Agile Automation Testing

Aman Arora
Adobe Systems India Pvt. Ltd.
I-1A, sector-25A, Noida
aarora@adobe.com

Abstract:

What is Agile Automation Testing? Automation of the test cases done for the Agile project means applying the agile values and principles for doing the automation of the test cases.

The biggest difference between agile methods and traditional method of testing is the short feedback loop. The concept of agility is nothing more, than "build the most important module of the system, evaluate, adjust, and repeat". Effective automation requires thoughtful investment and is a very important tool for shortening the feedback loop.

In agile methods the majority of the automated tests consist of unit tests that verify the smallest possible modules of software and can be executed very quickly. Applying agile values and principles helps teams get traction in starting their automation efforts.

It makes it possible to execute the test set many times a day or even many times an hour and shortens the feedback loop even more.

The paper describes how to apply agile values, principles and practices to develop an automation strategy. Where to start from, how to start, what you shouldn't automate, and where you should proceed with caution this paper describes it all.

Keywords:

Agile Manifesto: The objective of the agility to the software.
Agile Automation: Automating the test cases for an agile system.
Automation tools: Tools used to automate the test cases.

1. Introduction:

What is *Agile Testing?*

Agile testing is a software testing practice that follows the statutes of the agile manifesto, treating software development as the customer of testing.

What is *Automation Testing?*

Manual Testing, consumes a lot of time and so using the automated tools for the testing ensures the lesser time and faster way to test.

2. Agile Manifesto:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

3. Values and Principles of Agile:

The principles that form the basis of Agile Testing are:
Communication
Simplicity
Feedback
Iterations

In the traditional software methodology building the software systems is accomplished through documentation such as SRS, Functional Specification etc. whereas Agile talks about building the software rapidly and to give all the team a shared view of the system. *Communication* of system requirements to the developers and testers is required and this is achieved by small team meetings. No documentation is done as the time is the key factor for all the systems being agile.

Agile encourages in finding the simple solutions for the problems. The most critical but the *simplest* function is taken first and then thereafter extra functionality is added to it.

Feedback is very useful if the system is to be done rapidly. The customer is contacted in more frequent iterations which is unlike to the traditional methods. The customer has a clear insight into the system, for which they provide the feedback.

Agile says that building the entire all at once doesn't works. It encourages the teams to break down the entire system in to small modules which then are completed in small *iterations*. By making small iterations, customers have more control over the system.

4. Agile Automation Testing:

By applying the agile values and principles we can do the automation of the test cases.
Agile approach to Automation:

4.1 Simple:

The author feels that the first and foremost important principle that one should keep in mind is that the tools that are selected for automating the test cases should be as simple as possible. The tools should not be complex that it should take more time to learn the tool than the time to create the test cases.

4.2 Maintenance:

Maintaining the automated test scripts takes a lot of time if they are not prepared well. The tester should divide the long test cases into the multiple smaller test cases which are simple and easily scripted and thus easily maintained.

4.3 Joint effort:

If say there are 10 testers in the team then all the members should help each other in finding the quick solution for the problems they are facing. Knowledge of one can help in solving the problems of other team members. Now, if the developers are also included in automating the test cases then an extra view point is added that can solve and unleash many problems and can give new and efficient methods for resolving the problems.

4.4 Short Iterations:

The entire project is divided into smaller modules, which then depending on the user requirements are prioritized. Time frame for each of the module is set and the development of the most critical module is then started, making sure that a standalone working module is ready at the end of the time frame. This sort of iterations helps in choosing the right tool for the automation such as while automating the unit test cases JUNIT tool can be used and then while automating the functional test cases WATIR can be used.

4.5 Knowledge of Tools:

Lots of tools are there in the market that can be used to automate the test cases, but it is very important to have the knowledge about the automation tool that best suits your requirements. The requirement for the tool may differ depending on the size of the project, time allocated for the project or the cost for the project.

Either the entire team of the testers should be well versed with the knowledge of the different tools that can be used for testing or a person should be added to the team which is well versed with the tools.

5. Agile Automation Strategy:

We cannot reach to the Agile Automation unless and until we are following an agile approach to the development. The strategy for the Agile development starts with breaking the entire project into the smaller modules and then prioritizing them on the basis of the criticality in terms of the impact on the customer's business.

5.1   Where to start:

Test automation creates very good results when employed with effective development practices are in place. The team needs to focus on one story at a time, building functionality step-by-step. Coding must be driven by both technology and business-facing tests as a collaborative effort of customers and developers, so that the right features are delivered on the first try. Continuous integration running a robust suite of unit tests is a first step towards automating other tests.

