



# ***ANarchy* v *The* QA**

**The Ongoing Battle to Improve Software Quality.**



**Original Software**



**Ā'narch|ŷ n. absence of government in a society; political or social disorder; confusion as to how to properly test software quality in the face of corporate lethargy and over hyped, under performing automation tools.**

### **Introduction**

There is unrest brewing in IT QA departments all over the globe. A frustration borne out of a lack of recognition within their own organization as to the importance of software quality, the potential competitive advantage their company could enjoy if they get software quality right, and the major negative impact that can result from poor software quality processes. On top of this, the major proponents of software test automation have repeatedly failed to offer a reliable solution, thus those QA departments that have been lucky enough to secure funding to improve their software quality through automation, have been let down by their vendors (often multi-national giants that really should know better), and have often failed to deliver positive ROI.

More often than not, even after substantial investment in software test automation, these QA departments are no better off than they were before they spent the money. No wonder then, that in a recent survey of CIOs, Original Software found that only 6% were totally happy with their automation investment.

This whitepaper will look at the issues surrounding software quality in more detail, try to understand corporate attitudes and pinpoint the distinct advantages to getting software quality right (and the problems that can arise if things go wrong). Finally, we will discuss a revolution brewing in software quality. A revolution that is gaining momentum, and is forcing people to re-think their attitude towards software test automation.

**"Anarchists know that a long period of education must precede any great fundamental change in society, hence they do not believe in vote begging, nor political campaigns, but rather in the development of self-thinking individuals."** Lucy Parson, Chicago Revolutionary (1853-1942)



## **Anarchy in the QA**

Times are tough. You know it, I know it, your competition knows it and your boss definitely knows it.

With greater competition, more consumer choice and a greater dependency on technology than ever before, every organization faces the challenge of building quality IS systems to support rapidly evolving business requirements. New applications need to get out the door quicker than ever, and you know that the new application upgrade will be hot on its heels. Time is money and IS departments are constantly looking for ways to cut down their application time to market. On average 40% of total application delivery time taken up with testing, so is it no wonder that this is an area marked out for special attention.

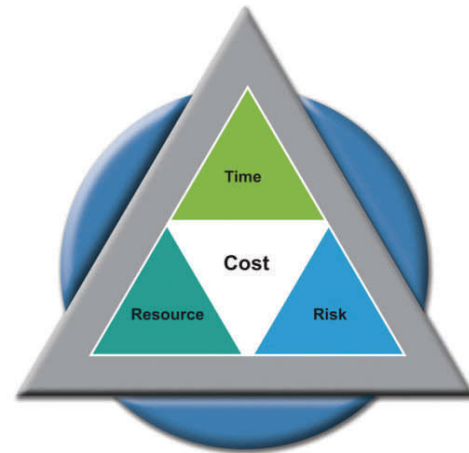
Why does testing take so long? Is the question often thrown at the QA department. Well it's a highly manual process — even though some departments may use automation tools, it is still a time consuming activity. There is a skill in balancing the required resources, with time pressures, and software quality. Original Software refers to this as the Quality Conundrum, and it is something that IT management need to get to grips with. And fast.

You want better quality software? Right, well it will take more time to get right. You want the application live next week? OK, but we will either need to double the team of testers or we will only complete 75% of testing and the quality will be compromised. We don't have any more budget for more resources? Well there is a decision to be made between an on-time application or a high quality application—you can't have both.

A familiar conversation? **This is the Quality Conundrum.**

And this is not all. It seems IT departments globally are in need of education as to the importance of software quality. In IDC's recent Software Quality Survey 2008, 72% of respondents described their de-bugging process as "problematic", but paradoxically 62% said their defect management and testing approach either "did not require improvement" or it simply wasn't possible to change their approach, despite having issues.

It is this quality conundrum issue, along with perhaps a slightly blasé attitude to software quality from the relevant IT management, that anarchy has manifested itself into the realms of QA departments across the world.



**The Quality Conundrum.**

This widespread corporate lethargy is not going un-noticed. Principal analyst Bola Rotibi, at research firm MWD recently wrote "The attitude of 'good enough' has been hijacked as an excuse for sloppy attitudes and processes and a 'let's suck it and see' mentality. Surely such an attitude cannot continue to exist in delivering software that is increasingly underpinning business critical applications?"\*

QA teams are paid to produce top quality applications, but without the correct backing from their superiors, (in a recent survey carried out by Original Software, 40% of CIO's revealed they view software quality as a "nice to have" or a "non-defined process" in their organization\*\*); adequate automation tools from vendors, and with the added pressure to cut corners from other areas of the business, it is becoming increasingly difficult for these professionals to carry out their duty properly.

QA need help. Application quality is no longer an activity within the software development lifecycle that IS teams can aspire to improve in their own time. It is a business imperative that demands a solution now. It's enough to make you scream.

