Requirements Engineering
BCS Certificate in Requirements Engineering

RE-2 v7.6
Contents

How to use this workbook ............................................................... 3
Introduction .................................................................................. 5
Module 1 – Requirements Engineering in Context ......................... 12
Module 2 – Requirements Stakeholders and Elicitation ................... 43
Module 3 – Organising and Analysing Requirements ...................... 91
Module 4 – Use of Models in Requirements Engineering ................ 130
Module 5 – Requirements Documentation .................................... 170
Module 6 – Requirements Validation and Management ................. 196
Examination Hints and Tips ......................................................... 215
How to use this workbook

Activity
Alongside this icon you will find details of the group/individual activity or a point for everyone to discuss.

Definition
Where a word with a very specific definition (or one that could be described as jargon) is introduced this will highlight that a definition is provided.

Exam tips
This icon indicates an examinable technique or will highlight information that you may find useful in the exam.

Expansion materials
This manual contains examinable materials. The QA++ symbol contains further information. Skip over these during class. They are not needed for the examination.

Glossary
Definition of a term.

Helpful hint
This icon guides you to tips or hints that will help you avoid the standard pitfalls that await the unwary practitioner or to show you how you might increase your effectiveness or efficiency in practising what you have learnt.
Important idea or concept

Generally this icon is used to draw your attention to ideas that you need to understand by this point in the course. Let your trainer know if you do not understand or see the relevance of this idea or concept.

Key point

This icon is used to indicate something that practitioners in this field should know. It is likely to be one of the major things to remember from the course, so check you do understand these key points.

Reference material

When we have only touched briefly on a topic this icon highlights where to look for additional information on the subject. It may also be used to draw your attention to International or National Standards or Web addresses that have interesting collections of information.

Reinforcement

From time to time, there will be places within the course where it is useful for you to reinforce your understanding. This might be in the form of a question to ponder or a short end-of-module test.

Useful tool

This icon indicates a technique that will help you put what you have learnt into action.

Warning

This icon is used to point out important information that may affect you and your use of the product or service in question.
Introduction

Welcome to QA’s Requirements Engineering course. During the next few days, you will learn the skills, techniques and knowledge required to attempt the BCS examination in this subject. Full details of the syllabus are at:

http://certifications.bcs.org/upload/pdf/ba-re-syllabus.pdf

Course Administration

Before we begin the course, your instructor needs to take you through a number of administrative points as shown below.

- Safety
- Timings
- Breaks/Meals
- Rooms
- Security
BCS Course Objectives

Holders of the BCS Certificate in Requirements Engineering should be able to:

- Explain the importance of linking requirements to the Business Case
- Describe the roles and responsibilities of key stakeholders in the requirements engineering process
- Explain the use of a range of requirements elicitation techniques and the relevance of the techniques to business situations
- Analyse, prioritise and organise elicited requirements
- Document requirements
- Identify problems with requirements and explain how requirements documentation may be improved
- Create a model of the features required from a system
- Interpret a model of the data requirements for an information system
- Describe the principles of Requirements Management and explain the importance of managing requirements
- Describe the use of tools to support Requirements Engineering
- Explain the process and stakeholders involved in Requirements Validation
BCS Requirements Engineering Exam

- 1-1.5 pages
- Includes some requirements

Case study

- Course manual
- Notes
- Exercises
- Solutions

Open book

- No writing
- No highlighting
- No applying or moving sticky tabs

15 minutes’ reading time

- 50 marks available
- A minute per mark
- 50% to pass

60 minutes’ writing time

- Read these first
- Answers specific to the case study
- Be concise

4 or 5 questions

- All delegates must produce photo ID before the exam (BCS’ rules)

Photo ID
## BCS Business Analysis Diploma

<table>
<thead>
<tr>
<th>Core</th>
<th>Knowledge-based Specialism</th>
<th>Practitioner Specialism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Analysis Practice</td>
<td>Commercial Awareness</td>
<td>Modelling Business Processes</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>Foundation Certificate in IS Project Management</td>
<td>Systems Modelling Techniques</td>
</tr>
<tr>
<td></td>
<td>Foundation Certificate in Business Analysis</td>
<td>Systems Development Essentials</td>
</tr>
<tr>
<td></td>
<td>Foundation Certificate in Business Change</td>
<td>Benefits Management and Business Acceptance</td>
</tr>
</tbody>
</table>

*Both of the above plus* 

*1 of the above and* 

*1 of the above*
BCS Oral Examination

**An Oral Examination is required to complete the Solution Development or Business Analysis Diploma**

- Can be booked once all required written exams have been passed
- Must be taken within 12 months of the written notification of passing the final exam

**2 BCS Examiners, 50 minute interview**

- Questions range over the latest BCS syllabus for the relevant oral exam
- Each Oral has its own syllabus, and so may cover topics additional to those covered in the written examination modules taken **

**Review BCS 'Candidate Guidelines'**

- Review the 'Candidate Guidelines' available from the BCS covering the Oral Examinations

**Oral Preparation Workshops**

- Attendance at a Preparation Workshop covering all the relevant syllabus topics from an 'oral examination' perspective is recommended, but not mandatory

** see the candidate guidelines and syllabus on the BCS website**
Recommended Reading References

Title: Business Analysis (3rd Edition)
Author: Debra Paul, Donald Yeates and James Cadle
Publisher: BCS
Publication Date: 2014
ISBN: 9781780172774
URL: http://shop.bcs.org

Title: Mastering the Requirements Process
Author: Suzanne Robertson and James Robertson
Publisher: Addison Wesley
Publication Date: 1999
ISBN: 978-0201360462

Title: Writing Effective Use Cases
Author: Alistair Cockburn
Publisher: Addison-Wesley
Publication Date: October 2000
ISBN: 0201702258
Title: User Stories Applied: For Agile Software Development
Author: Mike Cohn
Publisher: Addison Wesley
Publication Date: March 2004
ISBN: 9780321205681

Title: Requirements Engineering: Processes and Techniques
Author: Gerald Kotonya and Ian Sommerville
Publisher: John Wiley & Sons
Publication Date: April 1998
ISBN: 0471972088

Also see the RE syllabus for further and recommended reading.
Module 1 – Requirements Engineering in Context

Topics

In this section of the course, we will cover:

- A rationale for Requirements Engineering
- Problems with Requirements
- Requirements and the Business Case/ToR/PID
- Importance of Requirements planning and estimating
- A framework for Requirements Engineering
What do you think?

How would you define ‘Requirement’?
What is a Requirement?

Note that the following definitions cover both the user and the developer viewpoint.

IEEE (Standard Glossary, 610.12-1990)

“A condition or capability:

- Needed by a user to solve a problem or achieve an objective that must be met by a system or system component to satisfy a contract, standard, specification or other formally imposed document
- A documented representation of a condition or capability”

BCS ‘Business Analysis’ Glossary definition

“A feature that the business users needs the new system (business or IT) to provide”

Sommerville and Sawyer

“… a specification of what should be implemented. Descriptions of how the system should behave, or of a system property or attribute. May be a constraint on the development process”.

Requirements and Business as Usual (BaU)

The Business Change is intended to be supported by the IT system or solution created by the project.

In this course we will focus on the IT requirements that make up the IT system, which support the business, people, and processes in Business as Usual (BaU).

The diagram above shows that the requirements are central to successfully moving the business from an as-is state to the future state, or to-be.

Requirements are not related solely to a project, they reflect the overall needs of the business and must be implemented, via the IT solution, in a way that supports the business and addresses the issues they were experiencing.

The change (the resolution of whatever issue is being addressed) itself is facilitated by the project, which temporarily takes it away from
business as usual for the duration of the project, however long this may be.

As the project progresses, the business is still moving towards its business objectives and strategic goals, which could change and impact project scope and budget. It is important therefore to ensure that these business objectives are kept in sight, reviewed and that the requirements are clearly linked to them.

Not every set of requirements is relevant to an IT system or new software requirements.

Remember too that there can be multiple change projects arising from the need to resolve a business issue and therefore many links back to business objectives.
Origin of Requirements

Businesses are constantly changing and are faced with new threats and new problems, demand from the customers and fresh opportunities.

Requirements come from a need to address an issue within the current business environment.

They can come from pretty much anywhere but typically arise due to:

- Business Changes
  - Operational changes
  - Business process changes
- Strategic Changes
  - These are covered in the Business Analysis Practice course
- New business, products, business rules or regulations
- Opportunities for improvement, enhancements etc.

Requirements will support the business needs and are fundamental to successful change.
What do you think?

What problems have you encountered with requirements?
Common problems in Requirements Engineering

Requirements are notoriously difficult to write well.

Requirements specifications range from a scrawl on the back of an envelope to multiple volumes, and all points in between. Rarely does a team or organisation express the opinion that their method of producing them, or that the contents when produced, are ideal.

The customer, users and developers all think they know which functionality is to be included, but their understandings are all somewhat different, resulting in expectations that are unlikely to be met.

Some common problems are:

- Problem not well defined
- Users unsure what is needed
- Ambiguous expressions
- Contradiction between requirements
- Conflict between requirements
- Overlapping and duplicate requirements
- Unrealistic requirements
- Not testable
- Solutionising
- Missing requirements
- Not linked to the business objective
Module 1: Requirements Engineering in Context

Origin of Errors

% Origin of conceptual errors found at acceptance testing – James Martin

Poor Requirements are Expensive

Requirements definition is the key activity of analysis and most people are aware of the importance of getting them right. Yet we still have the situation where almost 60% of the faults found in user acceptance testing are caused by errors occurring during analysis and specifically during requirements capture.

The data in the graph above was first correlated in the 1980s and has remained true to this day. It indicates failures in:

- Capturing the correct requirements
- Analysing, correlating, negotiating the requirements
- Validating the requirements in formal reviews
It also indicates failures in the management of requirements including base-lining, change control, identification, tracking etc.

In other words, the graph indicates poor Requirements Engineering.

**Correction Cost**

If most errors are errors in the requirements engineering process, it follows that, if we could find those errors during analysis, we could correct them at a relatively low cost.

Conversely, if the requirements errors are allowed to propagate through design and build and into operation they will become exponentially more costly to fix.

*Relative cost of correcting errors – Barry Boehm*

Deploy all of the tools, communications skills, analysis skills, inspection skills and iterative activities needed to get the requirements ‘right’ and to find any errors early.
That is, introduce a defined Requirements Engineering process.

Although the majority of errors are introduced at the analysis stage, the BCS book specifies that 12% of project time is allocated to requirements analysis, and often delivered systems do not properly match business objectives as a result.
Planning and Estimating

Requirements engineering, just like any other project activity, needs careful planning…

Why?

Requirements can:

- Form the basis of project estimates
- Clarify scope
- Reveal more about the problem, the people, the processes, the business rules
- Rationalise at a task level why the problem needs addressing
- Take time to uncover

A balance must be struck between investing enough time in requirements engineering, but guarding against the dreaded ‘paralysis by analysis’

Project Managers and Sponsors will require an estimate of the number of days’ effort and elapsed time so that the project can be costed and scheduled.

- Requirements are often seen as being ‘just’ a project deliverable; a set of guidelines that drive a solution
- We have already seen what can happen if the analysis stage is not properly considered in the graphs earlier in this section
- To enable this to be done properly the right amount of time is needed to plan and execute the requirements

Project constraints, such as time, budget, resource availability and time are likely to be considered as criteria for shaping the project and often separately from the business objectives that frame the solution.

Ultimately though, realising the business benefits and fulfilling the business objectives is the lasting impression that the project makes, long after the project has been closed down.
Sometimes, in the rush to scope the project, delivery of the solution is mistakenly seen as the end goal, when in fact implementing a solution to solve a business problem is the end goal, and the users are the ones left picking up the pieces of a failed project.

Planning and estimating are therefore key activities that help identify the tasks that will be required to ensure that the resulting solution, delivered by the project, is fit for purpose.

*Use Case Function Points, a recycling of the well-known ‘Function Point Analysis’, technique can give guidance to the estimating activity:*

[http://www.softwaremetrics.com/Articles/using.htm](http://www.softwaremetrics.com/Articles/using.htm)
The importance of planning

- Many different stakeholders may be involved in a Business situation, in the development of a suitable supporting system, and they may have subtly (or not so subtly) different ideas of what is required
- Often it is unclear what is actually needed – even the problem owner may be unsure
- Users can find it hard to articulate what is wrong with their business area and will often describe the symptoms of the problem in relation to how they ‘see’ the solution
  - In most cases the restricted view that stakeholders have of the problem limits their understanding of the relative complexity required to make it work better

It is the Business Analyst’s job to help the stakeholders to communicate and clarify their understanding of the requirements as efficiently as possible. The Business Analyst needs time to do this properly.
A further rationale for Requirements Engineering

Requirements Engineering is focused on defining the characteristics of the product(s) that will solve the business problems and therefore facilitate the needs of the business towards making money.

Without adequate requirements:

- We may end up with the wrong solution
- It might be difficult to evaluate and rationalise alternative solutions
- We may find it difficult to interpret or understand the solution in the future
- It might be difficult to change the solution in the future

Whatever solution we advise the business to adopt, and however big or small, we should keep in sight how that change will be measured and tracked through the project or programme.

We can also plan for how we measure the transition, through implementation, into Business as Usual (BaU).
Solutions are not Requirements

Solutions define how the requirements will be met, and requirements should be solution agnostic.

There may be many solutions available for a given requirement:

- We can’t choose or configure a solution to a problem unless we understand the requirements (and the business objectives that drive the business need)
- A complete set of requirements should narrow down the list of suitable solutions, which can be further refined by using the business objectives to prioritise

Solutions can constrain the requirement and limit the possibilities for how it could be met. For example, if we need to be able to tell the time, we want to keep our options open for how the requirement is met, and therefore the solutions could include:

- A wrist watch
- Checking a mobile phone
- A clock
- Asking an intelligent personal assistant (such as Apple’s ‘Siri’)
- Understanding the position of the sun

We wouldn’t build a house before we have considered the plans, materials, the number of rooms, the potential market for purchase, the budget etc.
Characteristics of ‘good’ Requirements

- They address a well-defined problem
- They specify features that stakeholders need
- Where possible, requirements expressions are SMART:
  - Specific: specific features are defined without ambiguity and without solution-orientated language
  - Measurable: we will be able to measure their delivery in the solution
  - Achievable: can be realised in the solution within time, cost, risk constraints
  - Realistic: technically feasible or Relevant (relevant to the problem)
  - Time-bound: we know when they will be delivered or Testable (each requirement is testable and we know how to test it)
- A compromise has been reached over any conflicts between requirements
- Requirements are atomic and self-contained
- Requirements have been assessed and linked to business objectives (and the Business as Usual)
Good Requirements Checklist

There are multiple ways to write requirements. This course doesn’t go into the detail of constructing the requirements. However, it is possible to apply some good practice to your requirements.

To be considered “good”, a requirement should be:

**Cohesive** – it must address only one thing.

**Complete** – all the information about it should be in that one place and nothing should be missing.

**Consistent** – it should not contradict other requirements or documentation.

**Atomic** – it must not contain more than one requirement, indicated by the use of words such as “and” or “or”.

**Traceable** – we should be able to trace this back to the documented needs of the stakeholders and forwards to the solution.

**Current** – it must be up-to-date.

**Feasible** – it must be possible to implement the requirement, given the project constraints and business objectives.

**Unambiguous** – this is perhaps the hardest of all. Most requirements are written in idiomatic English and misinterpretations are highly possible, especially when the implementation is outsourced.

**Verifiable** – through inspection, demonstration, test or analysis, it must be able to be shown that the requirement has been implemented.
Business inputs to requirements

Every change project needs a starting point and the above documents are key to ensuring that the project clearly expresses the business need, has a defined scope and forms a basis for communication.
The Business Case

BCS “Business Analysis” Glossary Definition

A business case is a document that describes the findings from a business analysis study and presents a recommended course of action for senior management to consider. The business case justifies investment in the new/amended products, including any IT products.

The business case:

- Justifies investment in the product or in the project
- Provides options for resolving issues and measures of success

The business case brings together components that describe:

- The problem description
- The high level requirements
- The outline solution and deployment
- Costs, benefits, impacts and risks

It is important to keep focused on the business case during Requirements Engineering:

- As stakeholders emerge
- As requirements change
- As the product evolves
- As the business continually changes
- As costs, benefits and risk become clearer

The Business Case is often created as a standalone product of a feasibility study, commissioned to investigate area/s of the business that need attention.

It is usually the starting point of a project, once the preferred option has been selected.
Terms of Reference (ToR)

One of the ways that the components of a project, and time, can be identified and documented is through a Terms of Reference or ToR.

