Better Business Analysis

360° requirements analysis

A quick reference guide

360° requirements analysis

Every project is unique, with different emphasis and needs. When defining solution requirements, ensuring you're covering all aspects can be a real challenge. Even an experienced requirements engineer can sometimes find they can't see the wood for the trees! What does "complete" look like? Are we done yet? What might we be missing?

This quick reference guide explores different angles for approaching requirements definition, identifying sources to pursue and useful techniques to consider. Whether you're working on a formal Waterfall procurement project or the Agile development of a new product, you can use it as a prompt or starting point to:

- Start building a robust plan for requirements definition activities
- Identify potential elicitation and analysis techniques to deploy
- Check your requirements coverage for gaps and assumptions
- Inspire further thoughts, insights, and questions
- Consider ways to combine, communicate, and validate information to your stakeholders and consumers of the requirements

Not every area will always be equally relevant, so this guide can help you ensure you've considered what's appropriate and understand where your efforts will create the most value.

Business requirements

Before getting stuck into the detail, it’s vital to clarify what the organisation is expecting to achieve. Business requirements are statements of the key outcomes a solution should deliver.

- ❑ Automate manual steps
- ❑ Apply preventative controls to eliminate or reduce errors
- ❑ Reduce cycle time
- ❑ Compatibility with existing systems
- ❑ Tracking and monitoring of key performance indicators
- ❑ Compliance with regulatory framework

Business requirements may already have been articulated in project documentation, but are often implicit or assumed. Be sure to check the following sources:

Project documentation
Project request, PID, CONOPS, business case

Published strategy
Mission statements, value propositions, presentations, departmental objectives

Stakeholders
Sponsors, operational teams, users, customer journey mapping

Functionality and features

Identifying what a solution should actually do relies on understanding the business processes it will support, and how users will interact with it. There are a whole host of techniques available that you can apply to reveal the required functions and features of a solution. Here are some top approaches to try:

RACI

Understanding who does what will highlight key actor-system interactions, inputs, and potential reporting/oversight needs.

SIPOC

Clearly articulates how inputs are transformed into outputs while highlighting interactions with suppliers and customers.

Process Models

Decomposing a process into its basic steps makes identifying ***use cases*** *really straightforward.* ***BPMN*** *is ideal for detailed analysis, but lightweight approaches can be effective too!*

Personas

Different classes of user are likely to have different needs – creating personas to sense-check expectations and outcomes can uncover new scenarios or ***user stories****.*

Activity Diagrams

Activity diagrams delve into the explicit, structured interactions between users and systems. Great for confirming logic!

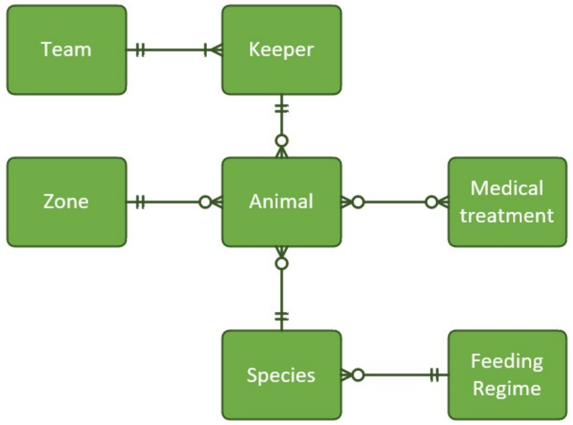
Prototyping

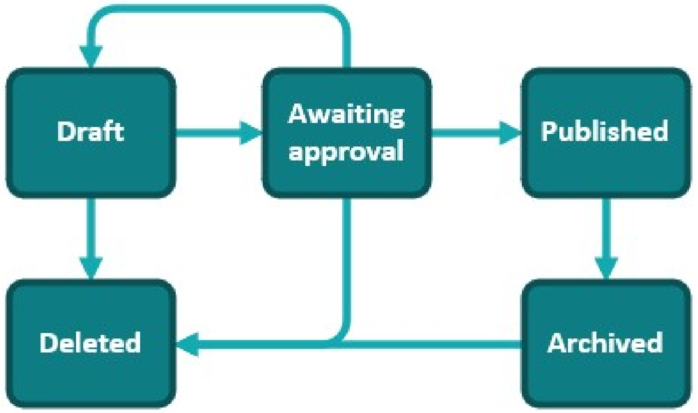
Working through mock-ups of interfaces with users can quickly reveal unexpected needs or nuanced expectations.

Interactions with the system can then be examined in more detail as **use cases** or **user stories**.

Information structures and states

Understanding the different entities involved(e.g. ‘customer’, ‘product’, ‘office’) is crucial. Exploring both the “static” structures and the “dynamic” changes to these entities will reveal many fundamental requirements.





