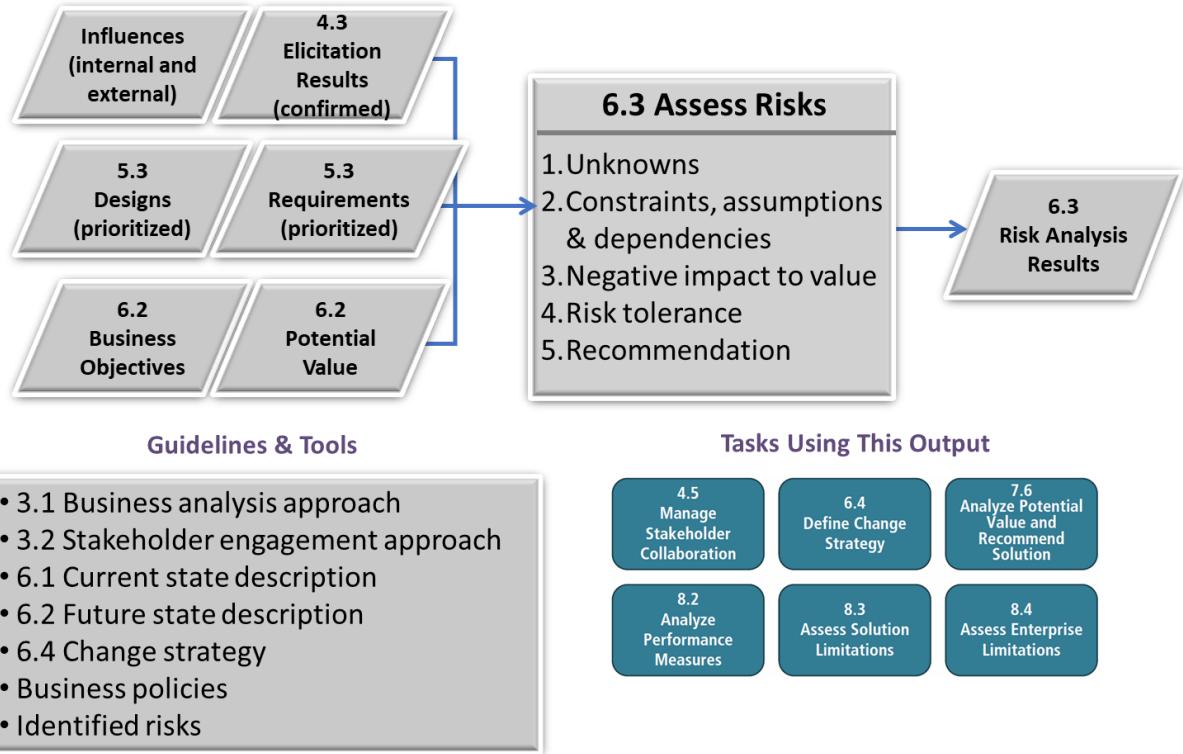
Note: A number of methods include 'positive risk' as a way of managing opportunities. Although the formal definition of risk in the BABOK® Guide doesn't preclude this usage, 'opportunities' are

captured as NEEDS (and managed accordingly), and RISK is used for uncertain events that can produce negative outcomes.

6.3.3 INPUTS

- **Business Objectives:** describing the desired direction needed to achieve the future state can be used to identify and discuss potential risks.
- **Elicitation Results (confirmed):** an understanding of what the various stakeholders perceive as risks to the realization of the desired future state.
- **Influences:** factors inside of the enterprise (internal) and factors outside of the enterprise (external) which will impact the realization of the desired future state.
- **Potential Value:** describing the value to be realized by implementing the proposed future state provides a benchmark against which risks can be assessed.
- **Requirements (prioritized):** depending on their priority, requirements will influence the risks to be defined and understood as part of solution realization.

Figure 6.3.1: Assess Risks Input/Output Diagram



6.3.4 ELEMENTS

.1 UNKNOWNS

When assessing a risk, there will be uncertainty in the likelihood of it occurring, and the impact if it does occur. Business analysts collaborate with stakeholders to assess risks based on current understanding. Even when it is not possible to know all that will occur as a result of a particular change strategy, it is still possible to estimate the impact of unknown or uncertain events or conditions occurring. Business analysts consider other historical contexts from similar situations to assess risks. The lessons learned from past changes and expert judgment from stakeholders assist business analysts in guiding the team in deciding the impact and likelihood of risks for the current change.

.2 CONSTRAINTS, ASSUMPTIONS, AND DEPENDENCIES

Constraints, assumptions, and dependencies can be analyzed for risks and sometimes should be managed as risks themselves. If the constraint, assumption, or dependency is related to an aspect of a change, it can be restated as a risk by identifying the event or condition and consequences that could occur because of the constraint, assumption, or dependency.

.3 NEGATIVE IMPACT TO VALUE

Risks are expressed as conditions that increase the likelihood or severity of a negative impact to value. Business analysts clearly identify and express each risk and estimate its likelihood and impact to determine the level of risk. Business analysts estimate a total risk level from the aggregated set of risks, indicating the overall potential impact for the risks being assessed. In some cases overall risk level can be quantified in financial terms, or in an amount of time, effort, or other measures.

.4 RISK TOLERANCE

How much uncertainty a stakeholder or an enterprise is willing to take on in exchange for potential value is referred to as risk tolerance. In general, there are three broad ways of describing attitude toward risk:

- **Risk-aversion:** An unwillingness to accept much uncertainty; there may be a preference to either avoid a course of action which carries too high a level of risk, or to invest more (and therefore accept a lower potential value) to reduce the risks.
- **Neutrality:** some level of risk is acceptable, provided the course of action does not result in a loss even if the risks occur.
- **Risk-seeking:** A willingness to accept or even take on more risk in return for a higher potential value.

An individual or organization may exhibit different risk tolerances at different times. If there is low tolerance for risk, there may be more effort on avoidance, transfer or mitigation strategies. If the tolerance for risk is high, more risks are likely to be accepted. Typically, the highest level risks are dealt with no matter what the risk tolerance level.

.5 RECOMMENDATION

Based on the analysis of risks, business analysts recommend a course of action. Business analysts work with stakeholders to understand the overall risk level and their tolerance for risk.

The recommendation usually falls into one of the following categories:

- pursue the benefits of a change regardless of the risk,
- pursue the benefits of a change while investing in reducing risk (likelihood and/or impact),
- seek out ways to increase the benefits of a change to outweigh the risk,
- identify ways to manage and optimize opportunities, and
- do not pursue the benefits of a change.

If the change proceeds with risks, stakeholders should be identified to monitor the risks and consequences if the risk event occurs. The risk may alter the current state of the enterprise and require revision of the change strategy. A plan of action in this case may be developed before the risk materializes.

6.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Guides how the business analyst analyzes risks.
Business Policies	Define the limits within which decisions must be made. These may mandate or govern aspects of risk management.

Change Strategy	Provides the plan to transition from the current state to the future state and achieve the desired business outcomes. This approach must be assessed to understand risks associated with the change.
Current State Description	Provides the context within which the work needs to be completed. It can be used to determine risks associated with the current state.
Future State Description	Determines risks associated with the future state.
Identified Risks	Can be used as a starting point for more thorough risk assessment. These can come from risk analysis results, from elicitation activities, from previous business analysis experience, or based on expert opinion.
Stakeholder Engagement Approach	Understanding stakeholders and stakeholder groups helps identify and assess the potential impact of internal and external forces.

6.3.6 TECHNIQUES

Techniques	Usage
Brainstorming	Used to collaboratively identify potential risks for assessment.
Business Cases	Used to capture risks associated with alternative change strategies.
Decision Analysis	Used to assess problems.
Document Analysis	Used to analyze existing documents for potential risks, constraints, assumptions, and dependencies.
Financial Analysis	Used to understand the potential effect of risks on the financial value of the solution.
Interviews	Used to understand what stakeholders think might be risks and the various factors of those risks.
Lessons Learned	Used as a foundation of past issues that might be risks.
Mind Mapping	Used to identify and categorize potential risks and understand their relationships.
Risk Analysis and Management	Used to identify and manage risks.
Root Cause Analysis	Used to identify and address the underlying problem creating a risk.
Survey or Questionnaire	Used to understand what stakeholders think might be risks and the various factors of those risks.
Workshops	Used to understand what stakeholders think might be risks and the various factors of those risks.

6.3.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	Provides input to the risk assessment based on their knowledge of preparation required in their area of expertise.

Implementation Subject Matter Expert	Provides input to the risk assessment based on their knowledge of preparation required in their area of expertise.
Operational Support	Supports the operations of the enterprise and can identify likely risks and their impact.
Project Manager	Helps to assess risk and is primarily responsible for managing and mitigating risk to the project.
Regulator	Identifies any risks associated with adherence to laws, regulations, or rules.
Sponsor	Needs to understand risks as part of authorizing and funding change.
Supplier	There might be risk associated with using a supplier.
Tester	Identifies risks in the change strategy, from a validation or verification perspective.

6.3.8 OUTPUTS

- **Risk Analysis Results:** an understanding of the risks associated with achieving the future state, and the mitigation strategies which will be used to prevent those risks, reduce the impact of the risk, or reduce the likelihood of the risk occurring.

6.4 DEFINE CHANGE STRATEGY

6.4.1 PURPOSE

The purpose of Define Change Strategy is to develop and assess alternative approaches to the change, and then select the recommended approach.

6.4.2 DESCRIPTION

Developing a change strategy is simpler when the current state and the future state are already defined because they provide some context for the change. The change strategy clearly describes the nature of the change in terms of:

- context of the change,
- identified alternative change strategies,
- justification for why a particular change strategy is the best approach,
- investment and resources required to work toward the future state,
- how the enterprise will realize value after the solution is delivered,
- key stakeholders in the change, and
- transition states along the way.

The appropriate representation of a change strategy depends on the perspective of the change team and their stakeholders. The change strategy might be presented as part of a business case, Statement of Work (SOW), an enterprise's strategic plan, or in other formats.

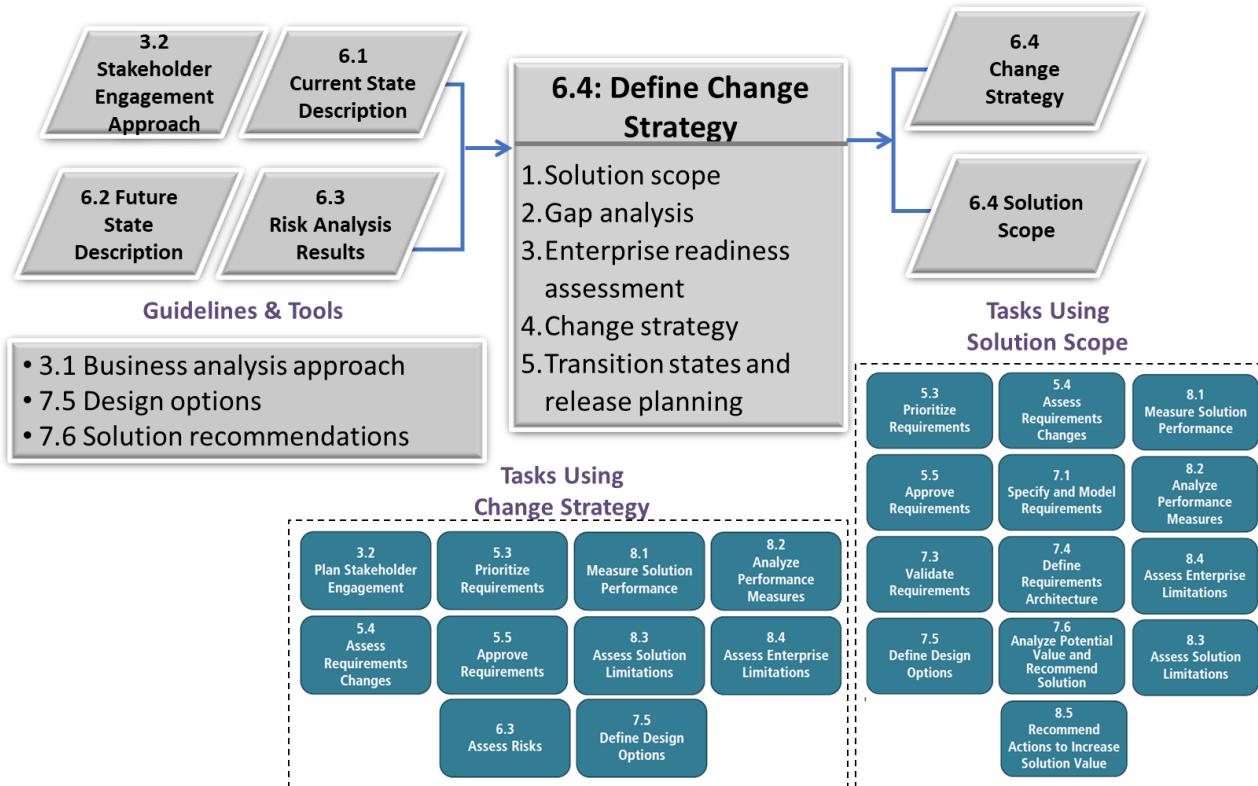
Defining a change strategy usually involves identifying several strategies and ultimately selecting the strategy that is most appropriate for the situation. Change strategies can entail attaining only parts of a future state initially, and therefore include only some components of a complete

solution. For each transition state along the path to reaching the future state, the change strategy should clarify which parts of the solution are completed and which are not, as well as which parts of the value can be realized and which cannot.

6.4.3 INPUTS

- **Current State Description:** provides context about the current state, and includes assessments of internal and external influences to the enterprise under consideration.
- **Future State Description:** provides context about the desired future state.
- **Risk Analysis Results:** describe identified risks and exposure of each risk.
- **Stakeholder Engagement Approach:** understanding stakeholders' communication and collaboration needs can help identify change-related activities that need to be included as part of the change strategy.

Figure 6.4.1: Define Change Strategy Input/Output Diagram



6.4.4 ELEMENTS

.1 SOLUTION SCOPE

The solution is the outcome of a change that allows an enterprise to satisfy a need. Multiple solution options might be evaluated and, as part of a change strategy, the best solution approach is justified and selected. The solution scope defines the boundaries of the solution, and is described in enough detail to enable stakeholders to understand which new capabilities the change will deliver. It also

describes how the proposed solution enables the future state's goals. The solution scope might evolve throughout an initiative as more information is discovered.

The solution scope might be described in different ways, including the use of:

- models and descriptions of markets,
- capabilities,
- technology,
- business rules,
- business decisions,
- data,
- processes,
- resources,
- knowledge and skills,
- functions,
- locations,
- networks,
- organizational structures,
- workflows,
- events,
- sequence,
- motivations, or
- business logic.

The solution scope can also include descriptions of out-of-scope solution components to provide clarity.

.2 GAP ANALYSIS

A gap analysis identifies the difference between current state and future state capabilities. To perform gap analysis, both current state and future state should be defined. Using the same techniques to describe both current and future states assists in gap analysis, as it simplifies the comparison.

Gap analysis can help identify the gaps that prevent the enterprise from meeting needs and achieving goals. It can be used to determine if the enterprise can meet its needs using its existing structure, resources, capabilities, and technology. If the enterprise can meet the need with the current state capabilities, then the change will likely be relatively small, or there may be no change at all. In any other case, a change strategy is needed to create the missing capabilities or improve the existing ones. The capabilities analyzed in a gap analysis can include:

- processes,
- functions,
- lines of business,
- organizational structures,
- staff competencies,
- knowledge and skills,
- training,
- facilities,
- locations,
- data and information,
- application systems, and

- technology infrastructure.

The gaps will need to be addressed in the transition and future states.

.3 ENTERPRISE READINESS ASSESSMENT

Business analysts analyze the enterprise to assess its capacity to make the change and to sustain the change in the future state. The readiness assessment considers the enterprise's capacity not only to make the change, but to use and sustain the solution, and realize value from the solution. The assessment also factors in the cultural readiness of the stakeholders and operational readiness in making the change, the timeline from when the change is implemented to when value can be realized, and the resources available to support the change effort.

.4 CHANGE STRATEGY

A change strategy is a high-level plan of key activities and events that will be used to transform the enterprise from the current state to the future state. Change strategies may be a singular initiative composed of smaller changes which might be structured as a set or sequence of projects, or as various continuous improvement efforts. Each element of change might not completely address the need, so multiple changes might be necessary.

During the course of the development of a change strategy, several options are identified, explored, and described in enough detail to determine which options are feasible. Alternatives can be identified through brainstorming and consulting subject matter experts (SMEs). Sources of ideas can include historical ideas, historical changes, other markets' strategies, and competitors' approaches.

A preferred change strategy is selected from this set of options and developed in more detail. The preferred change strategy should be selected considering:

- organizational readiness to make the change,
- major costs and investments needed to make the change,
- timelines to make the change,
- alignment to the business objectives,
- timelines for value realization, and
- opportunity costs of the change strategy.

Business analysts may develop a business case for each potential change strategy to support decision making. The opportunity cost of each change strategy also needs to be considered. Opportunity cost refers to the benefits that could have been achieved by selecting an alternative change strategy. The options considered and rejected are an important component of the final strategy, providing stakeholders with an understanding of the pros and cons of various approaches to making the change.

When defining the change strategy, the investment to make the change to the future state is also considered. The net benefits of a future state may be very high, but if the investment is unbearable ("they just can't afford the change") the enterprise may pass on the opportunity, and invest in something else.

The potential value, including the details of the expected benefit and costs, are key components to making a business case for the change. Relating descriptions of potential value to measures of actual value currently being achieved enables stakeholders to understand the expected change in value. While every change facilitated by business analysts is intended to increase value, some changes decrease value in parts of an enterprise while increasing it in others.

.5 TRANSITION STATES AND RELEASE PLANNING

In many cases, the future state will need to be achieved over time rather than through a single change, meaning that the enterprise will have to operate in one or more transition states. Release planning is concerned with determining which requirements to include in each release, phase, or iteration of the change. Business analysts help facilitate release planning discussions to help stakeholders reach decisions. There are many factors that guide these decisions, such as the overall budget, deadlines or time constraints, resource constraints, training schedules, and the ability of the business to absorb changes within a defined time frame. There may be organizational restraints or policies that must be adhered to in any implementation. Business analysts assist in planning the timing of the implementation in order to cause minimal disruption to business activities, and to ensure all parties understand the impact to the organization.

6.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Analysis Approach	Guides how the business analyst defines a change strategy.
Design Options	Describe various ways to satisfy the business needs. Each option will come with its own set of change challenges and the change strategy will be impacted by the option selected as well as the specific change approach that will be used.
Solution Recommendations	Identifying the possible solutions which can be pursued in order to achieve the future state, which includes the recommendations of various subject matter experts (SMEs), helps the business analyst determine the types of changes to the organization.

6.4.6 TECHNIQUES

Techniques	Usage
Balanced Scorecard	Used to define the metrics that will be used to evaluate the effectiveness of the change strategy.
Benchmarking and Market Analysis	Used to make decisions about which change strategy is appropriate.
Brainstorming	Used to collaboratively come up with ideas for change strategies.
Business Capability Analysis	Used to prioritize capability gaps in relation to value and risk.
Business Cases	Used to capture information about the recommended change strategy and other potential strategies that were assessed but not recommended.
Business Model Canvas	Used to define the changes needed in the current infrastructure, customer base, and financial structure of the organization in order to achieve the potential value.
Decision Analysis	Used to compare different change strategies and choose which is most appropriate.

Estimation	Used to determine timelines for activities within the change strategy.
Financial Analysis	Used to understand the potential value associated with a change strategy, and evaluate strategies against targets set for return on investments.
Focus Groups	Used to bring customers or end users together to solicit their input on the solution and change strategy.
Functional Decomposition	Used to break down the components of the solution into parts when developing a change strategy.
Interviews	Used to talk to stakeholders in order to fully describe the solution scope and change scope, and to understand their suggestions for a change strategy.
Lessons Learned	Used to understand what went wrong in past changes in order to improve this change strategy
Mind Mapping	Used to develop and explore ideas for change strategies
Organizational Modelling	Used to describe the roles, responsibilities, and reporting structures that are necessary during the change and are part of the solution scope
Process Modelling	Used to describe how work would occur in the solution scope or during the change.
ScopeModelling	Used to define the boundaries on the solution scope and change scope descriptions.
SWOT Analysis	Used to make decisions about which change strategy is appropriate.
VendorAssessment	Used to determine whether any vendors are part of the change strategy, either to implement the change or to be part of the solution.
Workshops	Used in work with stakeholders to collaboratively develop change strategies.

6.4.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Might be purchasing or consuming the solution that results from the change. Customers can also be involved in a change as testers or focus group members, whose input is considered in the enterprise readiness assessment.
Domain Subject Matter Expert	Have expertise in some aspect of the change.
End User	Uses a solution, is a component of the solution, or is a user temporarily during the change. End users could be customers or people who work within the enterprise experiencing a change. Users might be involved in a change as testers or focus group members, whose input is considered in the enterprise readiness assessment.
Implementation Subject Matter Expert	Have expertise in some aspect of the change.

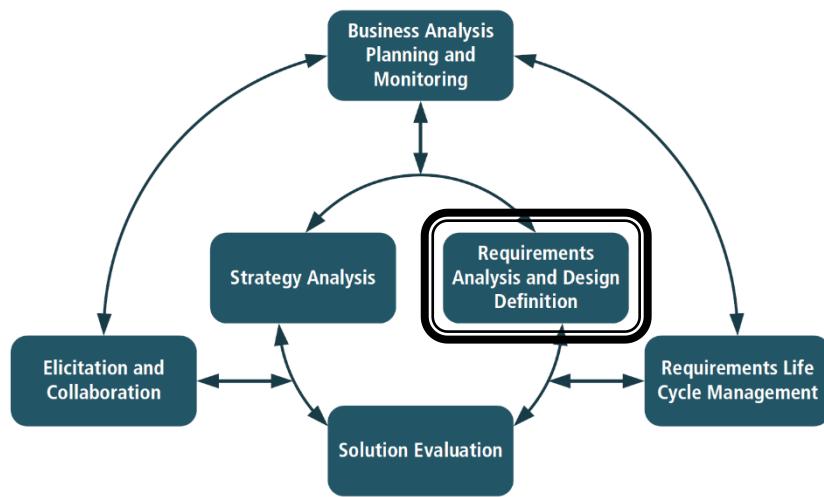
Operational Support	Directly involved in supporting the operations of the enterprise, and provide information on their ability to support the operation of a solution during and after a change.
Project Manager	Responsible for managing change and planning the detailed activities to complete a change. In a project, the project manager is responsible for the project scope, which covers all the work to be performed by the project team.
Regulator	Ensures adherence to laws, regulations, or rules during and at the completion of the change. The regulator might have unique input to the enterprise readiness assessment, as there might be laws and regulations that must be complied with prior to or as a result of a planned or completed change.
Sponsor	Authorizes and ensures funding for solution delivery, and champions the change.
Supplier	Might help implement the change or be part of the solution once the change is completed.
Tester	Responsible for ensuring that the change will function within acceptable parameters, accomplish the desired result, and deliver solutions that meet an appropriate level of quality. The tester is often involved in validation of components of a solution for which the results will be included in an enterprise readiness assessment.

6.4.8 OUTPUTS

- **Change Strategy:** the approach that the organization will follow to guide change.
- **SolutionScope:** the solution scope that will be achieved through execution of the change strategy.

7. Requirements Analysis and Design Definition

The Requirements Analysis and Design Definition knowledge area describes the tasks that business analysts perform to structure and organize requirements discovered during elicitation activities, specify and model requirements and designs, validate and verify information, identify solution options that meet business needs, and estimate the potential value that could be realized for each solution option. This knowledge area covers the incremental and iterative activities ranging from the initial concept and exploration of the need through the transformation of those needs into a particular recommended solution.



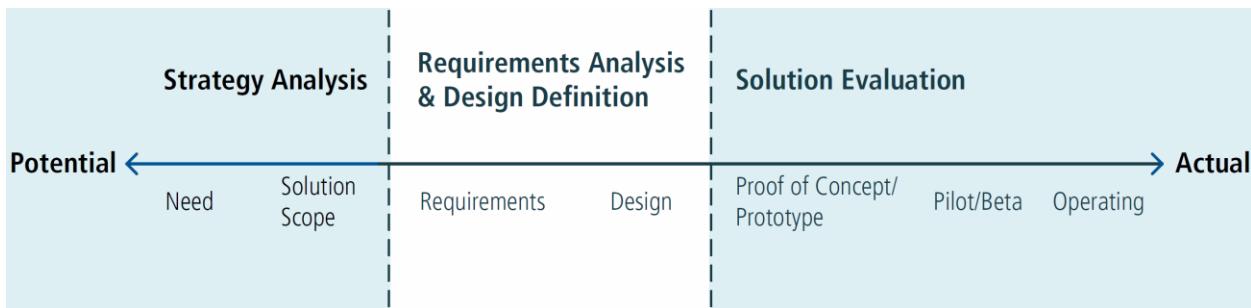
Both requirements and designs are important tools used by business analysts to define and guide change. The main difference between requirements and designs is in how they are used and by whom. One person's designs may be another person's requirements. Requirements and designs may be either high-level or very detailed based upon what is appropriate to those consuming the information.

The business analyst's role in modelling needs, requirements, designs, and solutions is instrumental in conducting thorough analysis and communicating with other stakeholders. The form, level of detail, and what is being modelled are all dependent on the context, audience, and purpose.

Business analysts analyze the potential value of both requirements and designs. In collaboration with implementation subject matter experts, business analysts define solution options that can be evaluated in order to recommend the best solution option that meets the need and brings the most value.

The following image illustrates the spectrum of value as business analysis activities progress from delivering potential value to actual value.

Figure 7.1: Business Analysis Value Spectrum



The Requirements Analysis and Design Definition knowledge area includes the following tasks:

SVV RA O AR

- **Specify and Model Requirements**
- **Verify Requirements**
- **Validate Requirements**
- **Define Requirements Architecture**
- **Define Solution Options**
- **Analyze Potential Value and Recommend Solution**

The Core Concept Model in Requirements Analysis and Design Definition

The Business Analysis Core Concept Model™ (BACCM™) describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Requirements Analysis and Design Definition.

Table 7.1: The Core Concept Model in Requirements Analysis and Design Definition

Core Concept	During Requirements Analysis and Design Definition, business analysts...
Change	Transform elicitation results into requirements and designs in order to define the change.
Need	Analyze the needs in order to recommend a solution that meets the needs.
Solution	Define solution options and recommend the one that is most likely to address the need and has the most value.
Stakeholder	Tailor the requirements and designs so that they are understandable and usable by each stakeholder group.
Value	Analyze and quantify the potential value of the solution options.
Context	Model and describe the context in formats that are understandable and usable by all stakeholders.

7.1 SPECIFY AND MODEL REQUIREMENTS

7.1.1 PURPOSE

The purpose of Specify and Model Requirements is to analyze, synthesize, and refine elicitation results into requirements and designs.

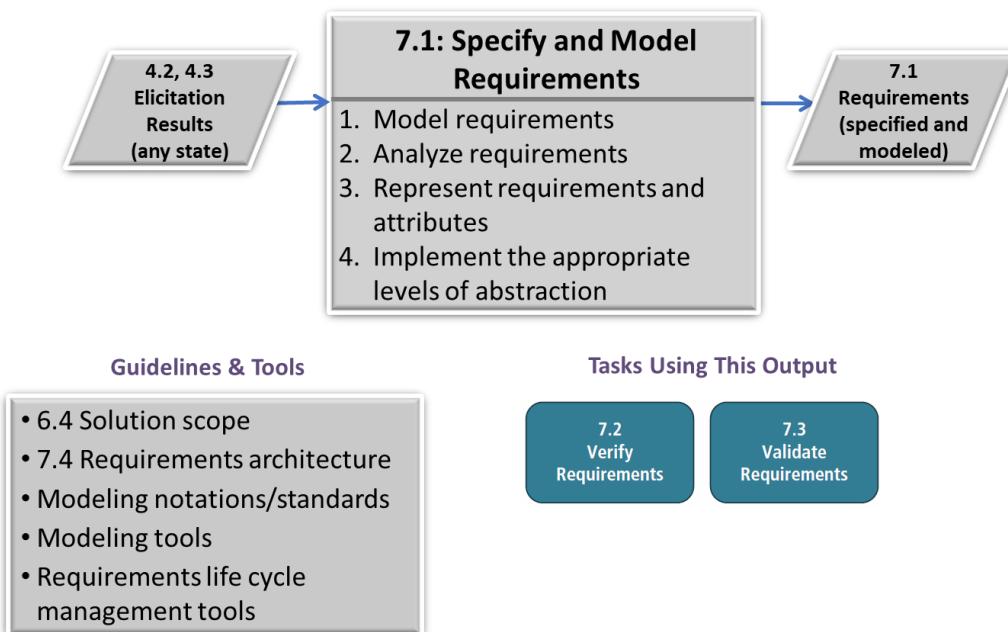
7.1.2 DESCRIPTION

Specify and Model Requirements describes the practices for analyzing elicitation results and creating representations of those results. When the focus of the specifying and modelling activity is on understanding the need, the outputs are referred to as requirements. When the focus of the specifying and modelling activity is on a solution, the outputs are referred to as designs.

Note: In many IT environments, the word 'design' is used specifically for technical designs created by software developers, data architects, and other implementation subject matter experts. All business deliverables are referred to as 'requirements'.

In addition to the models used to represent the requirements, this task also includes capturing information about attributes or metadata about the requirements. The specifying and modelling activities relate to all requirement types.

Figure 7.1.1: Specify and Model Requirements Input/Output Diagram



7.1.3 INPUTS

- **Elicitation Results (any state):** modelling can begin with any elicitation result and may lead to the need for more elicitation to clarify or expand upon requirements. Elicitation and modelling may occur sequentially, iteratively, or concurrently.

7.1.4 ELEMENTS

.1 MODEL REQUIREMENTS

A model is a descriptive and visual way to convey information to a specific audience in order to support analysis, communication, and understanding. Models may also be used to confirm knowledge, identify information gaps that the business analyst may have, and identify duplicate information.

Business analysts choose from one or more of the following modelling formats:

- **Matrices:** a matrix is used when the business analyst is modelling a requirement or set of requirements that have a complex but uniform structure, which can be broken down into elements that apply to every entry in the table. Matrices may be used for data dictionaries, requirements traceability, or for gap analysis. Matrices are also used for prioritizing requirements and recording other requirements attributes and metadata.
- **Diagrams:** a diagram is a visual, often pictorial, representation of a requirement or set of requirements. A diagram is especially useful to depict complexity in a way that would be difficult to do with words. Diagrams can also be used to define boundaries for business domains, to categorize and create hierarchies of items, and to show components of objects such as data and their relationships.
- Using one or more of the model formats, business analysts determine specific categories and specific models within categories to be used. Model categories can include:
- **People and Roles:** models represent organizations, groups of people, roles, and their relationships within an enterprise and to a solution. Techniques used to represent people and their roles include Organizational Modelling, Roles and Permission Matrix, and Stakeholder List, Map, or Personas.
- **Rationale:** models represent the ‘why’ of a change. Techniques used to represent the rationale include Decision Modelling, Scope Modelling, Business Model Canvas, Root Cause Analysis, and Business Rules Analysis.
- **Activity Flow:** models represent a sequence of actions, events, or a course that may be taken. Techniques used to represent activity flows include Process Modelling, Use Cases and Scenarios, and User Stories.
- **Capability:** models focus on features or functions of an enterprise or a solution. Techniques used to represent capabilities include Business Capability Analysis, Functional Decomposition, and Prototyping.
- **Data and Information:** models represent the characteristics and the exchange of information within an enterprise or a solution. Techniques used to represent data and information include Data Dictionary, Data Flow Diagrams, Data Modelling, Glossary, State Modelling, and Interface Analysis.

Business analysts should use any combination of models best suited to meet stakeholder needs in a given context. Each modelling technique has strengths and weaknesses and provides unique insights into the business domain.

.2 ANALYZE REQUIREMENTS

Business analysis information is decomposed into components to further examine for:

- anything that must change to meet the business need,
- anything that should stay the same to meet the business need,
- missing components,
- unnecessary components, and
- any constraints or assumptions that impact the components.

The level of decomposition required, and the level of detail to be specified, varies depending on the knowledge and understanding of the stakeholders, the potential for misunderstanding or miscommunication, organizational standards, and contractual or regulatory obligations, [among other factors](#).

Analysis provides a basis for discussion to reach a conclusion about solution options.

.3 REPRESENT REQUIREMENTS AND ATTRIBUTES

Business analysts identify information for requirements and their attributes as part of the elicitation results. Requirements should be explicitly represented and should include enough detail such that they exhibit the characteristics of requirements and designs quality. Various attributes can be specified for each requirement or set of requirements. These attributes are selected when planning for information management.

As part of specifying requirements, they can also be categorized according to the schema described in task Requirements Classification Schema. Typically, elicitation results contain information of different types, so it is natural to expect that different types of requirements might be specified at the same time. Categorizing requirements can help ensure the requirements are fully understood, a set of any type is complete, and that there is appropriate traceability between the types.

.4 IMPLEMENT THE APPROPRIATE LEVELS OF ABSTRACTION

The level of abstraction of a requirement varies based on the type of requirement and audience for the requirement. Not all stakeholders require or find value in the complete set of requirements and models. It may be appropriate to produce different viewpoints of requirements to represent the same need for different stakeholders. Business analysts take special care to maintain the meaning and intent of the requirements over all representations.

The business analysis approach may also influence the level of abstraction and choice of models used when defining requirements.

7.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Modelling Notations/Standards	Allow requirements and designs to be precisely specified, as is appropriate for the audience and the purpose of the models. Standard templates and syntax help to ensure that the right information is provided about the requirements.
Modelling Tools	Software products that facilitate drawing and storing matrices and diagrams to represent requirements. This functionality may or may not be part of requirements life cycle management tools.
Requirements Architecture	The requirements and interrelationships among them can be used to ensure models are complete and consistent.

Requirements Life Cycle Management Tools	Software products that facilitate recording, organizing, storing, and sharing requirements and designs.
Solution Scope	The boundaries of the solution provide the boundaries for the requirements and designs models.

7.1.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to represent the acceptance and evaluation criteria attributes of requirements.
Business Capability Analysis	Used to represent features or functions of an enterprise.
Business Model Canvas	Used to describe the rationale for requirements.
Business Rules Analysis	Used to analyze business rules so that they can be specified and modelled alongside requirements.
Concept Modelling	Used to define terms and relationships relevant to the change and the enterprise.
Data Dictionary	Used to record details about the data involved in the change. Details may include definitions, relationships with other data, origin, format, and usage.
Data Flow Diagrams	Used to visualize data flow requirements.
Data Modelling	Used to model requirements to show how data will be used to meet stakeholder information needs.
Decision Modelling	Used to represent decisions in a model in order to show the elements of decision making required.
Functional Decomposition	Used to model requirements in order to identify constituent parts of an overall complex business function.
Glossary	Used to record the meaning of relevant business terms while analyzing requirements.
Interface Analysis	Used to model requirements in order to identify and validate inputs and outputs of the solution they are modelling
Non-Functional Requirements Analysis	Used to define and analyze the quality of service attributes.
Organizational Modelling	Used to allow business analysts to model the roles, responsibilities, and communications within an organization.
Process Modelling	Used to show the steps or activities that are performed in the organization, or that must be performed to meet the desired change.
Prototyping	Used to assist the stakeholders in visualizing the appearance and capabilities of a planned solution.
Roles and Permissions Matrix	Used to specify and model requirements concerned with the separation of duties among users and external interfaces in utilizing a solution.
Root Cause Analysis	Used to model the root causes of a problem as part of rationale.
Scope Modelling	Used to visually show a scope boundary.

Sequence Diagrams	Used to specify and model requirements to show how processes operate and interact with one another, and in what order.
Stakeholder List, Map, or Personas	Used to identify the stakeholders and their characteristics.
State Modelling	Used to specify the different states of a part of the solution throughout a life cycle, in terms of the events that occur.
Use Cases and Scenarios	Used to model the desired behaviour of a solution, by showing user interactions with the solution, to achieve a specific goal or accomplish a particular task.
User Stories	Used to specify requirements as a brief statement about what people do or need to do when using the solution.

7.1.7 STAKEHOLDERS

Stakeholder	Contribution
Any stakeholder	Business analysts may choose to perform this task themselves and then separately package and communicate the requirements to stakeholders for their review and approval, or they might choose to invite some or all stakeholders to participate in this task.

7.1.8 OUTPUTS

- **Requirements(specified and modelled)**:any combination of requirements and/or designs in the form of text, matrices, and diagrams.

7.2 VERIFY REQUIREMENTS

7.2.1 PURPOSE

The purpose of Verify Requirements is to ensure that requirements and designs specifications and models meet quality standards and are usable for the purpose they serve.

7.2.2 DESCRIPTION

Verifying requirements ensures that the requirements and designs have been defined correctly. Requirements verification constitutes a check by the business analyst and key stakeholders to determine that the requirements and designs are ready for validation, and provides the information needed for further work to be performed.

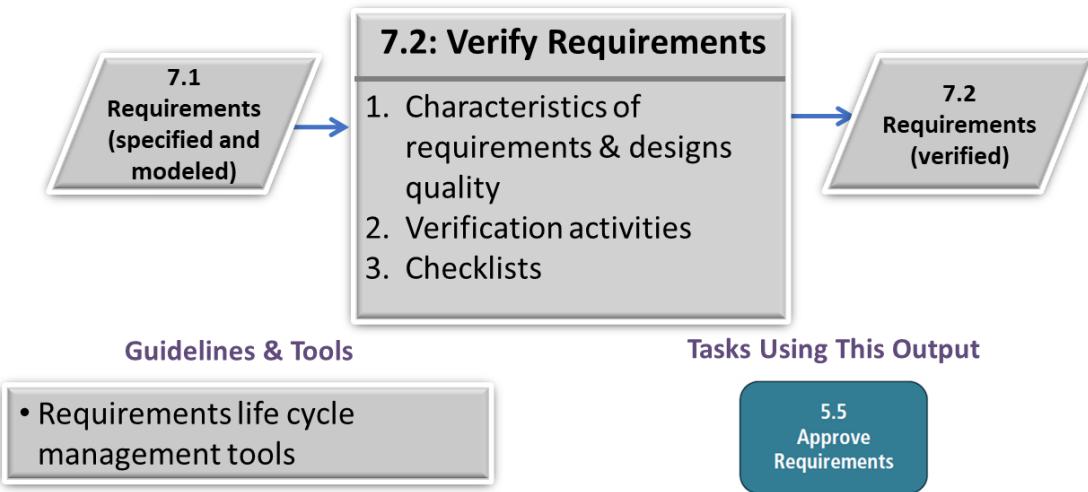
A high-quality specification is well written and easily understood by its intended audience. A high-quality model follows the formal or informal notation standards and effectively represents reality.

The most important characteristic of quality requirements and designs is fitness for use. They must meet the needs of stakeholders who will use them for a particular purpose. Quality is ultimately determined by stakeholders.

7.2.3 INPUTS

- **Requirements (specified and modelled):** any requirement, design, or set of those may be verified to ensure that text is well structured and that matrices and modelling notation are used correctly.

Figure 7.2.1: Verify Requirements Input/Output Diagram



7.2.4 ELEMENTS

.1 CHARACTERISTICS OF REQUIREMENTS AND DESIGNS QUALITY

While quality is ultimately determined by the needs of the stakeholders who will use the requirements or the designs, acceptable quality requirements exhibit many of the following characteristics:

- **Atomic:** self-contained and capable of being understood independently of other requirements or designs.
- **Complete:** enough to guide further work and at the appropriate level of detail for work to continue. The level of completeness required differs based on perspective or methodology, as well as the point in the life cycle where the requirement is being examined or represented.
- **Consistent:** aligned with the identified needs of the stakeholders and not conflicting with other requirements.
- **Concise:** contains no extraneous and unnecessary content.
- **Feasible:** reasonable and possible within the agreed-upon risk, schedule, and budget, or considered feasible enough to investigate further through experiments or prototypes.
- **Unambiguous:** the requirement must be clearly stated in such a way to make it clear whether a solution does or does not meet the associated need.
- **Testable:** able to verify that the requirement or design has been fulfilled. Acceptable levels of verifying fulfillment depend on the level of abstraction of the requirement or design.
- **Prioritized:** ranked, grouped, or negotiated in terms of importance and value against all other requirements.
- **Understandable:** represented using common terminology of the audience.