The ToR provides a structured way of defining the key components of a project. It (or an equivalent) should be considered an essential part of the project planning activities. It can be produced at many levels, for example by each key resource group involved in the project, i.e. one for Business Analysis activities, one for Testing, one for User Experience etc.

It will typically provide, before a project gets underway, the following information, often documented using the framework, BOSCARD:

- **Background** – what is happening in / to the business
- **Objectives** – what the project is aiming to achieve
- **Scope** – what is included and what is excluded
- **Constraints** – limits on time, budget, etc.
- **Authority** – the ultimate arbiter of any conflicts
- **Resources** – the people and equipment available to the project
- **Deliverables** – what we will measure to see if the objectives have been met

There is also a briefer version in use – OSCAR.

Terms of Reference are necessary for a pre-project stage, such as feasibility studies, to:

- Focus on understanding the scope of the problem, confirming the project boundaries, deliverables and high level requirements
- Deriving the business case that will provide detail for investment decisions
- Communicate what we will do

The ToR can also serve as a starting point for negotiation of what is achievable within the usual project constraints of time, cost and resources and may sometimes contain the estimates from the
planning stage. It usually feeds into the documentation for the Project, such as the Project Initiation Document, or PID.

Definition of a Project

Now we have the ToR and an understanding of what’s involved the project can be further formalised.

‘A project is a discrete piece of work with an agreed start and end date.

It consists of a set of co-ordinated and controlled activities undertaken to deliver a product conforming to specific requirements within the constraints of time, cost and resources.

Projects exist to deliver products that the business can use to deliver benefits.’

- A project is required once we have agreed ‘what to do’.
- Software development is usually part of wider business changes that are include in the project’s remit too.
Project Initiation Document (PID)

The Project Initiation Document is a project management product that forms the contract between the business and the project team.

- It defines the “What, Why, Where, Who, How and When” of the project and provides a basis for the management of a project and for measuring success.

The PID includes:

- The ‘project proposal’ is formalised as a Project Brief or Project Charter and consolidates the initial review, consolidated from the ToR into the Project Definition.
- It provides links back to the Business Case.
- Estimates, communication plans and resource requirements, success criteria (linked to the Business Objectives) are also included.

The Project Brief as defined in PRINCE 2™ is a ‘full and firm foundation for the initiation of the project’.
Requirements Engineering Framework

The following diagram shows the components of the BCS Requirements Engineering Framework – we shall examine each of these during this course.

The BCS Requirements Engineering Syllabus is structured around this five-part framework, which is typically applied to a project initiated by an approved business case.

Although some high level ‘feasibility’ work across all five areas may have been conducted prior to the project start, as part of a wider set of change initiatives.

Requirements Elicitation: suggestions covering the drawing out of requirements from the relevant stakeholders.

Requirements Analysis: suggestions covering a collection of activities needed to shape the requirements into a suitable form such that they are clear and understandable.

Requirements Validation: description and execution of the process by which the business, and in particular the Sponsor, signs off on the
set of requirements as being an accurate statement of what is needed.

**Requirements Documentation:** recommendations and means of documenting requirements.

**Requirements Management:** suggestions covering the need for change management, version and configuration control.
Case Study – Goatilicious

Read the Case Study Scenario (in your exercise workbook).

We will use the Case Study as the basis for future exercises.

Having read the case study, do you have any questions?
Summary

The key points of this module are as follows:

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A requirement is something a business user needs the (IT) system to provide.</td>
<td>✔</td>
</tr>
<tr>
<td>2</td>
<td>A requirement should not specify how the requirement is to be implemented.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Requirements are the basis for acceptance testing and need to be written in a way that makes this easy to do.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Requirements arise because of changes in the business from internal and external sources.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Requirements are difficult to capture and document well.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>The majority of conceptual errors occur during the analysis phase of a project. The later they are discovered, the more expensive they are to correct.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Projects exist to deliver products that the business will use to deliver benefit. New/amended IT products are often part of the project’s focus.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A business case defines the recommendations of a business analysis study, and justifies investment in the products.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Terms of Reference for pre-project work, might be based on the acronym BOSCARD or the shorter OSCAR.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A Project Initiation Document formally launches a project and forms the contract between the business and the project team.</td>
<td></td>
</tr>
</tbody>
</table>
Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You'll find the answers on the next page.

1. Name three classes of change which give rise to the need for new requirements.
2. List three problems often faced by requirements engineers when trying to write requirements.
3. What percentage of errors (approximately) occur in the analysis phase of a project?
4. State the definition of a “project”.
5. What is a business case?
6. What is a Project Initiation Document?
7. Name the words behind the acronym BOSCARD.
Answers to Module 1

1. Business, Operational and Software.
2. Any three of:
   - Scope is implicit
   - Too little definition / too much definition
   - Ambiguity
   - Contradiction
   - Written from differing ‘viewpoints’
   - Lack of traceability to business rules, processes and data

3. 50-60%.

4. A project is a discrete piece of work with an agreed start and end date. It consists of a set of co-ordinated and controlled activities undertaken to deliver a product conforming to specific requirements within the constraints of time, cost and resources. Projects exist to deliver benefits to the business.

5. A business case is a document that describes the findings from a business analysis study and presents a recommended course of action for senior management to consider.

6. The Project Initiation Document is a project management product that forms the contract between the business and the project team.

7. Background, Objectives, Scope, Constraints, Authority, Resources, Deliverables.
Further Reading

Module 2 – Requirements Stakeholders and Elicitation

Topics

In this section of the course, we will cover:

- Stakeholders involved in RE
- Elicitation problems and knowledge types
- Requirements list
- Elicitation techniques
- Understanding the applicability of techniques
**Stakeholders involved in RE**

**What is a Stakeholder?**

_A stakeholder is anyone affected by or interested in the business situation/area being investigated. They may be internal to the organisation or external, such as a supplier._

**Who are the stakeholders?**

There are many potential stakeholders in the RE process.

The BCS syllabus mentions the following:
Project Sponsor

The sponsor is the ‘owner’ of the new system.

- As such, the sponsor has the responsibility for ensuring that the system meets its goals and realises its benefits
- This implies that the sponsor has sign-off powers and has the final decision on all project matters
- Additionally, the sponsor is accountable for the project terms of reference, initiates project reviews, resolves project issues and acts as Champion for the new system

As far as the Requirements Engineering Process is concerned, the sponsor must ensure that the requirements capture the functionality and constraints of the new system such that it satisfies the business goals and yields the business benefits.

System End-Users and Managers

System end-users and managers represent the views of those who will use the system on a day-to-day basis.

- They must make sure that functionality and usability requirements are properly specified and therefore have a large say in the requirements
- They must be happy that the requirements are ‘workable’ and complete

Subject Matter Expert (SME)

The subject matter expert is the ‘super user’ who provides the specialist knowledge of the business area to be improved.

- They may be an internal or external resource, and may well have ‘consultancy’ type experience
• The expert provides information and ideas on business issues and possibilities and works with the analysts to ensure that the knowledge and ideas are represented in the requirements

**Project Manager**

The Project Manager, unsurprisingly, plans and controls the project, ensuring it keeps to time and cost constraints.

• Additionally, the project manager must ensure that the Requirements Engineering Process is being properly followed
• He/she may be called upon to resolve conflicts between stakeholders over the requirements

**Business Analysts**

The analysts are, in fact, the Requirements Engineers.

• Analysts elicit, document, analyse, validate and often manage the requirements
• In some cases the analysts will write the User Acceptance Tests and run them with the users to ensure the requirements have been met
• Analysts often produce supporting models, such as business process models, etc.

**Developers**

The developers are responsible for creating the new IT system to meet the requirements.

• The developers’ main concern is that the requirements can be built and tested
• Sloppy requirements will mean that the developers must constantly refer back to the analysts or simply ‘make it up’
Customers

Customers are affected by the interfaces to the system (inputs, outputs).

- Projects must comply with legal requirements concerning customers
- Often there is the need to communicate with them to inform them of changes
- Analysts may need to consult with them to discover preferences

Regulators

Many industries have statutory regulatory bodies (e.g. financial, utilities, telecoms).

- Regulatory constraints give rise to requirements on projects
- Changes to the law can also give rise to requirements

Suppliers

Suppliers may be affected by changes to interfaces.

- They will need time to change their systems if affected
- Our degree of influence over the supplier will partly depend on the relative sizes of the businesses

Others

There is of course plenty of scope for other stakeholders that are relevant to your own organisation or project requirements. Examples of other stakeholders could include:

- The government
- The general public (not necessarily your customers, perhaps those whom you ask an opinion of via surveys)
- Prospects/Hand-raisers (groups of potential customers)
- Previous Customers
- Testers and other project team members
What do you think?

- What sort of problems do Stakeholders have in stating requirements?

- What sort of problems does the Analyst have in eliciting requirements?

<table>
<thead>
<tr>
<th>Stakeholders</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Analyst</th>
</tr>
</thead>
</table>
Stakeholder problems

Stakeholders are subject to a number of problems, they:

- Might not know what they really want
- Find it hard to articulate what they want
- Make unrealistic demands through not understanding cost implications
- Express their requirements from their own viewpoints and the requirements may conflict
- Are subject to, and may be swayed by, politics
- They change their minds!

Analyst problems

Analysts may not have much experience in the domain in question and:

- Find it difficult to identify all of the stakeholders
- Must find a way to express requirements in terms that ALL stakeholders and actors can understand and agree
- Must resolve requirements conflicts and uncover commonalities
- Must cope with a changing businesses environment which might affect the priority of, or need for, requirements

Preparation

To elicit requirements, we must first understand:

- The business objectives
- The problem area and problem to be solved
- The business processes in scope
- The systems supporting the processes
- The operational constraints and business rules
- The stakeholders and their attitudes

We must also foster a sympathetic attitude to the needs of the stakeholders, the fact that this could be a big change for them: it could impact on how they work, the processes they are using now,
the way the work and the teams are structured and also the technology they have. They could be fearful of that change. So we must be prepared to take the time to find out their real needs.

**Stakeholder viewpoints**

Requirements are not normally elicited from a single stakeholder.

- Different stakeholders will have their own viewpoints on a requirement, the solution and the issues that are being experienced. All may be valuable
- The problem with single-viewpoint requirements can be that other stakeholders might not agree with it, resulting in unused functionality
- On the other hand, multiple viewpoints help us prioritise requirements and understand the important issues that our stakeholders face day to day

As an example, consider a business goal which is to ‘Improve the Customer Experience’:

- What problems does the customer currently perceive to be spoiling their experience with us?
  - The quote system is too slow and prone to errors
- What will improve the customer experience?
  - An improved response time for requests for quotes
  - The elimination of errors and lost quotes
  - Instant response to queries about requested quotes
- How can we achieve these improvements?
  - By integrating IT to improve logging of the quotes
  - By changing the structure of the quote team
  - By placing an audit trail on each quote so we can track them
  - By introducing a ‘workflow’ approach to the quote status

The requirements that will support the business objective should flow from these.
Tie in the Business Objectives

Despite all our efforts the stakeholders will still bombard us with their wants and needs, and in some cases this can be overwhelming.

Using the business objectives as assessment criteria helps us to rationalise the requirements more effectively, and can be used to justify why the requirements have been prioritised in a certain way.

Even better still is using the business objectives as a way to inform the stakeholders as early as possible and manage their expectations for what the business is aiming to achieve. Over time this will become a clearer message and will be part of the project infrastructure:

To summarise:

- Stakeholders won’t automatically know what the business objectives are, and their focus will be on resolving problems they perceive are present, in a way that allows them to articulate their concerns (which could be ‘I want the system to do what Google does’)
- We can use the business objectives as the driver for requirements elicitation and the rationale for validation
- All requirements must contribute to the business objectives in some way so that we can ensure the solution moves the business towards their objectives and strategic direction
- If they do not, they are superfluous and could consume valuable resources with zero return
Knowledge Types

Unconscious Competence

“Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know.”

Donald Rumsfeld

Very often our users do not tell us everything and this is not because they want to keep information from us (most of the time!) but because it does not occur to them to tell us.

We encounter plenty of people who are good at their jobs and who perform many of their tasks without thinking too much about what they are doing and the way in which they are doing it. As a result, our understanding of the situation may be flawed.
Knowing that this is a problem allows us to take action.

The four stages of competence (on the previous page), or four stages of learning helps us understand how we learn a new skill or competence:

- **Stage 1: Unconscious incompetence**, an individual has no awareness of the need to know a skill and has no awareness of the deficit
- **Stage 2: Conscious incompetence**, the individual still has no knowledge of the skill but they now have knowledge of what they don’t know: they are aware of the deficit
- **Stage 3: Conscious competence**, the individual understands or knows something but they need to concentrate on its application
- **Stage 4: Unconscious competence**, the skill has become second nature and the individual no longer has to consciously think about the application in order to perform it

**Tacit Knowledge**

Unconscious competence refers to unknown knowns; those areas of knowledge that we have forgotten we know, do automatically, take for granted, have a skill in or simply can’t explain.

We refer to the pieces of knowledge that fall under unconscious competence as **‘tacit knowledge’**.

**Tacit Knowledge** is ‘those aspects of business work that a user is unable, or omits, to articulate or explain. This may be due to a failure to recognise that the information is required or to the assumption that the information is already known to the analyst.’

The opposite of tacit knowledge is explicit knowledge – areas of knowledge that can be accessed by anyone, can be fully explained and can be documented.

**Explicit Knowledge** is ‘the knowledge of procedures and data that is foremost in the business users’ minds, and which they can easily articulate.’

*BCS FCBC Glossary of Terms, 2014*
Tacit knowledge takes many forms, consider the following:

- **Skills**
  - Trying to explain every single step towards achieving a goal is very difficult. Imagine having to explain in detail how to start a new word document, apply formatting and add a table of contents

- **Taken for granted information**
  - This category relates to the sort of information that users may not feel is worth mentioning and the analyst doesn’t question further. It could be around the specific use of terminology for instance, which the business user knows has different meanings across the business and assumes you know that too

- **Front story / back story**
  - This occurs when the user paints a more positive picture of their workplace or situation than is actually the case. Whether this is deliberate or not depends on the user. Sometimes it's about 'protecting' the team from criticism where the analyst is perceived to be representing management or head office, for example

Much of this is important for us to be aware of if we are going to design a solution that will enable the business to achieve its objectives.
Tacit vs. Explicit

The analyst can experience problems trying to elicit tacit knowledge:

- Need to adopt particular techniques
- Once elicited, ensure it is documented

Types of tacit and explicit knowledge at an individual and corporate level are shown in the table below:

<table>
<thead>
<tr>
<th>Tacit</th>
<th>Explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Individual</strong></td>
<td><strong>Corporate</strong></td>
</tr>
<tr>
<td>Skills, values, taken-for-granted,</td>
<td>Procedures, style</td>
</tr>
<tr>
<td>intuitiveness</td>
<td>guides, processes,</td>
</tr>
<tr>
<td></td>
<td>knowledge-sharing</td>
</tr>
<tr>
<td></td>
<td>repositories, manuals,</td>
</tr>
<tr>
<td></td>
<td>company reports</td>
</tr>
<tr>
<td></td>
<td>Norms, back story,</td>
</tr>
<tr>
<td></td>
<td>culture, communities of</td>
</tr>
<tr>
<td></td>
<td>practice, organisation</td>
</tr>
<tr>
<td></td>
<td>history</td>
</tr>
<tr>
<td></td>
<td>Tasks, job descriptions,</td>
</tr>
<tr>
<td></td>
<td>targets, volumes and frequencies</td>
</tr>
</tbody>
</table>

Business users will be able to pass on to us their explicit knowledge; it is more difficult for them to articulate the tacit knowledge. Much of this tacit knowledge is important for us to be aware of if we are going to design a solution that will enable the business to achieve its objectives. Once elicited and documented it becomes explicit knowledge and we can make use of it.
Tacit to Explicit Knowledge: AERO

Once elicited and documented, the tacit knowledge becomes explicit knowledge we can understand and use.

As you can see from the diagram, there are four basic ways to make tacit information **explicit**.

These can be summarised through the acronym AERO, and are:

- Apprentice: shadowing; protocol analysis
- Enact: prototyping; scenario role-play
- Recount: story-telling; scenarios
- Observe: observation (formal, informal)

![Diagram showing the four basic ways to make tacit information explicit.](image-url)
Documenting your findings

Once we begin the process of eliciting our requirements from our stakeholders we need to start documenting them.

A standard approach is to begin with a simple, informal requirements list (as shown below) which is later analysed and developed into an organised requirements catalogue (see Requirements Documentation).

