

**The Great  
SOFTWARE TESTING  
SWINDLE**

**A Whitepaper.**



Original Software

## The Great Software Testing Swindle.

### Abstract

Since the first automation tools appeared on the horizon over 20 years ago, software test automation has been heralded as one of the saviours of IT and IS development teams the world over. During that period, many millions of dollars have been spent on software test automation - indeed, according to IDC, spend on software automation is over \$1,000,000,000, and is destined to keep on growing.

At the same time, research carried out by Original Software consistently indicated that a maximum of 20% of the total amount of testing is carried out by QA and testing departments using automation tools. Indeed, many development teams carry out zero automation of their testing - choosing instead to run a completely manual process.

In the last 20 years we have seen breath-taking advances in technology in all areas of our lives which has simply not been matched by the software testing industry. The vast majority of tools available use scripting language which is fundamentally limiting, requiring specialist technical skills which can be difficult to find and expensive to hire.

If you bought a house that had only a coal fire in the lounge, you would immediately install central heating. A much more efficient and cost effective way of heating that covers the whole house, not just one room.

Shouldn't this be the same for software testing tools? Shouldn't you insist on the newest technology that offers the most testing coverage with minimal maintenance?

This whitepaper blows the whistle on the Great Software Testing Swindle. We will be asking why the process of software testing continues to be the major bottle neck of application development and we will look at the antidote - next generation software testing solutions that by their unique design enable testers and QA departments to automate significantly more of their testing, and empower their manual testing - thus eliminating the testing bottle neck.

Areas we will cover include:

- The Rise and Rise of Software Test Automation
- Why is Software Testing Always the Bottleneck of Application Development
- The Limitations of Traditional Software Test Automation Tools
- The antidote. Next Generation Software Test Automation

**Swindle:** Despite the proliferation of automated software over the last 20 years, in 2007 around 80% of all testing will still take place manually.

## The Rise and Rise of Software Automation.

In their most recent market research report, IDC expects the market for Automated Software Quality Tools to almost double in the five years from 2005 (\$947.6m) to 2010 (\$1842.1m) citing increasing software complexity and the increased attention to corporate compliance as two of the key market drivers for this rise.  
(IDC Market Analysis: Worldwide Distributed Automated Software Quality Tools 2006-2010 Forecast and Vendor Shares. Dec 2006)

So there you have it. More and more people each year are buying these tools. So they must be working, right? They *must* be having some kind of affect otherwise people just wouldn't buy them?

Wrong.

Lets look at one such example.

We recently spoke to a large USA financial company (who shall remain nameless) about their software testing requirements. Back came the reply; "Our automated software testing requirements are fully satisfied - we have just spent \$1m on a large amount of tools to do this."

After a small amount of persistence we were able to understand a little bit more of the detail about the very impressive sounding "complete solution" they had bought:

- The "solution" was purchased almost two years previously and had taken this long with 6 people in the set up team to get everything ready for testing.
- The automation could only work on a small stable area of the application.
- Only 5% of their testing was covered by this \$1m solution because it was too slow and complex to use it on applications that change frequently.

So after a \$1m investment, and two years of implementation they had a software solution that was able to automate just 5% of their total testing - and this was the 5% that was stable and relatively risk free. The rest still needed to be tested manually, and there was no mention of testing at the database level. What a poor investment.

A million dollars is a significant investment, and it is easy to see how the market is growing so rapidly with these kind of implementations. However this is an all too common scenario - thousands of companies have been convinced that after such an investment, automating 5%, 10%, 20% of the software testing is a great return. This is simply not the case and is a big part of the **Great Software Testing Swindle**.

**Swindle:** \$1m investment, 2 years to implement and a 5% automation coverage is a common scenario but NOT a clever Return on Investment.

## Why is Software Testing The Bottleneck of Application Development?

Before we consider the issue of bottlenecks, let us understand what the business really wants from IT. In a nutshell it comes down to three simple things: reliable applications that are error free, are delivered on time and enjoy a positive return on investment.

