ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Software Testing

Chapter 4

TEST IMPLEMENTATION AND TEST REPORT

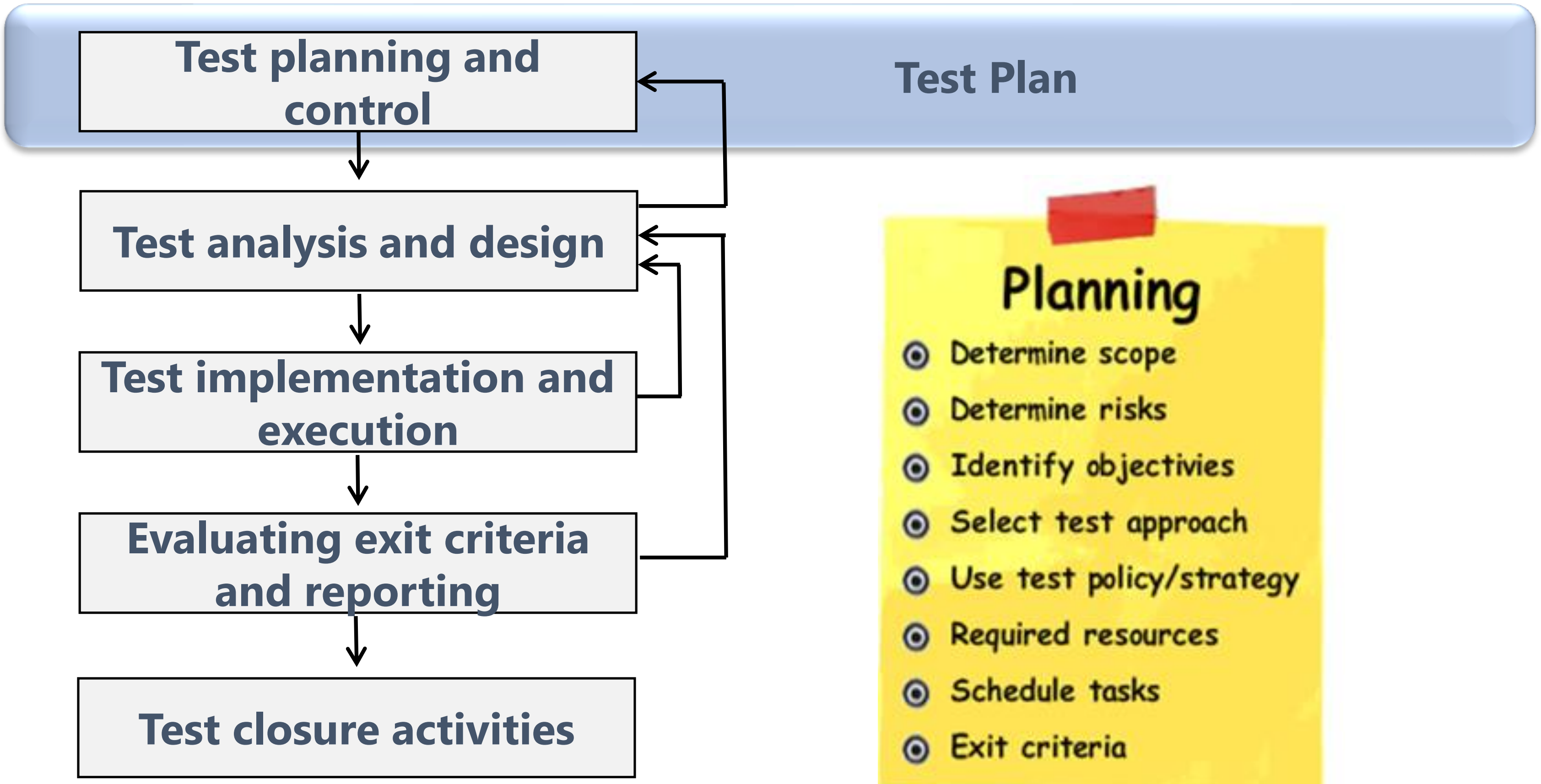
Session 1

TEST CASE DESIGN AND IMPLEMENTATION

Contents

1.Test plan

Test Plan it's a document describing the scope, approach, resources and schedule of intended test activities.

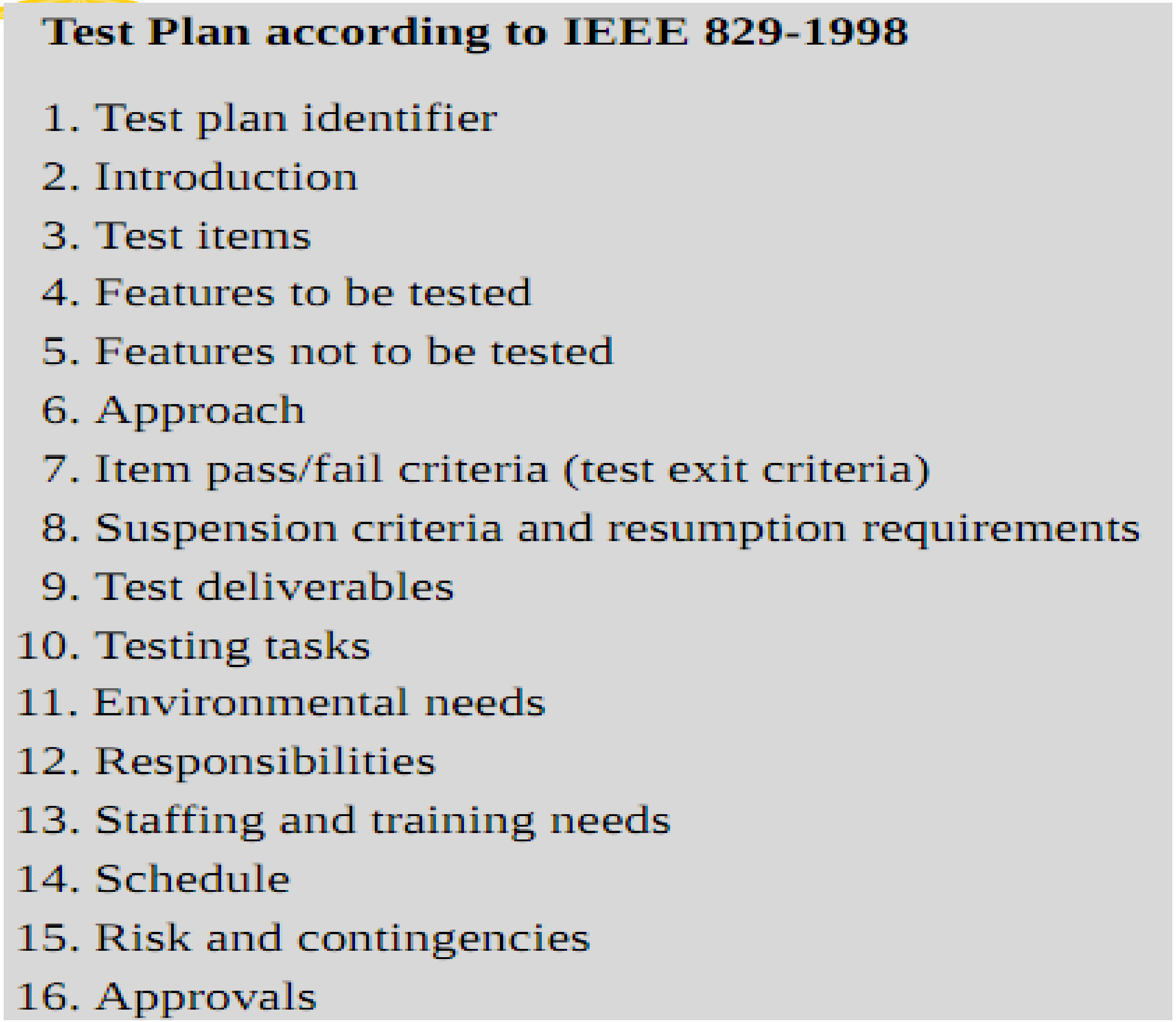


1.Test plan

❖Why do we write test plans?

