Test Cases

Software Testing 1

Lector: Aleksandar Karamfilov
Skype: sanders_kar
Email: alex@pragmatic.bg
Facebook: http://facebook.com/a.karamfilov

PRAGMATIC

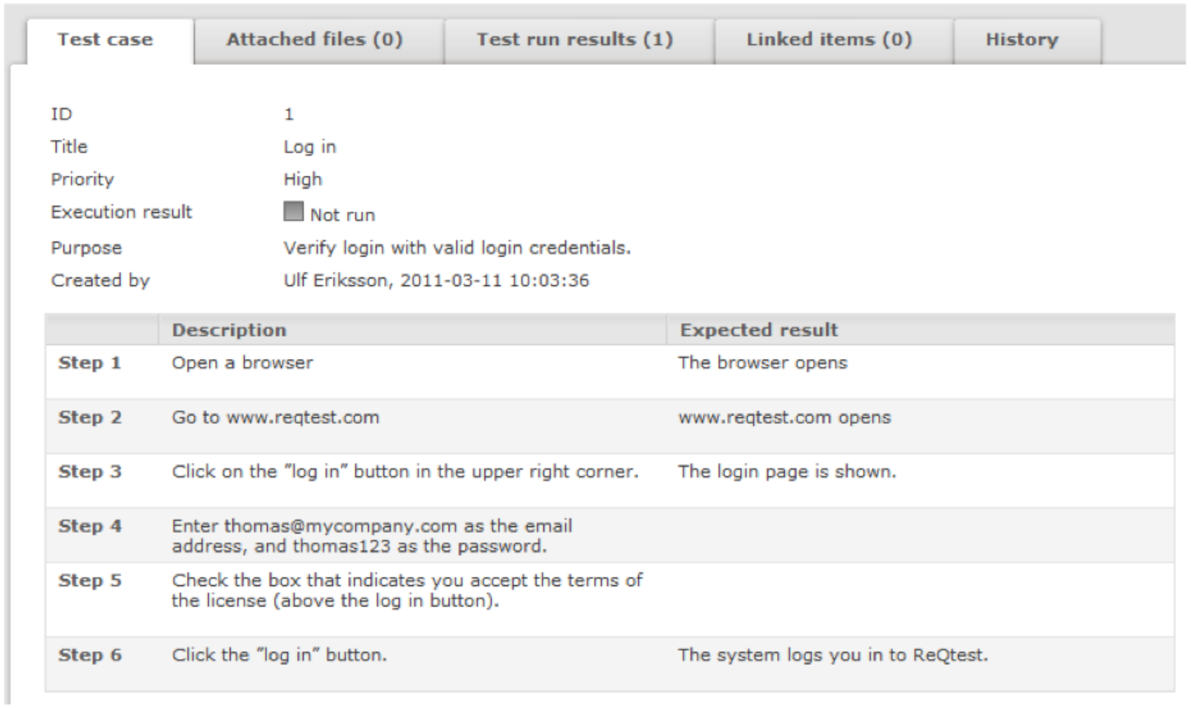
www.pragmatic.bg

Agenda

- Test case
- The value of test case
- Anatomy of a test case
- Good practices in writing test cases
- Common test case mistakes
- Test case management
- Test case workflow
- Test case metrics

Test case

“Specific set of input values given to an element of software with an intent to expose a certain of defect”



ID	1
Title	Log in
Priority	High
Execution result	▨ Not run
Purpose	Verify login with valid login credentials.
Created by	Ulf Eriksson, 2011-03-11 10:03:36

	Description	Expected result
Step 1	Open a browser	The browser opens
Step 2	Go to www.reqtest.com	www.reqtest.com opens
Step 3	Click on the ”log in” button in the upper right corner.	The login page is shown.
Step 4	Enter thomas@mycompany.com as the email address, and thomas123 as the password.	
Step 5	Check the box that indicates you accept the terms of the license (above the log in button).	
Step 6	Click the ”log in” button.	The system logs you in to ReQtest.

Test scope

A test case describes specific actions to take against a particular software component and the expected outcomes. The component can be as small as an application programming interface(API), a control in user interface(UI), or a port handler of a device driver. It can be as large as software system with multiple computers and applications working together.

