4.

Agenda

- Test design techniques. EP and BVA.
  - Test Design Techniques lvl2. State Transition and others.
- Specification. What is it?
  - Specification Types. User Story. Acceptance Criteria.
  - Practice “Adventure Park”.
- Project Documentation.
  - Agile Documentation
  - DOD
  - DOR
  - Grooming session. Goals. Estimation.

BVA and EP. Test design techniques.

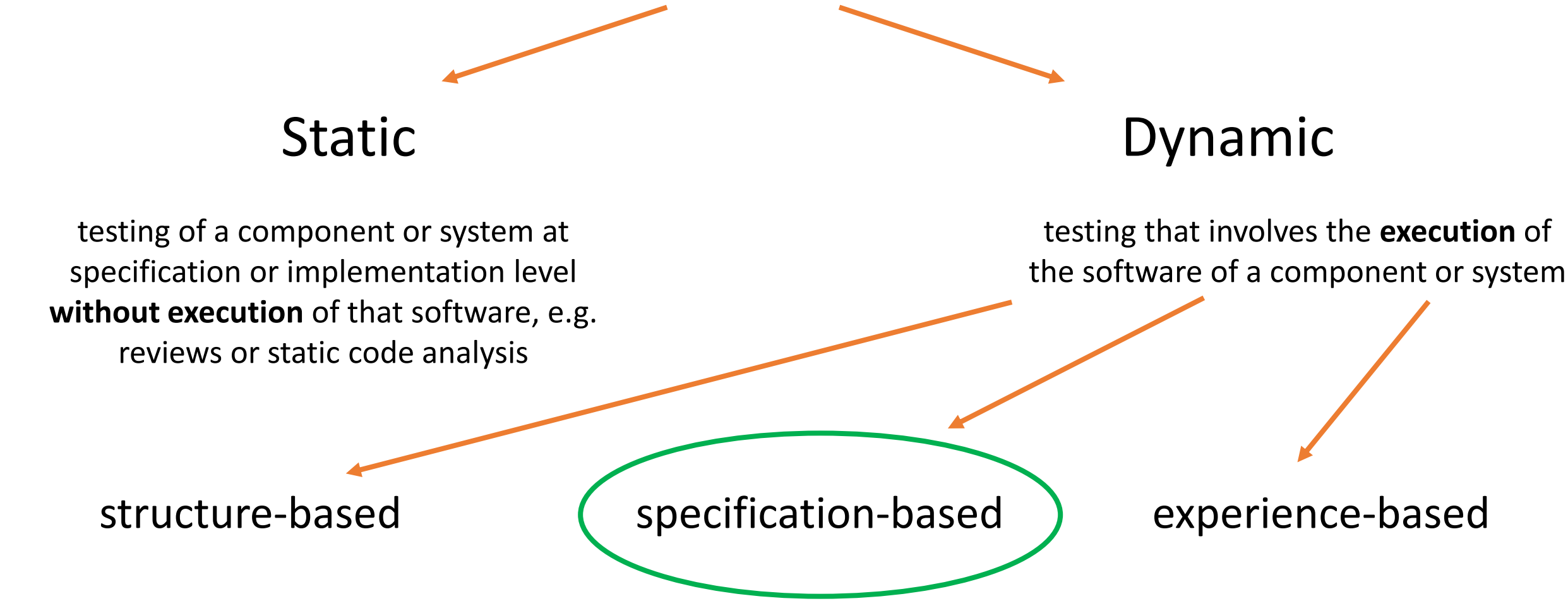
Fig. Boundary Value Analysis -

Example - As per the requirement, the textbox should accept the values from 4 to 20

Test Cases -

	Partitions	Input Value	Expected Result
Negative Test	Below Minimum Boundary Value	3	Fail
	Minimum Boundary Value	4	Pass
	Above Minimum Boundary Value	5	Pass
	Below Maximum Boundary Value	19	Pass
	Maximum Boundary Value	20	Pass
Negative Test	Above Maximum Boundary Value	21	Fail

