Galmont CONSULTING

Determining What to Automate

Because 100% automation of the entire test suite is typically not possible, developing a set of criteria to decide which tests to automate for regression is essential.

Building Quality Foundations for Business Applications

Galmont CONSULTING

Contents

Galmont CONSULTING

1. INTRODUCTION

This document will outline the approach for Test Analysts to take when considering which types of manual system test cases to automate, or when creating manual test cases that may later become candidates for automation.

This document contains three main sections. The first section outlines considerations for creating regression test cases for automation, the second section considers automation for new functionality, and the third section provides useful criteria to determine whether automation makes sense for particular test cases.

Also of use is the *Galmont Automation Feasibility Questionnaire*, which can help determine the overall "automatability" of an application based on its technology and the state of existing test cases.

2. SCOPE

The goal of this paper is to establish a process for making decisions about how to create or structure manual test cases so that they are reusable for automation purposes. It provides useful auditing guidelines to help test analysts in deciding which existing test cases are likely automation candidates.

This process does not advocate the use of any specific test automation tool, and the methods detailed here are entirely tools-agnostic. This document is entirely focused on the method for determining which manual test cases are appropriate for automation, not specifics on automation execution, status reporting or engineering best practices.

3. WHEN IS THE CONCEPT OF AUTOMATION A GOOD IDEA?

Regression testing focuses on ensuring existing functionality is not compromised by changes in the code. When running regression tests, a lot of time is invested in running repetitive tests that do not find many defects. This is a prime area to automate because it will allow the tester to refocus their testing on areas where defects might be found (i.e. new functionality, introducing new technology to an application, or when new development resources have been added).

4. WHEN IS THE CONCEPT OF AUTOMATION A BAD IDEA?

Automation is a bad idea when the automation objective is unclear, unrealistic or the foundation (requirements and manual test cases) is not solid.

Common beliefs that cause automation to fail:

- *Automation will find more bugs:*

In theory it seems reasonable that an automation testing tool running all of the test scripts quickly will uncover several defects. **In reality, it depends on the quality of the original manual test cases.** If a test case is poorly written and doesn't test all of the requirements to begin with, then how will automating that test uncover "more bugs?" When test cases have vague test steps and general expected results, it's inevitable that the real requirement will not be tested in the automation scripts. The correct objective should be to improve the quality of an application by reporting its status early on.

- *Automation is completely suitable for new functionality:*

While it is possible to automate tests that cover new or developing functionality, automation is best suited for stable applications that have already been through a manual testing cycle. Creating automation code based on application code that is in constant flux during a development phase is generally a bad idea. Automation needs to be constantly updated to meet the needs of new code.

Galmont CONSULTING

5. EXAMPLE OF AN AUTOMATABLE TEST CASE

Test Case: Generic Application Home Page Login/Navigate Test		
Name	**Description**	**Expected Result**
Step 1	**Steps:** Enter the URL in the browser **Data:** http://application/login.html	**Verification / Validation:** User is navigated to the Generic Application website and Login Home Page is displayed
Step 2	**Steps:** In the Login Function Box, located on the left navigation menu, enter: Login = John Doe Password = JohnDoe1234 Click “Submit” button **Data:**	**Verification / Validation:** Login and password data is accepted and successfully validated. The user is successfully taken to the home page The Welcome User title is displayed at the top left of the home page. The User ID is displayed at the top right of the home page.
Step 3	**Steps:** Verify there are no broken link icons on the home page **Data:**	**Verification / Validation:** Home page - all icons are displayed without red x's next to them Home page – all navigation links display a tool tip when hovered over indicating where the link will take the logged in user
Step 4	**Steps:** Click the "My Information" link on the right hand side of the page **Data:**	**Verification / Validation:** "My Information" page renders Under the “Profile" label, the logged in user’s name appears in the name field.
Step 5	**Steps:** Click the "Home" button at the top of the “My Information” page **Data:**	**Verification / Validation:** The Welcome User title is displayed at the top left of the page. The User ID is displayed at the top right of the home page.
Step 6	**Steps:** Click “Logout” button at the top right of the home page **Data:**	**Verification / Validation:** The following message displays: “Logout was successful” The Generic Application is closed