- Writing a test plan guides our thinking. By using a template when writing test plans helps to organize, schedule and manage the testing effort
- The test planning process and the plan itself serve as a mean of communication with other members of the project team, testers, managers and other stakeholders.
- The test plan also helps us manage change.

Test plan template



Test Plan according to IEEE 829-1998

1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria (test exit criteria)
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing and training needs
14. Schedule
15. Risk and contingencies
16. Approvals

1. Test Plan

❖ **Why to identify test items/features?**

- To identify what is being tested.
- To determine the overall test effort.
- Used as the basis for test coverage.

❖ **Features to be tested:**

Identify all the software features and combinations of the software features that will be tested.

- Verifiable: they have an observable, measurable outcome.

1. Test Plan

❖ **Features not to be tested:**

Identify all the features and significant combinations of features that will not be tested along with the reasons.

- Not to be included in this release of the Software.
- Low risk, has been used before and is considered stable.
- Will be tested by the client

1. Test Plan

❖Test Entry and Exit Criteria

- Defining clear test entry and exit criteria is an important part of test planning.
- They define when testing can be started and stopped (totally or within a test level).

1. Test Plan

❖ **Entry Criteria:**

- Entry criteria are used to **determine when a given test activity can start**. This could include the planning, when test design and/or when test execution for each level of testing is ready to start.
- Examples:
  - ✓Test environment available and ready for use (it functions).
  - ✓Test tools installed in the environment are ready for use.
  - ✓Testable code is available.
  - ✓All test data is available and correct.
  - ✓All test design activity has completed.
- These criteria are preconditions for starting test execution. They prevent the test team from wasting time trying to run tests that are not ready.

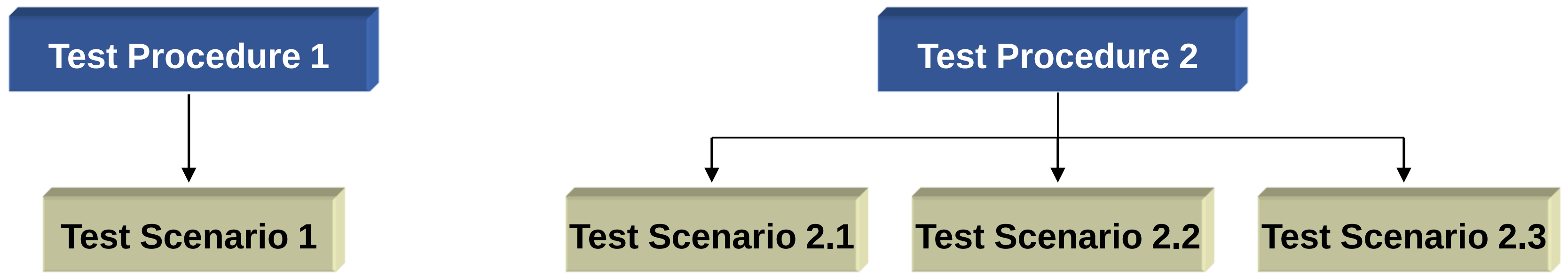
1. Test Plan

❖**Exit Criteria:**

- Exit criteria are used to determine when a given test activity has been completed or when it should stop. Exit criteria can be defined for all of the test activities, such as planning, specification and execution as a whole, or to a specific test level for test specification as well as execution.
- Examples:
  - ✓All tests planned have been run.
  - ✓A certain level of requirements coverage has been achieved
  - ✓No high-priority or severe defects are left outstanding.
  - ✓ All high-risk areas have been fully tested, with only minor residual risks left outstanding
  - ✓Cost – when the budget has been spent
  - ✓The schedule has been achieved

2. Test procedure

- Test procedure is document providing detailed instructions for preconditions for a test, data inputs necessary for the test, actions to be taken, expected results, and verification methods
- A Test Procedure could be testing of a system, a service, a function, an interface, or a component
- Each Test Procedure are 1 or more Test Scenarios



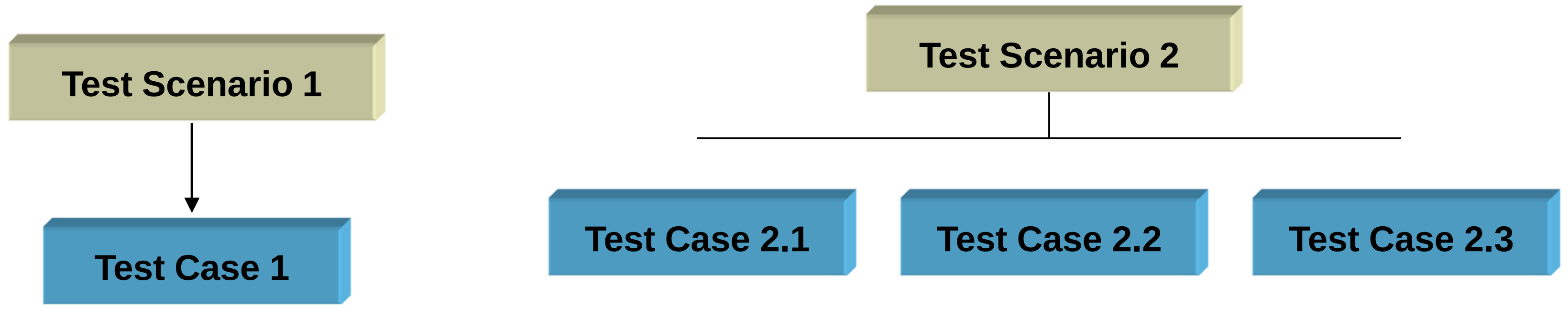
3. Test Scenario

❖ Test scenario (like a high-level test case):

- It is also known as Test Suite or less commonly Validation Suite.
- A test scenario is a collection of test cases that are intended to be used to test a software program
- Purpose is to show that it has some specified set of behaviors.
- It often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing.