1. Reliability
  - Is the software robust (error free), and fit for purpose?
2. Delivered on Time
  - The opportunity window is “now”... don't miss it
3. Cost/Value Balance
  - Positive Return on Investment while delivering points 1 & 2

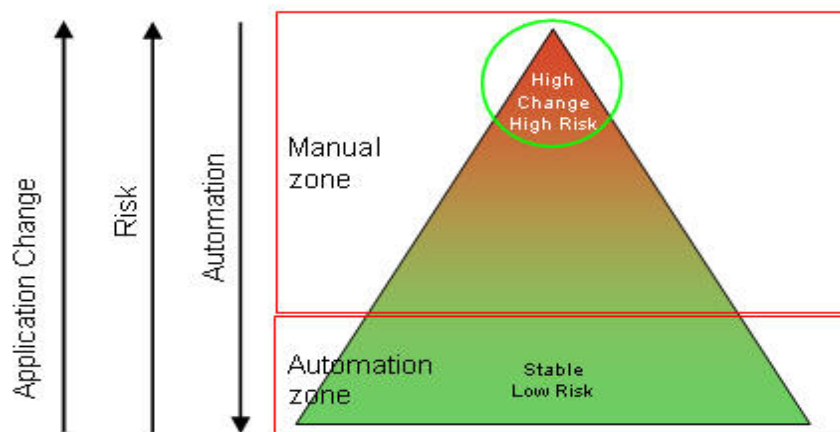
From this we can see that the prospect of automating software testing is an enticing one. If you can get a computer program to test everything then this will reduce errors. If the same program can test it more quickly then the delivery schedules will improve. Shorter delivery time scales and higher quality applications will in turn provide a positive return on investment. No problem.

Well actually there is a problem.

The reality is that only a small percentage of testing is subjected to automation. These areas tend to be the most stable areas of the application that carry the lowest risk to the business. In some cases, if applicable, stress or load testing is also carried out.

But what happens to the areas being changed, and therefore carry the risk for the business? In 2007, (the age of satellite navigation, Internet ready mobile phones, and intelligent windscreen wipers), such critical business application software is still being tested manually. And as we all know, the repetitious nature of manual testing leaves it prone to errors over time.

**The Risk/Change Triangle.**



**SWNtd:** Traditional “old style” software testing tools can only be deployed for testing applications that are stable and relatively risk free. They are unable to cope with high change high / risk areas, the areas that *really need* comprehensive testing.

## The Limitations of Traditional Software Test Automation Tools.

So those million dollar automation solutions are used only to test the most stable areas of the application, those areas that shouldn't have changed in the first place. Why can't they be used everywhere? Why are they limited to the most stable and low risk areas?

Here is the answer:

```
Sub Main
Dim Result(50) As Integer
Dim i as Integer
Dim NewResult as String
StartBrowser "http://pandora.ple.blahblah.co.uk/action.pegaf1000=SIGNON&profile=test3", "WindowTag=WEBBROWSER"
Window SetContext, "WindowTag=WEBBROWSER", ""
Window WMaximize, "", ""
delayfor 3000
Browser setFrame,"Type=HTMLFrame:HTMLId=__pegMainFrame", ""
Browser NewPage,"HTMLTitle=PANDORA - TEST1", ""
Result(1) = TextBoxVP (CompareProperties, "Type=TextBox;Name=f5p1", "VP=Object Properties;Wait=2,30")
Result(2) = TextBoxVP (CompareProperties, "Type=TextBox;Name=f6p1", "VP=Object Properties2;Wait=2,30")
Result(3) = TextBoxVP (CompareProperties, "Type=TextBox;Name=f8p1", "VP=Object Properties3;Wait=2,30")
Result(4) = TextBoxVP (CompareProperties, "Type=TextBox;Name=f9p1", "VP=Object Properties4;Wait=2,30")
Result(5) = PushButtonVP (CompareProperties, "Type=PushButton;Name=@ACTION=|ENTR", "VP=Object Properties5;Wait=2,30")
Result(6) = PushButtonVP (CompareProperties, "Type=PushButton;Name=@ACTION=|APPR", "VP=Object Properties6;Wait=2,30")
Result(7) = PushButtonVP (CompareProperties, "Type=PushButton;Name=@ACTION=|BACK", "VP=Object Properties7;Wait=2,30")
For i = 1 to 3
Select Case i
Case 1
InputKeys "robir"
PushButton Click, "Type=PushButton;Name=@ACTION=|ENTR"
```

These old style tools are still all based on specialist **scripting languages** that are complicated and haven't really changed much in over 20 years.