.2 VERIFICATION ACTIVITIES

Verification activities are typically performed iteratively throughout the requirements analysis process. Verification activities include:

- checking for compliance with organizational performance standards for business analysis, such as using the right tools and methods,
- checking for correct use of modelling notation, templates, or forms,
- checking for completeness within each model,
- comparing each model against other relevant models, checking for elements that are mentioned in one model but are missing in other models, and verifying that the elements are referenced consistently,
- ensuring the terminology used in expressing the requirement is understandable to stakeholders and consistent with the use of those terms within the organization, and
- adding examples where appropriate for clarification.

.3 CHECKLISTS

Checklists are used for quality control when verifying requirements and designs. Checklists may include a standard set of quality elements that business analysts use to verify the requirements, or they may be specifically developed to capture issues of concern. The purpose of a checklist is to ensure that items determined to be important are included in the final requirements deliverables, or that steps required for the verification process are followed.

7.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Requirements Life Cycle Management Tools	Some tools have functionality to check for issues related to many of the characteristics, such as atomic, unambiguous, and prioritized.

7.2.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to ensure that requirements are stated clearly enough to devise a set of tests that can prove that the requirements have been met.
Item Tracking	Used to ensure that any problems or issues identified during verification are managed and resolved.
Metrics and Key Performance Indicators (KPIs)	Used to identify how to evaluate the quality of the requirements.
Reviews	Used to inspect requirements documentation to identify requirements that are not of acceptable quality.

7.2.7 STAKEHOLDERS

Stakeholder	Contribution
All stakeholders	The business analyst, in conjunction with the domain and

implementation subject matter experts, has the primary responsibility for determining that this task has been completed. Other stakeholders may discover problematic requirements during requirements communication. Therefore, all stakeholders could be involved in this task.

7.2.8 OUTPUTS

- **Requirements (verified):** a set of requirements or designs that is of sufficient quality to be used as a basis for further work.

7.3 VALIDATE REQUIREMENTS

7.3.1 PURPOSE

The purpose of Validate Requirements is to ensure that all requirements and designs align to the business requirements and support the delivery of needed value.

7.3.2 DESCRIPTION

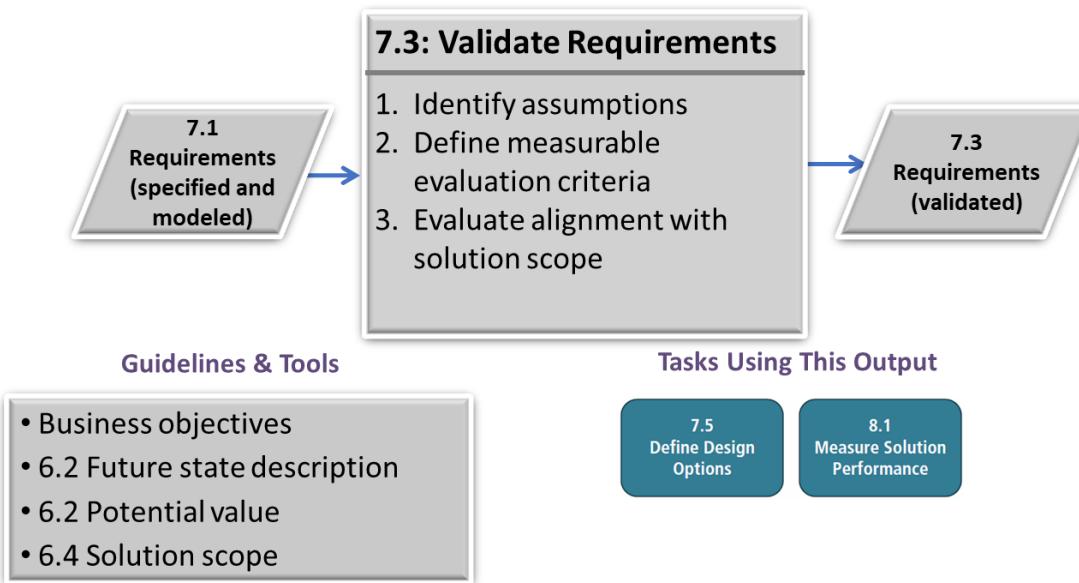
Requirements validation is an ongoing process to ensure that stakeholder, solution, and transition requirements align to the business requirements and that the designs satisfy the requirements.

Understanding what the desired future state looks like for stakeholders after their needs have been met is valuable to business analysts when validating requirements. The overall goal of implementing the requirements is to achieve the stakeholders' desired future state. In many cases, stakeholders have different, conflicting needs and expectations that may be exposed through the validation process.

7.3.3 INPUTS

- **Requirements (specified and modelled):** any types of requirements and designs can be validated. Validation activities may begin before requirements are completely verified. However, validation activities cannot be completed before requirements are completely verified.

Figure 7.3.1: Validate Requirements Input/Output Diagram



7.3.4 ELEMENTS

.1 IDENTIFY ASSUMPTIONS

If an organization is launching an unprecedented product or service, it may be necessary to make assumptions about customer or stakeholder response, as there are no similar previous experiences on which to rely. In other cases, it may be difficult or impossible to prove that a particular problem derives from an identified root cause. Stakeholders may have assumed that certain benefits will result from the implementation of a requirement. These assumptions are identified and defined so that associated risks can be managed.

.2 DEFINE MEASURABLE EVALUATION CRITERIA

While the expected benefits are defined as part of the future state, the specific measurement criteria and evaluation process may not have been included. Business analysts define the evaluation criteria that will be used to evaluate how successful the change has been after the solution is implemented. Baseline metrics might be established based on the current state. Target metrics can be developed to reflect the achievement of the business objectives or some other measurement of success.

.3 EVALUATE ALIGNMENT WITH SOLUTION SCOPE

A requirement can be of benefit to a stakeholder and still not be a desirable part of a solution. A requirement that does not deliver benefit to a stakeholder is a strong candidate for elimination. When requirements do not align, either the future state must be re-evaluated and the solution scope changed, or the requirement removed from the solution scope.

If a design cannot be validated to support a requirement, there might be a missing or misunderstood requirement, or the design must change.

7.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Objectives	Ensure the requirements deliver the desired business benefits.
Future State Description	Helps to ensure the requirements that are part of the solution scope do help achieve the desired future state.
Potential Value	Can be used as a benchmark against which the value delivered by requirements can be assessed.
Solution Scope	Ensures the requirements that provide benefit are within the scope of the desired solution.

7.3.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to define the quality metrics that must be met to achieve acceptance by a stakeholder.
Document Analysis	Used to identify previously documented business needs in order to validate requirements.
Financial Analysis	Used to define the financial benefits associated with requirements.
Item Tracking	Used to ensure that any problems or issues identified during validation are managed and resolved.
Metrics and Key Performance Indicators (KPIs)	Used to select appropriate performance measures for a solution, solution component, or requirement.
Reviews	Used to confirm whether or not the stakeholder agrees that their needs are met.
Risk Analysis and Management	Used to identify possible scenarios that would alter the benefit delivered by a requirement.

7.3.7 STAKEHOLDERS

Stakeholder	Contribution
All stakeholders	The business analyst, in conjunction with the customer, end users, and sponsors, has the primary responsibility for determining whether or not requirements are validated. Other stakeholders may discover problematic requirements during requirements communication. Therefore, virtually all project stakeholders are involved in this task.

7.3.8 OUTPUTS

- **Requirements (validated):** validated requirements and designs are those that can be demonstrated to deliver benefit to stakeholders and align with the business goals and objectives of the change. If a requirement or design cannot be validated, it either does not benefit the organization, does not fall within the solution scope, or both.

7.4 DEFINE REQUIREMENTS ARCHITECTURE

7.4.1 PURPOSE

The purpose of Define Requirements Architecture is to ensure that the requirements collectively support one another to fully achieve the objectives.

7.4.2 DESCRIPTION

Requirements architecture is the structure of all of the requirements of a change. A requirements architecture fits the individual models and specifications together to ensure that all of the requirements form a single whole that supports the overall business objectives and produces a useful outcome for stakeholders. Business analysts use a requirements architecture to:

- understand which models are appropriate for the domain, solution scope, and audience,
- organize requirements into structures relevant to different stakeholders,
- illustrate how requirements and models interact with and relate to each other, and show how the parts fit together into a meaningful whole,
- ensure the requirements work together to achieve the overall objectives, and
- make trade-off decisions about requirements while considering the overall objectives.

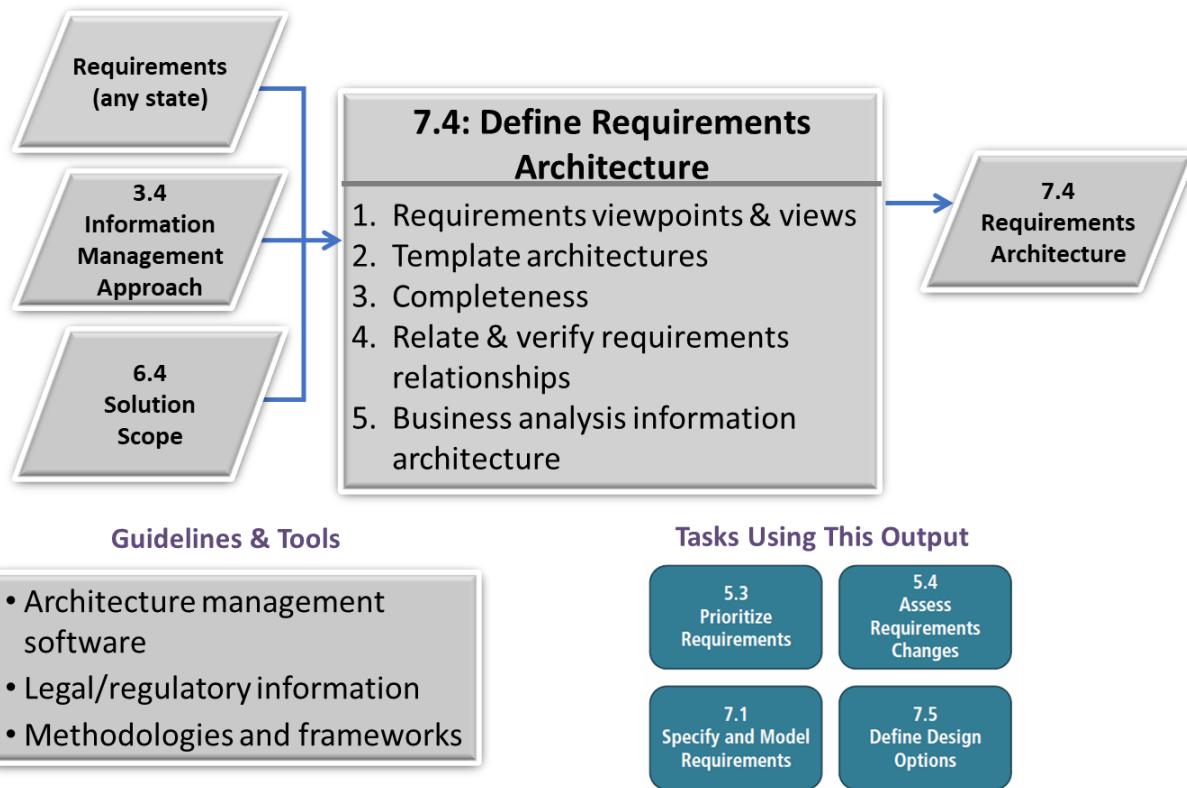
Requirements architecture is not intended to demonstrate traceability, but rather to show how elements work in harmony with one another to support the business requirements, and to structure them in various ways to align the viewpoints of different stakeholders. Traceability is often used as the mechanism to represent and manage these relationships. Traceability proves that every requirement links back to an objective and shows how an objective was met.

Traceability does not prove the solution is a cohesive whole that will work.

7.4.3 INPUTS

- **Information Management Approach:** defines how the business analysis information (including requirements and models) will be stored and accessed.
- **Requirements (any state):** every requirement should be stated once, and only once, and incorporated into the requirements architecture so that the entire set may be evaluated for completeness.
- **Solution Scope:** must be considered to ensure the requirements architecture is aligned with the boundaries of the desired solution.

Figure 7.4.1: Define Requirements Architecture Input/Output Diagram



7.4.4 ELEMENTS

.1 REQUIREMENTS VIEWPOINTS AND VIEWS

A viewpoint is a set of conventions that define how requirements will be represented, how these representations will be organized, and how they will be related. Viewpoints provide templates for addressing the concerns of particular stakeholder groups. Requirements viewpoints frequently include standards and guidelines for the:

- model types used for requirements,
- attributes that are included and consistently used in different models,
- model notations that are used, and
- analytical approaches used to identify and maintain relevant relationships among models.

No single viewpoint alone can form an entire architecture. Each viewpoint is stronger for some aspects of the requirements, and weaker for others, as different groups of stakeholders have different concerns. Trying to put too much information into any one viewpoint will make it too complex and degrade its purpose. Examples of viewpoints include:

- Business process models,
- Data models and information,
- User interactions, including use cases and/or user experience,
- Audit and security, and
- Business models.

Each of those viewpoints has different model notations and techniques, and each is important to ensure a cohesive final solution. The solution would likely not be a success if the business analyst only looked at the business process viewpoint. Similarly, trying to put conventions from many viewpoints in one single viewpoint would make it overwhelming to analyze and contain information irrelevant to particular stakeholder groups.

The actual requirements and designs for a particular solution from a chosen viewpoint are referred to as a view. A collection of views makes up the requirements architecture for a specific solution. Business analysts align, coordinate, and structure requirements into meaningful views for the various stakeholders. This set of coordinated, complementary views provides a basis for assessing the completeness and coherence of the requirements.

In short, the viewpoints tell business analysts what information they should provide for each stakeholder group to address their concerns, while views describe the actual requirements and designs that are produced.

.2 TEMPLATE ARCHITECTURES

An architectural framework is a collection of viewpoints that is standard across an industry, sector, or organization. Business analysts can treat frameworks as predefined templates to start from in defining their architecture. Similarly, the framework can be populated with domain-specific information to form a collection of views that is an even more useful template to build architecture from if it is accurate because the information is already populated in it.

.3 COMPLETENESS

An architecture helps ensure that a set of requirements is complete. The entire set of requirements should be able to be understood by the audience in way that it can be determined that the set is cohesive and tells a full story. No requirements should be missing from the set, inconsistent with others, or contradictory to one another. The requirements architecture should take into account any dependencies between requirements that could keep the objectives from being achieved.

Structuring requirements according to different viewpoints helps ensure this completeness. Iterations of elicitation, specification, and analysis activities can help identify gaps.

.4 RELATE AND VERIFY REQUIREMENTS RELATIONSHIPS

Requirements may be related to each other in several ways when defining the requirements architecture. Business analysts examine and analyze the requirements to define the relationships between them. The representation of these relationships is provided by tracing requirements.

Business analysts examine each relationship to ensure that the relationships satisfy the following quality criteria:

- **Defined:** there is a relationship and the type of the relationship is described.
- **Necessary:** the relationship is necessary for understanding the requirements holistically.
- **Correct:** the elements do have the relationship described.
- **Unambiguous:** there are no relationships that link elements in two different and conflicting ways.
- **Consistent:** relationships are described in the same way, using the same set of standard descriptions as defined in the viewpoints.

.5 BUSINESS ANALYSIS INFORMATION ARCHITECTURE

The structure of the business analysis information is also an information architecture. This type of architecture is defined as part of the task **Plan Business Analysis Information Management**. The information architecture is a component of the requirements architecture because it describes how all of the business analysis information for a change relates. It defines relationships for types of information such as requirements, designs, types of models, and elicitation results. Understanding this type of information structure helps to ensure that the full set of requirements is complete by verifying the relationships are complete. It is useful to start defining this architecture before setting up infrastructure such as requirements life cycle management tools, architecture management software, or document repositories.

7.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Architecture Management Software	Modelling software can help to manage the volume, complexity, and versions of the relationships within the requirements architecture.
Legal/Regulatory Information	Describes legislative rules or regulations that must be followed. They may impact the requirements architecture or its outputs. Additionally, contractual or standards-based constraints may also need to be considered.
Methodologies and Frameworks	A predetermined set of models, and relationships between the models, to be used to represent different viewpoints.

7.4.6 TECHNIQUES

Techniques	Usage
Data Modelling	Used to describe the requirements structure as it relates to data.
Functional Decomposition	Used to break down an organizational unit, product scope, or other elements into its component parts.
Interviews	Used to define the requirements structure collaboratively.
Organizational Modelling	Used to understand the various organizational units, stakeholders, and their relationships which might help define relevant viewpoints.
Scope Modelling	Used to identify the elements and boundaries of the requirements architecture.
Workshops	Used to define the requirements structure collaboratively.

7.4.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert, Implementation Subject Matter Expert, Project Manager, Sponsor, Tester	May assist in defining and confirming the requirements architecture.

Any stakeholder	May also use the requirements architecture to assess the completeness of the requirements.
------------------------	--

7.4.8 OUTPUTS

- **Requirements Architecture:** the requirements and the interrelationships among them, as well as any contextual information that is recorded.

7.5 DEFINE DESIGN OPTIONS

7.5.1 PURPOSE

The purpose of Define Design Options is to define the solution approach, identify opportunities to improve the business, allocate requirements across solution components, and represent design options that achieve the desired future state.

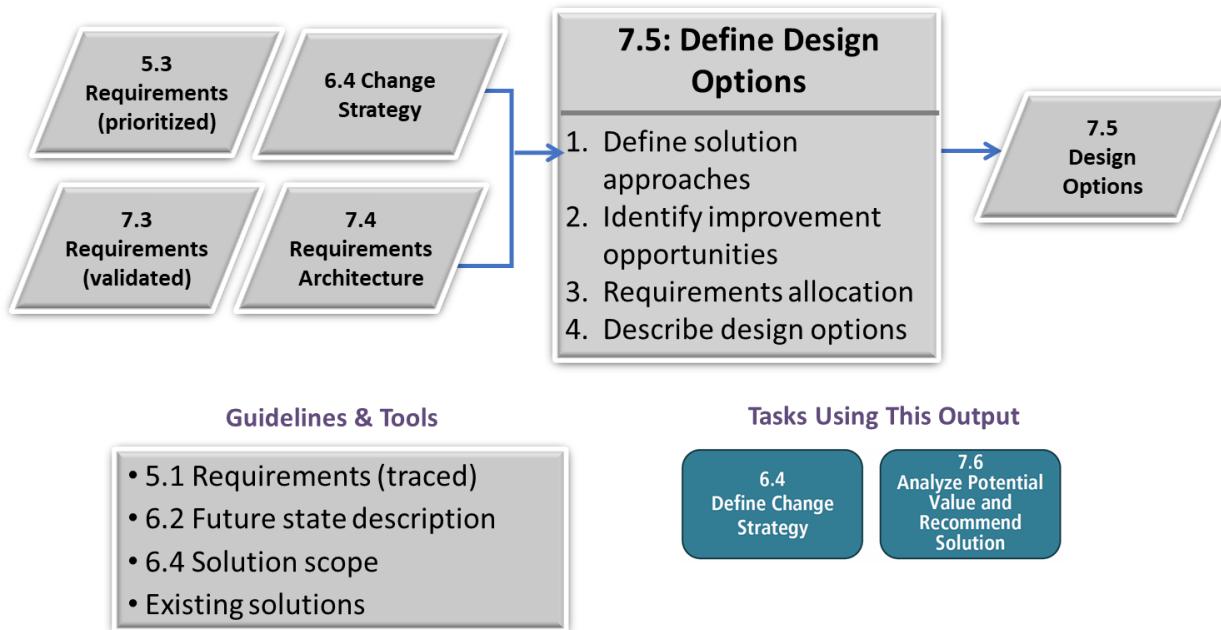
7.5.2 DESCRIPTION

When designing a solution, there may be one or more design options identified. Each design option represents a way to satisfy a set of requirements. Design options exist at a lower level than the change strategy, and are tactical rather than strategic. As a solution is developed, tactical trade-offs may need to be made among design alternatives. Business analysts must assess the effect these trade-offs will have on the delivery of value to stakeholders. As initiatives progress and requirements evolve, design options evolve as well.

7.5.3 INPUTS

- **Change Strategy:** describes the approach that will be followed to transition to the future state. This may have some impact on design decisions in terms of what is feasible or possible.
- **Requirements (validated, prioritized):** only validated requirements are considered in design options. Knowing the requirement priorities aids in the suggestion of reasonable design options. Requirements with the highest priorities might deserve more weight in choosing solution components to best meet them as compared to lower priority requirements.
- **Requirements Architecture:** the full set of requirements and their relationships is important for defining design options that can address the holistic set of requirements.

Figure 7.5.1: Define Design Options Input/Output Diagram



7.5.4 ELEMENTS

.1 DEFINE SOLUTION APPROACHES

The solution approach describes whether solution components will be created or purchased, or some combination of both. Business analysts assess the merits of the solution approaches for each design option. Solution approaches include:

- **Create:** solution components are assembled, constructed, or developed by experts as a direct response to a set of requirements. The requirements and the design options have enough detail to make a decision about which solution to construct. This option includes modifying an existing solution.
- **Purchase:** solution components are selected from a set of offerings that fulfill the requirements. The requirements and design options have enough detail to make a recommendation about which solution to purchase. These offerings are usually products or services owned and maintained by third parties.
- **Combination of both:** not all design options will fall strictly into one of the categories above. Design options may include a combination of both creation and purchase of components.

In all of these types of approaches, proposed integration of the components is also considered within the design option.

.2 IDENTIFY IMPROVEMENT OPPORTUNITIES

When proposing design options, a number of opportunities to improve the operation of the business may occur and are compared. Some common examples of opportunities include:

- **Increase Efficiencies:** automate or simplify the work people perform by re-engineering or sharing processes, changing responsibilities, or outsourcing. Automation may also increase consistency of behaviour, reducing the likelihood of different stakeholders performing the same function in distinctly different fashions.
- **Improve Access to Information:** provide greater amounts of information to staff who interface directly or indirectly with customers, thereby reducing the need for specialists.
- **Identify Additional Capabilities:** highlight capabilities that have the potential to provide future value and can be supported by the solution. These capabilities may not necessarily be of immediate value to the organization (for example, a software application with features the organization anticipates using in the future).

.3 REQUIREMENTS ALLOCATION

Requirements allocation is the process of assigning requirements to solution components and releases to best achieve the objectives. Allocation is supported by assessing the trade-offs between alternatives in order to maximize benefits and minimize costs. The value of a solution might vary depending on how requirements are implemented and when the solution becomes available to stakeholders. The objective of allocation is to maximize that value.

Requirements may be allocated between organizational units, job functions, solution components, or releases of a solution. Requirements allocation typically begins when a solution approach has been determined, and continues until all valid requirements are allocated. Allocation typically continues through design and implementation of a solution.

.4 DESCRIBE DESIGN OPTIONS

Design options are investigated and developed while considering the desired future state, and in order to ensure the design option is valid. Solution performance measures are defined for each design option.

A design option usually consists of many design components, each described by a design element. Design elements may describe:

- business policies and business rules,
- business processes to be performed and managed,
- people who operate and maintain the solution, including their job functions and responsibilities,
- operational business decisions to be made,
- software applications and application components used in the solution, and
- organizational structures, including interactions between the organization, its customers, and its suppliers.

7.5.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Existing Solutions	Existing products or services, often third party, that are considered as a component of a design option
Future State Description	Identifies the desired state of the enterprise that the design options will be part of, and helps to ensure design options are viable.
Requirements (traced)	Define the design options that best fulfil known requirements.

Solution Scope	Defines the boundaries when selecting viable design options.
-----------------------	--

7.5.6 TECHNIQUES

Techniques	Usage
Benchmarking and Market Analysis	Used to identify and analyze existing solutions and market trends.
Brainstorming	Used to help identify improvement opportunities and design options
Document Analysis	Used to provide information needed to describe design options and design elements.
Interviews	Used to help identify improvement opportunities and design options.
Lessons Learned	Used to help identify improvement opportunities.
Mind Mapping	Used to identify and explore possible design options.
Root Cause Analysis	Used to understand the underlying cause of the problems being addressed in the change to propose solutions to address them.
Survey or Questionnaire	Used to help identify improvement opportunities and design options.
Vendor Assessment	Used to couple the assessment of a third party solution with an assessment of the vendor to ensure that the solution is viable and all parties will be able to develop and maintain a healthy working relationship.
Workshops	Used to help identify improvement opportunities and design options.

7.5.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	Provides the expertise within the business to provide input and feedback when evaluating solution alternatives, particularly for the potential benefits of a solution.
Implementation Subject Matter Expert	Use their expertise in terms of the design options being considered to provide needed input about the constraints of a solution and its costs.
Operational Support	Can help evaluate the difficulty and costs of integrating proposed solutions with existing processes and systems.
Project Manager	Plans and manages the solution definition process, including the solution scope and any risks associated with the proposed solutions.
Supplier	Provides information on the functionality associated with a particular design option.

7.5.8 OUTPUTS

- **Design Options:** describe various ways to satisfy one or more needs in a context. They may include solution approach, potential improvement opportunities provided by the option, and the components that define the option.

7.6 ANALYZE POTENTIAL VALUE AND RECOMMEND SOLUTION

7.6.1 PURPOSE

The purpose of Analyze Potential Value and Recommend Solution is to estimate the potential value for each design option and to establish which one is most appropriate to meet the enterprise's requirements.

7.6.2 DESCRIPTION

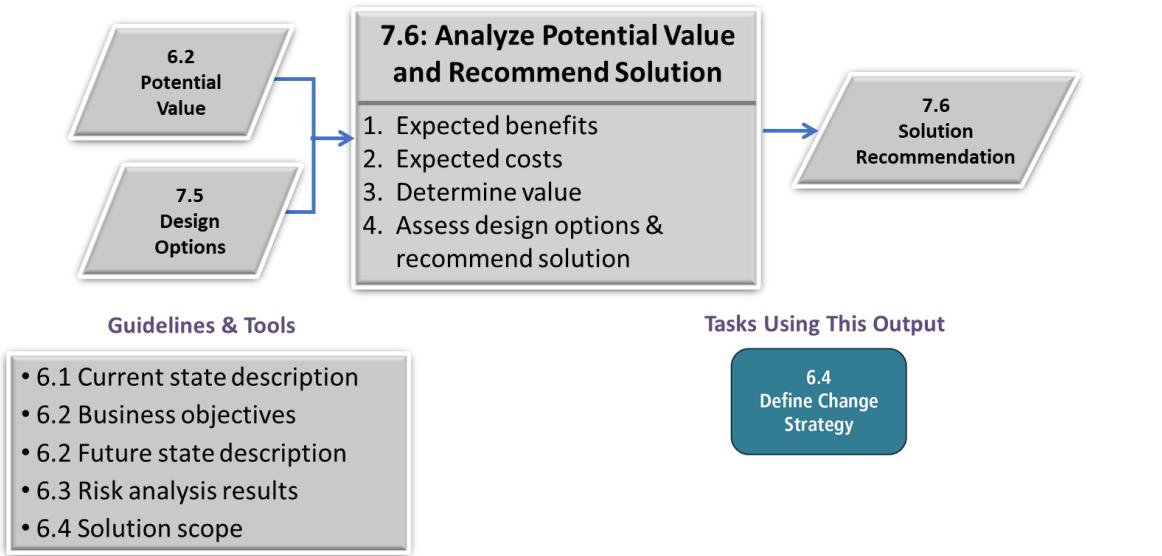
Analyze Potential Value and Recommend Solution describes how to estimate and model the potential value delivered by a set of requirements, designs, or design options. Potential value is analyzed many times over the course of a change. This analysis may be a planned event, or it may be triggered by a modification to the context or scope of the change. The analysis of potential value includes consideration that there is uncertainty in the estimates. Value can be described in terms of finance, reputation, or even impact on the marketplace. Any change may include a mix of increases and decreases in value.

Design options are evaluated by comparing the potential value of each option to the other options. Each option has a mix of advantages and disadvantages to consider. Depending on the reasons for the change, there may be no best option to recommend, or there may be a clear best choice. In some cases this means the best option may be to begin work against more than one design option, perhaps to develop proofs of concept, and then measure the performance of each. In other instances, all proposed designs may be rejected and more analysis may be needed to define a suitable design. It is also possible that the best recommendation is to do nothing.

7.6.3 INPUTS

- **Potential Value:** can be used as a benchmark against which the value delivered by a design can be evaluated.
- **Design Options:** need to be evaluated and compared to one another to recommend one option for the solution.

Figure 7.6.1: Analyze Potential Value and Recommend Solution Input/Output Diagram



7.6.4 ELEMENTS

.1 EXPECTED BENEFITS

Expected benefits describe the positive value that a solution is intended to deliver to stakeholders. Value can include benefits, reduced risk, compliance with business policies and regulations, an improved user experience, or any other positive outcome. Benefits are determined based on the analysis of the benefit that stakeholders desire and the benefit that is possible to attain. Expected benefits can be calculated at the level of a requirement or set of requirements by considering how much of an overall business objective the set of requirements contribute to if fulfilled. The total expected benefit is the net benefit of all the requirements a particular design option addresses. Benefits are often realized over a period of time.

.2 EXPECTED COSTS

Expected costs include any potential negative value associated with a solution, including the cost to acquire the solution, any negative effects it may have on stakeholders, and the cost to maintain it over time. Expected costs can include:

- timeline,
- effort,
- purchase and/or implementation costs,
- operating costs,
- maintenance costs,
- physical resources,
- information resources, and
- human resources.

Expected costs for a design option consider the cumulative costs of the design components.

Business analysts also consider opportunity cost when estimating the expected cost of a change. Opportunity costs are alternative results that might have been achieved if the resources, time, and funds devoted to one design option had been allocated to another design option. The opportunity cost of any design option is equal to the value of the best alternative not selected.

.3 DETERMINE VALUE

The potential value of a solution to a stakeholder is based on the benefits delivered by that solution and the associated costs. Value can be positive (if the benefits exceed the costs) or negative (if the costs exceed the benefits).

Business analysts consider potential value from the points of view of stakeholders. Value to the enterprise is almost always more heavily weighted than value for any individual stakeholder groups. There might be increases in value for one set of stakeholders and decreases in value for another set, but an overall positive increase in value for the enterprise as a whole justifies proceeding with the change.

Potential value is uncertain value. There are always events or conditions that could increase or decrease the actual value if they occur. Many changes are proposed in terms of intangible or uncertain benefits, while costs are described as tangible, absolute, and might grow. When benefits are described as intangible and costs expressed as tangible, it may be difficult for decision makers to compare their options. Business analysts define a complete estimate of the purpose-driven and monetary effects of a proposed change by considering the tangible and intangible costs alongside the tangible and intangible benefits. The estimate of costs and benefits must take into account the degree of uncertainty pertaining at the time the estimates are made.

.4 ASSESS DESIGN OPTIONS AND RECOMMEND SOLUTION

Each design option is assessed based on the potential value it is expected to deliver. At any point in analyzing the design options, it may become necessary to re-evaluate the initial allocation of design elements between components.

As costs and effort are understood for each solution component, business analysts assess each design option to ensure that it represents the most effective trade-offs. There are several factors to take into consideration:

- **Available Resources:** there may be limitations regarding the amount of requirements that can be implemented based on the allocated resources. In some instances, a business case can be developed to justify additional investment.
- **Constraints on the Solution:** regulatory requirements or business decisions may require that certain requirements be handled manually or automatically, or that certain requirements be prioritized above all others.
- **Dependencies between Requirements:** some capabilities may in and of themselves provide limited value to the organization, but need to be delivered in order to support other high-value requirements.

Other considerations may include relationships with proposed vendors, dependencies on other initiatives, corporate culture, and sufficient cash flow for investment.

Business analysts recommend the option or options deemed to be the most valuable solution to address the need. It is possible that none of the design options are worthwhile and the best recommendation is to do nothing.

7.6.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Objectives	Used to calculate the expected benefit.
Current State Description	Provides the context within which the work needs to be completed. It can be used to identify and help quantify the value to be delivered from a potential solution.
Future State Description	Describes the desired future state that the solution will be part of in order to ensure the design options are appropriate.
Risk Analysis Results	The potential value of design options includes an assessment of the level of risk associated with the design options or initiative.
Solution Scope	Defines the scope of the solution that is being delivered so that a relevant evaluation can be made that is within the scope boundaries.

7.6.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to express requirements in the form of acceptance criteria to make them most useful when assessing proposed solutions and determining whether a solution meets the defined business needs.
Backlog Management	Used to sequence the potential value.
Brainstorming	Used to identify potential benefits of the requirements in a collaborative manner.
Business Cases	Used to assess recommendations against business goals and objectives.
Business Model Canvas	Used as a tool to help understand strategy and initiatives.
Decision Analysis	Used to support the assessment and ranking of design options.
Estimation	Used to forecast the costs and efforts of meeting the requirements as a step towards estimating their value.
Financial Analysis	Used to evaluate the financial return of different options and choose the best possible return on investment.
Focus Groups	Used to get stakeholder input on which design options best meet the requirements, and to evaluate a targeted, small group of stakeholders' value expectations.
Interviews	Used to get stakeholder input on which design options best meet the requirements, and to evaluate individual stakeholders' value expectations.
Metrics and Key Performance Indicators (KPIs)	Used to create and evaluate the measurements used in defining value.
Risk Analysis and Management	Used to identify and manage the risks that could affect the potential value of the requirements.
Survey or Questionnaire	Used to get stakeholder input on which design options best

	meet the requirements, and to identify stakeholders' value expectations.
SWOT Analysis	Used to identify areas of strength and weakness that will impact the value of the solutions.
Workshops	Used to get stakeholder input on which design options best meet the requirements, and to evaluate stakeholders' value expectations.

7.6.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Represents the market segments affected by the requirements and solutions, and will be involved in analyzing the benefit of those requirements and costs of the design options.
Domain Subject Matter Expert	May be called upon for their domain knowledge to assist in analyzing potential value and benefits, particularly for those requirements where they are harder to identify.
End User	Provides an insight into the potential value of the change.
Implementation Subject Matter Expert	May be called upon for their expertise in implementing the design options in order to identify potential costs and risks.
Project Manager	Manages the selection process so that when effecting the change they are aware of potential impacts on those supporting the change, including the risks associated with the change.
Regulator	May be involved in risk evaluation concerning outside regulatory bodies or place constraints on the potential benefits.
Sponsor	Approves the expenditure of resources to purchase or develop a solution and approve the final recommendation. The sponsor will want to be kept informed of any changes in potential value or risk, as well as the resulting opportunity cost, as he/she may prefer another course of action.

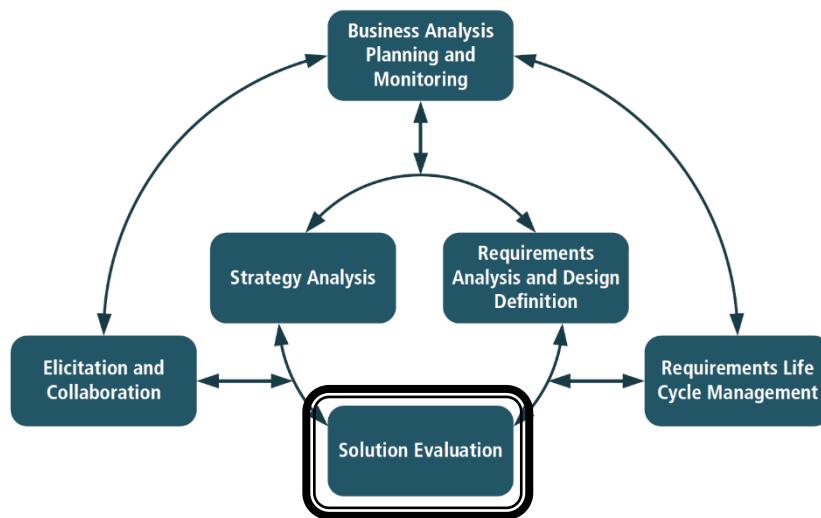
7.6.8 OUTPUTS

- **Solution Recommendation:** identifies the suggested, most appropriate solution based on an evaluation of all defined design options. The recommended solution should maximize the value provided to the enterprise.

8. Solution Evaluation

The Solution Evaluation knowledge area describes the tasks that business analysts perform to assess the performance of and value delivered by a solution in use by the enterprise, and to recommend removal of barriers or constraints that prevent the full realization of the value.

While there may be some similarities to the activities performed in **Strategy Analysis** or **Requirements Analysis and Design Definition**, an important distinction between the Solution Evaluation knowledge area and other knowledge areas is the existence of an actual solution. It may only be a partial solution, but the solution or solution component has already been implemented and is operating in some form. Solution Evaluation tasks that support the realization of benefits may occur before a change is initiated, while current value is assessed, or after a solution has been implemented.



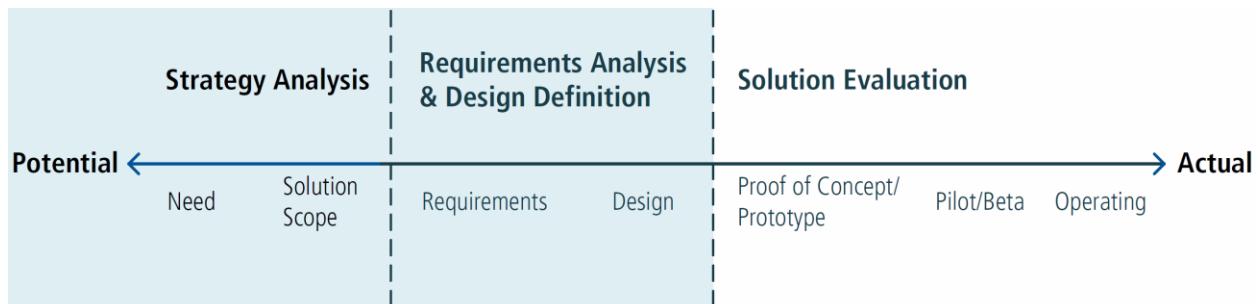
Solution Evaluation tasks can be performed on solution components in varying stages of development:

- **Prototypes or Proofs of Concept:** working but limited versions of a solution that demonstrate value.
- **Pilot or Beta releases:** limited implementations or versions of a solution used in order to work through problems and understand how well it actually delivers value before fully releasing the solution.
- **Operational releases:** full versions of a partial or completed solution used to achieve business objectives, execute a process, or fulfill a desired outcome.

Solution Evaluation describes tasks that analyze the actual value being delivered, identifies limitations which may be preventing value from being realized, and makes recommendations to increase the value of the solution. It may include any combination of performance assessments, tests, and experiments, and may combine both objective and subjective assessments of value. Solution Evaluation generally focuses on a component of an enterprise rather than the entire enterprise.

The following image illustrates the spectrum of value as business analysis activities progress from delivering potential value to actual value.

Figure 8.1: Business Analysis Value Spectrum



The Solution Evaluation knowledge area includes the following tasks:

MASER

1. **Measure Solution Performance**
2. **Analyze Performance Measures**
3. **Assess Solution Limitations**
4. **Assess Enterprise Limitations**
5. **Recommend Actions to Increase Solution Value**

The Core Concept Model in Solution Evaluation

The Business Analysis Core Concept Model™ (BACCM™) describes the relationships among the six core concepts. The following table describes the usage and application of each of the core concepts within the context of Solution Evaluation.

Table 8.1: The Core Concept Model in Solution Evaluation

Core Concept During Solution Evaluation, business analysts...	
Change	Recommend a change to either a solution or the enterprise in order to realize the potential value of a solution.
Need	Evaluate how a solution or solution component is fulfilling the need.
Solution	Assess the performance of the solution, examine if it is delivering the potential value, and analyze why value may not be realized by the solution or solution component.
Stakeholder	Elicit information from the stakeholders about solution performance and value delivery.
Value	Determine if the solution is delivering the potential value and examine why value may not be being realized.
Context	Consider the context in determining solution performance measures and any limitations within the context that may prohibit value from being realized.

8.1 MEASURE SOLUTION PERFORMANCE

8.1.1 PURPOSE

The purpose of Measure Solution Performance is to define performance measures and use the data collected to evaluate the effectiveness of a solution in relation to the value it brings.