The value of a test case

- Historical reference
- Tracking test progress
- Repeatability

Historical reference

Test case last far beyond product release. Sustained engineering as well as owners of future product versions often need to refer to test cases to understand what was tested and how it was tested.

Tracking test progress

By documenting test cases, you can track several additional attributes, such as the number of test cases run, the number of tests that have passed or failed, and total number of test cases per feature.

Repeatability

A well-documented test case can be run by anyone at any time. This is equally applicable to both automated and manual test cases. The ability to repeat the same test accurately is essential to generate reproduction steps or detect regression.

Drawbacks of a test case

- Documentation time
- Test cases get out of date as features change
- Difficult to assume knowledge of reader

Documentation time

If the time to document the test case takes longer than running the test does, it might not make sense to document the test case. This is frequently true in situations where the test case needs to execute only a few times in single environment.

Test cases get out of date

The time needed to document a test case can quickly grow out of control if the underlying feature changes often. It does not always make sense to document test cases for feature areas that change often. This scenario is frequently true when attempting to write test cases to verify user interface components.

Difficult to assume knowledge of reader

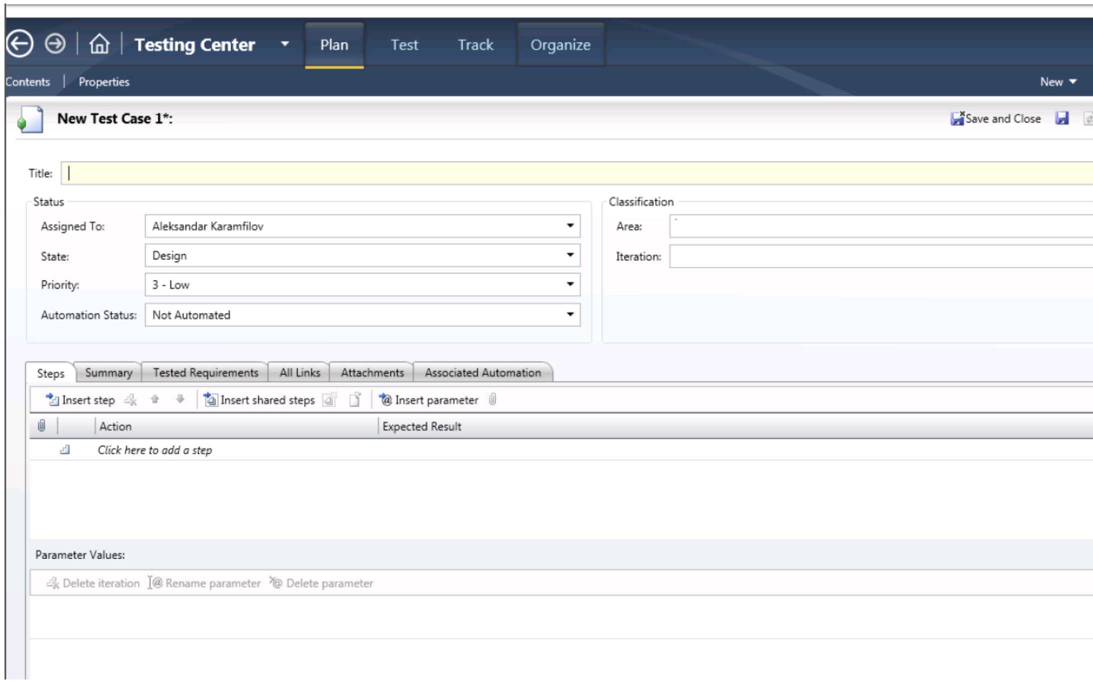
People who are extremely familiar with the feature under test often are those who are writing the test cases. A mistake commonly made by these people is the use of jargon or acronyms in the test case that might be understandable to those who run the test case in the future. When this occurs, the test case is no longer accurately repeatable.