Reminder: Test design techniques



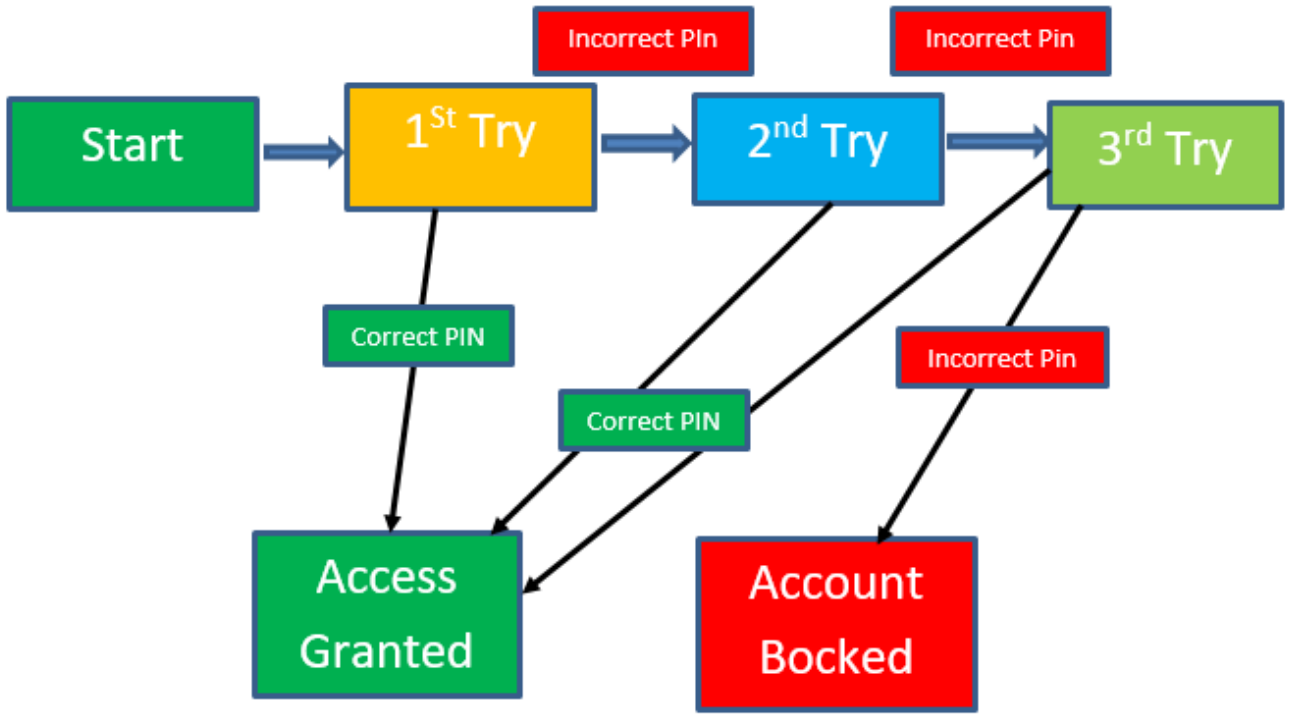
Test design techniques lvl2 (Spec based.)

State Transition Diagrams:

State Transition testing is defined as the software testing technique in which changes in input conditions cause's state changes in the Application under Test (AUT).

It is a black box testing technique in which the tester analyzes the behavior of an application under test for different input conditions in a sequence. In this technique, tester provides both positive and negative input test values and record the system behavior.

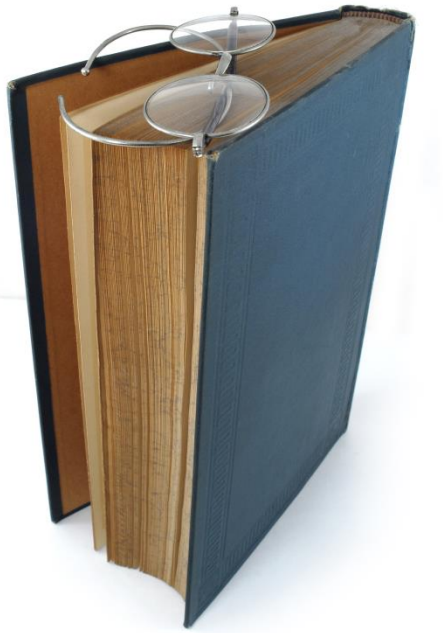
Example: Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.