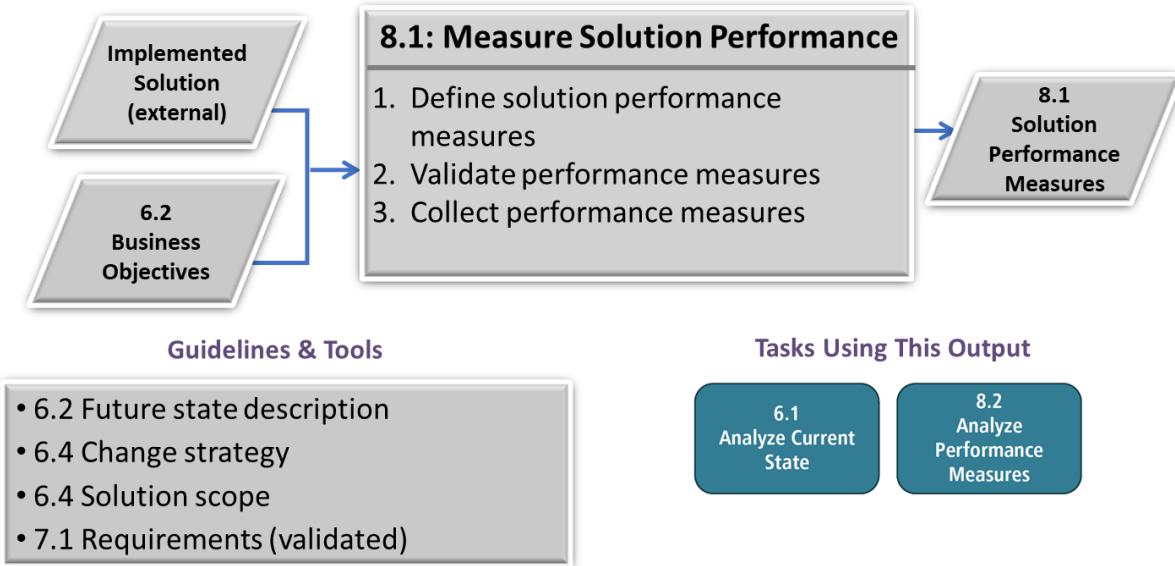
8.1.2 DESCRIPTION

Performance measures determine the value of a newly deployed or existing solution. The measures used depend on the solution itself, the context, and how the organization defines value. When solutions do not have built-in performance measures, the business analyst works with stakeholders to determine and collect the measures that will best reflect the performance of a solution. Performance may be assessed through key performance indicators (KPIs) aligned with enterprise measures, goals and objectives for a project, process performance targets, or tests for a software application.

8.1.3 INPUTS

- **Business Objectives:** the measurable results that the enterprise wants to achieve. Provides a benchmark against which solution performance can be assessed.
- **Implemented Solution (external):** a solution (or component of a solution) that exists in some form. It may be an operating solution, a prototype, or a pilot or beta solution.

Figure 8.1.1: Measure Solution Performance Input/Output Diagram



8.1.4 ELEMENTS

.1 DEFINE SOLUTION PERFORMANCE MEASURES

When measuring solution performance, business analysts determine if current measures exist, or if methods for capturing them are in place. Business analysts ensure that any existing performance measures are accurate, relevant and elicit any additional performance measures identified by stakeholders.

Business goals, objectives, and business processes are common sources of measures. Performance measures may be influenced or imposed by third parties such as solution vendors, government bodies, or other regulatory organizations. The type and nature of the measurements are considered when choosing the elicitation method. Solution performance measures may be quantitative, qualitative, or both, depending on the value being measured.

- **Quantitative Measures:** are numerical, countable, or finite, usually involving amounts, quantities, or rates.
- **Qualitative Measures:** are subjective and can include attitudes, perceptions, and any other subjective response. Customers, users, and others involved in the operation of a solution have perceptions of how well the solution is meeting the need.

.2 VALIDATE PERFORMANCE MEASURES

Validating performance measures helps to ensure that the assessment of solution performance is useful. Business analysts validate the performance measures and any influencing criteria with stakeholders. Specific performance measures should align with any higher-level measures that exist within the context affecting the solution. Decisions about which measures are used to evaluate

solution performance often reside with the sponsor, but may be made by any stakeholder with decision-making authority.

.3 COLLECT PERFORMANCE MEASURES

When defining performance measures, business analysts may employ basic statistical sampling concepts. When collecting performance measures, business analysts consider:

- **Volume or Sample Size:** a volume or sample size appropriate for the initiative is selected. A sample size that is too small might skew the results and lead to inaccurate conclusions. Larger sample sizes may be more desirable, but may not be practical to obtain.
- **Frequency and Timing:** the frequency and timing with which measurements are taken may have an effect on the outcome.
- **Currency:** measurements taken more recently tend to be more representative than older data.

Using qualitative measures, business analysts can facilitate discussions to estimate the value realized by a solution. Stakeholders knowledgeable about the operation and use of the solution reach a consensus based on facts and reasonable assumptions, as perceived by them.

8.1.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Change Strategy	The change strategy used or in use to implement the potential value.
Future State Description	Boundaries of the proposed new, removed, or modified components of the enterprise, and the potential value expected from the future state.
Requirements (validated)	A set of requirements that have been analyzed and appraised to determine their value.
Solution Scope	The solution boundaries to measure and evaluate.

8.1.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to define acceptable solution performance.
Benchmarking and Market Analysis	Used to define measures and their acceptable levels.
Business Cases	Used to define business objectives and performance measures for a proposed solution.
Data Mining	Used to collect and analyze large amounts of data regarding solution performance.
Decision Analysis	Used to assist stakeholders in deciding on suitable ways to measure solution performance and acceptable levels of performance.
Focus Groups	Used to provide subjective assessments, insights, and

	impressions of a solution's performance.
Metrics and Key Performance Indicators (KPIs)	Used to measure solution performance.
Non-functional Requirements Analysis	Used to define expected characteristics of a solution.
Observation	Used either to provide feedback on perceptions of solution performance or to reconcile contradictory results.
Prototyping	Used to simulate a new solution so that performance measures can be determined and collected.
Survey or Questionnaire	Used to gather opinions and attitudes about solution performance. Surveys and questionnaires can be effective when large or disparate groups need to be polled.
Use Cases and Scenarios	Used to define the expected outcomes of a solution.
Vendor Assessment	Used to assess which of the vendor's performance measures should be included in the solution's performance assessment.

8.1.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	May be consulted to provide feedback on solution performance.
Domain Subject Matter Expert	A person familiar with the domain who can be consulted to provide potential measurements.
End User	Contributes to the actual value realized by the solution in terms of solution performance. They may be consulted to provide reviews and feedback on areas such as workload and job satisfaction.
Project Manager	Responsible for managing the schedule and tasks to perform the solution measurement. For solutions already in operation, this role may not be required.
Sponsor	Responsible for approving the measures used to determine solution performance. May also provide performance expectations.
Regulator	An external or internal group that may dictate or prescribe constraints and guidelines that must be incorporated into solution performance measures.

8.1.8 OUTPUTS

- **Solution Performance Measures:** measures that provide information on how well the solution is performing or potentially could perform.

8.2 ANALYZE PERFORMANCE MEASURES

8.2.1 PURPOSE

The purpose of Analyze Performance Measures is to provide insights into the performance of a solution in relation to the value it brings.

8.2.2 DESCRIPTION

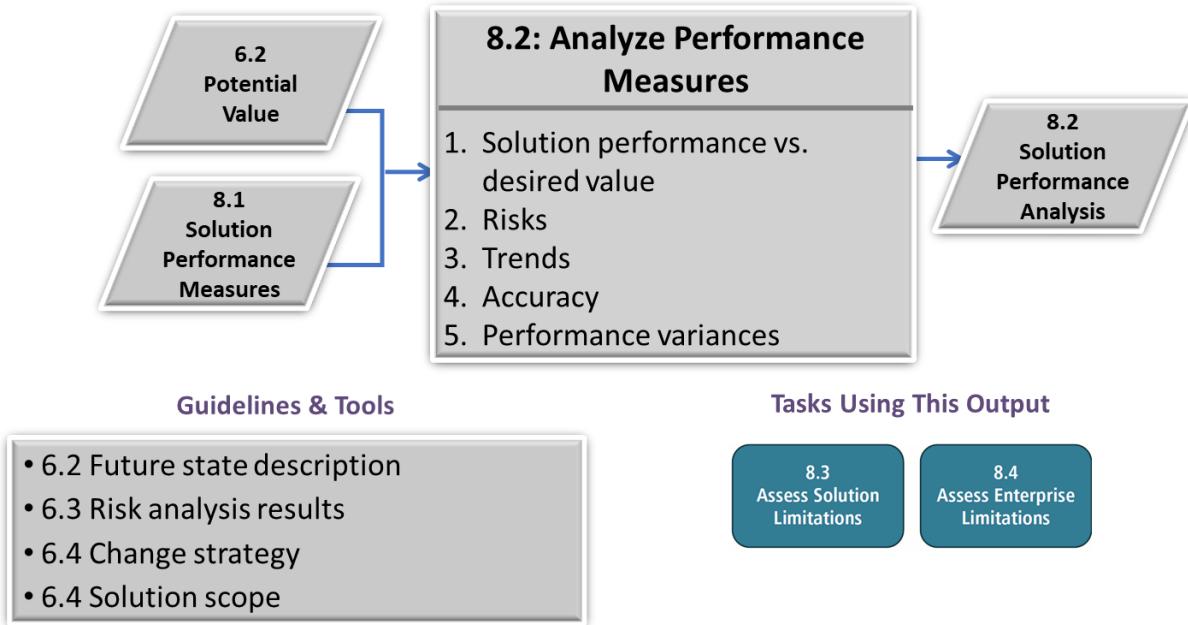
The measures collected in the task Measure Solution Performance often require interpretation and synthesis to derive meaning and to be actionable. Performance measures themselves rarely trigger a decision about the value of a solution.

In order to meaningfully analyze performance measures, business analysts require a thorough understanding of the potential value that stakeholders hope to achieve with the solution. To assist in the analysis, variables such as the goals and objectives of the enterprise, key performance indicators (KPIs), the level of risk of the solution, the risk tolerance of both stakeholders and the enterprise, and other stated targets are considered.

8.2.3 INPUTS

- **Potential Value:** describes the value that may be realized by implementing the proposed future state. It can be used as a benchmark against which solution performance can be evaluated.
- **Solution Performance Measures:** measures and provides information on how well the solution is performing or potentially could perform.

Figure 8.2.1: Analyze Performance Measures Input/Output Diagram



8.2.4 ELEMENTS

.1 SOLUTION PERFORMANCE VERSUS DESIRED VALUE

Business analysts examine the measures previously collected in order to assess their ability to help stakeholders understand the solution's value. A solution might be high performing, such as an efficient online transaction processing system, but contributes lower value than expected (or compared to what it had contributed in the past). On the other hand, a low performing but potentially valuable solution, such as a core process that is inefficient, can be enhanced to increase its performance level. If the measures are not sufficient to help stakeholders determine solution value, business analysts either collect more measurements or treat the lack of measures as a solution risk.

.2 RISKS

Performance measures may uncover new risks to solution performance and to the enterprise. These risks are identified and managed like any other risks.

.3 TRENDS

When analyzing performance data, business analysts consider the time period when the data was collected to guard against anomalies and skewed trends. A large enough sample size over a sufficient time period will provide an accurate depiction of solution performance on which to make decisions and guard against false signals brought about by incomplete data. Any pronounced and repeated trends, such as a noticeable increase in errors at certain times or a change in process speed when volume is increased, are noted.

.4 ACCURACY

The accuracy of performance measures is essential to the validity of their analysis. Business analysts test and analyze the data collected by the performance measures to ensure their accuracy. To be considered accurate and reliable, the results of performance measures should be reproducible and repeatable.

.5 PERFORMANCE VARIANCES

The difference between expected and actual performance represents a variance that is considered when analyzing solution performance. Root cause analysis may be necessary to determine the underlying causes of significant variances within a solution. Recommendations of how to improve performance and reduce any variances are made in the task Recommend Actions to Increase Solution Value.

8.2.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Change Strategy	The change strategy that was used or is in use to implement the potential value.
Future State Description	Boundaries of the proposed new, modified, or removed components of the enterprise and the potential value expected from the future state.
Risk Analysis Results	The overall level of risk and the planned approach to modifying the individual risks.
Solution Scope	The solution boundaries to measure and evaluate.

8.2.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used to define acceptable solution performance through acceptance criteria. The degree of variance from these criteria will guide the analysis of that performance.
Benchmarking and Market Analysis	Used to observe the results of other organizations employing similar solutions when assessing risks, trends, and variances.
Data Mining	Used to collect data regarding performance, trends, common issues, and variances from expected performance levels and understand patterns and meaning in that data.
Interviews	Used to determine expected value of a solution and its perceived performance from an individual or small group's perspective.
Metrics and Key Performance Indicators (KPIs)	Used to analyze solution performance, especially when judging how well a solution contributes to achieving goals.
Observation	Used to observe a solution in action if the data collected does not provide definitive conclusions.
Risk Analysis and Management	Used to identify, analyze, develop plans to modify the risks, and to manage the risks on an ongoing basis.
Root Cause Analysis	Used to determine the underlying cause of performance variance.

Survey or Questionnaire	Used to determine expected value of a solution and its perceived performance.
--------------------------------	---

8.2.7 STAKEHOLDERS

Stakeholder	Contribution
Domain Subject Matter Expert	Can identify risks and provide insights into data for analyzing solution performance.
Project Manager	Within a project, responsible for overall risk management and may participate in risk analysis for new or changed solutions.
Sponsor	Can identify risks, provide insights into data and the potential value of a solution. They will make decisions about the significance of expected versus actual solution performance.

8.2.8 OUTPUTS

- **Solution Performance Analysis:** results of the analysis of measurements collected and recommendations to solve performance gaps and leverage opportunities to improve value.

8.3 ASSESS SOLUTION LIMITATIONS

8.3.1 PURPOSE

The purpose of Assess Solution Limitations is to determine the factors internal to the solution that restrict the full realization of value.

8.3.2 DESCRIPTION

Assessing solution limitations identifies the root causes for under-performing and ineffective solutions and solution components.

Assess Solution Limitations is closely linked to the task Assess Enterprise Limitations. These tasks may be performed concurrently. If the solution has not met its potential value, business analysts determine which factors, both internal and external to the solution, are limiting value. This task focuses on the assessment of those factors internal to the solution.

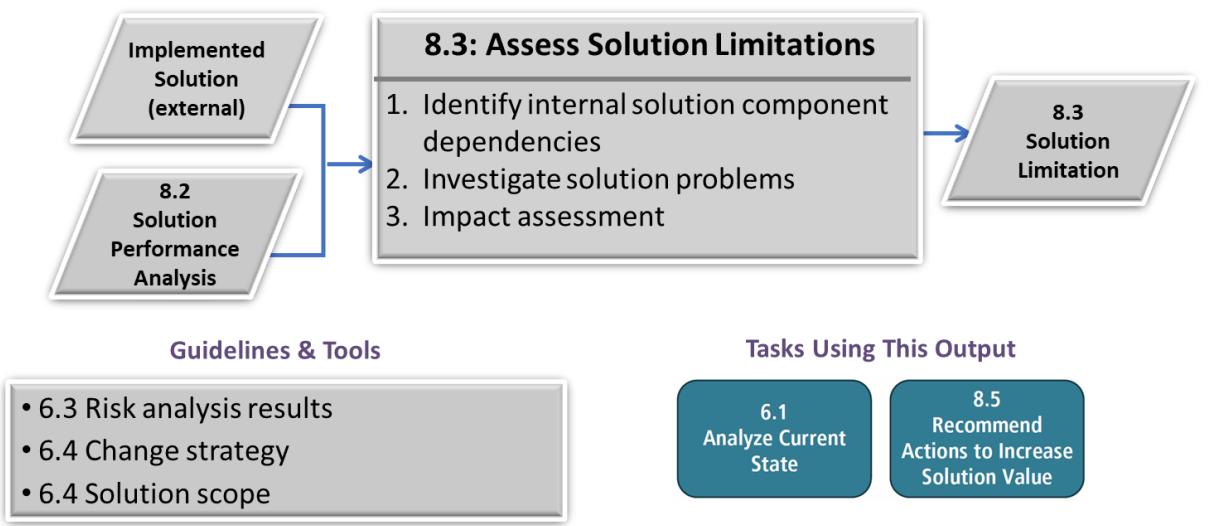
This assessment may be performed at any point during the solution life cycle. It may occur on a solution component during its development, on a completed solution prior to full implementation, or on an existing solution that is currently working within an organization. Regardless of the timing, the assessment activities are similar and involve the same considerations.

8.3.3 INPUTS

- **Implemented Solution (external):** a solution that exists. The solution may or may not be in operational use; it may be a prototype. The solution must be in use in some form in order to be evaluated.

- **Solution Performance Analysis:** results of the analysis of measurements collected and recommendations to solve for performance gaps and leverage opportunities to improve value.

Figure 8.3.1: Assess Solution Limitations Input/Output Diagram



8.3.4 ELEMENTS

.1 IDENTIFY INTERNAL SOLUTION COMPONENT DEPENDENCIES

Solutions often have internal dependencies that limit the performance of the entire solution to the performance of the least effective component. Assessment of the overall performance of the solution or its components is performed in the tasks **Measure Solution Performance** and **Analyze Performance Measures**. Business analysts identify solution components which have dependencies on other solution components, and then determine if there is anything about those dependencies or other components that limit solution performance and value realization.

.2 INVESTIGATE SOLUTION PROBLEMS

When it is determined that the solution is consistently or repeatedly producing ineffective outputs, problem analysis is performed in order to identify the source of the problem.

Business analysts identify problems in a solution or solution component by examining instances where the outputs from the solution are below an acceptable level of quality or where the potential value is not being realized. Problems may be indicated by an inability to meet a stated goal, objective, or requirement, or may be a failure to realize a benefit that was projected during the tasks **Define Change Strategy** or **Recommend Actions to Increase Solution Value**.

.3 IMPACT ASSESSMENT

Business analysts review identified problems in order to assess the effect they may have on the operation of the organization or the ability of the solution to deliver its potential value. This requires determining the severity of the problem, the probability of the re-occurrence of the problem, the impact on the business operations, and the capacity of the business to absorb the impact. Business

analysts identify which problems must be resolved, which can be mitigated through other activities or approaches, and which can be accepted.

Other activities or approaches may include additional quality control measures, new or adjusted business processes, or additional support for exceptions to the desired outcome.

In addition to identified problems, business analysts assess risks to the solution and potential limitations of the solution. This risk assessment is specific to the solution and its limitations.

8.3.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Change Strategy	The change strategy used or in use to implement the potential value.
Risk Analysis Results	The overall level of risk and the planned approach to modifying the individual risks.
Solution Scope	The solution boundaries to measure and evaluate.

8.3.6 TECHNIQUES

Techniques	Usage
Acceptance and Evaluation Criteria	Used both to indicate the level at which acceptance criteria are met or anticipated to be met by the solution and to identify any criteria that are not met by the solution.
Benchmarking and Market Analysis	Used to assess if other organizations are experiencing the same solution challenges and, if possible, determine how they are addressing it.
Business Rules Analysis	Used to illustrate the current business rules and the changes required to achieve the potential value of the change.
Data Mining	Used to identify factors constraining performance of the solution.
Decision Analysis	Used to illustrate the current business decisions and the changes required to achieve the potential value of the change.
Interviews	Used to help perform problem analysis.
Item Tracking	Used to record and manage stakeholder issues related to why the solution is not meeting the potential value.
Lessons Learned	Used to determine what can be learned from the inception, definition, and construction of the solution to have potentially impacted its ability to deliver value.
Risk Analysis and Management	Used to identify, analyze, and manage risks, as they relate to the solution and its potential limitations, that may impede the realization of potential value.
Root Cause Analysis	Used to identify and understand the combination of factors and their underlying causes that led to the solution being unable to deliver its potential value.
Survey or Questionnaire	Used to help perform problem analysis.

8.3.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	Is ultimately affected by a solution, and therefore has an important perspective on its value. A customer may be consulted to provide reviews and feedback.
Domain Subject Matter Expert	Provides input into how the solution should perform and identifies potential limitations to value realization.
End User	Uses the solution, or is a component of the solution, and therefore contributes to the actual value realized by the solution in terms of solution performance. An end user may be consulted to provide reviews and feedback on areas such as workload and job satisfaction.
Regulator	A person whose organization needs to be consulted about the planned and potential value of a solution, as that organization may constrain the solution, the degree to which actual value is realized, or when actual value is realized.
Sponsor	Responsible for approving the potential value of the solution, for providing resources to develop, implement and support the solution, and for directing enterprise resources to use the solution. The sponsor is also responsible for approving a change to potential value.
Tester	Responsible for identifying solution problems during construction and implementation; not often used in assessing an existing solution outside of a change.

8.3.8 OUTPUTS

- **Solution Limitation:** a description of the current limitations of the solution including constraints and defects.

8.4 ASSESS ENTERPRISE LIMITATIONS

8.4.1 PURPOSE

The purpose of Assess Enterprise Limitations is to determine how factors external to the solution are restricting value realization.

8.4.2 DESCRIPTION

Solutions may operate across various organizations within an enterprise, and therefore have many interactions and interdependencies. Solutions may also depend on environmental factors that are external to the enterprise. Enterprise limitations may include factors such as culture, operations, technical components, stakeholder interests, or reporting structures.

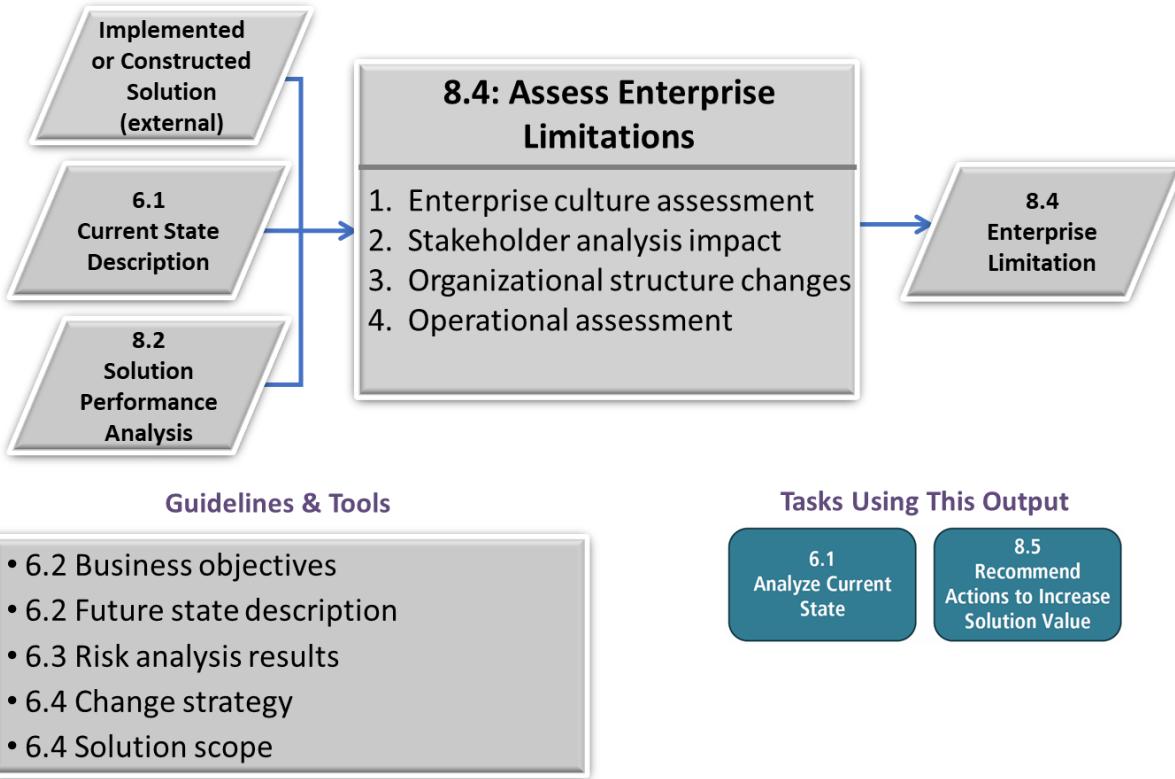
Assessing enterprise limitations identifies root causes and describes how enterprise factors limit value realization.

This assessment may be performed at any point during the solution life cycle. It may occur on a solution component during its development or on a completed solution prior to full implementation. It may also occur on an existing solution that is currently working within an organization. Regardless of the timing, the assessment activities are similar and require the same skills.

8.4.3 INPUTS

- **Current State Description:** the current internal environment of the solution including the environmental, cultural, and internal factors influencing the solution limitations.
- **Implemented (or Constructed) Solution (external):** a solution that exists. The solution may or may not be in operational use; it may be a prototype. The solution must be in use in some form in order to be evaluated.
- **Solution Performance Analysis:** results of the analysis of measurements collected and recommendations to solve performance gaps and leverage opportunities to improve value.

Figure 8.4.1: Assess Enterprise Limitations Input/Output Diagram



8.4.4 ELEMENTS

.1 ENTERPRISE CULTURE ASSESSMENT

Enterprise culture is defined as the deeply rooted beliefs, values, and norms shared by the members of an enterprise. While these beliefs and values may not be directly visible, they drive the actions taken by an enterprise.

Business analysts perform cultural assessments to:

- identify whether or not stakeholders understand the reasons why a solution exists,
- ascertain whether or not the stakeholders view the solution as something beneficial and are supportive of the change, and
- determine if and what cultural changes are required to better realize value from a solution.

The enterprise culture assessment evaluates the extent to which the culture can accept a solution. If cultural adjustments are needed to support the solution, the assessment is used to judge the enterprise's ability and willingness to adapt to these cultural changes.

Business analysts also evaluate internal and external stakeholders to:

- gauge understanding and acceptance of the solution,
- assess perception of value and benefit from the solution, and
- determine what communication activities are needed to ensure awareness and understanding of the solution.

.2 STAKEHOLDER IMPACT ANALYSIS

A stakeholder impact analysis provides insight into how the solution affects a particular stakeholder group. When conducting stakeholder impact analysis, business analysts consider:

- **Functions:** the processes in which the stakeholder uses the solution, which include inputs a stakeholder provides into the process, how the stakeholder uses the solution to execute the process, and what outputs the stakeholder receives from the process.
- **Locations:** the geographic locations of the stakeholders interacting with the solution. If the stakeholders are in disparate locations, it may impact their use of the solution and the ability to realize the value of the solution.
- **Concerns:** the issues, risks, and overall concerns the stakeholders have with the solution. This may include the use of the solution, the perceptions of the value of the solution, and the impact the solution has on a stakeholder's ability to perform necessary functions.

.3 ORGANIZATIONAL STRUCTURE CHANGES

There are occasions when business analysts assess how the organization's structure is impacted by a solution.

The use of a solution and the ability to adopt a change can be enabled or blocked by formal and informal relationships among stakeholders. The reporting structure may be too complex or too simple to allow a solution to perform effectively. Assessing if the organizational hierarchy supports the solution is a key activity. On occasion, informal relationships within an organization, whether alliances, friendships, or matrix-reporting, impact the ability of a solution to deliver potential value. Business analysts consider these informal relationships in addition to the formal structure.

.4 OPERATIONAL ASSESSMENT

The operational assessment is performed to determine if an enterprise is able to adapt to or effectively use a solution. This identifies which processes and tools within the enterprise are adequately equipped to benefit from the solution, and if sufficient and appropriate assets are in place to support it.

When conducting an operational assessment, business analysts consider:

- policies and procedures,
- capabilities and processes that enable other capabilities,
- skill and training needs,
- human resources practices,
- risk tolerance and management approaches, and
- tools and technology that support a solution.

8.4.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
--------------------	-------------

Business Objectives	Are considered when measuring and determining solution performance.
Change Strategy	The change strategy used or in use to implement the potential value.
Future State Descriptions	Boundaries of the proposed new, removed, or modified components of the enterprise, as well as the potential value expected from the future state.
Risk Analysis Results	The overall level of risk and the planned approach to modifying the individual risks.
Solution Scope	The solution boundaries to measure and evaluate.

8.4.6 TECHNIQUES

Techniques	Usage
Benchmarking and Market Analysis	Used to identify existing solutions and enterprise interactions.
Brainstorming	Used to identify organizational gaps or stakeholder concerns.
Data Mining	Used to identify factors constraining performance of the solution.
Decision Analysis	Used to assist in making an optimal decision under conditions of uncertainty and may be used in the assessment to make decisions about functional, technical, or procedural gaps.
Document Analysis	Used to gain an understanding of the culture, operations, and structure of the organization.
Interviews	Used to identify organizational gaps or stakeholder concerns.
Item Tracking	Used to ensure that issues are not neglected or lost and that issues identified by assessment are resolved.
Lessons Learned	Used to analyze previous initiatives and the enterprise interactions with the solutions.
Observation	Used to witness the enterprise and solution interactions to identify impacts.
Organizational Modelling	Used to ensure the identification of any required changes to the organizational structure that may have to be addressed.
Process Analysis	Used to identify possible opportunities to improve performance.
Process Modelling	Used to illustrate the current business processes and/or changes that must be made in order to achieve the potential value of the solution.
Risk Analysis and Management	Used to consider risk in the areas of technology (if the selected technological resources provide required functionality), finance (if costs could exceed levels that make the change salvageable), and business (if the organization will be able to make the changes necessary to attain potential value from the solution).
Roles and Permissions Matrix	Used to determine roles and associated permissions for stakeholders, as well as stability of end users.

Root Cause Analysis	Used to determine if the underlying cause may be related to enterprise limitations.
Survey or Questionnaire	Used to identify organizational gaps or stakeholder concerns.
SWOT Analysis	Used to demonstrate how a change will help the organization maximize strengths and minimize weaknesses, and to assess strategies developed to respond to identified issues.
Workshops	Used to identify organizational gaps or stakeholder concerns.

8.4.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	People directly purchasing or consuming the solution who may interact with the organization in the use of the solution.
Domain Subject Matter Expert	Provides input into how the organization interacts with the solution and identifies potential limitations.
End User	People who use a solution or who are a component of the solution. Users could be customers or people who work within the organization.
Regulator	One or many governmental or professional entities that ensure adherence to laws, regulations, or rules; may have unique input to the organizational assessment, as relevant regulations must be included in the requirements. There may be laws and regulations that must be complied with prior to (or as a result of) a planned or implemented change.
Sponsor	Authorizes and ensures funding for a solution delivery, and champions action to resolve problems identified in the organizational assessment.

8.4.8 OUTPUTS

- **Enterprise Limitation:** a description of the current limitations of the enterprise including how the solution performance is impacting the enterprise.

8.5 RECOMMEND ACTIONS TO INCREASE SOLUTION VALUE

8.5.1 PURPOSE

The purpose of Recommend Actions to Increase Solution Value is to understand the factors that create differences between potential value and actual value, and to recommend a course of action to align them.

8.5.2 DESCRIPTION

The various tasks in the Solution Evaluation knowledge area help to measure, analyze, and determine causes of unacceptable solution performance. The task Recommend Actions to Increase Solution Value focuses on understanding the aggregate of the performed assessments

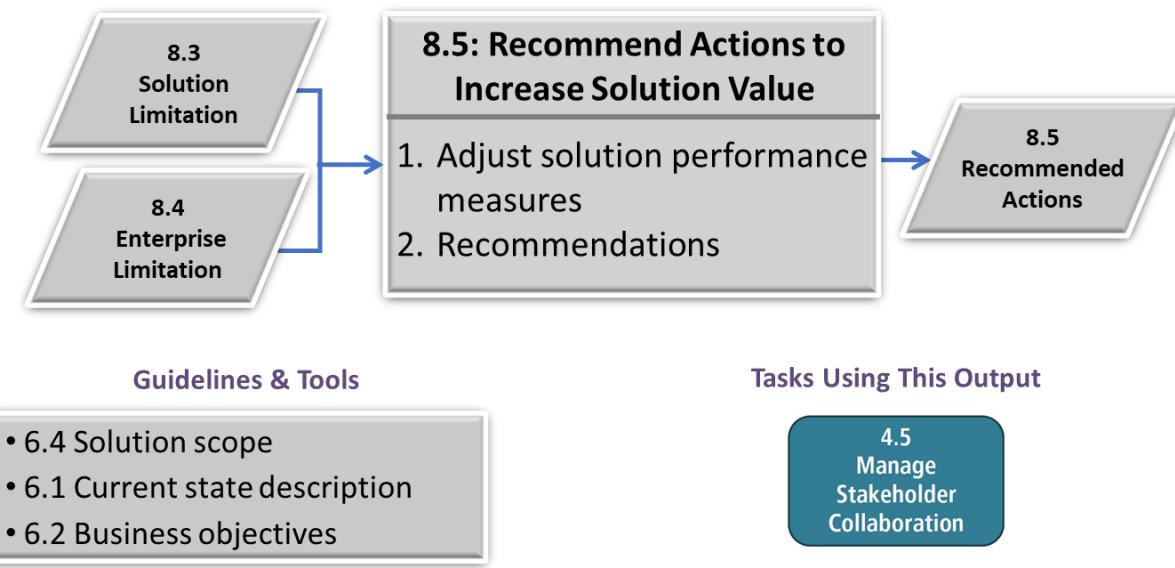
and identifying alternatives and actions to improve solution performance and increase value realization.

Recommendations generally identify how a solution should be replaced, retired, or enhanced. They may also consider long-term effects and contributions of the solution to stakeholders. They may include recommendations to adjust the organization to allow for maximum solution performance and value realization.

8.5.3 INPUTS

- **Enterprise Limitation:** a description of the current limitations of the enterprise including how the solution performance is impacting the enterprise.
- **Solution Limitation:** a description of the current limitations of the solution including constraints and defects.

Figure 8.5.1: Recommend Actions to Increase Solution Value Input/Output Diagram



8.5.4 ELEMENTS

.1 ADJUST SOLUTION PERFORMANCE MEASURES

In some cases, the performance of the solution is considered acceptable but may not support the fulfillment of business goals and objectives. An analysis effort to identify and define more appropriate measures may be required.

.2 RECOMMENDATIONS

While recommendations often describe ways to increase solution performance, this is not always the case. Depending on the reason for lower than expected performance, it may be reasonable to take no action, adjust factors that are external to the solution, or reset expectations for the solution.

Some common examples of recommendations that a business analyst may make include:

- **Do Nothing:** is usually recommended when the value of a change is low relative to the effort required to make the change, or when the risks of change significantly outweigh the risks of remaining in the current state. It may also be impossible to make a change with the resources available or in the allotted time frame.
- **Organizational Change:** is a process for managing attitudes about, perceptions of, and participation in the change related to the solution. Organizational change management generally refers to a process and set of tools for managing change at an organizational level. The business analyst may help to develop recommendations for changes to the organizational structure or personnel, as job functions may change significantly as the result of work being automated. New information may be made available to stakeholders and new skills may be required to operate the solution.

Possible recommendations that relate to organizational change include:

- Automating or simplifying the work people perform. Relatively simple tasks are prime candidates for automation. Additionally, work activities and business rules can be reviewed and analyzed to determine opportunities for re-engineering, changes in responsibilities, and outsourcing.
- Improving access to information. Change may provide greater amounts of information and better quality of information to staff and decision makers.
- **Reduce Complexity of Interfaces:** interfaces are needed whenever work is transferred between systems or between people. Reducing their complexity can improve understanding.
- **Eliminate Redundancy:** different stakeholder groups may have common needs that can be met with a single solution, reducing the cost of implementation.
- **Avoid Waste:** the aim of avoiding waste is to completely remove those activities that do not add value and minimize those activities that do not contribute to the final product directly.
- **Identify Additional Capabilities:** solution options may offer capabilities to the organization above and beyond those identified in the requirements. In many cases, these capabilities are not of immediate value to the organization but have the potential to provide future value, as the solution may support the rapid development or implementation of those capabilities if they are required (for example, a software application may have features that the organization anticipates using in the future).
- **Retire the Solution:** it may be necessary to consider the replacement of a solution or solution component. This may occur because technology has reached the end of its life, services are being insourced or outsourced, or the solution is not fulfilling the goals for which it was created.

Some additional factors that may impact the decision regarding the replacement or retirement of a solution include:

- **ongoing cost versus initial investment:** it is common for the existing solution to have increasing costs over time, while alternatives have a higher investment cost upfront but lower maintenance costs.
- **opportunity cost:** represents the potential value that could be realized by pursuing alternative courses of action.
- **necessity:** most solution components have a limited lifespan (due to obsolescence, changing market conditions, and other causes). After a certain point in the life cycle it will become impractical or impossible to maintain the existing component.
- **sunk cost:** describes the money and effort already committed to an initiative. The psychological impact of sunk costs may make it difficult for stakeholders to objectively assess the rationale for replacement or elimination, as they may feel reluctant to "waste" the effort or money already invested. As this investment cannot be recovered, it is effectively irrelevant when considering future action. Decisions should be based on the future investment required and the future benefits that can be gained.

8.5.5 GUIDELINES AND TOOLS

Guidelines & Tools	Description
Business Objectives	Are considered in evaluating, measuring, and determining solution performance.
Current State Description	Provides the context within which the work needs to be completed. It can be used to assess alternatives and better understand the potential increased value that could be delivered. It can also help highlight unintended consequences of alternatives that may otherwise remain undetected.
Solution Scope	The solution boundaries to measure and evaluate.

8.5.6 TECHNIQUES

Techniques	Usage
Data Mining	Used to generate predictive estimates of solution performance.
Decision Analysis	Used to determine the impact of acting on any of the potential value or performance issues.
Financial Analysis	Used to assess the potential costs and benefits of a change.
Focus Groups	Used to determine if solution performance measures need to be adjusted and used to identify potential opportunities to improve performance.
Organizational Modelling	Used to demonstrate potential change within the organization's structure.
Prioritization	Used to identify relative value of different actions to improve solution performance.
Process Analysis	Used to identify opportunities within related processes.
Risk Analysis and Management	Used to evaluate different outcomes under specific conditions.
Survey or Questionnaire	Used to gather feedback from a wide variety of stakeholders to determine if value has been met or exceeded, if the metrics are still valid or relevant in the current context, and what actions might be taken to improve the solution.

8.5.7 STAKEHOLDERS

Stakeholder	Contribution
Customer	People directly purchasing or consuming the solution and who may interact with the organization in the use of the solution.
Domain Subject Matter Expert	Provides input into how to change the solution and/or the organization in order to increase value.
End User	People who use a solution or who are a component of the solution. Users could be customers or people who work within the organization.
Regulator	One or many governmental or professional entities that ensure adherence to laws, regulations, or rules. Relevant regulations must be included in requirements.

Sponsor	Authorizes and ensures funding for implementation of any recommended actions.
----------------	---

8.5.8 OUTPUTS

- **Recommended Actions:** recommendation of what should be done to improve the value of the solution within the enterprise.

9. Underlying Competencies

The Underlying Competencies chapter provides a description of the behaviours, characteristics, knowledge, and personal qualities that support the practice of business analysis.

The underlying competencies described here are not unique to business analysis. They are described here to ensure readers are aware of the range of fundamental skills required and provide a basis for them to further investigate the skills and knowledge that will enable them to be accomplished and adaptable business analysts.

These competencies are grouped into six categories:

1. **Analytical Thinking and Problem Solving,**
2. **Behavioural Characteristics,**
3. **Business Knowledge,**
4. **Communication Skills,**
5. **Interaction Skills, and**
6. **Tools and Technology.**

Each underlying competency is defined with a purpose, definition, and effectiveness measures.

9.1 ANALYTICAL THINKING AND PROBLEM SOLVING

Analytical thinking and problem solving skills are required for business analysts to analyze problems and opportunities effectively, identify which changes may deliver the most value, and work with stakeholders to understand the impact of those changes.

Business analysts use analytical thinking by rapidly assimilating various types of information (for example, diagrams, stakeholder concerns, customer feedback, schematics, user guides, and spreadsheets), and identifying which are relevant. Business analysts should be able to quickly choose effective and adaptable methods to learn and analyze the media, audiences, problem types, and environments as each is encountered.

Business analysts utilize analytical thinking and problem solving as they facilitate understanding of situations, the value of proposed changes, and other complex ideas.

Possessing a sound understanding of the analytical thinking and problem solving core competencies allows business analysts to identify the best ways to present information to their stakeholders. For example, some concepts are more easily understood when presented in diagrams and information graphics rather than by paragraphs of text. Having this understanding assists business analysts when planning their business analysis approach and enables them to communicate business analysis information in a manner that suits the material being conveyed to their audience.

Analytical Thinking and Problem Solving core competencies include:

- Creative Thinking,

- Decision Making,
- Learning,
- Problem Solving,
- Systems Thinking,
- Conceptual Thinking, and
- Visual Thinking

9.1.1. CREATIVE THINKING

PURPOSE	Effectively generate new ideas, approaches and alternatives to solve problems and generate opportunities.
DEFINITION	<ul style="list-style-type: none"> • Involves combining, changing and reapplying existing concepts or ideas to overcome conventional approaches that limit options. • Promote creative thinking in others through asking questions, challenging assumptions and proposing alternatives.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Generating and productively considering new ideas and concepts. • Applying new ideas to resolve existing problems.