Anatomy of a test case

- Title/Name/Summary
- Purpose
- Preconditions
- Description of inputs/actions
- Test data
- Definition of expected results
- Attachments

Classification attributes

- Priority
- Component/Area/Requirement Tested
- State
- Automation status
- Iteration/Sprint
- Frequency



New Test Case 1*:

Title:

Status

Assigned To:	Aleksandar Karamfilov
State:	Design
Priority:	3 - Low
Automation Status:	Not Automated

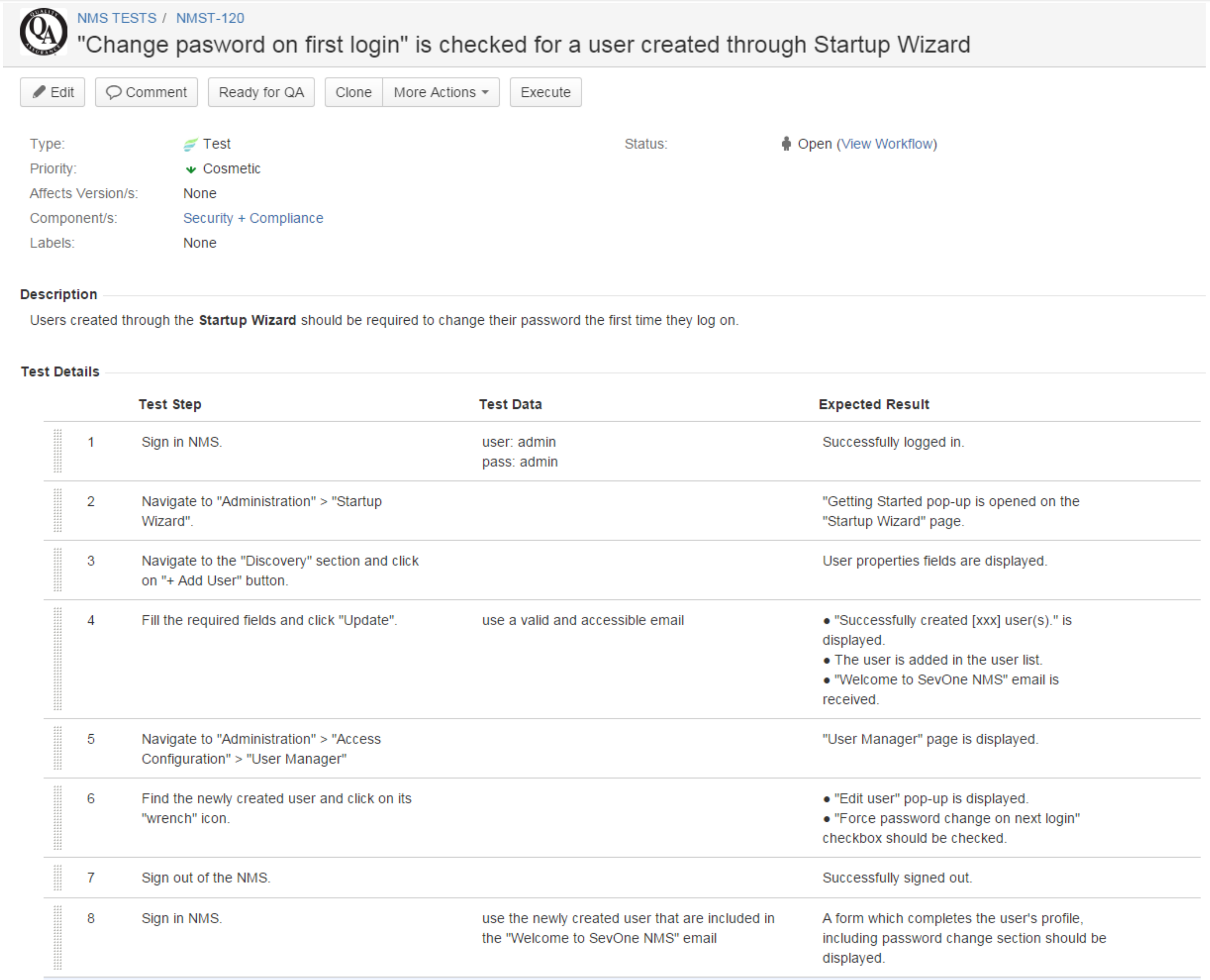
Classification

Area:	
Iteration:	

Action	Expected Result
Click here to add a step	

Parameter Values:

Example – Manual



NMS TESTS / NMST-120

"Change pasword on first login" is checked for a user created through Startup Wizard

