The business analyst performs the Requirements Analysis tasks and techniques to analyze the different requirements he has gathered so far in order to define (or refine) whether these requirements meet the stakeholders’ needs with respect to the proposed solution.

It includes defining:

- **Stakeholder Requirements** - statements of the need of a stakeholder (or a stakeholder group) and how the stakeholder will interact with the solution.
- **Solution Requirements** - a characteristic of the solution that meets both the Business and Stakeholder Requirements. It’s typically divided into Functional and Non-functional Requirements.

**Also**, the business analyst will develop models of the current state of the organization (i.e., domain models) using a variety of techniques and diagramming tools like flowcharts or UML models. This is useful for validating the solution scope with business and/or technical stakeholders, for analyzing the current state of the organization in order to identify opportunities for improvement, or for assisting stakeholders in understanding that current state.
Upon completing this lesson, you should be able to:

- Describe the activities and techniques for prioritizing requirements.
- Define the key objectives for organizing requirements.
- Describe the activities, tools and techniques used for specifying and modeling requirements.
- Identify factors other than requirements, like assumptions and constraints, that can impact the solution.
- Describe the activities and techniques for verifying and validating requirements.
**Prioritize Requirements**

The *Prioritize Requirements* task describes the decision process activities performed by the business analyst to determine the relative importance of requirements based on their business value to the organization, inherent risk, implementation difficulty, etc.

These characteristics will determine whether a specific requirement needs further analysis or whether it should be implemented ahead of others. For example, in selecting a requirement with a high degree of project failure and implementing it first (high priority), the business analyst makes sure that if the project fails it does so after as little expenditure as possible by the organization. If the same requirement would have been implemented later, the cost of the project failure (or rework) would have been much higher!

Does your organization have a process for prioritizing requirements? What criteria do you use?
Basis for Prioritization

Requirements could be prioritized based on the following criteria:

Keep in mind that it can be a challenge to get stakeholders to prioritize requirements that they consider "non-negotiable."


**Requirements Risk**

Each activity in the requirements process bears some degree of risk. Examples of risk include:

- Inability to find stakeholders willing to express their user needs
- Stakeholders whose needs conflict with other stakeholders
- Stakeholders with changing priorities (leading to changing requirements)
- Inability to communicate requirements effectively with senior management stakeholders or with technical stakeholders

Beware! Any of these risk factors could lead to an ineffective set of requirements and ultimately to a project solution that doesn't meet the needs of the stakeholders.
Requirements Risk Evaluation

Ideally, business analysts should work with all stakeholders and end users to identify risks that may adversely impact the requirements gathering process. Once the risks are identified, the analyst must evaluate the likelihood of each one occurring and the degree of potential damage to the project that each risk could cause. This will help the analyst create a priority list of risks that require immediate attention.

For each risk, the business analyst might choose to avoid it altogether, apply some form of mitigation, or simply do nothing and hope for the best (using more proper terminology, this is called risk assumption).

For example, consider a project in which a leading software company (MacroHard Corporation) is developing a new word processing application scheduled for release at the same time that another company (Blue Sox Linux, Inc.) is releasing a new computer operating system bearing the same name. MacroHard Corp. is counting on releasing its new product on time in order to meet shareholders' financial expectations. Up to now, MacroHard's application developers have been working with beta versions of the Blue Sox Linux in developing the word processor.

What are MacroHard's requirements risks? One requirements risk is that the final version of the operating system may have important differences compared to the beta version, differences that render the word processor inoperable. So, there's a risk that the original requirements may change, which would mean that MacroHard developers would have to rewrite some code, thereby delaying the word processor's release.
**Addressing Requirements Risk**

So, what can the business analyst do to reduce or eliminate the possibility that a change in the operating system could cause a delay (and subsequent loss of revenue and market share) in the release of the word processor?