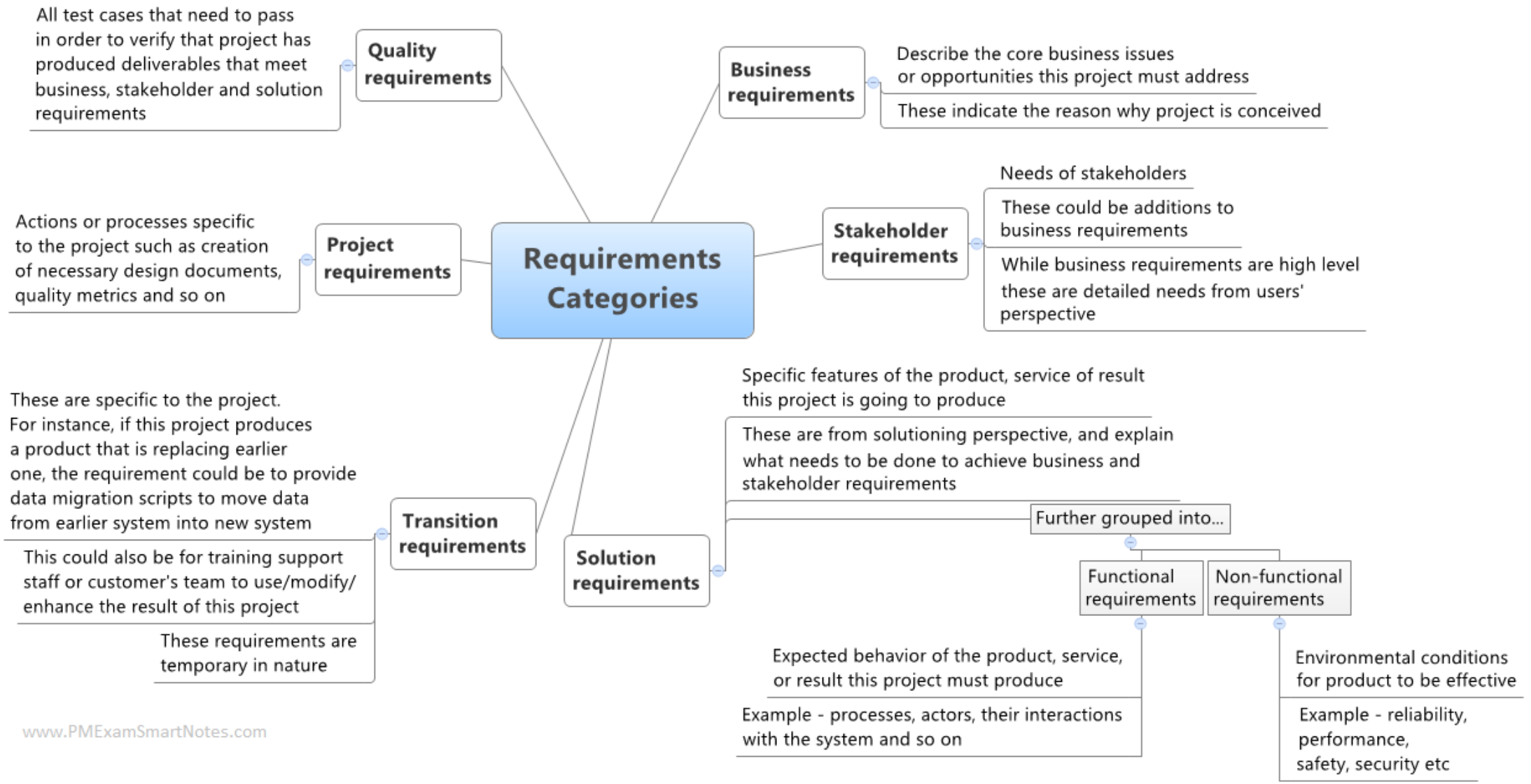
Agenda

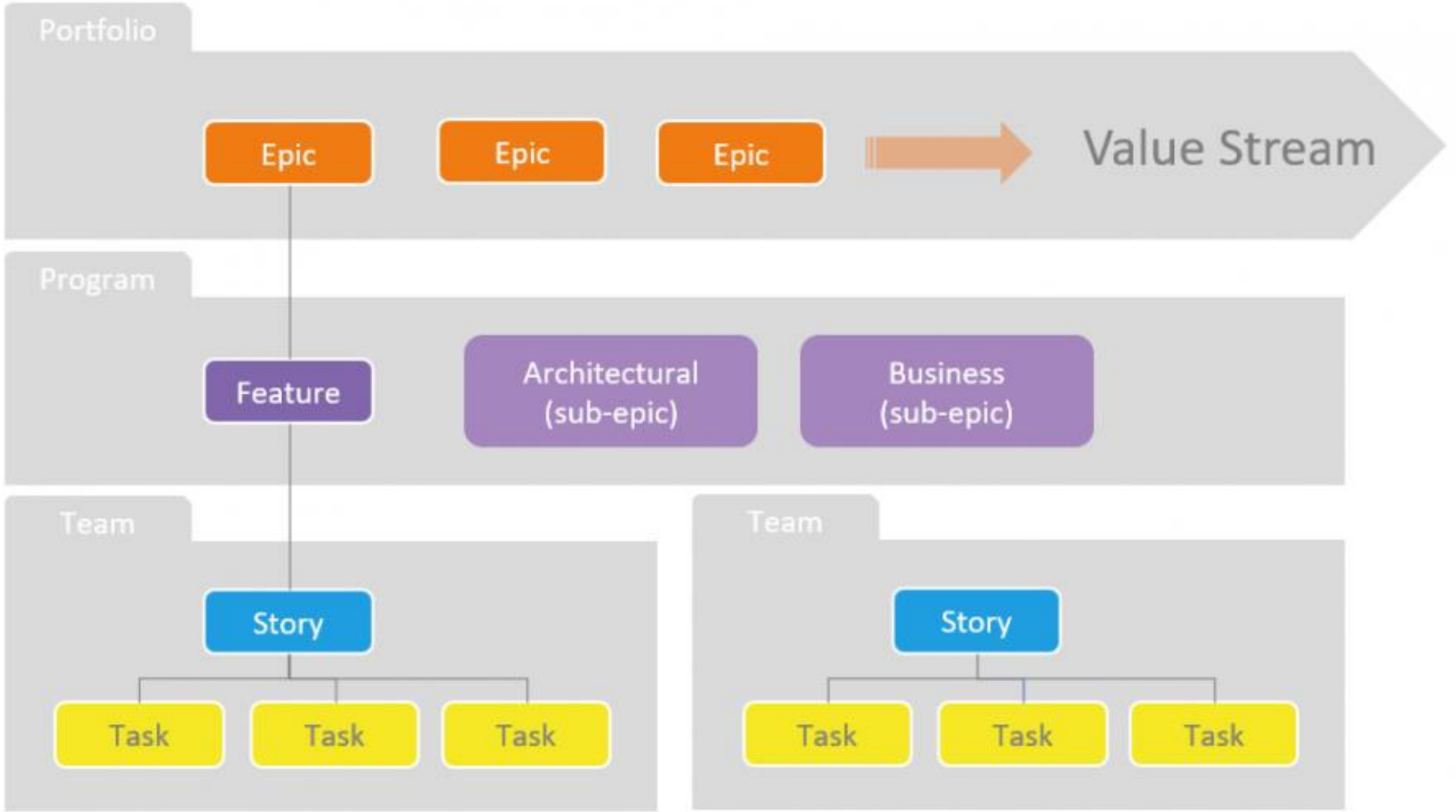
- Test design techniques. EP and BVA.
  - Test Design Techniques lvl2. State Transition and others.
- Specification. What is it?
  - Specification Types. User Story. Acceptance Criteria.
  - Practice “Adventure Park”.
- Project Documentation.
  - Agile Documentation
  - DOD
  - DOR
  - Grooming session. Goals. Estimation.

What is specification?



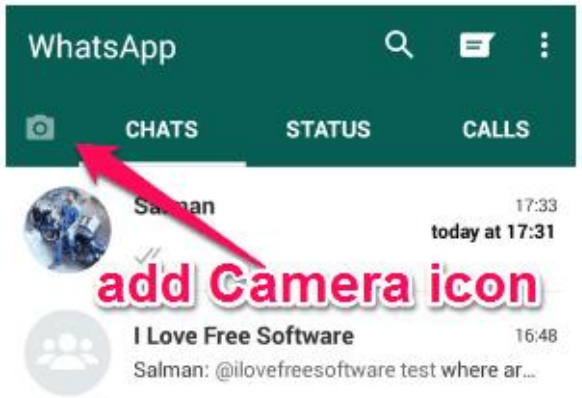
The **software requirements specification** (SRS) lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide.