Galmont CONSULTING

6. EXAMPLE OF A TEST CASE THAT CANNOT BE AUTOMATED

Test Case: Generic Application Home Page Login/Navigate Test		
Name	**Description**	**Expected Result**
Step 1	**Steps:** Log into the Generic Application and verify all links are displayed **Data:**	**Verification / Validation:**
Step 2	**Steps:** "Home Page" information is displayed **Data:**	**Verification / Validation:** Information is displayed
Step 3	**Steps:** Navigate to the information page where user information is displayed **Data:**	**Verification / Validation:** Information is displayed
Step 4	**Steps:** Verify links work **Data:**	**Verification / Validation:** Links navigate user to other pages
Step 5	**Steps:** Exit the application **Data:**	**Verification / Validation:** Application closes

Why the test case is not automatable:

Step 1	**Steps:** Log into the Generic Application and verify all links are displayed **Data:**	**Verification / Validation:**

1. There are no instructions on how to execute each test step, leaving too many open questions for the test engineer to figure out.

Unanswered Questions - Log in where? What is the URL? What should be displayed and how? What shouldn't be displayed?

Step 2	**Steps:** "Home Page" information is displayed **Data:**	**Verification / Validation:** Information is displayed

1. There are no details in the expected results to tell the test engineer how the system will respond or what it will display to prove the test step fulfilled the functional requirements.

Unanswered Questions - What information should be displayed, how and where? What shouldn't be displayed? Is it data dependent? Was I supposed to do something to make information display?

Step 3	**Steps:** Navigate to the information page where user information is displayed **Data:**	**Verification / Validation:**

1. There are no instructions on how to navigate to the information page. The "information page" should be correctly named according to what information it will display when the user gets there. Where should the information on this page be located?

Unanswered Questions – How do I log in to the system? Is there a login and password page? Is this scenario data driven to follow the test steps and receive those specific expected results? Do I need to verify that specific data appears on each screen based on my login?

Step 4	**Steps:** Verify links work **Data:**	**Verification / Validation:** Links navigate user to other pages

1. There are no instructions on how to tell if the links work

Unanswered Questions – How can you tell that the links work? Should you click on each link and navigate away from the current page? If so, how do you navigate back? What format are links displayed in and what information should they contain? What kind of error message should display if a link does not work?

7. CRITERIA FOR CONVERTING A MANUAL TEST CASE TO AN AUTOMATED TEST

Regression automation enables testers to run specific tests to help ensure that changes to a system's software have not affected the unchanged parts of the system. This testing needs to run easily and automatically whenever a change is introduced to a system.

Because 100% automation for the entire test suite is typically not possible, defining criteria to decide which tests to automate for regression testing is essential. ***Criteria List 1*** below determines automation feasibility. ***Criteria List 2*** helps justify automation and prioritizes test cases. ***Criteria List 3*** details considerations for automation at a project's onset. The lists aren't meant to be completely comprehensive, but they can serve as a useful guide when first thinking about test automation.

8. CRITERIA LIST 1

Criteria List 1	Details
The test case must be repeatable.	A repeatable test case can be executed an infinite number of times where the expected result for each test case step will be the same for every execution. Execute the entire manual test case at least twice and verify that the expected result is the same for each test case step.
The test case steps must be easy to follow.	The manual test case steps must guide the test engineer easily so that he/she can continue to the next step without doing actions over and above those specified in the manual test. The test engineer will be new to the application - ensure that the manual test case is descriptive and detailed so that assumptions and misinterpretations can't invalidate the automation script. Automation will be difficult if the test case is hard to follow, too vague or missing steps.
The expected results must be consistent.	Generally, for every test case step, there is an expected result, which in turn must be the same for every test execution. The expected result should only deviate if a true failure occurs in the application. If the expected result changes for every test case execution, the validation in the automation script will always report a failure.
The expected result must be easily verified.	An expected result cannot be subjective, and it cannot be vague, for example stating, "The field color changed." Instead it must specify the color. The expected result for each test case step must be concrete, measurable or quantifiable so that it can be easily verified.
The test case must not have a dependency on an external application that is not installed on the same test PC.	If the test case relies on external applications that the automation tool cannot access, it will be difficult or impossible to automate. If the external application is used for pre-conditions for the test case, the pre-conditions must be set prior to test case execution. For example, if the test application requires an employee number, that employee number must be established from an external application to generate the number. The external application is a thick client application that resides outside the test PC, and therefore impossible to automate.
The test data for the manual test case must be established.	If the test case requires test data that does not exist, execution of the test case cannot be done. Make sure test data has been established and won't be deleted. Also, try to make sure that the test data is not constantly changing. Frequently changing test data will require the automation script or its associated data file to change constantly as well.