Edit | Comment | Ready for QA | Clone | More Actions | Execute

Type:	Test	Status:	Open (View Workflow)
Priority:	Cosmetic		
Affects Version/s:	None		
Component/s:	Security + Compliance		
Labels:	None		

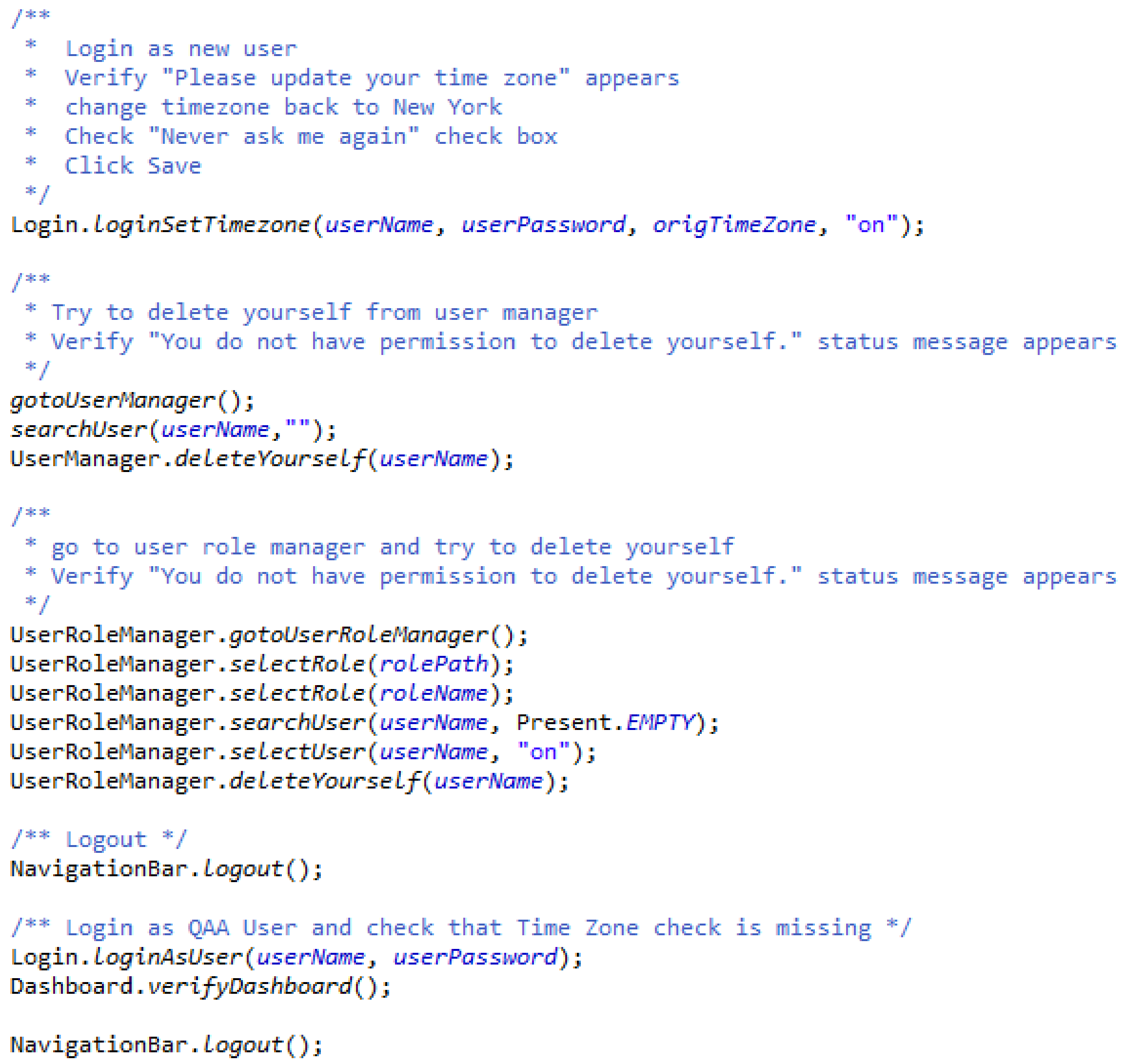
Description

Users created through the **Startup Wizard** should be required to change their password the first time they log on.

