odin

Axe in the Agile World

WHITE PAPER

Executive Summary

This paper explains the way in which Axe (Odin's Enterprise Test Automation Platform) allows the automated testing to take place in a range of project approached including agile, "wagile" and waterfall.

Clearly agile approaches are based on the need to develop smaller deliveries of functionality during compressed timed sprints. This doesn't mean that testing should be affected adversely by this, Axe tackles this to maintain quality and a high level of documentation relating to the tests by using the following:

- Axe allows the scripting (in Axe) of the tests to meet the functional requirements before the functionality is delivered within sprint. It doesn't use outdated approaches such as capture replay which the need the application to be present to work, thus the automation process can start earlier
- Axe generates scripts for underlying execution tools (dependent on the complexity of the application being developed) and also generates the documentation to reflect what is actually being tested and the related results produced by Axe also reflect this
- Axe actually speeds up the overall sprint with on the basis that Axe allows parallel testing preparation alongside the development coding effort. Thus once the code is released, it can be tested immediately by Axe
- Axe gives the real possiblity of utilising test expertise within the scrum team and ensures that regression testing can still be performed across the overall functionality of the application. Axe also ensures that the backward slip towards manaul testing doesn't happen thus protecting your investment in automation tooling
- Axe deploys a standard automation process irrespective of the development approach or the underlying automated test tooling (e.g. Microsoft, HP, IBM, Selenium etc…). This means once a tester has been trained in Axe, they can work with any of the integrations supported by Axe in the same way, they simply do not need to know the technicalities of the underlying test execution tool to do an excellent and productive job

Axe is a well proven platform with over 100 clients globally including Bank of America, Microsoft, ABB and National Grid.

Axe supports the leading automation tools from Microsoft, IBM, HP, Open source, SAP and Oracle inluding the test management aspects of these tools (i.e ALM from HP, MTM from Microsoft, RQM from IBM etc..)

Finally the Enterprise aspect of Axe means that these underlying execution tools can be joined to carry out real enterprise testing and this includes a "tool less" integration with SAP.

If you have any queries or comments, please do not hesitate to contact David Tracey (david.tracey@odintech.com or info@odintech.com) or via mobile number 07768-138512, enjoy the white paper!

Axe in the Agile world

Terminology is an ever changing characteristic of the IT world, whether it is the evolution of new acronyms or the development of new methodologies for the delivery of user requirements for all types of application ranging from simple to complex systems.

One of the most used words in today's IT terminology is the use of the word agile. A dictionary defines the explanation of this adjective as "quick and well-coordinated in movement; nimble". This means that as deliverers of IT solutions we need to modify our behaviour significantly from traditional approaches to deliver and conform to this agile definition.

This White Paper focuses on the need for agile test automation approaches based around the use of the Axe platform that facilitates implementation of the principles of agile being used for automated testing. Axe satisfies the requirement for delivery of increments of functionality generated in the sprint as well as allowing full regression testing of the application when required.

Many test automation approaches being used in the agile world have probably not really advanced in their approach from that used in Waterfall developments and some still take very traditional approaches to test automation that simply don't fit into the agile way.

Axe works equally well with agile and waterfall – more importantly Axe can service both concurrently via the same automation process unlike most frameworks or test tools…

By this we mean that because agile is about small and frequent, the need for an application to be present to test simply would replicate deficiencies found in the waterfall method of development. Many tools still rely on "Capture/Replay" approaches which always need the application to be present thus replicating the "Waterfall" development approach but this time in smaller functional test "chunks", clearly not the agile way to do things!