The expected results must not require verification against an external hardware.	Automation is difficult if not impossible for any application that has dependency on external hardware. For example, a spectrometer application is used to control the settings on a spectrometer machine. To verify that correct settings are applied, the machine has to be verified and this cannot be automated.
All the components of the applications must be recognized by the automation tool.	This question will not be answered until the test engineer has a chance to start developing the automation scripts. Some applications are straightforward and are easily recognized by the automation tool. Other applications have embedded objects which are less easily recognized or require add-ons or workarounds. Applications do exist that are not recognized at all by the automation tool, but they are rare. Check all the components of the application to avoid this problem.
The application must be in a stable state.	For automation to work, we need a stable foundation on which to create the automation script. If the application is missing major components or functionality where the manual test case cannot be completed, automation will be difficult. If the application constantly crashes, the automation effort will be lengthy.
The test case must not be a one-time execution test case.	If the manual test case is only going to be used as a one-time test, there's no reason to automate it.

8.1 *Criteria List 2*

Criteria List 2	Details			
Development Difficulty	***Not Possible: 0 points.*** The test includes steps that cannot be replicated with a utility or other application.	***Hard: 1 point.*** The test includes steps that three or more applications must perform.	***Intermediate: 2 points.*** The test includes steps that two applications must perform.	***Easy: 3 points.*** The test can be performed from a single browser or interface.
Time Savings	***No Time Savings: 0 points.*** Setting and executing the automation would take longer than manually executing the tests.	***Low: 1 point.*** The test takes less than 10 minutes to execute manually.	***Medium: 2 points.*** The test takes more than 10 minutes but less than 60 minutes to execute manually.	***High: 3 points.*** The test takes more than 60 minutes to execute manually.

Frequency of Use	***No Plans for Continued Execution of the Test Suite: 0 points.*** The test will only be run a single time, for the current release.	***Low: 1 point.*** The test is not used regularly in any test cycles.	***Medium: 2 points.*** The test is used regularly in at least one test cycle.	***High: 3 points.*** The test is used regularly in multiple test cycles or multiple configurations in at least one test cycle.

To determine the score of a test case, multiply the points for each category:

(Score = Development Difficulty x Time Savings x Frequency of Use). The higher the score, the higher the priority for the test case to be automated. A test case with a score of 0 would not be automated.

9. OVERALL PROJECT CRITERIA FOR AUTOMATION

Software development projects are always seeking quicker and more efficient ways to identify defects. Test automation is often seen as the answer. This is only true if the testing projects are stable and suitable for automation…not every project is a candidate.

When considering automation of new functionality, there are several aspects that need to be considered. Understanding the project components (budget, time and resources), application history, company process, and testing experience all need to be taken into account.

The following criteria should be used when determining if a QA project is desirable for test automation.

9.1 *Criteria List 3*

Criteria List 3	Details
Cost	The number-one factor influencing automation in a testing project is the cost of automated versus manual testing. The question is, “Will the cost of automation provide a return on investment?” Determine the ROI of the automation cost for the testing project.
Development Cycle	First, determine the number of releases planned for the year. Of these releases, how many builds are given to the QA group to test? This gives us the number of testing cycles for the product. The higher the number, the more regression testing that will be needed. Automating the regression tests early will provide the greatest automation benefits.
Automation Test Engineers	It will not make sense to take an automation testing approach if test engineer resources are not available. Only software quality assurance professionals who are trained to be automation test engineers should be allocated as the resource for automation tasks.
Project Schedule	If the project schedule is tight, or there is urgency for project completion, creating new automation may not be suitable. The project schedule needs to include the amount of time required to create the automated scripts. Allocate the right resource and timeframe to complete the automation tasks just as you would development tasks.

Number of Test Cases	Determine the number of test cases for the new functionality. The greater the number, the more time it takes to manually execute. If the test cases meet the criteria in “Criteria List” above, the more sense it may make to do the automation.
Solid Functional Requirements	If the functional requirements are vague, missing information or do not cover all positive, negative and boundary use cases, automation may not prove successful on this new functionality.
Regression Candidate	Will the new functionality be part of the regression suite? If yes, queue up the manual test cases for automation. The faster the automation gets done, the faster it can be used for regression as new builds or new functionality gets developed. The test cycle time benefits and cost savings of test automation are quickly realized the more frequently the automation is leveraged.