Test Details

	Test Step	Test Data	Expected Result
1	Sign in NMS.	user: admin pass: admin	Successfully logged in.
2	Navigate to "Administration" > "Startup Wizard".		"Getting Started pop-up is opened on the "Startup Wizard" page.
3	Navigate to the "Discovery" section and click on "+ Add User" button.		User properties fields are displayed.
4	Fill the required fields and click "Update".	use a valid and accessible email	● "Successfully created [xxx] user(s)." is displayed. ● The user is added in the user list. ● "Welcome to SevOne NMS" email is received.
5	Navigate to "Administration" > "Access Configuration" > "User Manager"		"User Manager" page is displayed.
6	Find the newly created user and click on its "wrench" icon.		● "Edit user" pop-up is displayed. ● "Force password change on next login" checkbox should be checked.
7	Sign out of the NMS.		Successfully signed out.
8	Sign in NMS.	use the newly created user that are included in the "Welcome to SevOne NMS" email	A form which completes the user's profile, including password change section should be displayed.

Example – Automated



```
/**
 *  Login as new user
 *  Verify "Please update your time zone" appears
 *  change timezone back to New York
 *  Check "Never ask me again" check box
 *  Click Save
 */
Login.loginSetTimezone(userName, userPassword, origTimeZone, "on");

/**
 * Try to delete yourself from user manager
 * Verify "You do not have permission to delete yourself." status message appears
 */
gotoUserManager();
searchUser(userName,"");
UserManager.deleteYourself(userName);

/**
 * go to user role manager and try to delete yourself
 * Verify "You do not have permission to delete yourself." status message appears
 */
UserRoleManager.gotoUserRoleManager();
UserRoleManager.selectRole(rolePath);
UserRoleManager.selectRole(roleName);
UserRoleManager.searchUser(userName, Present.EMPTY);
UserRoleManager.selectUser(userName, "on");
UserRoleManager.deleteYourself(userName);

/** Logout */
NavigationBar.logout();

/** Login as QAA User and check that Time Zone check is missing */
Login.loginAsUser(userName, userPassword);
Dashboard.verifyDashboard();

NavigationBar.logout();
```

Common test case mistakes

- Dependency between test cases
- Unclear pass/fail criteria
- Missing steps
- Too verbose
- Too much jargon
- Wrong test priority
- Wrong component assignment
- Very long test procedure
- Bulk tests

Poor test case example

Title: Check encrypted message

Steps:

1. Get a v3 certificate
2. Send an encrypted message
3. Create a message with 0 RSA/DH lockboxes, send it
4. Create a message with RSA/DSA signature, send it
5. Create a message with an RSA/DSA signature, send it

Verify:

- Everything works for all messages
- Certificate server produces a good certificate
- Clear text and blobs come through properly

Good Test Case Features

- Simple to execute (fast is good too)
- Repeatable
- Make failure obvious
- Distinguish different kinds of failures
- Have reasonable chance of exposing previously unknown defects
- Easy to maintain
- Test exactly on thing (ASF)

Good test case example

Title: Clear text with v3 certificate
Purpose: Verify that an encrypted clear text message using a v3 certificate can be opened and read by a recipient
Steps:

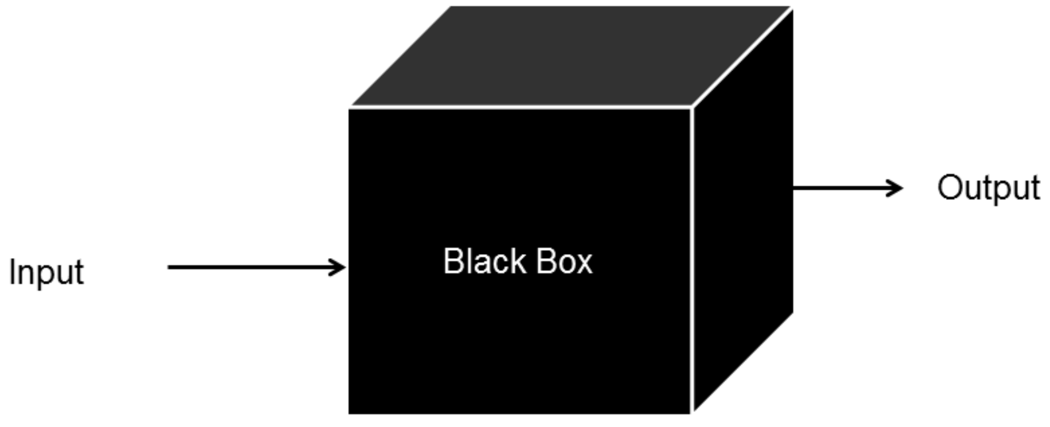
1. User1 starts Microsoft Office Outlook.
2. On the Tools menu, click Options, and then click Security tab.
3. Ensure that the Send clear text signed message when sending signed messages option is selected.
4. Ensure that v3 certificates are properly set up
5. Click OK.
6. Open a new message, give it a subject “foo” and body “bar”.
7. Address and send the message to User2
8. On another computer, start Outlook with User2 account.

Verify:

- The encrypted message has been received in the inbox and can be opened by User2.
- The subject and body text of the message are “foo” and “bar”.

Keep in mind!

- Not all test cases are possible
- As a black-box testers we are limited on the field we can cover
- Interaction only with UI hides complexity of the product



Internal behavior of the code is unknown

Positive Test Cases

Does software do what it is supposed to?

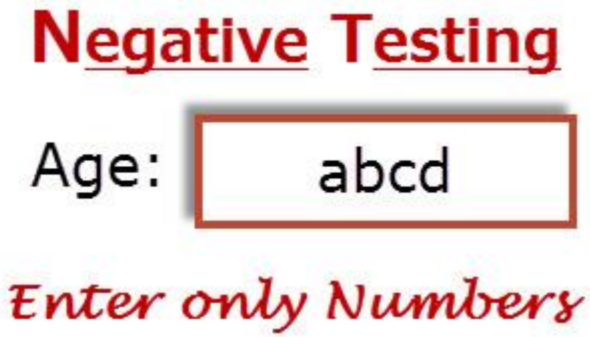
- Single test case can test for as many “positives” as practical!

Positive Testing
Age:
999
Enter only Numbers

Negative Test Cases

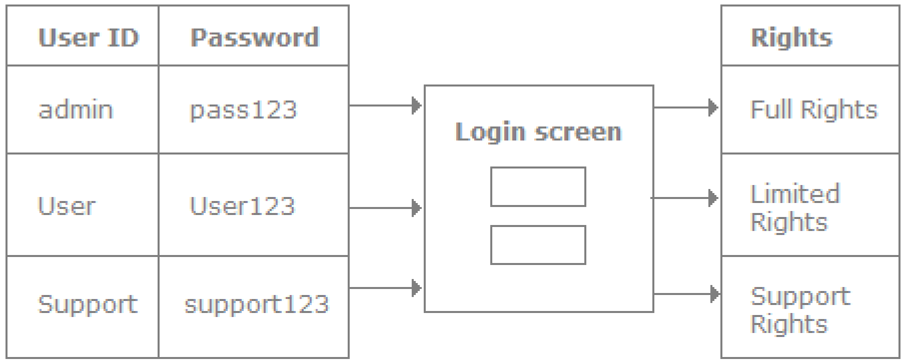
Whether the program does what is not supposed to do?

- Start testing only one “negative” at a time!



Data-Driven testing

“Data-driven testing: a scripting technique that stores test input and expected results in a table or spreadsheet, so that a single control script can execute all of tests in the table.”



Data-Driven testing

- Test data can be stored in one or more data sources, such as external files(xls, csv, xml,…) or databases.
- Data-driven testing is used generally for applications requiring fixed set of actions to be performed, but with a lot of permutations and combinations of various parameters which form the test cases.

How to Create Data-Driven Tests

1. Create a test script with a set of constant test data.
2. Replace constant test data with some variables.
3. Create multiple sets of test data in a data storage(e.g. Excel, CSV, XML,…)
4. Assign to variables the values read from the data storage.

Test Sets & Test Suites

A Test Set is a collection of test cases with common characteristics:

- Same element being tested
- Same feature being tested
- Same test strategy
- Same test environment

A Test Suite is a collection of all test cases/sets for specific component or system.