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. All transactions must be secure</td>
<td>K Harrison</td>
<td>What kind of security?</td>
</tr>
<tr>
<td>2. A report of all transactions in the last seven days needs to be produced at 0900 each Monday morning</td>
<td>A King</td>
<td>What’s in the report? File format? Feed? Fields?</td>
</tr>
<tr>
<td>3. Customers need the option of English or Welsh</td>
<td>K Harrison</td>
<td>Where? When?</td>
</tr>
<tr>
<td>4. The system is to capture customer details</td>
<td>R King</td>
<td>Ambiguous – what details are to be captured?</td>
</tr>
</tbody>
</table>

A crucial consideration as you begin the process of documentation is to think about how you might need to use the information at a later stage.

For example, if you omit the source and date of capture now will this prove problematic when you need to go back and understand the context that initial requirement was captured within?

This traceability will be useful later on!
Activity – Elicitation Techniques

Your Instructor will now give you syndicate time to prepare presentations on Elicitation Techniques.

Use the information on the following pages for some guidance, but also include:

- Your own experience
- What has worked well for you (and what hasn’t)
- Situations in which you have used the technique
- The kind of information you have been looking for
- When you have faced tacit knowledge and what techniques helped to make this explicit
- The kind of stakeholders you have involved

Each syndicate group will be assigned an Elicitation Technique or Techniques by the instructor.

Your presentation should last no more than ten minutes.
Elicitation Techniques

Elicitation techniques include:

- Interviews
- Workshops
- Observation:
  - Formal/informal
  - Shadowing
- Focus Groups
- Prototyping
- Scenarios
- Document Analysis
- Special Purpose Records
- Questionnaires

Over the following pages, we will look at each of these techniques in turn.
Interviews

An interview is a structured discussion between the analyst and a stakeholder to elicit facts and information about the business situation and the stakeholder’s role in it. Interviewing is an excellent elicitation technique and a key analysis skill.

The analyst uses the interview as a technique for gaining understanding of the business area under investigation and the problems being experienced therein, eliciting and understanding the requirements.

Among the advantages of using interviews is that they enable us to quickly build up rapport with key stakeholders and build a working relationship with them as we are in a confidential, one-on-one setting with them.

However, if we were to rely exclusively on interviews it would be very time consuming, and each interview only gives us one stakeholder’s point of view.

Interview Lifecycle

Interviews should never be attempted without proper planning and structure, as shown on the following pages.

Plan

It will probably not be possible to interview all of our stakeholders due to time and geographical constraints. In order to make best use of this technique we should consider using it with senior stakeholders. Consider meeting the sponsor, department heads, directors etc. We should set clear objectives for each interview that we are going to conduct. This will help us as we prepare and conduct the interviews.
Prepare

Having ensured that we have management clearance to carry out the interview and that we know the stakeholder’s availability, we schedule the interview. It is important that we inform the stakeholder of our objectives for the interview and any preparation they need to make before we meet. Agree with them when the interview is to take place, where it will happen and how long it will last. Consider where the interviewee might feel relaxed, how you can reduce the risk of interruptions, whether back-up material must be available and whether the physical environment will be suitable.

We also need to ensure that we are fully prepared – what we need to cover in the introduction, what type of questions we are going to ask and in what order, etc. The type of questions we ask will obviously depend on the purpose of the interview. We are trying to understand the current business processes and systems so that we can uncover the problems and create the requirements for the new system.

A good ‘Questioning Strategy’ will serve as a framework (or checklist) for structuring our questions during the interview. We may include questions on the processes, problems associated with the processes, what the key business data entities are for each process, what business rules constrain the process, which stakeholders are involved, what are their attitudes, how often the process runs, and of course what are this stakeholder’s requirements for change.

It is vital to go into an interview with a clear set of prioritised and sequenced topics that you want to cover. Maintain a logical sequence of topics and cover the most important topics first in case you run out of time. You might like to introduce sensitive topics later in the interview once rapport is established.

It is also important to follow a time plan and to not exceed the time allocated for each topic. To this end, prepare the key questions in advance for each topic and ensure the questions are ‘open’. Open questions encourage the interviewee to feel relaxed and ‘open up’. Examples are:
• “I understand that you are the best person to talk to about ordering. Could you tell me how you organise the process?”
• “What are the benefits of monthly stock checks over more frequent ones?”
• “I’m very interested in the way you control your bookings. Could you walk me through your experiences in that area?”
• “That’s really interesting. Why do you do it that way?”

In contrast, closed questions invite closed answers and can induce defensiveness:

• “You are in charge of ordering, aren’t you?”
• “Do you do a stock check every week?”
• “Did you make last quarter’s target?”

Closed questions may, however, be useful when starting the interview to get the interviewee comfortable.
Conduct

Initial impressions are vital in building up rapport and putting the interviewee at ease. Make sure you:

• Introduce yourself
• Smile and make eye contact
• Shake hands, where appropriate
• Thank the interviewee for seeing you
• Restate purpose, timing and needs
• Explain why the information is necessary
• Explain the need for note-taking
• Explain what feedback will be given
• Explain the structure
• While you are doing this, attempt to assess attitude of interviewee

During the body of interview, use the interview structure that you prepared, moving from topic to topic and introducing each topic with an open question. Use the prepared follow-up questions BUT listen to the answers you are getting and adjust your questioning accordingly. A successful interviewer will build up rapport quickly, listen actively and think quickly. You will need to show that you’re listening by using appropriate body language and sending out small verbal messages.

Show that you understand what is being said by demonstrating empathy, paraphrasing important points and summarising at the end of each topic.

Throughout the interview, try to read the reactions of the interviewee. Is he or she disassociating, frowning, nodding at inappropriate times, looking at their watch, fidgeting, showing signs of distress. Be prepared to exit gracefully if necessary.

At the end of the interview, review the interview notes and confirm the main points with the interviewee. Tell the interviewee what will happen next and check on future availability for further interviews if necessary. Thank the interviewee for their time and contribution.
Always try to conclude the interview by the agreed time.

**Follow-Up**

Write up notes as soon as possible while the memory is fresh, and then send a copy to the interviewee for comment. Amend the notes as necessary and then publish them. Carry out an analysis of the information gathered in order to extract and organise the facts so that you can create the requirements.

**Summary**

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build rapport building and one to one engagement</td>
<td>Time consuming</td>
</tr>
<tr>
<td>Allow discussion of confidential information</td>
<td>Only provide one viewpoint</td>
</tr>
<tr>
<td>Useful for senior stakeholders</td>
<td>Can be difficult to manage (and can go off track easily)</td>
</tr>
</tbody>
</table>

**Workshops**

A facilitated workshop is a team-based information gathering and decision-making technique designed to accelerate business planning and development. It is an interactive communication technique involving experienced and empowered personnel working in one or more sessions run by an independent facilitator. A workshop is a process to be implemented when there is a requirement to make decisions, explore ideas and exchange knowledge to solve a business problem.

The strength of the facilitated workshop technique is that it enables the exchange of information between key individuals and enables them to reach decisions that are mutually acceptable. A workshop provides a forum for exchanging views and achieving consensus.
decisions in a structured framework across and within areas of the business. Clear deliverables are produced during the workshops enabling all attendees to review decisions taken by the group.

**Advantages**

Workshops offer the following benefits:

- **Speed** – by gathering all the relevant stakeholders in a workshop, it is possible to achieve an agreed set of user requirements, for example, in a day or two as opposed to weeks or months using traditional methods.
- **Ownership** – stakeholders who are part of the group decision-making process are more likely to be committed to its decisions and to feel a greater sense of ownership for subsequent development work.
- **Productivity** – the workshop often benefits from the participants building on each other’s ideas and gaining a better understanding of each other’s viewpoint. Often misunderstandings in the business areas can be swiftly clarified during workshops.
- **Consensus** – the workshop provides an opportunity for the participants to discuss relevant subject matters, including the major issues and problems, with a view to reaching a consensus on all the important decisions.
- **Quality of decision-making** – because stakeholders are active players in the decision-making process, the levels of confidence and understanding of the workshop output are likely to be high.
- **Overall perspective** – workshop participants can appreciate the importance of business areas contained within the project but in which they are not directly involved.

The simple principle is that several brains pulling together can (usually) produce better solutions than one brain working in isolation. However, badly planned and conducted workshops can be a complete waste of everyone’s time. The symptoms of a bad workshop include:

- A vague, ambiguous statement of the objectives
• A lack of clear focus and direction
• A lack of consensus and commitment to the solutions
• Time-wasting, side discussions, arguments etc.

Facilitated workshops are an excellent forum for requirements analysis and negotiation.

Workshop Roles

Sponsor

The sponsor ultimately ‘owns’ the workshop and its objectives and is the ultimate decision-maker. The sponsor (or delegated person) works with the facilitator to create the terms of reference for the workshop. They will agree the rules for the workshop and the sponsor empowers the facilitator to enforce them. The sponsor also empowers the participants of the workshop to make the necessary decisions.

Facilitator

The facilitator is the key player in the workshop. He or she ensures the group meets the agreed objectives in the agreed timeframe. It is important that the facilitator manages the workshop process and NOT the content of workshop.

The facilitator maintains the focus of the workshop, manages the group dynamics and encourages individual contributions. Naturally, the facilitator also upholds the workshop rules.

The facilitator needs a range of attributes, including:

• Detachment, neutrality and objectivity
• Professionalism and integrity
• Positive, patient, cheerful, organised demeanour
• Active listening ability
• Ability to question and to give feedback
• Ability to maintain group participation and consensus
• Ability to handle group dynamics and conflict
Participants

The participants are required to contribute fully and positively to the workshop by providing the ideas and expertise. They must work towards group success and stay within the workshop rules. It is vital that the participants are empowered to speak and make decisions on behalf of their areas.

The Scribe

The scribe works closely (and often interchangeably) with the facilitator. He or she is responsible for ensuring workshop output is captured accurately and published as formal documentation. The scribe takes no other part in the process.

The Workshop Lifecycle

Planning and Preparation

When (as facilitator) you plan and prepare for the workshop, you will need to understand and document the terms of reference, including:

- The context of the workshop
- The objectives of the workshop
- When it will be held
- Where will it be held
- Who will be invited
- How long will it run for
- What the rules will be
- What the time plan will be
- What form the output is to take

The analyst will then need to agree the plan with the sponsor and send invitations to participants along with the rules, the terms of reference and any other documentation they require. The analyst will also need to brief the scribe before the workshop and personally check the venue for layout, lighting, ventilation and equipment such as stationery, flip charts, white boards, sticky notes, projectors etc.
Introduction

Before the workshop starts, review and agree the terms of reference with the participants as some may not have had the opportunity to read it themselves. Cover at least: BOSCARD, structure and time-plan.

Then review and agree the workshop rules with the participants. Some example rules are:

- All are equal
- All are to contribute
- All are to be treated with respect and consideration
- Facilitator is the sole arbiter of process decisions
- Side-discussions are to be conducted outside the workshop

Brainstorming

Many workshops start with some sort of brainstorming session that generates ideas for the objective. For instance, we might hold a risk workshop for a set of requirements and this might start with a brainstorm to throw up candidate risks.

When brainstorming we are looking for the quantity of contributions not the quality, so the facilitator accepts all ideas without discussion and ensures that all contributions are visible (typically with sticky notes). It is a good idea to capture the contributor’s name for the later discussion.

**Nominal Group Technique (NGT)**

NGT is a form of brainstorming/brainwriting.

NGT aims to help groups of suitably skilled and motivated participants to deliver high-quality products in a short timebox.

Using sticky notes, ideas are generated without talking. Quality, and not quantity, is expected. The facilitator should arrange the post-its
on a flipchart into groups and give a representative name to each. Care should be taken if is impossible to name a group.

It probably means that you do not have a homogenous group. If groups have to be prioritised, the team should be asked to write down a number between 1 and 5 on a group-by-group basis (again, without conversation).

By averaging the results for each group, the order of preference is clear. NGT can be used to arrive at a candidate list of requirements and prioritise them very quickly and without protracted discussion/argument.

---

**Selection / Validation**

Once the brainstorming is complete, the facilitator invites discussion on each contribution in turn, asking the contributor to speak first. The facilitator should try to elicit a short description of the contribution in order to allow the group to agree on the validity of it. In any case, the group will determine whether the contribution is valid (and where appropriate, its relative merit) and will reject invalid contributions.

Often the contributions can be categorised, grouped, ordered or in other ways classified. For instance, agreed risks could be placed into categories and assigned a risk factor.

**Review**

Once the contributions have been agreed and any classification achieved, the result is reviewed using a formal walkthrough of the workshop findings. It might be appropriate to run the walkthrough as a benefits/concerns session. The group is looking to check for completeness, ambiguities, duplication etc. and to check descriptions are adequate. If the review fails, the group must decide if more brainstorming, selection or validation is required.
**Follow Up**

The scribe formally documents the output of the workshop, distributes the documentation to the sponsor, participants and other stakeholders and requests comments within a short time-frame.

**Summary**

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can help get buy-in</td>
<td>Can be expensive</td>
</tr>
<tr>
<td>Achieves consensus</td>
<td>Can be difficult to manage and document</td>
</tr>
<tr>
<td>Speeds things up</td>
<td>No consensus achieved</td>
</tr>
<tr>
<td>Gives an overall perspective to all participants</td>
<td>Can go off track</td>
</tr>
</tbody>
</table>
Observation

Watching people actually carrying out the tasks rather than just asking about them is a very useful way of getting information about the work practices and environment. This can be done in a number of ways. If you are going to carry out observation it is important to get the agreement of those being observed to avoid possible industrial relations problems.

**Advantages**

- We can get a good understanding of the processes, problems, politics etc.
- Helps devise workable, acceptable solutions

**Disadvantages**

- Can feel like ‘Big Brother’ – unsettles the observed
- Your presence may impact the process

**Types of Observation**

**Formal**

This involves watching staff carrying out specific tasks. It is important for the staff to be prepared for your observation, and to emphasise that you are watching to gain an understanding of the tasks and not assess performance. There is a danger that staff may perform the tasks “by the book” rather than how they would normally if they think you have been sent by management.

**Informal**

Really just wandering about to see what goes on. The dangers here are that staff may be unnerved by your presence as they have not been prepared and that you will not see the full range of tasks that you need to see. If the workers are used to your presence, however, this can be a good way of seeing what really goes on.
Shadowing

This means following a worker for a period of time, and can be usefully done before some of the more structured requirements elicitation in interviews or workshops. It may include Protocol Analysis in which the worker performs a task and describes each step as they perform it. This gives excellent opportunities for asking “why?” type questions.

Advantages

- Enables an understanding of the ‘real world’, i.e. what the behind the scenes looks like
- Gives access to the users in their usual environment
- Can facilitate discussion and exploration of tacit knowledge

Disadvantages

- Can intimidate users
- May not provide insight into typical day to day activities (e.g. busy periods, fluctuations on workload, seasonal variations)
Focus Groups

Focus groups are useful for business and market research. They consist of a group of people with a common interest.

Groups like this help to identify and understand:

- Attitudes
- Concerns
- Beliefs
- Opinions

There is no need for consensus or ownership but remember that findings need to be analysed.

Advantages

- Facilitates a broad view of opinions and concerns
- Can be used to achieve market research activities that help understand how the customers feel about a particular solution or option before investment takes place
- No consensus is required

Disadvantages

- Results have to be analysed carefully and may uncover further investigation
- The objectives need to be understood fully
- The results may not yield the expected results
Prototyping

A common complaint from analysts is that users are much better at telling you what they do and don't like or want when something is presented to them than they are at telling you what they want off the top of their heads.

By creating a ‘demonstration system’, prototyping shows the users how a system might work which can help clarify their thinking and firm up requirements before we are too far down the development path.

The ‘system’ can be anything from sticky notes or a set of screen mock-ups to a fully-fledged application.

Prototyping is an integral part of iterative / incremental systems development, and has a strong link with scenarios.

A fuller treatment of prototyping can be found in the module on Requirements Validation.

Advantages

- Helps the user to really determine the requirements, especially user interface, performance, navigation paths
- Validates requirements
- Reduces risk of “getting it wrong”

Disadvantages

- Can run out of control
- Can raise unrealistic expectations
- Can lead user to overestimate progress
Scenarios

Scenarios tell the story of the task, including:

- All steps from business trigger to outcome
- Pre-conditions
  - Must be true for the scenario to begin
- Post-conditions
  - Must be true following conclusion
- Alternative paths
- Exception situations

Advantages

- Aids visualisation and discussion of real life situations
- Can be used to build test scenarios

Disadvantages

- Can become complex, especially where there are many alternative paths
Document Analysis

This involves the reading and analysis of source documentation such as forms, screen layouts and reports. For each document analysed, we need to find out:

- How is the document completed
- Who uses it
- When is it used / produced
- How many are used / produced
- How is the document retained
- Details of the information shown on the document (source, rules, etc.)

