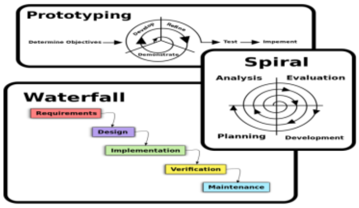
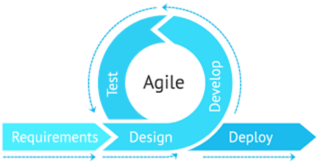


Software Engineering
Agile Requirement

Agile Development
Develop Functionality 1
Integrate and Test
Develop Functionality 2
Integrate and Test
Develop Functionality N
Integrate and Test
Demo Release
Client's Feedback
Make Changes
System Testing
All Functionalities Complete?
Yes
No
Next Iteration



Outline

- Agile Requirements Abstraction Model
- Requirements Management in Agile Environment
- Agile Requirements Prioritization
- Agile Requirements Modelling and Generation
- Collaborative User Story Creation

What are Agile Requirements modeling?

- The basic idea of agile requirements modeling is to create a foundation for your project's specifications for higher-level understanding.
- This is done in the beginning stages with details being added as you need them on a just-in-time basis.
- This is the process of developing software where requirements and solutions are constantly evolving through a collaborative effort.
- With the use of self-organizing and cross-functional teams, customer and stakeholder specifications are always met.
- By underpinning the project's requirements at the beginning of the software build, it is easy to identify the critical aspects.
- This approach is generally extremely iterative and collaborative while remaining flexible to address the risks of the project.

What are Agile Requirements modeling?

- As a result, a collection of values, principles, and practices are established for the development of a model to ensure that builders and inputs remain on track.
- The best way to understand this process is to look at its main goals.
- Agile requirements modeling is designed to support the goals of software development and aims to achieve the following objectives:
  - Establish the best practices for an effective model
  - Outline the ways to put those practices into place
  - Display alternatives to improve the modeling approach

Types of Requirements

- There are two categories that agile requirements modeling can be divided into.
- The distinction between the two is important to establish as they distinguish the technicalities of a project from the user interactions.
- Although differentiating these categories isn't essential, it can establish a clear outline of key components for your project.
- These two categories are:
  - Behavioral: This form of requirement refers to any user interface issues, usage, and business rules. This can also be seen as the functional requirements of a project.
  - Non-behavioral: On the technical side, non-behavioral looks at a system's features relating to availability, security, performance, interoperability, dependability, and reliability.

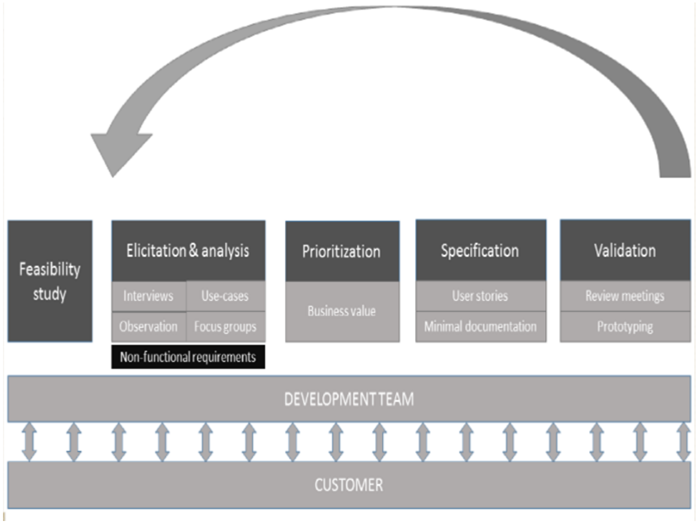
Applying Agile Requirements Modeling Principles

- These principles include:
  - Outlining how the software is your primary goal
  - Establishing the next effort as your secondary goal
  - Remaining agile by focusing only on the models required
  - Assuming simplicity
  - Embracing change
  - Modeling with purpose
  - Creating multiple models
  - Producing quality work
  - Maximizing stakeholder investment

Critical Aspects of Agile Requirements modeling

- There are several modeling-orientated aspects of agile requirements modeling. Depending on your point of view and understanding of agile modeling, effective practices will differ.
- However, the following factors make up the key components of this method of project management and planning.
- The end goal is to gain a better understanding of the requirements at hand and develop a clear plan of action.
- Some schools of thought separate these critical aspects into two categories high-level models and models in iterations. High-level models provide teams with a quick picture of the project's breadth. This is a broader approach to the project where teams can quickly determine where to dive in.

