**BUSINESS ANALYST**

* **What is Business Analysis?**

Business Analysis is the set of tasks, knowledge, and techniques required to identify business needs and determine solutions to enterprise business problems. In Information technology industry, solutions often include a systems development component, but may also consist of process improvement or organizational change. Business analysis may also be performed to understand the current state of an organization or to serve as a basis for the identification of business needs. In most cases, however, business analysis is performed to define and validate solutions that meets business needs, goals, or objectives.

* **Who is a Business Analyst?**

A business analyst is someone who analyses an organization or business domain (real or hypothetical) and documents its business, processes, or systems, assessing the business model or its integration with technology. However, organizational titles vary such as analyst, business analyst, business systems analyst or maybe systems analyst.

* **Why a Business Analyst?**

Organizations employ business analysis for the following reasons:

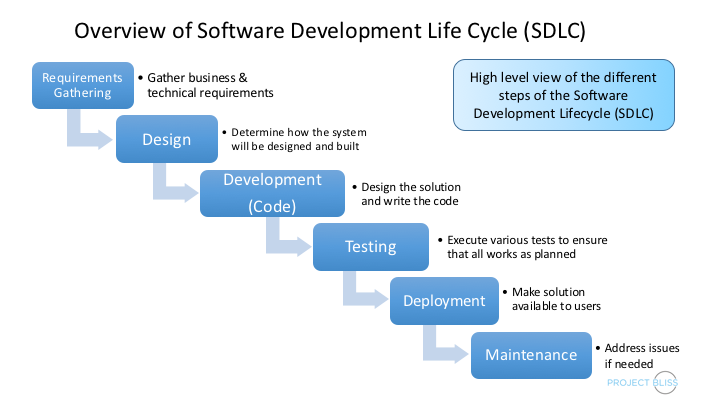
* To understand the structure and the dynamics of the organization in which a system is to be deployed.
* To understand current problems in the target organization and identify improvement potentials.
* To ensure that the customer, end user, and developers have a common understanding of the target organization. In the initial phase of a project, when the requirements are being interpreted by the solution and design teams, the role of a Business analyst is to review the solutions documents, work closely with the solutions designers (IT team) and Project managers to ensure that requirements are clear. In a typical large-size IT organization, especially in a development environment, you can find On-site as well as offshore delivery teams having the above-mentioned roles. You can find a “Business Analyst” who acts as a key person who has to link both the teams.

**Hence, the role of BA is very crucial in the effective and successful jumpstart for any project.**

* **Software Development Life Cycle (SDLC)**

Software Development Life Cycle (SDLC) is a process followed in a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. It defines a methodology for improving the quality of software and the overall development process.

* SDLC is a process used by IT analysts in order to develop or redesign high quality software system, which meets both the customer and the real-world requirement.
* It takes into consideration all the associated aspects of software testing, analysis and post-process maintenance.



* **Planning Stage**

Every activity must start with a plan. Failing to plan is planning to fail. The degree of planning differs from one model to another, but it's very important to have a clear understanding of what we are going to build by creating the system's specifications.

* **Defining Stage**

In this phase, we analyze and define the system's structure. We define the architecture, the components, and how these components fit together to produce a working system.

* **Designing Stage**

In system design, the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

* **Building Stage**

This is the development phase. We start code generation based on the system's design using compilers, interpreters, debuggers to bring the system to life.

* **Implementation**

Implementation is a part of the Building Stage. In this phase, we start code generation based on the system's design using compilers, interpreters, debuggers to bring the system to life.

* **Testing Stage**

As different parts of the system are completed; they are put through a series of tests. it is tested against the requirements to make sure that the product is actually solving the needs addressed during the requirement phase.

* Test plans and test cases are used to identify bugs and to ensure that the system is working according to the specifications.
* In this phase, different types of testing like unit testing, manual testing, acceptance testing and system testing is done.

**Post SDLC Process**

After the product is released in the market, its maintenance is done for the existing customer base. Once in the production environment, the system will suffer modifications because of undetected bugs or other unexpected events. The system is evaluated and the cycle is repeated for maintaining the system.