Axe addresses this by allowing the automated testing to be synchronised with the development effort so that automated testing can be performed immediately that the code becomes available to test, this is true for both agile and waterfall methodologies. This means:

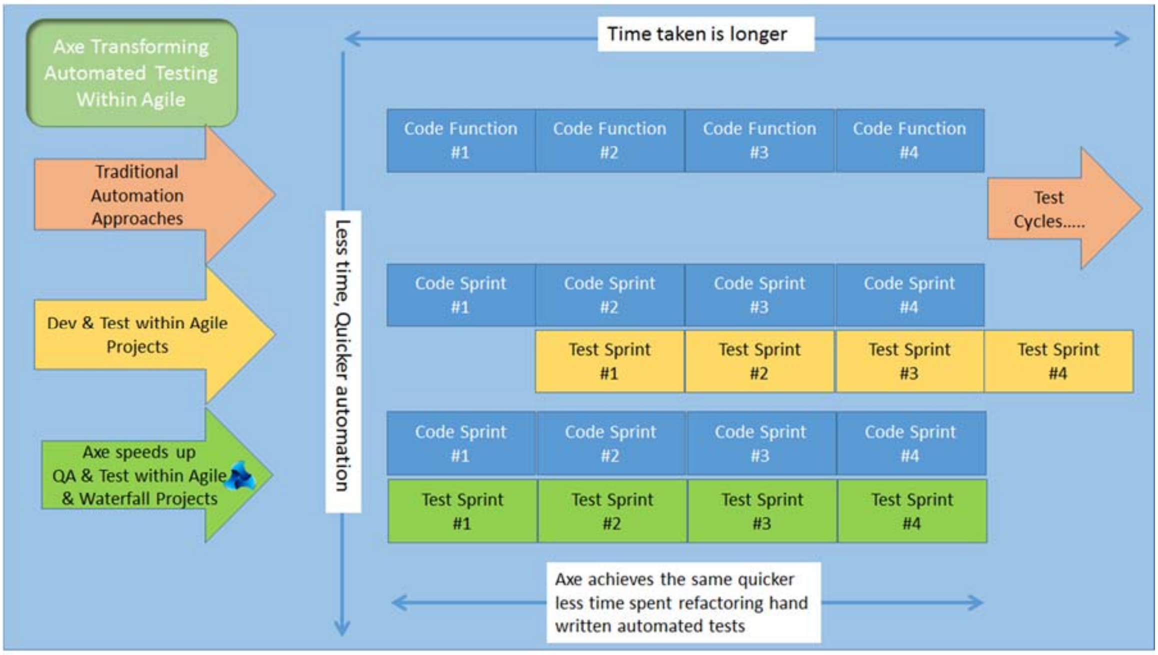
- There is no time lag in terms of being able to test "shippable product". Test Automation begins at the same time as code delivery and is ready at the same time to execute the required tests
- There is no dependency on the application being complete before starting the test automation process within Axe by defining test scenarios based on user stories and acceptance criteria
- Agile team members can get on with their relevant skill related tasks without the need for cross over between development and testing, often the case with BDD approaches where scripting is required
- Developers maximise their own productivity by being able to allow developers to concentrate on unit level testing and allowing testers to focus on the testing levels beyond unit testing.
- Axe can be used across large programmes which may mix agile and waterfall approaches and different automation test tooling. It can allow the QA process around test automation to be standardised and still

return high quality, productivity and consistent reporting irrespective of the chosen development methodology or automation tool

- It improves on the BDD approach by generating scripts and detailed test documentation to the same standard and conformity but managed by the QA and Testing sprint team members. It also gives the test pack scalability from simple functional tests to large regression packs which will be required to meet the continuous integration requirement
- The important points are significant productivity gains, synchronised and common test processes, high quality and standardised reporting by using the Axe platform approach a opposed to a mixture of tools

Axe makes even the agile process quicker!

This advantage is shown in the diagram below:



As agile employs the concept of "Definition of Done" which must be satisfied for a task to be considered as complete during a sprint. The sprint team members tasked with testing and quality assurance need to generate tests to match the user story and acceptance criteria for the task.