3. Test Scenario

- Test Scenario provides one-line information about what to test.
- Each Test Scenario are 1 or more Test Cases



- Example:

Test Scenario	Test Case
User receives an error message when he enters invalid parameters in the login page	TC 1: User receives an error message when he enters valid user_id and invalid password.
	TC 2: User receives an error message when he enters invalid user_id and valid password
	TC 3: User receives an error message when he enters invalid user_id and invalid password

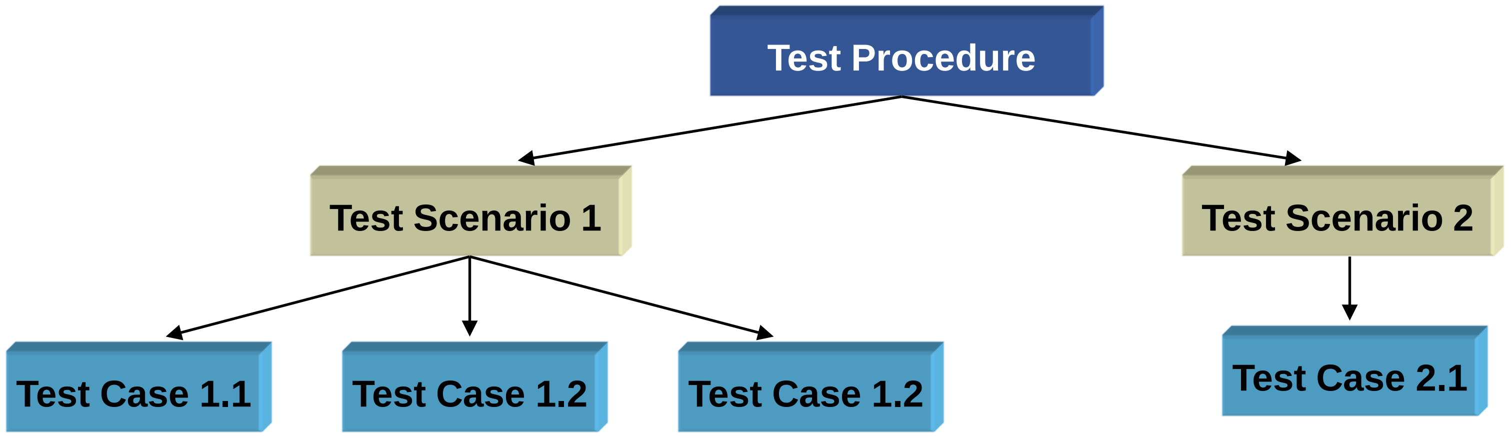
4. Test case

- Test case:
  - is a set of input values, execution preconditions, expected results and execution post conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.
  - Test Case is like a check list to make sure that all the requirements are covered in the testing
- Test case vs Test scenario

Test scenario: “What to be tested”
Test case is: “How to be tested”

4. Test case

❖Test Cases are derived from Test Scenarios that are identified in a Test Procedure



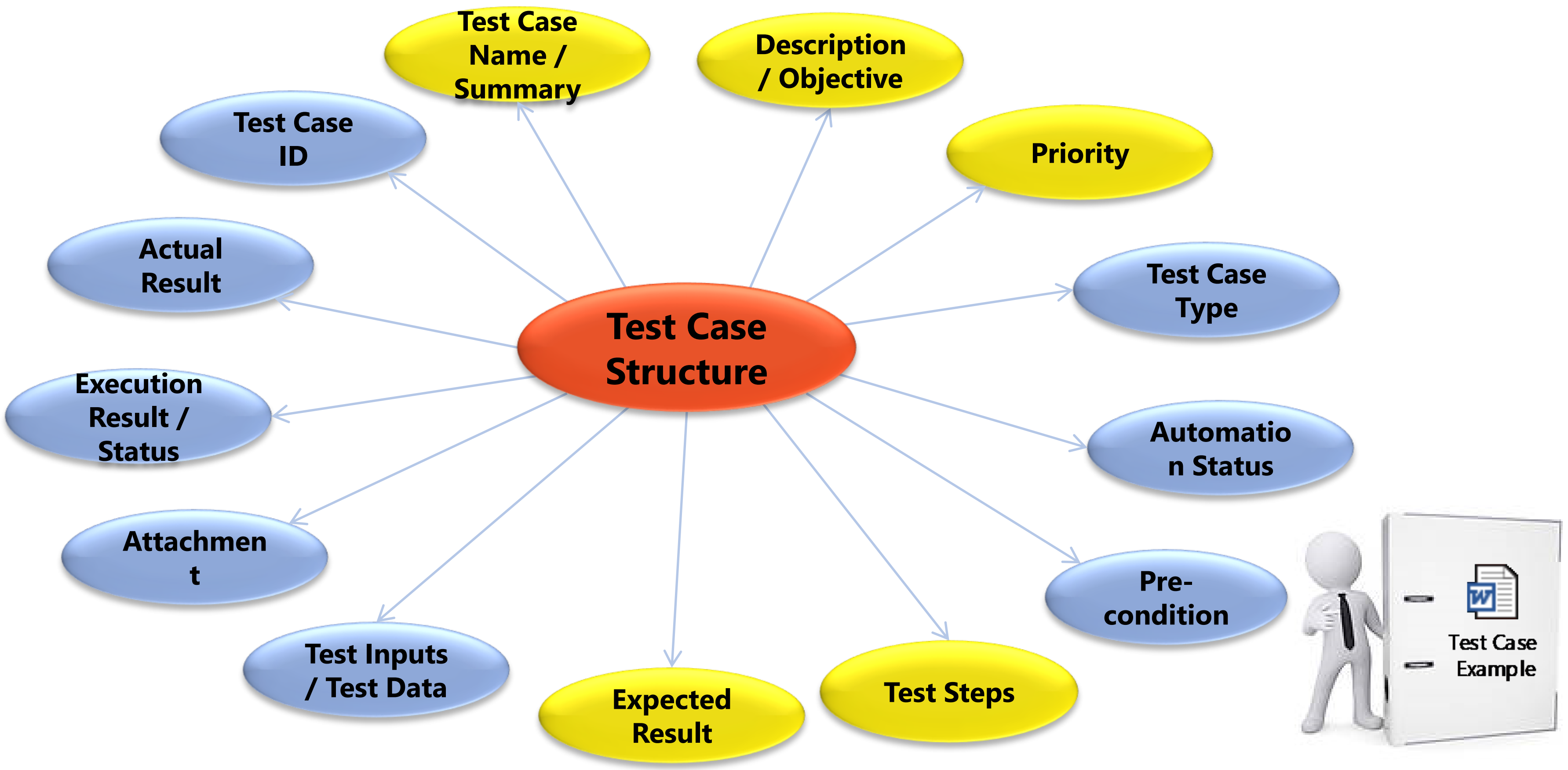
4. Test case

❖ The key purpose of a test case is to ensure if different features within an application are working as expected. It helps tester, validate if the software is free of defects and if it is working as per the expectations of the end users.

❖ Other benefits of test cases include:

- Test cases ensure good test coverage
- Help improve the quality of software
- Decreases the maintenance and software support costs
- Help verify that the software meets the end user requirements
- Allows the tester to think thoroughly and approach the tests from as many angles as possible
- Test cases are reusable for the future – anyone can reference them and execute the test.

4.1 Test Case Structure



**** Test Case structure might vary depending on particular project needs.***

4.2 Test Case Structure

❖Test case ID:
- Identification of the test case
- It should be unique across Test Case Specification
- Can consist of numbers or/and letters

  Example: *1, 2, 3, etc; UR.001, UR.002, etc*

❖Test Case Name:
- Short name of test case which briefly indicates what will be verified

❖Objective:
- Describes the functionality/actions that test case validates/does
- It should be detailed enough to understand purpose of test case

❖Test Case Type/Specification Areas
- Reflects the type of test case depending on what kind of testing is covered by particular test case
- Example: GUI, Functional, System, Performance, etc.