9.1.2. DECISION MAKING

PURPOSE	To effectively understand the criteria involved in making a decision and assist others to make better decisions.
DEFINITION	<ul style="list-style-type: none"> • Enable stakeholders to select an option from a set of alternatives by making an informed decision. • The BA gathers, analyses and interprets the relevant information, makes comparisons and outlines risks & benefits of each option.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Stakeholders understand the decision making process. • Stakeholders understand all the conditions, environment and measures in which the decision will be made. • All pros and cons are clearly communicated. • The decision reduces or eliminates uncertainty.

9.1.3. LEARNING

PURPOSE	Quickly absorb new and different types of information or modify / adapt existing knowledge to the current situation.
----------------	---

DEFINITION	<ul style="list-style-type: none"> • Process of gaining skills and knowledge. • BA's must adopt the best learning techniques to acquire, comprehend, apply, synthesize and evaluate the information to identify opportunities, evaluate or create new solutions.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Learn concepts presented then demonstrate an understanding. • Effectively presenting new facts, ideas, concepts and opinions to others. • Demonstrate ability to apply concepts to new areas or relationships.

9.1.4. PROBLEM SOLVING

PURPOSE	Define and solve problems to ensure the real, underlying root cause of a problem is understood and agreed, and the solution options address that problem.
DEFINITION	<ul style="list-style-type: none"> • Problem solving requires stakeholder points of view to be addressed, conflicts between goals and objectives resolved and assumptions identified and validated. • Requires alternatives to be measured against agreed objectives to identify the value and trade-offs.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Confidence of the participants in the problem solving process. • Selected solutions meet the defined objectives and solve the root cause of the problem.

9.1.5. SYSTEMS THINKING

PURPOSE	Understanding how the people, processes and technology within an organisation interact to gain a holistic point of view.
DEFINITION	Systems thinking is understanding the whole system (people, processes, technology, environment and market), the interaction of the component parts within the system and their affect on each other.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Communicate how changes to a component affect the system as a whole. • How changes to a system affects the environment it is in. • How systems adapt to internal/external changes.

9.1.6. CONCEPTUAL THINKING

PURPOSE	Ability to understand how detailed and disparate information fits
----------------	--

	into a larger picture, what details are relevant or misleading, how to connect seemingly abstract information to find valuable information.
DEFINITION	<ul style="list-style-type: none"> Understanding linkage between contexts, solutions, needs, changes, stakeholders and value in both the abstract and the big picture. Finding patterns and connecting information that is of value.
EFFECTIVENESS MEASURES	Connecting disparate information and acting to better understand the relationships / situation.

9.1.7. VISUAL THINKING

PURPOSE	Ability to communication complex concepts and models into pictures that facilitate stakeholder engagement and understanding.
DEFINITION	<ul style="list-style-type: none"> Graphical representation of information, data, systems, and models to convey and integrate non-visual information and complex connections. Stakeholders learn more quickly and are able to connect points in their own context.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> Visuals allow for comparisons, pattern finding, and idea mapping. Stakeholders are engaged at a deeper level than with text alone.

9.2 BEHAVIOURAL CHARACTERISTICS

Behavioural characteristics are not unique to business analysis but they have been found to increase personal effectiveness in the practice of business analysis. These characteristics exist at the core of every business analyst's skill set. Each of the behavioural characteristics described here can impact the outcome of the practitioner's efforts.

The core competencies of behavioural characteristics focus on the skills and behaviours that allow a business analyst to gain the trust and respect of stakeholders. Business analysts do this by consistently acting in an ethical manner, completing tasks on time and to expectations, efficiently delivering quality results, and demonstrating adaptability to changing needs and circumstances.

Behavioural Characteristics core competencies include:

- Ethics,
- Personal Accountability,
- Trustworthiness,
- Organization and Time Management, and

- Adaptability.

9.2.1 ETHICS

PURPOSE	Behaving ethically and thinking of ethical impacts on others allows business analysts to earn the respect of the stakeholders. The ability to recognize when a proposed solution or requirement may present ethical difficulties to an organization or its stakeholders is an important consideration that business analysts can use to help reduce exposure to risk.
DEFINITION	Ethics require an understanding and focus on fairness, consideration, and moral behaviour through business analysis activities and relationships. Ethical behaviour includes consideration of the impact that a proposed solution can have on all stakeholder groups and working to ensure that those groups are treated as fairly as possible. Fair treatment does not require that the outcome be beneficial to a particular stakeholder group, but it does require that the affected stakeholders understand the reasons for decisions. Awareness of ethical issues allows business analysts to identify when ethical dilemmas occur and recommend resolutions to these dilemmas.
EFFECTIVENESS MEASURES	Measures of effective ethical behaviour include: <ul style="list-style-type: none"> • prompt identification and resolution of ethical dilemmas, • feedback from stakeholders confirming they feel decisions and actions are transparent and fair, • decisions made with consideration of the interests of all stakeholders, • reasoning for decisions that is clearly articulated and understood, • full and prompt disclosure of potential conflicts of interest, and • honesty regarding one's abilities, the performance of one's work, and accepting responsibility for failures or errors.

9.2.2 PERSONAL ACCOUNTABILITY

PURPOSE	Personal accountability is important for a business analyst because it ensures business analysis tasks are completed on time and to the expectations of colleagues and stakeholders. It enables the business analyst to establish credibility by ensuring that business analysis efforts meet the needs of the business.
DEFINITION	Personal accountability includes effectively planning business analysis work to achieve targets and goals, and ensuring that value delivered is aligned with business needs. It involves chasing down all leads and loose ends to fully satisfy the stakeholder's needs. Following through on and fully completing business analysis tasks produces complete,

	accurate, and relevant solutions traceable to a need. Business analysts take responsibility for identifying and escalating risks and issues. They also ensure that decision makers have the appropriate information in order to assess impact.
EFFECTIVENESS MEASURES	Measures of effective personal accountability include: <ul style="list-style-type: none"> • work effort is planned and easily articulated to others, • work is completed as planned or re-planned with sufficient reasoning and lead time, • status of both planned and unplanned work is known, • stakeholders feel that work is organized, • risks and issues are identified and appropriately acted on, • completely traceable requirements are delivered on time, and • stakeholder needs are met.

9.2.3 TRUSTWORTHINESS

PURPOSE	Earning the trust of stakeholders helps business analysts elicit business analysis information around sensitive issues and enables them to help stakeholders have confidence that their recommendations will be evaluated properly and fairly.
DEFINITION	Trustworthiness is the perception that one is worthy of trust. A business analyst being considered trustworthy may offset the natural fear of change experienced by many stakeholders. Several factors can contribute to being considered trustworthy: <ul style="list-style-type: none"> • intentionally and consistently completing tasks and deliverables on time, within budget, and achieving expected results so that colleagues and stakeholders consider the business analyst's behaviour dependable and diligent, • presenting a consistent attitude of confidence, so that colleagues and stakeholders consider the business analyst's demeanor as strong, • acting in an honest and straightforward manner, addressing conflict and concerns immediately so that colleagues and stakeholders consider the business analyst's morals as being honest and transparent, and <p>maintaining a consistent schedule over a long period of time so that colleagues and stakeholders consider the business analyst's availability predictable and reliable</p>

EFFECTIVENESS MEASURES	<p>Measures of effective trustworthiness include:</p> <ul style="list-style-type: none"> • stakeholders involve the business analyst in discussions and decision making, • stakeholders bring issues and concerns to the business analyst, • stakeholders are willing to discuss difficult or controversial topics with the business analyst, • stakeholders do not blame the business analyst when problems occur, • stakeholders respect the business analyst's ideas and referrals, and • stakeholders respond to the business analyst's referrals with positive feedback.
-------------------------------	---

9.2.4 ORGANIZATION AND TIME MANAGEMENT

PURPOSE	<p>Organization and time management skills help business analysts perform tasks effectively and use work time efficiently.</p>
DEFINITION	<p>Organization and time management involves the ability to prioritize tasks, perform them efficiently, and manage time effectively.</p> <p>Business analysts are constantly acquiring and accumulating significant quantities of information, and this information must be organized and stored in an efficient manner so that it can be used and reused at a later date. Business analysts must also be able to differentiate important information that should be retained from less important information.</p> <p>Effective time management requires the ability to prioritize tasks and deadlines.</p> <p>Techniques of organization include establishing short- and long-term goals, action plans, prioritizing tasks, and utilizing a checklist.</p> <p>Techniques for effective time management include establishing time limits on non-critical tasks, focusing more time on high risk and priority tasks, setting aside focus time, and managing potential interruptions.</p>

EFFECTIVENESS MEASURES	<p>Measures of effective organization and time management include:</p> <ul style="list-style-type: none"> • the ability to produce deliverables in a timely manner, • stakeholders feel that the business analyst focuses on the correct tasks at the right time, • schedule of work effort and deadlines is managed and communicated to stakeholders, • stakeholders feel their time in meetings and in reading communications is well spent, • complete preparation for meetings, interviews, and requirements workshops, • relevant business analysis information is captured, organized, and documented, • adherence to the project schedule and the meeting of deadlines, • provides accurate, thorough, and concise information in a logical manner which is understood by stakeholders, and • maintains up-to-date information on the status of each work item and all outstanding work.
-------------------------------	--

9.2.5 ADAPTABILITY

PURPOSE	<p>Business analysts frequently work in rapidly changing environments and with a variety of stakeholders. They adjust their behavioural style and method of approach to increase their effectiveness when interacting with different stakeholders, organizations, and situations.</p>
DEFINITION	<p>Adaptability is the ability to change techniques, style, methods, and approach. By demonstrating a willingness to interact with and complete tasks in a manner preferable to the stakeholders, business analysts can maximize the quality of service delivered and more efficiently help the organization achieve its goals and objectives. Having the curiosity to learn what others need and possessing the courage to try a different behaviour is adapting to situations and context.</p> <p>Business analysts sometimes have to modify the way they interact with stakeholders, such as the way they conduct interviews or the way they facilitate workshops. Different stakeholders have different levels of comfort with techniques that are in the business analysis tool kit. Some stakeholders are more visual and respond better to information that is represented visually in models, diagrams, and pictures. Other stakeholders are more verbal and prefer textual descriptions. Being able to determine which techniques will work and which will not, and then adapt accordingly increases the likelihood of a successful interaction.</p> <p>In the event that the goals and objectives of the organization change, business analysts respond by accepting the changes and adapting to a new mandate. Similarly, when circumstances arise or unanticipated problems occur, business analysts adapt by altering their plans and identifying options that can be used to deliver maximum value. The</p>

	<p>business analyst adapts when the business or stakeholder needs change, or when the context of the goal or the objective changes. When the need itself changes, the business analyst adapts by altering the plans and the approach in order to ensure that value is provided and delivered as part of the solution.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective adaptability include:</p> <ul style="list-style-type: none"> • demonstrating the courage to act differently from others, • adapting to changing conditions and environments, • valuing and considering other points of view and approaches, • demonstrating a positive attitude in the face of ambiguity and change, • demonstrating a willingness to learn new methods, procedures, or techniques in order to accomplish goals and objectives, • changing behaviour to perform effectively under changing or unclear conditions, • acquiring and applying new information and skills to address new challenges, • acceptance of having changes made to tasks, roles and project assignments as organizational realities change, • altering interpersonal style to highly diverse individuals and groups in a range of situations, and • evaluating what worked, what did not, and what could be done differently next time.

9.3 BUSINESS KNOWLEDGE

Business knowledge is required for the business analyst to perform effectively within their business, industry, organization, solution, and methodology. Business knowledge enables the business analyst to better understand the overarching concepts that govern the structure, benefits, and value of the situation as it relates to a change or a need.

Business Knowledge underlying competencies include:

- Business Acumen,
- Industry Knowledge,
- Organization Knowledge,
- Solution Knowledge, and
- Methodology Knowledge.

9.3.1 BUSINESS ACUMEN

PURPOSE	Business analysis requires an understanding of fundamental business principles and best practices in order to ensure they are considered as solutions are reviewed.
DEFINITION	Business acumen is the ability to understand business needs using experience and knowledge obtained from other situations. Organizations

	<p>frequently share similar practices, such as legal and regulatory requirements, finance, logistics, sales, marketing, supply chain management, human resources, and technology. Business acumen is the ability to understand and apply the knowledge based on these commonalities within differing situations.</p> <p>Understanding how other organizations have solved challenges may be useful when seeking possible solutions. Being aware of the experiences or challenges encountered in the past may assist a business analyst in determining which information may be applicable to the current situation. Factors that may cause differences in practices can include industry, location, size of organization, culture, and the maturity of the organization.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective business acumen include:</p> <ul style="list-style-type: none"> • demonstrating the ability to recognize potential limitations and opportunities, • demonstrating the ability to recognize when changes to a situation may require a change in the direction of an initiative or effort, • understanding the risks involved and the ability to make decisions on managing risks, • demonstrating the ability to recognize an opportunity to decrease expenses and increase profits, and • understanding the options available to address emerging changes in the situation.

9.3.2 INDUSTRY KNOWLEDGE

PURPOSE	<p>Industry knowledge provides the business analyst with an understanding of current practices and activities within an industry, and similar processes across industries.</p>
DEFINITION	<p>Industry knowledge is an understanding of:</p> <ul style="list-style-type: none"> • current trends, • market forces, • market drivers, • key processes, • services, • products, • definitions, • customer segments, • suppliers,

	<ul style="list-style-type: none"> • practices, • regulations, and • other factors that impact or are impacted by the industry and related industries. <p>Industry knowledge is also an understanding of how a company is positioned within an industry, and its impacts and dependencies, in regards to the market and human resources.</p> <p>When developing knowledge about a particular industry, competitor, or company the following set of questions can provide guidance:</p> <ul style="list-style-type: none"> • Who are the top leaders in the industry? • Which organizations promote or regulate the industry? • What are the benefits of being involved with these organizations? • Who is creating publicity releases, participating in conventions, and delivering marketing materials? • What are the comparisons of products and services? • What are the satisfaction indicators/benchmarking projects that are applicable? • What are the suppliers, practices, equipment and tools used by each company, and why do they use them? • What are the potential impacts of weather, political unrest, or natural disasters? • Who are the target customers and are they the same for the competition? • What impacts the seasonal cycles for production, marketing, and sales? Does it impact staffing or require changes in processes?
EFFECTIVENESS MEASURES	<p>Measures of effective industry knowledge include:</p> <ul style="list-style-type: none"> • being aware of activities within both the enterprise and the broader industry, • having knowledge of major competitors and partners, • the ability to identify key trends shaping the industry, • being familiar with the largest customer segments, • having knowledge of common products and product types, • being knowledgeable of sources of information about the industry, including relevant trade organizations or journals, • understanding of industry specific terms, standards, processes and methodologies, and • understanding of the industry regulatory environment.

9.3.3 ORGANIZATION KNOWLEDGE

PURPOSE	Organization knowledge provides an understanding of the management structure and business architecture of the enterprise.
DEFINITION	Organization knowledge includes an understanding of how the enterprise generates profits, accomplishes its goals, its organizational structure, the relationships that exist between business units, and the persons who occupy key stakeholder positions. Organization knowledge also includes understanding the organization's formal and informal communication channels as well as an awareness of the internal politics that influence decision making.
EFFECTIVENESS MEASURES	Measures of effective organization knowledge include: <ul style="list-style-type: none">• the ability to act according to informal and formal communications and authority channels,• understanding of terminology or jargon used in the organization,• understanding of the products or services offered by the organization,• the ability to identify subject matter experts (SMEs) in the organization, and• the ability to navigate organizational relationships and politics.

9.3.4 SOLUTION KNOWLEDGE

PURPOSE	Solution knowledge allows business analysts to leverage their understanding of existing departments, environments, or technology to efficiently identify the most effective means of implementing a change.
DEFINITION	When the business analysis effort involves improving an existing solution, business analysts apply knowledge and experience from the previous work on the solution. Familiarity with the range of commercially available solutions or suppliers can assist with the identification of possible alternatives. The business analyst may leverage knowledge gained from prior experiences to expedite the discovery of potential changes through elicitation or in-depth analysis.

EFFECTIVENESS MEASURES	<p>Measures of effective solution knowledge include:</p> <ul style="list-style-type: none"> • reduced time or cost to implement a required change, • shortened time on requirements analysis and/or solution design, • understanding when a larger change is, or is not, justified based on business benefit, and • understanding how additional capabilities that are present, but not currently used, can be deployed to provide value.
-------------------------------	---

9.3.5 METHODOLOGY KNOWLEDGE

PURPOSE	<p>Understanding the methodologies used by the organization provides the business analyst with information regarding context, dependencies, opportunities, and constraints used when developing a business analysis approach.</p>
DEFINITION	<p>Methodologies determine the timing (big steps or small increments), the approach, the role of those involved, the accepted risk level, and other aspects of how a change is approached and managed. Organizations adopt or create their own methodologies to fit varying levels of culture, maturity, adaptability, risk, uncertainty, and governance.</p> <p>Knowledge regarding a variety of methodologies allows the business analyst to quickly adapt to, and perform in, new environments.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective methodology knowledge include:</p> <ul style="list-style-type: none"> • the ability to adapt to changes in methodologies, • the willingness to use or learn a new methodology, • the successful integration of business analysis tasks and techniques to support the current methodology, • familiarity with the terms, tools, and techniques prescribed by a methodology, and • the ability to play multiple roles within activities prescribed by a methodology.

9.4 COMMUNICATION SKILLS

Communication is the act of a sender conveying information to a receiver in a method which delivers the meaning the sender intended. Active listening skills help to deepen understanding and trust between the sender and the receiver. Effective communication benefits all stakeholders.

Communication may be accomplished using a variety of delivery methods: verbal, non-verbal, physical, and written. Most communication methods deal with words, while some methods deal

with movements and expressions. Words, gestures, and phrases may have different meanings to different individuals. Effective communication involves both the sender and receiver possessing the same understanding of the information being communicated. A shared glossary of terms and clear goals are effective tools to avoid misunderstandings and the resulting complications.

Effective communication includes adapting communication styles and techniques to the knowledge level and communication styles of recipients. Effective communicators understand how tone, body language, and context change the meaning of words. Gaining an understanding of the terms and concepts (prior to the exchange) can provide fruitful benefits.

Planning effective communication includes the sender reviewing the information that is known about the receiver. Differences between the sender and the receiver, such as native language, culture, motivations, priorities, communication, learning, and thinking styles may call for specific communication methods. Each piece of information must be carefully crafted and packaged to ensure it is clear and understood.

When planning to communicate information, the following considerations may be helpful:

- consider what the receiver knows or does not know,
- structure the information in a logical, comprehensible manner,
- determine how to best present the information to convey the intended meanings (for example, using visual aids, graphs, diagrams, or bullet points), and
- understand the expectations of the recipients.
- Communication Skills core competencies include:
- Verbal Communication,
- Non-Verbal Communication,
- Written Communication, and
- Listening.

9.4.1 VERBAL COMMUNICATION

PURPOSE	Business analysts use verbal communication to convey ideas, concepts, facts, and opinions to a variety of stakeholders.
DEFINITION	Verbal communication uses spoken words to convey information from the sender to the receiver. Verbal communication skills are used to express business analysis information, ideas, concepts, facts, and opinions. It allows for the efficient transfer of information, including emotional and other non-verbal cues. It can be paired with both written and non-verbal communication. Verbal communication deals specifically with the sender's choice of words and the tone of voice. When the receiver is able to see the sender, the sender's non-verbal communication impacts the meaning of the message being understood by the receiver. When the sender is able to see the receiver, the receiver is providing a response and both the

	<p>sender and receiver are engaged in a dialogue, even though the receiver may not be speaking verbally. Monitoring the receiver's non-verbal communication allows the sender to consider adapting the message for the receiver.</p> <p>Having an understanding of the tone of the communication and how it can positively or negatively influence the listener allows the business analyst to more effectively communicate verbally. Effective verbal communication skills include the ability to make one's meaning understood. The sender should partner verbal communication with active listening to ensure that information presented is being understood by the receiver.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective verbal communication include:</p> <ul style="list-style-type: none"> • restating concepts to ensure all stakeholders clearly understand the same information, • assisting conversations to reach productive conclusions, • delivering effective presentations by designing and positioning content and objectives appropriately, and • communicating an issue's important points in a calm and rational manner, and presenting solution options.

9.4.2 NON-VERBAL COMMUNICATION

PURPOSE	<p>Non-verbal communication skills enable the effective sending and receiving of messages through—but not limited to—body movement, posture, facial expressions, gestures, and eye contact.</p>
DEFINITION	<p>Communication is typically focused upon words that are written or spoken. Non-verbal communication, however, is believed to convey much more meaning than words alone. Moods, attitudes, and feelings impact body movement and facial expressions. Non-verbal communication begins immediately when one person is able to see another. The effective use of non-verbal communication skills can present a trustworthy, confident, and capable demeanor. Being aware of non-verbal communication provides the opportunity to be aware and address the feelings of others that are not expressed verbally.</p> <p>Observing gestures or expressions cannot provide a complete understanding of the message being expressed by these non-verbal cues. These cues are indicators of the feelings and intent of the communicator. For example, when a stakeholder's non-verbal communication does not agree with their verbal message, the business analyst may want to explore the conversation further to uncover the source of this disagreement.</p>

EFFECTIVENESS MEASURES	<p>Measures of effective non-verbal communication include:</p> <ul style="list-style-type: none"> • being aware of body language in others, but not assuming a complete understanding through non-verbal communication, • intentional awareness of personal non-verbal communication, • improving trust and communication as a result of non-verbal communication, and • effectively addressing and resolving situations when a stakeholder's non-verbal communication does not agree with their verbal message.
-------------------------------	--

9.4.3 WRITTEN COMMUNICATION

PURPOSE	<p>Business analysts use written communication to convey ideas, concepts, facts, and opinions to variety of stakeholders.</p>
DEFINITION	<p>Written communication is the practice of using text, symbols, models (formal or informal), and sketches to convey and share information. An understanding of the audience is beneficial to effectively use written communication. Presenting information and ideas requires selecting the correct words so the audience will understand the intended meaning. Written communication has the added challenge of presenting information at a time or place that is remote from the time and place it was created.</p> <p>Effective written communication requires a broad vocabulary, strong grasp of grammar and style, and an understanding of the terms which will be understood by the audience. Written communication has the potential to convey a great deal of information; however, conveying information effectively is a skill which must be developed.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective written communication include:</p> <ul style="list-style-type: none"> • adjusting the style of writing for the needs of the audience, • proper use of grammar and style, • choosing words the audience will understand the intended meaning of, and • ability of the reader to paraphrase and describe the content of the written communication.

9.4.4 LISTENING

PURPOSE	Effective listening allows the business analyst to accurately understand information that is communicated verbally.
DEFINITION	Listening is the process of not just hearing words but understanding their meaning in context. By exhibiting effective listening skills, business analysts not only have a greater opportunity to accurately understand what is being communicated, but also to demonstrate that they think what the speaker is saying is important. Active listening involves both listening and interpreting what the other person is trying to communicate beyond the words used in order to understand the essence of the message. Active listening includes summarizing and repeating what was stated in different terms in order to ensure that both the listener and the speaker have the same understanding.
EFFECTIVENESS MEASURES	Measures of effective listening include: <ul style="list-style-type: none"> • giving the speaker undivided attention, • acknowledging the speaker with verbal or non-verbal encouragement, • providing feedback to the person or the group that is speaking to ensure there is an understanding, and • using active listening skills by deferring judgment and responding appropriately.

9.5 INTERACTION SKILLS

Interaction skills are represented by the business analyst's ability to relate, cooperate, and communicate with different kinds of people including executives, sponsors, colleagues, team members, developers, vendors, learning and development professionals, end users, customers, and subject matter experts (SMEs).

Business analysts are uniquely positioned to facilitate stakeholder communication, provide leadership, encourage comprehension of solution value, and promote stakeholder support of the proposed changes.

Interaction Skills core competencies include:

- Facilitation,
- Leadership and Influencing,
- Teamwork,
- Negotiation and Conflict Resolution, and
- Teaching.

9.5.1 FACILITATION

PURPOSE	Business analysts facilitate interactions between stakeholders in order to help them make a decision, solve a problem, exchange ideas and information, or reach an agreement regarding the priority and the nature of requirements. The business analyst may also facilitate interactions between stakeholders for the purposes of negotiation and conflict resolution (as discussed in Negotiation and Conflict Resolution).
DEFINITION	Facilitation is the skill of moderating discussions within a group in order to enable all participants to effectively articulate their views on a topic under discussion, and to ensure that participants in the discussion are able to recognize and appreciate the differing points of view that are articulated.
EFFECTIVENESS MEASURES	<p>Measures of effective facilitation include:</p> <ul style="list-style-type: none"> • making it clear to the participants that the facilitator is a third party to the process and not a decision maker nor the owner of the topic, • encouraging participation from all attendees, • remaining neutral and not taking sides, but at the same time being impartial and intervening when required in order to make suggestions and offer insights, • establishing ground rules such as being open to suggestions, building on what is there, not dismissing ideas, and allowing others to speak and express themselves, • ensuring that participants in a discussion correctly understand each other's positions, • using meeting management skills and tools to keep discussions focused and organized, • preventing discussions from being sidetracked onto irrelevant topics, and • understanding and considering all parties' interests, motivations, and objectives.

9.5.2 LEADERSHIP AND INFLUENCING

PURPOSE	Business analysts use leadership and influencing skills when guiding stakeholders during the investigation of business analysis information and solution options. They build consensus and encourage stakeholder support and collaboration during change.
DEFINITION	Leadership and influencing involves motivating people to act in ways that enable them to work together to achieve shared goals and objectives.

	<p>Understanding the individual motives, needs, and capabilities of each stakeholder and how those can be effectively channeled assists business analysts in meeting the shared objectives of the organization. The business analyst's responsibility for defining, analyzing, and communicating business analysis information provides opportunities for leadership and influencing, whether or not there are people formally reporting to the business analyst.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective leadership and influencing include:</p> <ul style="list-style-type: none"> • reduced resistance to necessary changes, • articulation of a clear and inspiring vision of a desired future state, • success in inspiring others to turn vision into action, • influence on stakeholders to understand mutual interests, • effective use of collaboration techniques to influence others, • influence on stakeholders to consider broader objectives over personal motivations, and • re-framing issues so alternate perspectives can be understood and accommodated to influence stakeholders towards shared goals.

9.5.3 TEAMWORK

PURPOSE	Teamwork skills allow business analysts to work productively with team members, stakeholders, and any other vested partners so that solutions can be effectively developed and implemented.
DEFINITION	<p>Business analysts often work as part of a team with other business analysts, project managers, stakeholders, and subject matter experts (SMEs). Relationships with people in those roles are a critical part of the success of any project or enterprise. It is important for the business analyst to understand how a team is formed and how it functions. Recognizing team dynamics and how they play a part as the team progresses through various stages of a project is also crucial. Knowing and adapting to how and when a team is progressing through a project's life cycle can lower the negative influences that impact a team.</p> <p>Building and maintaining trust of teammates contributes to the integrity of the team as a whole and helps the team perform at its fullest capacity. When team members actively foster an environment for positive and trusting team dynamics, difficult decisions and challenges become less complicated.</p> <p>Team conflict is common. If handled well, the resolution of conflict can benefit the team. Resolving conflict requires the team to focus on examining the positions, assumptions, observations, and expectations of all team members. Working through such problems can have the beneficial effect of strengthening the foundation of the analysis and the</p>

EFFECTIVENESS MEASURES	<p>solution.</p> <p>Measures of effective teamwork include:</p> <ul style="list-style-type: none"> • fostering a collaborative working environment, • effectively resolving conflict, • developing trust among team members, • support among the team for shared high standards of achievement, and • promoting a shared sense of ownership of the team goals.
-------------------------------	---

9.5.4 NEGOTIATION AND CONFLICT RESOLUTION

PURPOSE	<p>Business analysts occasionally mediate negotiations between stakeholders in order to reach a common understanding or an agreement. During this process, business analysts help resolve conflicts and differences of opinion with the intent of maintaining and strengthening working relationships among stakeholders and team members.</p>
DEFINITION	<p>Negotiation and conflict resolution involves mediating discussions between participants in order to help them recognize that there are differing views on the topic, resolve differences, and reach conclusions that have the agreement of all participants. Successful negotiation and conflict resolution includes identifying the underlying interests of the parties, distinguishing those interests from their stated positions, and helping the parties identify solutions that satisfy those underlying interests. The business analyst accomplishes this while ensuring that the outcome of the resolution aligns with the overall solution and the business needs.</p>
EFFECTIVENESS MEASURES	<p>Measures of effective negotiation and conflict resolution include:</p> <ul style="list-style-type: none"> • a planned approach to ensure that the negotiation takes into account the tone of voice, the conveyed attitude, the methods used, and the concern for the other side's feelings and needs, • the ability to recognize that the needs of the parties are not always in opposition and that it is often possible to satisfy both parties without either side losing, • an objective approach to ensure the problem is separated from the person so that the real issues are debated without damaging working relationships, and • the ability to recognize that effective negotiation and conflict resolution are not always achieved in a single autonomous meeting, and that sometimes several meetings are required in order to achieve the stated goals.

9.5.5 TEACHING

PURPOSE	Teaching skills help business analysts effectively communicate business analysis information, concepts, ideas, and issues. They also help ensure that information is understood and retained by stakeholders.
DEFINITION	Teaching is the process of leading others to gain knowledge. Business analysts are responsible for confirming that the information communicated has been understood by stakeholders. Business analysts lead stakeholders to discover clarity in ambiguity by helping them learn about the contexts and value of the needs being investigated. This requires teaching skills in selecting the most appropriate visual, verbal, written, and kinesthetic teaching approaches according to the information or techniques being taught. The intent is to draw out stakeholder engagement and collaborative learning to gain clarity. Business analysts frequently elicit and learn new information, and then teach this information to stakeholders in a meaningful way.
EFFECTIVENESS MEASURES	Measures of effective teaching include: <ul style="list-style-type: none">• utilizing different methods to communicate information to be learned by stakeholders,• discovering new information through high levels of stakeholder engagement,• validating that audiences have a clear understanding of the key messages that are intended to be learned, and• verifying that the stakeholders can demonstrate the new knowledge, facts, concepts, and ideas.

9.6 TOOLS AND TECHNOLOGY

Business analysts use a variety of software applications to support communication and collaboration, create and maintain requirements artifacts, model concepts, track issues, and increase overall productivity.

Requirements documentation is often developed using word processing tools, while the process of developing business requirements may require the use of prototyping and simulation tools, as well as specialized tools for modelling and diagramming.

Requirements management technologies support requirements workflow, approvals, baselining, and change control. These technologies can also support the traceability between requirements and assist in determining the impact of changes to requirements.

Interacting with the stakeholders and team members may require the use of communication and collaboration tools, as well as presentation software in order to showcase ideas and generate discussion among stakeholders and team members.

Business Analysis Tools and Technology core competencies include:

- Office Productivity Tools and Technology,
- Business Analysis Tools and Technology, and
- Communication Tools and Technology.

9.6.1 OFFICE PRODUCTIVITY TOOLS AND TECHNOLOGY

PURPOSE	These tools are used to document and track information and artifacts.
DEFINITION	<ul style="list-style-type: none"> • Provide the ability to organize, dissect, manipulate, communicate, collaborate and understand information . • Tools include word processors, presentation software, spreadsheets, communication tools, collaboration and knowledge management tools, and hardware.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Awareness of available tools, their operations and abilities. • Ability to determine the tool that will best meet stakeholder needs and ability to use features of the tool.

9.6.2 BUSINESS ANALYSIS TOOLS AND TECHNOLOGY

PURPOSE	These tools are used to model, document and manage the outputs of business analysis activities and deliverables.
DEFINITION	Many tools are specific to business analysis, these include modelling, diagramming, analyzing and mapping requirements, mapping to business rules, requirements workflow and approvals, traceability, issue and risk analysis and tracking, and conflict and defect tracking.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Apply an understanding of the major tools currently available; describing strengths and weaknesses and best use. • Use the tools to complete requirement related activities to shorten time frames.

9.6.3 COMMUNICATION TOOLS AND TECHNOLOGY

PURPOSE	These tools are used to manage team interactions and collaborate with stakeholders.
DEFINITION	Knowing how to use a variety of communication tools and collaboration environments can enable more efficient and accurate communication and effective decision making.
EFFECTIVENESS MEASURES	<ul style="list-style-type: none"> • Selection of appropriate and effective tools for the audience and the purpose. • Ability to identify tools that meet communication needs and ability to use the features of the tool.

10. Techniques

The Techniques chapter provides a high-level overview of the techniques referenced in the Knowledge Areas of the BABOK® GUIDE. Techniques are methods business analysts use to perform business analysis tasks.

The techniques described in the BABOK® GUIDE are intended to cover the most common and widespread techniques practiced within the business analysis community. Business analysts apply their experience and judgment in determining which techniques are appropriate to a given situation and how to apply each technique. This may include techniques that are not described in the BABOK® GUIDE. As the practice of business analysis evolves, techniques will be added, changed, or removed from future iterations of the BABOK® GUIDE.

In a number of cases, a set of conceptually similar approaches have been grouped into a single technique. Any approach within a technique may be used individually or in combination to accomplish the technique's purpose.

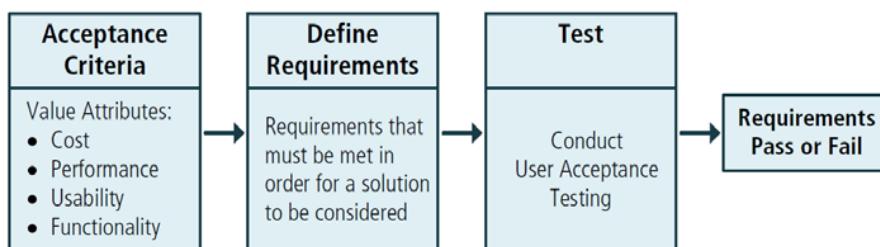
10.1 Acceptance and Evaluation Criteria	
Purpose	<ul style="list-style-type: none">Acceptance criteria are used to define the requirements, outcomes, or conditions that must be met in order for a solution to be considered acceptable to key stakeholders.Evaluation criteria are the measures used to assess a set of requirements in order to choose between multiple solutions.
Description	<ul style="list-style-type: none">Can be applied at all levels of a project, from high-level to a more detailed level.Acceptance criteria:<ul style="list-style-type: none">Describe the minimum set of requirements that must be met in order for a particular solution to be worth implementing.To determine if a solution or solution component can meet a requirement. <p>Acceptance criteria:</p> <ul style="list-style-type: none">Are typically used when only one possible solution is being evaluated, and are generally expressed as a pass or fail <p>Evaluation criteria:</p> <ul style="list-style-type: none">Define a set of measurements which allow for ranking of solutions and alternative designs according to their value for stakeholdersEach evaluation criterion represents a continuous or discrete scale for measuring a specific solution attribute such as cost, performance, usability, and how well the functionality represents the stakeholders' needsBoth evaluation and acceptance criteria may be defined with the same value attributes.When evaluating various solutions, the solutions with lower costs and better performance may be rated higher.
Elements	<p>Value Attributes</p> <ul style="list-style-type: none">Are the characteristics of a solution that determine or substantially influence its value for stakeholders.Represent a meaningful and agreed-upon decomposition of the value proposition into its constituent parts, which can be described as qualities that the solution should either possess or avoid.Ensure that the Acceptance and Evaluation Criteria are valid and relevant to

stakeholder needs and should be considered when accepting and evaluating the solution.

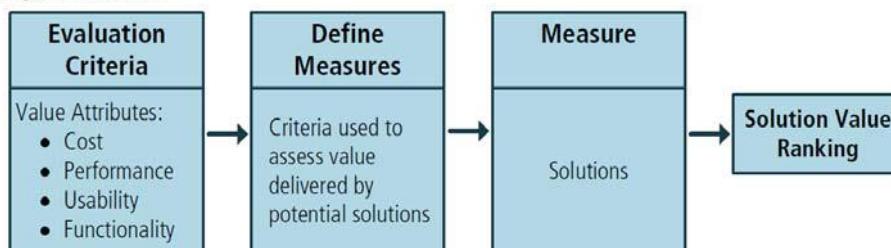
Examples:

- Ability to provide specific information
- Ability to perform or support specific operations
- Performance and responsiveness characteristics
- Applicability of the solution in specific situations and contexts
- Availability of specific features and capabilities
- Usability, security, scalability, and reliability
- Business analyst ensures that the definition of all value attributes are agreed upon by all stakeholders. Business analyst may design tools and instructions for performing the assessment as well as for recording and processing its results.

One Solution



Multiple Solutions



Assessment

- In order to assess a solution against acceptance or evaluation criteria, it must be constructed in a measurable format.

Testability

- Acceptance criteria are expressed in a testable form. This may require breaking requirements down into an atomic form so that test cases can be written to verify the solution against the criteria.
- Acceptance criteria are presented in the form of statements which can be verified as true or false. This is often achieved through user acceptance testing (UAT).

Measures

- Evaluation criteria provide a way to determine if features provide the value necessary to satisfy stakeholder needs.
- Evaluation criteria are presented as parameters that can be measured against a continuous or discrete scale.
- The definition of each evaluation criterion allows the solution to be measured through various methods such as benchmarking or expert judgment.
- Defining evaluation criteria may involve designing tools and instructions for performing the assessment, as well as for recording and processing its

	results.
Usage Considerations	<p>Strengths</p> <ul style="list-style-type: none"> • Agile methodologies may require that all requirements be expressed in the form of testable acceptance criteria. • Acceptance criteria are necessary when the requirements express contractual obligations. • Acceptance criteria provide the ability to assess requirements based on agreed-upon criteria. • Evaluation criteria provide the ability to assess diverse needs based on agreed-upon criteria, such as features, common indicators, local or global benchmarks, and agreed ratios. • Evaluation criteria assist in the delivery of expected return on investment (ROI) or otherwise specified potential value. • Evaluation criteria helps in defining priorities. <p>Limitations</p> <ul style="list-style-type: none"> • Acceptance criteria may express contractual obligations and as such may be difficult to change for legal or political reasons. • Achieving agreement on evaluation criteria for different needs among diverse stakeholders can be challenging.