The solutions are limited by one's imagination. Here are just a few:

- Convince MacroHard's CEO to enter into an agreement with Blue Sox Linux to do some revenue sharing in exchange for an agreement that MacroHard's engineers will be kept up-to-date on any changes that may occur in Blue Sox Linux's technical specifications.
- Encourage MacroHard's engineers and developers to build as much flexibility as possible to any code modules that interact with the computer's operating system.
- Revise the risk assessment by surveying the market to see how many users are likely to begin using the newer version of Blue Sox Linux.
- (If the risk is low) take a chance that there won't be much difference between the final release of Blue Sox Linux and the beta version (i.e., assume the risk).
- Develop a second version of the word processor application that runs on a different operating system.
Risk Analysis

A technique used to evaluate risk is Risk Analysis. When assessing requirements risk, take the following into consideration:

- Likelihood it will occur (usually expressed as a probability)
- Impact on the requirements process if the risk becomes an issue
- Impact on project costs
- Impact on project schedule
- Impact on project scope
- Impact on end product quality
- Impact on project benefits
- Degree of difficulty in preventing the risk
- Mitigation strategy (avoid, mitigate, or assume)
- Action plan: who will put the mitigation strategy into effect
- Contingency plan: what actions to undertake if the risk materializes
- Materialization criteria: what are the criteria for declaring that a risk has materialized
1. Which of the following risks must a business analyst consider as potentially having an adverse impact on the requirements gathering process?
   a. ○ An unexpectedly high turnover rate among staff employees after a new technological solution is deployed.
   b. ○ The resignation of the Chief Technology Officer, who has been the only high-management person advocating the technology upgrade on which you are working.
   c. ○ Microsoft releasing a new operating system after the company's new human resources system is built on top of the "old" operating system.
   d. ○ Potential users of a new software upgrade "revolting" and refusing to use the newly redesigned system.
**Organize Your Requirements**

For requirements to be understood by all stakeholders, they need to be organized in a comprehensive, complete, and consistent way. This means that these requirements could be organized in a such a way that a technical stakeholder can understand what is needed to build the solution, or in another, less technical way, for end users of the system to understand the new capabilities that the solution will deliver.

When organizing your requirements, follow your organization’s standards in describing requirement types, use consistent definitions for each of the types of requirements and use the standard language and terminology prevalent in your organization. Refer to and use models and other documents to support your efforts.

There are two key goals to keep in mind when organizing your requirements:

1. Understand what models are appropriate for describing the business domain and/or the solution scope
2. Identify the interrelationships and dependencies among requirements
Ways to Organize Requirements

You can organize requirements at different levels of abstraction. For example:

- "What"/"How" requirements - typically defined as describing what needs to be done and not how to do it. However, depending on the audience, a "what" or a "how" designation can become problematic. For example, a business process improvement project can define "how" the process is going to be more efficient.

- High-level/Low-level requirements - the level chosen depends on the level of detail included in the description

- Methodology - may also determine the level of abstraction used when defining requirements.
Model Selection

Model selection is the organization of requirements into a cohesive Requirements Structure that is supported by a set of models and other documents that describe the solution scope. The business analyst must determine which types of models will be required to best describe this. She must also make it understood to the project stakeholders. By choosing a particular model, the BA is presenting a different view of the business domain.

It is usually necessary to develop multiple models using different modeling techniques to completely analyze and document requirements.

Modeling techniques are described in more detail in the Specify and Model Requirements (6.3) task.
General Modeling Concepts

Models can be categorized by what aspect of the domain model they describe.

Please refer to pages 105-106 in the BABOK® to see the different techniques associated with these models. In the next pages, we'll look at Business Rules in more detail.
Business Rules Examples

Business rules, quite simply, describe how business is to be conducted. A well-crafted business rule has an identifier (a "name") and a textual component that unambiguously describes how a particular business function or activity is to be carried out. Business rules exist independently of the means by which they are implemented. Here are a few examples:
**Specify and Model Requirements**

The BA needs to support the Requirements Structure with formal artifacts such as text statements, matrices, and models.

**Text** is used to describe the capabilities of the solution, specifying any pre-conditions to use the system, as well as any constraints.