4.2 Test Case Structure

❖Pre-Condition (optional):

- Defines conditions that should be met before test case can be executed
- Usually pre-condition field lists data/actions which should exist/be done in system and links to appropriate test cases/test functions which can setup required pre-conditions

❖Test Data/Test Inputs

- Lists data which is used while test case execution
- Can be presented in this field directly or via link to attached files
- Data should be accurate!

❖Attachment

- May contain files which can be used while test case execution

4.2 Test Case Structure

❖Priority

- Reflects the relative importance of the test case taking into consideration different aspects
- Can be presented by words or numbers

  Example: *High, Medium, Low, etc;*

  Major, Minor, Trivial, etc;

  1, 2, 3 (where 1-the most important, and 3-the least important), etc

❖Automation Status

- Indicates whether test cases is candidate for automation taking into consideration different aspects or not
- Also indicates whether test cases is automated already or not
- Example:

  ✓*Candidate* – test case is recommended for automation, but it is not automated yet

  ✓*Not Candidate* – test case is not recommended to be automated and should be run manually

  ✓*Automated* – test case is automated and during the next execution can be run automatically etc.

4.2 Test Case Structure

❖Actual Result

- Shows the actual output of the system. This field is used when actual behavior of the system doesn't meet expected results of test case

❖Expected Results:

- Shows how the system must react based on the test steps
- Expected results should be mentioned only for test case objective!
- "**Verify**", "**Correctly**", "**Successfully**" words are for bidden for expected results! Exact behavior of the system, which is going be verified, should be mentioned
- Example:

<u>Incorrect:</u> Verify "TestUser" user is created

-> It is not understandable how to verify it

<u>Correct:</u> "TestUser" user appears in the list of users

4.2 Test Case Structure

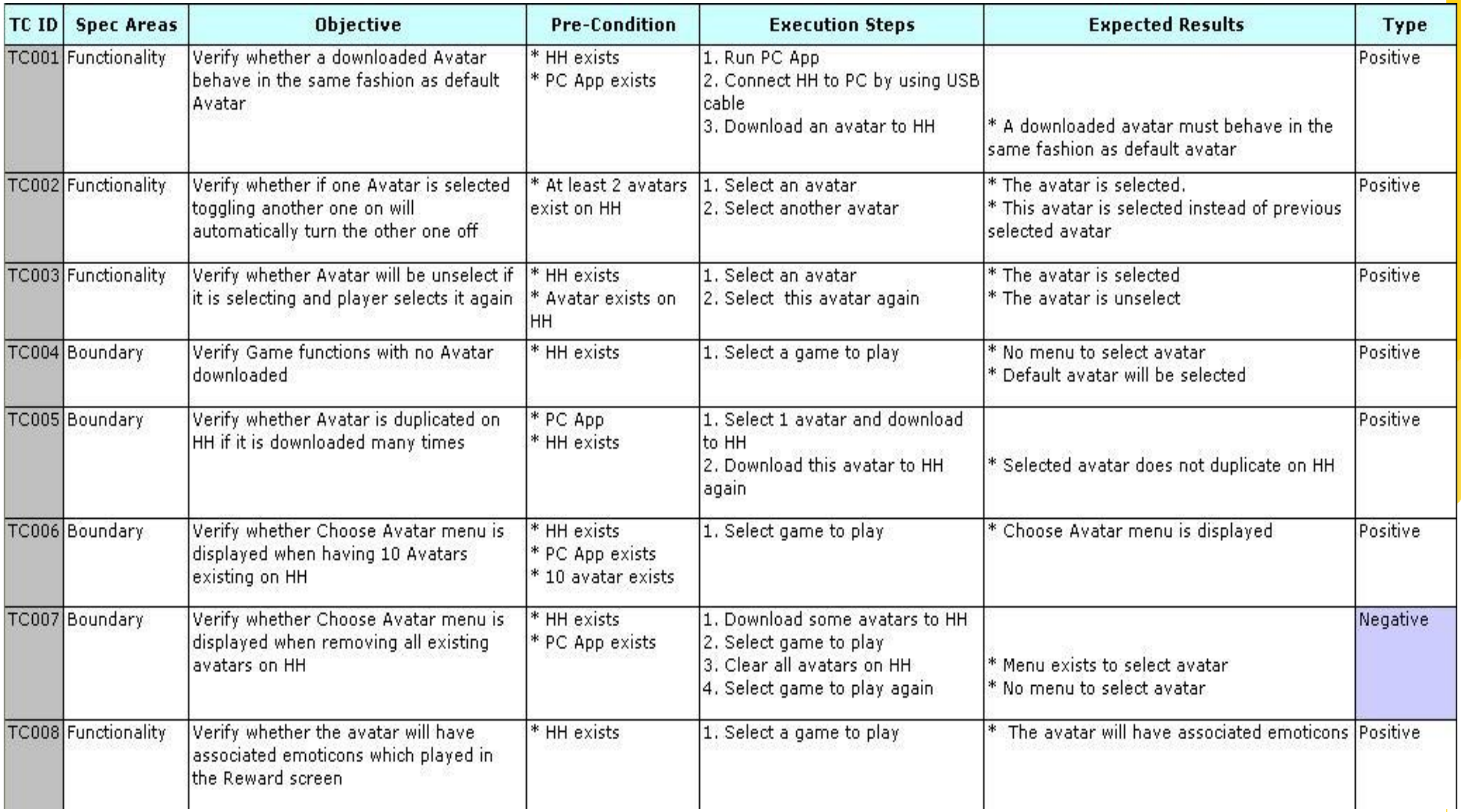
❖Execution Result / Status

- Shows the result of test case execution to indicate whether behavior of the system meets expected results of test case or not

Example:

- **Pass** – expected results of test case and behavior of the system match
- **Fail** – expected results of test case and behavior of the system do not match
- **Blocked** – test case was unable to be executed due to some reasons (e.g. blocker issue, etc)
- **Skipped** – test case was untested since it wasn't planned to be executed this time, etc.

4.2 Test Case Structure - Example



TC ID	Spec Areas	Objective	Pre-Condition	Execution Steps	Expected Results	Type
TC001	Functionality	Verify whether a downloaded Avatar behave in the same fashion as default Avatar	* HH exists * PC App exists	1. Run PC App 2. Connect HH to PC by using USB cable 3. Download an avatar to HH	* A downloaded avatar must behave in the same fashion as default avatar	Positive
TC002	Functionality	Verify whether if one Avatar is selected toggling another one on will automatically turn the other one off	* At least 2 avatars exist on HH	1. Select an avatar 2. Select another avatar	* The avatar is selected. * This avatar is selected instead of previous selected avatar	Positive
TC003	Functionality	Verify whether Avatar will be unselect if it is selecting and player selects it again	* HH exists * Avatar exists on HH	1. Select an avatar 2. Select this avatar again	* The avatar is selected * The avatar is unselect	Positive
TC004	Boundary	Verify Game functions with no Avatar downloaded	* HH exists	1. Select a game to play	* No menu to select avatar * Default avatar will be selected	Positive
TC005	Boundary	Verify whether Avatar is duplicated on HH if it is downloaded many times	* PC App * HH exists	1. Select 1 avatar and download to HH 2. Download this avatar to HH again	* Selected avatar does not duplicate on HH	Positive
TC006	Boundary	Verify whether Choose Avatar menu is displayed when having 10 Avatars existing on HH	* HH exists * PC App exists * 10 avatar exists	1. Select game to play	* Choose Avatar menu is displayed	Positive
TC007	Boundary	Verify whether Choose Avatar menu is displayed when removing all existing avatars on HH	* HH exists * PC App exists	1. Download some avatars to HH 2. Select game to play 3. Clear all avatars on HH 4. Select game to play again	* Menu exists to select avatar * No menu to select avatar	Negative
TC008	Functionality	Verify whether the avatar will have associated emoticons which played in the Reward screen	* HH exists	1. Select a game to play	* The avatar will have associated emoticons	Positive