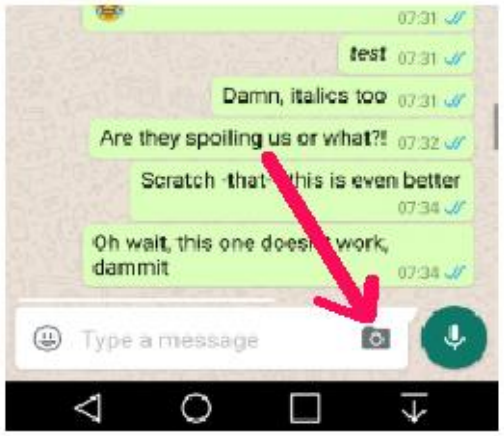

Portfolio
Epic
Epic
Epic
Value Stream
Program
Feature
Architectural (sub-epic)
Business (sub-epic)
Team
Story
Task
Task
Task
Team
Story
Task
Task
Task

User Story: Example

As a <user role> I want to < goal > so that I can <reason of the goal>.

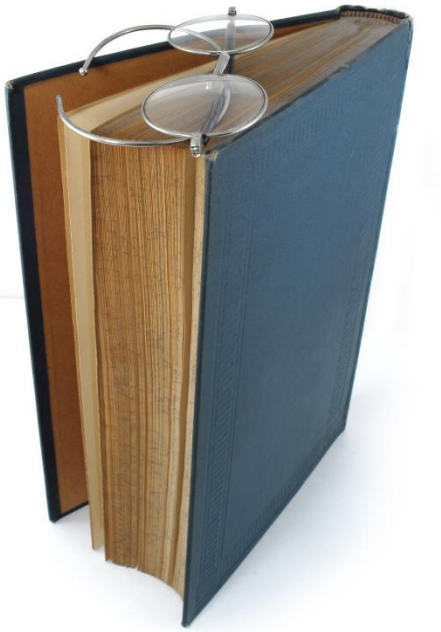


OR



As a WhatsApp user, I want a camera icon in the chat write box to capture and send pictures so that I can click and share my pictures simultaneously with all my friends.

User Story: Acceptance Criteria



Acceptance criteria is a set of accepted conditions or business rules which the functionality or feature should satisfy and meet, in order to be accepted.

Acceptance criteria

- Acceptance criteria are a formalized list of requirements items that ensure that the task/ enhancement specification completed and all test cases are taken into account. We put acceptance criteria specifying conditions under each task/ enhancement.

- It’s natural that different people see the same problem from different angles. Clearly written criteria introduce a single solution to the functionality you intend to implement supporting task/ enhancement specification.

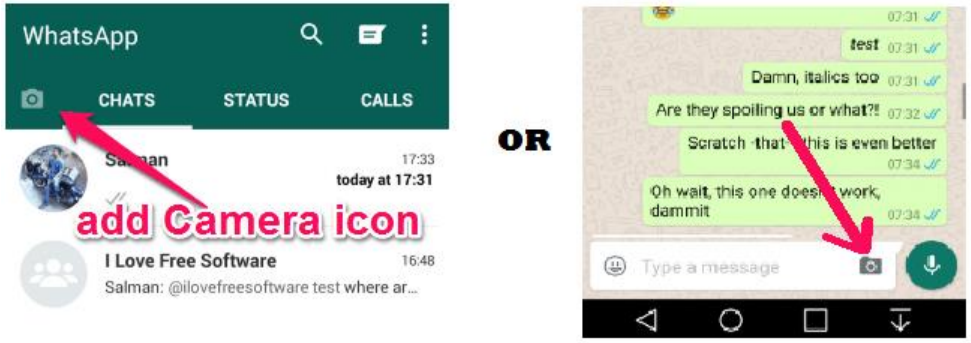
Acceptance criteria: What are Acceptance Criteria Used For?

- To define boundaries. Acceptance criteria help development teams define the boundaries of tasks/ enhancements. In other words, acceptance criteria help you confirm when the application functions as desired, meaning that a user story is completed. To reach consensus. Having acceptance criteria synchronizes the development team with the client. The team knows exactly what conditions should be met, just as the client knows what to expect from the app.

- To serve as a basis for tests. Last but not least, acceptance criteria are a cornerstone of positive and negative testing aimed at checking if a system works as expected.

- To allow for accurate planning and estimation. Acceptance criteria scenarios allow for the correct division of tasks/ enhancements into tasks/ enhancements so they are correctly estimated and planned.