Testing thus occurs in a Continuous Integration environment and the ability to regression test becomes an essential component. From both a quality and test automation perspective this can also be perceived as conforming to the dictionary definition of agile as "well-coordinated in movement and nimble".

In order for any approach for automated testing to be considered as agile, the following would need to be embraced and covered:

- The ability for the sprint test specialists to be able to take "requirements" whether in User story format and acceptance criteria or some other definitive requirement structure and deliver automated tests within the sprint
- If the previous point is correct then this would also mean that finally the idea of waiting for the application to be modified and delivered and using outdated and maintenance intensive techniques such as Capture/Replay also become invalid. They simply don't satisfy the meaningful "nimble" approach to developing automated tests in time to converge with the development aspects of the sprint. This for many tools is the only approach deployed for developing automated tests quickly and fails on this important criteria

- It needs to be capable of building incremental test suites quickly and in a "Just-In-Time" method, ensuring the sprint time is kept to a minimum. By this we mean that automated tests can be executed once a sprint deliverable is available
- Being able to amend existing tests in response to removal of technical debt from the development process easily
- Compatible with the task based process, i.e. tests are produced for individual tasks and when the DoD (Definition of Done) is satisfied these need to be merged into the regression tests
- Capable of reporting testing results into the sprint (reporting based on task, as above and also reporting for end of sprint test coverage)

Consider all of the above and if one or more of the bullet points cannot be "ticked" and satisfied then your automated testing process is neither nimble nor "Agile" whatever you tell management! Perhaps there are lessons to be learnt here?

Focusing on the Automated Testing Process (QA and Test during the sprint)…

At Odin Technology, we have taken a pragmatic view of test automation within the overarching agile process itself and have drawn a number of significant conclusions.

If agile development is relevant as an approach at any time of a development, then the automation process itself is a good place to start as it is a fundamental component of agile delivery.

This means in essence:

- The automation test process itself needs to be common irrespective of any automation tools used
- Simplicity needs to be the key for the tester to be able to automate the user stories or equivalent into automated test cases – productivity and speed need to be at a maximum and maintainability needs to be at an absolute minimum
- Outputs and documentation must be standard and where appropriate due to the global nature of the IT industry possibly available in different languages to ensure errors are not introduced by the misunderstanding of language or more importantly what is actually planned to be tested or has been tested
- Knowledge needs to be captured, easily modified and recreated at the highest level, thus low-level and detailed tools will probably negate conforming to the desired level of quality and introduce maintainability problems in the process and the related scripts produced

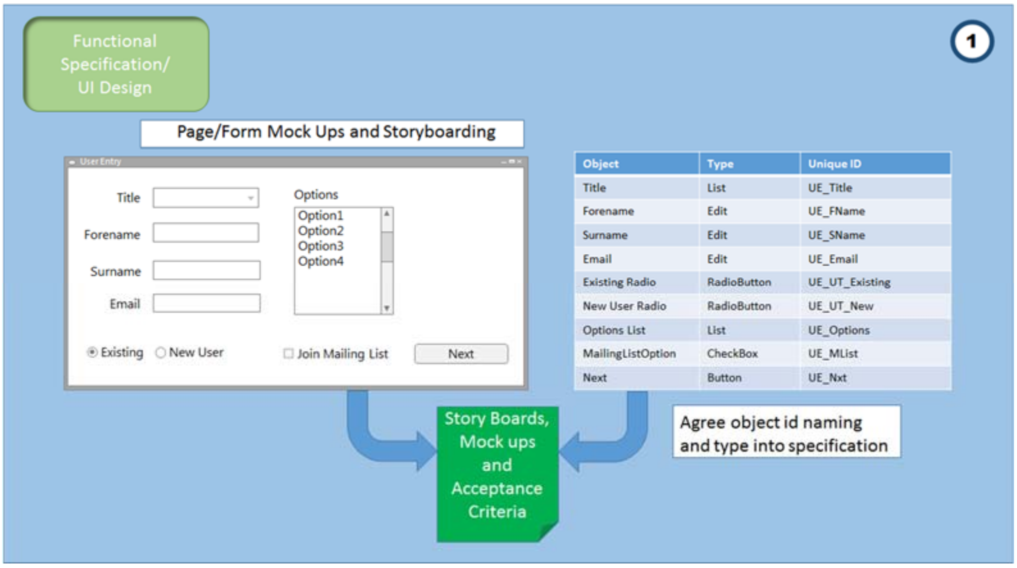
To look at the process in more detail, let's step through the approach through the agile process and look at the impact and tasks to be carried out on the Development and Automated testing paths. An overview of the start of the process is shown below:

Step 1 takes the User Story from the backlog, and as a task will produce Page and form "Mock ups" along with the object id naming are gathered into the technical solution. Although the overarching agile process means both strands (automation test and development) actually co-exist, their tasks differ.

Although the tasks differ (code vs. automated tests), they both have the same inputs. The converged outcome is a product that is ready to be shipped.

The rest of this paper is targeted at the way in which the automation aspects are handled by the Axe platform, as previously mentioned this process is the same irrespective of the underlying tool used with Axe. It is worth stressing at this point that the Axe automation process delivers generated scripts and documentation to ensure that the user story and acceptance criteria are satisfied.

"If it is potentially shippable, it must be tested… period!"…..


Functional Specification/ UI Design
1
Page/Form Mock Ups and Storyboarding
User Entry
Title
Forename
Surname
Email
Options
Option1
Option2
Option3
Option4
● Existing
○ New User
□ Join Mailing List
Next
Story Boards, Mock ups and Acceptance Criteria
Agree object id naming and type into specification

Object	Type	Unique ID
Title	List	UE_Title
Forename	Edit	UE_FName
Surname	Edit	UE_SName
Email	Edit	UE_Email
Existing Radio	RadioButton	UE_UT_Existing
New User Radio	RadioButton	UE_UT_New
Options List	List	UE_Options
MailingListOption	CheckBox	UE_MList
Next	Button	UE_Nxt

The key output here becomes the common agenda for the sprint and allows both the development team and automation test team to follow the agile process. The important point here is that the team is a multi-disciplinary team but the tasks are actually delivered as tasks in two parallel strands. This ensures the most productive and effective approach to using resources within the sprint team.

At the start of a sprint the team reviews the PBI (Product Backlog Item) list and estimates are generated.   The team decides how many PBI's can be completed in the sprint and if necessary subdivides large epic PBI's into smaller more manageable chunks.

The sprint team generates tasks in order to divide the work into pieces relevant to the individual specialisms with development and test being two examples. This parallelism starts as work begins on each "chunk" of work.

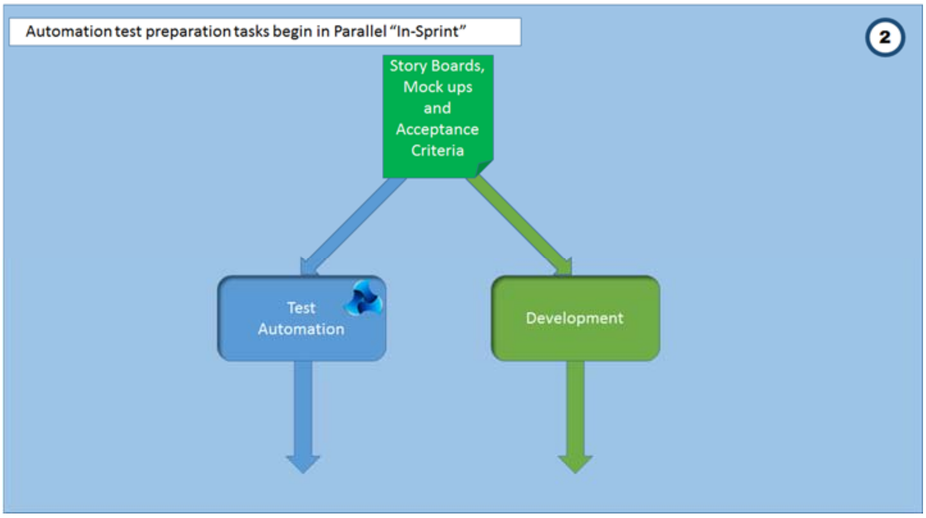
Capture Replay tools simply can't deliver an agile Solution – they rely on the application being present, Axe delivers a solution that works with BDD, TDD and DDT approaches….