\*The Dilemma of "Good Enough" in Software Quality—Bola Rotibi

\*\*For more results and full analysis read our CIO whitepaper at [www.origsoft.com/ciosurvey](http://www.origsoft.com/ciosurvey)



## ***The Importance of Getting Software Quality Right.***

Why should we bother? Does improving software quality *really* matter that much?

Way back in 2002, the National Institute of Standards and Technology (NIST) estimated that software errors cost the US economy **\$59.5 Billion** annually, (or about 0.6% of GDP).

So there you have 59.5 billion examples as to why software quality *is* important, not just to the IT Manager, but to the CIO, CFO, CEO, Chairman of the Board, shareholders, and anyone else with a vested interest in your organization (not least your customers). Oh and that figure is a few years old now, these days it's probably higher.

With this reasoning in mind, it was a surprise to us that 40% of senior C-Level executives in our recent survey admitted that they see software quality as a nice to have or as a non defined process. This indicates that too many companies are not taking software quality seriously, at least not at senior management level. Continuing her article on software quality, Principal Analyst Bola Rotibi tends to concur with our thoughts here. "What I find incredible after all this time, given the weight of evidence and eminent studies on the cost savings and the growing complexity and importance of software in our modern lives, is that the "sloppy" mentality and attitude still holds such sway in software delivery processes." Rotibi continues, "Many organizations don't spend nearly enough effort on improving the quality of the software they produce. More often than not they pay lip service to the concept whilst secretly holding the belief that it is a waste of resources (time, staff and money).

Our surprise was backed up even more by a recent IDC survey that revealed that more than 40% of all software applications are released with between 1 and 10 critical defects, and what's more, the management are aware and still let the applications go live!

Software defects in live systems can bring businesses to a halt, and can be very costly even if repaired quickly. When a live application fails, consider what can happen:

- Users may be unable to process business transactions.
- The failing application(s) must be identified, corrected and re-tested.
- If the failure has caused data corruption then the extent of this must be analyzed.
- In the event of repeated failures the users will become disenchanted and lose confidence.

If you replace the word 'users' with 'customers' in the above scenario, the problem reaches a new level of magnitude. A system failure has the potential to affect every prospect, customer and supplier. An unreliable internet application equates to commercial suicide, and online customers are your least forgiving audience. Once customers lose confidence in the application, and that loss of confidence happens very quickly, they will simply seek an alternative source of supply.

Today's business environment means risks sometimes have to be taken to obtain competitive advantage. However unmanaged risks in IS developments can have disastrous consequences. Application failures lead to a loss of external and internal confidence, cause damage to the brand and ultimately can have a negative effect on the revenue stream.



Recently there has been some very high profile incidents of organizations that have suffered due to substandard software applications...

- The launch of Terminal 5 at London Heathrow was thrown into chaos with baggage delays, cancelled flights and check-in difficulties due to a "systems fault".
- The failed IBM ERP installation into American LaFrance that has caused the fire engine company to go into bankruptcy.
- Her Majesty's Revenue and Customs department suffered a £2.8Bn embarrassment this summer due to software glitches.
- A website fault forced supermarket giant J.Sainsbury to shut down its online shop for 1 day, costing it £1.5m in lost sales and compensation.
- A software flaw in a drug pump led to patients being given potentially lethal overdoses.

This list is long, yet it seems organizations are failing to learn from other's mistakes.

So what can be done? What can the QA department, and indeed the IT department as a whole do to ensure any application that goes live is in the best shape it can be?

Test. Test, test, test, and then test some more. This is the painful truth. Unless an application is thoroughly and rigorously tested before going live, there is no way anyone can be sure it is of sufficient quality to be released.

And herein lies the problem. You see testing can take a long time, sometimes up to 40% of the entire application development process can be taken up with testing. So it is not surprising when corners are cut to speed up time to market. But we go back to our friend the Quality Conundrum (see page 2) and discover that if you want your testers to be speedy gonzales, something else will have to give. Either they test less of the application which leaves you open to risk, or much more money has to be spent on it. Either in more physical testers, (but just by throwing resource at the problem is no guarantee that it will be completed quicker), or some kind of automation solution that will help streamline the whole process.

It was in response to this problem that the entire software test automation industry was born. Over the last twenty or so years, millions of dollars have been spent on software test automation. It was heralded as the savior to all of the inherent issues around testing. The quality conundrum would no longer exist because testing would now be automated. All the bugs would be found in a fraction of the time, meaning high quality applications going live in shorter time frames. Easy.

Well, actually...no.

As Public Enemy once said "Don't believe the hype"

You see, automation has largely failed to live up to its billing. There are many reasons for this which we shall look into on the next page, but it is a sad indictment of the industry when only 6% of C-Level executives admit they are completely satisfied with the performance of their software test automation tools\*. Ouch.