Acceptance Criteria Example



- When I click on a picture, I should be able to add a caption to the image before sending it.
- If there is some problem with starting my phone camera, an error message like ‘Camera could not be started’ should be shown accordingly.
- …

Good Requirements

Characteristic	Explanation
Complete	Everything the software is supposed to do and responses of the software to all classes of input data are specified in the SRS
Consistent	The requirement does not contradict any other requirement
Correct	Every requirement in the SRS represents something required in the final system.
Unambiguous	The requirement is concisely stated without recourse to technical jargon, acronyms (unless defined elsewhere in the Requirements document). Every requirement has one and only one interpretation.
Verifiable	There is a cost effective method that can check whether the final software meets the requirement.

Requirements specification: Practice

Adventure park provides routes of different complexity levels for children and adults. To apply please consider below information.

Task: Find out a questions to the specification of Adventure park

Prices:

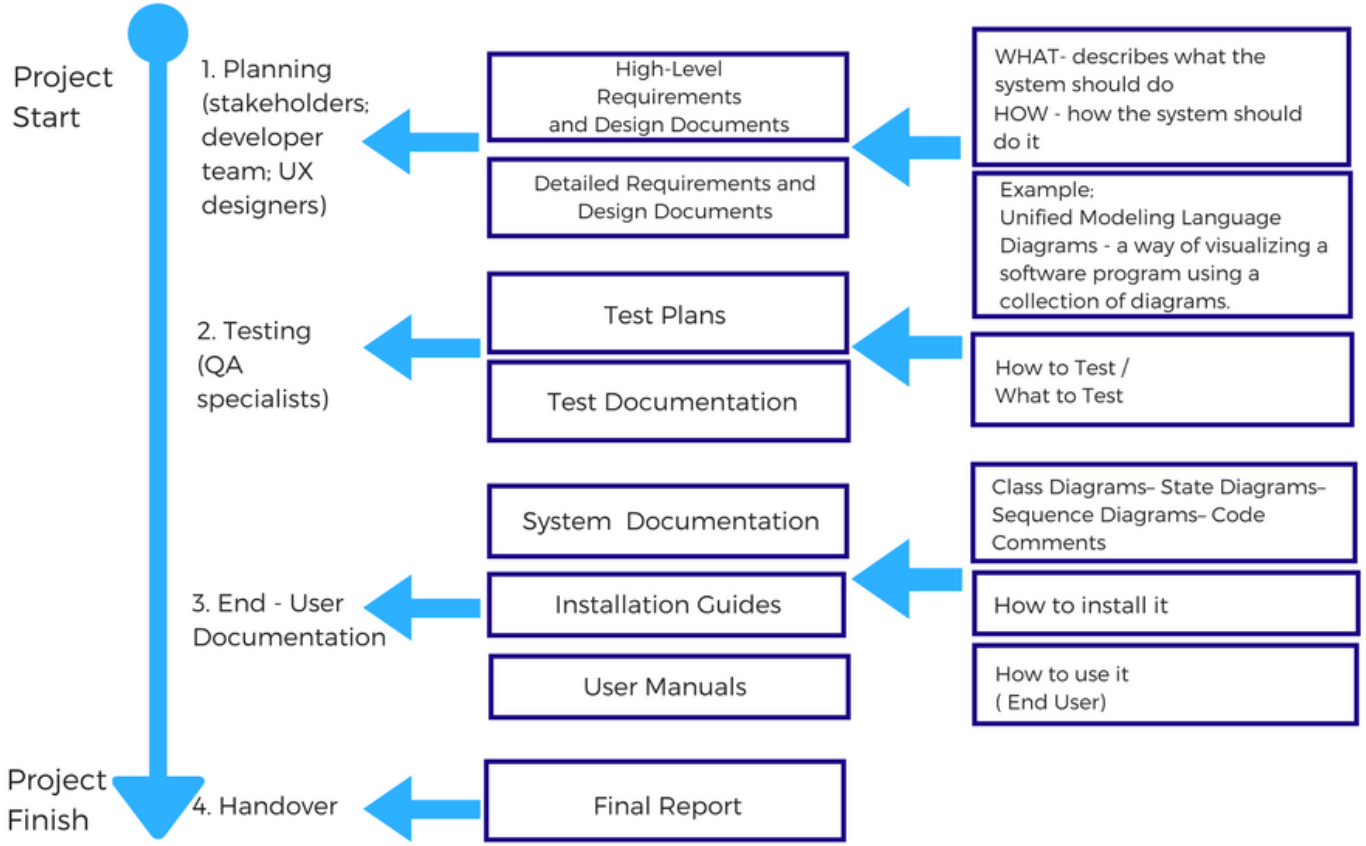
Kids who can use only **yellow** routes = 5.00 EUR
Kids who can use **green** routes = 8.00 EUR
Kids (up to 17 years old) = 12.00 EUR
Adults = 18.00 EUR

