07 Test Automation and Risk

PowerPoint by Wanida Khamprapai

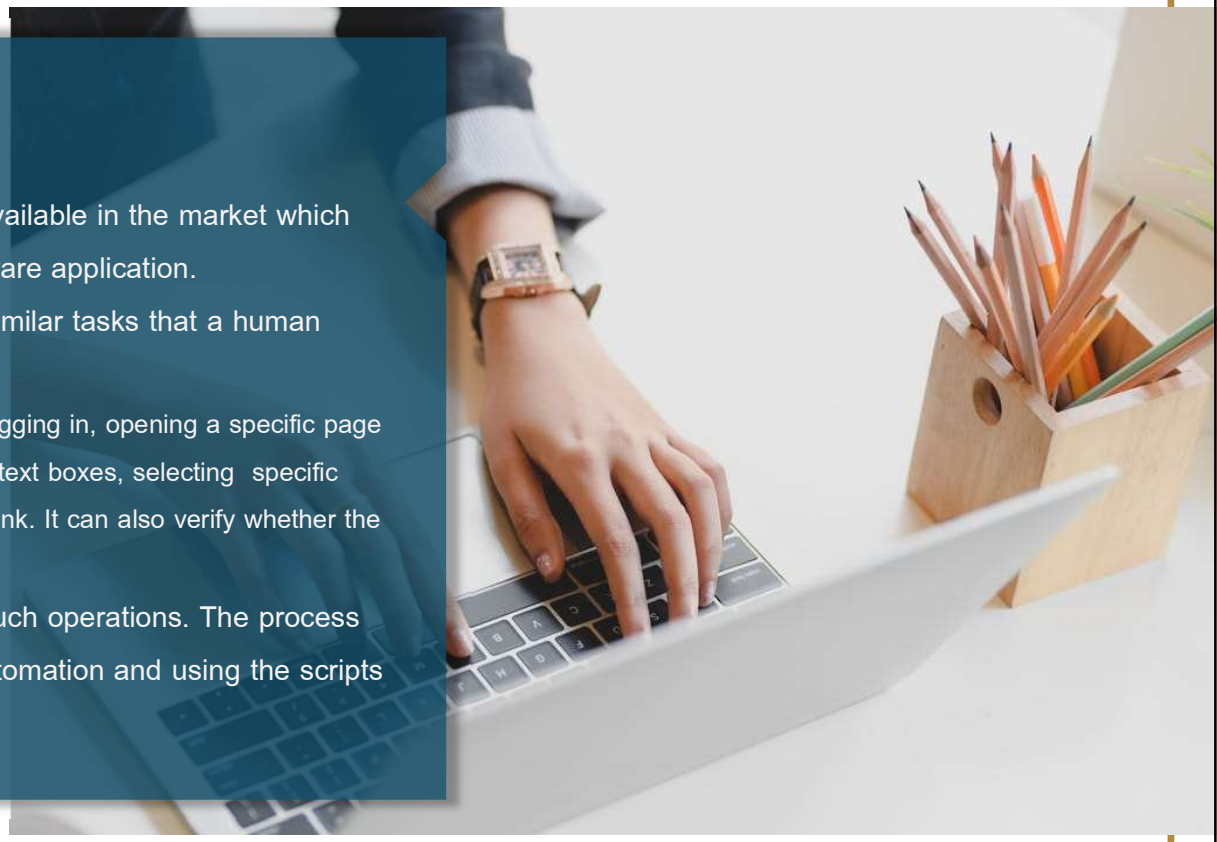
Software test optimization

In earlier chapters, we learned software testing types, levels, techniques, and project execution processes (STLC phases). Software testing does require a good amount of effort to ensure complete coverage and accuracy. Most software projects have constraints in terms of time and effort for multiple reasons

- Overall budget for the software project is not enough to take up the cost of all the testing types, activities, and coverage.
- Even if a sufficient budget is provided, development activities many times consume more than expected budget and time requiring a reduction in time and effort for testing
- The quality of the software is worse than expected requiring more testing cycles but the deadline cannot be extended It is hence important to identify various means to reduce the effort without compromising much on the quality.
- Test Automation
- Risk Based Testing (RBT)