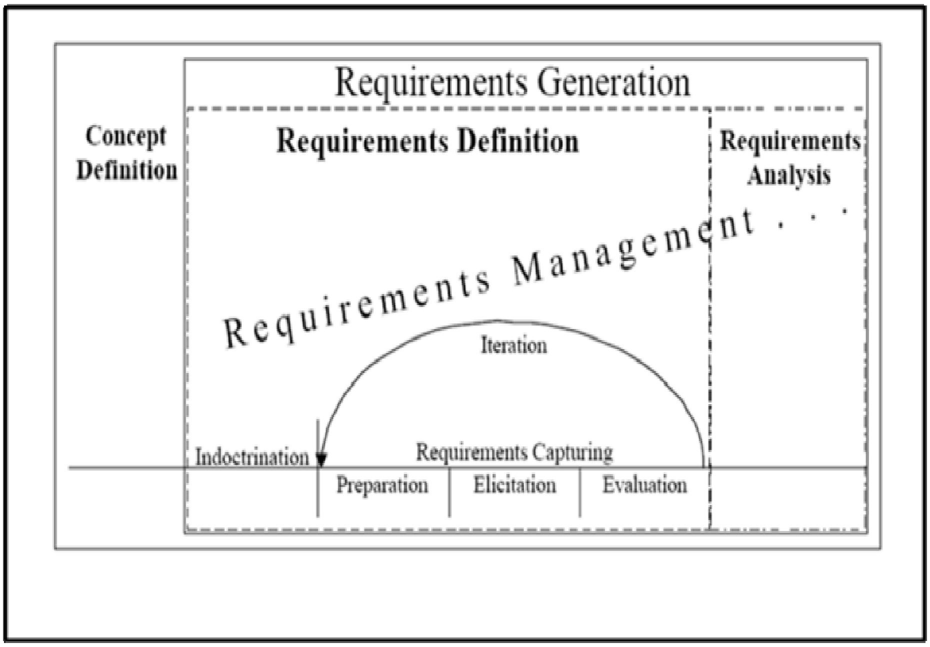
A model for agile requirements engineering



Requirements Generation Model (RGM)

- The RGM is a structured approach to capturing requirements.
- The RGM covers all the activities of the requirements engineering process namely requirements elicitation, analysis, specification, verification and management.
- "The RGM is based on two components,
  1. a framework that structures and controls the activities within which the customer and the requirements engineer should define requirements, and
  2. a monitoring methodology that ensures that all requirements elicitation activities follow proper procedures".

Requirements Generation Model Framework



The phases in the RGM

- **Indoctrination phase :**
  - Governs all preparation activities up to and including the initial requirements elicitation meeting between the customer and the requirements engineer.
  - The objectives of this phase are to educate the customers about the RGM and to ensure that the developers learn the business process of the customers, their problem domain and needs.

The phases in the RGM

- **Requirements Capturing phase :**
  - This phase involves the three activities of preparation, elicitation and evaluation.
    - <u>Preparation</u> – scope of the elicitation meeting is defined and identified issues are resolved.
    - <u>Elicitation</u> – requirements are identified and recorded accurately
    - <u>Evaluation</u> – identified requirements are reviewed, new or unresolved issues are identified and the need for additional iterations of the requirements capturing phase is determined

Agile Requirements Generation Model(Agile RGM)

- Agile RGM is designed for short term (having less than one year development period) agile projects with 60 to 90 day release cycles.
- Agile practices such as minimal documentation, refactoring and TDD are not considered suitable for large systems.
- The Agile RGM is designed to adopt these practices. Hence, the target systems for the Agile RGM are small scale systems.
- As the length of the development lifecycle is taken into account, the Agile RGM describes not only the RE process activities but the complete development process as well.
- The RGM is a structured approach to capturing requirements. It provides guidelines and protocols that practitioners must adopt in order to gather requirements. The structure of the RGM is preserved in the Agile RGM.

Agile Requirements Generation Model(Agile RGM)

- The Agile RGM attempts to structure the agile RE process with minimal impact on agility.
- It reflects the agile principles such as direct stakeholder involvement, evolutionary requirements, refactoring, just-in-time gathering of details and minimal documentation.
- Verification and validation activities in the existing agile methods are predominantly validation efforts carried out implicitly. The Agile RGM explicitly specifies validation as an activity for each phase.