* **JIRA BOARD**

A Jira board is a visual representation of a team's work in Jira, a project management tool. It helps teams track and discuss the progress of a project.

What does a Jira board show?

* A Jira board displays tasks and their status in columns.
* The board can show up to 5,000 issues at a time.
* The default status columns are To Do, In Progress, and Done.

What can a Jira board be used for? Reviewing and manipulating individual issues, Passing issues from one stage to another, and Defining and configuring workflows.

**How to create a Jira board?**

1. Click Search then View all boards
2. Click Create board
3. Select a board type, such as scrum or kanban
4. Choose to start with a new project template or add the board to an existing project

**What is Jira used for?**

* Planning, tracking, releasing, and supporting software
* Providing context for teams to move quickly while staying connected to business goals.

**What are two type of board in Jira?**

The two board types are:

|  |  |
| --- | --- |
| * [**Scrum boards**](https://www.atlassian.com/software/jira/features/scrum-boards) | * Manage stories, tasks, and, and bugs in **sprints** * Suits teams that deliver work on a regular schedule |
| * **Kanban boards** | * Manage stories, tasks, and bugs in a **continuous flow** * Suits teams who control work volume from a backlog |

* **What is Kanban**

Kanban is a popular [Agile Software Development Methodology](https://www.geeksforgeeks.org/software-engineering-agile-software-development/). It is a signalling device that instructs the moving of parts in a ‘pull’ production system, developed as part of the TPS (Toyota Production System). Kanban is about envisioning the existing workflow in terms of steps. These steps can be created on the whiteboard.

The Kanban method is an approach to evolutionary and incremental systems and process change for organizations. A work-in-progress limited pull system is the central mechanism to uncover system operation (or process) complications and encourage collaboration to continuously improve the system.

Electronic Kanban boards are also available in ALM tools like Rally (CA Agile), [Jira](https://www.geeksforgeeks.org/jira-tutorial/), [Swift](https://www.geeksforgeeks.org/swift-tutorial/), Kanban, LeanKit, Kanban, etc. Stages could be configured in these tools, and the movement of tickets between stages could be viewed in these tools.

**When Would The Kanban Approach Be Needed?**

Kanban is best suited in the below scenarios:

* Dynamic/ frequent changing requirements which need to be delivered faster.
* In case of changing priorities, the team can pull the prioritized work as soon as the WIP limit drops.
* Frequent releases are there (Periodically).
* When incoming work is continuous.
* Where task priority needs to be decided dynamically based on task nature and type.
* The best suit is for Ticket or Production support projects.
* Kanban could be used by any function of an organization as well, for instance in Marketing, Sales, and HR.

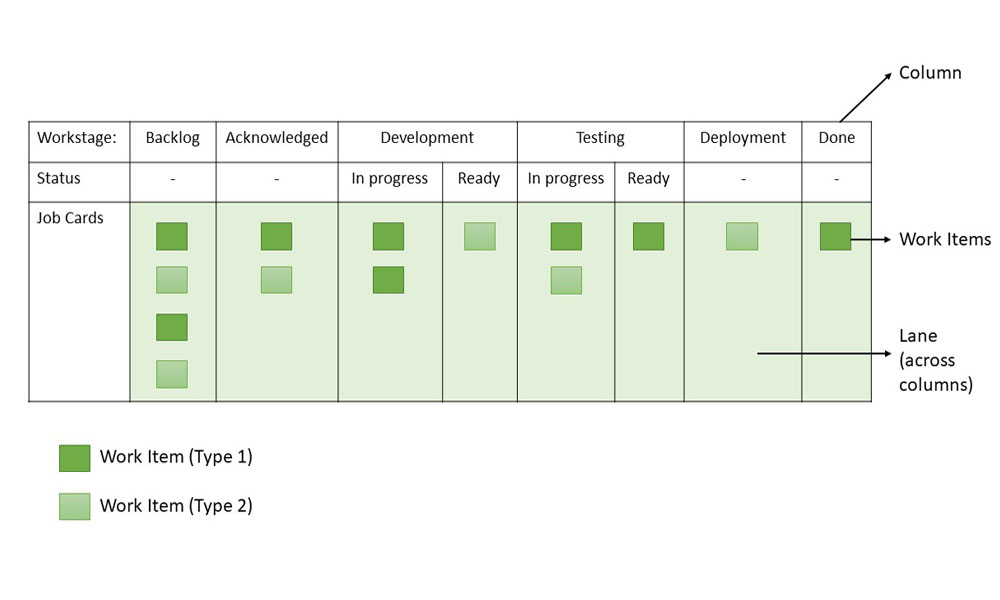
However, Kanban might not be the right fit for projects where:

* Tasks could remain in the ‘wait’ state for long.
* Mainly research-oriented takes are there.
* For enhancements where requirements are evolving/ unclear.
* No prior scope is not defined and tasks keep on evolving.
* Too much dependency is there between tasks.
* If all the items across work stages need to be collated, then only deployed.

**Kanban Board/Card**

It is critical to understand the visualization of workflow stages in the task execution pipeline. [Kanban board](https://www.geeksforgeeks.org/how-can-you-create-a-kanban-board-in-jira/) provides a simple way to understand the process. It can be explained as follows:

1. Every request received is put on the Kanban board.
2. A column on the board represents a stage (these stages are termed as the Work stage) during the lifecycle of bugs/ tickets. For instance, the Kanban board can have 4 stages- Received/ Acknowledged, In-progress, UAT & Done.
3. The received stage could be called a “Backlog” also.
4. The team could decide the names for the phases based on the terminology used by their respective teams.
5. Kanban board could be a simple whiteboard on which sticky notes could be used with ticket details or an electronic Kanban board could be used.
6. ALM tools like Rally/ Jira could be configured to use the Kanban board.
7. The board can give a signal in case the bugs/ tickets are stuck in one stage for a long.
8. For electronic boards, one can configure the Kanban board in a way that tickets/ user stories along with the time stamp are visible.
9. For whiteboards that are maintained manually, the team can enter the date/ time.



**Principles of Kanban**

Kanban is based on four key principles which are mentioned below:

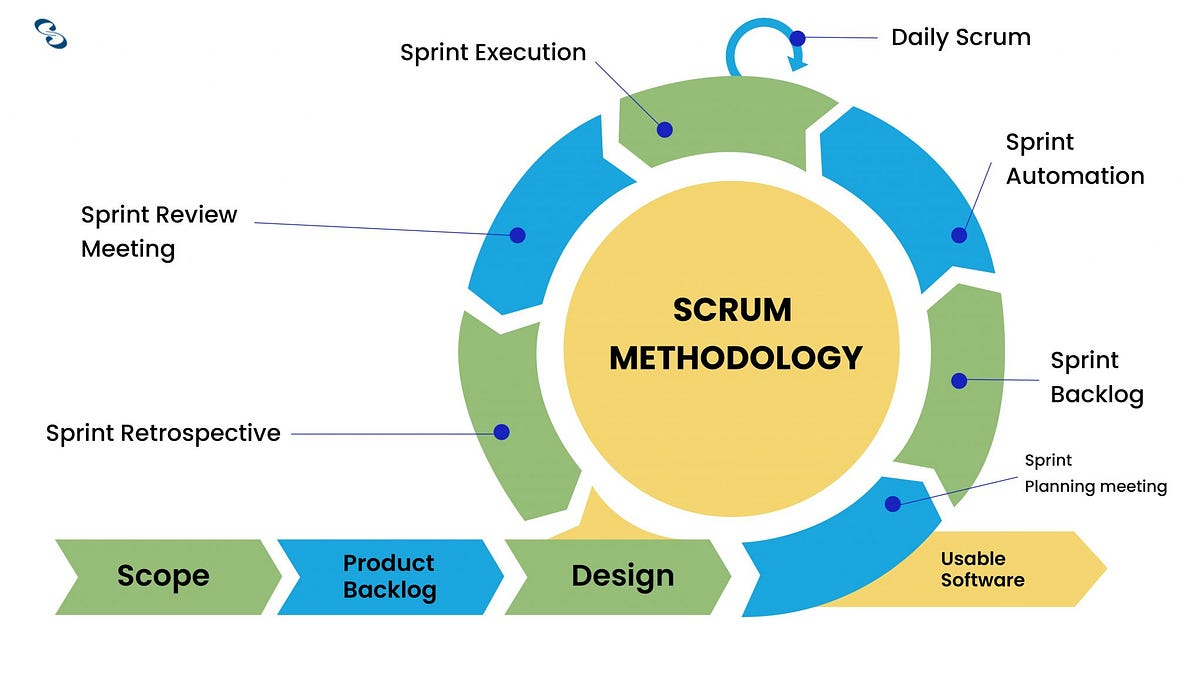
1. **Start with the existing process:**It is a change management method that starts with the existing process. Changes are done in the system in incremental and evolutionary ways. Unlike [Scrum](https://www.geeksforgeeks.org/what-is-a-scrum-team-structure-roles-and-responsibilities/), there’s no specific process or roles defined in Kanban.
2. **Agree to continue evolutionary and incremental changes:**After starting with the existing process, the team must agree on continuous, incremental, and evolutionary changes. The changes should be small and incremental. Rapid and substantial changes may be effective but they will be subjected to larger resistance as well by the Team.
3. **Admire current roles, processes, responsibilities & titles:**Though Kanban suggests continuous incremental changes in the process, it respects current roles, responsibilities, and job titles. This helps the team to gain confidence as they get started with Kanban.
4. **Leadership at all levels:**Kanban does not expect leadership from a specific set, rather the actions of leadership at all levels in the organization, are very much encouraged.

