**Requirement Definition**

**Per** BABOK **guide, official definition of requirement is:**

1. “A condition or capability needed by a stakeholder to solve a problem or achieve an objective.” In simpler words, a decision-making process to derive requirements from needs.

2. “A condition or capability that must be met or possessed by a solution or solution component to satisfy a contract, standard, specification or other formally imposed documents.”It is a step where business requirements are drafted as solutions requirements to get started with developing the solution.

3. “A documented representation of a condition or capability as in 1 or 2.” The documentation is itself a requirement as it helps all the stakeholders and consumers in understanding the requirements for the solution.

# Where do these requirements come from?

All the requirements arise from a need. We need to understand those Business Needs or the Business Problem Statement. Unless there is a problem, we can’t think of providing a solution. A lot of analysis and research is carried on before requirement analysis to understand the problem. These involve feasibility study, knowing business terms and concepts, cost/benefit analysis, business strategy etc.

A Business Analyst (BA) starts with a broad and general description of what needs to be done and then starts working with key stakeholders to define the project scope (inclusions and exclusions), high-level business requirements, solution requirements, technical requirements and transition requirements primarily.

Talking about stakeholders, it’s important to know about who they are and how critical is their involvement in the project.

## Who are stakeholders?

A stakeholder is a generic term for a person or group of people who are involved with the project (directly or indirectly) and have a say in decision making process of the project. They can be:

* Executive Sponsor — Mostly concerned about funding of the project and high-level information
* Product Manager — Make important decisions for the project and review & approve requirements
* Project Manager — Prepares project plan, resource allocation plan, manages the execution of the project and works very closely with the BA (Business Analyst
* Subject Matter Experts — Assists in defining project scope, works with the BA to define business rules, processes and user interface

There is a whole set of other roles such as Technical Personnel, Technical Writers, Quality Assurance Personnel, Database Administrators and others who can be important stakeholders in a project, depending upon the needs.

## How stakeholders help with requirements?

Since each stakeholder plays a different role in the project, they have different requirements too. Sometimes, what emerges as the biggest challenge for a BA is to extract useful information from all these stakeholders that matches the preferences best with the client’s business. This is because each of them view the project from their own perspective!

To create best requirements for a project, identifying responsible stakeholders is important. There are many techniques available such as RACI matrix. Not all stakeholders are important for every project. It is primary task to identify the right stakeholders and their say in the decisions to keep things smooth in long run. (When all speak, it becomes real hard to reach to any conclusion).

# How are requirements identified?

“The most difficult part of requirements gathering is not the act of recording what the user wants, it is the exploratory development activity of helping users figure out what they want.” — Steve McConnell

The process to identify requirements from the needs or conditions is termed as Requirements Analysis. A detailed analysis is critical to solution’s success or failure. It involves the tasks as analyzing, documenting, defining, validating, documenting and managing the changing requirements. There are different standards set in different organizations for the requirement analysis though the primary steps could be identified as:

* Identify stakeholders
* Requirements Elicitation — capture stakeholder requirements through various techniques such as interviews, questionnaire, joint group discussions, prototypes or use cases
* Identify requirement categories — categorize all the gathered requirements into functional, non-functional, business, technical or transitional requirements
* Analyze requirements — analyze the requirements whether they are clear, complete, unambiguous, consistent and testable
* Requirements documentation — requirements are documented in various forms such as Business Requirements Document (BRD) to describe business requirements, Software Requirements Specification (SRS) to describe functional and non-functional requirements, Use cases and User stories (in agile context)

These recorded requirements documents are then collaborated upon to receive feedback from all involved stakeholders. Once the requirements match the needs for the project, it is important to take sign-off to freeze the scope of work and avoid frequent scope creeps at later stages.

**Type Of Requirements**

Business requirements. These include high-level statements of goals, objectives, and needs.

Stakeholder requirements. The needs of discrete stakeholder groups are also specified to define what they expect from a particular solution.

Solution requirements. Solution requirements describe the characteristics that a product must have to meet the needs of the stakeholders and the business itself.