Test automation

- There are ready-to-use software testing products/tools available in the market which allows the creation of various scripts for testing the software application.
- Most of these tools have the capability to perform very similar tasks that a human being is performing to do manual testing.
  - It involves opening a browser, opening an application, logging in, opening a specific page (E.g. Ticket Reservation) and entering values in various text boxes, selecting specific options from available options, or clicking a button or a link. It can also verify whether the application is responding correctly as expected or not.
- We need to create scripts using tools so that it can do such operations. The process of creating scripts using these tools is known as test automation and using the scripts to test the application is known as automated testing.



Test automation

- Test automation is the use of special software (separate from the software being tested) to set up test preconditions, control the execution of tests, and then the comparison of actual outcomes to predicted outcomes and report.
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.

Test automation tools

There are many commercial and open source tools available in the market and newer tools continue to be developed. Different tools are available for different testing types and objectives. So, there are tools available for functional testing, performance, and load testing, security testing, and user interface testing.

Example, test automation tools

- Commercial tools:
  - HP Quicktest professional
  - Borland SilkTest
  - Compuware TestPartner
  - IBM Rational Functional Tester
- Open source tools:
  - Katalon Studio
  - Selenium
  - Watir FIT/Fitness
- Open source unit testing tools:
  - jUnit
  - nUnit

How does the tool work?

In order to imitate human beings for executing some steps, the automation tool must understand various objects first. Everything which is visible in an application such as a label, textbox, button, image, checkbox, radio button, dropdown, hyperlink, etc. is known as an "Object". Automation is nothing but identifying such objects and taking action on that object. The action could include entering text, clicking, moving the mouse over, or changing any attribute of the object.

Example: The automated script for login function

- Open the browser and maximize it
- Enter a specific URL in the browser to open the application's login page
- Identify textbox for User ID and enter specific user ID say 'Admin'
- Identify textbox for Password and enter specific password say 'admin12'
- Identify the submit button and Click it

How does the tool work?

Please find below a small portion of the Login script developed in the Katalon Studio tool.

Example: The automated script for login function (Cont.)

1. WebUI.openBrowser(‘’)
2. WebUI.maximizeWindow()
3. WebUI.navigateToUrl(GlobalVariable.a_url)
4. WebUI.setText(findTestObject('Registration object/Page_THIMS Log in/ input_User_ID'), 'admin’)
5. WebUI.setText(findTestObject('Registration object/Page_THIMS Log in/ input_User_Passwd'), 'admin12').
6. WebUI.click(findTestObject('Registration object/Page_THIMS Log in/ button_Sign in'))
7. WebUI.verifyElementPresent(findTestObject('test_login/Page_T HIMS Login/p_User Name or Password is Incorrect'), 0)

In order to do such scripting, one has to learn the scripting language supported by the tool. The tool/supported scripting language provides many such commands to take various actions (very similar to the ones taken by human beings) and also for verification.

How does the tool work?

Additionally, tools also provide many features as described below:

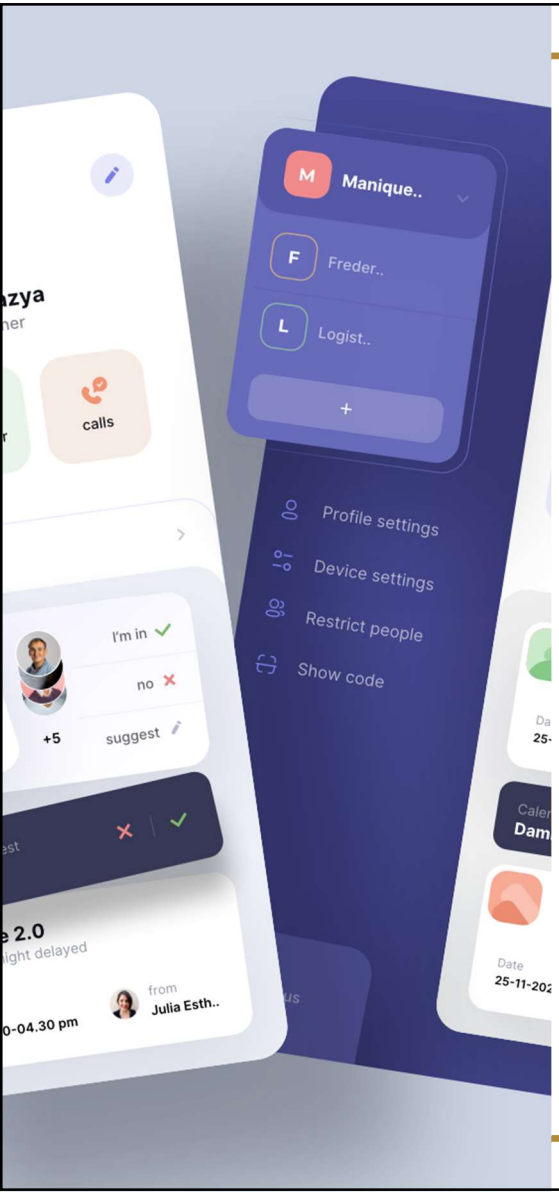
Record and playback features:

It allows users to interactively record user actions and replay them back any number of times. You have to activate recording in the tool and then run the application under test.

Programming capabilities

In addition to the recording feature, the tools also provide a programming feature that allows you to modify the recorded scripts or create a new script for some or more of the following aspects

- **Setting up of test preconditions**.
- In corporate various **programming constructs** such as If-else, Loop.
- **Parameterize** the inputs and take data from the data source instead of hardcoding input values in the script itself, you can use variables whose value can be read from excel or database tables.
- **Incorporate verification points**. The tool can capture various HTML elements and their properties that can be used for comparison with the expected results.
- **Report test results** without human assistance.



How does the tool work?

Using various commands one can

- Test the existence of UI elements based on their HTML tags,
- Check for specific content,
- Check for broken links and input fields,
- Selection list options,
- Submit forms, and table data

Different tools provide many other features as described below:

- Support multiple browsers and platforms.
- Minimal script development effort (excel-like input instead of scripting command).
- Supports data driven testing (use different data sheets/excel or database tables to store data and pick the data while execution). Develop and use data files with a set of variables and corresponding values for data-driven testing.
- Allows reusable functions (also referred to as keywords).
- Clubbing of test scripts to make test suits for end-to-end scenario testing.
- Exception handling and reporting.
- Detailed (Test case wise) and summary wise HTML reports which can be customized.
- Set up global variables which can be used for some standard details.

Automation process

There are three main steps involved as part of test automation

Creating test scripts

- Recording a session on a site or application.
- Modify the tests as per the requirements.
- Insert checkpoints into the test.
- Broaden the scope of the test by replacing fixed values with parameters.

Running tests

- Run your test to check the site or application.
- Run a test to debug the test.