* **What is Scrum ?**

Scrum is an [agile project management](https://www.atlassian.com/en/agile/project-management) framework that helps teams’ structure and manage their work through a set of values, principles, and practices. Much like a rugby team (where it gets its name) training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

While the scrum I’m talking about is most frequently used by software development teams, its principles and lessons can be applied to all kinds of teamwork. This is one of the reasons scrum is so popular. Often thought of as an agile project management framework, scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.

The definition of scrum is based on empiricism and lean thinking. Empiricism says that knowledge comes from experience and that decisions are made based on what is observed. Lean thinking reduces waste and focuses on essentials. The scrum framework is heuristic; it’s based on continuous learning and adjustment to fluctuating factors. It acknowledges that the team doesn’t know everything at the start of a project and will evolve through experience. Scrum is structured to help teams naturally adapt to changing conditions and user requirements, with re-prioritization built into the process and short release cycles so your team can constantly learn and improve.



**The scrum framework**

The scrum framework outlines a set of values, principles, and practices that scrum teams follow to deliver a product or service. It details the members of a scrum team and their accountabilities, “artifacts” that define the product and work to create the product, and scrum ceremonies that guide the scrum team through work.

**Members of a scrum team**

A scrum team is a small and nimble team dedicated to delivering committed product increments. A scrum team’s size is typically small, at around 10 people, but it’s large enough to complete a substantial amount of work within a sprint. A scrum team needs three specific roles: product owner, [scrum master](https://www.atlassian.com/en/agile/scrum/scrum-master), and the development team. And because scrum teams are cross-functional, the development team includes testers, designers, UX specialists, and ops engineers in addition to developers.

**The scrum product owner**

Product owners are the champions for their product. They are focused on understanding business, customer, and market requirements, then prioritizing the work to be done by the engineering team accordingly. Effective product owners:

* Build and manage the product backlog.
* Closely partner with the business and the team to ensure everyone understands the work items in the product backlog.
* Give the team clear guidance on which features to deliver next.
* Decide when to ship the product with a predisposition towards more frequent delivery.

The product owner is not always the [product manager](https://www.atlassian.com/en/agile/product-management/product-manager). Product owners focus on ensuring the development team delivers the most value to the business. Also, it's important that the product owner be an individual. No development team wants mixed guidance from multiple product owners.

**The scrum development team**

Team members have differing skill sets, and cross-train each other so no one person becomes a bottleneck in the delivery of work. Strong scrum teams are self-organizing and approach their projects with a clear ‘we’ attitude. All members of the team help one another to ensure a successful sprint completion.