Entity relationships

For each entity you uncover, consider the how it connects to other entities.

Entity relationship diagrams *and* ***UML class diagrams*** *are great ways to model and analyse these structures, but even simple* ***rich pictures*** *can be a handy way in to the topic!*

Entity lifecycles

Examine the various “states” an entity can be in, and how it can move between these states.

UML State Machine diagrams *are really useful for checking your understanding with stakeholders!*

Think about what attributes change with each transition.

Data

While you shouldn’t need to worry about database tables or data transfer protocols, but you should make sure you have a strong understanding of the nature of the data relevant to the problem domain.

Having identified the various conceptual entities involved (and the connections between them) you can explore their attributes to understand the data involved. Be sure to consider the following:

ENTITY ATTRIBUTES

What information belongs to each of the entities you’ve identified? Build a data dictionary to clarify things.

DATA OPERATIONS

*How is the data created? When is it updated? Can it be deleted? **CRUD** is a great way of exploring this!*

FORMAT

Is it a number, a string of text, or picked from a list? Is it calculated based on something else? What’s optional vs mandatory?

MASTER DATA

Make sure you’ve established how any master data is controlled and managed (and by whom).

Business rules

Business rules place constraints on how a process can work or a how a system can behave. There are different types of business rule to consider:

LEGAL

- Health and safety
- Data protection
- Company law
- Tax and reporting rules

POLICY

- Tolerances
- Approval limits
- Permissions / security
- Review / sign-off

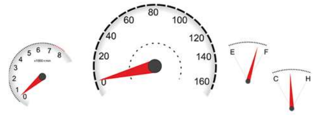
FACTS

- Information entity structures
- Real-world constraints (e.g. headcount, budget)

Artefacts and outputs

Explore what stakeholders are expecting the system to produce for them. Sometimes these will be obvious physical outputs or products, but it's also worth establishing what might be needed in the way of records and reports.

It's important not to simply translate reporting requests directly into requirements. Instead, explore needs around both form and content, and establish how and when data will be used for decision-making.

PRODUCTS	RECORDS	REPORTS	DASHBOARDS
Think about product specifications, format, and quality. How will they be moved or stored?	*What information needs to be stored? In what format? Where? How can they be accessed?*	*Who needs snapshots of information? What should they see? How should they receive it?*	*Who needs real-time information? Consider rate of change as well as current position.*
			

It's important not to simply translate reporting requests directly into requirements. Instead, explore needs around both form and content, and establish how and when data will be used for decision-making.

Consider whether users may actually need some form of "dashboard", and whether being able to drill down from aggregated information to individual records would be useful. Visibility of rate of change may be as important as current position (distance - velocity - acceleration - jerk).

Non-functional requirements

As well as establishing what the system must *do*, it’s crucial to determine the organisation’s wants and needs around *how* it does it. There are many different categories of non-functional requirements – and some will be more pertinent to your project than others. These are some of the more common types:

USABILITY

What's important about how users navigate, control, and interact with the system?

SERVICEABILITY

How do we want the system and user accounts to be supported, maintained, or updated?

PERFORMANCE

What is an acceptable level of performance (speed, wait times etc.)?

RECOVERABILITY

How long could we live without the system being available?
How much work can we afford to lose?

SECURITY

Who can access the system (or specific features)?
Where from?

DATA INTEGRITY

How must records be kept in sync across systems?
What happens if data changes elsewhere?

CAPACITY / THROUGHPUT

What does typical usage look like (for users, for the organisation)?
What about peak flow?
Future trends?

AVAILABILITY

When must the system always be available to users?
If it's not available, how should users know this?

ACCESSIBILITY

Do different groups of users have specific access needs (e.g. blind or partially-sighted users)?

RECORD RETENTION

How long should records be held?
Should they be automatically destroyed?

DEPLOYMENT

Does the system need to integrate with others?
Does it need to operate on different types of device (mobile, tablet etc.)?

TIMELINESS

Do changes to data need to be enacted instantly?
Must users always see real-time data?