It is very important that the right tool is selected for the automation as it will simplify the road for the maintenance of the automation ahead. The developers and the testers can together work on creating a new tool which suits their needs. Again, the development of tool will be based on the Agile values. The functions or objects that are needed to automate the test cases for that particular increment is taken into consideration first and then the others.

This helps the tester to build a tool as per their requirements and do not have to compromise with the shortfalls of the pre-existing tools in the market. If, in market there is already a tool that can be enhanced somewhat to meet their requirements then it is a wise decision to go for that tool.

The test cases written for the first module being developed should be the first customer for the automation tool.

5.2   What should be automated?

A million dollar question is what all should be automated and where to start. Any tedious or repetitive task involved in developing software is a candidate for automation. Such as the

Build installation, everyday one has to download and install the build which in turn takes nearly 2-3 hrs on an average. Automating this works saves a lot of manual productive time.

Manual unit tests don't go far towards preventing regression failures, since performing a suite of manual tests before every check-in would not be very practical. So the regression test cases should be automated as it will help testers to work on other priority modules and need not to give their time in testing the same things on every check-in.

Some types of testing can't be done without automation. Manual load tests aren't usually feasible or accurate. Performance testing requires both monitoring tools and a way to drive actions in the system under test. You can't generate a high-volume attack to verify whether your application is capable of handling a large load without some tool framework.

All the legacy areas or modules should be automated which are less prone to changes, so as to save the manual effort in testing those areas.

5.3 *Selection of right tool:*

The selection of the right tool is very important in order to save your time and money both. It is not necessary that the tool that works best for the unit-level test cases will go good for the functional test cases. Basically the selection of tool depends on many things such as System requirements, costing, and approach to the automation etc. By the approach to automation the author means the framework for the automation.

Performance, Load, Security testing may require different tools for the testing.

5.4 *Proceed with Caution:*

Once the automation gets started addition of new test cases to the automation becomes very easy. Now there are few things that the engineers automating the test cases need to keep in mind, they should not just go on automating the stuff without knowing the usability and importance of the test case. One should not automate the area which is more likely prone to changes, as that will result in the rework.

5.5   *Not everything is automatable:*

Some people feel and think that it is very easy to automate everything but it's not true as automation has its own shortcomings. Sometimes tool that you are using is not meant to verify some test scenarios but is very good for the other scenarios depending on the situations one has to take the decision what all can be automated. For example a tool that we use in our company is capable of verifying all the actions performed in the application but is not good for verifying the colors. All it can do is that compare the color values to the baseline, now if there is a change in the representation of the color for some specific value, where value remains same ie it is a bug then it can't be found by the automation as the tool is only comparing the values and the color representation. There are some test cases that can't be automated at all, as they need human intelligence and eyes to verify. And for some there is no need of automation like the test cases that will never fail, as there is always a cost behind the automation.

5.5.1   *Adhoc Testing*

Adhoc testing is to learn more about the product by doing random testing and not following the documented pattern of the testing, and then use that information to improve future development. Automated scripts won't do that for you.

5.5.2   *Usability Testing*

Real usability testing requires someone to actually use the software. Automation might be helpful in setting up scenarios to subsequently examine for usability. Observing users in action, debriefing them on their experiences and judging the results is a job for a person who understands usability aspects of software cannot be automated.

5.5.3   Tests That Will Never Fail

There are few test cases that will never fail. If a requirement is so obvious that there's only one way to implement it, and no programmer will ever look at that code later without knowing exactly what it should do, the chances of someone introducing a defect in that code are next to nothing. For e.g. for selecting an item there is a radio button, now it is not possible that the radio button will be removed by someone accidently.

If you feel comfortable that one-time manual testing does the job and the risk of future failures doesn't justify automating regression tests, don't automate them. If you aren't sure, and it's not terribly difficult to automate, go for it.

6. Conclusion:

If the management wants to have agility in the software being developed then they should make sure that there is Communication among the developers, testers and users. Smaller iterations of the product should be practiced; emphasis should be given to the feedbacks. It has already been discussed by the author that 100% of the automation for the test cases for the agile software is not possible but a maximum number of the test cases automated can be achieved in a very less amount of time and effort if the right selection of the automation tool is done and if the team works on the principles of the agility.

References

[1] http://agilemanifesto.org/

[2] Bach, James: Agile Test Automation, White Paper

[3] Martin, C., Robert and Martin, Micah: 2007. Agile Principles, Patterns and Practices. India: Dorling Kindersley Publishing: licensees of Pearson Education, South Africa.

[4] Gregory, Janet and Crispin, Lisa: 2007. Agile Automation Solutions.