Analyzing test results

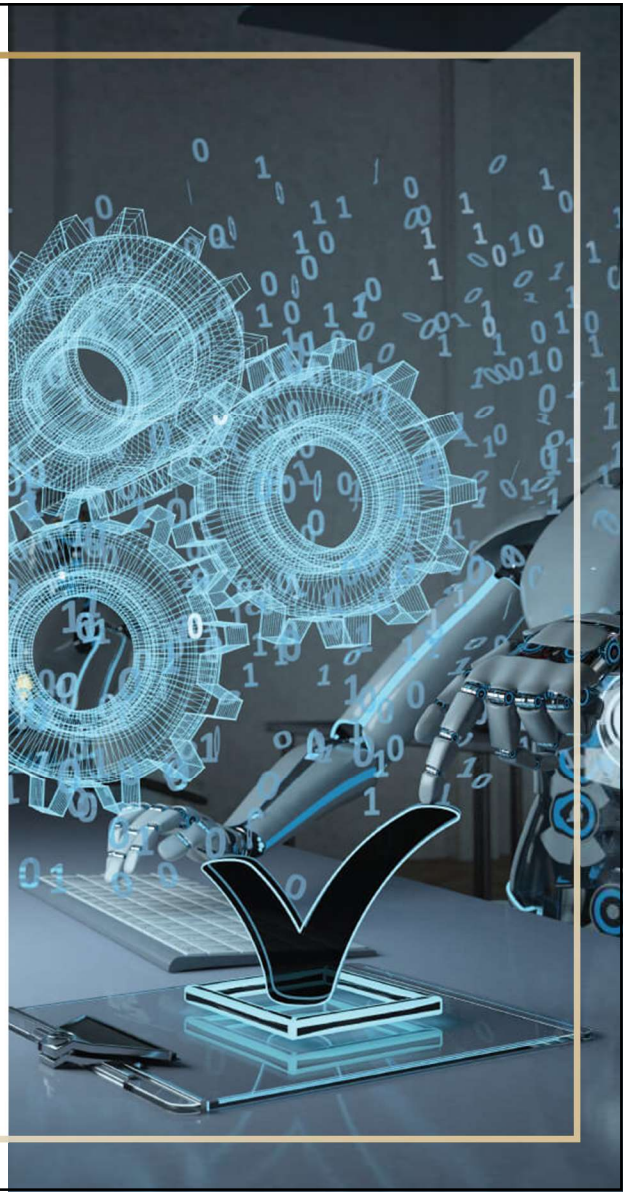
- View the test results in the test results windows.
- Analyze defects detected during the test run.

Test case selection for automation

It is also important to note that developing testing scripts using these tools also requires effort and the scripts developed should be also be correct. Automation hence should be used carefully after considering its pros and cons of the same.

Test automation is useful for

- Tests that need to be run frequently for every build of the application (sanity check, regression test). There is no point in automating tests that are going to be executed only once or twice because test automation (creating scripts) itself takes time.
- Tests that use multiple data values for the same actions (data-driven tests). Tests that are executed multiple times with different datasets can also be good candidates for automation.
- Testing that would be difficult to perform manually or tests requiring a great deal of precision. Scientific applications or any other such applications that require very accurate results may be difficult to test manually or even if done, there are more chances of human error. Such, test cases, if automated, can be used with accuracy.
- Supporting Agile and extreme methodologies requiring high regression. These approaches require lots of regression testing of previously built functionalities as you add more and more functionalities to the system. Automating regression tests would help better coverage.
- Other than testing tasks. If you want to create, say 1,000 transactions in the system, you can use automated scripts to insert data rather than typing them manually.



Test case selection for automation

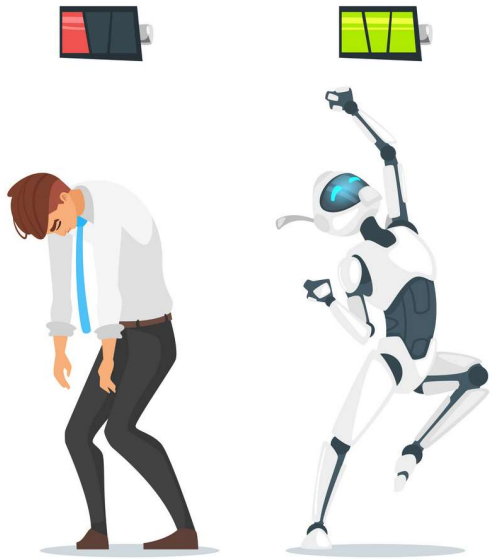
Test automation is not used to automate

- Exploratory testing – the objective is simultaneous learning and testing. If such tests are automatically performed, one cannot explore the functionalities in detail. And without knowing actual functionalities, it is not possible to develop scripts.
- Test that will never fail – why test when it is always going to pass? There could be very simple tests and having known that these tests are not going to fail once tested in the beginning.
- One of the tests – is that it does not pay back the investment in automation.
- Usability tests – Objectives can be best checked by the end users themselves.
- When the application UI changes frequently. Most test automation tools are GUI-based tools and operate in the same manner as actual users.