It is often easier to determine this if we are able to use completed forms or real printouts rather than just layouts.

Advantages

- Can yield good information on:
  - Organisation
  - Process
  - System
- Can be useful to go back to the source of information, particularly where it is written, such as in:
  - Order forms
  - Questionnaires
  - Requests for information

Disadvantages

- Doesn’t get differing viewpoints
- Dry and tedious
Special Purpose Records

Here, stakeholders are asked to keep a record of what they do when engaged in a task or process. This can range from a simple diary to filling in specially prepared questionnaires at intervals during the day. For this to succeed we need the workers to buy in to the process; if they don’t see the need we are more likely to experience the problems shown below. We need to be realistic about what we can reasonably expect people to record while carrying out their normal work.

**Advantages**

- Good for capturing metrics, problems, time and motion etc.
- Avoids ‘big brother’ aspect of observation

**Disadvantages**

- People dislike the extra work
- Must be done real time or accuracy will suffer
- People might ‘bend’ the records to put themselves in a better light
Questionnaires

Questionnaires are a very useful way of eliciting limited amounts of information from a large group of people, especially if they are geographically spread. The skill comes in formulating the questions so that you are able to get the needed information in a way that is easy to analyse and turn into requirements.

Advantages

- Can reach a large audience, even if geographically separated
- Can uncover common problems
- Can determine attitudes

Disadvantages

- People don't like surveys
- The survey must be kept small and focused
- It takes skill to formulate a survey
Understanding the Applicability of Techniques

The table below shows how far application of the techniques can be useful in uncovering different types of knowledge.

After Maiden & Rugg (1996)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Interviewing</th>
<th>Shadowing</th>
<th>Workshops</th>
<th>Prototyping</th>
<th>Scenario analysis</th>
<th>Protocol analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Future requirements</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Skills</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Front/back story</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taken-for-granted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tacit knowledge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explicit knowledge</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

After Maiden & Rugg (1996)
## In Summary

<table>
<thead>
<tr>
<th>Technique</th>
<th>Who?</th>
<th>Information</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interviews</td>
<td>Project Sponsor, SMEs, Senior management</td>
<td>Clarify terms of reference, Understand objectives, Gain explicit knowledge</td>
<td>May need confidentiality, Specific detail, May need to build rapport/engage with their teams</td>
</tr>
<tr>
<td>Workshops</td>
<td>Peer groups/teams, Users, Decision-makers</td>
<td>Exploring ideas, Decisions, Knowledge exchange (can gain tacit knowledge)</td>
<td>Engage with users, Gain consensus, Engagement at group/team level</td>
</tr>
<tr>
<td>Formal Observation</td>
<td>The users themselves</td>
<td>Tacit knowledge, Processes and what 'actually happens' (if the users are comfortable!)</td>
<td>Sense of reality, To get to grips with any workarounds and problems</td>
</tr>
<tr>
<td>Shadowing</td>
<td>The users themselves</td>
<td>Why something is being done as well as how it is done</td>
<td>Sense of reality, To get to grips with any workarounds and problems</td>
</tr>
<tr>
<td>Technique</td>
<td>Who?</td>
<td>Information</td>
<td>Why?</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Focus Group</td>
<td>Groups of people with a common interest</td>
<td>Opinions and concerns (market research)</td>
<td>To gain a rich understanding of attitudes, opinions and concerns</td>
</tr>
<tr>
<td>Prototyping</td>
<td>The users themselves (and their managers)</td>
<td>Understanding of user needs Clarify requirements</td>
<td>Often to validate requirements by showing users something visual</td>
</tr>
<tr>
<td>Scenarios</td>
<td>The users themselves</td>
<td>Step by step detail of what should happen within a process, group of tasks or a task itself</td>
<td>To identify the success criteria To identify exceptions and alternate process requirements To identify business rules</td>
</tr>
<tr>
<td>Document Analysis</td>
<td>Not targeted towards a user specifically but will likely be user documentation</td>
<td>Processes Systems and their users Background and history Data capture requirements and usage</td>
<td>May need to understand more than the users reveal May need to see how a process was intended to be completed</td>
</tr>
<tr>
<td>Special Purpose Records</td>
<td>The users themselves</td>
<td>Metrics, frequencies, volumes etc.</td>
<td>To gain 'real' analysis that the users have participated in (which may help their buy in)</td>
</tr>
</tbody>
</table>
### Module 2: Requirements Stakeholders and Elicitation

<table>
<thead>
<tr>
<th>Technique</th>
<th>Who?</th>
<th>Information</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questionnaires</td>
<td>Usually large groups of people who we may not have access to</td>
<td>Factual information, Opinions and thoughts</td>
<td>Can reach a large audience who we may not have access to</td>
</tr>
</tbody>
</table>
What do you think?

Which elicitation techniques are the most useful for you? When do you use them?
Dos and Don’ts of Requirements Gathering

Do talk with a vertical span of business people, including executives, directors / managers and analysts.

Don’t rely on a single user to represent the business, even if it’s less intimidating and easier to schedule.

Do allow plenty of time to coordinate calendars for scheduling, especially with travelling business management.

Don’t get angry if someone has to reschedule at the last second.

Do get help from skilled assistants to manage scheduling.

Don’t offload interview coordination to someone unfamiliar with the project or organisation.

Do arrive for the interview on time.

Do be professional and respect the other person’s time. Leave your mobile switched off and out of sight so that you don’t cause any distractions.

Don’t use a laptop to capture notes as this creates a barrier between you and the interviewee.

Do plan on using a two-person requirements team in each interview, if possible – one to interview, one to scribe.

Don’t overwhelm a lone interviewee with six people sitting across from him/her, Inquisition-style.

Do designate one person as the lead interviewer with primary responsibility for steering the session.

Don’t turn the interview into a free-for-all, bouncing randomly from one interviewer to the next.

Do flesh out your scribbled interview notes immediately, or you'll lose much of the interesting detail by the next day.

Don’t schedule multiple interviews per day.

Do document what you learned during the requirements gathering and feedback results to close the loop with participants.

Don’t lose sight of the scope of your requirements process.
Exercise 1

Open your exercise workbook and complete Exercise 1 on the case study:

‘Stakeholders and Elicitation Techniques’.
## Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>There are many stakeholders in RE, including:</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Project</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• External</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Stakeholders have problems in expressing their requirements realistically, completely and unambiguously.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Business Analysts face the challenge of understanding the system, the stakeholders and the requirements.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Requirements are not normally elicited from a single stakeholder.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Knowledge falls into two main categories – Individual and Corporate – and within those into Tacit and Explicit.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Tacit Information is that which is understood but not explicitly documented, nor can be easily articulated</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>There are many requirements elicitation techniques, such as interviewing, prototyping and workshops.</td>
<td></td>
</tr>
</tbody>
</table>
Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You’ll find the answers on the next page.

List the major types of Requirements Stakeholders.

1. What is another name for the Domain Expert?
2. List three problems faced by Stakeholders when they are assisting the Analyst to create the business requirements.
3. List three problems faced by Analysts when crafting requirements.
4. List the strategies aimed at changing tacit knowledge to explicit.
5. List three benefits of workshops as an elicitation technique.
Answers to Module 2

2. Subject Matter Expert or SME.
3. Any three of:
   - Might not know what they really want
   - Find it hard to articulate what they want
   - Make unrealistic demands
   - Viewpoints
   - Subject to politics

4. Any three of:
   - May find it difficult to identify all of the stakeholders
   - Must find a way to express requirements in terms that ALL stakeholders and actors can understand and agree
   - Must resolve requirements conflicts and uncover commonalities
   - Must cope with a changing businesses environment which might affect the priority of, or need for, requirements

5. Apprentice, Observe, Recount, Enact.
6. Any three of:
   - Speed
   - Ownership
   - Productivity
   - Consensus
   - Quality of decision-making
   - Overall perspective
Further Reading

Module 3 – Organising and Analysing Requirements

Topics:

In this section of the course, we will cover:

- Prioritising and packaging requirements for delivery
- Organising requirements
- Ensuring well-formed requirements: the filters
- Prototyping requirements
- Verifying requirements
Once we have elicited and documented the requirements, we need to organise them (see Hierarchy and Documentation below), consider the releases in which they will be included and analyse them to ensure they are complete, consistent and unambiguous, and then negotiate with the stakeholders to seek agreement and resolve conflicts.

Analysis often raises additional questions and we may need to revisit the elicitation stage.

We can see from the Requirements Engineering Process that Elicitation and Analysis make up an iterative cycle.
Poorly Expressed Requirements

Requirements are often couched in ambiguous, “woolly” terms.

For example:

“I want a user friendly system”

- Define a way of measuring user friendliness, perhaps based on speed to learn or popularity

“The system must never fail”

- Not testable – and probably not realistic

“Response time must be 2 seconds”

- Give a range, 50% of transactions within 2 secs; 30% within 4 secs...

“Do what we currently do but faster”

- Don’t just re-implement current system

“Input will be with a mouse”

- Don’t give solutions

It’s the analyst’s job to help the business express their requirements accurately.
Business Objectives to Solution Delivery

Business objectives drive the need for change and give rise to new business requirements.
Defining the Requirements

As discussed previously the business objectives can help us to prioritise our business needs, and therefore our requirements.

- By understanding the business objectives we can also make sure that implementation activities and integration into Business as Usual is managed to the benefit of the business.
- And map to the people, processes, technology and organisational changes that will be implemented as part of the change
- Longer-term business focused requirements may influence the requirements for IT solutions, requirements which will be expressed in functional and non-functional categories
- Solution requirements may be subject to frequent change
The Hierarchy of Requirements

**Business Requirements**, which we can further categorise as general and technical requirements, tend to be those that are defined at an Enterprise, Organisation or Departmental level. They are usually requirements that, irrespective of the solution, are governed by over-arching needs of the organisation.

Business objectives are intrinsic to understanding business requirements which in turn are key to understanding and uncovering solution requirements.

**Solution requirements** tend to be defined at a product level, which means that whilst each product we create is different they are all compatible within the overall requirements of the business. Solution requirements are typically categorised as Functional and Non-Functional (NFR).

Requirements start to evolve very early in a project, or even at a pre-project stage:
• These address goals and objectives set by the business, or specific problems that need to be resolved
• The level of detail and scope increases as the requirements are investigated, defined and transformed throughout the project lifecycle, thus creating a hierarchy of increasing detail
• It is important to note that the level of detail and format of requirements in one project will not necessarily be the same as in another
• The scope, level of detail and format of requirements should be based upon the characteristics of the product to be produced
• Deciding how much detail and formal documentation is required at any given point is a matter for Business Analysts and developers to judge, based on the individual project’s characteristics

The next section explores General, Technical, Functional and Non-Functional Requirement categories.

Other Categories

It’s important to acknowledge here that there are numerous ways of categorising requirements.

Along with defining business and solution requirements, there may also be a need to categorise by, for example:

• Department
• Stakeholder grouping
• Technology
• Legal
• Language
Business: General Requirements

The General group of Business requirements includes:

- **Business constraints**
  - Budget, timescale, resources etc.

- **Business policies**
  - Standards, business rules

- **Legal**
  - Legislative and regulatory constraints

- **Branding**
  - Image, style guide

- **Cultural**
  - Vision, approach, management style etc.

- **Language**
  - If operating across international boundaries

- **Business continuity**
  - Coping with disaster
Business: Technical Requirements

These are examples of the Technical group of Business Requirements:

- **Hardware**
  - IT and other hardware

- **Software**
  - Operating systems, package applications, networking, communications etc.

- **Interoperability**
  - Standards for communicating between systems and devices

- **Internet**
  - Policies on Internet use and web services

These requirements may well derive from policies set by, for example, the Enterprise Architecture group, aimed at aligning Business and IT Strategies.
Solution Requirements

- Requirements are expressed as statements of what the ‘system’ must do in order to meet its business goals (Functional Requirements) and statements about the way the functionality should be delivered in order for it to be a successful product (Non-Functional Requirements or NFR)
- There is a relationship between the two types of requirements in that non-functional requirements are intended to act as constraints within which functional requirements must operate
- All requirements must be implementable and testable, although some operational NFRs are difficult to test before implementation and should be the subject of appropriate Service Level Agreements (SLAs)
- Solution requirements inform us as to what the system needs to do (the functionality)
Solution: Functional Requirements

Example categories of Functional Requirements:

<table>
<thead>
<tr>
<th>Data entry</th>
<th>Data maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Gathering and recording data</td>
<td>• Changes to data, including data deletion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedural</th>
<th>Retrieval requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Implementation of business rules</td>
<td>• Reporting, responding to enquiries</td>
</tr>
</tbody>
</table>

Since we are talking about software products, the description of functionality relates to things like data capture, manipulation or reporting, all of which must be provided by the IT system.

- Functional requirements are expressions of stakeholder needs of a system to achieve their particular goals
- They are **what** the system must do, functionality it must offer

This functionality is often summarised as **CRUD**:

- **Create** – data or records added to the system
- **Read** – data or records available to be viewed or retrieved such as from a search or for a report
- **Update** – data or records that need to be edited. Note they usually also need to be viewed (i.e. read) first
- **Delete** – data or records that are removed from the system

- Functional requirements also reflect procedural needs as well
Solution: Functional Requirement Examples

The system shall:

- Hold details of all customers, including name, address, credit limit, date of first order (Create)
- Allow changes to be made to customer details (Update)
- Report on all orders placed in the last week (Read)
- Orders can be cancelled (Delete)
- An insurance policy can have a maximum of 3 policy holders (Procedural, business rule)

It is considered good practice to express Functional Requirements using a verb-noun phrase.

Note: in many organisations the exact information that would make up a customer record is covered in the general requirements. It could though, be specific to the product being defined and so may also fit within the functional requirements.

Often these sorts of decisions are dependent on the organisational policies and impact to other systems. If the product being built needs to add a new field to the customer data then this may impact all of the systems that use customer details. This is why it is important to properly analyse your requirements!
Solution: Non-Functional Requirements (NFR)

Non-Functional Requirements (NFRs) are all to do with how well the functional requirements will be provided.

They must be implementable and testable, or subject to a Service Level Agreement (SLA).

Non-Functional Requirements are often applicable to major costs, time and risk parameters within a project.

Requirements of this nature include:

Debra Paul, in the BCS book, describes non-functional requirements as being “concerned with how well the solution will operate, and answer questions such as “How quickly will it respond?” and “How easy will it be to use?””. 

<table>
<thead>
<tr>
<th>Performance</th>
<th>Security</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Speed of processing transactions</td>
<td>• Security levels for protection of data</td>
<td>• Permissions, who has access to which functionality and how</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup &amp; recovery</td>
<td>Archiving &amp; retention</td>
<td>Robustness</td>
</tr>
<tr>
<td>• Protection against loss of data</td>
<td>• Duration, methods, eventual deletion</td>
<td>• Reliability, data integrity, user error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability</td>
<td>Usability</td>
<td>Capacity</td>
</tr>
<tr>
<td>• Timeframe for availability of functionality</td>
<td>• Ease of learning, ease of use</td>
<td>• Data volumes, transaction volumes, user volumes</td>
</tr>
</tbody>
</table>
Solution: Non-Functional Requirements

Examples

Examples of NFRs, include:

*Only the Accounts Supervisor may change a customer’s credit limit.*

- This is a security or access constraint

**Reprint the report on demand.**

- This is an availability constraint

**Respond to an enquiry on available customer credit within 5 seconds.**

- This is a performance constraint

You can test for a Non-Functional Requirement by asking “Is this requirement about the way the product must be, or does it describe an attribute that the product must have?”

Usually Non-Functional Requirements will be constraints that limit the way the functionality will be used, such as who has access to the functionality or data, how quickly the system will respond, availability of data and functionality and how easy the system should be to use (the latter being relative to the system complexity).
Uncovering Non-Functional Requirements

The important thing is to get the stakeholders involved.

Use both a ‘top-down’ and a ‘bottom-up’ approach:

**Top-down**

- Look at each standard NFR category and ask:
  - “What constraints apply to this area?”
- Most development organisations develop their own list of NFRs to check against

**Bottom-up**

- Look at each functional requirement and process activity and ask:
  - “Is this subject to any specific constraints?”
- Ensure that the functional requirement is cross referenced to any non-functional requirement(s) by which it is constrained

What do you think?