As such there needs to be a proven test automation process which is not dependent upon the applications being present in order to start the test automation process, this is where Axe is unique in the automation industry.

All test automation process that relies on Capture/Replay techniques thus fail on being able to embrace a parallel process with development and fail of course due to the time taken to deliver the automation tasks required to complete the sprint itself. In addition, Axe allows the use of Data Driven Testing (DDT) for all of its tool integrations thus allowing the generation of greater scaling of test scenario variants with a very low maintenance overhead.

Traditional test automation approaches usually are "end on" to the development process, thus in practice this elongates the overall delivery timescales for delivery of the change to the business. Clearly a serious point of failure for an agile approach, one that Axe process defined in this document makes sure doesn't fail.

The next step (Step 2) is to pass the specification to both the development and test automation teams to commence their part of the process as shown below:



Automation test preparation tasks begin in Parallel "In-Sprint"

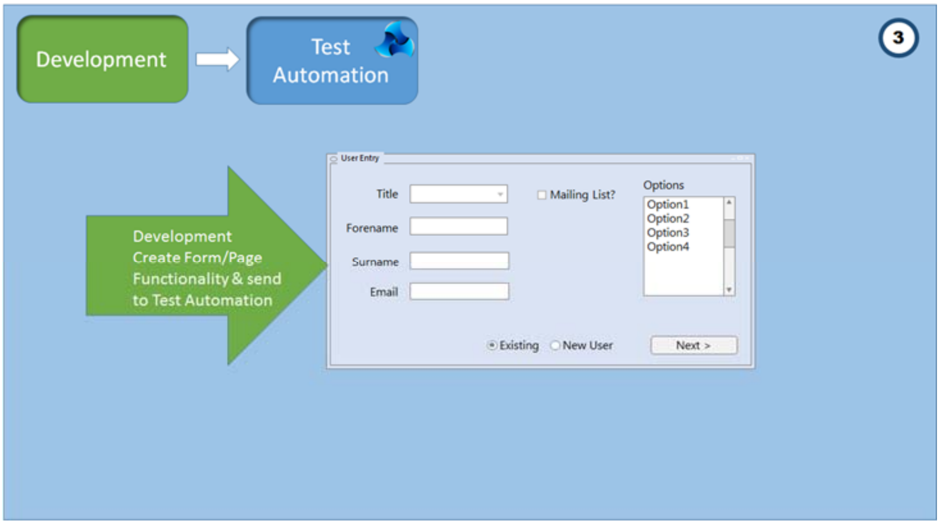
2

Story Boards, Mock ups and Acceptance Criteria

Test Automation

Development

In the next step (Step 3) Development creates the form/page functionality and this is past to the counterparts on the Test Automation strand. This is the start point for the automation team to start building the automated tests as per the diagram below:



Development

Test Automation

3

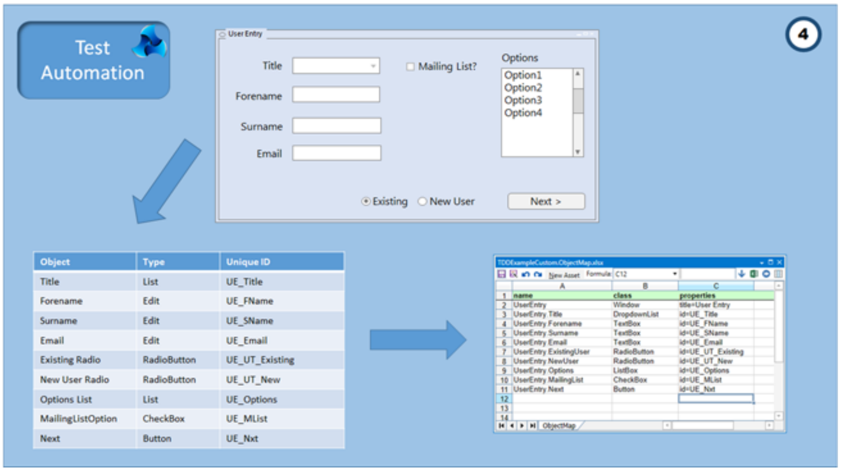
Development Create Form/Page Functionality & send to Test Automation

With Axe the test automation can start before the Application is delivered

In parallel to the development process (which is actually the long and hard part) the test automation process (Step 4) begins by creating the Object map based on the contents of the specification.

This involves

- Giving each object a business name
- Defining the object class (considering the underlying execution tool to be used
- Defining the properties used to identify the object



Object	Type	Unique ID
Title	List	UE_Title
Forename	Edit	UE_FName
Surname	Edit	UE_SName
Email	Edit	UE_Email
Existing Radio	RadioButton	UE_UT_Existing
New User Radio	RadioButton	UE_UT_New
Options List	List	UE_Options
MailingListOption	CheckBox	UE_MList
Next	Button	UE_Nxt

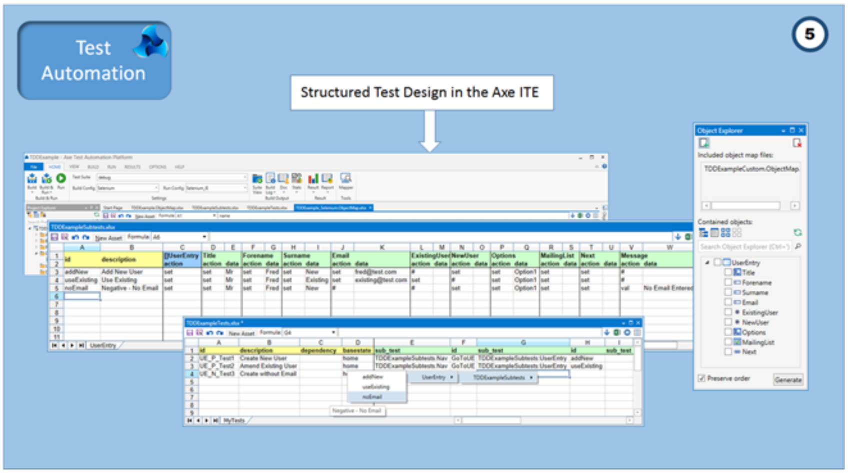
TDDExampleCustom.ObjectMap.xlsx

Formula: C12

name	class	properties
UserEntry	Window	title=User Entry
UserEntry.Title	DropdownList	id=UE_Title
UserEntry.Forename	TextBox	id=UE_FName
UserEntry.Surname	TextBox	id=UE_SName
UserEntry.Email	TextBox	id=UE_Email
UserEntry.ExistingUser	RadioButton	id=UE_UT_Existing
UserEntry.NewUser	RadioButton	id=UE_UT_New
UserEntry.Options	ListBox	id=UE_Options
UserEntry.MailingList	CheckBox	id=UE_MList
UserEntry.Next	Button	id=UE_Nxt

The next stage (Step 5) within the test automation strand is to structure the tests using the Axe ITE (Integrated Test Environment), this creates tests using the pre-defined object map to generate worksheets for population by the tests required to meet the story board or specified requirements. This involves:

- Defining the sub-tests (reusable components)
- Defining test scenarios by joining together a sequence of sub-tests to produce a route through the Application Under Test (AUT)



TDDExampleSubtests.xlsx

Formula: A6

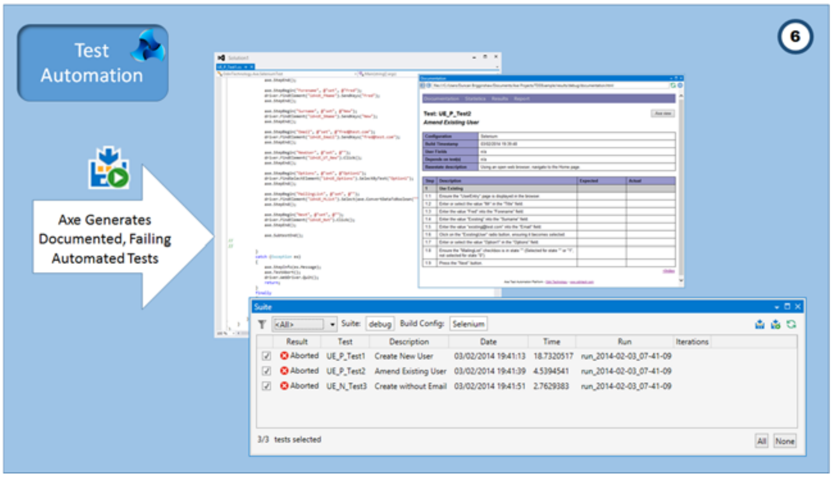
id	description	[]UserEntry	Title		Forename		Surname		Email		ExistingUser		NewUser		Options		MailingList		Next		Message	
		action	action	data	action	data	action	data	action	data	action	data	action	data	action	data	action	data	action	data	action	data
addNew	Add New User	set	set	Mr	set	Fred	set	New	set	fred@test.com	#		set		set	Option1	set		set		#	
useExisting	Use Existing	set	set	Mr	set	Fred	set	Existing	set	existing@test.com	set		#		set	Option1	set		set		#	
noEmail	Negative - No Email	set	set	Mr	set	Fred	set	New	#		#		set		set	Option1	set		set		val	No Email Entered

TDDExampleTests.xlsx *

Formula: G4

id	description	dependency	basestate	sub_test	id	sub_test	id	sub_test
UE_P_Test1	Create New User		home	TDDExampleSubtests.Nav	GoToUE	TDDExampleSubtests.UserEntry	addNew	
UE_P_Test2	Amend Existing User		home	TDDExampleSubtests.Nav	GoToUE	TDDExampleSubtests.UserEntry	useExisting	
UE_N_Test3	Create without Email		h					

The ultimate step in the test automation strand (Step 6) is to generate documented failing tests, i.e. these are scripts that are awaiting the delivery of the parallel development stream (i.e. the application). This can be shown from a process perspective in the diagram below:



Test: UE_P_Test2
Amend Existing User

Configuration	Selenium
Build Timestamp	03/02/2014 19:39:40
User Fields	n/a
Depends on test(s)	n/a
Prerequisite description	Using an open web browser, navigate to the Home page

Step	Description	Expected	Actual
1	Use Existing		
1.1	Ensure the "UserEntry" page is displayed in the browser		
1.2	Enter or select the value "Mr" in the "Title" field		
1.3	Enter the value "Fred" into the "Forename" field		
1.4	Enter the value "Existing" into the "Surname" field		
1.5	Enter the value "existing@test.com" into the "Email" field		
1.6	Click on the "ExistingUser" radio button, ensuring it becomes selected		
1.7	Enter or select the value "Option1" in the "Options" field		
1.8	Ensure the "MailingList" checkbox is in state "" (Selected for state "" or "Y", not selected for state "0")		
1.9	Press the "Next" button.		