**Matrix Documentation** (tabular format) describes relationships in a row/column structure.

**Models** are abstractions of the domain presented in a way that is useful for understanding a specific view of the domain reality.

**Diagrams** are a particular representation of certain model types.
**Guidelines for Specifying Requirements**

Let's say we want to specify functional requirements. As you know, these requirements address how the solution affects or impacts the organization in which it is deployed, how people or external systems interact with the solution, and how that solution complies with any applicable standards. Following are several guidelines for specifying functional requirements.

A well-formed requirement has the following components:

- An **event or condition** that specifies when the requirement must be fulfilled.
- A **subject** (person or system) that performs an action to fulfill the requirement.
- An **action verb** that specifies what the subject does.
- An **object** that is acted upon or otherwise is involved in fulfilling the requirement.
- A **rule** (or set of rules) that affects or modifies the outcome (see below).
- An **outcome**, which consists of a desired result (and the criteria for success that accompany it).

If a requirement is written in prose, it should:

- Be stated simply.
- Use consistent terminology throughout.
- Be expressed as a verb or verb phrase.
- Have only one requirement per sentence.
- Use active voice with clear actors, actions, and subjects.
Specifying Requirements

Here is an excerpt from a list of requirements...

**Requirement 31:** If the user is already logged on, display the message "User Already Logged On."

**Requirement 32:** After the message is displayed, return control to the home page.

We can tell that these are well-formed requirements because they have the following components:

**Event/Condition:** The user clicks "OK" on the login window.

**Subject:** The system responds to the clicking of the "OK" button.

**Action Verb:** Display (the message "User Already...")

**Object:** Logged-in status

**Rules:** Users are not allowed to be logged into the system more than once.

**Outcome:** If the user is already logged in and tries to log in a second time, he will see the warning message appear and be prompted to click "OK," after which the website's home page loads.
Diagrams

**Diagrams** are especially useful for clarifying text and for demonstrating visually the relationships between different components and objects. Diagrams are also good at showing the sequence of events (useful for identifying parallel activities) and the physical arrangement of objects in a solution.

In general, we want our diagrams to be as simple as possible but still communicate essential information.

In some situations, analysts can use diagrams to represent **models**. The complete requirements document, which may contain text, diagrams, and even a matrix of requirements, is called a **model**. This is perhaps the most abstract form of documentation for a solution and does not usually need to address every single aspect of that solution. In fact, this is one of the benefits of a model: you can focus on the most important issues by filtering out distractions.

Models can be **logical** or **physical**. A logical model describes a system (a solution) in terms of tangible elements that are familiar to users. Business analysts usually develop the logical model for a solution and pass it along to technical developers who use it to create a physical model. A physical model corresponds to actual artifacts that developers assemble in order to implement the logical model.

For more information on data flow diagrams, refer to section 9.6 in the BABOK®.

See an example of a [data flow diagram](https://www.nasa.gov/) from NASA.
Example of Logical vs. Physical Models

Consider a database, which consists of several tables of data. For instance, a human resources database might have a table of employee data, a table of job titles and descriptions, a table of salary grades and corresponding salaries, and other appropriate data. The data architect has the job of deciding how to set up the tables and what data should go into each table.
Modeling Techniques

The BABOK® lists several techniques associated with specifying and modeling requirements:

- Acceptance and Evaluation Criteria Definition (9.1)
- Business Rules Analysis (9.4)
- Data Dictionary and Glossary (9.5)
- Data Flow Diagrams (9.6)
- Data Modeling (9.7)
- Functional Decomposition (9.12)
- Interface Analysis (9.13)
- Metrics and Key Performance Indicators (9.16)
- Non-functional Requirements Analysis (9.17)
- Organization Modeling (9.19)
- Process Modeling (9.21)
- Prototyping (9.22)
- Scenarios and Use Cases (9.26)
- Sequence Diagrams (9.28)
- State Diagrams (9.29)
- User Stories (9.33)
Use Cases

A *use case* description is a text-based description of how a user (or other external entity) interacts with a proposed solution. It is a narrative description of the steps the user takes and the responses of the system.