Which are more important – functional requirements or non-functional requirements?
Exercise 2 – Ticket Sales System

Open the exercise workbook and do Exercise 2 – Ticket Sales.


**FURPS+**

This is a model for categorising software quality attributes. It was developed by Hewlett-Packard and first publicly described by Grady and Caswell. The acronym stands for:

- **Functionality** – Feature set, Capabilities, Generality, Security
- **Usability** – Human Factors, Aesthetics, Consistency, Documentation
- **Reliability** – Frequency/Severity of Failure, Recoverability, Predictability, Accuracy, Mean Time To Failure (MTF)
- **Performance** – Speed, Efficiency, Resource consumption, Throughput, Response time
- **Supportability** – Testability, Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Serviceability, Installability, Localisability, Portability

*FURPS+ is now widely used in the software industry.*

**SWEBOK**

This stands for the Software Engineering Book Of Knowledge and the document, created by the IEEE, includes major chapters on:

- Software Engineering Requirements
- Software Engineering Design
- Software Quality
- Related Disciplines of Software Engineering

Chapter 2 of the Book, which can be found at [http://www.computer.org/portal/web/swebok/html/contentsch2#ch2](http://www.computer.org/portal/web/swebok/html/contentsch2#ch2) has a great deal of knowledge appropriate and useful to the business analyst.
Requirements Analysis

Requirements analysis is carried out against a checklist of analysis areas. It is like passing the requirements through a set of filters, each one catching specific types of problem. The syllabus suggests the areas listed in the diagram below.
Priority (MoSCoW)

The first stage of requirements prioritisation is to identify the business objectives and then keep these to hand. Any rationalisation should map back the need against the priority.

It is important to assign priorities to requirements to allow the business and IT to phase implementation of the change and to manage the expectations of all stakeholders.

<table>
<thead>
<tr>
<th>M – must have</th>
<th>• Mandatory in the first increment; cannot meet business and project objectives without it</th>
</tr>
</thead>
<tbody>
<tr>
<td>S – should have</td>
<td>• Mandatory but may wait until second increment; the system will have short-term value without it</td>
</tr>
<tr>
<td>C – could have</td>
<td>• Beneficial if time or funds allow, but not central to project objectives</td>
</tr>
<tr>
<td>W – want to have (won’t have this time)</td>
<td>• Will not be met in this delivery; may be included in a future delivery</td>
</tr>
</tbody>
</table>

The priority is not generally assigned when the requirement is first captured (although you may find you have a good idea of how the priorities may pan out), but later on during review and negotiation.

MoSCoW is widely used as a framework for prioritisation as it allows the requirements to be clearly linked to the business objectives.

All requirements should be prioritised and the priorities should be used to focus the analysis, negotiation, implementation and test. It is pointless expending energy and resources on a requirement that is merely cosmetic as opposed to one having massive business impact.
Note that the priority of requirements may change over time as conflicts are resolved and the descriptions are clarified.

**MoSCoW Questions**

We should examine each requirement to ensure it has been correctly prioritised:

- Does this requirement help to achieve a business goal or goals
- How high is its business impact (MoSCoW)?
- How much risk does it carry?
- How difficult is it to implement? Will this affect its priority?
- What ‘iteration’ should it be implemented in?
Kano Analysis

Noriaki Kano developed his model in the late 1980s to help analysts separate the real stakeholder requirements from the purely incremental. The technique allows analysts to prioritise requirements as a function of customer satisfaction.

There are four categories of requirement:

- **Surprise and Delight** – these are capabilities which differentiate the product from the competition, e.g. the iPod nav-wheel
- **More is better** – improvements along a clear continuum of increasing utility – e.g. better battery life in an MP3 player
- **Must be** – without these capabilities, customers will not buy/use the product
- **Better not be** – these represent the aspects which would prevent customers from buying / using the product

The first development phase of the solution should primarily include the “Must be” requirements. The “More is better” set are prioritised on ROI. Finally the “Surprise and delight” are examined, not because they are whimsical, but because they are valuable differentiators from the competition. Marketing strategies play a strong part in this activity.
Realism and Feasibility

Requirements Engineering should not be conducted in a vacuum. It is important to keep realism and feasibility in mind during the process (and the business objectives can help to achieve this).

There are three aspects to feasibility:

- **Technical**: is there technology available to implement the requirement? It is important to discuss technology with our technicians early in the process
- **Business**: assessment of acceptability by the business. By engaging with the business stakeholders, we can ensure that the proposals are realistic in their view
- **Financial**: is the expenditure to implement the requirement justified? In the end the product's Sponsor must make this judgement
Module 3: Organising and Analysing Requirements

Overlaps and Duplication

- Where requirements overlap, they must be separated
  - Overlapping requirements may mean they are not atomic
  - Separate out the individual, atomic requirements and delete any duplication
- Where requirements duplicate each other, either delete one of them or merge them where each has useful information

It is good practice to cross-reference requirements when deleting or merging. Some Requirements Engineers create a new requirement that incorporates the merged requirements and archive the requirements that are merged (so as to never re-use requirement IDs and thus allow the requirements to be fully traceable).
Conflicts

Sometimes requirements will conflict with one another, sometimes a requirement may contradict another. In either case the analyst must work to resolve the issue.

- Conflict can exist for many reasons, and is completely normal, e.g. security of the systems vs usability of the solution
- The differing views (perspectives) of the stakeholders in conflict needs to be properly understood
  - Identify the areas of agreement / disagreement and facilitate discussion to find a compromise position that still meets the main objectives or allows the stakeholders to sell the new position to colleagues
- Be prepared to escalate but understand the consequences.
- Also be aware of conflicting data requirements, e.g. data manipulated does not match the data captured
Ambiguity

Ambiguity is inherent in requirements definition as requirements are typically expressed using text.

Business Analysts can also add ambiguity through documenting requirements using language they themselves understand. This is no guarantee that another reader will interpret them in the same way.

Business Analysts can also try to add some variety into their requirements through use of language – we have to maintain and use simple, standard terminology, despite it being boring to write!

Stakeholders may express their requirements in a way that makes perfect sense to them, but may mean other things to other people. This is particularly true of non-functional requirements.

For example, if the stakeholder says that they want the system to be easy to use we need to define what they mean in measurable terms.

‘Easy to use’ may be measured by means of a test or a survey with acceptable results defined in advance, e.g. ‘Easy to learn’ means that a novice can pass a competence test with score of at least 70% after 1 hour of training.

Dealing with ambiguity

There are some simple steps we can take to ensure we are not introducing or retaining ambiguity in our requirements:

- Use clear, simple, standard language expressions, see below
- NFRs need to be defined in measurable terms – use standards
- Models often support driving out ambiguity
  - In this course we look at the effect of using the requirements to interpret a data model
- Consider the testing – if you can’t test it, it might be ambiguous
- Use standard requirements templates
- Use standard definitions
  - E.g. define what a functional, non-functional, technical requirement means in your team
• Use a glossary and distribute it to all stakeholders
• Use prototypes to help clarify understanding

**Structuring your requirements**

There are many ways to write requirements. Good practice is to use the following notation:

1. User (or user role)
2. Verb phrase
3. Object or Noun phrase

For example:

• The Help Desk Supervisor shall be able to view all outstanding tickets by date

Note: using ‘shall’ also helps avoid any implicit prioritisation that Must, Should, Could or Would may bring.
Exercise 3

Analysing Requirements

Open the exercise workbook and with reference to the Goatilicious Case Study scenario do Exercise 3.
Testability

- Testers should be invited to reviews of requirements specifications and this should start early on so that the requirements can be properly formed
- Only a few testers, preferably the more experienced or senior ones, need be involved

This approach helps to limit those nasty surprises that tend to appear towards the end of the project. In turn, this should improve overall productivity.

Note that if testers have been involved in the initial reviews of requirements, they should also be involved in reviews of changes to requirements.

*Testing should be built in to the project from the outset.*

What the Testers look for

Primarily, Testers are thinking “Can we test this requirement?”

- Do we understand the requirement as written?
  - Is it too vague or ambiguous to test?
- Can we create a test script directly from the description?
  - Do we understand the data sufficiently to write the test?
  - Do we understand the constraining business rules?
  - Do we know the pre-conditions for the tests?
  - Do we know the post-conditions for the tests?
  - Do we know the required response to exceptions?
The test scripts written at this stage are high-level detail, e.g. of user interfaces, will follow. A requirement that cannot be tested is no use to anybody. However, operational requirements may be covered by SLAs.

The test plans written at an early stage may be high level but at least some assurance of testability has been obtained.
Prototyping Requirements

What is Prototyping?

- Prototyping is a technique that enables the development of a set of screens for one or more requirements
- The screens might be true GUI screens or simply mock-ups and system functionality might be actual or simulated

Prototyping is a good method to refine requirements by avoiding the “blank page” syndrome. It is especially suitable in ‘greenfield’ scenarios, or where major changes carry high risk. Once the broad requirements are known, an initial screen set (and perhaps functionality) is built which can be tested for usability and fitness for purpose.

Why do it?

The users need to be involved so they can confirm acceptability, understanding or specify further changes (i.e. requirements).

- It is considerably easier to validate requirements against even simple prototypes
- It helps the reviewers visualise the requirement and spot errors and inconsistencies
- Evaluates the initial look and feel
- Promotes shared responsibility for the requirement
- Results in early error identification and resolution

The dangers with prototypes include scope creep, and slipping into design mode, before we have clarified the requirements.

Whether full application code and data are used is a decision that needs to be made before prototyping starts, as this can cause issues where users see further advancement than is actually the case.
Types of Prototype

Paper

- Screen mock ups are created on paper or sticky notes and used to walk through requirements scenarios
- This method can be surprisingly effective and it also avoids the issues with ‘designing’ the solution

User Interface

- An IT user interface prototype is created but the functionality is lacking or partly simulated

Full Prototype

- Both functionality and interface are created using the technology that will be used

Note: for requirements purposes, it is better if the prototype is not too realistic. We want to try to avoid a design or solution focus as users may lose the purpose of the prototype (and start critiquing the design or solution!).

See the QA++ on Wireframes further on if you’re interested in creating solution or design agnostic prototypes.
Verification

Verification is the process of ensuring that the requirements are ready to be validated. We verify that we have captured the requirements correctly and they are complete before we gain agreement.

A final check once all filters have been applied and prototyping completed:

Is each requirement well-written, clear and SMART?

Are the requirements ready for validation?
Wireframes

Wireframing is a tool used in Information Architecture (designing screens and deciding how to organise them). It is commonly associated with website design.

Whilst it is not a skill that a Business Analyst is expected to master, using wireframes can help to avoid some of the pitfalls of prototyping.

A wireframe is similar to an Architect’s schematic: whilst the customer can easily see where the doors and windows are, the size of the rooms and the overall layout, they do not at this stage get distracted by what the wooden floor will look like in the hallway.

The idea of wireframing is that the skeleton of a screen can be shown with all of the components which can sometimes be helpful in allowing stakeholders to visualise what they will need, and therefore draw out the requirements. Traditionally, wireframes are created using paper templates but more and more there are software tools which do the same job.

We use modelling in a similar way, as you will see later on.

Some common tools for wireframing, aside from pen and paper, can be found at the following:

http://balsamiq.com/

http://www.axure.com/

Illustrator, Omnigraffle and Visio all have templates that can be downloaded to use for wireframing.
Exercise 4

Prioritising Requirements

Open the exercise workbook and with reference to the Goatilicious scenario do Exercise 4.
**Summary**

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements are often couched in ambiguous terms.</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>Requirements are hierarchical in nature.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Business requirements are divided into General and Technical.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Solution requirements are divided into functional and non-functional.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NFRs constrain how functional requirements deliver their benefit.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NFRs are uncovered using a combination of top-down and bottom-up analysis.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>It is important to prioritise requirements to allow phased implementation (most important first) and to manage the stakeholders’ expectations.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A common approach to prioritising requirements is MoSCoW.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Requirements should be filtered for:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Priority, Realism and Feasibility</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Overlaps and Duplication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Conflicts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Ambiguity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Testability</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Prototyping allows the analyst to help the stakeholders firm up their requirements, particularly in graphical areas.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Verification is the process of ensuring that the requirements are ready to be validated.</td>
<td></td>
</tr>
</tbody>
</table>
Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You'll find the answers on the next page.

1. The Requirements Engineering Process is built of four steps, run iteratively. Three of these are Elicitation, Documenting and Negotiation. What's the fourth?
2. Two major groupings of requirements fall out of the Business Objectives. What are they?
3. General and Technical Requirements make up which major group of requirements?
4. To which group of requirements does “Interoperability” belong?
5. What is a Non-Functional Requirement?
6. What is the acronym commonly used when prioritising requirements?
7. What are the three main levels of prototyping?
Answers to Module 3

1. Analysis.
2. Business and Solution.
5. A requirement which constrains the way in which a functional requirement operates.
6. MoSCoW.
7. Paper, User Interface, Full.
Further Reading

1. SWEBOK Manual Chapter 2
   IEEE: http://www.computer.org/portal/web/swebok/html/contentsch2#ch2
Module 4 – Use of Models in Requirements Engineering

Topics

In this section of the course, we will cover:

- The purpose of modelling requirements
- Modelling the business context for the system
- Developing a model to represent the system processing requirements
- Interpreting a data model
What do you think?

Why do we model requirements?
The Purpose of Modelling Requirements

Requirements are modelled in order to check understanding, completeness and to provide a vehicle for clear communication between all parties in a project.
Modelling the Business Context for IT systems

The process model above is often used to confirm requirements. Only some of the interactions shown will need to be supported through IT functionality.

- The Business Analyst will usually model this using some form of process model, typically a flow chart with swimlanes, as shown above.
- Business Analysts also make use of techniques such as Scenario building that we have mentioned earlier, and perhaps even create personas (fictitious users).
  - Personas help us to understand how the system will be used by different types of users; encompassing their capabilities, expectations and needs.
- Each Task in the process model is examined to determine what IT support could be deployed. Conversely, if we are given an IT requirement we need to know what Task within what Process it supports.
Modelling Functionality and Data

- We need models to communicate and describe the functionality and data requirements for our system
- These models help us to ensure we have a complete view of where users need interaction with IT to complete their goals and what data will support that process from start to end

For example, for the process 'Rent Vehicle':

- For the Record Rental activity, a sales person needs **functionality** to allow them to capture customer details such as name, address, contact details and billing information which would result in data groups detailing Customer, Vehicle, Rental details and Address
- For the Take Payment activity, a sales person needs **functionality** to advise on the Billing details (who to take payment from), Address and Payment details (how much and method of payment)
- For Record Return, the sales person needs **functionality** to find the vehicle and record the inspection details against as well as log that vehicle back in so other customers can rent it

Each function also uses data that is then passed between each role.
Use Cases

The use case diagram is used as a summary of the required functional requirements for the IT system.

Use cases:

- Encapsulate business functionality in a highly graphical way, and represent the user’s perspective and needs
- Use cases can be identified by:
  - Examining process models
  - Reviewing documentation, procedures
  - Interview, workshops etc.

Each use case is supported by descriptions showing the “happy day” path and the exceptions / alternates which can arise

- Use cases can also be supported by prototypes and other forms of documentation and diagrams
- Optionally, they may be further supported by activity diagrams

Once we have an idea of the functionality our users will require, we can begin to document this using a Use Case Diagram.

What is a Use Case?

A use case is a ‘structured requirement’ that describes a piece of system functionality. It describes the ‘what’ not the ‘how’ (that comes later!).

- The use case is expected to deliver a result of value to an actor, allowing them to achieve a goal they have of the IT system; a ‘case of use’, hence ‘use case’
- A use case is a description of the external sequence of events between an actor and the system needed to meet the actor’s goals
- All use cases are, by definition, within the scope of the system. The sum of the use cases is “The System”
- The naming of use cases is important:
Use cases should be named using a verb-noun phrase such as ‘Create Purchase Order’, ‘Accept Delivery’ etc.
Use Cases: levels of detail

There are a number of different levels of detail that we can use to describe our functionality within the use case notation.

**Context Diagram**

The first is a **Context Diagram**. This describes the actors of the system and the system boundary itself, but treats the system as a ‘black box’. We do not need to go into the detail of these here.

**Use Case Diagram**

The diagram shows interactions between the Sales Team and the Workshop Supervisor, with activities like Rent Vehicle, Record Inspection, and Record Return.
The next level is the **Use Case Diagram** itself. The context diagram is expanded and we now show what happens within the system boundary using Use Cases to represent the goals of the users. This step is useful in analysis because it helps the Analyst uncover what functionality the user needs from the system.