10.2 Backlog Management

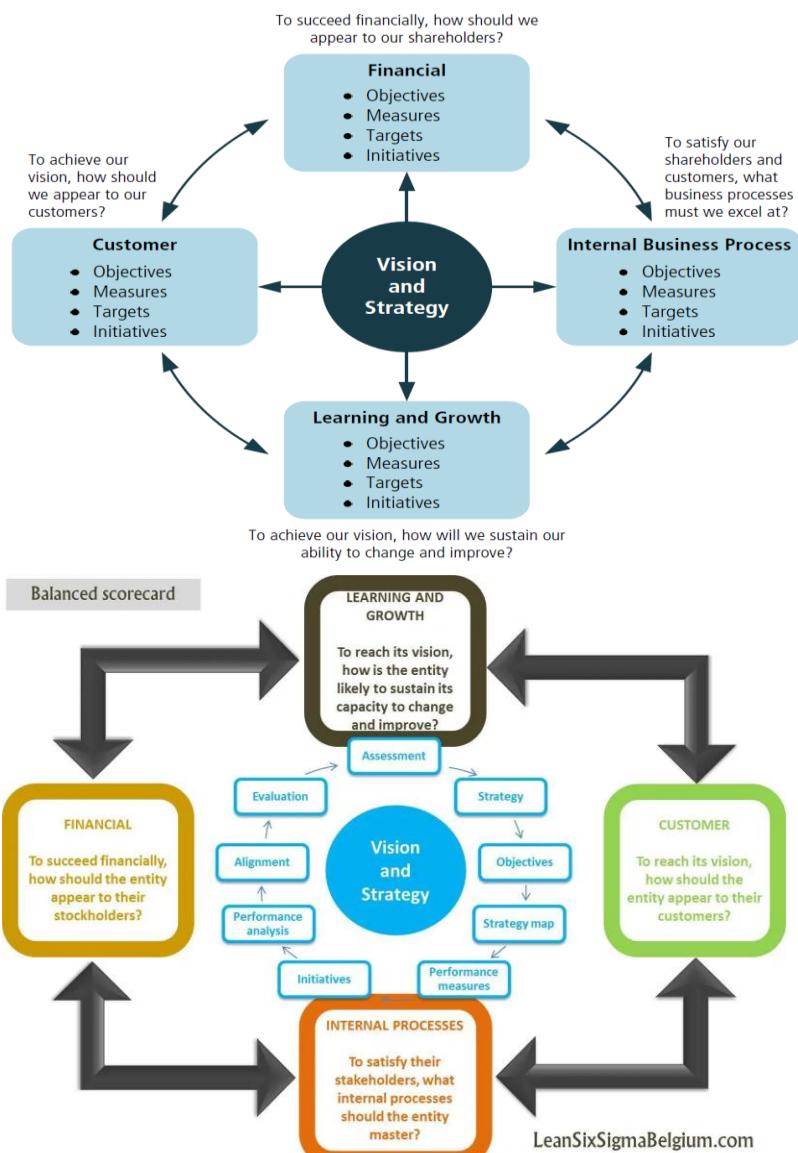
Purpose	The backlog is used to record, track, and prioritize remaining work items.
Description	<ul style="list-style-type: none"> • A backlog occurs when the volume of work items to be completed exceeds the capacity to complete them. • Backlog management refers to the planned approach to determine: <ul style="list-style-type: none"> • What work items should be formally included in the backlog • How to describe the work items • How the work items should be tracked • How the work items should be periodically reviewed and prioritized in relation to all other items in the backlog • How the work items are eventually selected to be worked on • How the work items are eventually removed from the backlog • In a managed backlog, the items at the top have the highest business value and the highest priority. • The backlog is reviewed at planned intervals. • The root causes for the changes are investigated: <ul style="list-style-type: none"> • A growing backlog could indicate an increase in demand or a drop in productivity • A declining backlog could indicate a drop in demand or improvements in the production process • There may be more than one backlog. For example, one backlog may be used to manage a global set of items, while a second backlog may be used to manage the items that are due to be worked on within the very near future.
Elements	<ul style="list-style-type: none"> • Items in the Backlog <ul style="list-style-type: none"> • Backlog items may be any kind of item which may have work associated with it. • An item is added to the backlog if it has value to a stakeholder. • There may be one person (for example Business Analyst) with the authority to add new items to the backlog, or there could be a committee which adds new items based on a consensus. • Prioritization

	<ul style="list-style-type: none"> Items in the backlog are prioritized relative to each other. The priorities will change as stakeholders' priorities change, or as dependencies between backlog items emerge. Rules on how to manage the backlog may also impact priority. A multi-phased prioritization approach can be used. <p>• Estimation</p> <ul style="list-style-type: none"> When an item is first added, there may be very little detail included, especially if the item is not likely to be worked on in the near term. Items near the top of the backlog are usually described in more detail, with a correspondingly accurate estimate about their relative size and complexity that would help to determine the cost and effort to complete them. A minimal amount of work is done on each item while it is on the backlog; just enough to be able to understand the work involved to complete it. As the work progresses on other items in the backlog, an individual item's relative priority may rise, leading to a need to review it and possibly further elaborate or decompose it to better understand and estimate its size and complexity. <p>• Managing Changes to the Backlog</p> <ul style="list-style-type: none"> When new or changed requirements are identified, they are added to the backlog and ordered to their relative to the other items already there. Whenever work capacity becomes available the backlog is reviewed and items are selected based on the available capacity, dependencies between items, current understanding of the size, and complexity. Items are removed from the backlog when they are completed, or if a decision has been made to not do any more work on them. However, removed items can be re-added to the backlog for a variety of reasons, including: <ul style="list-style-type: none"> Stakeholder needs could change significantly It could be more time-consuming than estimated Other priority items could take longer to complete than estimated The resulting work product might have defects
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> An effective approach to responding to changing stakeholder needs and priorities. Only items near the top of the backlog are elaborated and estimated in detail. Can be an effective communication vehicle. Limitations <ul style="list-style-type: none"> Large backlogs may become cumbersome and difficult to manage. It takes experience to be able to break down the work to be done into enough detail for accurate estimation. A lack of detail in the items in the backlog can result in lost information over time.

10.3 Balanced Scorecard

Purpose	The balanced scorecard is used to manage performance in any business model, organizational structure, or business process.
Description	<ul style="list-style-type: none"> The balanced scorecard:

- Is a strategic planning and management tool used to measure organizational performance beyond the traditional financial measures.
- Is outcome focused and provides a balanced view of an enterprise by implementing the strategic plan as an active framework of objectives and performance measures.
- Includes tangible objectives, specific measures, and targeted outcomes derived from an organization's vision and strategy.
- Can be used at multiple levels within an organization. This includes at an enterprise-wide level (macro level), departmental or function level, and even at the level of a project or initiative. an organization's vision and strategy.
- The balanced scorecard is composed of four dimensions:
 - Learning and Growth
 - Business Process
 - Customer
 - Financial



Elements	<ul style="list-style-type: none"> • Learning and Growth Dimension <ul style="list-style-type: none"> • Includes measures regarding employee training and learning, product and service innovation, and corporate culture. Metrics guide the use of training funds, mentoring, knowledge sharing, and technology improvements. • Business Process Dimension <ul style="list-style-type: none"> • Includes metrics that indicate how well the enterprise is operating and if their products meet customer needs. • Customer Dimension <ul style="list-style-type: none"> • Includes metrics on customer focus, satisfaction and delivery of value. • Financial Dimension <ul style="list-style-type: none"> • Identifies what is financially necessary to realize the strategy. Examples of financial measures indicate profitability, revenue growth, and added economic value. • Measures or Indicators <ul style="list-style-type: none"> • There are two basic types of measures or indicators: • Lagging indicators that provide results of actions already taken • Leading indicators that provide information about future performance • Objectives tend to have lagging indicators, but using related leading indicators can provide more real-time performance information
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Facilitates holistic and balanced planning and thinking. • Short, medium, and long-term goals can be harmonized into programs with incremental success measures. • Strategic, tactical, and operational teams are more easily aligned in their work. • Encourages forward thinking and competitiveness. • Limitations <ul style="list-style-type: none"> • A lack of a clear strategy makes aligning the dimensions difficult. • Can be seen as the single tool for strategic planning rather than just one tool to be used in a suite of strategic planning tool. • Can be misinterpreted as a replacement for strategic planning, execution, and measurement.

10.4 Benchmarking & Market Analysis

Purpose	Benchmarking and market analysis are conducted to improve organizational operations, increase customer satisfaction, and increase value to stakeholders.
Description	<ul style="list-style-type: none"> • The objective of benchmarking is to evaluate enterprise performance and ensure that the enterprise is operating efficiently. • Benchmarking may also be performed against standards for compliance purposes. • The objective of market analysis is to acquire this information in order to support the various decision-making processes within an organization. • Market analysis can also help determine when to exit a market. • It may be used to determine if partnering, merging, or divesting are viable alternatives for an enterprise.
Elements	<ul style="list-style-type: none"> • Benchmarking includes: <ul style="list-style-type: none"> • Identifying the areas to be studied

	<ul style="list-style-type: none"> Identifying enterprises that are leaders in the sector (including competitors) Conducting a survey of selected enterprises to understand their practices Using a Request for Information (RFI) to gather information about capabilities Arranging visits to best-in-class organizations Determining gaps between current and best practices Developing a project proposal to implement best practices Market Analysis requires that business analysts: <ul style="list-style-type: none"> Identify customers and understand their preferences Identify opportunities that may increase value to stakeholders Identify competitors and investigate their operations Look for trends in the market, anticipate growth rate, and estimate potential profitability Define appropriate business strategies Gather market data Use existing resources such as company records, research studies, and books and apply that information to the questions at hand Review data to determine trends and draw conclusions
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Benchmarking provides organizations with information about new and different methods, ideas, and tools to improve organizational performance. An organization may use benchmarking to identify best practices by its competitors in order to meet or exceed its competition. Benchmarking identifies why similar companies are successful and what processes they used to become successful. Market analysis can target specific groups and can be tailored to answer specific questions. Market analysis may expose weaknesses within a certain company or industry. Market analysis may identify differences in product offerings and services that are available from a competitor. Limitations <ul style="list-style-type: none"> Benchmarking is time-consuming; organizations may not have the expertise to conduct the analysis and interpret useful information. Benchmarking cannot produce innovative solutions or solutions that will produce a sustainable competitive advantage because it involves assessing solutions that have been shown to work elsewhere with the goal of reproducing them. Market analysis can be time-consuming and expensive, and the results may not be immediately available. Without market segmentation, market analysis may not produce the expected results or may provide incorrect data about a competitor's products or services
10.5 Brainstorming	
Purpose	<ul style="list-style-type: none"> Brainstorming is an excellent way to foster creative thinking about a problem. The aim of brainstorming is to produce numerous new ideas, and to derive from them themes for further analysis.
Description	<ul style="list-style-type: none"> Brainstorming is a technique intended to produce a broad or diverse set of options. It helps answer specific questions such:

	<ul style="list-style-type: none"> • What options are available to resolve the issue at hand? • What factors are constraining the group from moving ahead with an approach or option? • What could be causing a delay in activity 'A'? • What can the group do to solve problem 'B'? <p>• This technique is best applied in a group as it draws on the experience and creativity of all members of the group.</p> <pre> graph TD subgraph Phase1 [1. Preparation] P1[Define Area of Interest] --> P2[Determine Time Limit] P2 --> P3[Identify Participants] P3 --> P4[Establish Evaluation Criteria] end subgraph Phase2 [2. Session] S1[Share Ideas] --> S2[Record Ideas] S2 --> S3[Build on each others ideas] S3 --> S4[Elicit as many ideas as possible] end subgraph Phase3 [3. Wrap-up] W1[Discuss and Evaluate] --> W2[Create List] W2 --> W3[Rate Ideas] W3 --> W4[Distribute Final List] end </pre>
Elements	<ul style="list-style-type: none"> • Preparation <ul style="list-style-type: none"> • Develop a clear and concise definition of the area of interest. • Determine a time limit for the group to generate ideas; the larger the group, the more time required. • Identify the facilitator and participants in the session (6-8 participants who represent a range of backgrounds and experience with the topic). • Set expectations with participants and get their buy-in to the process. • Establish the criteria for evaluating and rating the ideas. • Session <ul style="list-style-type: none"> • Share new ideas without any discussion, criticism, or evaluation. • Visibly record all ideas. • Encourage participants to be creative, share exaggerated ideas, and build on the ideas of others. • Don't limit the number of ideas as the goal is to elicit as many as possible within the time period. • Wrap-up <ul style="list-style-type: none"> • Once the time limit is reached, discuss and evaluate the ideas using the predetermined evaluation criteria. • Create a condensed list of ideas, combine ideas where appropriate, and eliminate duplicates. • Rate the ideas, and then distribute the final list of ideas to the appropriate parties
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Ability to elicit many ideas in a short time period. • Non-judgmental environment enables creative thinking.

	<ul style="list-style-type: none"> • Can be useful during a workshop to reduce tension between participants. • Limitations <ul style="list-style-type: none"> • Participation is dependent on individual creativity and willingness to participate. • Organizational and interpersonal politics may limit overall participation. • Group participants must agree to avoid debating the ideas raised during brainstorming.
--	--

10.6 Business Capability Analysis

Purpose	Business capability analysis provides a framework for scoping and planning by generating a shared understanding of outcomes, identifying alignment with strategy, and providing a scope and prioritization filter.
Description	<ul style="list-style-type: none"> • Business capability analysis describes: <ul style="list-style-type: none"> • What an enterprise, or part of an enterprise, is able to do • The ability of an enterprise to act on or transform something that helps achieve a business goal or objective • Capabilities may be assessed for performance and associated risks to identify specific performance gaps and prioritize investment. • As long as an enterprise continues to perform similar functions, the capabilities required by the enterprise should remain constant – even if the method of execution for those capabilities undergoes significant changes.
Elements	<ul style="list-style-type: none"> • Capabilities <ul style="list-style-type: none"> • Capabilities are the abilities of an enterprise to perform or transform something that helps achieve a business goal or objective. • Capabilities describe the purpose or outcome of the performance or transformation, not how the performance or transformation is performed. • Each capability is found only once on a capability map, even if it is possessed by multiple business units. • Using Capabilities <ul style="list-style-type: none"> • Capabilities impact value through increasing or protecting revenue, reducing or preventing cost, improving service, achieving compliance, or positioning the company for the future. • Not all capabilities have the same level of value. There are various tools that can be used to make value explicit in a capability assessment. • Performance Expectations <ul style="list-style-type: none"> • Capabilities can be assessed to identify explicit performance expectations. • When a capability is targeted for improvement, a specific performance gap can be identified. The performance gap is the difference between the current performance and the desired performance, given the business strategy. • Risk Model <ul style="list-style-type: none"> • Capabilities alone do not have risks – the risks are in the performance of the capability, or in the lack of performance. These risks fall into the usual business categories: <ul style="list-style-type: none"> • Business risk • Technology risk • Organizational risk • Market risk

	<ul style="list-style-type: none"> Strategic Planning <ul style="list-style-type: none"> Business capabilities for the current state and future state of an enterprise can be used to determine where that enterprise needs to go in order to accomplish its strategy. A business capability assessment can produce a set of recommendations or proposals for solutions. This information forms the basis of a product roadmap and serves as a guide for release planning. At the strategic level, capabilities should support an enterprise in establishing and maintaining a sustainable competitive advantage and a distinct value proposition. Capability Maps <ul style="list-style-type: none"> Capability maps provide a graphical view of elements involved in business capability analysis. There is no set standard for the notation of capabilities maps 																																																																																																																																																																																																																																																																																																																																													
<table border="1"> <thead> <tr> <th rowspan="2">ORGANIZATIONAL ANALYSIS</th> <th colspan="3">Business Value</th> <th colspan="3">Customer Value</th> <th colspan="3">Performance Gap</th> <th colspan="3">Risk</th> </tr> <tr> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> </tr> </thead> <tbody> <tr> <td>Capability Analysis</td><td>Dark Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Root Cause Analysis</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Process Analysis</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Stakeholder Analysis</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Roadmap Construction</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th rowspan="2">PROJECT ANALYSIS</th> <th colspan="3">Business Value</th> <th colspan="3">Customer Value</th> <th colspan="3">Performance Gap</th> <th colspan="3">Risk</th> </tr> <tr> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> </tr> </thead> <tbody> <tr> <td>Requirements Elicitation</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Requirements Management</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Requirements Communication</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>User Acceptance Testing</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Usability Testing</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th rowspan="2">PROFESSIONAL DEVELOPMENT</th> <th colspan="3">Business Value</th> <th colspan="3">Customer Value</th> <th colspan="3">Performance Gap</th> <th colspan="3">Risk</th> </tr> <tr> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> </tr> </thead> <tbody> <tr> <td>Organizational Consulting</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Project Analysis Consulting</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Training</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Mentoring</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Resources Maintenance</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th rowspan="2">MANAGEMENT</th> <th colspan="3">Business Value</th> <th colspan="3">Customer Value</th> <th colspan="3">Performance Gap</th> <th colspan="3">Risk</th> </tr> <tr> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> <th>High</th> <th>Med</th> <th>Low</th> </tr> </thead> <tbody> <tr> <td>Performance Management</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Dark Blue</td><td>Light Blue</td><td>Light Blue</td></tr> <tr> <td>Resource Allocations</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> <tr> <td>Employee Dev Planning</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Light Blue</td><td>Medium Blue</td><td>Light Blue</td></tr> </tbody> </table>	ORGANIZATIONAL ANALYSIS	Business Value			Customer Value			Performance Gap			Risk			High	Med	Low	Capability Analysis	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Root Cause Analysis	Light Blue	Process Analysis	Light Blue	Medium Blue	Light Blue	Medium Blue	Light Blue	Stakeholder Analysis	Dark Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Roadmap Construction	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	PROJECT ANALYSIS	Business Value			Customer Value			Performance Gap			Risk			High	Med	Low	Requirements Elicitation	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Requirements Management	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Requirements Communication	Dark Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	User Acceptance Testing	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Usability Testing	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Medium Blue	Light Blue	PROFESSIONAL DEVELOPMENT	Business Value			Customer Value			Performance Gap			Risk			High	Med	Low	Organizational Consulting	Light Blue	Medium Blue	Light Blue	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Project Analysis Consulting	Light Blue	Training	Dark Blue	Light Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Mentoring	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Resources Maintenance	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Dark Blue	Medium Blue	Light Blue	MANAGEMENT	Business Value			Customer Value			Performance Gap			Risk			High	Med	Low	Performance Management	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Resource Allocations	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Employee Dev Planning	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Medium Blue	Light Blue																																																																																															
ORGANIZATIONAL ANALYSIS		Business Value			Customer Value			Performance Gap			Risk																																																																																																																																																																																																																																																																																																																																			
	High	Med	Low	High	Med	Low	High	Med	Low	High	Med	Low																																																																																																																																																																																																																																																																																																																																		
Capability Analysis	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Root Cause Analysis	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Process Analysis	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Stakeholder Analysis	Dark Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Roadmap Construction	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
PROJECT ANALYSIS	Business Value			Customer Value			Performance Gap			Risk																																																																																																																																																																																																																																																																																																																																				
	High	Med	Low	High	Med	Low	High	Med	Low	High	Med	Low																																																																																																																																																																																																																																																																																																																																		
Requirements Elicitation	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Requirements Management	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Dark Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Requirements Communication	Dark Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
User Acceptance Testing	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Usability Testing	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
PROFESSIONAL DEVELOPMENT	Business Value			Customer Value			Performance Gap			Risk																																																																																																																																																																																																																																																																																																																																				
	High	Med	Low	High	Med	Low	High	Med	Low	High	Med	Low																																																																																																																																																																																																																																																																																																																																		
Organizational Consulting	Light Blue	Medium Blue	Light Blue	Dark Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Project Analysis Consulting	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Training	Dark Blue	Light Blue	Light Blue	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Mentoring	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Resources Maintenance	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Dark Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
MANAGEMENT	Business Value			Customer Value			Performance Gap			Risk																																																																																																																																																																																																																																																																																																																																				
	High	Med	Low	High	Med	Low	High	Med	Low	High	Med	Low																																																																																																																																																																																																																																																																																																																																		
Performance Management	Dark Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Dark Blue	Light Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Resource Allocations	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Employee Dev Planning	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue	Light Blue	Light Blue	Light Blue	Light Blue	Medium Blue	Light Blue																																																																																																																																																																																																																																																																																																																																		
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Provides a shared articulation of outcomes, strategy, and performance, which help create very focused and aligned initiatives. Helps align business initiatives across multiple aspects of the organization. Useful when assessing the ability of an organization to offer new products and services. Limitations <ul style="list-style-type: none"> Requires an organization to agree to collaborate on this model. When created unilaterally or in a vacuum it fails to deliver on the goals of alignment and shared understanding. Requires a broad, cross-functional collaboration in defining the capability model and the value framework. 																																																																																																																																																																																																																																																																																																																																													

10.7 Business Cases	
Purpose	A business case provides a justification for a course of action based on the benefits to be realized by using the proposed solution, as compared to the cost, effort, and other considerations to acquire and live with that solution.
Description	<ul style="list-style-type: none"> • A business case captures the rationale for undertaking a change and is used to: <ul style="list-style-type: none"> • Define the need • Determine the desired outcomes • Assess constraints, assumptions, and risks • Recommend a solution
Elements	<ul style="list-style-type: none"> • Need Assessment <ul style="list-style-type: none"> • Is the relevant business goal or objective that must be met. • Identifies the problem or the potential opportunity. • Desired Outcomes <ul style="list-style-type: none"> • Describe the state which should result if the need is fulfilled. • Should include measurable outcomes that can be utilized to determine the success of the business case or the solution. • Should be revisited at defined milestones and at the completion of the initiative (or initiatives) to fulfill the business case. • Should also be independent of the recommended solution. As solution options are assessed, their ability to achieve the desired outcomes will help determine the recommended solution. • Assess Alternatives <ul style="list-style-type: none"> • Alternatives: <ul style="list-style-type: none"> • May include different technologies, processes, or business models. • May include different ways of acquiring these and different timing options. • Will be affected by constraints such as budget, timing, and regulatory. • The 'do-nothing' alternative should be assessed and considered for the recommended solution. • Each alternative should be assessed in terms of: <ul style="list-style-type: none"> • Scope • Feasibility • Assumptions, Risks and Constraints • Financial analysis and Value assessment • Recommended Solution <ul style="list-style-type: none"> • Describes the most desirable way to solve the problem or leverage the opportunity. • May also include some estimates of cost and duration to implement the solution.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Provides an amalgamation of the complex facts, issues, and analysis required to make decisions regarding change. • Provides a detailed financial analysis of cost and benefits. • Provides guidance for ongoing decision making throughout the initiative. • Limitations <ul style="list-style-type: none"> • May be subject to the biases of authors. • Frequently not updated once funding for the initiative is secured.

	<ul style="list-style-type: none"> Contains assumptions regarding costs and benefits that may prove invalid upon further investigation.
10.8 Business Model Canvas	
Purpose	A business model canvas describes how an enterprise creates, delivers, and captures value for and from its customers.
Description	<ul style="list-style-type: none"> A business model canvas is comprised of nine building blocks that describe how an organization intends to deliver value. <ul style="list-style-type: none"> These are arranged on a business canvas that shows the relationship between the organization's operations, finance, customers, and offerings. A business model canvas: <ul style="list-style-type: none"> Serves as a blueprint for implementing a strategy. Can be used as a diagnostic and planning tool regarding strategy and initiatives. Allows for the mapping of programs, projects, and other initiatives to the strategy of the enterprise.
Elements	<ul style="list-style-type: none"> Value Propositions <ul style="list-style-type: none"> Represents what a customer is willing to exchange for having their needs met. May consist of a single product or service, or may be comprised of a set of goods and services that are bundled together to address the needs of a customer or customer segment to help them solve their problem. Customer Segments <ul style="list-style-type: none"> Group customers with common needs and attributes so that the enterprise can more effectively and efficiently address the needs of each segment. An organization within an enterprise may consider defining and targeting distinct customer. Customer Relationships <ul style="list-style-type: none"> Are classified as customer acquisition and customer retention. Depending on the level of interaction desired and the method of communication the methods used in establishing and maintaining customer relationships vary. Channels <ul style="list-style-type: none"> Are the different ways an enterprise interacts with and delivers value to its customers. <ul style="list-style-type: none"> Communication-oriented – e.g. marketing channel Delivery-oriented – e.g. distribution channel Others, such as sales channels and partnering channels Understanding channels involves identifying the processes, procedures, technologies, inputs, and outputs (and their current impact), as well as understanding the relationship of the various channels to the strategies of the organization. Key Resources <ul style="list-style-type: none"> Are the assets needed to execute a business model. Can be classified as: <ul style="list-style-type: none"> Physical: applications, locations, and machines Financial: what is needed to fund a business model, such as cash and lines of credit Intellectual: any proprietary aspects that enable a

	<ul style="list-style-type: none"> business model to thrive, such as knowledge, patents and copyrights, customer databases, and branding • Human: the people needed to execute a particular business model <ul style="list-style-type: none"> • Key Activities <ul style="list-style-type: none"> • Are those that are critical to the creation, delivery, and maintenance of value, as well as other activities that support the operation of the enterprise. • Can be classified as: <ul style="list-style-type: none"> • Value-add • Non-value-add • Business non-value-add • Cost Structure <ul style="list-style-type: none"> • Enterprises seek to reduce, minimize, or eliminate costs wherever possible. • Reducing costs may increase the profitability of an organization and allow those funds to be used in other ways to create value for the organization and for customers. • Revenue Streams <ul style="list-style-type: none"> • Is a way or method by which revenue comes into an enterprise from each customer segment in exchange for the realization of a value proposition. • Two basic ways of generating revenue: <ul style="list-style-type: none"> • Revenue resulting from a onetime purchase of a good or service • Recurring revenue from periodic payments for a good, service, or ongoing support • Some types of revenue streams include: <ul style="list-style-type: none"> • Licensing or Subscription fees • Transaction or Usage fees • Sales • Lending, Renting, or Leasing • Key Partnerships <ul style="list-style-type: none"> • Frequently involve some degree of sharing of proprietary information, including technologies. • Can, in some cases, lead to more formalized relationships such as mergers and acquisitions. • The benefits in engaging in key partnerships include: <ul style="list-style-type: none"> • Optimization and economy • Reduction of risk and uncertainty • Acquisition of particular resources and activities • Lack of internal capabilities
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • It is a widely used and effective framework that can be used to understand and optimize business models. • It is simple to use and easy to understand. • Limitations <ul style="list-style-type: none"> • Does not account for alternative measures of value such as social and environmental impacts. • The primary focus on value propositions does not provide a holistic insight for business strategy. • Does not include the strategic purpose of the enterprise.

10.9 Business Rules Analysis	
Purpose	To identify, express, validate, refine, and organize the rules that shape day-to-day business behaviour and guide operational business decision making.
Description	<ul style="list-style-type: none"> • Business policies and rules guide the day-to-day operation of the business and its processes, and shape operational business decisions. • A business policy is a directive concerned with broadly controlling, influencing, or regulating the actions of an enterprise and the people in it. • A business rule: <ul style="list-style-type: none"> • Is a specific, testable directive that serves as a criterion for guiding behaviour, shaping judgments, or making decisions • Must be practicable and is always under the control of the business • Basic principles for business rules include: <ul style="list-style-type: none"> • Basing them on standard business vocabulary to enable domain subject matter experts to validate them • Expressing them separately from how they will be enforced • Defining them at the atomic level and in declarative format • Separating them from processes they support or constrain • Mapping them to decisions the rule supports or constrains • Maintaining them in a manner such that they can be monitored and adapted as business circumstances evolve over time
Elements	<ul style="list-style-type: none"> • Definitional Rules <ul style="list-style-type: none"> • Shape concepts, or produce knowledge or information. • Indicate something that is necessarily true (or untrue) about some concept, thereby supplementing its definition. • Represent operational knowledge of the organization. • Cannot be violated but they can be misapplied. • Often prescribe how information may be derived, inferred or calculated based on information available to the business. • An example of a definitional rule is: <ul style="list-style-type: none"> • A customer must be considered a Preferred Customer if they place more than 10 orders per month • An example of a calculation rule is: <ul style="list-style-type: none"> • An order's local jurisdiction tax amount must be calculated as (sum of the prices of all the order's taxable ordered items) × local jurisdiction tax rate amount • Behavioural Rules <ul style="list-style-type: none"> • Are people rules—even if the behaviour is automated. • Serve to shape (govern) day-to-day business activity, by placing some obligation or prohibition on conduct, action, practice, or procedure. • Are rules the organization chooses to enforce as a matter of policy, often to reduce risk or enhance productivity. • Frequently make use of the information or knowledge produced by definitional rules (which are about shaping knowledge or information). • Are intended to guide the actions of people working within the organization, or people who interact with it. • May oblige individuals to perform actions in a certain way, prevent them from carrying out actions, or prescribe the conditions under which something can be correctly done. • Are rules that can be violated directly. • Various levels of enforcement may be specified for a behavioural rule. For example:

	<ul style="list-style-type: none"> Allow no violations (strictly enforced) Override by authorized actor Override with explanation No active enforcement An example of a behavioural rule is: <i>An order must not be placed when the billing address provided by the customer does not match the address on file with the credit card provider.</i>
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> When enforced and managed by a single enterprise-wide engine, changes to business rules can be implemented quickly. A centralized repository creates the ability to reuse business rules across an organization. Business rules provide structure to govern business behaviours. Clearly defining and managing business rules allows organizations to make changes to policy without altering processes or systems. Limitations <ul style="list-style-type: none"> Organizations may produce lengthy lists of ambiguous business rules. Business rules can contradict one another or produce unanticipated results when combined unless validated against one another. If available vocabulary is insufficiently rich, not business-friendly, or poorly defined and organized, resulting business rules will be inaccurate or contradictory.

10.10 Collaborative Games

Purpose	Collaborative games encourage participants in an elicitation activity to collaborate in building a joint understanding of a problem or a solution.
Description	<ul style="list-style-type: none"> Refer to several structured techniques inspired by game play and are designed to facilitate collaboration. Are used to help the participants share their knowledge and experience on a given topic, identify hidden assumptions, and explore that knowledge in ways that may not occur during the course of normal interactions. Can be used to understand the perspectives of various stakeholder groups. Often benefit from the involvement of a neutral facilitator who helps the participants understand the rules of the game and enforces those rules. Usually involve a strong visual or tactile element. Activities such as moving sticky notes, scribbling on whiteboards, or drawing pictures help people to overcome inhibitions, foster creative thinking, and think laterally.
Elements	<ul style="list-style-type: none"> Game Purpose <ul style="list-style-type: none"> Each collaborative game has a defined purpose — usually to develop a better understanding of a problem or to stimulate creative solutions. The facilitator helps the participants in the game understand the purpose and work toward the successful realization of that purpose. Process <ul style="list-style-type: none"> Games typically have at least three steps: <ol style="list-style-type: none"> An opening step, in which the participants get involved, learn the rules of the game, and start generating ideas. The exploration step, in which participants engage with one another and look for connections between their ideas, test those ideas, and experiment with new ideas. A closing step, in which the ideas are assessed and participants work out which ideas are likely to be the most

	<p>useful and productive.</p> <ul style="list-style-type: none"> Outcome <ul style="list-style-type: none"> At the end of a collaborative game, the facilitator and participants work through the results and determine any decisions or actions that need to be taken as a result of what the participants have learned. Example of Collaborative Games <table border="1"> <thead> <tr> <th>Game</th><th>Description</th><th>Objective</th></tr> </thead> <tbody> <tr> <td>Product Box</td><td>Participants construct a box for the product as if it was being sold in a retail store.</td><td>Used to help identify features of a product that help drive interest in the marketplace.</td></tr> <tr> <td>Affinity Map</td><td>Participants write down features on sticky notes, put them on a wall, and then move them closer to other features that appear similar in some way.</td><td>Used to help identify related or similar features or themes.</td></tr> <tr> <td>Fish-bowl</td><td>Participants are divided into two groups. One group of participants speaks about a topic, while the other group listens intently and documents their observations.</td><td>Used to identify hidden assumptions or perspectives.</td></tr> </tbody> </table> 	Game	Description	Objective	Product Box	Participants construct a box for the product as if it was being sold in a retail store.	Used to help identify features of a product that help drive interest in the marketplace.	Affinity Map	Participants write down features on sticky notes, put them on a wall, and then move them closer to other features that appear similar in some way.	Used to help identify related or similar features or themes.	Fish-bowl	Participants are divided into two groups. One group of participants speaks about a topic, while the other group listens intently and documents their observations.	Used to identify hidden assumptions or perspectives.
Game	Description	Objective											
Product Box	Participants construct a box for the product as if it was being sold in a retail store.	Used to help identify features of a product that help drive interest in the marketplace.											
Affinity Map	Participants write down features on sticky notes, put them on a wall, and then move them closer to other features that appear similar in some way.	Used to help identify related or similar features or themes.											
Fish-bowl	Participants are divided into two groups. One group of participants speaks about a topic, while the other group listens intently and documents their observations.	Used to identify hidden assumptions or perspectives.											
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> May reveal hidden assumptions or differences of opinion. Encourages creative thinking by stimulating alternative mental processes. Challenges participants who are normally quiet or reserved to take a more active role in team activities. Some collaborative games can be useful in exposing business needs that aren't being met. Limitations <ul style="list-style-type: none"> The playful nature of the games may be perceived as silly and make participants with reserved personalities or cultural norms uncomfortable. Games can be time-consuming and may be perceived as unproductive, especially if the objectives or outcomes are unclear. Group participation can lead to a false sense of confidence in the conclusions reached. 												
10.11 Concept Modeling													
Purpose	<p>Used to understand and convey the vocabulary of a domain.</p> <p>Facilitates consistent communication of the business knowledge among stakeholders</p>												
Description	<ul style="list-style-type: none"> Identifies core concepts of a domain and their essential relationships. Is invariant to designs and data models. Suggests standard wording for referencing concepts unambiguously. Conveys the meaning using statements of a natural language. 												

	<ul style="list-style-type: none"> Uses verb concepts and standard connections for representing relationships between noun concepts.
Elements	<p>Noun Concepts</p> <ul style="list-style-type: none"> Are the most basic concepts of the domain. Represent things of importance to the business. <p>Verb Concepts</p> <ul style="list-style-type: none"> Provide basic structural connections between noun concepts. Suggest domain-specific standard wordings for composing meaningful sentences. May be derived, inferred, or computed by definitional rules. <p>Other Connections</p> <ul style="list-style-type: none"> Represent universal ontological relationships. May include: categorizations, classifications, whole-part relationships, roles.
	<p>Example: Concept Map of Car Rental Company</p> <pre> graph TD Branch -- has --> Booking Branch -- issues --> Rental Branch -- "is start of" --> Rental Customer -- "may accumulate" --> ClubMembershipPoints Customer -- "may generate" --> BadExperiences Customer -- "may get" --> DiscountsUpgrades Customer -- "may be used for" --> Rental Booking -- concerns --> CarGroup Booking -- "may request a" --> CarModel Booking -- "of a" --> Rental Rental -- "has a" --> RentalStartDateTime Rental -- "has a" --> RentalEndDateTime Rental -- "accepts" --> ClubMembershipPoints Rental -- "is return point of" --> Booking Rental -- "describe organize" --> Car Rental -- "has a" --> Rate CarGroup -- has --> Rate CarGroup -- "organize" --> Car CarModel -- "organize" --> Car </pre>
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Provides a business-friendly way to convey domain knowledge. Is independent of data model designs and technologies. Facilitates understanding of knowledge-rich processes.

	<ul style="list-style-type: none"> Ensures consistent and unambiguous use of concepts. <p>Limitations</p> <ul style="list-style-type: none"> Requires elaborate abstract and ontological thinking skills. May be foreign to stakeholders and require explanation. May set incorrect expectations regarding further implementation. May need special tools to support real-time
--	--

10.12 Data Dictionary

Purpose	<ul style="list-style-type: none"> Used as a repository to define data elements. Serves stakeholders by providing a common interpretation of data elements. 								
Description	<ul style="list-style-type: none"> Describes primitive and composite data elements. Documents standard definitions, meanings and allowable values. Standardizes usage of key data elements. Ensures common understanding of data elements. Is often used in conjunction with Data Modeling technique. Can be maintained manually or via automated tools. 								
Elements	<ul style="list-style-type: none"> Data Elements <ul style="list-style-type: none"> Define primitive data elements and how they can be combined into composite data elements. Describe essential characteristics of each data element. Include standard definitions, meanings, and allowable values of data elements. Primitive Data Elements <ul style="list-style-type: none"> Unique Name for the data element Aliases or alternate names Acceptable Values/Meanings Composite Elements <ul style="list-style-type: none"> Are used to define composite structures using primitive and composite elements. May include: <ul style="list-style-type: none"> Sequences – specify a required order of data elements, e.g. using “+” Example: Customer Name = First Name+Middle Name+Family Name Repetitions – one or more data elements repeated multiple times Example: Customer Name = First Name + Middle Name+ (Family Name) Optional Elements – may or may not occur in a particular instance of the composite element Example: Customer Name = First Name+[Middle Name]+Family Name Example <table border="1"> <thead> <tr> <th>Primitive Data Elements</th> <th>Data Element 1</th> <th>Data Element 2</th> <th>Data Element 3</th> </tr> </thead> <tbody> <tr> <td>Name (referenced by data elements)</td> <td>First Name</td> <td>Middle Name</td> <td>Last name</td> </tr> </tbody> </table> 	Primitive Data Elements	Data Element 1	Data Element 2	Data Element 3	Name (referenced by data elements)	First Name	Middle Name	Last name
Primitive Data Elements	Data Element 1	Data Element 2	Data Element 3						
Name (referenced by data elements)	First Name	Middle Name	Last name						

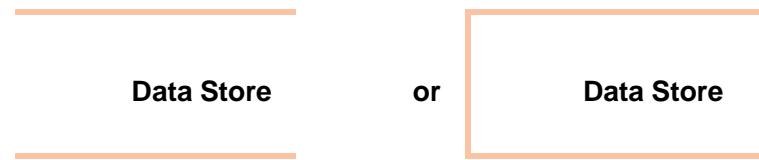
		Alias (referenced by stakeholders)	Given Name	Middle Name	Surname	
		Values/Meanings (enumerated list or description of data element)	Minimum 2 characters	Can be omitted	Minimum 2 characters	
		Composite	Customer Name = First Name + Middle Name + Last Name			

Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Provides shared understanding of the format and content of data. Promotes consistent use of data throughout the organization. Limitations <ul style="list-style-type: none"> May become obsolete or incorrect without regular maintenance. Updates must be completed in a consistent manner. Maintaining accurate data dictionary requires time and effort. May have limited value if not used in multiple scenarios across the enterprise
----------------------	--

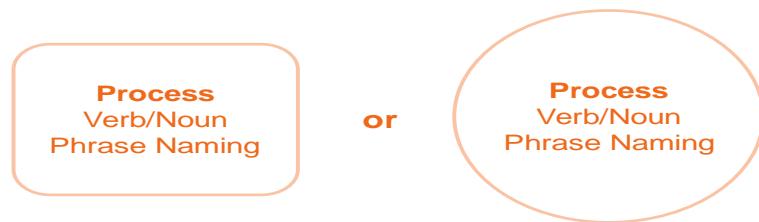
10.13 Data Flow Diagrams

Purpose	<ul style="list-style-type: none"> Used to illustrate movement and transformation of data in a system. Help in understanding and explaining transactions and boundaries of the system.
Description	<ul style="list-style-type: none"> Display flow of information between data stores, processes, and external entities of a system The data defined should be described in a Data Dictionary Can represent the system on various levels of abstraction Can be decomposed as needed May depict either logical or physical flows of data Can show either the current state of a system or its target future state
Elements	<ul style="list-style-type: none"> Externals (Entity, Source, Sink) <ul style="list-style-type: none"> An external entity is a person, organization, automated system, or any device capable of producing or receiving data. Serve as sources and/or destinations (sinks) of data. Are represented by using a noun inside a rectangle. Must have at least one incoming or outgoing data flow.  Data Store <ul style="list-style-type: none"> A collection of data or a physical location where data can be stored and repeatedly read.

- Is represented as either two parallel lines or as an open-ended rectangle with a label.
- Must have at least one incoming or outgoing data flow.



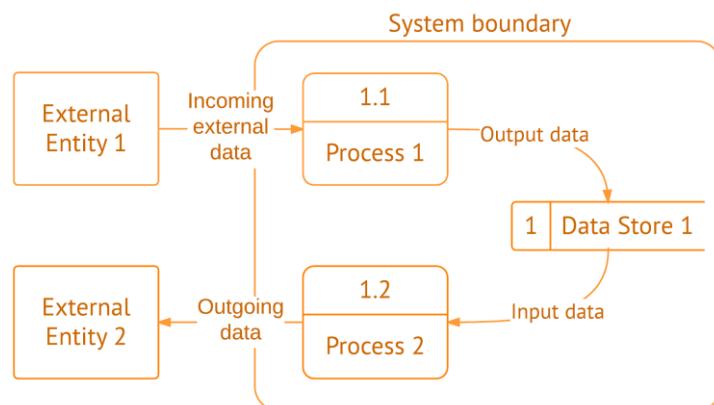
- **Process**
 - Transforms the input data into an output.
 - Can be a manual or automated activity performed for a business reason
 - Is represented as a circle or rectangle with rounded corners.
 - Naming for a process should contain a verb and a noun.
 - Each process must have at least one data flow going to it and one data flow coming from it.



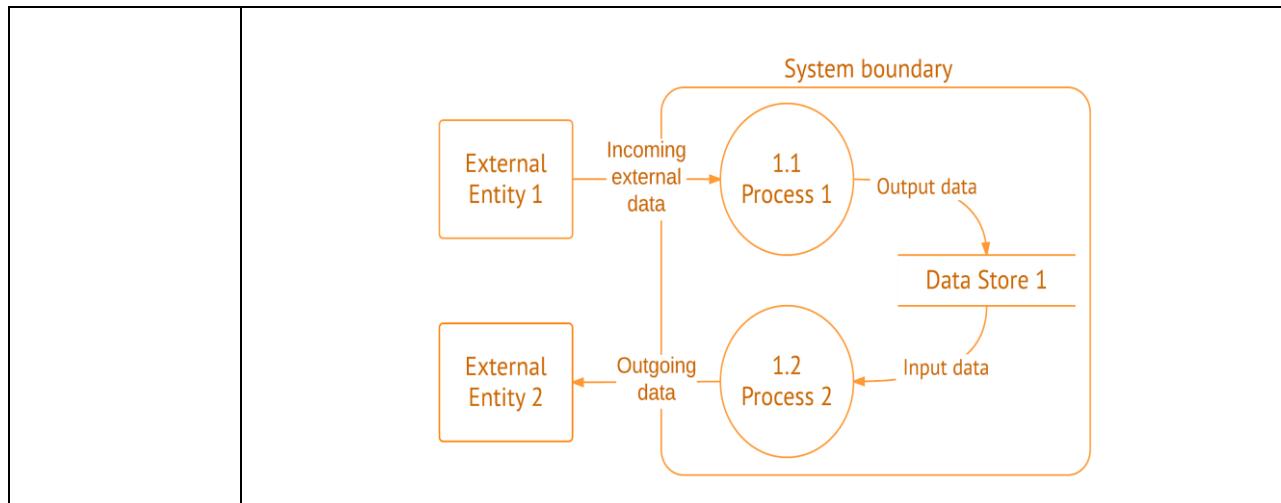
- **Data Flow**
 - Depicts a transfer of data between processes, data stores, and external entities.
 - Is represented as a line with an arrow.
 - Is named using one or more nouns or noun clauses.