Phases of (Agile RGM)

1. **Education Phase –**

- The objective of this phase is to establish rapport [friendly relation] among the various project stakeholders.
- The development team learns the business process of the customers.
- Also, a high level mission statement for the project is created which serves as the input to the next phase in the model.

Phases of (Agile RGM)

2. **Feature Development Phase –**

- The stakeholders derive the high level features of the system from the high level mission statement created during the Education Phase.
- These features are sets of functionality that have business value to the customer.
- Only one feature or a subset of the identified features is chosen for development during a release.
- The identified features are validated, the time for their completion estimated and are prioritized based on their value to the customers.
- The output of this phase is a prioritized feature stack.
- During any release cycle, the subset of features chosen for development during the current release will be the input to the Story Development Phase.

Phases of (Agile RGM)

3. Story Development Phase –

- Each identified feature is decomposed into stories.
- The stories are brief descriptions of user- or customer-valued functionality.
- The stories are validated and the developers estimate the time required for their completion.
- The stories are implemented during the iterations.
- Hence, the developers estimate the time required to implement each story in this phase.
- The stories are then prioritized and stored in a prioritized story stack.
- A subset of the prioritized stories is chosen for development during an iteration which lasts for about 2 to 4 weeks.
- This subset serves as the input to the Task Development Phase.

Phases of (Agile RGM)

4. Task Development Phase –

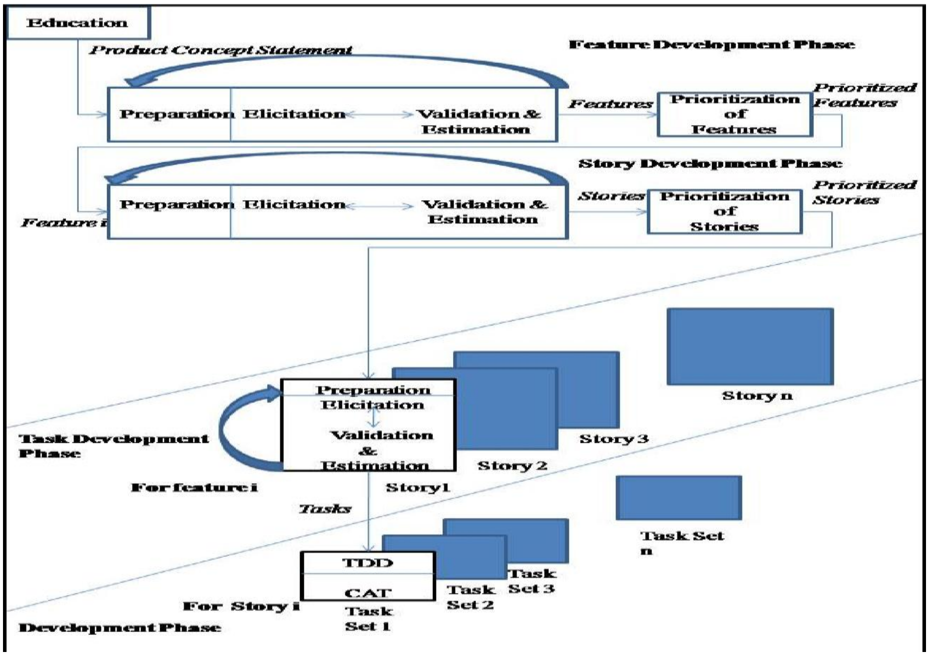
- Tasks are essentially checklists for developers which outline the details required for implementing the stories.
- Each story identified in the previous phase is decomposed into tasks.
- The developers validate the tasks and estimate the time required to complete the tasks.
- Each task is estimated to take a few hours to be implemented.
- These tasks are then developed during the Development Phase.

Phases of (Agile RGM)

5. Development Phase –

- The tasks identified in the previous phase are implemented using Test-Driven Development (TDD) approach.
- The customers and developers perform acceptance testing to ensure that the system meets the customer and user needs.

Phases of (Agile RGM)



Agile RGM Priority

- The customers prioritize the identified features based on the business value of each feature.
- These prioritized features are stored in a stack in the order of their priorities. This stack is known as a <u>prioritized feature stack.</u>
- Only one feature or a subset of the identified features is chosen for implementation during a release cycle.
- The details for this feature or subset of features are gathered just-in-time.
- The remaining features will be implemented during the future release cycles.
- The stakeholders should strive to identify as many features as possible before proceeding to the next phase in order to be informed of the scope of the system.

