



## THE CUTTING EDGE TEST AUTOMATION PLATFORM



## **Axe – Beyond Productivity, Improving Quality**

Author: David Tracey  
Managing Director  
Odin Technology Ltd  
[david.tracey@odintech.com](mailto:david.tracey@odintech.com)  
[www.axetest.com](http://www.axetest.com)

## **Copyright Notice**

Copyright © 2010 Odin Technology Limited

All rights reserved. All text and figures included in this publication are the exclusive property of Odin Technology Limited or its licensors, and may not be copied, reproduced or used in any other way without the express permission in writing of Odin Technology Limited.

This document may also contain registered trademarks, trademarks, service marks and/or trade names that are owned by their respective companies or organizations. Odin Technology Limited disclaims any responsibility for specifying which marks are owned by which companies or organisations

## Table of Contents

Background .....	4
How Axe improves Quality beyond Productivity .....	6
Accuracy.....	6
Timeliness .....	7
The Testing Delivery process .....	8
Scalability.....	9

## Background

This document does not focus on the productivity benefits of Axe over traditional automation or manual testing methods but a quick re-cap never goes amiss! These include:

1. The ability for manual testers with no automation expertise (we expect an ISTQB foundation or similar knowledge through experience though) to be able to produce both automated scripts and the synchronized documentation after one day Axe tester training
2. To allow the best practice elements of testing to be encompassed in the approach and process used within Axe. These include the use of day-to-day tools deployed in testing such as Excel. The ability to rapidly expand the coverage of test cases (by copy, paste and amend) and to evidence this productivity by the production of standardized test cases by all testing staff using Axe.
3. Axe standard documentation allows ease of use and understanding by all parties involved in the business change process including business users, programme/project management, testers and developers
4. Axe allows productivity improvements and ease of integration with other test execution tools (without having to learn the tool) and test management tools. This means that documentation and results produced by Axe can be loaded into the test management tool to dramatically reduce any possibility of duplication or omissions
5. As a rule of thumb, Axe is four times more productive than standard automation approaches and five times more efficient in terms of maintenance. These figures have also been exceeded on many occasions in building large scale automation packs

This document has been produced to explain to clients the added Quality benefits which are delivered as a result of using the Axe automation platform. Traditional automation approaches have employed technicians to produce scripts from documented test requirements, Axe as a new generation platform has re-written the rules of test automation and ensured that the gaps left from a quality perspective by traditional automation approaches have now been closed.

The reason behind this is that the Axe platform takes a “holistic” approach to test automation, combining the best of testing practices and the best practices of software engineering.

Let’s consider the weaknesses of the traditional approach to test automation. The first lies in the process itself which simplistically, is as follows:

1. The business defines its business requirements to both development and the test teams, the test team define test conditions from these
2. The development team build the code for the application (to be tested)
3. The test team produces manual test cases and awaits delivery of the application
4. The test team gives the test automation specialist the manual test cases to build the automated scripts
5. The application is tested using both manual tests and the automated scripts
6. The testing report is compiled and delivered!

These are logical steps followed by generations of testers, developers and automation specialists. The problem is one of communication and consistency.

First of all, the gap between steps 1 and steps 2 & 3 cause the initial problems. The business never really gets to understand what will be tested and developed to meet their business requirements. The application is developed and “tested” and then defects corrected, new business requirements added and finally re-tested. Rarely if ever does the business actually get the opportunity to sign off the Test cases as covering the scope of the business requirements and even if they did get the chance, inconsistency of style and level of detail would surely prove wasteful and nigh impossible for an application of any magnitude.

***Axe solves this problem by employing a single point of test case definition ensuring that as well as producing automated test scripts it also creates “word perfect” test case documentation***

Secondly steps 2 & 3 cause problems in terms of delay waiting for the application to be built. Too many fundamentally flawed approaches such as capture/replay mean that test case automation require the application to be in place. Furthermore, the tester has little ability to define “acceptance criteria” as a quality gateway to the developer. The result is excessive delays; testing gets compressed and overall less testing is completed than would normally be required from a risk perspective.