- **Example - Data Flow Diagram Gane Sarson Notation**



- **Example - Data Flow Diagram Gane Sarson Notation**



Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> May be used for discovering processes and data. May be used for verifying functional decompositions or data models. Facilitate estimation of the effort needed to do the work. Are relatively easy to understand. Help to identify duplicated or misapplied data elements. Help define the boundaries of a system. Illustrate connections to other systems. Can be used as part of the system documentation. Help to explain the logic behind the data flow within a system. Limitations <ul style="list-style-type: none"> Data flow diagrams for large-scale systems can become complex and difficult for stakeholders to understand. Different notations with different symbols could create challenges pertaining to documentation. Does not illustrate a sequence of activities. Data transformations say little about the process or stakeholder.
-----------------------------	--

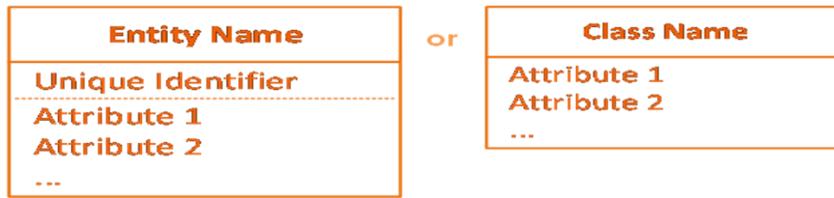
10.14 Data Mining

Purpose	<ul style="list-style-type: none"> Used to discover useful patterns and insights from data. Supports decision making.
Description	<ul style="list-style-type: none"> Examines data for verifying (top-down) or generating (bottom-up) hypotheses about existing patterns and relationships. Can be used to answer existing questions or to get new ideas. Typically presents results in a form of mathematical models or equations. Can be used to support human or automated decision making. Can be descriptive, diagnostic, or predictive. The goals of analysis define the right type, volume, and quality of data to be analyzed, as well as the methods of analysis.
Elements	<ul style="list-style-type: none"> Requirements Elicitation <ul style="list-style-type: none"> The goals and scope of data mining are defined in terms of either decisions to be supported or a functional area undergoing analysis. Top-down data mining can make use of decision modelling techniques (see 10.17 Decision Modelling). Bottom-up pattern discovery can be related to existing decision models. Best to be conducted as an iterative approach.

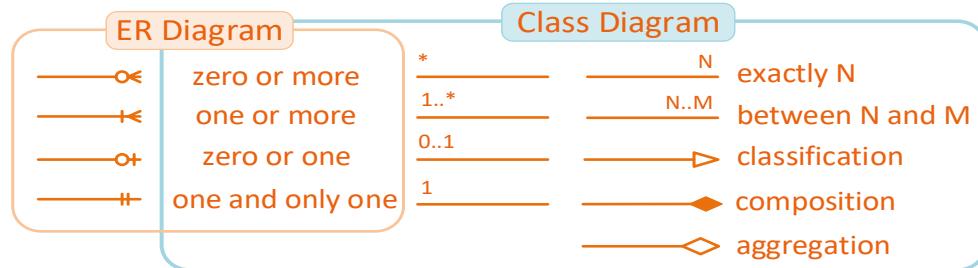
	<ul style="list-style-type: none"> • Data Preparation: Analytical Dataset <ul style="list-style-type: none"> • Typically involves merging records from multiple sources into a single dataset. • Analytical data can be split into multiple sub-sets for the purposes of developing, testing, and validating analytical models. • Data Analysis <ul style="list-style-type: none"> • Typically applies statistical methods and measures. • Makes use of visualization tools for understanding essential characteristics of data. • Modeling Techniques <ul style="list-style-type: none"> • Classification and regression trees (CART), C5 and other decision tree analysis techniques. • Linear and logistic regression. • Neural networks. • Support vector machines. • Predictive (additive) scorecards • Deployment <ul style="list-style-type: none"> • The deployment may target to support either a human decision maker or an automated decision-making system. • Data mining results can be deployed as complimentary data and/or executable business rules. • Involves making decisions and planning effort on integration with existing infrastructure of the organization.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Reveals hidden patterns and creates useful insight. • Can be integrated into a system design. • Can eliminate or reduce human bias in making decisions. • Limitations <ul style="list-style-type: none"> • Misapplication of some techniques may result in erroneous decisions. • Use of techniques and tools requires advanced specialist knowledge. • Due to the need for advanced background in math, stakeholders may have difficulties with understanding and using the results. • Data mining results may be hard to deploy.

10.15 Data Modeling

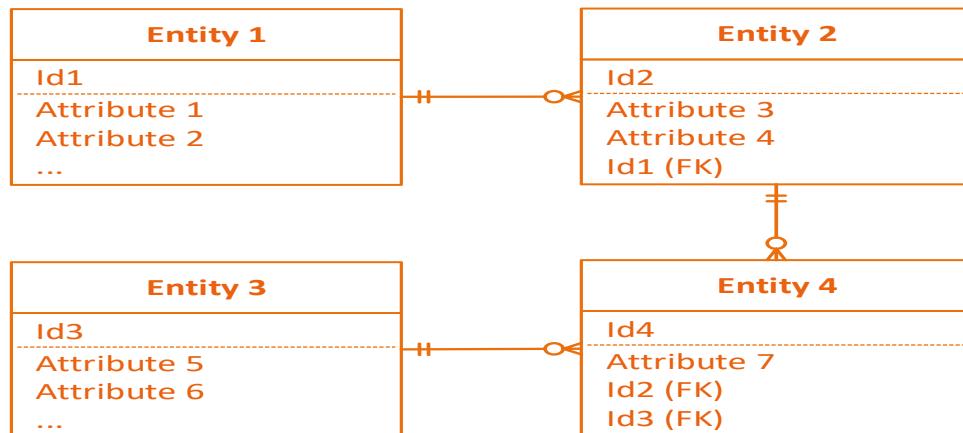
Purpose	<ul style="list-style-type: none"> • Used to provide a common understanding of data for analysis and implementation. • Supports elicitation, requirements analysis, and design.
Description	<ul style="list-style-type: none"> • Represent information elements, their attributes and relationships that are important to the business. • Can be conceptual, logical, or physical. • Depending on the purpose may be significantly different even when depicting the same domain. • Can be visually represented as an entity-relationship diagram or a class diagram.
Elements	<ul style="list-style-type: none"> • Entity or Class <ul style="list-style-type: none"> • May represent physical, organizational, or abstract thing or phenomenon. • Contains attributes that define or describe the entity. • In a class diagram is referred as “class”. • Can be related to other entities.



- **Attribute**
 - Defines a particular piece of information associated with an entity.
 - Can include a name, description, type of information it represents, and allowable values.
 - Can be described in a data dictionary.
 - Allowable values may be specified as business rules.
- **Relationship or Association**
 - Indicates which entities and how relate to each other.
 - Indicates cardinality on each side of the relationship.
 - Relationships are referred to as "associations" in a class diagram.
 - Class diagrams use term "multiplicity" instead of "cardinality".
 - Class diagrams enrich the modeling vocabulary with additional types of relationships.



- **Diagram – Entity Relationship Diagram (ERD)**



- **Diagram – Class Diagram**

	<pre> classDiagram class Class1 { Attribute1 Attribute2 } class Class5 { Attribute8 } class Class2 { Attribute3 Attribute4 } class Class3 { Attribute5 Attribute6 } class Class4 { Attribute7 } Class1 < -- Class5 Class1 < -- Class2 Class1 < -- Class3 Class5 "1..*--> Class2 Class3 "1..*--> Class4 </pre>
	<ul style="list-style-type: none"> • Metadata <ul style="list-style-type: none"> • Provides optional explanations and guidance regarding the model. • May explain what the entities represent. • May suggest when and how the entities are to be used. • May specify rules and constraints that apply to entities and relationships. • May add security, privacy and auditing considerations.

Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Provides a common vocabulary to be used by domain subject matter experts and implementation subject matter experts. • Helps to ensure that the design of data correctly represents the business need. • Provides a consistent approach to analyzing and documenting data and its relationships. • Allows choosing levels of detail appropriate for the target audience. • Helps in validating and discovering new requirements. • Limitations <ul style="list-style-type: none"> • May require special background or learning to be understood. • May extend beyond the business knowledge of each individual stakeholder.
-----------------------------	---

10.16 Data Analysis

Purpose	<ul style="list-style-type: none"> • Used to explore possible consequences of different decisions. • Helps to determine the value of alternate outcomes under conditions of uncertainty or in highly complex situations.
Description	<ul style="list-style-type: none"> • A decision is the act of making a choice among multiple viable alternatives with different values of their outcomes. • The evaluation of an outcome may take a form of financial value, scoring, or a relative ranking. • There are a number of decision analysis tools available to assist in Decision Analysis. • The appropriate approach depends on the type of decision, level of uncertainty, risk, quality of information, and evaluation criteria. • Decision analysis requires an understanding of: <ul style="list-style-type: none"> • The values, goals, and objectives related to the decision problem • The nature of the decision that must be made • The areas of uncertainty that affect the decision • The consequences of each potential decision

	<ul style="list-style-type: none"> Decision analysis activities include: <ul style="list-style-type: none"> Define problem statement Define alternatives Evaluate alternatives Choose the alternative to implement Implement the chosen alternative Examples of decision analysis tools and techniques: <ul style="list-style-type: none"> Pro versus con considerations Force field analysis Decision tables Decision trees Comparison analysis Analytical hierarchy process (AHP) Totally-partially-not (TPN) Multi-criteria decision analysis (MCDA) Computer-based simulations and algorithms 																																													
Elements	<ul style="list-style-type: none"> Components of Decision Analysis <ul style="list-style-type: none"> Description of the decision to be made or a Problem Statement. Decision Maker: the person or people responsible for making the final decision. Alternative: a possible proposition or course of action. Decision criteria used to evaluate the alternatives. Decision Matrices <ul style="list-style-type: none"> Can be simple or weighted. A simple decision matrix totals the number of evaluation criteria matched for each alternative. A weighted decision matrix computes the values of alternatives by totaling up ranks of evaluation criteria multiplied by factors of their importance. Decision Matrices <ul style="list-style-type: none"> Simple decision matrix <table border="1"> <tr> <td><input checked="" type="checkbox"/> Calculate Base Price</td> <td colspan="4"></td> </tr> <tr> <td><input type="checkbox"/> (auditorium_no, seat_no)</td> <td colspan="4"></td> </tr> <tr> <td>U</td> <td>auditorium_no</td> <td>seat_no</td> <td>special_seating</td> <td>price</td> </tr> <tr> <td></td> <td></td> <td></td> <td>true,false</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td>[1..200]</td> <td>true</td> <td>7</td> </tr> <tr> <td>2</td> <td>1</td> <td>[201..600]</td> <td>false</td> <td>5</td> </tr> <tr> <td>3</td> <td>2</td> <td>[1..100]</td> <td>true</td> <td>6</td> </tr> <tr> <td>4</td> <td>2</td> <td>[101..400]</td> <td>false</td> <td>5</td> </tr> <tr> <td>5</td> <td>3</td> <td>[0..999]</td> <td>false</td> <td>5</td> </tr> </table> Decision Matrices <ul style="list-style-type: none"> Weighted decision matrix 	<input checked="" type="checkbox"/> Calculate Base Price					<input type="checkbox"/> (auditorium_no, seat_no)					U	auditorium_no	seat_no	special_seating	price				true,false		1	1	[1..200]	true	7	2	1	[201..600]	false	5	3	2	[1..100]	true	6	4	2	[101..400]	false	5	5	3	[0..999]	false	5
<input checked="" type="checkbox"/> Calculate Base Price																																														
<input type="checkbox"/> (auditorium_no, seat_no)																																														
U	auditorium_no	seat_no	special_seating	price																																										
			true,false																																											
1	1	[1..200]	true	7																																										
2	1	[201..600]	false	5																																										
3	2	[1..100]	true	6																																										
4	2	[101..400]	false	5																																										
5	3	[0..999]	false	5																																										

				Alternative 1		Alternative 2		Alternative 3	
		Weight	Rank	Value	Rank	Value	Rank	Value	
Risk	1	3	3	5	5	2	2		
Profit	1	5	5	4	4	3	3		
Cost	3	5	15	1	3	5	15		
Time to Market	5	1	5	5	25	3	15		
Score			28		27		35		

- **Decision Trees**
 - Are used for assessing the preferred outcome where multiple sources of uncertainty exist.
 - Allow for assessment of responses to uncertainty to be factored across multiple strategies.
 - May include decision nodes, chance nodes, and terminator or end nodes.
 - Are also used in Decision Modelling.
- **Trade-offs**
 - Are relevant when a decision problem involves multiple, possibly conflicting, objectives.
 - For more than one objective it is not sufficient to simply find the maximum value for single variable.
 - Methods for handling trade-offs include elimination of dominated alternatives and ranking objectives on a similar scale.

Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Provides a prescriptive approach for determining alternate options. • Helps to eliminate subjective biases in making decisions. • Helps to avoid false assumptions. • Encourages constructing appropriate metrics for comparing both the financial and non-financial outcome evaluation criteria. • Limitations <ul style="list-style-type: none"> • Due to the effort required and the lack of necessary information, may not suit decisions that must be made immediately. • The results may be perceived as more certain than they are. • Analysis paralysis can occur. • May require specialized knowledge, such as math knowledge in probability and strong skills with decision analysis tools.
10.17 Decision Modeling	
Purpose	<ul style="list-style-type: none"> • Used to show how data and knowledge are combined to make repeatable business decisions. • Helps stakeholders to understand as-is and to-be decision making models. • Can be used in conjunction with business process models to elaborate on activities that involve decision making.

Description	<ul style="list-style-type: none"> Distinguishes straightforward and complex decisions. Represents definitional and behavior business rules. Makes use of decision tables, decision trees, and decision model diagrams. May depict where information and business rules come from. Can be related to business processes and organizations. Can represent decisions at various levels of detail. 																					
Elements	<ul style="list-style-type: none"> Types of Models and Notations <ul style="list-style-type: none"> Decision models use three common key elements: decision, information, and knowledge. May also include the sources of knowledge and data. Three typical notations are Decision tables, Decision trees, and Decision requirements diagrams. The choice depends on the nature and complexity of the decision being considered. Can be combined for elaborating on a complex set of interrelated decisions. Decision Tables <ul style="list-style-type: none"> Are most useful when the decision follows from a combination of varying values of a limited homogenous set of parameters. Represent all the rules required to make an atomic decision in a tabular form. Each row represents a unique viable combination of decision parameter values, which leads to a particular decision. Columns correspond to decision parameters. The last column represents final outcomes. 																					
	<table border="1" data-bbox="417 1062 1302 1600"> <thead> <tr> <th colspan="3">Eligibility rules</th> </tr> <tr> <th>Loan Amount</th> <th>Age</th> <th>Eligibility</th> </tr> </thead> <tbody> <tr> <td rowspan="2"><=1000</td> <td><18</td> <td>Ineligible</td> </tr> <tr> <td>>=18</td> <td>Eligible</td> </tr> <tr> <td rowspan="2">1000-2000</td> <td><21</td> <td>Ineligible</td> </tr> <tr> <td>>=21</td> <td>Eligible</td> </tr> <tr> <td rowspan="2">>2000</td> <td><25</td> <td>Ineligible</td> </tr> <tr> <td>>=25</td> <td>Eligible</td> </tr> </tbody> </table> <ul style="list-style-type: none"> Decision Trees <ul style="list-style-type: none"> Are most useful when each decision is comprised by a sequential series of smaller decisions that involve heterogeneous parameters. Represent all the rules required to make an atomic decision in the form of a directed graph. The graph is composed of nodes and branches. May distinguish decision nodes, chance nodes, and end nodes. Branches represent viable alternatives. 	Eligibility rules			Loan Amount	Age	Eligibility	<=1000	<18	Ineligible	>=18	Eligible	1000-2000	<21	Ineligible	>=21	Eligible	>2000	<25	Ineligible	>=25	Eligible
Eligibility rules																						
Loan Amount	Age	Eligibility																				
<=1000	<18	Ineligible																				
	>=18	Eligible																				
1000-2000	<21	Ineligible																				
	>=21	Eligible																				
>2000	<25	Ineligible																				
	>=25	Eligible																				

	<pre> graph TD Age[Age] --> <18 Ineligible1[Ineligible] Age --> 18-21 Amount1[Amount] Age --> >21 Amount2[Amount] Amount1 --> >\$1000 Ineligible2[Ineligible] Amount1 --> <=\$1000 Eligible1[Eligible] Amount2 --> <=\$2000 Eligible2[Eligible] Amount2 --> <=\$5000 Income[Income] Amount2 --> >\$5000 Ineligible3[Ineligible] Income --> >=\$30000 Eligible4[Eligible] Income --> <\$30000 Ineligible5[Ineligible] </pre> <ul style="list-style-type: none"> Decision Requirements Diagrams <ul style="list-style-type: none"> Contain decisions, input data, business knowledge, and knowledge sources. The network comprised of nodes and links shows the decomposition of a complex decision into simpler building blocks. Solid arrows show the information requirements for a decision and information. The source of information can be an existing data or another decision. The dashed links with a round ending can represent a use of knowledge, a source of the knowledge, or an authority for the decision. <pre> graph TD KS1[Knowledge Source] -.-> BK1[Business Knowledge] KS2[Knowledge Source] -.-> BK2[Business Knowledge] DM1[Decision Maker 1] -.-> BK1 DM2[Decision Maker 2] -.-> BK2 BK1 -.-> D1[Decision 1] BK2 -.-> D1 InputData1[Input Data] -.-> D1 InputData2[Input Data] -.-> D2[Decision 2] </pre>
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Facilitates understanding and communication of complex decision rules. Supports impact analysis. Can combine and share multiple perspectives. Simplifies complex decision making. Assists with managing and reusing large number of rules. Can be used for rule-based automation, data mining, predictive analytics, human decisions, and business intelligence projects. Limitations

	<ul style="list-style-type: none"> • Adds yet another diagram style that may require learning. • May add unnecessary complexity for simple decisions. • May limit rules to those required by known decisions, not covering the whole range of real-life situations. • Does not support flexible ad-hoc decisions. • May lock an organization into a current-state decision making approach. • May induce an illusion that an organization has a standard way of making decisions when it does not. • Can be difficult to acquire necessary sign-off when cross-cuts organizational boundaries. • Requires a clearly defined and shared business terminology.
--	--

10.18 Document Analysis

Purpose	<ul style="list-style-type: none"> • Used to elicit business analysis information, including contextual understanding and requirements. • Helps to ensure that the need is fully understood in terms of the environment in which it exists. • May be used to validate findings from other elicitation techniques such as interviews and observations. • Helps to understand how the current solution works and why it was implemented in its current form.
Description	<ul style="list-style-type: none"> • Is done by examining available materials that describe either the business environment or existing organizational assets. • Materials may originate from public or proprietary sources. • Materials of interest include business rules, technical documentation, training documentation, problem reports, previous requirements documents, and procedure manuals. • May include marketing studies, industry guidelines or standards, company memos, and organizational charts. • The purpose and type of the business analysis information being explored define topics and scope of documents to be researched. • May take a form of data mining. • The results of analysis should be recorded within a work product.
Elements	<ul style="list-style-type: none"> • Preparation <ul style="list-style-type: none"> • Involves identifying and assessing materials to be analyzed. • Materials are assessed for relevance, accuracy, and credibility. • May take into consideration comprehensibility of the materials for the target stakeholder audience. • Includes identifying both the data to be mined and the logical grouping of the data. • Document Review and Analysis <ul style="list-style-type: none"> • Conducting a detailed review of each document's content. • Identifying omissions and conflicts in the information being explored. • Identifying additional topics and materials that may need to be explored. • Recording relevant notes associated with each topic. • Notes can include the topic, type, source, verbatim details, a paraphrased critique, follow-up issues, questions, and actions for each document that is reviewed. • Record Findings <ul style="list-style-type: none"> • Use of the information elicited through document analysis for composing a work product to be shared with other stakeholders.

	<ul style="list-style-type: none"> Includes deciding on the content and level of detail that is appropriate for the intended audience. May involve transforming the materials into visual aids to facilitate understanding by the intended audience.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Existing materials can serve as a readily available basis for business analysis. Existing sources, although possibly outdated, can be used as a point of reference for other elicitation techniques. Can be used to validate the results of other elicitation techniques. The findings can be easily reviewed and confirmed. Can be used to close information gaps when subject matter experts are not available. Provides high level of objectivity of the business analysis results. Limitations <ul style="list-style-type: none"> Existing documentation may be outdated or invalid. Authors may be unavailable for answering questions. Primarily helpful only for evaluating the current state via review of the as-is documentation. May be very time-consuming and lead to information overload and confusion.

10.19 Estimation

Purpose	<ul style="list-style-type: none"> Is used to forecast the cost and effort involved in pursuing a course of action. Is used to support decision making.
Description	<ul style="list-style-type: none"> Predicts uncertain values of important attributes based on knowledge and past experience. Can be expressed as a single number or a probable range of values. May include an assessment of the level of accuracy and uncertainty associated with the forecast. May involve revising the forecasts as more information becomes available.
Elements	<ul style="list-style-type: none"> Methods <ul style="list-style-type: none"> Are selected according to the purpose and object of estimation. Top-down: starts at the highest level in a hierarchical breakdown. Bottom-up: aggregates estimates up from the lowest-level elements of a hierarchical breakdown. Parametric Estimation: uses a calibrated parametric model of the element attributes being estimated. Rough Order of Magnitude (ROM): a high-level estimate, based on common sense and meaningful analogies. Rolling Wave: extrapolates actually achieved performance for the remainder of the initiative or project. Delphi: uses a combination of expert judgment and history. PERT: computes the final estimate as a weighted average of its optimistic, pessimistic and most likely value. Accuracy of the Estimate <ul style="list-style-type: none"> Is a measure of uncertainty that evaluates how close an estimate is to the actual value. Can be calculated as a ratio of the width of the confidence interval to its mean value. Can be expressed as a percentage. ROM estimates may be no more than +50% to -50% accurate. Definitive estimates should be accurate within 10% or less.

	<ul style="list-style-type: none"> ROM estimates and definitive estimates can be combined using rolling wave estimates. Sources of Information <ul style="list-style-type: none"> Are selected based on what is available and what is being estimated. Analogous Situations: uses known information about an element that is similar to the element being estimated. Organization History: relies on previous experiences of the organization with similar work. Expert Judgment: leverages the knowledge of individuals who have performed similar work in the past. Precision and Reliability of Estimates <ul style="list-style-type: none"> The precision is a measure of agreement between multiple estimates made for the same attribute. Can be done by examining statistical measures of imprecision such as variance or standard deviation. The reliability of an estimate is a measure of its repeatability through different methods or by different estimators. Estimation of the same attribute can be performed using multiple alternative methods for assessing its reliability. Contributors to Estimates <ul style="list-style-type: none"> The estimators of an element are frequently those responsible for that element. The estimate of a team is usually more accurate than the estimate of one individual. A dedicated group can be assigned for performing various estimations within an organization. Organizations may involve external experts to perform or to review the estimation. An organization may compare its internal estimates against independent estimates for making adjustments to its methods of estimation.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Estimates provide a rationale for making decisions regarding the budget, scope, or time frame. Without an estimate, teams making a change may be provided an unrealistic budget or schedule for their work. Systematic use of defined techniques generally results in a closer prediction of the actual value than an ad-hoc "intuitive" estimation. Updating an estimate throughout a work cycle allows accumulating knowledge and ensuring success. Limitations <ul style="list-style-type: none"> Estimates are only as accurate as the level of knowledge about the elements being estimated. Using just one estimation method may lead stakeholders to form unrealistic expectations.
10.20 Financial Analysis	
Purpose	<ul style="list-style-type: none"> Is used to understand the financial aspects of an investment, a solution, or a solution approach. Can be used to make a solution recommendation by comparing financial benefits of available alternatives.
Description	<ul style="list-style-type: none"> Assesses expected financial viability, stability, and benefit realization of an investment option.

	<ul style="list-style-type: none"> Takes into consideration the cost of the change as well as the total costs and benefits of using and supporting the solution. Deals with uncertainty throughout the initiative to determine the likelihood of the change to deliver a justified business value. May provide a rationale for recommending to continue, adjust or stop the initiative.
Elements	<ul style="list-style-type: none"> Cost of the Change <ul style="list-style-type: none"> Cost of Change = cost to acquire + cost to deploy May include the costs associated with changing equipment and software, facilities, staff and other resources, buying out existing contracts, subsidies, penalties, converting data, training, communicating the change, and managing the roll out. The costs may be shared between organizations within the enterprise. Total Cost of Ownership (TOC) <ul style="list-style-type: none"> TCO = cost to acquire + cost of use + cost of support Cost of use may include cost of depreciation based on the life expectancy of the solution. When the solution's life expectancy is unknown, organizations may assume a standard period of time, such as three or five years. Value Realization <ul style="list-style-type: none"> Value is typically realized over time. The planned value could be expressed on an annual basis, or as a cumulative value over a specific time period. Cost-Benefit Analysis <ul style="list-style-type: none"> Is a prediction of the expected financial benefit. Net benefit = total benefits - total costs Includes statement of assumptions about the factors that make up the costs and benefits. The time period of a cost-benefit analysis should be long enough so that the planned value is being realized. As the expected costs become actual, the cost-benefit analysis may be re-examined to determine if the solution is still viable. Financial Calculations <ul style="list-style-type: none"> Are performed to understand different perspectives about when and how the investment will deliver value. May take into consideration: <ul style="list-style-type: none"> Inherent risks The amount of upfront money to be invested The amount of time it will take to recoup the original investment The amount of time it will take to fully realize benefits of the solution A comparison to other investments the organization could make

	<ul style="list-style-type: none"> • Return on Investment (ROI) <ul style="list-style-type: none"> • Is used to compare the relative returns of different investment opportunities. • $ROI = \frac{\text{Total Benefits} - \text{Investment Cost}}{\text{Investment Cost}}$ • Discount Rate <ul style="list-style-type: none"> • Is the assumed interest rate that could be earned from the most reliable investment opportunity. • Typically is slightly above the current level of inflation. • A larger discount rate can be used for more remote time periods to reflect greater uncertainty and risk. • Present Value (PV) <ul style="list-style-type: none"> • Is used to objectively compare different solutions that could realize benefits at different rates and over a different time. • $PV = \sum_{i=1}^n \frac{\text{Net Benefits in the period}}{(1 + \text{Discount Rate for the period})^i}$ • Present value does not consider the cost of the original investment. • Net Present Value (NPV) <ul style="list-style-type: none"> • $NPV = PV - \text{Cost of Investment}$ • Is used to compare different investment options and different benefit patterns in terms of their value at the present day. • Internal Rate of Return (IRR) <ul style="list-style-type: none"> • Is the interest rate at which the investment breaks even. • Is used to determine if the change, solution or solution approach is worth investing in. • May be compared to a minimum threshold that the organization expects to earn from its investments, such as the cost of funds. • Can be used for comparing investments of the same duration. • Does not consider external influencers such as inflation or fluctuating interest rates. • Can be calculated using numeric approximation algorithms. • Payback Period <ul style="list-style-type: none"> • Is a projection on the time period required to generate enough benefits to recover the cost of the change.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Allows executive decision makers to objectively compare very different investments from different perspectives. • Assumptions and estimates built into the financial calculations, are clearly stated so that they may be challenged or approved. • Reduces the uncertainty of a change or solution by requiring the

	<p>identification and analysis of factors that may influence the investment.</p> <ul style="list-style-type: none"> Allows to objectively re-evaluate the recommended solution if the context, business need, or stakeholder needs change Limitations <ul style="list-style-type: none"> Some costs and benefits are difficult to quantify financially. Because financial analysis is forward looking, there will always be some uncertainty about expected costs and benefits. The errors in estimation of the parameters of calculations may accumulate into inaccurate results of analysis. Positive financial numbers may give a false sense of security while they may not provide all the information required to understand an initiative.
--	---

10.21 Focus Groups

Purpose	<ul style="list-style-type: none"> To elicit ideas and opinions about a product, service or opportunity. For a product under development, the resulting report may be used for requirements analysis. For a completed product, the results may influence placement in the marketplace. While customer satisfaction could be assessed for a product in production.
Description	<ul style="list-style-type: none"> A form of qualitative research. Consists of pre-qualified individual selected as participants. Participants share perspectives about a topic in a group setting. Structured session facilitated by a trained moderator, which may or may not be the Business Analyst. Session feedback captured is reported on as themes or perspectives.
Elements	<ul style="list-style-type: none"> Focus Group Objective <ul style="list-style-type: none"> Defined purpose and outcome of the group. Focus Group Plan <ul style="list-style-type: none"> Defines activities to plan, conduct, and report on the focus group <ul style="list-style-type: none"> Purpose Location Logistics Participants Budget Timelines Outcomes Participants <ul style="list-style-type: none"> Typically, 6 to 12 attendees. Demographics determined by objective of the focus group. Be willing to share insights & perspectives. Be willing to listen to other opinions. Discussion Guide <ul style="list-style-type: none"> Includes structure or framework to be followed by moderator. Prepared ahead of time & used by moderator to keep group on track. Reminder to welcome & introduce session as well as obtain comments. Assign a Moderator and Recorder <ul style="list-style-type: none"> The moderator helps guide the session and ensures the focus group remains on task. The recorder observes session and takes notes. The business analyst can fill either roll. Neither role is an active participant in the focus group and therefore should not provide feedback.

	<ul style="list-style-type: none"> • It is best to have separate people fill the rolls as it is difficult to facilitate a group and record accurate notes. • Conduct the Focus Group <ul style="list-style-type: none"> • Typically 1 to 2 hours & prepared in advanced, the session should appear to be free flowing to the participants, while the recorder captures the group's comments. • After the Focus Group <ul style="list-style-type: none"> • The feedback and observations are transcribed as soon as possible after the session has ended to ensure all possible insights and opinions are recorded as accurately as possible. • The BA analyzes the session results looking for trends, insights, and other information to prepare a summary of the outcome
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Saves time & cost to elicit data from a number of individuals at once. • Effective for learning people's attitudes, experiences, and desires. • Participants can consider personal views along with other perspectives. • Limitations <ul style="list-style-type: none"> • Concerns of trust and privacy may influence participants to limit their sharing. • What people say may be inconsistent with their behaviors. • A group may be too homogeneous (too similar). • Skilled, professional moderator needed. • One vocal participant could sway the results of the group.

10.22 Functional Decomposition

Purpose	<ul style="list-style-type: none"> • To break down processes, systems, functional areas, or deliverables into smaller, more manageable parts to more easily analyze. • Manage complexity. • Reduce uncertainty.
Description	<ul style="list-style-type: none"> • The process of analyzing systems or concepts by breaking down larger components into sub-components. • As sub-components, scaling, tracking and measuring the work effort is more manageable than as one huge component.
Elements	<ul style="list-style-type: none"> • Decomposition Objectives <ul style="list-style-type: none"> • Defines the process by which to break down components into smaller pieces. • Defines what to analyze and how deep to breakdown a component. • May include: <ul style="list-style-type: none"> • Measuring & Managing • Reusing • Designing • Optimization • Analyzing • Substitution • Estimating & Forecasting • Encapsulation • Subjects of Decomposition <ul style="list-style-type: none"> • Things that can be decomposed include:

	<ul style="list-style-type: none"> • Business Outcomes • Solution Component • Work to be Done • Activity • Business Processes • Product and Services • Function • Decisions • Business Unit • <p>• Level of Decomposition</p> <ul style="list-style-type: none"> • Defines where, why, and when to stop breaking down components. • The BA stops decomposing the subject at the level where he has just enough detail to proceed to the next task or analysis to be done <p>• Representation of Decomposition Results</p> <ul style="list-style-type: none"> • Can take the form of text, hierachal lists, and/or visual diagrams. • Diagramming techniques often used include: <ul style="list-style-type: none"> • Tree Diagrams • Cause-Effect Diagrams • Nested Diagrams • Decision Trees • Use Case Diagrams • Mind Maps • Flow Diagrams • Component Diagram • State Transition Diagrams • Decision Mode and Notation
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> • Complex & large undertakings can be separated into related but manageable parts • Provides structured approach to building a shared understanding of complex systems and processes • Simplifies measurement and estimation of the amount of work involved in a course of action Limitations <ul style="list-style-type: none"> • Revision of decomposition may be needed if missing or incorrect information used for initial breakdown of subject • Multiple variations of decompositions are possible with certain systems or components. Completing this exercise for each possibility may not be feasible. • Subject Matter Expert may be needed to gain level of knowledge needed to complete decomposition.

10.23 Glossary

Purpose	Defines key terms relevant to a business domain.
Description	A list of terms and established definitions provides a common language that can be used to communicate and exchange ideas
Elements	<ul style="list-style-type: none"> • A term is added to the glossary when: <ul style="list-style-type: none"> • It is unique to a domain • There are multiple definitions

	<ul style="list-style-type: none"> The definition implied is outside the term's common usage There is a reasonable chance of misunderstanding
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Promotes common understanding of the business domain. Provides a single reference and encourages consistency. Simplifies the writing and maintenance of other work products. Limitations <ul style="list-style-type: none"> Ongoing maintenance required so as not to become outdated. Agreeing on single definition across all stakeholders may prove difficult.

10.24 Interface Analysis

Purpose	Identifies where, what, why, when, how, and for whom information is exchanged between components or across boundaries.
Description	<p>A connection between components to exchange information with other components, organizational units, or business processes.</p> <ul style="list-style-type: none"> Types include: <ul style="list-style-type: none"> User Interfaces External people such as, stakeholders or regulators Business processes Data interfaces between systems Application programming interfaces (API's) Hardware devices
Elements	<ul style="list-style-type: none"> Preparing for Identification <ul style="list-style-type: none"> Other techniques used to understand which interfaces need to be identified or further analyzed: <ul style="list-style-type: none"> Document analysis Observation Scope modeling (Such as context diagrams) Interviews Conduct Interface Identification <ul style="list-style-type: none"> Describe the function of the interface. Assess the frequency of usage. Evaluate which type of interface may be appropriate. Elicit initial details about the interface. Define Interfaces <ul style="list-style-type: none"> Requirements for interfaces describe: Inputs to and outputs from Validation rules that govern the data going in or out Any triggering events that initiate an exchange of data Interviews Interface definition includes: <ul style="list-style-type: none"> Interface name Interface scope Exchange method Message format Exchange frequency
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Functional coverage for requirements is provided by analyzing interfaces early in the process. Interface specifications provides structured means of allocating requirements, business rules, and constraints to the solution.

	<ul style="list-style-type: none"> • Avoids over analyzing details due to broad scope. • Limitations <ul style="list-style-type: none"> • Does not provide insight into internal solution components.
--	--

10.25 Interviews

Purpose	<ul style="list-style-type: none"> • Elicit information from individuals or groups. • Establish relationships and build trust with stakeholders.
Description	<ul style="list-style-type: none"> • Common technique to elicit requirements. • Involves direct communication, typically between two individuals. • Basic Interview Types include: <ul style="list-style-type: none"> • Structured • Unstructured
Elements	<ul style="list-style-type: none"> • Interview Goal <ul style="list-style-type: none"> • Purpose of asking a number of questions. • Individual goal for each interview based on what the interviewee can provide. • Examples of Interview Goals include: <ul style="list-style-type: none"> • Collect data • Research stakeholder's views • Elicit requirements for developing a proposed solution • Build rapport with stakeholders • Build support for proposed solution • Potential Interviewees <ul style="list-style-type: none"> • Identified with help of project team and/or stakeholders. • Identified based on the interview goals • Interview Questions <ul style="list-style-type: none"> • Designed based on interview goals. Types include: <ul style="list-style-type: none"> • Open-ended questions used to elicit a dialogue or series of steps • Closed questions used to elicit single response to clarify or confirm a previous answer • Customized questions used to elicit information specific or unique to an individual interviewee • Standardized questions used to produce summarized report or analysis • Interview Logistics <ul style="list-style-type: none"> • Location of interview. • Mode of communication (in-person, online conferencing, phone). • Method of recording interview. • State if questions will be sent to interviewees before the interview or not. • State if interview results will be confidential. • State how interview results will be summarized and reported on. • Interview Flow <ul style="list-style-type: none"> • Opening the Interview • State the purpose, confirm interviewee's role in the effort, and address any concerns stated by the interviewee • State how results will be communicated and to whom it will be sent to • During the Interview <ul style="list-style-type: none"> • Stay on topic and maintain focus on interview goals • Practice active listening • Take written notes or records the interview • Closing the Interview

	<ul style="list-style-type: none"> • Summarize the session and how the results will be communicated • Provide contact information & thank interviewee for their time • Interview Follow-Up <ul style="list-style-type: none"> • Organize and communicate the results with the interviewee as soon as possible after the interview to confirm information. • Sharing the results allows the interviewee to point out any discrepancies or incomplete information recorded.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Encourages participation by stakeholders. • Establishes rapport with stakeholders. • Simple & direct technique that can be used in a variety of situations. • Enables full discussion and explanation of questions and answers between the interviewer and participants. • Enables observations of non-verbal behavior. • Allows participants to share opinions in private that they may be unwilling to express in public. • Limitations <ul style="list-style-type: none"> • Significant time is required to plan for and conduct interviews. • There is a risk of unintentionally leading the interviewee.

10.26 Item Tracking

Purpose	To capture and assign responsibility for issues & stakeholder concerns that pose an impact to the solution
Description	<ul style="list-style-type: none"> • Organized approach used to address stakeholder concerns. • Types of Items Tracked can include: <ul style="list-style-type: none"> • Actions • Assumptions • Constraints • Dependencies • Defects • Enhancements • Issues
Elements	<ul style="list-style-type: none"> • Item Record <ul style="list-style-type: none"> • Attributes can include the following: <ul style="list-style-type: none"> • Item identifier • Summary • Category • Type • Date Identified • Identified By • Impact • Priority • Resolution Date • Owner • Item Management <ul style="list-style-type: none"> • An item's lifecycle is governed by stakeholder needs & organizational process standards. • Each item must be tracked to its closure or resolution. • Metrics <ul style="list-style-type: none"> • Determines how well: <ul style="list-style-type: none"> • Items are being resolved • The initiative undertaken is progressing

	<ul style="list-style-type: none"> The tracking process is going
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Ensures concerns around stakeholder requirements are captured, tracked, and resolved. Allows ranking of items to determine importance level as compared to other items. Limitations <ul style="list-style-type: none"> Stakeholders could become immersed in overly detailed data. The time and effort used to manage items may outweigh the benefits of recording items.

10.27 Lessons Learned

Purpose	Compile & document successes, opportunities, for improvement, failures, and recommendations for improving the performance of future initiatives or efforts.
Description	<ul style="list-style-type: none"> A session can help identify processes and/or deliverables to be changed or incorporated into future efforts. Also known as a Retrospective.
Elements	<ul style="list-style-type: none"> Sessions can include a review of: <ul style="list-style-type: none"> BA activities or deliverables Final solution, service, or product Automation or technology that was introduced or eliminated Impact to organizational processes Performance expectations & results Positive or negative variances Root causes impacting performance results Recommendations for behavioral approaches
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Identifies opportunities or areas of improvement. Can assist in building team morale. Reinforces positive experiences & successes. Reduces risks for future actions. Results in tangible value or metrics that are actionable. Allows team members to recognize strengths or shortcomings with the project structure, methodology, or tools that were used during the effort in question. Limitations <ul style="list-style-type: none"> Honest discussion can be obstructed if 'blame game' ensues. Participants may be reluctant to discuss & document problems.

10.28 Metrics & Key Performance Indicators (KPI)

Purpose	To measure the performance of solutions, solution components, and other items of interest to stakeholders.
Description	<ul style="list-style-type: none"> Quantifiable mechanism by which to measure the degree of progress toward achieving some goal, objective, output, activity, or other input. Metrics are specific levels achieved by an indicator that represents progress. Metrics & reporting are key components of monitoring and evaluation. Key performance is a specific type of indicator that measures progress towards a strategic goal or objective.
Elements	<ul style="list-style-type: none"> Indicators <ul style="list-style-type: none"> Displays the result of analyzing one or more specific measures for addressing a concern about a need, value, output, activity, or input in a table or graphical form.