**Scrum artifacts**

[Scrum artifacts](https://www.atlassian.com/en/agile/scrum/artifacts) are important information used by the scrum team that helps define the product and what work to be done to create the product. There are three artifacts in scrum: product backlog, a sprint backlog, and an increment with your definition of “done”. They are the three constants a scrum team should reflect on during [sprints](https://www.atlassian.com/en/agile/scrum/sprints) and over time.

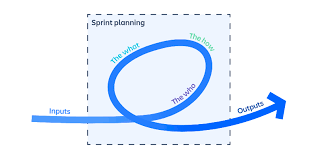
* **Product Backlog** is the primary list of work that needs to get done and maintained by the product owner or product manager. This is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog. It is, essentially, the team’s “To Do” list. The product backlog is constantly revisited, re-prioritized and maintained by the Product Owner because, as we learn more or as the market changes, items may no longer be relevant or problems may get solved in other ways.
* **Sprint Backlog** is the list of items, [user stories](https://www.atlassian.com/en/agile/project-management/user-stories), or bug fixes, selected by the development team for implementation in the current sprint cycle. Before each sprint, in the sprint planning meeting (which we’ll discuss later in the article) the team chooses which items it will work on for the sprint from the product backlog. A sprint backlog may be flexible and can evolve during a sprint. However, the fundamental sprint goal – what the team wants to achieve from the current sprint – cannot be compromised.

## **What is sprint planning?**

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

In [scrum](https://www.atlassian.com/en/agile/scrum), the [sprint](https://www.atlassian.com/en/agile/scrum/sprints) is a set period of time where all the work is done. However, before you can leap into action you have to set up the sprint. You need to decide on how long the time box is going to be, the sprint goal, and where you're going to start. The sprint planning session kicks off the sprint by setting the agenda and focus. If done correctly, it also creates an environment where the team is motivated, challenged, and can be successful. Bad sprint plans can derail the team by setting unrealistic expectations.

* **The What** –  The product owner describes the objective(or goal) of the sprint and what backlog items contribute to that goal. The scrum team decides what can be done in the coming sprint and what they will do during the sprint to make that happen.
* **The How** – The development team plans the work necessary to deliver the sprint goal. Ultimately, the resulting sprint plan is a negotiation between the development team and product owner based on value and effort.
* **The Who** – You cannot do sprint planning without the product owner or the development team. The product owner defines the goal based on the value that they seek. The development team needs to understand how they can or cannot deliver that goal. If either is missing from this event it makes planning the sprint almost impossible.
* **The Inputs** – A great starting point for the sprint plan is the product backlog as it provides a list of ‘stuff’ that could potentially be part of the current sprint. The team should also look at the existing work done in the increment and have a view to capacity.
* **The Outputs** – The most important outcome for the sprint planning meeting is that the team can describe the goal of the sprint and how it will start working toward that goal. This is made visible in the sprint backlog.



**Prep for sprint planning meeting**

Running a great sprint planning event requires a bit of discipline. The product owner must be prepared, combining the lessons from the previous [sprint review](https://www.atlassian.com/en/agile/scrum/sprint-reviews), stakeholder feedback, and vision for the product, so they set the scene for the sprint. For transparency, the [product backlog](https://www.atlassian.com/en/agile/scrum/backlogs) should be up-to-date and refined to provide clarity. [Backlog refinement](https://www.atlassian.com/en/agile/scrum/backlog-refinement) is an optional event in scrum, because some backlogs don’t need it. However, for most teams, it’s better to get the team together to review and refine the backlog prior to sprint planning.

Story points are a unit of measurement used in Jira to estimate the amount of effort required to complete a task. They are a key concept in agile planning and are used to assess, prioritize, and track work.

How are story points used?

* **Estimating work**: Story points are used to estimate the effort required to complete a user story or other work item.
* **Tracking progress**: Story points are used to track the progress of work on a board.
* **Prioritizing work**: Story points are used to prioritize work by ranking it from simple to difficult.
* **Assigning resources**: Story points are used to assign resources efficiently and accurately.

How are story points assigned?

