

A Tester's Winter Tale

"How the Grinch Stole Scripting"



Original Software

Once Upon a Time

*Every Tester in Testville liked testing a lot.
But the Grinch, who sat north of Development - did not.
The Grinch was a Tester and he hated the whole damn lot!
Now, please don't ask why; no one quite knows the reason.
It could be, perhaps, that his shoes were too tight.
Or it could be that his head wasn't screwed on just right.
But I think that the most likely reason of all,
May have been that he sat alone near the wall,
Testing all night!
He looked such a fright!
Throwing quality away as the changes came again.*

*And there it sat on the shelf collecting dust.
Test Automation, "Oh what a fuss!"
How it failed to deliver him from his testing damnation.
So he finished his email with a large exclamation: "No not
another software release!"
Staring down at his desk in QA with a sour grinchy frown,
He observed busy Testville and its Testers in town.
For he knew that every Software Tester was busy right now,
testing the winter release.
Would there ever be peace!*



Software test automation was once heralded as a saviour within IS development teams and was supposed to ease the pains of software testing. It promised to release the QA or testing bottleneck...but in reality, it hasn't!

As painful as test automation was to implement, the use of traditional testing tools means that a specialist workforce is also needed, with specific skills to write and manage test scripts. Every time the application under test changes, the scripts that were written to originally test the application have to be re-written to accommodate the changes. This high maintenance burden is the single biggest reason why test automation software has failed to improve the software

“And there it sat on the shelf collecting dust. Test Automation, Oh what a fuss!”

testing process and is now being rejected or left to gather dust on a shelf.

Test automation that is confined to programmers and testers who code, is restricted to stable unchanging applications which are low risk. Attempting to keep up with application changes in the more dynamic and inherently higher risk areas of the application, means re-writing and maintaining test code – this is a testing nightmare. But it is these high-risk areas which really need the most comprehensive testing.

For a business to gain competitive advantage, deadlines need to be met and risks sometimes have to be taken. Unfortunately, businesses gamble on software quality mainly because of being under resourced. One need only check the IT press for examples of high profile companies whose business has been disrupted, due to software failures. Not having a broader testing scope and spending too much time manually testing unstable areas of an application is a hazard that could have disastrous consequences. The resulting application failures lead to a loss of external and internal confidence, cause damage to the brand and ultimately can have a negative effect on customer retention and revenue streams.

With a focus on test automation software, this seasonal article borrows the story of the Grinch* and highlights the challenges of software testing and the damage that can be done if the business gambles with software quality. It attempts to outline the main benefits of test automation in conjunction with pragmatic advice on software testing, whilst illustrating the negative impact that current market-dominating testing tools have had in the application quality lifecycle.



Part 1...

*You're a mean one, Mr. Grinch. You really are a heel.
You're as cuddly as a cactus;
You're as charming as an eel.
Mr. Grinch!
You're a bad banana with a greasy black peel!*



The Grinch can be identified in the QA team as a frustrated and miserable software tester who is isolated from the other areas of the business. In our experience, this transformation from tester to Grinch occurs when there is a poorly executed software quality process. Frustration born out of a lack of recognition within the organisation as to the importance of software quality also plays a large part. Test automation tools serve to fuel this dissatisfaction, since the vendors who sold them, especially one who has dominated the market for years, have failed to deliver the required Return on Investment (ROI) that the business demands.

Why Did Test Automation Fail?

The business requires IT to deliver these main values:

1. Software that is aligned to meet the demands of the business.
2. Software that is reliable and fit for purpose.
3. Software and business applications delivered on time.
4. A balance between value and cost of delivering the software.

So the prospect of automating software testing is an enticing one. If tools can be programmed to test the application quickly and with minimal intervention, delivery schedules would improve and higher quality business applications will in turn provide a positive ROI. But in reality, and research has shown, that only a small percentage of testing is subject to automation, i.e. The areas that are most stable.

Test automation software that has failed to deliver real automation has dominated the market place for the past decade. It is this dominance that has inevitably caused it to stay the same. The problem with the software is that it scripts requires a very specialist skill set to create, adapt and manage. Although there is a place for coding within testing, there are also a number of flaws with a script-based design. As has already been mentioned, you need a specialist

"...transformation from tester to Grinch occurs when there is a poorly executed software quality process."

“Some software testers who are reading this paper may even admit to being just like the Grinch, especially having shared the same test automation frustration.”

workforce in place to actually use the tool. Specialist workers are difficult to find and expensive to hire. One only has to scour the testing job adverts, to see hundreds of vacancies with various testing skills in a particular tool set. In fact, there is a whole eco-system of training courses on how to use tools that require coding. In addition, there is the issue of UAT, (User Acceptance Testing), where other members of the organisation get involved to test the application. Traditional tools have failed to deliver where UAT is concerned. We are told time and time again that end users of an application cannot get to grips with the tools, so they resort to manual methods and certainly cannot afford the time out from the business to go on a crash course in scripting!!

Creating and maintaining the automated testing scripts is cumbersome and complex. It could take a team of people months and even years to set up and maintain properly. Not particularly useful if the application is time critical or if there are multiple releases per year.

Worse still - every time the application under test changes, the scripts that were written to originally test the application have to be re-written to incorporate the changes. This is a time-consuming task and can often take longer than manually testing the application in the first place. The maintenance burden is just too high to make it worthwhile.