4.3 Test Case: Priority

❖ Software testers may prioritize their test cases in order:
- to **reduce the cost of regression testing**, so that those which are more important, by some measure (e.g. if the time limits means…), are run earlier in the regression testing process
- to **increase a test suite's rate of fault detection**, thus allowing developers to fix severe faults earlier in software development process

❖ Prioritizing test cases can be done by considering:
- high critical functionalities, which are the parts of Core test cases or new functionality, implemented in build/release
- modules containing more bugs, more complex or more dependent areas which are highly accessed by the customer/end users
- positive test cases
- risk analysis

4.4 Test Case: Automation Status

It is impossible to automate all testing!

❖Good Candidates for Automation are:

- Repetitive test cases that run for multiple builds
- Test cases that run on several different configurations (hardware/OS)
- Frequently used functionality that introduces high risk conditions
- Test cases that tend to cause human error
- Test cases that require multiple data sets
- Test cases that are impossible to perform manually
- Test cases that take a lot of effort and time when manual testing

4.4 Test Case: Automation Status

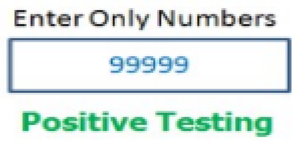
❖ Bad Candidates for Automation are:
- If automation efforts are few times higher than manual execution
- Test cases that are only performed a few times
- Some test steps cannot be automated

4.5 Type of test case

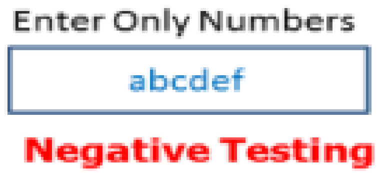
- There are 3 main categories:
  1. Positive or “happy path” test case
  2. Negative test case
  3. Combination test case
- Test Case Types comes under these categories:
  - Functional Test Cases
  - Performance Test Cases
  - Security Test Cases
  - Integration Test Cases
  - Database Test Cases
  - Usability Test Cases
  - Acceptance Test Cases

4.5 Type of test case

- ❖ **Positive test case**: The most obvious set are Positive or "happy path" test cases. These test cases are designed to return what are expected according to the requirement



- ❖ **Negative test case**: The test is designed to test the robustness of the software and the system to unexpected or varying inputs, states, or conditions.



- ❖ **Combination test case**: The test is designed to determine whether the response of the product is still expected when doing a series of tests that follow each other

4.6 Test case development

1. How to analyze Requirement
2. How to design Test Case ID
3. How to analyze Objective
4. How to create Steps
5. How to analyze Expected Result

4.6 Test case development

1. How to analyze Requirement

- Read documentation carefully to understand system, application or function
- Pick out sentences that describing specifications from a system, application or function in technical document of project

2. How to design Test Case ID

- Test Case ID must be unique in test case suite
- Test Case ID should be involved the abbreviation of specification area and figures
- Example:
  - ✓Test cases ID relate to Tutorials should be TUT001, TUT002,…
  - ✓Test cases ID relate to Power Management should be PM001, PM002,…

4.6 Test case development

3. How to analyze Objective (test case)

Each requirement can have one or more cases to verify. These cases can be in 3 groups:

- Positive case
- Negative case
- Combination case

4.6 Test case development

3. How to analyze Objective (test case)

- **Positive Case:**
  - ✓Create positive cases by following right instruction of requirements to verify expectation of system, application, function,…
  - ✓Each requirement have at least a positive test case

4.6 Test case development

3. How to analyze Objective (test case)

➢ **Negative Case**:

✓ Create negative cases by deliberately entering bad data or using other techniques to try to break the functionality.

✓ Top **10** negative test cases should consider when designing test effort

1. **Embedded Single Quote** - Most SQL based database systems have issues when users store information that contain single quotes
2. **Required Data Entry** - Write test cases to ensure it forces you to enter data in the field
3. **Field Type Test** - Write test cases to ensure it forces you to enter data in the correct format based on the field type
4. **Field Size Test** - Write test cases to ensure that you can only enter the specified number of characters
5. **Numeric Bounds Test** - For numeric fields, it is important to test for lower and upper bounds

4.6 Test case development

3. How to analyze Objective (test case)

➢ **Negative Case**:

6. **Numeric Limits Test** - Write test cases to ensure that it does not get an numeric overflow error.
7. **Date Bounds Test** - Write test cases to test for lower and upper bounds of date
8. **Date Validity** - Write test cases to ensure that invalid dates are not allowed and test cases should also check for leap years
9. **Web Session Testing** - Many web applications rely on the browser session to keep track of the person logged in, settings for the application, etc. Create test cases to launch web pages within the application without first logging in
10. **Performance Changes** - should include test cases that compare the prior release performance statistics to the current release. This can aid in identifying potential performance problems

4.6 Test case development

3. How to analyze Objective (test case)

➢**Combination Case**:

✓Think about complex test scenarios that can affect to a system, functions,…

✓Complex test scenarios are aggregated use cases that combine one or more of the user functions and conditions to create practical usage scenarios

Ex: an HR manager will log on to the system, check existing employees, add a new employee, verify the addition, and then log out of the system

✓Combination test cases are derived complex test scenarios

What's happen if I do … while …?
What's happen if I do like this, next I do… and then I do…?

4.6 Test case development

4.How to create Steps

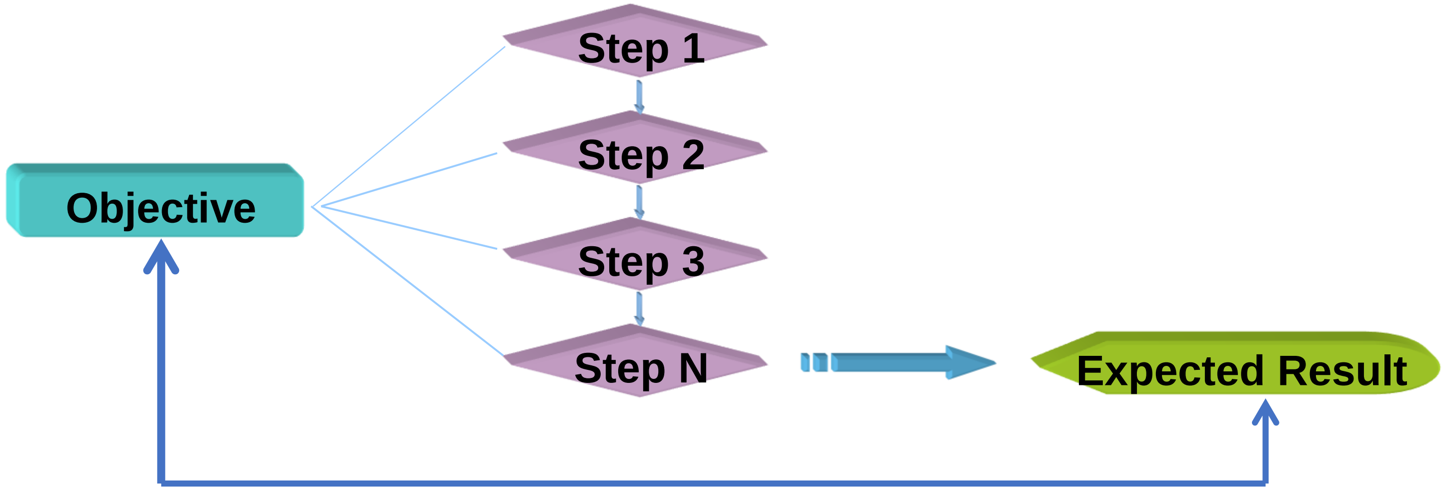
Upon objectives of test case we can design 2 types of Steps:

- ➢Detailed steps:
  - ✓Often apply for atomic or small test cases
  - ✓Steps are described detailed and queued up
  - ✓Steps must relate to objective of test case and clearly
- ➢Main steps:
  - ✓Often apply for combination or general test cases
  - ✓Steps are described in general (show up main steps or important steps only)
  - ✓Steps must relate to objective of test case and clearly

4.6 Test case development

5.How to analyse Expected Result

- ✓Expected result must describe what you expect the function or system to do
- ✓Expected result is corresponding to the step that caused
- ✓Expected result must be clearly and relate to the Objective of the test case

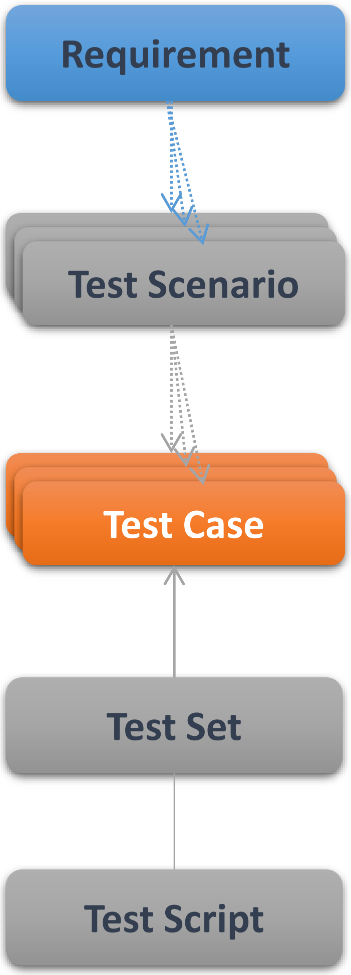


4.6 Test case development

❖ Test case development:

- Boundary testing
- Equivalence classes
- Decision tables
- State transition diagrams
- Risk Analysis

5. Test case implementation - Flow



Any statement describing a functionality that is expected of the system

Any condition that could possibly happen in production

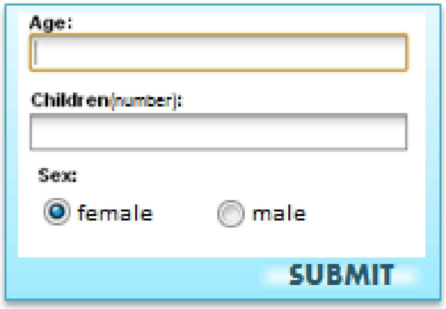
Pre-conditions + Input -> Output + Post Conditions

A group of similar test cases that require the same steps to be executed

A set of steps, manual or automated, to execute a set of similar test cases

5. Test case implementation - example

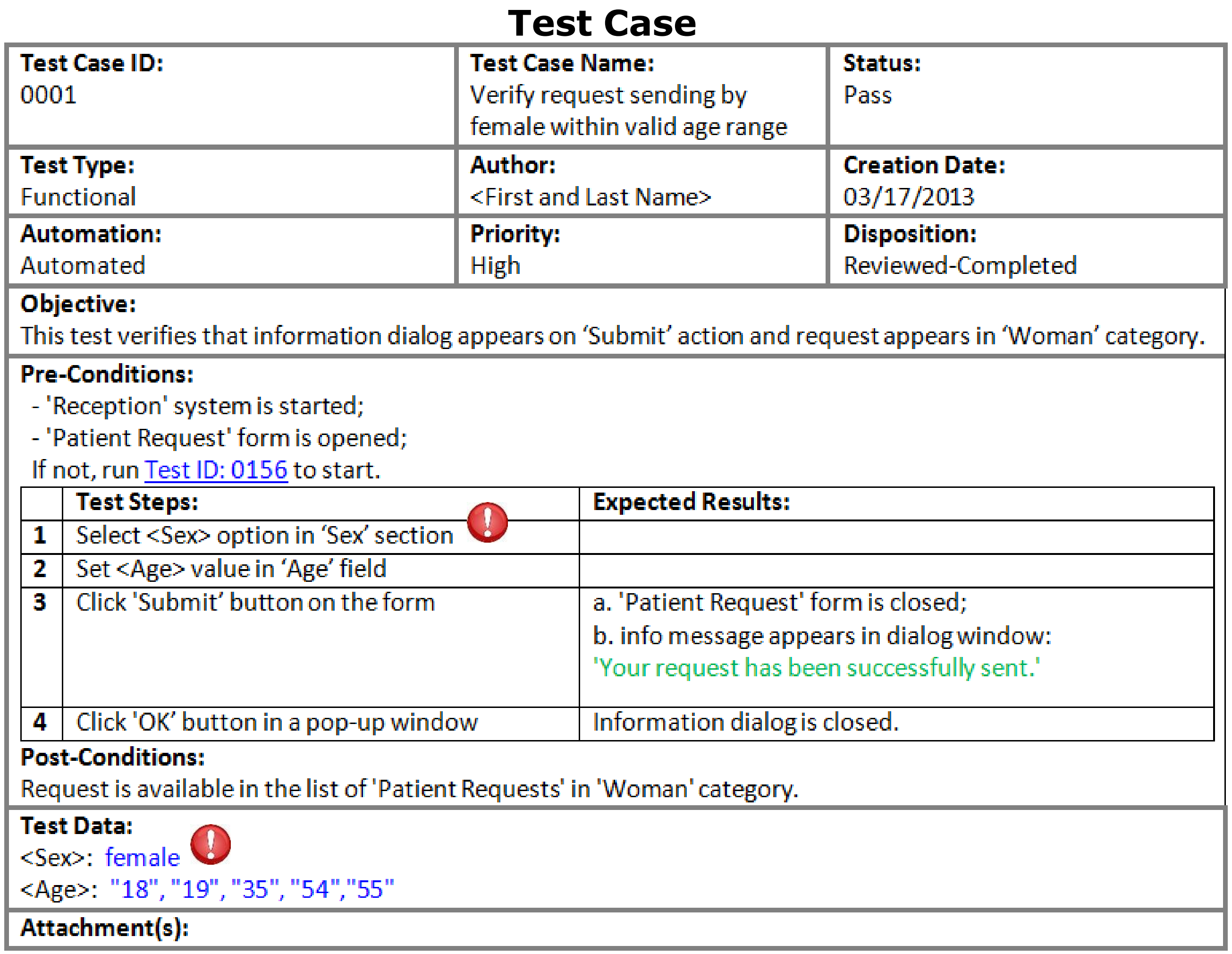
Requirement : **18-55 patient should be able to post a request, which should be processed within 1 hour. If patient is 45-55 female with more then 1 child – request should be processed within 30 minutes.**