	<ul style="list-style-type: none"> • Six (6) Characteristics of a Good Indicator: • Clear • Relevant • Economical • Adequate • Quantifiable • Trustworthy & Credible <ul style="list-style-type: none"> • Metrics <ul style="list-style-type: none"> • Quantifiable levels of indicators that are measured at a specified point in time. • Can be a specific point, a threshold, or a range. • Structure <ul style="list-style-type: none"> • The establishment of a data collection procedure, data analysis procedure, reporting procedure, and the collection of baseline data. • Reporting <ul style="list-style-type: none"> • Comparing the baseline, current metrics, & target metrics with calculations of the differences. • Visuals typically used to explain details versus tables.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Shows the extent to which a solution meets an objective. • Shows how effective the inputs and activities of developing the solution (outputs) were. • Facilitates organizational alignment by: • Linking goals to objectives • Supporting solutions, underlying tasks, and resources • Limitations <ul style="list-style-type: none"> • The time and effort used to collect, analyze, and report detailed metrics may outweigh the benefits. • The amount of data collected may cause a delay in creating useful reports that would allow timely action.
10.29 Mind Mapping	
Purpose	<ul style="list-style-type: none"> • Articulate and capture thoughts, ideas, and information in a fashion similar to how human minds process information. • Help generate new ideas. • Assist in conducting a ‘deep dive’ of existing concepts. • Facilitate the creative and critical thinking about a specific problem. • Create an overview or summary of complex ideas or problems.
Description	<ul style="list-style-type: none"> • Form of note taking with no standardized format that can be created individually or with a group. • Captures ideas, thoughts, and information in a non-linear diagram. • Uses images, words, color, and connected relationships to apply structure and logic to information. • Has a main idea connected to secondary ideas that are also linked to multiple sub-topics as needed to describe a concept • Example of Mind Map Structure

Elements	<ul style="list-style-type: none"> • Main Topic <ul style="list-style-type: none"> • Central theme or concept that is the focus of the effort. • Typically placed in the center of the diagram to denote its significance. • Topics <ul style="list-style-type: none"> • Thoughts or concepts that are directly linked to the main topic. • Links from the main topic is typically annotated with a keyword to describe the nature of the association. • Sub-Topics <ul style="list-style-type: none"> • Further expounds upon a topic and is directly linked to the topic. • Links from topic is typically annotated with a keyword to describe the nature of the association. • Branches <ul style="list-style-type: none"> • The association (line) between the main topic, topics, and sub-topics. • The line is typically where the keyword is located that describes the nature of the association. • Keywords <ul style="list-style-type: none"> • Single word. • Articulates the nature of the association of topics or sub-topics to the main topic. • Helps generate categories and for triggering additional associations. • Color <ul style="list-style-type: none"> • Can be used to categorize, prioritize, and analyze concepts. • Images <ul style="list-style-type: none"> • Used to express more complex ideas that may be difficult to describe in single words. • Can be used to encourage creativity & innovation similar to a group brainstorming session.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Effective collaboration and communication medium. • Good at summarizing complex ideas. • Facilitates stakeholder understanding and decision making. • Enables creative problem solving. • Helpful in preparing & delivering presentations. • Limitations <ul style="list-style-type: none"> • As an informal exercise, communicating the resulting information to

	<p>relevant stakeholders may prove difficult in certain situations.</p> <ul style="list-style-type: none"> • Can be misconstrued as brainstorming exercise. • Sharing the ultimate result of a mind map may prove difficult to communicate to those not involved in the exercise.
--	---

10.30 Non Functional Requirements Analysis

Purpose	<ul style="list-style-type: none"> • Examines the requirements for performance of a solution. • Examines & specifies the operational performance a solution must achieve to be deemed successful. 															
Description	<ul style="list-style-type: none"> • Also known as Quality Attributes & Quality of service requirements. • Augment the functional requirements of a solution, identify the constraints on those requirements, or describe quality attributes a solution must exhibit when based on those functional requirements. • Generally described in textual formats as declarative statements or in matrices. • Example: X transactions must be processed within S seconds 															
Elements	<ul style="list-style-type: none"> • Categories of Non-Functional Requirements <table border="1" style="margin-left: 20px;"> <tr> <td>Availability</td> <td>Portability</td> <td>Certification</td> </tr> <tr> <td>Compatibility</td> <td>Reliability</td> <td>Compliance</td> </tr> <tr> <td>Functionality</td> <td>Scalability</td> <td>Localization</td> </tr> <tr> <td>Maintainability</td> <td>Security</td> <td>Service Level Agreements</td> </tr> <tr> <td>Performance Efficiency</td> <td>Usability</td> <td>Extensibility</td> </tr> </table> • Measurement of Non-Functional Requirements <ul style="list-style-type: none"> • Testable requirement dictates non-functional requirements must be measurable, i.e., quantifiable, where possible. • Examples: <ul style="list-style-type: none"> • The systems must be available 98.9998% of the time • System response time must not exceed 10 ms • Context of Non-Functional Requirements <ul style="list-style-type: none"> • Ensure proper quality requirements are addressed based on the particular industry, situation, or circumstance. • Examples: <ul style="list-style-type: none"> • Regulations may impose constraints on medical device companies • Constraints on some quality requirements may change the success measurement for another quality requirement 	Availability	Portability	Certification	Compatibility	Reliability	Compliance	Functionality	Scalability	Localization	Maintainability	Security	Service Level Agreements	Performance Efficiency	Usability	Extensibility
Availability	Portability	Certification														
Compatibility	Reliability	Compliance														
Functionality	Scalability	Localization														
Maintainability	Security	Service Level Agreements														
Performance Efficiency	Usability	Extensibility														
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Clearly states constraints on functional requirements. • Provides measurable ways to judge the success of functional requirements on a solution. • Limitations <ul style="list-style-type: none"> • Stakeholders must accurately assess the needs of the users. 															

	<ul style="list-style-type: none"> Multiple users may have quality requirements that conflict. Certain quality attributes are contradictory and therefore must be proactively negotiated and communicated to stakeholders. Example: Certification & Compliance quality attributes may make it impossible to meet overly strict Performance efficiency or Usability requirements Some quality requirements require subjective measurement.
--	---

10.31 Observation

Purpose	<ul style="list-style-type: none"> It is used to elicit information by viewing and understanding activities and their context. It is used as a basis for identifying needs and opportunities, understanding business process, setting performance standards, evaluating solution performance or supporting training and development.
Description	<ul style="list-style-type: none"> Observation of activities, also known as job shadowing, involves examining a work activity firsthand as it is performed. There are 2 basic approaches: <ul style="list-style-type: none"> Active/Noticeable Passive/Unnoticeable
Elements	<ul style="list-style-type: none"> Observation Objectives <ul style="list-style-type: none"> A clear and specific objective establishes a defined purpose of the observation session. For example: <ul style="list-style-type: none"> Identifying opportunities for improvement Establishing performance metrics Prepare for Observation <ul style="list-style-type: none"> Preparing for an observation session involves planning the observation approach based on the objectives and deciding who should be performing which activities considering skills and experience levels. Conduct the Observation Session <ul style="list-style-type: none"> Before the observation session: <ul style="list-style-type: none"> Explain the purpose of this observation session and basic rule of conduct to be observed during the session During the observation session: <ul style="list-style-type: none"> Ensure the rule of conduct is being followed Look for any anomalies Gather data around the observation session in terms of time taken, notes, quality of session, any question or concerns Confirm and Present Observation Results <ul style="list-style-type: none"> After the observation session, business analysts review the notes and data recorded from the observation and follow up with the participant to obtain answers to any remaining questions or to fill any gaps or to validate data or notes. The validated notes and data are collated with other related observations to identify similarities, differences, and trends. Findings are aggregated, summarized, and analyzed. Needs and opportunities for improvement are communicated to stakeholders.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Observers can gain realistic and practical insight about the activities and their tasks within an overall process. Instances of informally performed tasks as well as any workarounds can be identified. Productivity can be viewed firsthand and realistically compared against

	<p>any established performance standards or metrics.</p> <ul style="list-style-type: none"> Recommendations for improvement are supported by objective and quantitative evidence. <p>Limitations</p> <ul style="list-style-type: none"> May be disruptive to the performance of the participant and the overall organization. Can be threatening and intrusive to the person being observed. While being observed, a participant may alter their work practices. Significant time is required to plan for and conduct observations. Not suitable for evaluating knowledge-based activities since these are not directly observable.
--	--

10.32 Organizational Modeling

Purpose	<ul style="list-style-type: none"> It is used to describe the roles, responsibilities and reporting structures that exist within an organization. It is used to align those structures with the organization's goal.
Description	<p>An organizational model is a visual representation of the organizational unit which defines:</p> <ul style="list-style-type: none"> The boundaries of the group (who is in the group) The formal relationships between members (who reports to whom) The functional role for each person The interfaces (interaction and dependencies) between the unit and other units or stakeholders
Elements	<ul style="list-style-type: none"> Types of Organizational Models <ul style="list-style-type: none"> Functionally oriented – This model groups staff together based on shared skills or areas of expertise and generally encourage a standardization of work or processes within the organization. It is prone to develop communication and cross-functional coordination problems. Market oriented – This model is intended to serve particular customer groups, geographical areas, projects or processes. It could lead to duplication of work. The Matrix Model – This model has separate managers for each functional area and for each product, service or customer group. A challenge of the matrix model is that each employee has two managers (who are focused on different goals) and accountability is difficult to maintain. Roles <ul style="list-style-type: none"> An organizational unit includes a number of defined roles. Each role requires a certain set of skills and knowledge, has specific responsibilities, performs certain kind of work and has defined relationships with other roles in the organization. Interfaces <ul style="list-style-type: none"> Each organizational unit has interfaces with other organizational units. Interfaces may be in the form of communication with people in other roles and work packages that the organizational unit receives from or delivers to other units. Organizational Charts <ul style="list-style-type: none"> The fundamental diagram used in organizational modeling is the organizational chart. Some conventions used in Org Chart are: <ul style="list-style-type: none"> A box depicts – Organizational unit or Roles and People

	<ul style="list-style-type: none"> • A line depicts – Lines of reporting • Solid line denotes direct authority • Dotted line indicates information transfer <p>Influencers</p> <ul style="list-style-type: none"> • Organization charts represent the formal structure of the organization but business analyst needs to find out who is representative for the group in reality to get correct and complete information.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> • Organizational models are common in most organizations. • Including an organizational model in business analysis information allows team members to provide support. Future projects may benefit from knowing who was involved in this project and what their role entailed. Limitations <ul style="list-style-type: none"> • Organizational models are sometimes out of date. • Informal lines of authority, influence, and communication not reflected in the org chart are more difficult to identify and may conflict with the organizational chart.

10.33 Prioritization

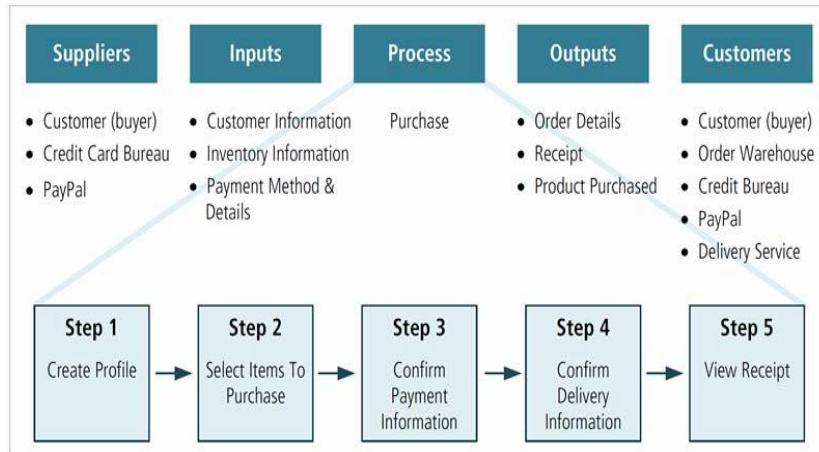
Purpose	Prioritization provides a framework for business analysts to facilitate stakeholder decisions and to understand the relative importance of business analysis information.
Description	<ul style="list-style-type: none"> Prioritization is a process used to determine the relative importance of business analysis information. The importance may be based on value, risk, difficulty of implementation or other criteria. There are many approaches to prioritization. For the purpose of this technique, prioritization is classified into one of four approaches. <ul style="list-style-type: none"> • Grouping • Ranking • Time boxing / Budgeting • Negotiation When choosing a prioritization approach, business analysts consider the audience, their needs, and their opinions on the value a requirement or business analysis information brings to a stakeholder's respective area. Business analysts revisit priorities and utilize different approaches when changes occur in the business environment, to the stakeholders, and to the business analysis information.
Elements	<ul style="list-style-type: none"> Grouping <ul style="list-style-type: none"> • Grouping consists of classifying business analysis information according to predefined categories such as high, medium, or low priority. Many requirements management tools support listing the priority category as an attribute of a requirement. Ranking <ul style="list-style-type: none"> • Ranking consists of ordering business analysis information from most to least important. Some adaptive approaches involve the explicit sequencing of requirements in an ordered list (a product backlog). Time Boxing/Budgeting <ul style="list-style-type: none"> • Time boxing or budgeting prioritizes business analysis information based on the allocation of a fixed resource. It is frequently used when the solution approach has been determined. • Time boxing is used to prioritize requirements based on the amount of

	<p>work that the project team is capable of delivering in a set period of time.</p> <ul style="list-style-type: none"> Budgeting is used when the project team has been allocated a fixed amount of money. This approach is most often used when a fixed deadline must be met or for solutions that are enhanced on a regular and frequent basis. Negotiation <ul style="list-style-type: none"> The negotiation approach involves establishing a consensus among stakeholders as to which requirements will be prioritized.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Facilitates consensus building and trade-offs and ensures that solution value is realized and initiative timelines are met. Limitations <ul style="list-style-type: none"> Some stakeholders may attempt to avoid difficult choices and fail to recognize the necessity for making trade-offs. The solution team may intentionally or unintentionally try to influence the result of the prioritization process by overestimating the difficulty or complexity of implementing certain requirements. Metrics and key performance indicators are often not available when prioritizing business analysis information; therefore, a stakeholder's perspective of the importance may be subjective.

10.34 Process Analysis

Purpose	Process analysis assesses a process for its efficiency and effectiveness, as well as its ability to identify opportunities for change.
Description	<ul style="list-style-type: none"> Process analysis is used for various purposes including: <ul style="list-style-type: none"> Recommending a more efficient or effective process Understanding factors to be included in a contract negotiation A number of frameworks and methodologies exists that focus on process analysis and improvement methods such as Six sigma and Lean. Methods for process improvement include value stream mapping, statistical analysis and control, process simulation, benchmarking and process frameworks. Common changes made to processes in order to improve them include: <ul style="list-style-type: none"> Reducing the time required to complete a task or tasks in the process, Automating steps that are more routine or predictable Increasing the degree of automation in the decision making required by the process When analyzing a process, business analysts look for: <ul style="list-style-type: none"> How the process adds or creates value for the organization How the process aligns to organizational goals or strategy To what degree the process is and needs to be efficient, effective, repeated, measured, controlled, used and transparent How the requirements for a solution cover the future state process and its external stakeholders including customers
Elements	<ul style="list-style-type: none"> Identify Gaps and Areas to Improve <ul style="list-style-type: none"> Identifying gaps and areas to improve using some industry standard models and process frameworks helps to identify what areas are in scope for analysis. When identifying gaps and areas to improve, business analysts: <ul style="list-style-type: none"> Identify gaps between the current and desired future state, Understand pain points and challenges of the process from

- multiple points of view
- Align the gaps and areas to improve with the strategic direction of the organization,
 - Understand the relationship of the gaps and areas to improve to changes in the enterprise
- **Identify Root Cause**
 - Identifying the root cause of the gaps and improvement areas ensures that the solution addresses the right gap and area. When identifying the root cause, business analysts understand:
 - There may be multiple root causes,
 - The inputs leading to the gap or area of improvement
 - Who the right people are to identify the root cause
 - The current measurements and motivators in place for those owning or performing the process.
 - **Generate and Evaluate Options**
 - Generating options and alternative solutions to solve for the gap or area of improvement helps the team evaluate and see different points of view for improving the process.
 - It is important for stakeholders to be involved in identifying the impact, feasibility and value of the proposed solution relative to alternative options.
 - **Common Methods**
 - SIPOC is a process analysis method covering Suppliers, Inputs, Process, Outputs and Customers of the process being analyzed.
 - SIPOC provides a simple overview of the process.
 - It shows the complexity of who and what is involved in creating inputs to the process and shows who receives outputs from the process.
 - SIPOC is a powerful tool used to create dialogue about problems, opportunities, gaps, root cause, options and alternatives during process analysis.
 - **Common Methods**
 - SIPOC model



	<p>Supplier</p> <ul style="list-style-type: none"> • Requester (sales manager) • Purchasing agent <p>Input</p> <ul style="list-style-type: none"> • Request • Approved P.O. <p>Process</p> <p>Output</p> <ul style="list-style-type: none"> • New car • Executed P.O. • Record added to on-line Fleet System <p>Customer</p> <ul style="list-style-type: none"> • Sales manager • Accounting • Fleet System operators <p>Purchase Car</p> <p>Subprocesses:</p> <ul style="list-style-type: none"> Research Car Visit Dealers Negotiate Price Record Purchase <ul style="list-style-type: none"> Common Methods <ul style="list-style-type: none"> Value stream mapping (VSM) is a process analysis method used in Lean methodologies. Value stream mapping involves the diagramming and monitoring of inputs and application points for processing these inputs starting from the front end of the supply chain. The value stream map provides a one-page picture of all the steps involved in the end-to-end process, including both value-adding (the value stream) and non-value-adding (waste) elements. Common Methods <ul style="list-style-type: none"> Value stream map <p>The detailed VSM diagram illustrates the flow from Supplier to Customer through four sequential processes. Each process consists of Activity/Tasks and Data. Arrows indicate the flow of Electronic Information Flow between processes and Shipment between the Supplier and Customer. Documents (Manual Information) are passed between the second and third processes. The diagram also shows time components: Wait time, Processing/Conversion time, and Total elapsed time, which include Value-adding time and Non-value adding time.</p>
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Can provide benefits to improving processes. Ensures solutions address the right issues, minimizing waste. Many different techniques and methodologies can be used and provide teams with great flexibility in approach. Limitations <ul style="list-style-type: none"> Can be time-consuming. There are many techniques and methodologies in process analysis. It can be challenging to decipher which to use and how rigorously to follow them, given the scope and purpose. May prove ineffective at process improvement in knowledge or

	decision intensive processes.
10.35 Process Modeling	
Purpose	Process modeling is a standardized graphical model used to show how work is carried out and is a foundation for process analysis.
Description	<ul style="list-style-type: none"> A process model can be constructed on multiple levels, each of which can be aligned to different stakeholder points of view. At a high (enterprise or context) level, the model provides a general understanding of a process and its relationship to other processes. At lower (operational) levels, it can define more granular activities and identify all outcomes, including exceptions and alternative paths. At the lowest (system) level, the model can be used as a basis for simulation or execution. The business analyst can use a process model to define the current state of a process (also known as an as-is model) or a potential future state (also known as a to-be model). Process models generally include: <ul style="list-style-type: none"> The participants in the process The business event that triggers the process The steps or activities of the process (both manual and automated) The paths (flows) and decision points that logically link those activities The results of the process
Elements	<ul style="list-style-type: none"> Types of Models and Notations <ul style="list-style-type: none"> Many different notations are used in process modeling. The most commonly used notations include the following: <ul style="list-style-type: none"> Flowcharts and Value Stream Mapping (VSM): used in the business domain Data Flow diagrams and Unified Modelling Language™ (UML®) diagrams: used in the information technology domain Business Process Model and Notation (BPMN) Integrated DEFinition (IDEF) notation and Input, Guide, Output, Enabler (IGOE) diagrams: used for establishing scope SIPOC and Value Stream Analysis: used for process modeling Types of Models and Notations <ul style="list-style-type: none"> Process models typically contain some or all of the following key elements: <ul style="list-style-type: none"> Activity: an individual step or piece of work that forms part of the business process Event: a zero-time occurrence which initiates, interrupts, or terminates an activity or task within a process or the process itself Directional Flow: a path that indicates the logical sequence of the workflow. In general, diagrams are drawn to show the passage of time in a consistent fashion Decision Point: a point in the process where the flow of work splits into two or more flows (paths), which may be mutually exclusive alternatives or parallels. A decision can also be used to locate rules where separate flows merge together Link: a connection to other process maps Role: a type of person or group involved in the process.

	<ul style="list-style-type: none"> Types of Models and Notations <ul style="list-style-type: none"> Flowchart: Flowcharts are used commonly with non-technical audiences and are good for gaining both alignment with what the process is and context for a solution. A flowchart can be simple, displaying just the sequence of activities, or it can be more comprehensive, using swimlanes. A swimlane is a partitioned area (horizontal or vertical) that segregates those activities in the process that are carried out by a particular role. definitions typically match those in the organizational model Types of Models and Notations – Flowchart
	<pre> graph TD IC[Incoming call] --> SA[Staff answers] SA --> SFR{Specific staff requested?} SFR -- Yes --> TFS[Transfer to one of the front desk staff] SFR -- No --> TCOFO[Transfer call outside the office] TFS --> CE1(Call ended) TCOFO --> CE1 SFR -- No --> SFFD{Staff from front desk?} SFFD -- Yes --> TFS SFFD -- No --> TCOFO TFS --> CE2(Call ended) TCOFO --> CE2 SFFD -- No --> NA{Needs assistance?} NA -- Yes --> GA[Get assistance] GA --> CE3(Call ended) NA -- No --> AR{Appointment related?} AR -- No --> EPS[Existing patient scheduling] AR -- Yes --> NP{New patient scheduling} NP -- Yes --> CPS[New patient scheduling] CPS --> CE4(Call ended) </pre>

	<pre> graph TD subgraph Customer A[Add Product to Basket] A --> D{ } D -- [no] --> End1(()) D -- [finished ?] --> C[Check Out] C --> B[Login / Register] B --> D2{ } D2 -- [user known] --> E[query order / delivery information] E --> F[Process Payment] E --> G[Process Delivery] E --> H[Send confirmation mail] end subgraph OnlineShop E F G H end </pre>
Usage Considerations	<ul style="list-style-type: none"> Types of Models and Notations <ul style="list-style-type: none"> The Activity Diagram is one of the use case realization diagrams defined in the Unified Modelling Language™ (UML®). Originally designed to elaborate on a single use case, the activity diagram has been adopted for more general process modeling purposes, including business process modeling. While similar in appearance to a flowchart, the activity diagram typically employs swimlanes to show responsibilities, synchronization bars to show parallel processing, and multiple exit decision points. Types of Models and Notations – Activity Diagram <pre> graph TD Start(()) --> RequestedOrder[Requested Order] RequestedOrder --> ReceiveOrder[Receive Order] ReceiveOrder --> Decision{ } Decision -- [order rejected] --> End2(()) Decision -- [order accepted] --> FillOrder[Fill Order] FillOrder --> SendInvoice[Send Invoice] FillOrder --> ShipOrder[Ship Order] SendInvoice --> AcceptPayment[Accept Payment] ShipOrder --> AcceptPayment AcceptPayment --> CloseOrder[Close Order] CloseOrder --> End3(()) </pre>

	<p>various stakeholder groups.</p> <ul style="list-style-type: none"> • Effective at showing how to handle a large number of scenarios and parallel branches. • Can help identify any stakeholder groups that may have otherwise been overlooked. • Facilitates the identification of potential improvements by highlighting “pain points” in the process structure (i.e. process visualization). • Likely to have value in its own right. They provide documentation for compliance purposes and can be used by business stakeholders for training and coordination of activities. • Can be used as a baseline for continuous improvement. • Ensures labeling consistency across artifacts. • Provides transparency and clarity to process owners and participants on activity responsibilities, sequence and hand-overs. <p>• Limitations</p> <ul style="list-style-type: none"> • To many people in IT, a formal process model tends to reflect an older and more document-heavy approach to software development. Therefore, project time is not allocated to developing a process model, especially of the current state or problem domain. • Can become extremely complex and unwieldy if not structured carefully. This is especially true if business rules and decisions are not managed separately from the process. • Complex processes can involve many activities and roles; this can make them almost impossible for a single individual to understand and ‘sign off’ • Problems in a process cannot always be identified by looking at a high-level model. A more detailed model with reference to metadata (such as path frequency, cost, and time factors) is usually required. It is often necessary to engage with stakeholders directly to find the operational problems they have encountered while working with a process. • In a highly dynamic environment where things change quickly, process models can become obsolete. • May prove difficult to maintain if the process model only serves as documentation, as stakeholders may alter the process to meet their needs without updating the model.
--	--

10.36 Prototyping

Purpose	<ul style="list-style-type: none"> • Prototyping is used to elicit and validate stakeholder needs through an iterative process that creates a model or design of requirements. • It is also used to optimize user experience, to evaluate design options, and as a basis for development of the final business solution.
Description	<ul style="list-style-type: none"> • Prototyping is a proven method for product design. It works by providing an early model of the final result, known as a prototype. • Prototyping is used to identify both missing or improperly specified requirements and unsubstantiated assumptions by demonstrating what the product looks like and how it acts in the early stages of design • Prototypes can be non-working models, working representations, or digital depictions of a solution or a proposed product.
Elements	<ul style="list-style-type: none"> • Prototyping Approach <ul style="list-style-type: none"> • There are 2 common approaches to prototyping: • Throw-away:

	<ul style="list-style-type: none"> • Prototypes are generated with simple tools (such as paper and pencil, a whiteboard, or software) to serve the goal of uncovering and clarifying requirements. • The prototype may be updated or evolve during the course of discussion and development, but does not become workable code or get maintained as a deliverable once the final system or process is implemented. • This method is helpful for identifying functionality or processes that are not easily elicited by other techniques, have conflicting points of view, or are difficult to understand. • These prototypes can be an inexpensive tool to uncover or confirm requirements that go beyond an interface including requirements related to processes, data, and business rules. • Evolutionary or Functional: <ul style="list-style-type: none"> • Prototypes are created to extend initial requirements into a functioning solution as requirements are further defined through stakeholder use. • This approach produces a working solution and usually requires a specialized prototyping tool or language. These prototypes may be used in the final solution. • If specialized software is used, business processes, rules, and data can be simulated to evaluate the impact of changes and validate desired outcomes. • Prototyping Examples <ul style="list-style-type: none"> • There are many forms of prototyping in use today. Each of the following can be considered a form of prototyping. • Proof of Principle or Proof of Concept: is a model created to validate the design of a system without modelling the appearance, materials used in the creation of work, or processes/workflows ultimately used by the stakeholders. • Form Study Prototype: is used to explore the basic size, look, and feel of a product that will be manufactured, without creating actual functionality. It is used to assess ergonomic and visual factors using a sculptural representation of the product made from inexpensive materials. This type of prototype may also be used to model a workflow or navigation at a high level in order to identify gaps or inconsistencies in the possible solution of the properties (for example, appearance, configuration). • Usability Prototype: is a product model created to test how the end user interacts with the system without including any of the properties (for example, appearance, configuration). • Visual Prototype: is a product model created to test the visual aspects of the solution without modeling the complete functionality. • Functional Prototype: is a model created to test software functionality, qualities of the system for the user (for example, appearance), and workflow. It is also referred to as a working model and is used both to simulate business processes and business rules and to evaluate software function calls. • Prototyping Methods <ul style="list-style-type: none"> • The following is a list of commonly used methods for prototyping: <ul style="list-style-type: none"> • Storyboarding: is used to visually and textually detail the sequence of activities by summing up different user
--	--

	<p>interactions with the solution or enterprise.</p> <ul style="list-style-type: none"> • Paper Prototyping: uses paper and pencil to draft an interface or process. • Workflow Modelling: depicts a sequence of operations that are performed and usually focuses solely on the human aspect. • Simulation: is used to demonstrate solutions or components of a solution. It may test various processes, scenarios, business rules, data, and inputs.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Provides a visual representation for the future state. • Allows for stakeholders to provide input and feedback early in the design process. • When using throw-away or paper prototyping methods, users may feel more comfortable being critical of the mock-up because it is not polished and release-ready. • A narrow yet deep vertical prototype can be used for technical feasibility studies, proof of concept efforts, or to uncover technology and process gaps. • Limitations <ul style="list-style-type: none"> • If the system or process is highly complex, the prototyping process may become bogged down with discussion of 'how' rather than 'what', which can make the process take considerable time, effort, and facilitation skill. • Underlying technology may need to be understood or assumed in order to initiate prototyping. • If the prototype is deeply elaborate and detailed, stakeholders may develop unrealistic expectations for the final solution. These can range from assumed completion dates to higher expectations of performance, reliability and usability. • Stakeholders may focus on the design specifications of the solution rather than the requirements that any solution must address. This can, in turn, constrain the solution design. Developers may believe that they must provide a user interface that precisely matches the prototype, even if more elegant technology and interface approaches exist.

10.37 Reviews

Purpose	Reviews are used to evaluate the content of a work product.
Description	<ul style="list-style-type: none"> • Different types of reviews tailored to the need of organization and business analysts are conducted for work products. • A review is conducted using these dimensions: <ul style="list-style-type: none"> • Objectives – defining the purpose of the review. • Techniques – identifying either a formal / informal way to perform the review. • Participants – identifying who should take part in the review activity. • Each review is focused on a work product, not the skill or actions of participants • The work product may be a package of deliverables, a single deliverable, a portion of deliverable or work in process. • For completed work product, the objective of the review is usually to remove defects or inform the reviewers about the content. For work in process, the review may be conducted to resolve an issue or question.

	<ul style="list-style-type: none"> Each review includes the business analyst as a participant. Reviewers may be peers, especially for work in process, or stakeholders, who validate that the work product is complete and correct. Reviews can include: An overview of the work product and review objectives, Checklists and reference materials that can be used by reviewers, Reviewing the work product and documenting the findings, and Verifying any rework. Business Analyst updates work product using reviewer's feedback
Elements	<ul style="list-style-type: none"> Objectives <ul style="list-style-type: none"> Objectives are clearly communicated to all participants prior to the review. Objectives may include one or more goals, for example: <ul style="list-style-type: none"> To remove defects To ensure conformance to specifications or standards To ensure the work product is complete and correct To establish consensus on an approach or solution To answer a question, resolve an issue, or explore alternatives To educate reviewers about the work product To measure work product Techniques <ul style="list-style-type: none"> Reviews can be formal or informal. The techniques used during a review are selected to support the objectives of the review. The following techniques are commonly used by business analysts when conducting reviews: <ul style="list-style-type: none"> Inspection: a formal technique that includes an overview of the work product, individual review, logging the defects, team consolidation of defects, and follow-up to ensure changes were made. The focus is to remove defects and create a high quality work product. While usually performed by peers, it can also be used for stakeholder reviews. Formal Walkthrough (also known as Team Review): a formal technique that uses the individual review and team consolidation activities often seen in inspection. Walkthroughs are used for peer reviews and for stakeholder reviews. Single Issue Review (also known as Technical Review): a formal technique focused on either one issue or a standard in which reviewers perform a careful examination of the work product prior to a joint review session held to resolve the matter in focus. Informal Walkthrough: an informal technique in which the business analyst runs through the work product in its draft state and solicits feedback. Desk Check: an informal technique in which a reviewer who has not been involved in the creation of the work product provides verbal or written feedback. Pass Around: an informal technique in which multiple reviewers provide verbal or written feedback. The work product may be reviewed in a common copy of the work product or passed from one person to the next. Ad hoc: an informal technique in which the business analyst seeks informal review or assistance from a peer. Participants

	<ul style="list-style-type: none"> Participant roles involved in any particular review depend on the objectives of the review, the selected technique, and any organizational standards that may be in place. Typical review roles are described below 		
Role	Description	Responsibility	Applicable Techniques
Author	Author of the work product	Answers questions about the work product and listens to suggestions and comments. Incorporates changes into the work product after the review.	All
Reviewer	A peer or stakeholder	Examines the work product according to the review objectives. For defect detection reviews, the reviewer examines the work product prior to a review session and keeps track of both defects found and suggestions for improvement.	All
Facilitator	A neutral facilitator (should not be the author in order to avoid compromising the review).	Facilitates the review session, keeps participants focused on the objectives of the review and ensures that each relevant section of the work product is covered. Validates that reviewers have examined the work product before the session begins and ensures that all reviewers participate in the review session.	<ul style="list-style-type: none"> Inspection Formal walkthrough May be helpful for single issue review
Scribe	A neutral participant with strong communication skills.	Documents all defects, suggestions, comments, issues, concerns, and outstanding questions that are raised during a review session. Familiarity with the subject matter enables the scribe to capture items clearly.	<ul style="list-style-type: none"> Inspection Formal and informal walkthrough
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Can help identify defects early in the work product life cycle, 		

	<p>eliminating the need for expensive removal of defects discovered later in the life cycle.</p> <ul style="list-style-type: none"> • All parties involved in a review become engaged with the final outcome; they have a vested interest in a quality result. • Desk checks and pass around reviews can be performed by a reviewer at a convenient time, rather than interrupting work in progress to attend a meeting. <p>• Limitations</p> <ul style="list-style-type: none"> • Rigorous team reviews take time and effort. Thus, only the most critical work products might be reviewed using inspection or formal walkthrough techniques. • Informal reviews by one or two reviewers are practical in terms of the effort required, but they provide less assurance of removing all significant defects than using a larger team and more formal process. • For desk checks and pass around reviews it may be difficult for the author to validate that an independent review was done by each participant. • If review comments are shared and discussed via e-mail there may be many messages to process, which makes it difficult for the author to resolve disagreements or differences in suggested changes.
--	---

10.38 Risk Analysis & Management

Purpose	Risk analysis and management identifies areas of uncertainty that could negatively affect value, analyzes and evaluates those uncertainties, and develops and manages dealing with the risks.
Description	<ul style="list-style-type: none"> • Failure to identify and manage risks may negatively affect the value of the solution. • Risk analysis and management involves identifying, analyzing, and evaluating risks. Where sufficient controls are not already in place, business analysts develop plans for avoiding, reducing, or modifying the risks, and when necessary, implementing these plans. • Risk management is an ongoing activity. Continuous consultation and communication with stakeholders helps to both identify new risks and to monitor identified risks.
Elements	<ul style="list-style-type: none"> • Risk Identification <ul style="list-style-type: none"> • Risks are discovered and identified through a combination of expert judgment, stakeholder input, experimentation, past experiences, and historical analysis of similar initiatives and situations. • The goal is to identify a comprehensive set of relevant risks and to minimize the unknowns. • Risk identification is an ongoing activity. • A risk event could be one occurrence, several occurrences, or even a non-occurrence. A risk condition could be one condition or a combination of conditions. One event or condition may have several consequences, and one consequence may be caused by several different events or conditions. • Each risk can be described in a risk register that supports the analysis of those risks and plans for addressing them. • Example of Risk Register

#	Risk Event or Condition With Consequence	Prob. (1-5)	Impact (1-5)	Risk Score	Risk Modification Plan	Risk Owner	Residual Risk		
							Prob.	Impact	Risk Level
1	Scheduling classes on new site fails to work with the ISP's new online scheduling services when going live. This would potentially prevent customers from scheduling classes and sending payment.	3	5	15	Mitigate - Parallel test all changes and keep current scheduling services running for 2 weeks after going live.	Mary	2	3	6
2	Database developer not available for project. This would completely stall the project as he is the only resource available.	2	5	10	Avoid - Postpone library phase to a later release.	Julie	0	0	0
3	ISP goes bankrupt. We would have site downtime of up to a week because we don't have other potential ISPs identified.	1	5	5	Mitigate - Research ISP options and get pricing, start-up costs, and time frame.	Sam	1	3	3

- **Analysis**
 - Analysis of a risk involves understanding the risk, and estimating the level of a risk.
 - The likelihood of occurrence could be expressed either as a probability on a numerical scale or with values such as Low, Medium, and High.
 - The consequences of a risk are described in terms of their impact on the potential value. The impact of any risk can be described in terms of cost, duration, solution scope, solution quality, or any other factor agreed to by the stakeholders such as reputation, compliance, or social responsibility. Typically, three to five broad categories of level are used to describe how to interpret the potential impact.
 - The level of a given risk may be expressed as a function of the probability of occurrence and the impact.
 - The risks are prioritized relative to each other according to their level. Risks which could occur in the near term may be given a higher priority than risks which are expected to occur later.
 - Risks in some categories such as reputation or compliance may be given higher priority than others.
- **Evaluation**
 - The risk analysis results are compared with the potential value of the change or of the solution to determine if the level of risk is acceptable or not.
 - An overall risk level may be determined by adding up all the individual risk levels.
- **Treatment**
 - One or more approaches for dealing with a risk may be considered, and any combination of approaches could be used to address a risk:
 - **Avoid:** either the source of the risk is removed, or plans are adjusted to ensure that the risk does not occur.
 - **Transfer:** the liability for dealing with the risk is moved to, or shared with, a third party.
 - **Mitigate:** reduce the probability of the risk occurring or the possible negative consequences if the risk does occur.

	<ul style="list-style-type: none"> • Accept: decide not to do anything about the risk. If the risk does occur, a workaround will be developed at that time. • Increase: decide to take on more risk to pursue an opportunity. • Once the approach for dealing with a specific risk is selected, a risk response plan is developed and assigned to a risk owner with responsibility and authority for that risk. • In the case of risk avoidance, the risk owner takes steps to ensure that the probability or the impact of the risk is reduced to nil. • For those risks which cannot be reduced to nil, the risk owner is responsible for monitoring the risk, and for implementing a risk mitigation plan. • Treatment <ul style="list-style-type: none"> • The risk is re-analyzed to determine the residual risk which is the new probability and new impact as a result of the measures taken to modify the risk. • There could be a cost-benefit analysis done to determine if the cost and effort of the measures reduces the level of risk enough to make it worthwhile. The risks may be re-evaluated in terms of the residual risk. • Stakeholders should be informed of the plans for modifying the risks.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Can be applied to strategic risks which affect long-term value of the enterprise, tactical risks which affect the value of a change, and operational risks which affect the value of a solution once the change is made. • An organization typically faces similar challenges on many of its initiatives. The successful risk responses on one initiative can be useful lessons learned for other initiatives. • The risk level of a change or of a solution could vary over time. Ongoing risk management helps to recognize that variation, and to re-evaluate the risks and the suitability of the planned responses. • Limitations <ul style="list-style-type: none"> • The number of possible risks to most initiatives can easily become unmanageably large. It may only be possible to manage a subset of potential risks. • There is the possibility that significant risks are not identified.

10.39 Roles & Permission Matrix

Purpose	A roles and permissions matrix is used to ensure coverage of activities by denoting responsibility, to identify roles, to discover missing roles, and to communicate results of a planned change.
Description	<ul style="list-style-type: none"> • Role and permission allocation involves identifying roles, associating these with solution activities, and then denoting authorities who can perform these activities. • A role is a label for a group of individuals who share common functions. • Each function is portrayed as one or more solution activities. • A single activity can be associated with one or more roles by designating authorities. or more roles by designating authorities. • Each individual that is assigned this authority can perform the associated activity. • The following is an example of a roles and permissions matrix for a software system.

