LogiGear®

WHITEPAPER

HOW TO WRITE AN ACTION-BASED TESTING (ABT) TEST MODULE

MAR 2015

logigear.com

TABLE OF CONTENTS

INTRODUCTION

Action Based Testing (ABT) is an efficient method of test development that provides a systematic approach to increase the success of automated testing.

ABT uses Test Modules to increase the efficiency of test development. Test modules provide a level of abstraction over test cases and make it possible to create well-defined test case flows. The top-down planning approach helps to create test cases that are free of unnecessary details and redundant checks. Test cases themselves are authored using actions that make the tests readable and that can be automated with minimal programming support.

Below is a template that shows the systematic construct of a Test Module. Following the template is an example of a complete Test Module and the step-by-step sequence of ABT test development.

LogiGear Headquarters

4100 E Third Ave, Suite 150
Foster City, CA 94404

Phone: (800) 322-0333
Fax: (650) 572-2822
sales@logigear.com

testarchitect.com
logigear.com

TEST MODULE TEMPLATE

Test Modules are containers for organizing tests of user stories or software requirements. Organized this way every test module will have a clear and well-differentiated scope from every other test module, and it will reduce redundant activities and checks, making tests less fragile and easier to maintain. Giving each Test Module a descriptive name makes it easy to identify what the test in the module cover.

There are four sections to a Test Module: Objectives, Initial, Test Case and Final

1. The **OBJECTIVES** section lists every objective associated with the module's test cases and defines the scope of the test module. Objectives allow readers to understand why test cases are designed the way they are, and give an auditor a quick insight into the correctness and completeness of a test.

2. The **INITIAL** section of a test module contains the action lines required for initialization of the test. For example, actions for launching the application under test would be written here. This will avoid repetitive steps in test cases when initialization is required for multiple test cases in the module.

1	**TEST MODULE**	ABT Test Module Template	
2			
3	OBJECTIVES	ID	Title described in "cause and effect" format
4	test objective	TO 1	Cause and effect
5	test objective	TO 2	Cause and effect
6	test objective	TO 3	Cause and effect
7			
8	INITIAL	Setting up	
9			
10	//set up steps: launch apps, setup test data, etc.		
11			
12	TEST CASE	TC 1	Title
13	test objective	TO 1	Cause and effect
14			
15	//activities and checks		
16			
17	TEST CASE	TC 2	Title
18	test objective	TO 2	Cause and effect
19			
20	//activities and checks		
21			
22	TEST CASE	TC 3	Title
23	test objective	TO 3	Cause and effect
24			
25	//activities and checks		
26			
27	FINAL		
28			
29	//clean up steps: delete created data, close browsers, close apps, etc.		

3. The **TEST CASE** section is where test cases are created. Each test case has a representative number, accompanied by title and test objective line. The title and objective explain the test case purpose. Test Modules can have multiple test cases and each may have one or more test objectives associated with it.

4. The **FINAL** section is for any cleanup operation upon test completion, such as closing the application under test.

EXAMPLE OF A COMPLETED TEST MODULE

The Test Module example below is for a test of a restaurant reservation application. It follows the user story "As a registered user, I want to be able to reserve, change and cancel a restaurant reservation".

1	**TEST MODULE**	Restaurant Reservations			
2					
3	OBJECTIVES				
4	test objective	TO 01	Reservation complete will send confirmation email		
5	test objective	TO 02	User can change reservation details		
6	test objective	TO 03	User can cancel the reservation		
7					
8	INITIAL	Setting up			
9					
10		username	password		
11	sign in	johnd	p@ssw0rd123		
12					
13	SECTION	Pick some dates to use in the tests			
14					
15		days from now	result		
16	pick date	1	>> tomorrow		
17	pick date	7	>> next week		
18	pick date	30	>> next month		
19					
20	TEST CASE	TC 01	Reserve a restaurant		
21	test objective	TO 01	Reservation complete will send confirmation email		
22					
23		restaurant	party size	date	time
24	make reservation	Evvia	2	# tomorrow	9 PM
25					
26		username	contains		
27	check confirmation email	johnd	Your reservation is confirmed for Evvia		
28					
29	TEST CASE	TC 02	Change reservation multiple times		
30	test objective	TO 02	User can change reservation details		
31					
32		restaurant	party size		
33	change reservation	Evvia	4		
34					
35		username	contains		
36	check confirmation email	johnd	Evvia will be ready for your party of 4		
37					
38		restaurant	date	time	
39	change reservation	Evvia	# next week	8 PM	
40					
41		username	contains		
42	check confirmation email	johnd	# "at 8:00 PM on " & next week		
43					
44	TEST CASE	TC 03	Change multiple details of one reservation		
45	test objective	TO 02	User can change reservation details		
46					
47		restaurant	party size	date	time
48	change reservation	Evvia	10	# next month	6 PM
49					
50		username	contains		
51	check confirmation email	johnd	# "party of 10 at 6:00 PM on " & next month		
52					
53	TEST CASE	TC 04	Cancel reservation		
54					
55		restaurant			
56	cancel reservation	Evvia			
57					
58		username	contains		
59	check confirmation email	johnd	You've successfully canceled your reservation at Evvia		
60					
61	FINAL	Cleaning up			
62					
63	sign out				
64	close all browsers				