Test Design

Requirement:	Scenario:	Test Case:	Objective:
Patient should be able to post a request.	Verify request can be posted by patient within valid age range	Verify request sending by female within valid age range	This test verifies that information dialog appears on 'Submit' action and request appears in 'Woman' category.
		Verify request sending by male within valid age range	This test verifies that information dialog appears on 'Submit' action and request appears in 'Man' category.
		Verify request sending by female within invalid age range	This test verifies that warning dialog appears on 'Submit' action. Request is not created in 'Woman' category.
		Verify request sending by male within invalid age range	This test verifies that warning dialog appears on 'Submit' action. Request is not created in 'Man' category.
		Verify request sending by patient with age which contains non-acceptable chars	This test verifies that error dialog appears on 'Submit' action.
		Verify request sending by patient without age defined	This test verifies that error dialog appears on 'Submit' action.
	Verify prerogatives for request posted by female	*<Test Case Name>*	*<Test Case Objective>*
	Verify prerogatives for request posted by patient with children	*<Test Case Name>*	*<Test Case Objective>*

5. Test case implementation - example



Test Case

Test Case ID: 0001	**Test Case Name:** Verify request sending by female within valid age range	**Status:** Pass
Test Type: Functional	**Author:** <First and Last Name>	**Creation Date:** 03/17/2013
Automation: Automated	**Priority:** High	**Disposition:** Reviewed-Completed

Objective:
This test verifies that information dialog appears on 'Submit' action and request appears in 'Woman' category.

Pre-Conditions:
- 'Reception' system is started;
- 'Patient Request' form is opened;

If not, run Test ID: 0156 to start.

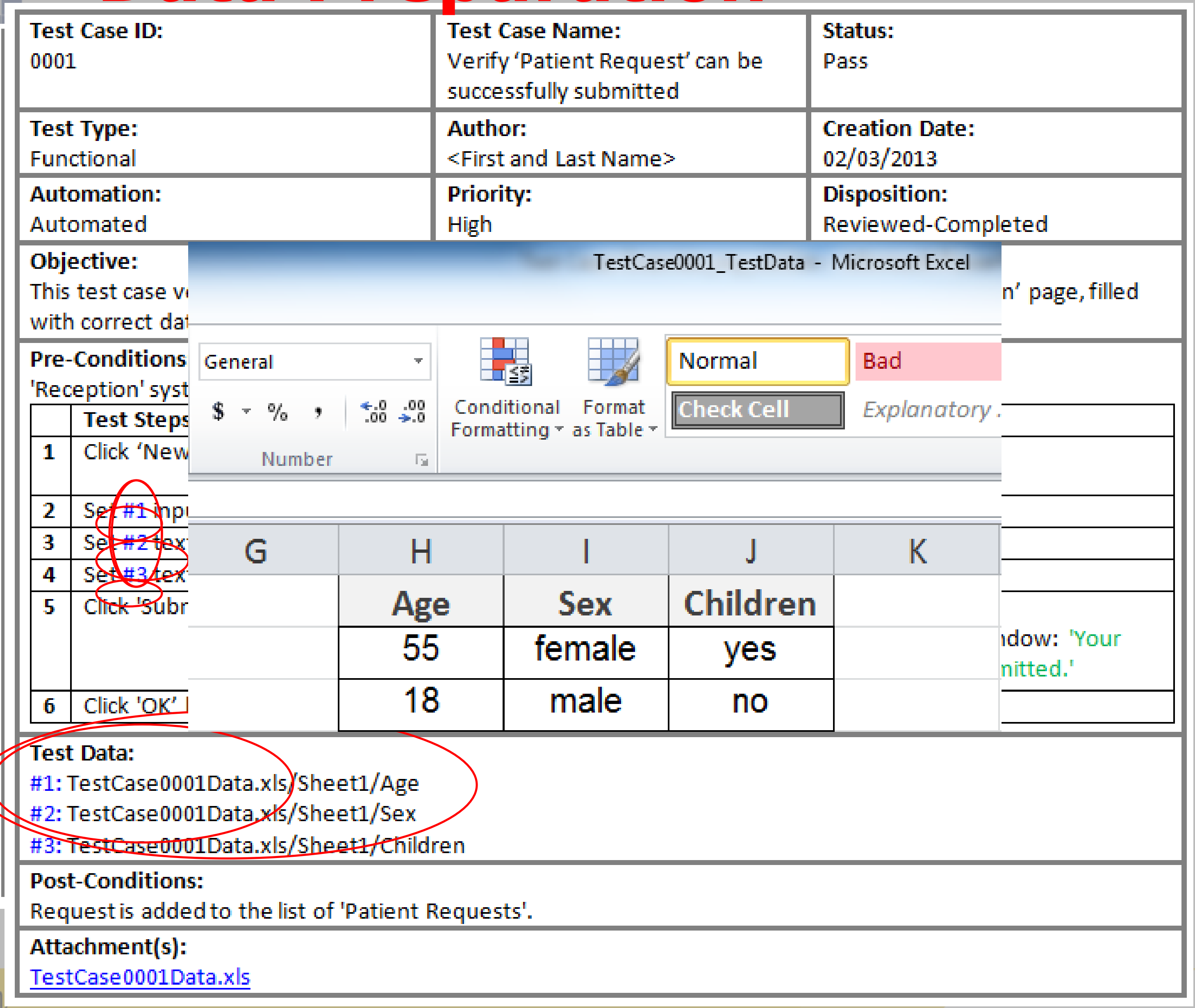
	Test Steps:	**Expected Results:**
1	Select <Sex> option in 'Sex' section	
2	Set <Age> value in 'Age' field	
3	Click 'Submit' button on the form	a. 'Patient Request' form is closed; b. info message appears in dialog window: 'Your request has been successfully sent.'
4	Click 'OK' button in a pop-up window	Information dialog is closed.

Post-Conditions:
Request is available in the list of 'Patient Requests' in 'Woman' category.

Test Data:
<Sex>: female
<Age>: "18", "19", "35", "54","55"

Attachment(s):

Test case implementation - Test Data Preparation



Test Case ID: 0001	**Test Case Name:** Verify 'Patient Request' can be successfully submitted	**Status:** Pass
Test Type: Functional	**Author:** <First and Last Name>	**Creation Date:** 02/03/2013
Automation: Automated	**Priority:** High	**Disposition:** Reviewed-Completed

Objective:
This test case v ... n' page, filled
with correct dat

Pre-Conditions
'Reception' syst

	Test Steps	
1	Click 'New	
2	Set #1 inpu	
3	Set #2 tex	
4	Set #3 tex	
5	Click 'Subr	ndow: 'Your nitted.'
6	Click 'OK'	