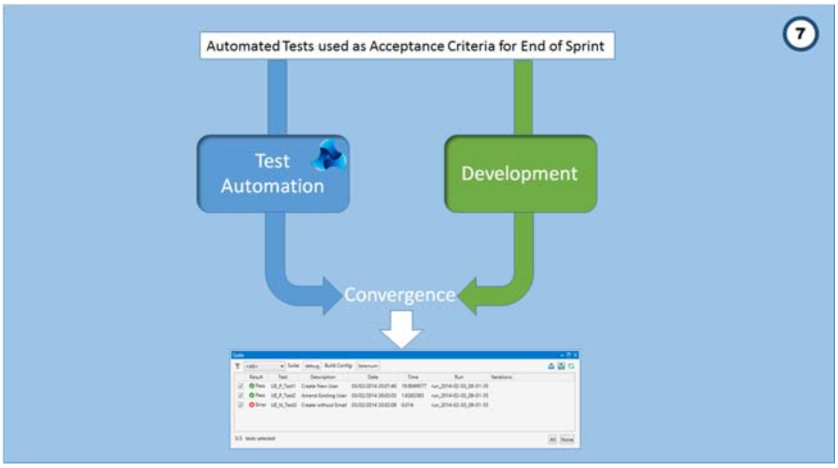
Suite: debug Build Config: Selenium

	Result	Test	Description	Date	Time	Run	Iterations
☑	Aborted	UE_P_Test1	Create New User	03/02/2014 19:41:13	18.7320517	run_2014-02-03_07-41-09	
☑	Aborted	UE_P_Test2	Amend Existing User	03/02/2014 19:41:39	4.5394541	run_2014-02-03_07-41-09	
☑	Aborted	UE_N_Test3	Create without Email	03/02/2014 19:41:51	2.7629383	run_2014-02-03_07-41-09	

3/3 tests selected

The final stage in the overall agile process (Step 7) is to "converge" the two streams together in order to test the developed application under test (AUT) by testing it with the automated tests we have generated as the final step of the test automation strand.

This is shown below:


7
Automated Tests used as Acceptance Criteria for End of Sprint
Test Automation
Development
Convergence

	Result	Test	Description	Date	Time	Run	Iterations
☑	Pass	UE_P_Test1	Create New User	[illegible]	[illegible]	[illegible]	
☑	Pass	UE_P_Test2	Amend Existing User	[illegible]	[illegible]	[illegible]	
☑	Error	UE_N_Test3	Create without Email	[illegible]	[illegible]	[illegible]	

At this point the test results can be analysed and as a result defects can be raised. As the object map which represents the way the application was intended to look may have changed without the test automation being amended means that the manual object map created in step 4 may need modification to reflect any changes and the scripts and documentation regenerated to encompass the change.

Conclusion

Odin Technology believes that Axe offers a unique Test Automation approach to agile implementation that is not offered by other automation tools and frameworks. The characteristics that exemplify this are:

- Parallelism in terms of development and test automation processes - but in an integrated agile process
- Productivity and better quality through the sprint process, especially the ability to "keep up" with the velocity of the sprint in hand as defined and managed by the Product Owner
- Full documentation at the lowest level to enhance the User Story to a practical level of detail and granularity to assist in defect resolution either of the delivered application components or the test automation itself –

in effect this generated documentation serves to define the exact granularity of testing carried out rather than redefine or detail the User Story further

- A standard process to be used in Enterprise organisations where complexity of automation tools is usually common but the enterprise process for automation differs due to the complexity and "personality" of the tools deployed
- Above all a simple, easy to learn/use and effective enterprise test automation process which allows "testers to automate" sprint deliverables and be able to run regression to ensure overall the end-to-end process still delivers what the business wants

In conclusion – Axe ticks all the boxes for the agile approach to test automation, does your current tooling and process?

If you would like to know more about Axe - The Enterprise Test Automation Platform, please contact info@odintech.com or David Tracey at david.tracey@odintech.com