- The test module starts with a listing of the objectives that the tests in this module need to meet.
- The initial section contains sign in data and variables for reservation dates.
- The "pick date" action assigns a date a given numbers of days from now, to a variable. (In TestArchitect variables are preceded with "#", to indicate an expression).
- Test Case 1. User signs in and creates a reservation at the restaurant Evvi that is confirmed by email. Verification is done by sampling the confirmation a customer will receive. This illustrates the flexibility of actions to express the intention of a tester.
- Test Case 2. Changes are made individually to
  1) first the party size
  2) then the date/time. Both check for email confirmation.
- Test Case 3. Several fields are changed in one transaction.
- Test Case 4. Tests that a reservation can be cancelled.
- The final section of the module signs out and closes the browser.

CREATING THE ABT TEST MODULE FOR RESTAURANT RESERVATIONS

1. Structure

Test modules typically contain tests for "business" or "interaction". Business tests will be like "Restaurant Reservations". The details of how to reserve a table are contained in the actions. Interaction tests verify the more detailed interactions with the UI (or API) of an application. Activities could be "enter user name", "enter password", "click log in button". It's important to make sure business functionality is not the focus of interaction tests, and that no interaction steps are used in a business test.

2. Defining Test Objectives

Test objectives are the break-down of the scope of test module. They should focus on what to test— not how to test. Test objective should be clear, descriptive and distinct. An objective can be described using "cause and effect" format as shown for Restaurant Reservations:

1	**TEST MODULE**	Restaurant Reservations	
2			
3	OBJECTIVES		
4	test objective	TO 01	Reservation complete will send confirmation email

3. Test Case authoring

Test case authoring in ABT is done in a structured way using actions. Actions make the test case is readable for outsiders with relatively little knowledge of technology and omit the details that are not essential to understand a test. Details that are needed to execute the test (like which menu item to choose to open a dialog box) should not be visible in a test module unless they are important to the scope of that test module. All the details to execute the test should be contained in the actions.

Test Case 1 is a business test case that uses two actions—"make reservation" and "check confirmation email"— and passes appropriate test data to those two actions. Without seeing the UI of the AUT, it can be deduced that the action "make reservation" reserves a table at the specified restaurant with details such as party size, date and time. The other action "check confirmation email" checks whether the email sent to the user contains the expected content. All the steps (and complexity), to make a reservation or check the email content, are abstracted within those two actions.

20	TEST CASE	TC 01	Reserve a restaurant		
21	test objective	TO 01	Reservation complete will send confirmation email		
22					
23		restaurant	party size	date	time
24	make reservation	Evvia	2	# tomorrow	9 PM
25					
26		username	contains		
27	check confirmation email	johnd	Your reservation is confirmed for Evvia		

(Note: '#' is an operator in ABT Language to retrieve value of the following variable)

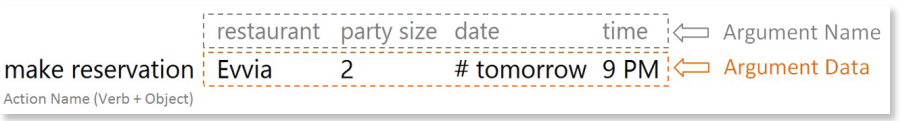
CREATING THE ABT TEST MODULE FOR RESTAURANT RESERVATIONS (cont.)

Structure and Syntax

Tests will typically consist of actions specifying activity and verification. Often verification will follow the activity, but they can also be mixed. It is customary to start the names of verification actions with the word "check".

20	TEST CASE	TC 01	Reserve a restaurant			
21	test objective	TO 01	Reservation complete will send confirmation email			
22						
23		restaurant	party size	date	time	⇦ Activity
24	make reservation	Evvia	2	# tomorrow	9 PM	
25						
26		username	contains			⇦ Verification
27	check confirmation email	johnd	Your reservation is confirmed for Evvia			

Lines 24 and 27 are action lines (or actions) in the test case.



To name an action, use a verb followed by an object. The name of an action and its argument should be descriptive and associated with AUT's GUI or business process.

Arguments are typically the input values needed to do the desired operation in action. All arguments carry default values (which can be empty), and are therefore optional. When the action is used, only input the arguments that matter for that particular context of the test.

4. Action Definition

At some point in the process actions have to be defined and created. Below is an example of the "make reservation" action definition.

1	ACTION DEFINITION	make reservation		
2				
3		name	default value	description
4	argument	restaurant	Evvia	Restaurant name
5	argument	party size	2	Size of party
6	argument	date		Date of party
7	argument	time		Specific time of party
8				
9		restaurant		
10	search restaurant	# restaurant		
11				
12		party size	date	time
13	set reservation details	# party size	# date	# time
14				
15		window	control	
16	click	complete reservation	reserve button	

(Note: '#' is an operator in ABT Language to retrieve value of the following variable)

The action definition provides more details about making a reservation , however it's not specified, nor necessarily important, what the AUT looks like or what kind of UI interactions have to be done (which are very easy to change). What's important is to push as much detail as possible to the lower level of actions.

In this example, the action "make reservation" also calls other actions ("search restaurant" and "set reservation details") that would each have their own action definitions with details.