* Nonfunctional requirements describe the general characteristics of a system. They are also known as quality attributes.
* Functional requirements describe how a product must behave, what its features and functions.

Transition requirements. An additional group of requirements defines what is needed from an organization to successfully move from its current state to its desired state with the new product.

Let’s explore functional and nonfunctional requirements in greater detail.

## Functional requirements and their specifications

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions. For instance:

A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.

Here’s another simple example: As a guest, I want a sofa that I can sleep on overnight.

Requirements are usually written in text, especially for Agile-driven projects. However, they may also be visuals. Here are the most common formats and documents:

* Software requirements specification document
* Use cases
* User stories
* Work Breakdown Structure (WBS) (functional decomposition)
* Prototypes
* Models and diagrams

### Software requirements specification document

Functional and nonfunctional requirements can be formalized in the requirements specification (SRS) document. (To learn more about software documentation, read our article on that topic.) The SRS contains descriptions of functions and capabilities that the product must provide. The document also defines constraints and assumptions. The SRS can be a single document communicating functional requirements or it may accompany other software documentation like user stories and use cases.

We don’t recommend composing SRS for the entire solution before the development kick-off, but you should document the requirements for every single feature before actually building it. Once you receive the initial user feedback, you can update the document.

SRS must include the following sections:

Purpose. Definitions, system overview, and background.

Overall description. Assumptions, constraints, business rules, and product vision.

Specific requirements. System attributes, functional requirements, database requirements.

It’s essential to make the SRS readable for all stakeholders. You also should use templates with visual emphasis to structure the information and aid in understanding it. If you have requirements stored in some other document formats, link to them to allow readers to find the needed information.

Example: If you’d like to see an actual document, download this [SRS example](https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc) created at Michigan State University, which includes all points mentioned above in addition to presenting use cases to illustrate parts of the product.

### Use cases

Use cases describe the interaction between the system and external users that leads to achieving particular goals.

Each use case includes three main elements:

Actors. These are the users outside the system that interact with the system.

System. The system is described by functional requirements that define an intended behavior of the product.

Goals. The purposes of the interaction between the users and the system are outlined as goals.

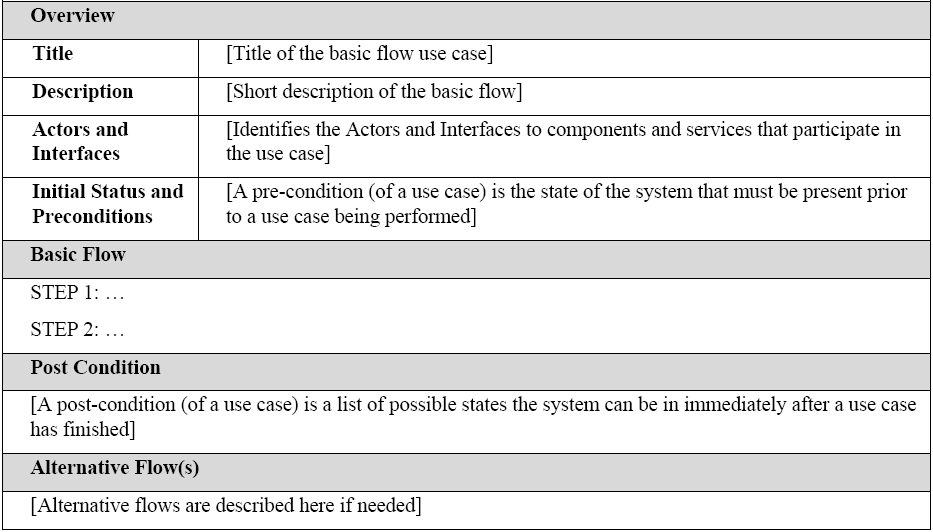
There are two formats to represent use cases:

* Use case specification structured in textual format
* Use case diagram

A use case specification represents the sequence of events along with other information that relates to this use case. A typical use case specification template includes the following information:

* Description
* Pre- and Post- interaction condition
* Basic interaction path
* Alternative path
* Exception path

Example:



Use case specification template

A use case diagram doesn’t contain a lot of details. It shows a high-level overview of the relationships between actors, different use cases, and the system.