Agile RGM Priority

- Each feature chosen to be implemented during the current release is decomposed into stories.
- If multiple teams are involved in the development of the system, each team can work towards decomposing one or more features into stories.
- Stories are user- or customer- expected functionality.
- The stories for each feature are identified over a number of iterations and are then validated, estimated and prioritized.
- The prioritized stories are stored in a prioritized story stack.
- Each story is then decomposed into tasks during the Task Development Phase. The task list for each story is a to-do list for the developers giving the details required to implement the stories.
- The tasks are then developed using TDD and the customers and developers test execute the acceptance tests

Three level Scale

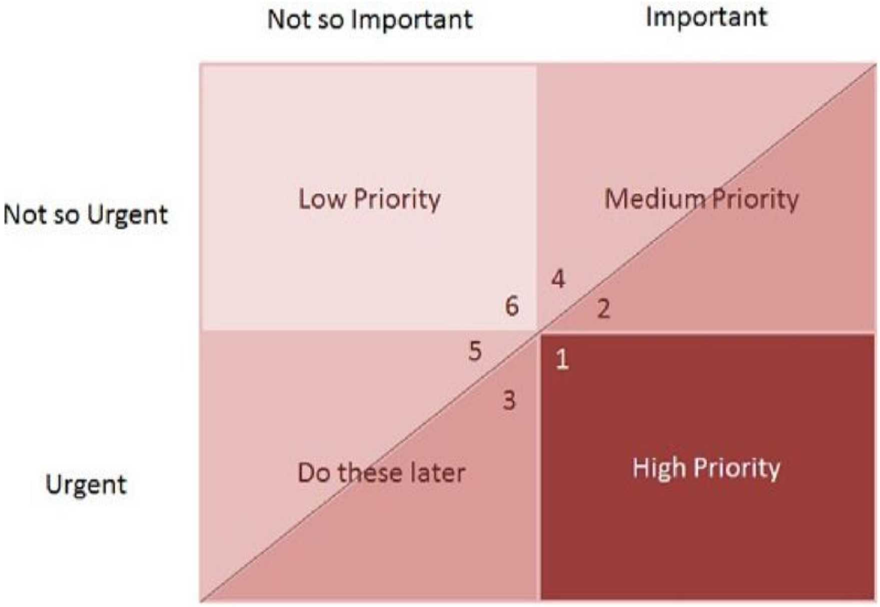
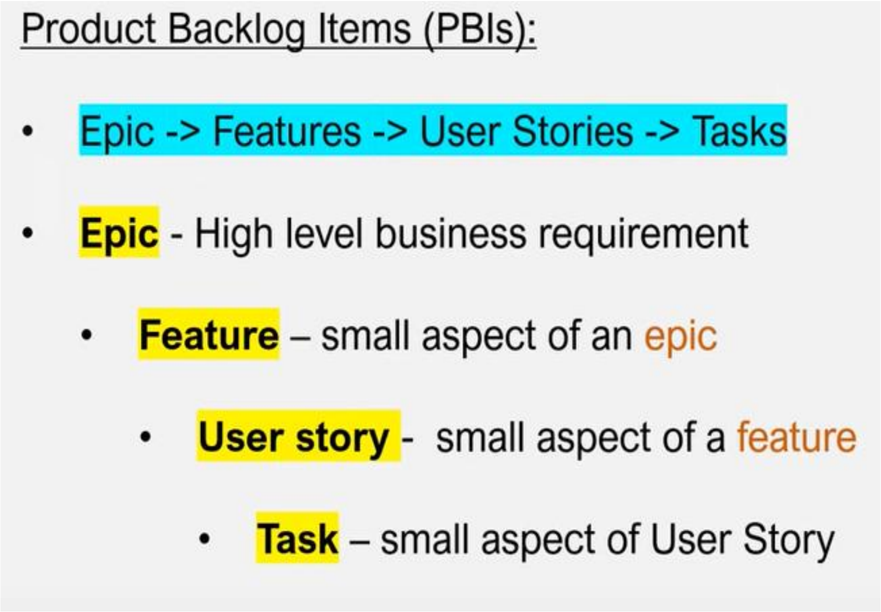