		Roles and Permissions Matrix				
		Role Group 1		Role Group 2		
		Administrator	Manager	Sales	Customer	
Activity						
Create new account		X	X			X
Modify account		X	X			X
Create order		X	X		X	X
View reports		X	X		X	
Create reports		X	X		X	

Elements	<ul style="list-style-type: none"> Identifying Roles <ul style="list-style-type: none"> To identify roles for either internal or external stakeholders, business analysts: <ul style="list-style-type: none"> Review any organizational models, job descriptions, procedure manuals and system user guides Meet with stakeholders to uncover additional roles Through this review and discussion, the business analyst considers both that individuals with the same job title may have different roles and that individuals with different job titles may have the same roles. When identifying roles, business analysts look for common functions that are performed by individuals with similar needs. Identifying Activities <ul style="list-style-type: none"> Business analysts frequently use functional decomposition to break down each function into sub-parts, process modeling to better understand the workflow and division of work among users, and use cases to represent tasks. There may be different levels of abstraction for roles and permission matrices based on the business analysis perspective. Initiative level roles and responsibilities may be identified in a RACI (Responsible, Accountable, Consulted, Informed) matrix. Specific information technology system roles and responsibilities may be identified in a CRUD (Create, Read, Update, and Delete) matrix. Identifying Authorities <ul style="list-style-type: none"> Authorities are actions that identified roles are permitted to perform. For each activity, the business analyst identifies the authorities for each role. When identifying authorities, business analysts consider the level of security needed and how the work flows through the process. Business analysts collaborate with stakeholders to validate identified authorities Refinements <ul style="list-style-type: none"> Delegations <ul style="list-style-type: none"> The business analyst may also identify which authorities can be delegated by one individual to another on a short-term or permanent basis. Inheritances <ul style="list-style-type: none"> Stakeholders may request that when an individual is assigned
----------	--

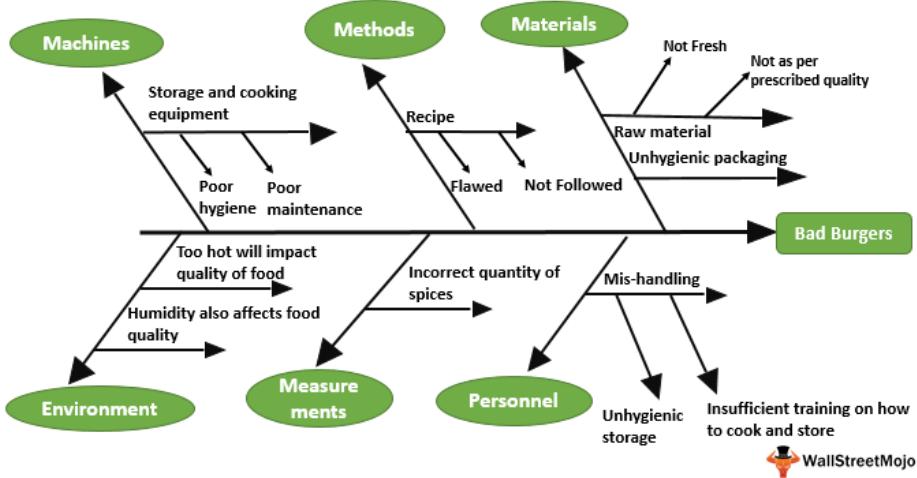
	<p>an authority at an organizational hierarchy level that this assignment pertain to only that user's organizational level and any subsidiary organizational unit levels.</p>
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Provides procedural checks and balances, as well as data security, by restricting individuals from performing certain actions. • Promotes improved review of transaction history, in that audit logs can capture details about any assigned authorities at the time. • Provides documented roles and responsibilities for activities. • Limitations <ul style="list-style-type: none"> • Need to recognize the required level of detail for a specific initiative or activity; too much detail can be time-consuming and not provide value, too little detail can exclude necessary roles or responsibilities.

10.40 Root Cause Analysis

Purpose	Root cause analysis is used to identify and evaluate the underlying causes of a problem.
Description	<ul style="list-style-type: none"> • Root cause analysis is a systematic examination of a problem or situation that focuses on the problem's origin as the proper point of correction rather than dealing only with its effects. • Root cause analysis looks at the main types of causes such as people (human error, lack of training), physical (equipment failure, poor facility), or organizational (faulty process design, poor structure). • Root cause analysis helps organize the information in a framework, which allows for deeper analysis if needed. Root cause analysis can be used for: <ul style="list-style-type: none"> • Reactive Analysis: identifying the root cause(s) of an occurring problem for corrective action • Proactive Analysis: identifying potential problem areas for preventive action. • Root cause analysis uses four main activities: <ul style="list-style-type: none"> • Problem Statement Definition: describes the issue to be addressed. • Data Collection: gathers information about the nature, magnitude, location, and timing of the effect. • Cause Identification: investigates the patterns of effects to discover the specific actions that contribute to the problem. • Action Identification: defines the corrective action that will prevent or minimize recurrence.
Elements	<ul style="list-style-type: none"> • The Fishbone Diagram <ul style="list-style-type: none"> • A fishbone diagram (also known as an Ishikawa or cause-and-effect diagram) is used to identify and organize the possible causes of a problem. • This tool helps to focus on the cause of the problem versus the solution and organizes ideas for further analysis. • The diagram serves as a map that depicts possible cause-and-effect relationships. • Steps to develop a fishbone diagram include: <ol style="list-style-type: none"> 1. Capturing the issue or problem under discussion in a box at the top of the diagram. 2. Drawing a line from the box across the paper or whiteboard (forming the spine of the fishbone). 3. Drawing diagonal lines from the spine to represent categories of potential causes of the problem. The categories may include

- people, processes, tools, and policies.
4. Drawing smaller lines to represent deeper causes.
 5. Brainstorming categories and potential causes of the problem and capturing them under the appropriate category.
 6. Analyzing the results. Remember that the group has identified only potential causes of the problem. Further analysis is needed to validate the actual cause, ideally with data.
 7. Brainstorming potential solutions once the actual cause has been identified.

Fishbone Diagram



WallStreetMojo

- The Five Whys**

- The five whys is a question asking process to explore the nature and cause of a problem.
- The five whys approach repeatedly asks questions in an attempt to get to the root cause of the problem.
- This is one of the simplest facilitation tools to use when problems have a human interaction component.
- To use this technique:
 - 1) Write the problem on a flip chart or whiteboard.
 - 2) Ask "Why do you think this problem occurs?" and capture the idea below the problem.
 - 3) Ask "Why?" again and capture that idea below the first idea.

Continue with step 3 until you are convinced the actual root cause has been identified.

- This may take more or less than five questions—the technique is called the five whys because it often takes that many to reach the root cause, not because the question must be asked five times.
- The five whys can be used alone or as part of the fishbone diagram technique.
- Once all ideas are captured in the diagram, use the five whys approach to drill down to the root causes.

Usage	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Helps to maintain an objective perspective when performing cause-
--------------	---

- Strengths**

- Helps to maintain an objective perspective when performing cause-

Considerations	<p>and-effect analysis.</p> <ul style="list-style-type: none"> Enables stakeholders to specify an effective solution at the appropriate points for corrective action. Limitations <ul style="list-style-type: none"> Works best when the business analyst has formal training to ensure the root causes, not just symptoms of the problem, are identified. May be difficult with complex problems; the potential exists to lead to a false trail and/or dead end conclusion.
-----------------------	--

10.41 Scope Modeling

Purpose	Scope models define the nature of one or more limits or boundaries and place elements inside or outside those boundaries.
Description	<ul style="list-style-type: none"> Provide the basis for understanding the boundaries of: <ul style="list-style-type: none"> Scope of Control Scope of Need Scope of Solution Scope of Change These models may show elements that include: <ul style="list-style-type: none"> In-scope Out-of-scope Both
Elements	<ul style="list-style-type: none"> Objectives <ul style="list-style-type: none"> Scope models are typically used to clarify the: span of control, relevance of elements, and where effort will be applied. Scope of Change and Context <ul style="list-style-type: none"> Business analysts are concerned with elements that will be altered as part of a change, as well as external elements that are relevant to the change. Level of Detail <ul style="list-style-type: none"> A proper level of detail provides a meaningful reduction of uncertainty while preventing 'analysis paralysis' at a scope definition stage. Relationships <ul style="list-style-type: none"> Helps to ensure completeness and integrity of the scope model by identifying their dependencies or by discovering other elements involved in or impacted by the change. Specific types of relationships <ul style="list-style-type: none"> Parent-Child or Composition-Subset Function-Responsibility Supplier-Consumer Cause-Effect Emergent Assumptions <ul style="list-style-type: none"> The definition of needs Causality of outcomes Impact of changes Applicability Feasibility of the solution Scope Modeling Results <ul style="list-style-type: none"> Textual descriptions of elements, including criteria for making in-scope or out-of scope decisions. Diagrams illustrating relationships of scope elements. Matrices depicting dependencies between scope elements.

Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • A scope model facilitates agreement as a basis for: <ul style="list-style-type: none"> • Defining contractual obligations • Estimating the project effort • Justifying in-scope/out-of-scope decisions in requirements analysis • Assessing the completeness and impact of solutions • Limitations <ul style="list-style-type: none"> • An initial, high-level model can lack a sufficient level of granularity. • Once a scope is defined, changing it may be difficult due to political reasons and contractual obligations. • Traditional scope models cannot address common complex boundaries, such as a horizon (a boundary that is completely dependent on the position of the stakeholder).
-----------------------------	--

10.42 Sequence Diagrams

Purpose	Model the logic of usage scenarios by showing the information passed between objects in the system through the execution of the scenario.
Description	<ul style="list-style-type: none"> • Show how processes or objects interact during a scenario. • Display the classes required to execute the scenario and the messages they pass to one another. • Show how objects used in the scenario interact, but not how they are related to one another. • The standard notation is defined as part of the Unified Modelling Language™ (UML®) specification.
Elements	<ul style="list-style-type: none"> • Lifeline <ul style="list-style-type: none"> • The lifespan of an object during the scenario being modelled. • Is drawn as a dashed line that vertically descends from each object box to the bottom of the page. • Activation Box <ul style="list-style-type: none"> • The period during which an operation is executed. • A call to activate is represented by an arrow with a solid arrowhead leading to the activation object. • The lifeline can be terminated with an X. • Message <ul style="list-style-type: none"> • An interaction between two objects. • Shown as an arrow coming from the activation box of the object that sends the message to the receiving object.

	<pre> sequenceDiagram participant Computer as :Computer participant Server as :Server Computer->>Computer: checkEmail activate Computer Computer->>Server: sendUnsentEmail activate Computer Server->>Computer: newEmail deactivate Computer Computer-->>Computer: response activate Computer Computer->>Computer: [newEmail] downloadEmail deactivate Computer Computer->>Computer: deleteOldEmail deactivate Computer </pre>
Usage Considerations	<ul style="list-style-type: none"> • Synchronous Call <ul style="list-style-type: none"> • Transfers control to the receiving object • The sender cannot act until a return message is received • Asynchronous Call (also known as a signal) <ul style="list-style-type: none"> • Allows the object to continue with its own processing after sending the signal • The object may send many signals simultaneously, but may only accept one signal at a time

10.43 Stakeholder List, Map or Personas

Purpose	<ul style="list-style-type: none"> • Assist the business analyst in analyzing stakeholders and their characteristics. • Important in ensuring that: <ul style="list-style-type: none"> • The business analyst identifies all possible sources of requirements • The stakeholder is fully understood so decisions made regarding stakeholder engagement, collaboration, and communication are the best choices for the stakeholder and for the success of the initiative
Description	<ul style="list-style-type: none"> • Stakeholder analysis involves identifying the stakeholders that may be affected by a proposed initiative or that share a common business need.

	<ul style="list-style-type: none"> Stakeholder analysis notes, considers, and analyzes the various characteristics of the identified stakeholders. Common types of stakeholder characteristics: <ul style="list-style-type: none"> Level of authority within the domain of change and within the organization Attitudes toward or interest in the change being undertaken Attitudes toward the business analysis work and role Level of decision-making authority 															
Elements	<ul style="list-style-type: none"> Stakeholder Lists <ul style="list-style-type: none"> Is central to both stakeholder analysis activities and the planning work the business analyst performs for elicitation, collaboration, and communication. Brainstorming and interviews are two common techniques that can be used to generate a stakeholder list. It is important to have an exhaustive list to ensure that no important stakeholder or stakeholder group has been overlooked, which opens up the risk that requirements will be missed later on. Stakeholder Map <ul style="list-style-type: none"> Diagram that depicts the relationship of stakeholders to the solution and to one another. The business analyst typically starts their stakeholder analysis by reviewing the proposed scope of the solution and then analyzing which groups will be impacted. Stakeholder analysis is considered iterative and reviewed frequently by the business analyst. Stakeholder Matrix <ul style="list-style-type: none"> Maps the level of stakeholder influence against the level of stakeholder interest <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2"></th> <th colspan="2">Impact on Stakeholder</th> </tr> <tr> <th colspan="2"></th> <th>Low</th> <th>High</th> </tr> <tr> <th rowspan="2">Influence of Stakeholder</th> <th>High</th> <td>Ensure stakeholder remains satisfied.</td> <td>Work closely with stakeholder to ensure that they are in agreement with and support the change.</td> </tr> </thead> <tbody> <tr> <th>Low</th> <td>Monitor to ensure stakeholders interest or influence do not change.</td> <td>Keep informed; stakeholder is likely to be very concerned and may feel anxious about lack of control.</td> </tr> </tbody> </table> Stakeholder Matrix: <ul style="list-style-type: none"> High Influence/High Impact: <ul style="list-style-type: none"> The stakeholders are key players in the change effort. The business analyst should focus their efforts and engage this group regularly. High Influence/Low Impact: <ul style="list-style-type: none"> The stakeholders have needs that should be met. The business analyst should engage and consult with them, while also attempting to engage them and increase their level of interest with the change activity. Low Influence/High Impact: <ul style="list-style-type: none"> The stakeholders are supporters of and potential goodwill 			Impact on Stakeholder				Low	High	Influence of Stakeholder	High	Ensure stakeholder remains satisfied.	Work closely with stakeholder to ensure that they are in agreement with and support the change.	Low	Monitor to ensure stakeholders interest or influence do not change.	Keep informed; stakeholder is likely to be very concerned and may feel anxious about lack of control.
		Impact on Stakeholder														
		Low	High													
Influence of Stakeholder	High	Ensure stakeholder remains satisfied.	Work closely with stakeholder to ensure that they are in agreement with and support the change.													
	Low	Monitor to ensure stakeholders interest or influence do not change.	Keep informed; stakeholder is likely to be very concerned and may feel anxious about lack of control.													

	<p>ambassadors for the change effort. The business analyst should engage this group for their input and show interest in their needs.</p> <ul style="list-style-type: none"> • Low Influence/Low Impact: <ul style="list-style-type: none"> • The stakeholders can be kept informed using general communications. Additional engagement may move them into the goodwill ambassador quadrant, which can help the effort gain additional support. • Onion Diagram: <ul style="list-style-type: none"> • Indicates how involved the stakeholders are with the solution, which stakeholders will directly interact with the solution or participate in a business process, which are part of the larger organization, and which are outside the organization. <pre> graph TD A[Affected External Stakeholders] --> B[Customers, suppliers, regulators, and others.] B --> C[Organization or Enterprise] C --> D[Affected Organizational Unit] D --> E[Solution Delivery] E --> F[Project team and others directly involved with creating the solution.] F --> G[End users, help desk, and others whose work changes when the solution is delivered.] G --> H[Sponsors, executives, domain SMEs, and others who interact with the affected group.] H --> I[Customers, suppliers, regulators, and others.] </pre> <ul style="list-style-type: none"> • Responsibility (RACI) Matrix <ul style="list-style-type: none"> • Responsible (R): <ul style="list-style-type: none"> • The persons who will be performing the work on the task • Accountable (A): <ul style="list-style-type: none"> • The person who is ultimately held accountable for successful completion of the task and is the decision maker. Only one stakeholder receives this assignment • Consulted (C): <ul style="list-style-type: none"> • The stakeholder or stakeholder group who will be asked to provide an opinion or information about the task. This assignment is often provided to the subject matter experts (SMEs) • Informed (I): <ul style="list-style-type: none"> • A stakeholder or stakeholder group that is kept up to date on the task and notified of its outcome. Informed is different from Consulted as with Informed the communication is one-direction (business analyst to stakeholder) and with Consulted the communication is two-way
--	---

	RACI Chart				
	Activity	Project Sponsor	Project Manager	Project Team	Department Manager
Prepare Bill of Materials			A	R	C
Prepare Estimate	I	A	R	I	
Authorize Expenditure	R	I	I	I	
Send Procurement Documents		R	C		
Evaluate Bids	A	R	C		
Perform Inspections	I	A	R		

R = Responsible A = Accountable C = Consult I = Inform

- **Personas**
 - A persona is defined as a fictional character or archetype that exemplifies the way a typical user interacts with a product.
 - Helpful when there is a desire to understand the needs held by a group or class of users.
 - Although the user groups are fictional, they are built to represent actual users.
 - The persona is written in narrative form and focuses on providing insight into the goals of the group.
 - Help bring the user to life, which in turn makes the needs feel real to those who design and build solutions.

Usage Considerations	- **Strengths** - Identifies the specific people who must be engaged in requirements elicitation activities. - Helps the business analyst plan collaboration, communication, and facilitation activities to engage all stakeholder groups. - Useful to understand changes in impacted groups over time. - **Limitations** - Business analysts who are continuously working with the same teams may not utilize the stakeholder analysis and management technique because they perceive change as minimal within their respective groups. - Assessing information about a specific stakeholder representative, such as influence and interest, can be complicated and may feel politically risky.				
10.44 State Modeling					
Purpose	Used to describe and analyze the different possible states of an entity within a system, how that entity changes from one state to another, and what can happen to the entity when it is in each state.				
Description	- In a state model (also sometimes called a state transition model), a state is a formal representation of a status. - A state model describes:				

	<ul style="list-style-type: none"> • A set of possible states for an entity • The sequence of states that the entity can be in • How an entity changes from one state to another • The events and conditions that cause the entity to change states • The actions that can or must be performed by the entity in each state as it moves through its life cycle
Elements	<ul style="list-style-type: none"> • State <ul style="list-style-type: none"> • An entity has a finite number of states during its life cycle, although it can be in more than one state at a time. • Each state is described with a name and the activities that could be performed while in that state. • There may be rules about which activities must or can be performed and which events it can respond to or trigger. • A complex state can be decomposed into sub-states. • State Transition <ul style="list-style-type: none"> • How the entity changes or transitions from one state to another could be determined by the steps of a process, by business rules, or by information content. • The sequence of states of an entity are not always linear; an entity could skip over several states or revert to a previous state, perhaps more than once. • A transition may be: <ul style="list-style-type: none"> • Conditional (triggered by a specific event or a condition being reached) or • Automatic (triggered by the completion of the required activities while in the previous state or by the passage of time) • It may also be recursive, leaving one state and returning back to the same state. • State Diagram <ul style="list-style-type: none"> • Shows the life cycle of one entity, beginning when the entity first comes into existence and moving through all of the different states that the entity may have until it is discarded and no longer of use. • A state on a state diagram is shown as a rectangle with rounded corners. There may be any number of states. A state may be decomposed into sub-states. • The transition from one state to another state is shown with a one-directional arrow pointing from the start state to the destination state. • The beginning and end of the entity's life cycle are shown with special symbols for: <ul style="list-style-type: none"> • The initial state - indicates that the entity has come into existence • The final state - indicates that the entity is discarded and the life cycle is complete

	<pre> graph TD Start(()) --> PatientRegistered[Patient Registered entry / Enter Patient Details exit / Generate Patient ID] PatientRegistered --> UndergoingTreatment[Undergoing Treatment do / Take Medicine Perform Test Present Report Undergo Operation] UndergoingTreatment -- "Doctor visits" --> Discharged[Discharged do / Payment of Bills Issue of Discharge Ticket] UndergoingTreatment -- "Review from Doctor" --> UndergoingTreatment Discharged --> End(()) UndergoingTreatment -- "Issue of Discharge advice from Doctor" --> Discharged </pre> <p>The diagram illustrates a state transition process for a patient. It begins with a start state (empty circle) leading to the 'Patient Registered' state. From 'Patient Registered', the process can either 'entry / Enter Patient Details' or 'exit / Generate Patient ID'. This leads to the 'Undergoing Treatment' state. In 'Undergoing Treatment', the patient can 'Take Medicine', 'Perform Test', 'Present Report', or 'Undergo Operation'. A 'Doctor visits' transition leads to the 'Discharged' state, where the patient can 'Payment of Bills' and 'Issue of Discharge Ticket'. A 'Review from Doctor' feedback loop returns to the 'Undergoing Treatment' state. Finally, the patient reaches the end state (empty circle).</p> <ul style="list-style-type: none"> State Tables <ul style="list-style-type: none"> A state table is a two-dimensional matrix showing states and the transitions between them. Each row shows a starting state, the transition, and the end state. If one state could respond to several transitions, there will be a separate row for each transition. A state that appears as an end state in one row could be a start state in another row
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Identifies business rules and information attributes that apply to the entity being modelled. Identifies and describes the activities that apply to the entity at different states of the entity. Is a more effective documentation and communication tool than plain text, especially if the entity being described has more than a few states, transitions, and conditions governing those transitions. Limitations <ul style="list-style-type: none"> Is usually only used to understand and communicate about information entities that are perceived to be complex; simple entities may be understood without the time and effort required to build a state model. Building a state model appears simple at the start, but achieving a consensus among domain SMEs about the details required by the model can be difficult and time consuming. A high degree of precision about states and transitions is required to build a state diagram; some domain SMEs and business analysis practitioners are uncomfortable trying to describe such a level of detail.

10.45 Survey or Questionnaire

Purpose	Used to elicit business analysis information - including information about customers, products, work practices, and attitudes - from a group of people in a structured way and in a relatively short period of time.
Description	<ul style="list-style-type: none"> A survey or questionnaire presents a set of questions to stakeholders and subject matter experts (SMEs), whose responses are then collected and analyzed in order to formulate knowledge about the subject matter of interest. The questions can be submitted in written form or can be administered in

	<p>person, over the telephone, or using technology that can record responses.</p> <ul style="list-style-type: none"> Questions should be asked in a way that does not influence the response data. Close-ended questions: <ul style="list-style-type: none"> The respondent is asked to select from a list of predefined responses, such as a yes/no response, a multiple-choice selection, a rank/order decision, or a statement requiring a level of agreement Useful when the anticipated range of user responses is fairly well defined and understood Open-ended questions : <ul style="list-style-type: none"> The respondent is asked to answer questions in a free form without having to select an answer from a list of predefined responses Useful when the issues are known and the range of user responses is not May result in more detail and a wider range of responses than closed-ended questions
Elements	<ul style="list-style-type: none"> Prepare <ul style="list-style-type: none"> An effective survey or questionnaire requires detailed planning in order to ensure that the needed information is obtained in an efficient manner. When preparing for a survey or questionnaire, business analysts do the following: <ul style="list-style-type: none"> Define the objective Define the target survey group Choose the appropriate survey or questionnaire type Select the sample group Select the distribution and collection methods Set the target level and timeline for response Determine if the survey or questionnaire should be supported with individual interviews Write the survey questions Test the survey or questionnaire Distribute the Survey or Questionnaire <ul style="list-style-type: none"> When distributing the survey or questionnaire it is important to communicate: <ul style="list-style-type: none"> The survey's objectives How its results will be used, Any arrangements for confidentiality or anonymity that have been made When deciding on a method of distribution (for example, in-person, e-mail, or survey tool), business analysts consider: <ul style="list-style-type: none"> The urgency of obtaining the results The level of security required The geographic distribution of the respondents Document the Results <ul style="list-style-type: none"> When documenting the results of the survey or questionnaire, business analysts: <ul style="list-style-type: none"> Collate the responses Summarize the results Evaluate the details and identify any emerging themes Formulate categories for encoding the data Break down the data into measurable increments

Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Quick and relatively inexpensive to administer. • Easier to collect information from a larger audience than other techniques such as interviews. • Does not typically require significant time from the respondents. • Effective and efficient when stakeholders are geographically dispersed. • When using closed-ended questions, surveys can be effective for obtaining quantitative data for use in statistical analysis. • When using open-ended questions, survey results may yield insights and opinions not easily obtained through other elicitation techniques. • Limitations <ul style="list-style-type: none"> • To achieve unbiased results, specialized skills in statistical sampling methods are needed when surveying a subset of potential respondents. • The response rates may be too low for statistical significance. • Use of open-ended questions requires more analysis. • Ambiguous questions may be left unanswered or answered incorrectly. • May require follow-up questions or more survey iterations depending on the answers provided.
-----------------------------	---

10.46 SWOT Analysis

Purpose	<p>SWOT analysis is a simple yet effective tool used to evaluate an organization's strengths, weaknesses, opportunities, and threats to both internal and external conditions.</p>
Description	<ul style="list-style-type: none"> • Used to identify the overall state of an organization both internally and externally. • The language used is brief, specific, realistic, and supported by evidence. • Serves as an evaluation of an organization against identified success factors. • Can be performed at any scale from the enterprise as a whole to a division, a business unit, a project, or even an individual. • A SWOT analysis can be used to: <ul style="list-style-type: none"> • Evaluate an organization's current environment • Share information learned with stakeholders • Identify the best possible options to meet an organization's needs • Identify potential barriers to success and create action plans to overcome barriers • Adjust and redefine plans throughout a project as new needs arise • Identify areas of strength that will assist an organization in implementing new strategies • Develop criteria for evaluating project success based on a given set of requirements • Identify areas of weakness that could undermine project goals • Develop strategies to address outstanding threats
Elements	<ul style="list-style-type: none"> • Specify the main points <ul style="list-style-type: none"> • Strengths (S): <ul style="list-style-type: none"> • Anything that the assessed group does well. May include experienced personnel, effective processes, IT systems, customer relationships, or any other internal factor that leads to success • Weaknesses (W): <ul style="list-style-type: none"> • Actions or functions that the assessed group does poorly or not

	<p>at all</p> <ul style="list-style-type: none"> Opportunities (O): <ul style="list-style-type: none"> External factors of which the assessed group may be able to take advantage. May include new markets, new technology, changes in the competitive marketplace, or other forces Threats (T): <ul style="list-style-type: none"> External factors that can negatively affect the assessed group. They may include factors such as the entrance into the market of a new competitor, economic downturns, or other forces <p>Example:</p> <table border="1"> <tbody> <tr> <td>The organisation</td><td>Strengths – S 1. Existing brand 2. Existing customer base 3. Existing distribution</td><td>Weaknesses – W 1. Brand perception 2. Intermediary use 3. Technology/skills 4. X-channel support</td></tr> <tr> <td>Opportunities – O 1. Cross-selling 2. New markets 3. New services 4. Alliances/Co-branding</td><td>SO strategies Leverage strengths to maximise opportunities = Attacking strategy</td><td>WO strategies Counter weaknesses through exploiting opportunities = Build strengths for attacking strategy</td></tr> <tr> <td>Threats – T 1. Customer choice 2. New entrants 3. New competitive products 4. Channel conflicts</td><td>ST strategies Leverage strengths to minimise threats = Defensive strategy</td><td>WT strategies Counter weaknesses and threats = Build strengths for defensive strategy</td></tr> </tbody> </table>	The organisation	Strengths – S 1. Existing brand 2. Existing customer base 3. Existing distribution	Weaknesses – W 1. Brand perception 2. Intermediary use 3. Technology/skills 4. X-channel support	Opportunities – O 1. Cross-selling 2. New markets 3. New services 4. Alliances/Co-branding	SO strategies Leverage strengths to maximise opportunities = Attacking strategy	WO strategies Counter weaknesses through exploiting opportunities = Build strengths for attacking strategy	Threats – T 1. Customer choice 2. New entrants 3. New competitive products 4. Channel conflicts	ST strategies Leverage strengths to minimise threats = Defensive strategy	WT strategies Counter weaknesses and threats = Build strengths for defensive strategy
The organisation	Strengths – S 1. Existing brand 2. Existing customer base 3. Existing distribution	Weaknesses – W 1. Brand perception 2. Intermediary use 3. Technology/skills 4. X-channel support								
Opportunities – O 1. Cross-selling 2. New markets 3. New services 4. Alliances/Co-branding	SO strategies Leverage strengths to maximise opportunities = Attacking strategy	WO strategies Counter weaknesses through exploiting opportunities = Build strengths for attacking strategy								
Threats – T 1. Customer choice 2. New entrants 3. New competitive products 4. Channel conflicts	ST strategies Leverage strengths to minimise threats = Defensive strategy	WT strategies Counter weaknesses and threats = Build strengths for defensive strategy								

Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Is a valuable tool to aid in understanding the organization, product, process, or stakeholders. Enables business analysts to direct the stakeholders' focus to the factors that are important to the business. Limitations <ul style="list-style-type: none"> The results of a SWOT analysis provide a high-level view; more detailed analysis is often needed. Unless a clear context is defined for the SWOT analysis the result may be unfocused and contain factors which are not relevant to the current situation.
-----------------------------	--

10.47 Use Cases and Scenarios

Purpose	Use cases and scenarios describe how a person or system interacts with the solution being modelled to achieve a goal.
Description	<ul style="list-style-type: none"> Use cases describe the interactions between the primary actor, the solution, and any secondary actors needed to achieve the primary actor's goal. Are usually triggered by the primary actor, but in some methods may also be triggered by another system or by an external event or timer. A use case describes the possible outcomes of an attempt to accomplish a particular goal that the solution will support. It details different paths that can be followed by defining primary and alternative flows.

	<ul style="list-style-type: none"> The primary or basic flow represents the most direct way to accomplish the goal of the use case. Special circumstances and exceptions that result in a failure to complete the goal of the use case are documented in alternative or exception flows. Use cases are written from the point of view of the actor and avoid describing the internal workings of the solution. Use case diagrams are a graphical representation of the relationships between actors and one or more use cases supported by the solution. A scenario describes just one way that an actor can accomplish a particular goal. Scenarios are written as a series of steps performed by actors or by the solution that enable an actor to achieve a goal. A use case describes several scenarios.
Elements	<ul style="list-style-type: none"> Use Case Diagram <ul style="list-style-type: none"> Visually depicts the scope of the solution, by showing the actors who interact with the solution, which use cases they interact with, and any relationships between the use cases. <pre> graph LR System["System"] --> UC1([Use Case 1]) System --> UC2([Use Case 2]) System --> UC3([Use Case 3]) System --> UC4([Use Case 4]) Actor1((Actor 1)) --- UC1 Actor2((Actor 2)) --- UC2 UC1 -.-> UC2 UC3 -.-> UC2 subgraph UC2_Box [Extension points: Call Use Case 3] UC2 end </pre> <ul style="list-style-type: none"> Relationships between actors and use cases <ul style="list-style-type: none"> Are called associations An association line indicates that an actor has access to the functionality represented by the use case Associations do not represent input, output, time, or dependency Relationships between use cases <ul style="list-style-type: none"> Extend Allows for the insertion of additional behavior into a use case The use case that is being extended must be completely functional in its own right and must not depend on the extending use case for its successful execution Include <ul style="list-style-type: none"> Allows for the use case to make use of functionality present in another use case The included use case does not need to be a complete use case in its own right if it is not directly triggered by an actor Use Case Description <ul style="list-style-type: none"> Name <ul style="list-style-type: none"> The use case has a unique name

	<ul style="list-style-type: none"> • The name generally includes a verb that describes the action taken by the actor and a noun that describes either what is being done or the target of the action • Goal <ul style="list-style-type: none"> • The goal is a brief description of a successful outcome of the use case from the perspective of the primary actor • Actors <ul style="list-style-type: none"> • An actor is any person or system external to the solution that interacts with that solution • Each actor is given a unique name that represents the role they play in interactions with the solution. • Some use case authoring approaches recommend against the use of systems or events as actors • Preconditions <ul style="list-style-type: none"> • Any facts that must be true before the use case can begin • Are not tested in the use case but act as a constraint on its execution • Trigger <ul style="list-style-type: none"> • An event that initiates the flow of events for a use case • The most common trigger is an action taken by the primary actor • A temporal event (for example, time) can initiate a use case. This is commonly used to trigger a use case that must be executed based on the time of day or a specific calendar date, such as an end-of-day routine or an end-of-month reconciliation of a system • Flow of Events <ul style="list-style-type: none"> • Is the set of steps performed by the actor and the solution during the execution of the use case • A basic, primary, or main success flow represents the shortest or simplest successful path that accomplishes the goal of the actor • Alternative flows describe other paths that may be followed to allow the actor to successfully achieve the goal of the use case • Exception flows describe the desired response by the solution when the goal is unachievable and the use case cannot be successfully completed • Post-conditions or Guarantees <ul style="list-style-type: none"> • Are any facts that must be true when the use case is complete • Must be true for all possible flows through the use case, including both the primary and alternative flows • The use case may describe separate post-conditions that are true for successful and unsuccessful executions of the use case
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Use case diagrams can clarify scope and provide a high-level understanding of requirements. • Use case descriptions are easily understood by stakeholders due to their narrative flow. • The inclusion of a desired goal or outcome ensures that the business value of the use case is articulated.

	<ul style="list-style-type: none"> • Use case descriptions articulate the functional behavior of a system. <p>Limitations</p> <ul style="list-style-type: none"> • The flexibility of the use case description format may lead to information being embedded that would be better captured using other techniques such as user interface interactions, non-functional requirements, and business rules. • The flexible format of use cases may result in capturing inappropriate or unnecessary detail in the attempt to show every step or interaction. • Use cases intentionally do not relate to the design of the solution and as a result, significant effort may be required in development to map use case steps to software architecture.
--	--

10.48 User Stories

Purpose	A user story represents a small, concise statement of functionality or quality needed to deliver value to a specific stakeholder.
Description	<ul style="list-style-type: none"> • Capture the needs of a specific stakeholder and enable teams to define features of value to a stakeholder using short, simple documentation. • Serve as a basis for identifying needs and allow for the prioritizing, estimating, and planning of solutions. • Is typically a sentence or two that describes who has the need addressed by the story, the goal the user is trying to accomplish, and any additional information that may be critical to understanding the scope of the story. • User stories can be used <ul style="list-style-type: none"> • To capture stakeholder needs and prioritize development of solutions, • As a basis of estimating and planning solution delivery • As a basis for generating user acceptance tests • As a metric for measuring the delivery of value • As a unit for tracing related requirements • As a basis for additional analysis • As a unit of project management and reporting
Elements	<ul style="list-style-type: none"> • Title (optional) <ul style="list-style-type: none"> • Describes an activity the stakeholder wants to carry out with the system. Typically, it is an active-verb goal phrase similar to the way use cases are titled. • Statement of Value <ul style="list-style-type: none"> • The most popular format includes three components: <ul style="list-style-type: none"> • Who: a user role or persona • What: a necessary action, behavior, feature, or quality • Why: the benefit or value received by the user when the story is implemented • For example, "As a <who>, I need to <what>, so that <why>". "Given...When...Then" is another common format • Conversation <ul style="list-style-type: none"> • User stories help teams to explore and understand the feature described in the story and the value it will deliver to the stakeholder. • The story itself doesn't capture everything there is to know about the stakeholder need and the information in the story is supplemented by further modelling as the story is delivered. • Acceptance Criteria <ul style="list-style-type: none"> • A user story may be supported through the development of detailed acceptance criteria. • Acceptance criteria define the boundaries of a user story and help the

	<p>team to understand what the solution needs to provide in order to deliver value for the stakeholders.</p>
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Easily understandable by stakeholders. • Can be developed through a variety of elicitation techniques. • Focuses on value to stakeholders. • A shared understanding of the business domain is enhanced through collaboration on defining and exploring user stories. • Tied to small, implementable, and testable slices of functionality, which facilitates rapid delivery and frequent customer feedback. • Limitations <ul style="list-style-type: none"> • In general, user stories are intended as a tool for short-term capture and prioritization of requirements and not for long-term knowledge retention or to provide a detailed analysis. • Neglecting this principle can lead to the following issues: <ul style="list-style-type: none"> • This conversational approach can challenge the team since they do not have all the answers and detailed specifications upfront. • Requires context and visibility; the team can lose sight of the big picture if stories are not traced back through validation or supplemented with higher-level analysis and visual artifacts. • May not provide enough documentation to meet the need for governance, a baseline for future work, or stakeholder expectations. Additional documentation may be required.

10.49 Vendor Assessment

Purpose	A vendor assessment assesses the ability of a vendor to meet commitments regarding the delivery and the consistent provision of a product or service.
Description	<ul style="list-style-type: none"> • A vendor assessment is conducted to ensure that the vendor is reliable and that the product and service meet the organization's expectations and requirements. • The assessment may be formal through the submission of: <ul style="list-style-type: none"> • Request for Information (RFI) • Request for Quote (RFQ) • Request for Tender (RFT) • Request for Proposal (RFP) • It may be informal through word of mouth and recommendations.
Elements	<ul style="list-style-type: none"> • Knowledge and Expertise <ul style="list-style-type: none"> • A common reason for using third-party vendors is that they can provide knowledge and expertise not available within the organization. • Licensing and Pricing Models <ul style="list-style-type: none"> • The licensing or pricing model is taken into account in cases where a solution or solution component is purchased from or outsourced to a third party vendor. • Vendor Market Position <ul style="list-style-type: none"> • Comparing each vendor with the competitors and decide with which market players the organization wants to get involved • Terms and Conditions <ul style="list-style-type: none"> • Refer to the continuity and integrity of the provided products and services. • The organization investigates whether the vendor's licensing terms, intellectual property rights and technology infrastructure are likely to

	<p>turn into challenges if the organization later chooses to transition to another supplier.</p> <ul style="list-style-type: none"> Vendor Experience, Reputation and Stability <ul style="list-style-type: none"> Vendors' experience with other customers may provide valuable information on how likely it is that they will be able to meet their contractual and non-contractual obligations. Vendors can also be evaluated for conformance and compliance with external relevant standards for quality, security, and professionalism.
Usage Considerations	<ul style="list-style-type: none"> Strengths <ul style="list-style-type: none"> Increases the chances of the organization to develop a productive and fair relationship with a suitable and reliable vendor, and to improve long-term satisfaction with the decision. Limitations <ul style="list-style-type: none"> May be consuming in regards to time and resources. Does not prevent risk of failure as the partnership evolves. Subjectivity may bias the evaluation outcome.
10.50 Workshops	
Purpose	Workshops bring stakeholders together in order to collaborate on achieving a predefined goal.
Description	<ul style="list-style-type: none"> Is a focused event attended by key stakeholders and subject matter experts (SMEs) for a concentrated period of time. May be held for different purposes including planning, analysis, design, scoping, requirements elicitation, modelling, or any combination of these. May be used to generate ideas for new features or products, to reach consensus on a topic, or to review requirements or design Workshops generally include: <ul style="list-style-type: none"> A representative group of stakeholders A defined goal Interactive and collaborative work A defined work product A facilitator The workshop is ideally facilitated by an experienced, neutral facilitator; however, a team member may also serve as the facilitator. A scribe documents the decisions reached and any outstanding issues. A business analyst may be the facilitator or the scribe in these workshops.
Elements	<ul style="list-style-type: none"> Prepare for the Workshop <ul style="list-style-type: none"> When preparing for a workshop, business analysts: <ul style="list-style-type: none"> Define the purpose and desired outcomes, Identify key stakeholders to participate, Identify the facilitator and scribe, Create the agenda, Determine how the outputs will be captured, Schedule the session and invite the participants, Arrange room logistics and equipment, Send the agenda and other materials in advance to prepare the attendees and increase productivity at the meeting, and If appropriate, conduct pre-workshop interviews with participants. Workshop Roles <ul style="list-style-type: none"> Sponsor Frequently not a participant in the workshop, but does have

	<p style="text-align: center;">ultimate accountability for its outcome</p> <ul style="list-style-type: none"> • Facilitator <ul style="list-style-type: none"> • Establishes a professional and objective tone for the workshop • Introduces the goals and agenda for the workshop • Enforces structure and ground rules • Keeps activities focused on the purpose and desired outcomes • Facilitates decision making and conflict resolution • Ensures that all participants have an opportunity to be heard • Workshop Roles <ul style="list-style-type: none"> • Scribe <ul style="list-style-type: none"> • Documents the decisions in the format determined prior to the workshop • Keeps track of any items or issues that are deferred during the session • Timekeeper <ul style="list-style-type: none"> • May be used to keep track of the time spent on each agenda item. • Participants <ul style="list-style-type: none"> • Includes key stakeholders and subject matter experts • Are responsible for providing their input and views, listening to other views, and discussing the issues without bias. • Conduct the Workshop <ul style="list-style-type: none"> • Facilitators generally begin the workshop with a statement of its purpose and desired outcomes. • Establishing agreed-upon ground rules can be an effective method for establishing a productive environment for collaboration. • Throughout the workshop, the facilitator maintains focus by frequently validating the session's activities with the workshop's purpose and outcomes. • Post Workshop Wrap-Up <ul style="list-style-type: none"> • Facilitator follows up on any open action items that were recorded at the workshop, completes the documentation, and distributes it to the workshop attendees and any stakeholders who need to be kept informed of the work done.
Usage Considerations	<ul style="list-style-type: none"> • Strengths <ul style="list-style-type: none"> • Can be a means to achieve agreement in a relatively short period of time. • Provides a means for stakeholders to collaborate, make decisions, and gain a mutual understanding. • Costs are often lower than the cost of performing multiple interviews. • Feedback on the issues or decisions can be provided immediately by the participants. • Limitations <ul style="list-style-type: none"> • Stakeholder availability may make it difficult to schedule the workshop. • The success of the workshop is highly dependent on the expertise of the facilitator and knowledge of the participants. • Workshops that involve too many participants can slow down the workshop process. • Conversely, collecting input from too few participants can lead to the overlooking of needs or issues that are important to some stakeholders, or to the arrival at decisions that don't represent the needs of the majority of the stakeholders.