Each use case (corresponding to each process in the system) has a unique identifying name. The users of the solution are called *actors* in that they act upon the system. Each role has a unique actor and is assigned a unique name. The concept of a use case is best illustrated with a simple example:

**Example: Web-Based Money Transfer Process**

*This use case describes how a bank customer can access his/her accounts online and conduct a transfer of funds from one account to another.*

The user must be logged into his or her account before proceeding. The first step is to select "transfer" from the menu bar options. Next, she selects the "from" account (checking or savings) and "to" account (checking or savings). If she tries to transfer money from an account into itself, a warning window pops up stating that the transaction is not allowed. The user clicks "ok" on the warning window and must again specify "from" and "to" accounts.

If there is no warning and there are sufficient funds in the "from" account, the application displays "transaction successful," and the user is returned to the home page. If there are not enough funds in the "from" account, the system displays the account's balance, the message "Sorry, you have insufficient funds to cover this request. Please select a lower amount to transfer" and allows the user to re-enter the amount to be transferred.
**User Stories**

A *user story* resembles a use case in that it is a textual description of the required business functions that a solution needs to address. However, user stories are created by the users themselves rather than a business analyst or applications developer.

User stories are necessarily high-level in scope and reflect only the users' needs without delving into any specific solution strategy. The project team can use these stories to validate requirements gathered from multiple stakeholders and to elicit agreement on project concepts among those stakeholders.

A complex system with many features may have tens or hundreds of user story cards. In addition to the description, a story card needs to provide information about the resources and time needed to develop a solution. In some situations, designers may need detailed user stories to supplement other requirements documents. Additional information that may be useful on user story cards includes constraints and specific business rules.
Non-functional Requirements Analysis

Non-functional Requirements (sometimes called "Quality of Service" or system requirements") address the context within which a solution must function or operate. They do not affect the design of a solution directly, but because they could potentially impose constraints indirectly, they need to be analyzed together with direct requirements. Let's look at several examples of quality of service requirements:

Environmental
These requirements reflect constraints imposed by anything outside the system boundary. They include political, regulatory, standardization, cultural, and globalization requirements as well as physical constraints (size, temperature range, humidity range, etc.).

Interface
Interface requirements address the connection or interaction between the solution system and the outside world. They may focus on functional aspects such as how one computer transfers data to another or they might focus on human aspects such as the ease with which a person can interact with and use the system.

Glossary
A glossary defines special terminology that has unique meaning to the organization. It helps everyone working on the project along with stakeholders understand and communicate with each other more precisely.
More on Non-functional Requirements Analysis

Operational
While interface requirements are usually unchanging, like the overall architecture of a system, operational requirements are dynamic. They address how the system works and how it is used. Operational requirements also include constraints such as the largest number of users that can log on to the system simultaneously or environmental factors such as the specific network operating platform on which the application must run.

Performance
Performance requirements focus on how the system functions. Microprocessor speed, data access time, and memory capacity are typical performance requirements. At a higher level of abstraction, database access time may define yet another performance requirement.

Privacy
Privacy constraints address the security of data stored in a system. Personal data such as social security numbers must be guarded carefully and even legitimate users need to be kept "under control."
More on Non-functional Requirements Analysis

Quality
Quality requirements address a solution's:

- Reliability - does it work when you need it?
- Usability - can everyone who needs to use it actually use it?
- Maintainability - is it easy to fix or upgrade?
- Scalability - can it handle more users or more activity without reprogramming?
- Portability - does it work on different operating systems or in different environments?

Quality requirements also address how gracefully the solution fails (for instance, without losing any data) and how quickly one can recover the system's functionality following a disaster.

Safety
This requirement focuses on possible loss, damage, or other type of harm that could occur as a result of a solution's operation. Safety issues range from ensuring that power cords are grounded to bolting large pieces of equipment to the wall to prevent tipping in case of an earthquake.
More on Non-functional Requirements Analysis

**Security**
Security requirements are designed to protect data or information created or processed by the system. It includes virtual access to information, such as by a hacker, to physical access to the machines on which the data are stored.