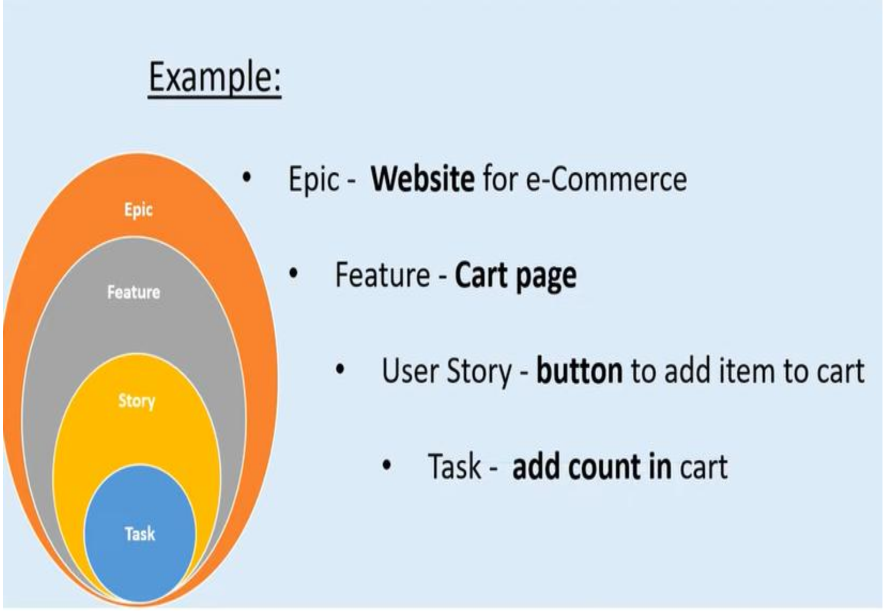
Figure shows how each feature is decomposed into multiple stories and each story into multiple tasks.



Product Backlog Items (PBIs):

- Epic -> Features -> User Stories -> Tasks
- **Epic** - High level business requirement
  - **Feature** – small aspect of an epic
    - **User story** - small aspect of a feature
      - **Task** – small aspect of User Story

Figure shows how each feature is decomposed into multiple stories and each story into multiple tasks.



Example:

- Epic - **Website** for e-Commerce
    - Feature - **Cart page**
        - User Story - **button** to add item to cart
            - Task - **add count in** cart

Use case & Usage Scenarios

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”
- An actor is a person or device that interacts with the software

Each scenario answers the following questions

1. Who is the primary actor, the secondary actor (s)?
2. What are the actor's goals?
3. What preconditions should exist before the story begins?
4. What main tasks or functions are performed by the actor?
5. What extensions might be considered as the story is described?
6. What variations in the actor's interaction are possible?
7. What information does the actor desire from the system?
8. What system information will the actor acquire, produce, or change?
9. Will the actor have to inform the system about changes in the external environment?
10. Does the actor wish to be informed about unexpected changes?

Usage Scenarios & Story Writing

- Scenarios are created by user researchers to help communicate with the design team.
- User stories are created by project/product managers to define the requirements prior to a sprint in agile development.
- Scenarios are stories that capture the goals, motivations, and tasks of a persona in a given system.
- User stories provide a rapid way of handling customer requirements instead of formal requirement documents
- Gherkin language is used to writing an effective story of the system requirement.
- Gherkin is a human-readable language for system behavior description, which uses indentation to define the structure of the document.
- Each line starts with one of the keywords and describes one of the steps.

Login Usage Scenarios and Story

Feature	Login
Scenario	User Login with valid username and password
Prerequisite	User must have proper client installed on user terminal.
Story	**Given**: User navigated to the Login Screen. **When**: User enters the correct User Name. **And** the user enters the correct password. **And** user Click “login” button. **Then**: System verify user information. **display** dashboard of user. **display** username on top of the right side. **display** logout button.