TestCase0001_TestData - Microsoft Excel

G	H	I	J	K
	Age	Sex	Children	
	55	female	yes	
	18	male	no	

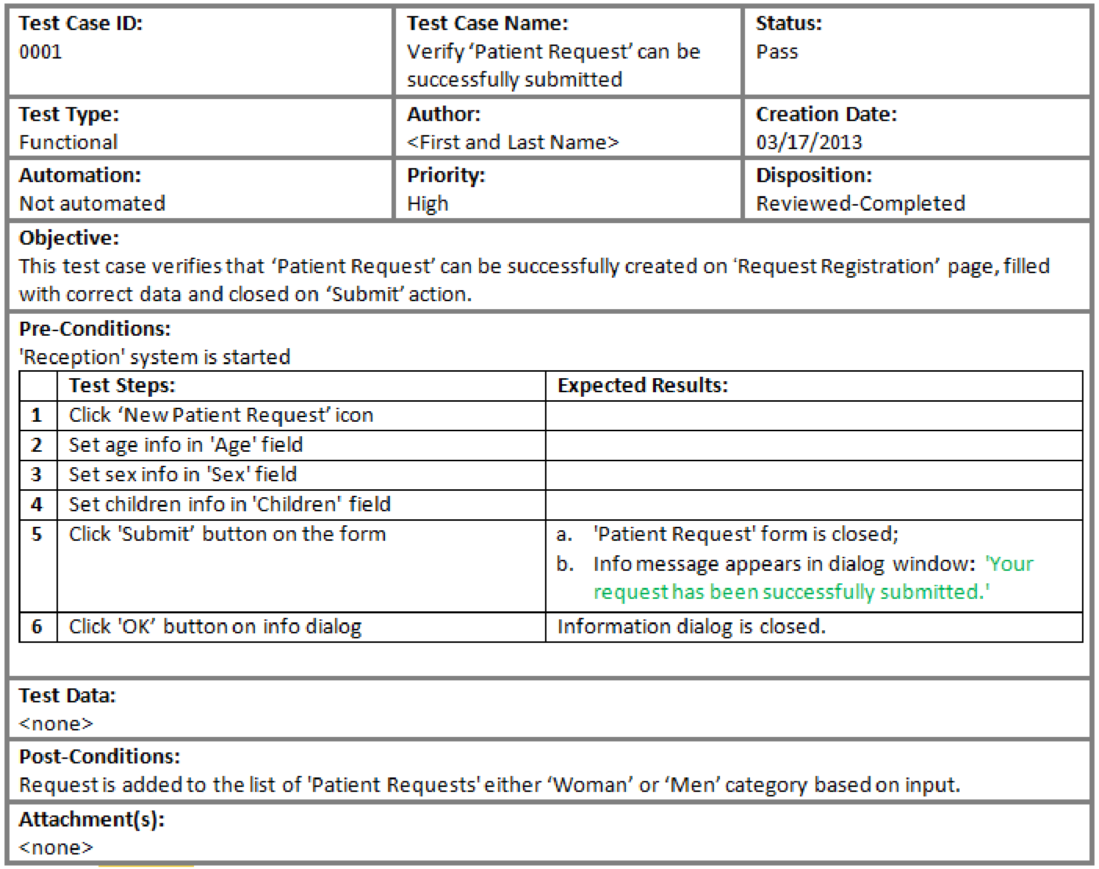
Test Data:
#1: TestCase0001Data.xls/Sheet1/Age
#2: TestCase0001Data.xls/Sheet1/Sex
#3: TestCase0001Data.xls/Sheet1/Children

Post-Conditions:
Request is added to the list of 'Patient Requests'.

Attachment(s):
TestCase0001Data.xls

5. Test case implementation - Test Data Preparation

Test Data on Tester's disposal ☺



Test Case ID: 0001	**Test Case Name:** Verify 'Patient Request' can be successfully submitted	**Status:** Pass
Test Type: Functional	**Author:** <First and Last Name>	**Creation Date:** 03/17/2013
Automation: Not automated	**Priority:** High	**Disposition:** Reviewed-Completed

Objective:
This test case verifies that 'Patient Request' can be successfully created on 'Request Registration' page, filled with correct data and closed on 'Submit' action.

Pre-Conditions:
'Reception' system is started

	Test Steps:	**Expected Results:**
1	Click 'New Patient Request' icon	
2	Set age info in 'Age' field	
3	Set sex info in 'Sex' field	
4	Set children info in 'Children' field	
5	Click 'Submit' button on the form	a. 'Patient Request' form is closed; b. Info message appears in dialog window: 'Your request has been successfully submitted.'
6	Click 'OK' button on info dialog	Information dialog is closed.

Test Data:
<none>

Post-Conditions:
Request is added to the list of 'Patient Requests' either 'Woman' or 'Men' category based on input.

Attachment(s):
<none>

Pros
- **time saving during test cases designing;**
- **time saving for experienced tester in specific area during test cases execution (not always, since some test cases require complex inputs, queries, etc.);**
- **important bugs can be found.**

Cons
- **time consuming for non-experienced tester in specific area during test cases execution;**
- **hard to entail issue due to chaotic inputs.**

5. Test Case implementation-Result Tracking

1. Update Test Case
2. Execute Test Case

5. Test Case Result Tracking – Update TC

❖Update test cases: When the game design or the documentations are changed, you must update the test case suite base on the newest documentations or base on the game design on current build if the client request:

- Add new test cases
  - ✓ *If you find new cases can apply to test, please add these cases as new test cases into test case suite*
- Edit test cases to match the newest requirements
  - ✓ *If the Steps (or Specification Area) to run test case has been changed in the newest documentations, the test case has to also update the Steps Execution (or Specification Area) again*
- Delete test cases no longer suitable
  - ✓ *If the requirement for feature has been changed in the newest documentations, the test case relate to that requirement has to be removed out*

5. Test Case Result Tracking –Execute TC

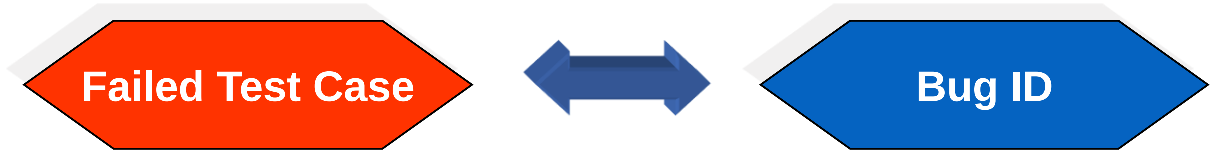
- Execute Test Case: Execute test cases to get result of testing
  - Read Objective of test case carefully to understand what will be verified here
  - Read Pre-Condition to prepare the need stuffs in advance (The stuffs really need to execute test case)
  - Execute application (game, web application,…) by following step by step in Execution Steps of test case
  - Compare observed result with expected result of test case
  - Update result of test case into:
    - ✓Status column
    - ✓Bug ID column

5. Test Case Result Tracking –Execute TC

- Execute Test Case (cont)
  - Update Status of test case
    - If observed result **matches** the expected result:
      - Set "Pass" to Status column for the test case
    - If observed result **does not match** the expected result:
      - Set "Fail" to Status column for the test case
    - If there is a reason that **prevent** you from executing the test case (cannot do a step, function is not implemented, a bug blocks testing the test case,…):
      - Set "Pending" to Status column for the test case
      - Note the reason why the test case is pending

5. Test Case Result Tracking –Execute TC

- Execute Test Case (cont)
  - ✓ Update Bug ID for test case
    - If observed result **matches** the expected result:
      - Bug ID for the test case must be removed out of "Bug ID" column if it has been noted already before
    - If observed result **does not match** the expected result:
      - Submit a bug to bug database and note the bug ID of submitted bug into "Bug ID" column if it has not noted yet before



ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN

Vietnam - Korea University of Information and Communication Technology

Thank You !

http://vku.udn.vn/