**Training**
This is the last requirement but far from the least important one! Without proper training, users will not know how to use a new solution and may miss out on useful but not obvious features.
Assumptions

An assumption is something the business analyst believes is true but has not independently verified, or something that is known to be true but could adversely impact the viability of a solution if it were to change. Assumptions usually have to do with environmental factors that affect what the project team can implement as a solution and how they can do it.

Here are several examples of assumptions:

- The operating system running the computers or network on which a solution is to be installed
- The location of users of a solution (in the office or out in the field, for instance)
- The costs of labor and material resources
- The language in which the user interface should be written
**Constraints**

A *constraint* limits the options available for the development of a solution. We have *business constraints*, which are related to how a solution may be developed, and *technical constraints*, which address physical and architectural limitations on the solution.

*See if you can match the kind of constraint to its right category:*
Requirements Verification and Validation

Verification and validation are related so we'll look at them together. Technically, validation focuses on the relationship between a set of requirements and the business model they represent. Verification focuses on whether those requirements are sufficiently detailed and thorough to allow the development of a solution without needing additional clarification. Validation asks the question, "Do the requirements adequately describe the business model they are supposed to represent?" Verification asks the question, "Do I have all the information I need to build the solution?"

Of course, the design and development process is not conducted on a remote island! There will and should be opportunities for interactions among developers, business analysts, and stakeholders along the way to ensure that the project remains focused on the correct business objectives.

A set of requirements are said to be valid when they completely and thoroughly capture all stakeholder needs that were expressed during the requirements elicitation process. The business analyst needs to review the requirements from this perspective (verification) before moving on to the next step, which is determining if the requirements provide enough detail for development to begin (validation). Just because the business requirements are complete doesn't mean that the technical specifications are also complete.
**Requirements Quality**

The business analyst needs to confer with the technical developers (and possibly technically-minded stakeholders) to evaluate the quality of the requirements.

High-quality requirements are:

- Cohesive
- Complete
- Consistent
- Correct
- Feasible
- Modifiable
- Unambiguous
- Testable

Review pages 115-116 of the BABOK® for a detailed description of these quality characteristics.
Requirements Verification/Validation Example

*Think about how you might improve the bad requirements below; once you have an idea, roll your mouse over the bolded items to reveal a better example.*
1. Decide whether these statements are true or false.

a. User stories are created by the users themselves rather than a business analyst.
   - True
   - False

b. Environmental requirements are constraints from outside the system.
   - True
   - False

c. Portability refers to a system's ability to be easily fixed or upgraded.
   - True
   - False

d. A good business analyst will never have assumptions.
   - True
   - False

e. Validation asks the question, "Do the requirements adequately describe the business model they are supposed to represent?" while verification asks the question, "Do I have all the information I need to build the solution?"
   - True
   - False

f. One of the benefits of a model is that you can focus on the most important issues by filtering out distractions.
   - True
   - False

g. A risk mitigation strategy has three choices: avoid, mitigate, or delegate.
   - True
   - False

h. Levels of abstraction refer to the different ways that business analysts can describe requirements.
   - True
   - False
2. Microprocessor speed, data access time, and memory capacity are typical:
   a. ○ performance requirements
   b. ○ privacy constraints
   c. ○ operational requirements
   d. ○ interface constraints
3. **Scalability, usability, and portability are examples of:**
   a. ☐ reliability barometers
   b. ☐ technical solutions
   c. ☐ quality requirements
   d. ☐ general modeling concepts
The *Requirements Analysis* Knowledge Area covers the tasks and techniques used by a business analyst to analyze the stated requirements in order to ensure that the proposed solution will fulfill stakeholder needs.

This lesson’s key topics stressed the need to:

- Prioritize requirements
- Organize requirements
- Specify and model requirements
- Define assumptions and constraints
- Verify and validate requirements