**Use Case Description**

1. Display vehicle list
2. Select vehicle
3. Display available dates
4. Select start date
5. Select duration…
6. .................

The next level is the **Use Case Description**. This expands on a single Use Case and describes the pre-conditions, inputs, outputs, post-conditions, actors, business rules etc. that govern the use case.

Contained within is also the **Use Case Description Flow**; the main dialogue that needs to occur between the users and the system, and the alternative paths. This is sometimes represented as an activity diagram.
This course covers Use Case Diagrams and Use Case Descriptions (primarily the main flow with a summary of the alternative and exception paths).

In the exam you may be asked to draw a Use Case Diagram to represent a set of requirements or you could be asked to describe a Use Case Description Flow.
A use case can also show different types of actors:

- A **primary actor** triggers the use case…
- Whereas **secondary actors** provide input to the system and/or receive output from it
Key points about Use Cases

- Use cases are always described from the actor’s point of view
- Use cases yield a result of value to that actor in the context of their work, and achieves a goal of the user
- Use cases support activities in business processes and therefore describe only interactions with the system and not events outside
- They should not be confused with business processes:
  - The use case is a piece of IT functionality
  - A business process is an end-to-end set of business activities, some of which may be supported by use cases

Use Case Notation

Use case diagrams are very simple to understand.

We can clearly and unambiguously define the scope of the project, the users and their roles. If we are building or updating multiple systems, then each system can be documented separately, allowing us to ‘see’ the requirements of each system and therefore each actor’s needs separately.

If no association exists between an actor and a use case then, when the system is built, it should NOT be possible for a person logged on in the role of that actor to gain access to that use case.

The main symbols are Actor, Use Case and User Association:

In the diagram above, the actor ‘Sales Team’ is involved with the use case ‘Record Rental’.
In UML 2 the association should only have an arrow on it when an actor is the recipient of output but hasn’t been involved directly with the use case.

**Use Case Diagram**

The use case diagram encapsulates the system boundary, actors and functionality in an easy-to-understand deliverable.

From our process model we can see that there are requirements for the following:

- The Sales Team shall record Vehicle rental details in the Vehicle Rental system
- The Sales Team shall record Vehicle return details in the Vehicle Rental system
- The Sales Team shall be able to search for Vehicles in the Vehicle Rental system
- The Workshop Supervisor shall record the results of an inspection in the Vehicle Rental system
- The Workshop Supervisor shall be able to search for Vehicles in the Vehicle Rental System
What do you think?

The use case also covers one category of Non-Functional Requirement… what is it?
Exercise 5(i)

Document Management System

In the exercise workbook you will find exercise 5 ‘Document Management System’.

Complete Part (i) of this exercise.
The «include» Relationship

There are pieces of functionality within a system that are common and used multiple times.

The example below shows a member of the Sales Team renting out a vehicle. When (s)he initiates this use case, it (in turn) triggers the functionality to take the payment for the hire. Take Payment is a common piece of functionality.

The «include» relationship indicates that this action is always done.

Using relationships like this permits a degree of re-usability of subordinate use cases like “Take Payment”.
Use Case Diagram with <<include>>

The diagram below shows the previous diagram, this time using «include».

![Use Case Diagram](image)

In this example (although not shown in the process model) a payment must be taken when the rental is recorded and when it is returned. Both use cases require access to the same functionality, so we have a common use case, ‘Take Payment’.

**The «extend» Relationship**

The extend relationship shows pieces of major functionality that only occur under certain conditions:

- If this functionality is very different and has some complexity it may be better to model it as a separate use case to avoid over complicating the main functionality
- In the process model we have a condition whereby if the fuel tank isn’t full when the customer returns the vehicle we need to take a payment. If the customer has returned the vehicle with a
full tank the vehicle passes the inspection and nothing happens except to record the return

- The diagram below shows that “Record Return” may be extended by the functionality of “Take Payment” if the customer returns the vehicle without filling up the petrol tank.

Sales Team
Use Case Diagram Complete

Here we can see the complete use case diagram, using normal associations and those with «include» and «extend».

Note that we have taken out the ‘Search Vehicle’ use case for simplicity. Use case diagrams by their nature can get quite complicated!
Exercise 5 (ii)

Document Management System

In the exercise workbook you will find exercise 5 ‘Document Management System’.

Complete part (ii) of the exercise.
Exercise 6

Use Case Diagram

In the exercise workbook, with reference to the Goatilicious case study scenario complete Exercise 6.
Modelling Data

So far we’ve looked at the functionality needed to satisfy and confirm the requirements. But what about the data?

The Use Case ‘Record Rental’ will allow the sales person to meet their goal to log the rental on the system and will also capture information that will identify the Customer (including we would assume such information as driving licence details), the Vehicle and the Rental period, enable Billing and update the customer account (if necessary).

Why model data?

A clear understanding of the meaning and structure of the required data is at the heart of effective requirements analysis and systems design. This is true of projects both large and small and in all technical environments.

Data analysis is the technique of examining the data requirements of a part (or all) of the business to produce a model of that data that will support the process requirements. The scope of investigation can vary from a relatively small application system to the organisation as a whole.

There are two common standards in use for modelling data: the Entity Relationship Diagram from Structured Systems Analysis and the Class Diagram from the Unified Modelling Language.

In this syllabus we look at the Class diagram from the increasingly popular UML notation.
A thorough data model is necessary for the following reasons:

- To determine the data requirements of the system
  - What data does the system need to store?
  - Omitting required data from the requirements can have very serious consequences, nearly always more serious than omitting required processing

- It provides a common understanding of the data
  - This is vital if different applications, or different components of an application system, are to share data

- It is the basis of the technical implementation
  - The data model is the start point, no matter what the eventual technical implementations are to be

- It defines integrity rules about the data
  - In a particular business, data is not haphazard; it has a structure or rules that reflect rules about the business

**The Class Diagram**

The data model defines what data is required, how it is structured, what it means and the integrity rules about that data. The data model consists of full descriptions of each:

- Class – a type of thing of interest to the business, the groupings of data
- Attribute – lowest level data item, a property of a class, the individual pieces of information we need about the data grouping
- Association – a link between classes, the business connections

The data model is a *conceptual model* of the business data.

That is, it models the data requirements independent of:

- The technology that will be used to store the data
- The way in which individual processes require to view the data
- Any performance constraints
A Class is...

... any *thing* about which data information is to be recorded.

In a garage some classes would be:

- **People and Organisations** like: Supplier, Customer, and Employee
- **Things** like: Vehicle Model, Vehicle, and Part
- **Events** like: Sale, Service
- **Places** like: Garage (assuming they have several)
Components of a Class

A class is shown as a simple, rectangular container with its name (a singular noun) in the uppermost of its two compartments. If the name has more than one word, the first letter of each should be capitalised and spaces should be omitted. Example names are:

- Customer
- QuoteType
- CourseBooking
- BankAccount

Think of a class as a template for the group of data you wish to know about.

Every Customer you have on the system will be shaped in the same way, as will every Quote Type and every Bank Account.
Attributes are...

... the *things, pieces of data*, we need to know about the class.

Note that each attribute can hold only one value at any one time.

Attributes are shown in the lower compartment of the class, as you can see here.

Attribute names are normally shown with a lower case first letter and all subsequent words in the name having uppercase first letters (‘camel case’).

We do not always show the attributes in the class model – sometimes at the early stages of the analysis we are simply trying to understand the data groupings and the business connections between them.

Again, think of the template – each Customer you have on the system will have a Customer Number, a Name and an Address.

Operations are...

- *The procedures you can perform using the attributes in the class*
- *They are invoked by messages sent from other classes*
- *The attributes within a class are only accessible by the operations of that class (known as encapsulation)*
- *If we wanted to update a credit limit:*
  - A message would be sent to the Account class with the Account Number and the Name and the amount the Credit Limit needs to be adjusted to
An Association is...

... a logical, meaningful business connection linking two classes.

- Delegate ATTENDS course
- Customer MAKES booking
- Passenger TRAVELS ON train
- Employee SELLS vehicle

For the last example, we have one possible association between the Employee and the Vehicle classes.

It will support the requirement to know, for a given Salesperson, which Vehicles they sold; and for a given Vehicle, which Salesperson sold it.

Associations need to be named precisely to reflect the business rules they are supporting. We only include those which are direct and of interest to the business.

Association

In UML an association is shown as a simple line between the two classes.

To enhance meaning, it is often also labelled and given an arrow to help understand the directionality of the description.

Note that the label and the arrow are not always present.
Multiplicity of the Association

Numbers, at each end of an association, indicate the range of numbers of instances of the class allowed in the relationship: the multiplicity.

Note that the asterisk means an unspecified “many” in UML.

Possible values for these multiplicities include:

- 0..1
- 0..10
- 0..* (or just *)
- 1..*
- 1..1 (or just 1)
- 3..12

Note that the separator in these ranges is a double dot, not a triple dot, which means an ellipsis in English.

- A ‘1’ on its own means 1..1, i.e. 1 and only 1.
- A ‘*’ on its own means 0..* i.e. zero to many

Here is an example of the use of multiplicities:

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-registration</td>
<td>-customerNumber</td>
</tr>
<tr>
<td>-make</td>
<td>-name</td>
</tr>
<tr>
<td>-model</td>
<td>-homeAddress</td>
</tr>
<tr>
<td>0..1</td>
<td>owns</td>
</tr>
<tr>
<td>0..*</td>
<td>0..*</td>
</tr>
</tbody>
</table>

A Customer owns zero or more Vehicles.

A Vehicle is owned by zero or one Customer.
Reading the Class Model

The direction of the relationship (if present) helps decide the requirement, starting from the Customer end.

Start with the class pointing away from the relationship, in this example that’s Customer.

**Customer** (class) **Owns** (relationship) **zero or many** (multiplicity: 0..*) **Vehicle(s)** (class).

And reading back the other way:

**Vehicle** (class) **is owned by** (relationship) **zero or one** (multiplicity: 0..1) **Customer** (class).

You may ask how a vehicle can be “owned” by zero customers but consider those vehicles which have not yet been purchased – the model must hold ‘true’ at all moments in time.

The fact that a Customer can own zero vehicle, allows the customer to be present on the system before they own anything and therefore allows for records to be kept that log enquiries and information sent. Perhaps while the customer is in the pre-purchase phase.
Generalisation Association

A generalisation association is not between classes but is a form of documenting a single class:

- The Base Class holds all the common attributes
- The Sub-classes inherit these but have their own attributes too
- In this example, Current Account and Savings Account are both types of Account

This strategy will make it easy to add new types of account later, if needed, without changing the existing definitions.

[Diagram showing generalisation association between Account, CurrentAccount, and SavingsAccount with attributes accountHolder, accountNumber, branch, balance, overdraftLimit, and rateOfInterest.]
Modelling Requirements

The question, now, is: “How do we take our requirements and construct a data model which supports them?” Consider the following requirements:

- The system is to record for a given Employee, which Vehicles they sold as new
- The system is to record details of all Services performed on each Vehicle
- The system is to record which Vehicles are currently owned by which Customers

You can see that there are four key “things” here:

- Customer
- Vehicle
- Service
- Employee

This will give us four classes. You must consider which of these relate to which others and how many of each there might be at each end of the association.

See the suggested solution below:
The CRUD Matrix – Cross Checking

Recall that use cases are built from **functional** requirements. Do all the use cases actually *do* something? Is all the data used? Does the system process data items that are never created? Can data be deleted?

As with the identification of functional requirements we can use CRUD.

In this case we can use a CRUD Matrix to ensure that we know which use cases create, read, update and/or delete each entity.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Class</th>
<th>Customer</th>
<th>Vehicle</th>
<th>Employee</th>
<th>Service</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take Payment</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
</tr>
<tr>
<td>Record Rental</td>
<td>R</td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Record Sale</td>
<td>R</td>
<td>RU</td>
<td>U</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Manage Customer</td>
<td>CRUD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Vehicle</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Service</td>
<td>R</td>
<td>U</td>
<td>U</td>
<td>C</td>
<td></td>
<td>R</td>
</tr>
</tbody>
</table>
Housekeeping Use Cases

Often we may find that we are missing a use case whose job is, for example, to delete obsolete service records (in our garage scenario). This might simply be because it never came up during interviews and the analyst didn’t think of it at the time. It may be that a completely different system is responsible for this but at least we will have checked. The CRUD Matrix helps us do this.

Use Cases like this are often called “housekeeping” use cases and often do not appear in the project until some form of verification is performed.
Exercise 7 – At the Vet

In the exercise workbook you will find Exercise 7 ‘At the Vet’.
Exercise 8

Class Diagram

In the exercise workbook with reference to the Goatilicious scenario complete Exercise 8.
Conclusion

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements are modelled in order to check understanding/completeness and to provide a vehicle for clear communication between all parties in a project.</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>A Process Model shows the context for the IT System.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>A use case is a ‘structured requirement’ that describes a piece of system functionality.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>An actor is an external entity performing a named role with respect to the system.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Use cases are always described from the <strong>actor’s</strong> point of view and yield a result of value to that actor.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Use Cases should not be confused with processes.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>The «include» relationship indicates that when the primary use case is triggered, the subordinate one is <strong>always</strong> subsequently triggered, too.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>The «extend» relationship means that when the primary use case is triggered, the subordinate one will also be triggered if the specified condition is met.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>The domain class diagram is a model of the business domain’s data which defines the entities, their attributes and the associations between them.</td>
<td></td>
</tr>
</tbody>
</table>
### Module 4: Use of Models in Requirements Engineering

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10</strong></td>
<td>A class is shown as a simple, rectangular container with its name, a singular noun, in the uppermost of its two compartments.</td>
</tr>
<tr>
<td><strong>11</strong></td>
<td>Attributes are the things we need to know about the class.</td>
</tr>
<tr>
<td><strong>12</strong></td>
<td>An association is a meaningful relationship between classes. It has a name and direction to enhance meaning.</td>
</tr>
<tr>
<td><strong>13</strong></td>
<td>Multiplicities are the number of possible instances at each end of a relationship.</td>
</tr>
<tr>
<td><strong>14</strong></td>
<td>The CRUD Matrix affords the analyst a useful way to check that the functionality of the use case diagram is complete.</td>
</tr>
</tbody>
</table>

### Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You’ll find the answers on the next page.

1. What is an Actor?
2. What is a use case?
3. From which perspective are use cases described?
4. What is the only occasion in which you would expect a normal use case association to have an arrow?
5. What are the conventions to be applied to class names?
6. How many values can a class attribute normally hold?
7. What do we call the numbers at each end of an association between two classes?
Answers to Module 4

1. A named role in relation to the business in scope.
2. A use case is a ‘structured requirement’ that describes a piece of system functionality.
3. From the point of view of the actor.
4. When the actor is the recipient of the output of a use case but is otherwise uninvolved.
5. The name should start with a capital letter. Names consisting of multiple words should not be separated and they, too, must have initial capitals.
6. One.
7. Multiplicities.
Further Reading

End of Module Notes
Module 5 – Requirements Documentation

Topics

In this section of the course, we will cover:

- Documentation styles
- User stories
- Use cases
- Requirements catalogue
What do you think?

How do you usually document requirements?
Documentation Styles

There are many styles of documentation to choose from:

- Catalogue
- Business Requirements Document (BRD) or Business Requirements Specification (BRS)
- Prioritised Requirements List (PRL)
- Functional Specification
- Software Requirements Specification (SRS)
- User Story
- Use Case
- IEEE 830, IEEE 1233, and so on

Often the decision about which documentation to use will depend on existing standards within your organisation.

It may also be governed by the nature of the deliverables you need, the project methodology and even the experience of the resources working on the project.
**Agile Approach**

Agile is an incremental, iterative development methodology that is promoted through self-organising teams and cross functional working practices.

**Agile Manifesto**

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

<table>
<thead>
<tr>
<th>Individuals and interactions</th>
<th>over</th>
<th>processes and tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Working software</strong></td>
<td>over</td>
<td><strong>comprehensive documentation</strong></td>
</tr>
<tr>
<td>Customer collaboration</td>
<td>over</td>
<td>contract negotiation</td>
</tr>
<tr>
<td>Responding to change</td>
<td>over</td>
<td>following a plan</td>
</tr>
</tbody>
</table>

That is, while there is value in the items on the right, we value the items on the left more”

In Agile, documentation is kept to a minimum due to the evolutionary nature of development, but contrary to popular belief this doesn’t mean that documentation doesn’t exist at all.