The use case diagram includes the following main elements:

Use cases. Usually drawn with ovals, use cases represent different use scenarios that actors might have with the system (log in, make a purchase, view items, etc.)

System boundaries.  Boundaries are outlined by the box that groups various use cases in a system.

Actors. These are the figures that depict external users (people or systems) that interact with the system.

Associations. Associations are drawn with lines showing different types of relationships between actors and use cases.

Example:



Use case diagram example

### User stories

A user story is a documented description of a software feature seen from the end-user perspective. The user story describes what exactly the user wants the system to do. In Agile projects, user stories are organized in a backlog, which is an ordered list of product functions. Currently, user stories are considered to be the best format for backlog items.

A typical user story is written like this:

As a <type of user>, I want <some goal> so that <some reason>.

Example:

As an admin, I want to add descriptions to products so that users can later view these descriptions and compare the products.

User stories must be accompanied by acceptance criteria. These are the conditions that the product must satisfy to be accepted by a user, stakeholders, or a product owner. Each user story must have at least one acceptance criterion. Effective acceptance criteria must be testable, concise, and completely understood by all team members and stakeholders. They can be written as checklists, plain text, or by using Given/When/Then format.

Example:

Here’s an example of the acceptance criteria checklist for a user story describing a search feature:

* A search field is available on the top-bar.
* A search is started when the user clicks Submit.
* The default placeholder is a grey text Type the name.
* The placeholder disappears when the user starts typing.
* The search language is English.
* The user can type no more than 200 symbols.
* It doesn’t support special symbols.  If the user has typed a special symbol in the search input, it displays the warning massage: Search input cannot contain special symbols.

Finally, all user stories must fit the INVEST quality model:

* I – Independent
* N – Negotiable
* V – Valuable
* E – Estimable
* S – Small
* T – Testable

Independent. This means that you can schedule and implement each user story separately. This is very helpful if you implement continuous integration processes.

Negotiable. This means that all parties agree to prioritize negotiations over specification. This also means that details will be created constantly during development.

Valuable. A story must be valuable to the customer.  You should ask yourself from the customer’s perspective “why” you need to implement a given feature.

Estimatable. A quality user story can be estimated. This will help a team schedule and prioritize the implementation. The bigger the story is, the harder it is to estimate it.

Small. Good user stories tend to be small enough to plan for short production releases. Small stories allow for more specific estimates.

Testable. If a story can be tested, it’s clear enough and good enough. Tested stories mean that requirements are done and ready for use.

### Functional decomposition or Work Breakdown Structures (WBS)

A functional decomposition or WBS is a visual document that illustrates how complex processes break down into their simpler components. WBS is an effective approach to allow for an independent analysis of each part. WBS also helps capture the full picture of the project.

We suggest the following logic of functional decomposition:

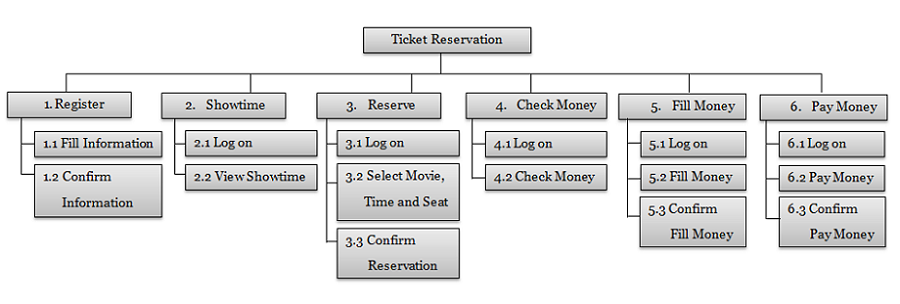
1. Find the most general function.
2. Find the closest sub function.
3. Find the next level of sub function.
4. Check your diagram.

Or the decomposition process may look like this:

High Level Function ->Sub-function -> Process -> Activity

The features should be decomposed to the point at which the lowest level parts can’t be broken down any further.