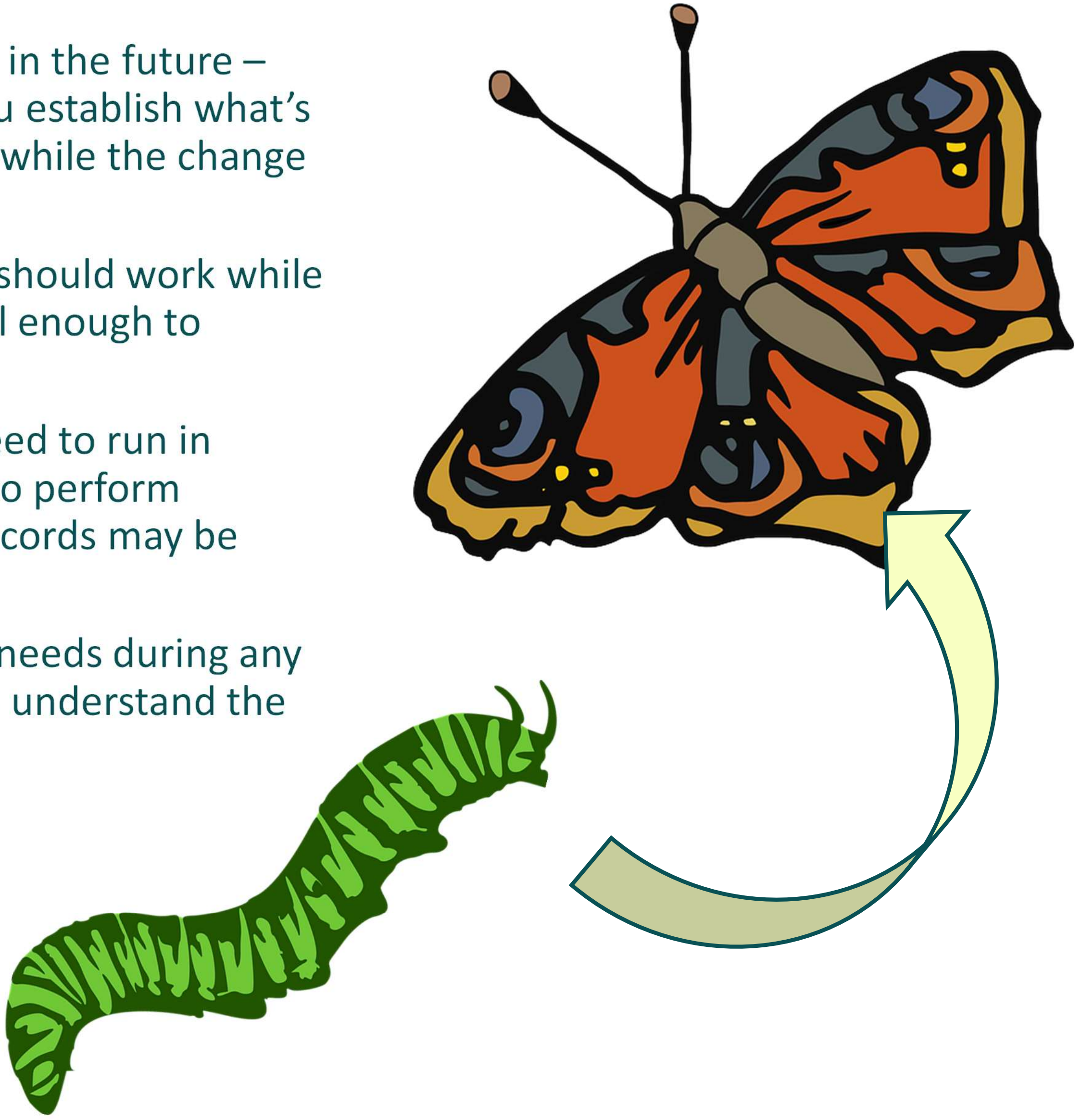
Transition requirements

It's all very well defining what a system will do in the future – getting there is another matter! Make sure you establish what's needed to avoid pain for users and customers while the change is happening.

Transition requirements deal with how things should work while a change takes place, and these may be critical enough to constrain solution options.

Consider whether old and new systems will need to run in parallel for a period, whether users will need to perform additional actions, or whether access to old records may be needed.

Talk to operational teams to understand their needs during any transition, and liaise with technology teams to understand the moving landscape.



What next?

Project, change, and technology development methodologies vary widely – Agile approaches focus on iterative delivery and adaptability, while Waterfall method focus on certainty and order. Whatever approach is applied in your project, you'll need to consider how the requirements will be agreed, managed, and respond to changing needs over time.

Think about the following activities and how they can best be tailored to the scale, complexity, and demands of your project:

- ❑ Documentation/cataloguing, publishing
- ❑ Stakeholder verification and validation or sign-off
- ❑ Prioritisation, strategic alignment, and impact/benefits mapping
- ❑ Change requests and approval
- ❑ Traceability management
- ❑ Further elaboration (e.g. development of acceptance criteria)
- ❑ Use in business acceptance activities

Also consider the needs of the people who will *use* the requirements – architects, developers, testers. What do they need from your work? Have you asked them? What *don't* they want?