There are a number of fundamental flaws with this approach:

### 1. Specialist workforce needed.

It requires a very specialist skill set to work with, write, and manage these scripts. The people with these skills are often difficult to find and expensive to hire. They also need regular training to keep up to date with new techniques.

### 2. Lengthy set-up times.

Creating your automated testing scripts using these tools is very cumbersome and complex. It can take a team of people months and even years to set up properly. Not particularly useful if the application is time critical.

### 3. Maintenance nightmare.

Every time the application under test changes, the scripts that were written to originally test the application have to be re-written for the changes. This is a time consuming task and can often take longer than manually testing the application in the first place. This high maintenance burden is one of the big reasons why so much automation software is just gathering dust on shelves. As soon as the application changes the tools are useless, unless significant time and resource is thrown at them.

### 4. Limited usefulness

This lack of flexibility is the reason why such tools are not used on the high risk applications that are constantly being changed or updated. QA teams simply cannot cope with the continual manual intervention needed to re-write scripts.

**SWND:** In this age of rapid technological advancement in all aspects of our lives from MP3 players, mobile phones, and satellite navigation, why is it acceptable to use a software testing approach that is 20 years old?

## The Antidote: Next Generation Software Test Automation Solutions.

After 20 years of treading water, it is about time the software testing industry hauled itself into the the 21st Century.

That time is now.

Next generation software test automation solutions do not depend on those old scripting languages. Therefore:

### No scripting language

A more diverse audience within your business can use them. Test automation is no longer confined to programmers and scripters. Graphical scripts are built based on the user's interaction with the system under test.

### Speedy Implementation

Gone are the many months of building and preparation before a single item has been tested. With next generation solutions, you can be testing in days, not months and years. This means you can utilise your investment earlier.

### Scripts that update themselves when applications change

The intelligent technology behind these solutions mean that they automatically recognise when an application interface has changed and the scripts are updated to reflect the changes. This means that the solutions are completely re-usable, no matter how often the application changes. As a result, the automation zone is no longer confined to those areas of the application that are stable and risk free.

### A helping hand for manual testing

Solutions specifically designed to streamline manual testing can give testers a helping hand on all those applications where automation is not yet appropriate.

### Broader Testing Scope - Database testing

Data underpins the entire testing process. Poor data will result in poor testing. With growing emphasis being placed on data protection, it is vital that the integrity of your test data is protected at all stages of your testing project. The next generation of solutions can ensure your data is in A1 condition all the way through the testing process.

All of this for browser, GUI or Green screen applications.

|  |  |  |
|--|--|--|
| <br>Browser     | <br>GUI       | <br>Legacy      |
| <br>Self Heal   | <br>Code Free | <br>Action Maps |
| <br>Data Driven | <br>Integrate | <br>Report      |

**Next generation software test automation tools are the only way to improve automation coverage and ROI.**

## About Original Software

Original Software offers next generation automated software testing and quality assurance solutions that deliver tangible benefits across a wide range of IT and application environments. As a recognized innovator, Original Software's goal is to reduce business risk and improve application time to market for IT departments through the development of class leading automated solutions.

Over the last 10 years, more than 400 organizations operating in 25 countries have come to depend on Original Software for their software testing solutions. Current users range from small software development shops to major multinationals, including: Cargill Global Financial Solutions, Circuit City Stores, Pfizer Pharmaceutical, BP, DHL, Coca-Cola, Skandia and hundreds of others.

Original Software operates central offices near Chicago, and London. Their solutions can be obtained through these offices or through a network of qualified and knowledgeable business partners throughout Europe, the Middle East, Australasia and the Americas.

**[solutions@origsoft.com](mailto:solutions@origsoft.com)**  
**[www.origsoft.com](http://www.origsoft.com)**



**Original Software**