***Axe allows the building of documented and scripted test cases before the application is actually delivered giving the ultimate “Agile” platform. The ability for the Tester to “export” acceptance criteria to the developer becomes a possibility further enhancing the Quality gate between phases.***

Steps 3 & 4 really are the “breeding ground” for defects! If you take two testers and ask them to produce test cases for the same application, there is unlikely to be any similarity between what is produced. Even with standard templates the deliverables are inconsistent. Odin Technology appreciates why this is and how it happens and offers some reasons why:

1. Different writing styles (verbose vs. concise)
2. Different understanding and knowledge of the application (Tester are famous at keeping knowledge (inadvertently) to themselves)
3. The differing levels of granularity of the test script (Tester 1: Login with User100 Vs. Tester 2: Position on login screen, enter 8 digit user id followed by 4 digit pass code). If you imagine the number of defects that can be produced in a complex integrated test case!
4. Different origins of the test team including language and geography (This even happens in small regions such as the UK)
5. The automation specialist is normally not a “tester mindset” and is more akin to a developer’s approach, assumptions are made and the script produced. This is where the most dangerous problem lies; the only person (or types thereof) that can decipher what has been written is an automation specialist. In our experience (almost 100% of the time) the end deliverable is different from the original test conditions defined

***Axe minimises the number of levels of passing of the information to the extent that the tester actually delivers synchronised scripts and documentation by using a simple test definition language implemented in Excel. One hundred percent of the time, the script***

***reflects what the documented test scripts are and more importantly this test case can (because it can be readily understood) be signed off by the business.***

Finally but not least, the business, project manager, test and development teams are delivered the results of the latest cycle of the tests. This reflects the potential gap between steps 5 & 6.

This is where we find excessive time has been spent to collate results, raise defects and in terms of time, the delivery of the cycle's results are often late too (remember testing is always late and compressed, this just makes it later). For most applications, complex integration testing is required but reporting is inconsistent, cumbersome and late.

***Not only does Axe produce a fully auditable documentation set for test cases, its reporting also reflects the same level of granularity as the test case documentation. Hence defect resolution is available in a “drill down” HTML fashion and reporting is produced in an English format and available as soon as the test cycle is finished. This end product can then be supplied in a fully documented, auditable and cohesive fashion to the business user (whether internal or external).***

## How Axe improves Quality beyond Productivity

If we take a dimensional approach to Quality, there are a number of areas where Quality is affected. This section outlines the way in which Axe minimises the chances of failure or risk in these areas. For the sake of discussion (these can be re-classified or re-defined if necessary), Odin believes the following areas have a direct (usually derogatory) effect on Quality:

1. Accuracy
2. Timeliness
3. The testing delivery process
4. Scalability

### Accuracy

Accuracy is of course the domain of the tester! However where Test cases are not confirmed or parts (components) are not reused then inaccuracy becomes the result. In Manual testing primarily everything is knowledge based, the critical aspect is whether or not the knowledge is actually documented or the understanding of what is right is actually confirmed.

As outlined in the previous section, if we had a team of ten testers, the quality of the documented test cases would differ. Strangely enough the quality of the execution of the testing could also differ but not necessarily in the same way as the documentation but because of the level of application knowledge of the tester.

The fact therefore that a Quality issue could arise due to a lack of knowledge in the application that has not been documented means that it becomes critical to ensure that the knowledge (even the smallest step's omission could be catastrophic) is captured and encapsulated in the documented (and executed via the synchronised script) test case.

In addition to ensuring that the level of the test cases are consistent and more importantly at the lowest level of granularity, Axe also ensures that a single test case reflects a single business process variant.

If we had for example a number of data variants for a part of the test (e.g. Salary) and later on in the test had other variants (e.g. Age), the business user can be furnished with completed test cases for each variant required to meet the business requirement.

In summary on accuracy, Axe ensures:

1. Standard test case notation and structure. This means that a companywide reporting and documentation standard can be introduced straight away. Axe also allows these reports to be customised (add logos etc) easily.
2. Axe produces a test case for each variant (i.e. data) required. Axe does employ programmatic constructs such as iterative looping in its test definitions, therefore taking the simplistic one-to-one approach to test cases, everybody understands what has been tested and if a test case fails, which of the variants failed.
3. Because Axe produces a full documented test case and subsequent reporting, clear responsibilities for sign-off can be implemented.

## Timeliness

Axe offer clients significant opportunities to “save time” and uses time more effectively. Frequently, testers complain that due to late delivery of application builds, they are often compressed in terms of the time to deliver the real level of testing required.