The artefact nearest to a ‘requirements’ document is the User Story and the Business Analyst typically manages the Product Backlog, as well as liaising with Product Owners and the project team to ensure that the ‘requirements’ meet the criteria of the business objectives.

There are many different variations of agile, including DSDM, Scrum and XP.
User Story (Agile)

- User stories are often brief, informal expressions of a requirement, written by the stakeholder and representing some feature to be developed
- User stories are used heavily in XP and other agile methodologies where there is close contact between the customer and the developer
- Very little documentation is produced (or even expected).

```
FRONT

As a user I want to be able to

Compare vehicles side by side
so I can see different features and prices

Priority: Should

BACK

- Vehicles added to the shortlist must be of the same type (eg: passenger vs CV)
- Up to 3 vehicles can be added to the shortlist
- A minimum of 2 vehicles must be in the shortlist to enable compare
```
Often the User Stories are written in the format:

| As a [role], I want to [use a feature] so that I can [achieve a goal/rationale] |

For example: “As a customer I want to register my details on the site so that I can book a hotel room quickly whenever I need to”.

User Story Characteristics

<table>
<thead>
<tr>
<th>Brief, informal expression of a requirement</th>
<th>Written by a business stakeholder</th>
<th>The basis of a feature to be developed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close contact between customer and developer</td>
<td>Little documentation</td>
<td>Used in XP and other agile methodologies</td>
</tr>
</tbody>
</table>
Reminder: Use Case Diagram

Use cases are the goals that the user requires of functionality from the system. They are linked to the actors through the associations in the Use Case Diagram.

Record Rental, Record Return are the goals of the actor ‘Sales Team’ in the above example.
Individual Use Case Documentation

Use Cases are documented as dialogues:

- Detailed description of the interaction between the user and system
- Primary scenario (‘happy/sunny path’) documented first
- Alternates and exceptions documented separately
- Pre-conditions and post-conditions
- May be supported by activity diagrams, or shown in a tabular format

You'll sometimes need to use a table to describe the interactions.

<table>
<thead>
<tr>
<th>Actor</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>M01</td>
<td>Selects vehicle type Displays list of available vehicles</td>
</tr>
<tr>
<td>M02</td>
<td>Selects vehicle from list Displays options form</td>
</tr>
</tbody>
</table>
Use Case Description: elements

The use case description is made up of elements that summarise the main requirements of the functionality needed to achieve the users goals, such as:

- What needs to be in place prior to the use case starting
  - The pre-conditions
- What needs to be there once completed
  - The post conditions
- The use case narrative, structured as a dialogue between the Actor (the User) and the IT system
  - The dialogue should refrain from using any language that suggests a solution

Although the Use Case Diagram is a part of the UML and, therefore, subject to its strictures, the Use Case Description has no template prescribed by OMG (Object Management Group).
Use Case Description Template

The following headings are commonly used:

- Name (verb-noun phrase)
- Identifier (unique identifier for this use case)
- Description (a few sentences describing the basic intent of the use case)
- Primary Actor (who/what initiates the use case)
- Secondary Actors (any other actors that may be involved)
- Preconditions (things the system can ensure will be true before the use case starts)
- Primary Scenario/Main Flow (describe the “normal” processing path as a dialogue between Actor and System)
- Alternate Flows (describe any variations on the main flow)
- Post conditions (the state of the system on completion of the use case)
# Use Case Description Example

A simple example for our Vehicle Rental system:

<table>
<thead>
<tr>
<th>Heading</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Record Rental</td>
</tr>
<tr>
<td>Identifier</td>
<td>UC01</td>
</tr>
<tr>
<td>Description</td>
<td>Sales Person selects the vehicle, rental period, and records the rental, as well as taking payment</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Sales Person</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>• Sales Person logged in, Rental screen displayed</td>
</tr>
<tr>
<td></td>
<td>• Customer selected</td>
</tr>
<tr>
<td></td>
<td>• Vehicle and Vehicle status on system</td>
</tr>
<tr>
<td></td>
<td>• Hire Rates on system</td>
</tr>
<tr>
<td>Main Flow</td>
<td>See below</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>See below</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>Rental recorded, vehicle status updated, payment taken</td>
</tr>
</tbody>
</table>

If a question asks you to document the interaction between users of a system you will be producing a Use Case Description – Main flow. It is unlikely you will be asked to document, in detail, the alternative flows but you may be asked to identify a specified number of them.
Use Case Description – Main and Alternate Flows

It is common practice to document the ‘main flow’ of the Use Case, the narrative, and then specify a number of alternative flows, to cover other possible ways things might go, and to specify exception handling. ‘Normal Path’, ‘Happy Day’, ‘Sunny Day’ are all names commonly used as labels for the main flow.

It is simply, what the use case should achieve to satisfy the users goals.

Main Flow

1. Sales Person selects ‘New Rental’
2. Vehicle Rental System displays Vehicle selection list
3. Sales Person selects vehicle
4. Vehicle Rental System displays available dates
5. Sales Person selects start date required (A1)
6. Sales Person selects end date required (A2)
7. Vehicle Rental System confirms selections
8. Sales Person confirms rental period (A3)
9. Vehicle Rental System calculates and displays price
10. Sales Person selects payment method
11. Vehicle Rental System records payment and updates vehicle status

Alternate Flows

- A1. Vehicle not available for selected start date
- A2. Vehicle not available for selected end date
- A3. Sales Person wishes to change rental period

Note how the flow is specified as a dialogue between the Actor and the IT System.

Note also that there is no solution language used here – the Use Case describes the requirements, as seen from a user’s perspective.
Column/Table Use Case Format

A columnar / table format for specifying this dialogue is also used, where the Actor / System dialogue structure may be clearer.

<table>
<thead>
<tr>
<th>Step</th>
<th>Sales Person</th>
<th>Vehicle Rental System</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Selects ‘New rental’</td>
<td>Displays Vehicle selection list</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Selects vehicle</td>
<td>Displays available dates</td>
<td></td>
</tr>
</tbody>
</table>
| 3    | Selects start date
Selects end date | Confirms selections                  | A1, A2           |
| 4    | Confirms rental period         | Calculate and display price               | A3                |
| 5    | Select payment method          | Record payment and update vehicle status  |                   |
Requirements Document

A typical Requirements Document is likely to contain at least the following sections:

- **Introduction and Background**
  - Describes the scope of the project and its objectives

- **Business Process Models**
  - Shows the “to-be” model as swimlane diagrams

- **Function Models**
  - Context diagrams and Use Case diagrams

- **Data Models**
  - A clear picture of the data requirements of the project in entity-relationship diagrams and/or class diagrams

- **Requirements Catalogue**
  - A detailed description of each requirement – we’ll be looking at this in more detail shortly

- **Glossary**
  - A clear definition of the terms used – it might be for this project only or organisation-wide
Business Dialects

Particularly in large, geographically-dispersed organisations, the tendency for regional business dialects to grow needs to be carefully considered. This is also true of organisations, which might be based in a single region but have a number of independent product lines or “silos”.

For example, one business line in the company might refer to a customer as an “Opportunity”, while another uses the more traditional name for the entity (in this example the more traditional name would be ‘Customer’).

“Customer” and “Opportunity” are referred to as synonyms. We are able to tell that they refer to the same entity because they will be used in the same way and will have an identical set of attributes.

A slightly more difficult situation to detect is that of homonyms. Here, we refer to two different entities, using the same name. The classic example in IT of an homonym is the “mouse”. Clearly, it is a pointing device but it is also the word for a furry rodent. The set of attributes will be very different and also the way in which the entity is used.

The purpose of a glossary is not only to define terms unambiguously but also to resolve issues like these.
**Sample Requirement Catalogue Entry**

<table>
<thead>
<tr>
<th>Author</th>
<th>Date Captured</th>
<th>Version</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna Liszt</td>
<td>02/06/2015</td>
<td>0.1</td>
<td>In development</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>F-102v0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Check in unreserved customer</td>
</tr>
<tr>
<td>Description</td>
<td>Receptionist must be able to check in a customer who has no reservation, providing a suitable room is available</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Client checked in, only if suitable room(s) available for those nights</td>
</tr>
<tr>
<td>Source</td>
<td>B. Welcome, Receptionist</td>
</tr>
<tr>
<td>Owner</td>
<td>L. Snoop, Duty Manager</td>
</tr>
<tr>
<td>Priority</td>
<td>Must</td>
</tr>
<tr>
<td>Rationale</td>
<td>10% of hotel bookings are walk-in, value £200k p.a.</td>
</tr>
<tr>
<td>Associated NFRs</td>
<td>Must be possible to enter all data and confirm room(s) within 3 minutes</td>
</tr>
<tr>
<td>Related Documents</td>
<td>Interview notes B. Welcome 26/05/2015</td>
</tr>
<tr>
<td>Related Requirements</td>
<td>N-001 Access Permissions</td>
</tr>
</tbody>
</table>

**What do you think?**

Do you use all of these fields?

Do you think they are all needed?
Requirement Catalogue Components

ID and Version

- It is vital to give each requirement a unique ID and a version number.
- The ID can be used to hold the classification such as General, Technical, FR or NFR.
  - Many organisations would hold this as a separate ‘type’ field, leaving the ID to be neutral and if documented using a spreadsheet can be easily filtered and sorted.
- This section allows the management of requirements and enables traceability between requirements and to other documents.

Author

The Requirements Engineer who is writing the requirement on behalf of the business.

Date Captured

The date on which the requirement is captured.

Name

A short, readily understood description.

Description

This is a clear definition of the requirement. One could use the following structure (for a functional requirement):

- Actor
- Verb phrase
- Object (noun or noun phrase)

The amount of detail captured tends to vary over time:

- Initially, only the intent of the requirement might be captured.
- More detail will be added after further elicitation and analysis.
Status

This is the current status of the requirement in terms of whether it has been validated, signed-off, baselined and so on.

Acceptance Criteria

Acceptance criteria may be considered as:

- The criteria that will demonstrate to the business that the requirement has been met
- The basis for acceptance of the system by the client

For example:

- A customer order that would take the customer over their credit limit will be rejected and the customer advised that they may not place further orders until payments have been cleared

Source

Stakeholders are the usual source of requirements. Other sources include documentation, standards, problem reports, other requirements etc.

Note the source(s) and their role on every requirement since it helps you to:

- Improve traceability
- Return to the source in case of change

Owner

The business ‘owns’ all requirements. Usually businesses appoint individuals to own the requirements in their area. The owner has the responsibility for approving the definition of the requirement.

Priority (MoSCoW)

It is important to be able to refer to the priority.
The BCS preference is MoSCoW. (See module 3 for more information on MoSCoW.)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M</strong> – must have</td>
<td>• Mandatory in the first increment; cannot meet business and project objectives without it</td>
</tr>
<tr>
<td><strong>S</strong> – should have</td>
<td>• Mandatory but may wait until second increment; the system will have short-term value without it</td>
</tr>
<tr>
<td><strong>C</strong> – could have</td>
<td>• Beneficial if time or funds allow, but not central to project objectives</td>
</tr>
<tr>
<td><strong>W</strong> – want to have (won’t have this time)</td>
<td>• Will not be met in this delivery; may be included in a future delivery</td>
</tr>
</tbody>
</table>

Rationale

The rationale explains why the requirement is needed and what benefits will accrue.

It is the link between the problem and the requirement.

Just thinking about the rationale helps to ensure that the requirement is justified.

The rationale also helps to determine the priority of the requirement and may be cross-referenced to benefits detailed in the business case.
Module 5: Requirements Documentation

Links to other Requirements/Documents

- The ‘Associated Non-Functional Requirements’ section can be used to:
  - Link the requirement to any non-functional requirements that constrain it
  - Describe any non-functional requirements that pertain only to this particular requirement

- Related Documents
  - Relates this requirement to any other documentation, from this project or elsewhere, which is relevant.
  - This may include the ToR or PID for this project, company strategy documentation, etc.

- The ‘Related Requirements’ section links the requirement to other requirements that are related in some way, such as:
  - Higher level requirements, i.e. those on which this requirement is based
  - Requirements which conflict with this requirement
  - Requirements which have superseded this requirement
  - Requirements which have been superseded by this requirement
Exercise 9

Use Case Description

In the exercise workbook with reference to the Goatilicious scenario complete Exercise 9.
## Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>There are many styles of documentation to choose from, including Catalogue, BRD, PRL, Functional Specification, SRS, User Story, Use Case, IEEE 830 and IEEE 1233.</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>User stories are brief, informal expressions of a requirement, written by the stakeholder and representing some feature to be developed.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Use cases encapsulate business functionality in a highly graphical way supported by textual descriptions.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A typical requirements document contains several sections, including Business Process Models, Function Models, Data Models, Requirements Catalogue and Glossary.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A typical requirement is described using a number of fields, some of which may be regarded as optional in some circumstances.</td>
<td></td>
</tr>
</tbody>
</table>
Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You'll find the answers on the next page.

1. What factors determine the style to be adopted when creating a business requirements document?
2. Name a methodology which makes significant use of user stories.
3. What is the purpose of a Glossary in a requirements document?
4. Why do we specify a rationale for a requirement?
Answers to Module 5

1. Company conventions and type of application.
2. XP or Agile.
3. To provide a clear definition of the terms used in the project – it might be for this project only or organisation-wide.
4. The rationale explains why the requirement is needed and what benefits will accrue.
Further Reading

End of Module Notes
Module 6 – Requirements Validation and Management

Topics

In this section of the course, we will cover:

- Validating requirements
- Dealing with change
- Traceability
- Requirements Engineering support tools
Requirements Validation

The majority of errors are made in the Analysis stage of the project and particular care has to be taken to ensure that the requirements are properly formulated.

After the Requirements Catalogue has been analysed and negotiated, it **MUST** be validated and signed-off by the senior stakeholders.

- Validation is concerned with proving that the ‘final’ draft is as free from defects as possible, complies with standards and the product to be built seems to be fit for purpose.
Types of Review

- The appropriate degree of formality of a review will depend on the characteristics of the product, e.g.:
  - Major safety issues
  - Major compliance/legal issues
  - Strategic importance
  - Scope, Budget, Risk etc.

- Reviews range from quick informal reviews with the sponsor to formally documented structured reviews with all the main stakeholders present
- A forum for the validation of requirements is a formal review – a peer group or stakeholder group review for the purposes of finding deficiencies in the requirements

A Review is Not...

- An appraisal of the Requirements Engineer
- A requirements elicitation or analysis meeting
- A management review
- A think tank

When conducted properly, the review has proved to be the most effective method for finding defects in requirements.

The review can often help to resolve misunderstandings and misinterpretations between the various Stakeholders involved in the requirements.

The objective of a review is to increase the overall "quality" of the requirements that are being developed and also to increase user confidence in the development effort.

Thus, the idea is:
• To catch errors as soon as possible
• To establish user buy-in
• To check adherence to standards

**Formal Review – Roles**

**Project Sponsor**
Checking for alignment to business goals

**Business Owners**
Checking that their requirements have been dealt with satisfactorily

**Domain Experts**
Checking that the business ‘rules’ and practices have been respected

**Developers**
Assessing technical feasibility

**Testers**
Affirm the testability of the product

**PMO**
Checking the implementation project proposal against Project/Programme standards

**Requirements Engineer (e.g. Business Analyst)**
Explanations and clarifications

**Moderator / Chair and Scribe**
Management of the review
At the Review

The Moderator:

- Calls the meeting to order
- Reminds the group of the nature and the purpose of the walkthrough

The Requirements Engineer walks methodically through the requirements.

The Reviewers:

- Criticise, comment, discuss with the Author
- Present their comments about the product without reference to the Author – "egoless" concept

Remember that the purpose of the walkthrough is defect detection and not defect correction (that is the Author's job).

The Scribe:

- Notes and clarifies action points.
After the review

There are three possible outcomes from a review:

- Accept the catalogue in its present form
- Accept the catalogue with the suggested revisions (provisional sign-off), trusting that they will be made
- Plan another review after the defects are fixed

The Moderator:

- Produces a management summary of the meeting and a set of detailed comments made by the participants
- Sends copies of the above to all participants

Note that the summary does not state how many errors were found, only what was reviewed, when the meeting took place, who was in attendance and what the outcome was.
Components of Requirements Management

The tight control of requirements is absolutely necessary in all projects but particularly so in software intensive projects which are often large, long term and subject to extensive change. Requirements may exist, quite legitimately, in various versions and releases and for various target projects.

Each version of each requirement is supported by a raft of documentation including specifications, system designs, code designs and test data and results.

These products are heavily interrelated, such that a change to one may impact on hundreds of others.

Unless these products and relationships are tightly controlled, the project will quickly and inevitably become chaotic.