Test Coverage

Measures how comprehensive a test set is!

- How much of some reference specification is exercised by that test set?
- Are any parts of the reference specification not tested by that test set?
- Normally presented in percentage(%)

Test sets that don’t cover the reference spec. are probably inadequate (incomplete).

Goal

The key to efficient and effective testing is to achieve some desired flavour of coverage with the fewest number of test cases!

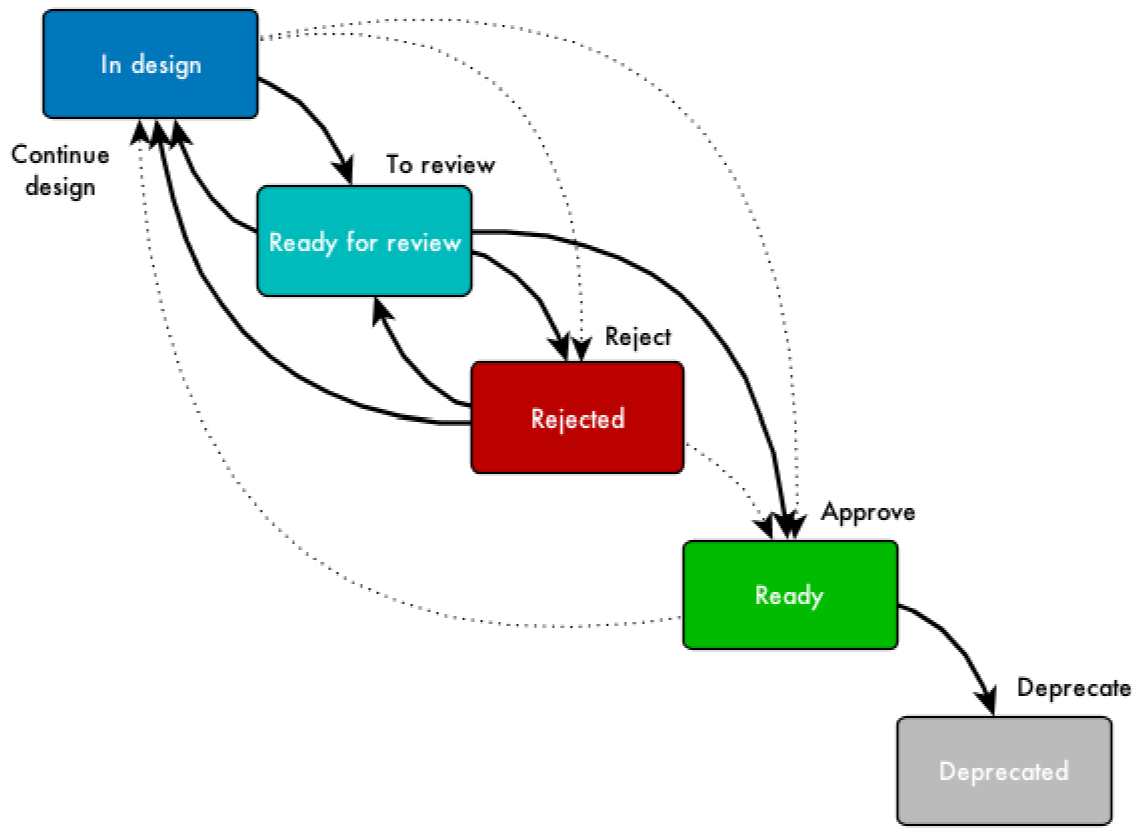
Test Case Management

- Collects it all in one place
- Makes it easy to abstract certain procedures and reuse them
- Makes it easy to track test statuses(Pass/Fail/Not Run/Blocked)
- Better traceability to requirements

Test Case Management (continued)

- Makes it easy to filter/sort/copy/delete tests
- Makes it easy to maintain test cases
- Can switch preconditions/setup faster
- Makes it more visible and consistent
- More secure

Design workflow



Test case metrics

- Pass rate
- Number of passes / failures
- Total number of test cases / total planned
- Automation ratio
- Number of tests by type
- Number or percentage of tests finding bugs

Questions