* Story points are assigned based on the amount of work, complexity, and risk or uncertainty.
* Story points are generally assigned on a scale of one (lowest) to five (highest).
* The Fibonacci sequence is a common way to assign story points.

Why are story points used?

* Story points help teams understand how much they can achieve in a period of time.
* Story points help teams build consensus and commitment to a solution.
* Story points help teams make tougher decisions around the difficulty of work.

**What is a product backlog?**

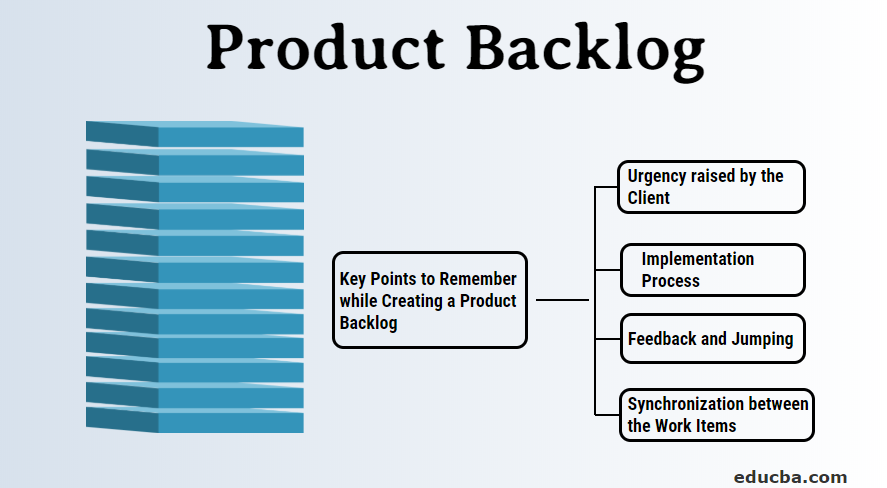
A product backlog is a prioritized list of work for the development team that is derived from the [product roadmap](https://www.atlassian.com/en/agile/product-management/product-roadmaps) and its requirements. The most important items are shown at the top of the product backlog so the team knows what to deliver first. The development team doesn't work through the backlog at the [product owner's](https://www.atlassian.com/en/agile/product-management) pace and the product owner isn't pushing work to the development team. Instead, the development team pulls work from the product backlog as there is capacity for it, either continually ([kanban](https://www.atlassian.com/en/agile/kanban)) or by iteration ([scrum](https://www.atlassian.com/en/agile/scrum)).

**How to effectively manage a product backlog**

Once the product backlog is built, it's important to regularly maintain it to keep pace with the [program](https://www.atlassian.com/en/agile/project-management/project-management-intro). Product owners should review the backlog before each iteration planning meeting to ensure prioritization is correct and feedback from the last iteration has been incorporated. Regular review of the backlog is often called "backlog grooming" in agile circles (some use the term [backlog refinement](https://www.atlassian.com/en/agile/scrum/backlog-refinement)).

Once the backlog gets larger, product owners need to group the backlog into near-term and long-term items. Near-term items need to be fully fleshed out before they are labelled as such. This means complete user stories have been drawn up, collaboration with design and development has been sorted out, and estimates from development have been made. Longer term items can remain a bit vague, though it's a good idea to get a rough estimate from the development team to help prioritize them. The key word here is "rough": estimates will change once the team fully understands and begins work on those longer term items.

The backlog serves as the connection between the product owner and the development team. The product owner is free to re-prioritize work in the backlog at any time due to customer feedback, refining estimates, and new requirements. Once work is in progress, though, keep changes to a minimum as they disrupt the development team and affect focus, flow, and morale.



**Product backlogs keep teams agile**

The product backlog also serves as the foundation for iteration planning. All work items should be included in the backlog: user stories, bugs, design changes, [technical debt](https://www.atlassian.com/en/agile/software-development/technical-debt), customer requests, action items from the retrospective, etc. This ensures everyone's work items are included in the overall discussion for each iteration. Team members can then make trade-offs with the product owner before starting an iteration with complete knowledge of everything that needs to be done.