Example:



      An example of a functional decomposition

### Software prototypes

Software prototype is an umbrella term for different forms of early stage deliverables that are built to showcase how requirements must be implemented. Prototypes help bridge the vision gaps and let stakeholders and teams clarify complicated areas of products in development. Traditionally, prototypes represent how the solution will work and give examples of how users will interact with it to accomplish their tasks.

Prototypes can be cheap and fast visual representations of requirements (throwaway prototypes) or more complex ones (evolutionary prototypes). The latter can even become the early versions of the product that already have some pieces of the final code. Effectively, evolutionary prototypes may even turn into [MVPs](https://www.altexsoft.com/blog/business/minimum-viable-product-types-methods-and-building-stages/) that we’ve described in a separate article.

### Design documents and prototypes

Design requirements are usually collected and documented using three main formats that morph into one another:

Wireframes. Wireframes are low-fidelity graphic structures of a website or an app. They help map different product pages with sections and interactive elements.

Mockups. Once wireframes are ready, they are turned into mockups, visual designs that convey the look and feel of the final product. Eventually, mockups can become the final design of the product.

Design prototypes. These documents contain visuals and allow for some interface interactions, like scrolling, clicking on links, or filling in forms. Design prototypes can be built from scratch using HTML and CSS, but most UX teams use prototyping services like InVision.

Example:

To learn more about how UX design processes are handled, check our case study about building a [travel management solution for Cornerstone](https://www.altexsoft.com/case-studies/travel/altexsoft-cornerstone-reimagining-the-user-experience-for-a-corporate-travel-management-solution/), a corporate SaaS provider, in which we used all three types of design requirements.

## Nonfunctional requirements

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system’s quality attributes.

Let’s have a close look at typical nonfunctional requirements.

### Usability

Usability defines how difficult it will be for a user to learn and operate the system. Usability can be assessed from different points of view:

Efficiency of use: the average time it takes to accomplish a user’s goals, how many tasks a user can complete without any help, the number of transactions completed without errors, etc.

Intuitiveness: how simple it is to understand the interface, buttons, headings, etc.

Low perceived workload: how many attempts are needed by users to accomplish a particular task.

Example: Usability requirements can consider language barriers and localization tasks: People with no understanding of French must be able to use the product. Or you may set [accessibility requirements](https://www.altexsoft.com/blog/uxdesign/reach-your-audience-with-accessible-and-inclusive-design/): Keyboard users who navigate a website using <tab>, must be able to reach the “Add to cart” button from a product page within 15 <tab> clicks.

### Security

Security requirements ensure that the software is protected from unauthorized access to the system and its stored data. It considers different levels of authorization and authentication across different users roles. For instance, data privacy is a security characteristic that describes who can create, see, copy, change, or delete information. Security also includes protection against viruses and malware attacks.

Example: Access permissions for the particular system information may only be changed by the system’s data administrator.

### Reliability

Reliability defines how likely it is for the software to work without failure for a given period of time. Reliability decreases because of bugs in the code, hardware failures, or problems with other system components. To measure software reliability, you can count the percentage of operations that are completed correctly or track the average period of time the system runs before failing.

Example: The database update process must roll back all related updates when any update fails.

### Performance

Performance is a quality attribute that describes the responsiveness of the system to various user interactions with it. Poor performance leads to negative user experience. It also jeopardizes system safety when it’s is overloaded.

Example: The front-page load time must be no more that 2 seconds for users that access the website using an LTE mobile connection.

### Availability

Availability is gauged by the period of time that the system’s functionality and services are available for use with all operations. So, scheduled maintenance periods directly influence this parameter. And it’s important to define how the impact of maintenance can be minimized. When writing the availability requirements, the team has to define the most critical components of the system that must be available at all time. You should also prepare user notifications in case the system or one of its parts becomes unavailable.

Example: New module deployment musn’t impact front page, product pages, and check out pages availability and mustn’t take longer than one hour. The rest of the pages that may experience problems must display a notification with a timer showing when the system is going to be up again.

Credit/sources:--concerned authority on WWW