Automation benefits

Manual testing

- Time consuming
- Low coverage
- Human resources
- Inconsistent results
- Error prone



Automated testing

- Speed – 10 to 100 times faster than human
- Wider testing coverage
- Save time – frees people
- Consistent quality and output
- Reliable – no error
- Reusability, frequent execution
- Save cost

Risk-based testing

Risk

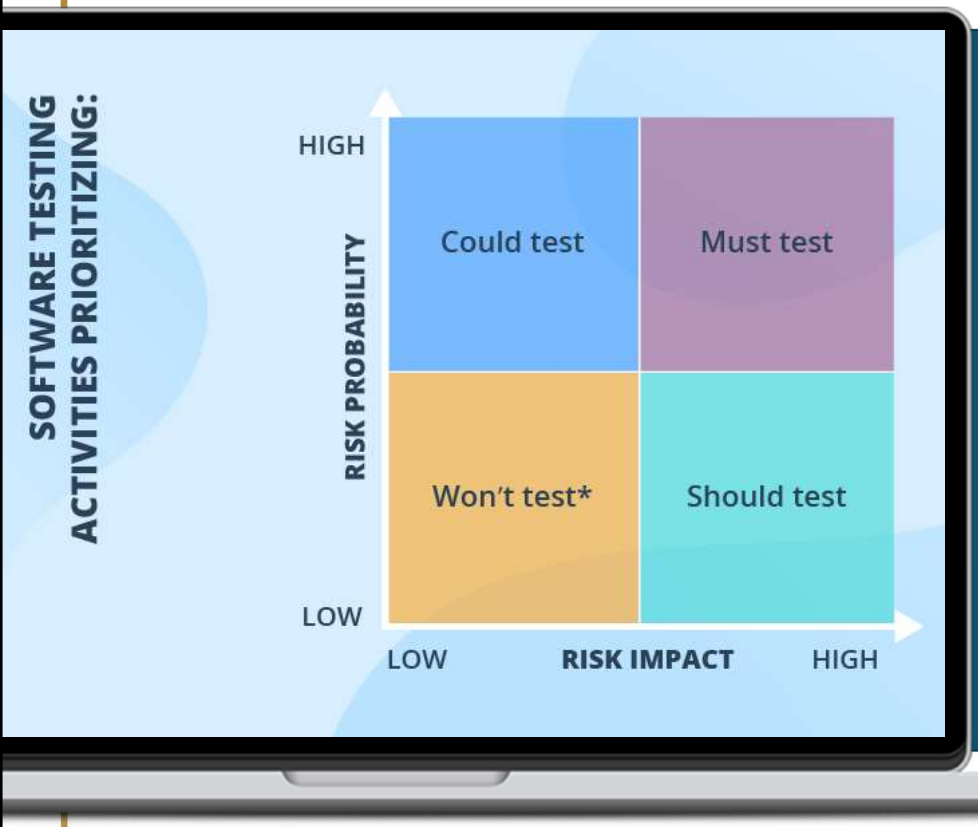
The possibility of a negative or undesirable result. In the future, a risk has some likelihood between 0% and 100%. The potential consequences or impact is an important consideration affecting the level of risk, too.

Risk-based testing

- Another technique is used to ensure minimum impact on the quality with a limited amount of effort and time. The primary objective of this technique is to focus on areas that are more critical as compared to areas that are less risky.
- The risk analysis is done on requirements, technical aspects, and people aspects and then a strategy is prepared.

Risk-based testing process

Risk-based testing



Benefits

- You can give up tests (do not execute) you worry about the least with low impact on quality so that the time and effort can be reduced but the overall quality is not much impacted.
- Pick the right tests out of the infinite possible tests reducing effort.
- Since you prioritize the tests based on risk analysis indicating which are must test, should test, could test and won's test, the likelihood of discovering high severity defects are faster to resolve.
- Make smart release decisions. With this approach, the analysis and test results can help us decide when to release the product.
- Risk-based testing involves both:
  - Mitigation – testing to provide opportunities to reduce the likelihood of defects.
  - Contingency – testing to identify workaround to make the defects that do get past us less painful.

Practical exercise

https://test-design.org/practical-exercises/