As Axe is an automation platform, promoting best practices like error recovery and basestates, the automation can be run (unmanned) out of hours or on weekends, making the prospect of 24/7 testing a true reality. This means that the testers are able to focus on testing the new requirements during the working day and running regression packs out of hours. Axe is designed to allow the testers to group or order tests for a given purpose building new groups as and when required (e.g. targeting a given functional area) using Excel spreadsheets and without the need to build technical driver scripts.

One of the striking features of Axe is its ability to operate in a truly “Agile” way. Using Axe, the testers are able to build their test cases, documentation and scripts without the application actually being present. By this we mean that from the design documents, object names can be defined in Excel spreadsheets, the templates for the tests generated, populated and the test case documentation produced for sign-off.

When the application is delivered, object mapping can be completed and the application can be tested automatically as soon as it is delivered to the test team. This “parallel” approach means that testing (preparation for automation) can actually start in parallel with the application development. It is generally considered that test preparation takes up to 90% of a tester’s time, execution being the remainder. It can therefore be seen from this facility alone that significant

tester time can be saved but more importantly the tester now has the tools to do things properly from a Quality perspective.

The productivity gains in using Axe are not the subject of this document, however a by-product of the speed in which test cases can be produced and maintained is worthy of comment in this section.

In summary on timeliness Axe ensures:

1. Axe makes the most of the tester's day by allowing test cases to be run "out of hours" unmanned. Thus increasing the testing throughput to a near 24/7 state if required.
2. Test case preparation and documentation can be started after the application design phase. Thus the building of the automated test cases can start in parallel with the development process.
3. Test case documentation can be produced early and signed off by the business user prior to the start of the test execution phase. Where a defect exists in the test case, this can be quickly rectified and new documentation produced.

## The Testing Delivery process

Axe brings with it a self-contained testing delivery process. This means that all of the reporting required to standardise this process is delivered "out of the box". Axe uses flexible tools to define these reports and these can be amended by the test team to change layout, colours and content.

From a qualitative perspective Axe delivers:

1. A fully documented test case index (reflecting the scope of the tests to be run).
2. A detailed document for each test case defined (tests are uniquely identified by a test case ID). It is important to stress here that the level of detail is the lowest level possible (interaction with an object in the case of UI automation). The language used by Axe can easily be amended to create the standard terminology required by a specific client.
3. A statistics report showing the number of times an "object" is referenced in the test cases to be run (thus showing an alternative view of coverage).
4. A full drill down report (down to the individual object action level). This gives timings at each step, subtest and test case to give the tester the ability to do non-functional indicative testing as well as functional testing. Thus from the qualitative perspective, run time defects can be easily isolated based on these timings (e.g. A service level agreement that login should take no longer than 10 secs).
5. Project reports covering for example all the runs of test cases against a particular application showing the failure numbers for each test run instance
6. Axe is based on industry standard formats and uses XML and HTML at the heart of its reporting. This means that Axe can integrate through API's with the major test management tools. From a quality perspective this is unique and allows the generated test cases to be imported into test management tools quickly and accurately. As well as having this feature, Axe also ensures the scripts and documentation are synchronised at all times, something which is rarely, if ever, the case in our experience.

## Scalability

The history of automation projects has been plagued by the inability to scale. This is due to the approach deployed in traditional automation projects which have been primarily technical in nature.

Axe was built with best practice of both the testing process and the automation process in mind. Because Axe uses “Tester friendly tools” such as Excel, this means that technical skills are not required by the Axe tester to achieve higher quality automation than the traditional approaches can deliver.

Furthermore Axe generated scripts are disposable and are regenerated as/when required. This means that the maintenance of scripts is performed in the Excel spreadsheets rather than having to maintain scripts (code). In addition of course, Axe produces the English representation of the script as detailed test case documentation, something which previous generations of automation methodologies failed to deliver or produce.

All of this means Axe becomes a very scalable approach. From a qualitative perspective, Axe enables an environment where test cases can be quickly built from reusable (and tested) components and can be easily replicated and changed to meet new business requirements.

Coverage thus becomes simple. By using productivity features, e.g. copy and paste in Excel, testers can produce large numbers of test cases quickly ensuring that re-use is a key component of this and from a quality perspective that the documented test case gives everybody the reassurance that a particular business requirement will be tested correctly.

So in conclusion, Axe allows large and complex test packs to be produced, backed-up by the fact Axe produces a manual documentation and the reporting that underpins the technical scripts. Furthermore, Axe also allows the ease of portability (spreadsheets) across geographical locations if/when required and from a technical perspective it allows the client to change test tools easily with the minimum of effort to migrate.