## ***The failure of the Automation Industry (By Companies Who Should Know Better!)***

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 just purchased.
- The "solution" was purchased almost 2 years previously and had taken a team of 6 people that long to prepare for their first automation.
- 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 any area of their applications that changed 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. What a poor investment!!

This kind of scenario is all too common, indeed it is very rare to find a completely happy automation user, so it is no surprise that software test automation is viewed with no small amount of skepticism from within the IT fraternity. But why? Where has it all gone wrong?

There are many reasons, but the single biggest issue with these tools are the fact they are based on specialist scripting language. What does this mean in terms of usability and flexibility?

Well on the face of it, it should be a good thing, because by writing your own scripts, you can pretty much configure these products to your choosing. Great. If you are a coder. Not great however if you are not lucky enough to have been trained in this particular scripting language. Although there are great coding opportunities here, there are a number of flaws with this script based design. First off, as has already been mentioned, you need a specialist workforce in place to actually use the tools.

```
Sub Main
Dim Result(S) As Integer
Dim i As Integer
Dim NewResult as String
StartBrowser "http://panorama.jib.jib11111.co.uk/rdm/pegaj?1100=SIGNON&profile=test", "WindowTag=WEBBrowser"
Window.SetContext "WindowTag=WEBBrowser", ""
delayfor 3000
Browser.SetFrame "HTMLTitle=PANODORA - TEST1"
Browser.NewPage "HTMLTitle=PANODORA - TEST1"
Result(1) = EditBoxVP (CompareProperties, "Type=EditBoxName=Sp1", "VP=Object Properties\Wait(2.30)")
Result(2) = EditBoxVP (CompareProperties, "Type=EditBoxName=Sp1", "VP=Object Properties\Wait(2.30)")
Result(3) = EditBoxVP (CompareProperties, "Type=EditBoxName=Sp1", "VP=Object Properties\Wait(2.30)")
Result(4) = EditBoxVP (CompareProperties, "Type=EditBoxName=Sp1", "VP=Object Properties\Wait(2.30)")
Result(5) = PushButtonVP (CompareProperties, "Type=PushButtonName=@ACTION=APPPR", "VP=Object Properties\Wait(2.30)")
Result(6) = PushButtonVP (CompareProperties, "Type=PushButtonName=@ACTION=APPPR", "VP=Object Properties\Wait(2.30)")
Result(7) = PushButtonVP (CompareProperties, "Type=PushButtonName=@ACTION=BACK", "VP=Object Properties\Wait(2.30)")
For i = 1 to 3
Select Case i
Case 1
InputKeys "tab"
PushButton Click, "Type=PushButtonName=@ACTION=ENTR"
```

### **The root of all evil. Code.**

Specialist workers are difficult to find and expensive to hire, plus there is the issue of UAT (user acceptance testing) where other members of your organization get involved to test the application. Unless they go on a crash course in scripting, our guess is they will probably struggle.

Then there is the set up. Let's re-visit our example at the top of the page. 6 people setting up for 2 years! Scripts need to be written, options need to be worked etc. etc. Lets say for arguments sake, each person is on \$50 per hour, that is another \$1.2million on top of the investment to just get it started. Where's that ROI gone?\*



Two years and \$1.2million later, and you are all set to automate! So, just press the “go” button, pour yourself a large Louis XIV, light up a cigar, kick back and watch as all of your testing nightmares melt away in a flash of computer wizardry and dreamy music...

Sorry to be the spoil-sport, but no. No computer wizardry, no dreamy music, nothing. What you do have though is the IT equivalent of a 1980’s Lamborghini. It has cost you a small fortune, and needs continuous attention and constant looking after by a team of specialists to be effective. But even if you offer lots of TLC, it is still prone to break down at any moment. Every time you update your application you will have to re-visit that lovely code and tweak and re-write parts of it so it will be able to continue showing off its silky skills. The upshot of this is that for those areas of your application that are changing frequently such automation tools are very difficult to maintain, with most QA departments resorting back to manual testing in order to cope with the testing. Ironic really, as these are the applications that are at most risk of defects, and are more likely to carry more of a business risk if things go wrong.

It’s this maintenance issue causing a difficulty in re-use, along with a limited scope to test across the whole application which has led to an enormous amount of shelf-ware among traditional automation tools and has undermined the industry as a whole.



**Rěvolu'tion.** *n.* 1. revolving, motion in orbit or circular course or round axis or centre. 2. complete change, turning upside down, great reversal of conditions.  
**3. next generation software testing solutions from Original Software that threaten to overthrow the dominant code based tools with solutions that are easy to use, quick to set up and reusable.**

### ***The Revolution is Here!***

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

Therefore:

#### **No scripting language**

The technology is built in to the solution, not the script, so 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 quality, 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. Brilliant.



## **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.

**[www.origsoft.com](http://www.origsoft.com)**  
**[www.manualtesting.com](http://www.manualtesting.com)**



**Original Software**