Login Usage Scenarios and Story

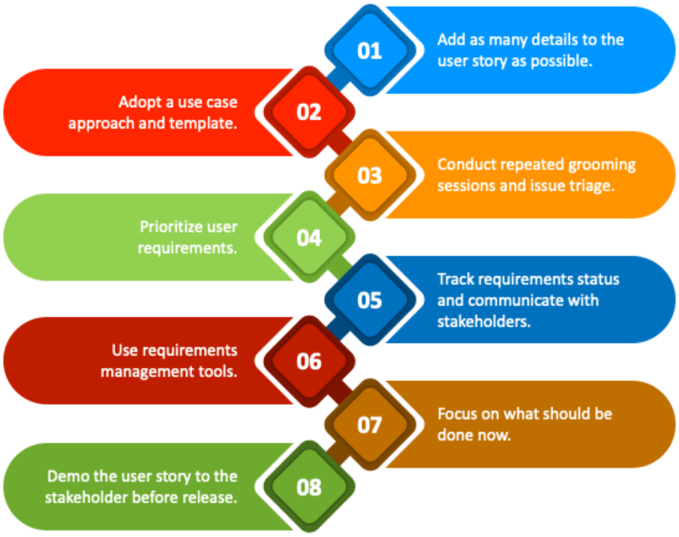
Feature	Login
Scenario	User Login with invalid username and password
Prerequisite	User must have proper client installed on user terminal.
Story	**Given**: User navigated to the Login Screen. **When**: User enters the wrong User Name. **And** the user enters the wrong password. **And** user Click “login” button. **Then**: System verify user information. **display** error message for invalid username and password.

Example of User Story with Acceptance Criteria

- User story: As a user, I want the system to recover the password so that I will be able to access my account in case I forgot my password.
- Scenario: Forgot password.
- **Given**: The user navigates to the login page.
- **When**: The user selects the <forgot password> option.
- **And**: Enter a valid email to receive a link for password recovery.
- **Then**: The system sends the link to the entered email.
- **Given**: The user receives the link via the email.
- **When**: The user navigates through the link received in the email.
- **Then**: The system enables the user to set a new password.

AGILE REQUIREMENTS GATHERING

8 Tips for Efficient Agile Requirements Gathering Process



01 Add as many details to the user story as possible.

02 Adopt a use case approach and template.

03 Conduct repeated grooming sessions and issue triage.

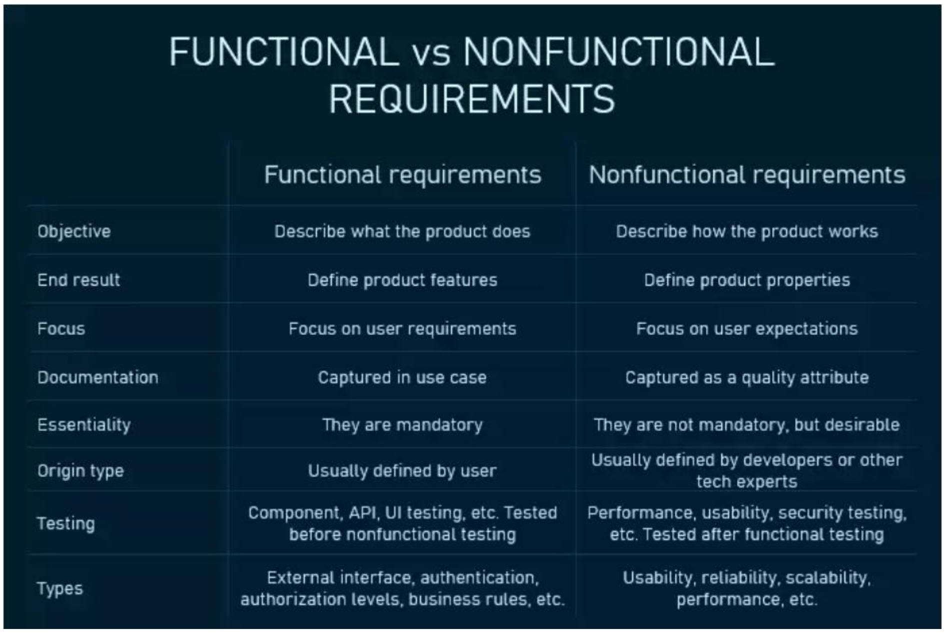
04 Prioritize user requirements.

05 Track requirements status and communicate with stakeholders.

06 Use requirements management tools.

07 Focus on what should be done now.

08 Demo the user story to the stakeholder before release.



FUNCTIONAL vs NONFUNCTIONAL REQUIREMENTS

	Functional requirements	Nonfunctional requirements
Objective	Describe what the product does	Describe how the product works
End result	Define product features	Define product properties
Focus	Focus on user requirements	Focus on user expectations
Documentation	Captured in use case	Captured as a quality attribute
Essentiality	They are mandatory	They are not mandatory, but desirable
Origin type	Usually defined by user	Usually defined by developers or other tech experts
Testing	Component, API, UI testing, etc. Tested before nonfunctional testing	Performance, usability, security testing, etc. Tested after functional testing
Types	External interface, authentication, authorization levels, business rules, etc.	Usability, reliability, scalability, performance, etc.