- **Yellow** route is allowed for beginners and kids from 110 cm height
- **Green** route is higher than yellow, more complex and allowed for kids from 140 cm height
- **Blue** route is higher than green and allowed for kids from 160 cm height
- **Black** route is more complex than previous ones and allowed for sports people

Agenda

- Test design techniques. EP and BVA.
  - Test Design Techniques lvl2. State Transition and others.
- Specification. What is it?
  - Specification Types. User Story. Acceptance Criteria.
  - Practice “Adventure Park”.
- Project Documentation.
  - Agile Documentation
  - DOD
  - DOR
  - Grooming session. Goals. Estimation.

Project Documentation



Agile Project usual documents list

- Project Scope
- Product Backlog Items
- Coding policy or Coding guidelines
- UX guideline
- DoD
- DoR
- Templates: Bug, test case
- RTM (traceability matrix)
- Test report(s)

Definition of Ready

- These considerations are often summarized as the "**INVEST** criteria", and they provide us with a useful **Definition of Ready** which can be applied to Task. By actively participating in Product Backlog refinement, a good Development Team will collaborate with the Product Owner in making sure that a standard such as this is observed.

- **I (Independent).** The Task should be self-contained and it should be possible to bring it into progress without a dependency upon another Task or an external resource.
- **N (Negotiable).** A good Task should leave room for discussion regarding its optimal implementation.
- **V (Valuable).** The value a Task delivers to stakeholders should be clear.
- **E (Estimable).** A Task must have a size relative to other Tasks.
- **S (Small)**. Tasks should be small enough to estimate with reasonable accuracy and to plan into a time-box such as a Sprint.
- **T (Testable).** Each Task should have clear acceptance criteria which allow its satisfaction to be tested.

Definition of Done

- For scrum teams, it's really important to have a solid definition of what "done" means. They work in sprints and need some way of deciding whether a user story is actually finished. It's no good ending a sprint with a user story that meets all its acceptance criteria, but had no code review, hasn't been tested and isn't deployable. Such a story is clearly not done. Scrum teams solve this by strictly defining "done".

- The definition of done (DoD) is when all conditions, or acceptance criteria, that a software product must satisfy are met and ready to be accepted by a user, customer, team, or consuming system. We must meet the definition of done to ensure quality. It lowers rework, by preventing user stories that don't meet the definition from being promoted to higher level environments. It will prevent features that don't meet the definition from being delivered to the customer or user.

Grooming session

- ***Grooming sessions(Pre-review of the Feature Specifications*.)** This activity should introduce testing team at the very beginning of the feature implementation, allowing to perform documentation testing, improving documentation and feature with testers feedback.

Grooming session: Main goals.

- Testing team on their side should concentrate on specification review from testing perspective to:
- Gain clear understanding about the planned feature.
- Ensure the feature is testable, all possible business cases are discussed and covered in documentation, possible impact/dependencies on other features/3rd parties are taken into account, negative cases like error handling was discussed, etc.
- Ensure each involved person have same understanding of the feature and its purpose.
- Which additional testing types should be included. (like Security testing, Performance testing, UI/UX testing, etc.)
- Subjective feature estimation is provided in agreed way (if planned).

Grooming session: Estimation.

- Estimate of story
  - T-shirts estimation
  - Human Hours estimation (Tasks or sub-tasks)
  - Story point estimation (effort, uncertainties, complexity)

HomeWork 4 (“Real Story testing”)

Description:
As a User, I want to be able to Register into the system, using my user data. User data: Name/Surname, email and Password.
Upon adding correct data User should be registered.

Acceptance Criteria:
1. Fields “First Name” and “Surname” -
   - allowed: latin characters
   - filed length 2-15
   - Special characters are not allowed
2. E-mail field
   - Mandatory to use @ symbol
   - No Spaces
3. Password Field –
   - Minimum 6 characters, maximum 20,
   - Allowed: Latin characters, numbers, special symbols,
   - Spaces are not allowed

HW Task:
Review and analyze the story description from the Testing perspective, using different test design techniques. Try to find out missed requirements, uncertainties, mistakes. Create a set of questions to solve them.

Additional task:
- Create checklist to test the Form (~ 10 – 15 checks)