Requirements Management is a subset of Configuration Management which controls all project deliverables.

- In its wider context, Requirements Management encompasses the activities involved in gathering, recording, analysing,
validating, verifying, managing changes to and tracking requirements throughout the project lifecycle

Requirements Management is concerned with the following activities:

<table>
<thead>
<tr>
<th>Requirements identification</th>
<th>Requirements cross-referencing</th>
<th>Requirements origin and ownership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software support</td>
<td>Change control</td>
<td>Configuration management</td>
</tr>
</tbody>
</table>

- **Requirements identification**
  - Each requirement is uniquely identified
  - A reference to it corresponds to only one requirement
- **Requirements cross-referencing**
  - Allows the analyst to identify requirements related to the one that is to be changed
  - The basis for impact analysis
- **Requirements origin and ownership**
  - The source will provide information on the impact of change
  - The owner has ultimate authority
- **Software support**
  - Documentation
  - Secure storage and access
  - Linkage
  - Version numbering
- **Change control**
  - Documenting a proposed change
  - Consulting stakeholders
  - Making a decision
- **Configuration management**
  - Controls the change process
- Configuration items are identified, validated, baselined and subjected to version control as they are checked out and back in

**Fundamentals for Management**

- Uniquely identify each requirement
- Establish a change control authority
- Create a change process
  - Document it
  - Apply it consistently
- Establish system of baselines and versions
- Maintain history of changes
- Track status of each and every requirement
Baselined Requirement

The concept of the Baseline is fundamental to Requirements Management.

A Baseline is a product that has been formally reviewed and agreed upon, it is a frozen version of a requirement.

Thereafter it serves as the basis for further development and can only be changed through formal Change Control Procedures.

It should be noted that the same rigorous review procedures are used to examine changed requirements as were used to examine the requirements on first development.

The baseline concept provides many benefits for requirements management:

- Definitive versions of all items are available at all times and changes are performed against a known baseline: the baselined version is never overwritten
- It has been approved by management and associated with a version number
- Once baselined, any change requires approval
- All changes can be tracked and it is therefore possible to ‘fall back’ to a previous, working version if the new version is proved faulty
Stable and Volatile Requirements

We have already discussed that requirements engineering is an iterative process.

Part of the reason for this is that, in business, the only constant is change.

Some environments are more volatile than others by their nature.

- Requirements tend to be quite volatile initially, but as we gain agreement during analysis and negotiation they should become more stable
- All changes are now governed through a change control process

After validation, requirements should be “baselined”.

Sources of Change

- Change of key stakeholder(s)
- Change in project scope
- Changed or new legislation
- Altered business priorities
- Competitor action
- Compatibility with new technology
- Users change their minds
- Users develop better understanding of need
Management of changes to Requirements

The change control process must support:

- Change proposal
  - Detail about the change and how the change will be managed
- Impact analysis
  - Cost; risk; scope; benefits; quality; time
  - Consider link to project objectives
- Resolution
  - Approval or rejection

All changes must be managed through the process.

Uncontrolled changes can lead to problems managing the budget, resources, testing and priorities.

Change Control approvals

Some organisations have formal Change Control Boards or Change Control Authorities – a group that meets to review and decide on whether changes will be approved or rejected.

- It is important to recognise though that not all Change Requests are the same and may be different in scale, impact and complexity
- It is good practice to have different levels of change authorities for different types of change
- The change process is built into the Project or Programme methodology (which may be company-wide or specific to the project needs)
- Some changes are reviewed at a project or requirements level by the Business Analyst or Project Manager, whilst others may have to go to the Change Board
- Each change is reviewed against cost, urgency, risk and impact
- The sponsor may be called upon to make the final decision
Traceability

If we manage our requirements properly we can also ensure full traceability forward from the requirements through to the implemented product and backwards from the implemented product to the requirements.

In Requirements Management, there are typically two types of traceability:

**Vertical Traceability:** backwards to the business objectives

**Horizontal Traceability:** from origin to delivery

The need for **ownership** as a mechanism for traceability is also emphasised
Software Support

There are many commercial **CASE** (Computer Aided Software Engineering) and **CARE** (Computer Aided Requirements Engineering) tools available.

Before rushing out to purchase one or more of these we must ensure that we have established an effective requirements management process first. We also need to be aware of the fact that such tools can be expensive and that installation is a project in its own right, possibly needing consultancy support. The use of a tool will also require administration and training.

A CASE/CARE tool can provide:

- Documentation creation and storage
- Secure storage and access
- Documentation linkage
- Version control
- Modelling
- Even code generation!
## Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Subject</th>
<th>Prepared?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The majority of errors are committed at the analysis phase and great care has to be taken to ensure that the requirements are valid.</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>The purpose of a formal review is to catch errors as soon as possible, establish user buy-in and check adherence to standards.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Formal reviews take place with the author, moderator and peer reviewers.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Requirements Management ensures the tight control of requirements.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Requirements are baselined after validation.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>The Change Control Board is the final arbiter to changes to baselines.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CARE and CASE tools can be used to help the engineer manage the complexities of requirements in large projects.</td>
<td></td>
</tr>
</tbody>
</table>
Post Test

To reinforce the materials we have just covered, try out the following questions in your own time.

You'll find the answers on the next page.

1. What is the proper forum to validate requirements?
2. Formal reviews are often mis-used. Give two ways in which this happens?
3. Name one purpose of a formal review.
4. List the roles involved in a formal review.
5. List two activities of Requirements Management.
6. What is a baseline?
7. Who is the final arbiter of change to requirements?
8. What does the acronym CARE stand for?
9. Name three benefits of using a CASE/CARE tool.
Answers to Module 6

1. Formal Review.
2. Any two of:
   - An appraisal of the Requirements Engineer
   - A requirements elicitation or analysis meeting
   - A management review
   - A think tank
3. Any one of:
   - To catch errors as soon as possible
   - To establish user buy-in
   - To check adherence to standards
4. Requirements Engineer (the Author), Moderator, Reviewer.
5. Any two of:
   - Requirements identification
   - Requirements cross-referencing
   - Requirements origin and ownership
   - Software support
   - Change control
   - Configuration management
6. A Baseline is a product that has been formally reviewed and agreed upon. Thereafter it serves as the basis for further development and can only be changed through formal Change Control Procedures.
7. The Change Control Board (CCB).
9. Any three of:
   - Documentation creation and storage
   - Secure storage and access
   - Documentation linkage
   - Version control
   - Modelling
• Code generation
Further Reading

2. List of CARE tools with descriptions: http://easyweb.easynet.co.uk/~iany/other/vendors.htm

End of Module Notes
Examination Hints and Tips

Note that these notes are for guidance in approaching the written exam only.

- The exam is about applying requirements engineering techniques to a business scenario
- The exam lasts for 75 minutes in total: 15 minutes reading time to look at the scenario, questions and reference materials; 60 minutes to answer the questions
- The key to answering the questions is to apply the techniques and come up with specific answers based on the scenario
- No credit is earned for generic answers and answers copied from the course notes
- The paper is worth 50 marks and the pass mark is 25 (50%)

Reading Period (15 mins)

- Valid reading materials include the examination paper scenario and questions, and your course notes (including any notes you make during the course). Use this time wisely to understand the questions and the scenario
- Read the questions first and understand what you are being asked. Be aware of the number of marks available for each question
- Read the scenario to pick up ideas for answers on your first pass through
- Make use of your reference materials but note that no writing, marking, highlighting or annotating is allowed during this reading period
Writing Period (60 Mins)

- Manage your time effectively. Try to earn a “mark a minute”, i.e. spend 10 minutes answering a 10 mark question; this will allow 10 minutes at the end to check your answers
- Answer only what you have been asked for, nothing more. If you are asked for a specific number of examples, give that number (no more). If you are asked to explain your answer, give explanations; otherwise don’t waste time giving something that is not required
- Do not rewrite the question!
- For each of the techniques, look at the number of marks at stake and how many components requested to give you an idea of how much time to spend coming up with the answer
- Play the percentages – the pass mark is 50%. 100% correct answers for each question are almost impossible
- Write as neatly as you can – it makes marking easier, and the marker cannot give you marks if your answer is illegible
- It is “Open Book” – open your books and look at the examples, but don’t copy them

Case Study

The Case Study Scenario will be between one and one-and-a-half pages long and will include a general description of an organisation, its operations, a business need and an outline project to meet that need, as well as some requirements for change. These requirements, for an IT application, may be presented as several paragraphs of prose or as numbered lists.

When you refer to a requirement in your answers it may be enough to give the requirement’s number (if it has one).

However, if the requirement needs to be broken down, you will need to indicate which part of the requirement you are referring to; the best way of doing this is by quoting the relevant part of the requirement verbatim.
**Important Note**

Our markers regularly see papers in which candidates have lost marks (sometimes many marks, sometimes making the difference between pass and fail) through either failing to read the questions or failing to follow the instructions. **READ THE QUESTIONS CAREFULLY AND DO WHAT THEY ASK OF YOU.**

**Examinable Topics**

The exam is about the practical application of the techniques you have studied. Some topics in the syllabus don’t have techniques associated with them; they will eventually become relevant though if you go forward for the oral examination.

Below is a list of the examinable topics followed by details on how to approach a question on that technique.

- Requirements Stakeholders
  - Project Stakeholders
  - Business Stakeholders
  - External Stakeholders
- Knowledge Types
  - Tacit/Explicit (non-tacit)
  - Individual/Corporate
- Elicitation Techniques
  - Interviews
  - Workshops
  - Observation
  - Focus Groups
  - Prototyping
  - Scenarios
  - Document Analysis
  - Special Purpose Records
  - Questionnaires
- Hierarchy of Requirements
  - General Requirements (e.g. legal and business policy)
  - Functional Requirements
  - Non-Functional Requirements
Examination Hints and Tips

• Requirements Analysis Filters
  o Priority
  o Feasibility
  o Overlaps & Duplication
  o Conflicts
  o Ambiguity
  o Testability
• Models
  o Use Case Diagram
  o Interpreting a Class Diagram
• Use Case
  o Use Case Description Main Flow
• Requirements Catalogue
  o Sample Template
• Prioritising Requirements
  o MoSCoW
• Requirements Validation
  o Prototyping

Other topic areas will not be included in the written exam, but must be revised in preparation for the (closed book) diploma oral exam.

Here are exam hints on some of these topics.

Elicitation Techniques

• This always comes up
• You may be matching stakeholders to techniques or techniques to stakeholders
• The question is often lengthy – read it carefully
• You may be asked about a specific stakeholder
• Unless instructed otherwise, consider only stakeholders mentioned in the case study
• Avoid generic answers – look for specifics in the case study to justify your matching of stakeholder to elicitation technique
  o Justification is vital here
• You may be asked about the kind of information you hope to elicit – consider tacit/explicit knowledge and look for clues in the case study
• If you are asked to justify the use of the technique then be clear why this technique is useful for this stakeholder
• Look out for questions asking for different techniques or different stakeholders

FR/NFR/Solution/General

• There is always a question asking you to identify functional and non-functional requirements
• Sometimes you’ll be asked to spot solutions too
• You will often need to break the requirements down (quote when you need to)
• You’ll be asked to state the category of NFRs and you may be asked to explain what makes your FRs functional, in terms of CRUD
• This question may also ask you to apply some of the filters (see below)
• You could be requested to identify general requirements as well

Requirements Filters

• You may be asked to apply filters to the requirements, identifying a given number of ambiguities, solutions, etc.
• Justification is usually required
• You may be asked about involving stakeholders to resolve the issues – look for specific information from the case study
• Conflicting requirements may contradict one another, compromise one another or may be to do with data mismatch. You will be looking for a single conflict between two requirements

Use Case Diagram

• You may be asked to draw a Use Case Diagram to represent the functions demanded by the requirements or a subset thereof
• Make sure you name the use cases appropriately (verb-noun phrase)
• Use <<include>> only if functions are to be re-used
• Remember to draw the system boundary and name the system
Class Diagram

- You may be asked for a given number of errors or inconsistencies in the diagram or to check it against each requirement in turn.
- You're looking for where the class diagram fails to support the requirements, not the other way round.
- Your answer should be in the form of text – you do not need to re-draw the diagram (in fact, a re-drawn diagram will not be marked).
- Remember that you are being asked to check whether the class diagram supports the requirements so be clear in your answer and don’t sit on the fence!
- If the class diagram contains no attributes then stating an attribute is missing, is not a valid answer!

Use Case Description Main Flow

- You may be asked to document the primary scenario (happy path) of a Use Case.
- Show the interaction between user and system as a numbered list.
- Remember that this is the primary flow and assumes a successful outcome.
- The question may then ask you for a given number of alternates. You need only outline these – do not detail them as you did the primary flow.
- Think about what might happen or how the user may wish to deviate from the happy path.

Requirements Catalogue

- You might be asked to review a catalogue entry or to create an entry using specified fields.
- For reviewing an entry, your justification is vital.
- Make sure you consider how the fields relate to one another.
- A rationale may look fine but check that it matches the description, i.e. that it is the rationale for this requirement!
MoSCoW

- You cannot prioritise requirements until you understand the objective they are intended to meet. Sometimes the objective will be stated explicitly, sometimes you will have to deduce it from the information given. Once you have identified the objective, write it down for yourself so you don’t have to look for it again, even if the question doesn’t ask for this
- You’ll probably be asked for justification here and this is where the marks are
- Make sure your justification matches your level of priority

Prototyping

- As well as perhaps using prototyping for elicitation you may be asked about prototyping for validation of a particular requirement
- Draw on specifics from the case study when explaining why prototyping is appropriate
Index

Agile, 174
Ambiguity, 115
Analyst Problems, 49
BCS RE Syllabus, 5

**Business Case**

Definition of, 31

Business Objectives, 94

Business Requirements
   General, 98
   Technical, 99

CARE, 209
CASE, 209

Case Study Exercises
   Ex-1, 85
   Ex-2, 117
   Ex-3, 124
   Ex-4, 150
   Ex-5, 164
   Ex-6, 190

Class Diagrams, 151
   Associations, 156
   Attributes, 155
   Class, 153
   CRUD, 161
   Multiplicity, 157

Classroom Exercises
   At the Vet, 163
   Document Management
      System, 144, 149
   Ticket Sales System, 106

Conflicts, 114
Document Analysis, 76
Documentation Styles, 172
Documenting Findings, 57
Duplication, 113
Elicitation Techniques, 59

Eva, Malcolm, 103
Feasibility, 112
Focus Groups, 73
Formal Review, 198
Hierarchy of Requirements, 96
Interview Lifecycle, 60
Interviews, 60
MoSCoW, 109
NFR, 103
NGT, 68
Nominal Group Technique, 68
Observation, 71
   Types of, 71
Overlaps, 113

**PID**, 34
Priority, 109, 110

**Project Initiation Document**

Definition of, 34

Prototyping, 74, 120
Questionnaires, 78
Realism, 112

Requirements
   Ambiguous, 93
   Baseline, 206
   Change Control, 203, 212
   Change, Sources of, 206
   Configuration Management, 203, 212
   Correction Cost, 21
   Definition of, 14
   Document, 183
   Documentation styles, 172
   Good, 29
   Hierarchy, 96
   Management of Change to, 207
Origin, 203, 212
Origin of Errors, 20
Ownership, 203, 212
Prioritising, 109
Problems, 19
Purpose of Modelling, 132
Software Support, 203, 212
Solutions, 27
Specifications, 19
Stable, 206
Traceability, 208
Validation, 197
Volatile, 206
Requirements Engineering
Activities, 35
Stakeholders, 44
   Business Analysts, 46
   Customers, 47
   Developers, 46
   Project Manager, 46
   Project Sponsor, 45
   Regulators, 47
   Subject Matter Expert, 45
   Suppliers, 47
   System End-User, 45
Requirements Identification, 203, 212
Requirements Management
Components of, 202
Reviews
   After, 201
   During, 200
Objectives of, 198
RM, 202
Sample Requirement, 185
Scenarios, 75
SME, 45
Software Support, 209
Solution Requirements
   Non-functional, 103
Special Purpose Records, 77
Stakeholder
   Problems, 49
   Viewpoints, 50
Subject Matter Expert, 45
Terms of Reference
   Definition of, 32
ToR, 32
Traceability, 208
Unconscious Competence, 52
Use Cases, 135
   Description, 178
   Diagram, 142
   Diagrams, 135
   Extend association, 146
   Focus of, 141
   Include association, 145
   Naming conventions, 136
User Stories, 174
Validation, 197
Verification, 122
Workshops, 64
   Lifecycle, 67
XP, 174