So after much investment and months of implementation, test automation only automates a mere portion of the total testing - the area which is a stable and relatively risk-free part of the application. The rest has to be tested manually using screen prints and emails to recreate the defects uncovered. At the same time, testing at the database level isn't even considered as an option due to its complexity.

Does any of this sound familiar? If it does, we urge you to continue reading in order to gain a better understanding of the road that lies ahead of you. Some software testers who are reading this paper may even admit to being just like the Grinch, especially having shared the same test automation frustration.

However in this story, and sticking to our Grinch theme, the Grinch software tester is evil and plans to cause anarchy within the QA department. Our Grinch plans to ruin the winter



software release, which means an end to all those updates and changes to scripts - no more coding, coding, coding!

So how does the Grinch set about ruining the winter release? Well, let's cut for an intermission and then pick up the story in Part II.

Intermission...

Wouldn't it be great if in the five minutes it takes you to drink your cup of coffee, or have an ice cream, during our intermission, you could have built a complete regression test from scratch? Imagine being able to build a regression test, change the application being tested and then re-run the test - all in 5 minutes and without touching a single piece of code! If only the Grinch had **watched this test automation video!**

The Grinch character might never have been created had not Dr. Seuss** been inspired by the concept of learning from one's own mistakes. The famous author had been quoted as saying: "There's an inherent moral in any story."

Had our Grinch learnt earlier that test automation need not be the burden that it has been in the past; he may never have become a prisoner of his own trade and a sad, lonely creature. In fact, by freeing test automation from the burden of code-based scripting, automation could actually be used by subject matter experts and not limited to frustrated developers and testers. If testers were to discover a solution that could monitor and adapt to the changes in the application, an intelligent solution that inherently understood the application under test, removing the need to develop logic in addition to the validation itself, then the testing community would be singing and dancing with joy!

Part II...

I must stop this whole thing! Why, for fifty-three years I've put up with it now.

I must stop Testville testing... but how?

Wearing his evil Grinch grin a plan started to form and our Grinch found a simple way to ruin the winter software release. His idea was to cause mayhem by letting the QA team down,

**Author Dr. Seuss, of the Children's story, "How the Grinch Stole Christmas".



“Test automation that is reliant on a tester to code it, means that only that tester can respond to any issues within the code.”



forcing the project to overrun and compromising on software quality. The Grinch’s thoughts included the following:

- If there are defects in the live system, it could bring businesses to a halt.
- When a live application fails, a customer may be unable to process business transactions.
- The failing application will need to be corrected and re-tested.
- If the failure has caused data corruption then the extent of this must be analysed.
- In the event of repeated failures customers will lose confidence.
- If the deadline is missed, it will ruin any competitive advantage the business has and could also destroy customer relationships relying on the winter upgrade.

The Grinch, who was a specialist script writer, knew that mayhem would result from his sabotage. All he would need to do, was to destroy all the script work that he had spent years creating. All it would take, is one late night after working hours to create errors within the scripts and in some areas to delete entire chunks of code! Test automation that is reliant on a tester to code it, means that only that tester can respond to any issues within the code. So with a plan to ruin the code and the Grinch not around to fix it, any test automation was futile – a cunning plan thought the Grinch.

With that evil thought, the deed was done quickly for the Grinch had stolen his own test scripting!

And then at home he would wait for a call, to help those sorry testers in tears through it all.

He would save their testing project and beg him to come back. But a week had past and still no call, so he searched online and created a post on a wall.

“No news oh darn it all!”

The application deadline was too tight, so Testville would have needed to double its team of Testers or only complete 75% of the required testing. Quality would most definitely be compromised. There was no budget for more resources, so Testville would have gambled on quality! Customers affected by the bug ridden software would have been in the news with their reputations shattered - wouldn't they?

So why had our Grinch heard no news at all? Was more resource thrown at the project at the very last hour - unlikely! The Grinch, who sat in a silo, was unaware that a test automation solution had been found! One that featured:

No scripting language

Test automation was not confined to programmers and testers writing coded scripts. Instead, graphical scripts were now being built based on the user's interaction with the system under test.

Speedy implementation

Gone were the many months of building and preparation before a single item was tested - testing was now done in days, not months and years.

Scripts that update themselves when applications change

When an application changed, the scripts can automatically adapt to the change. This meant that the solution was completely re-usable, no matter how often the application changed. As a result, the automation zone was no longer confined to those areas of the application that were stable and risk free.

Assistance for manual testing

All manual testing was now streamlined. Testers were now given the much needed help to make manual testing more productive and effective.

Database testing

With data underpinning the entire testing process, poor data resulted in poor testing. Now the testers in Testville had a way to uniquely create cut down, representative test data and reduce test times. It also supported the protection of the test data, through rollback, scrambling and masking. Database updates could be tracked and optionally verified too!

*He puzzled and puzzled till his puzzler was sore.
Then the Grinch thought of something he hadn't before.
Maybe test automation, he thought... doesn't have to be so
painful.
Maybe automation perhaps means a little bit more!*

Finally, a test automation solution that is easy to use, easy to maintain and has no complex scripting language!



“Leave the coding to development and allow testers to do what they do best.... TEST!”

The End....

As the year draws to a close and winter deadlines are looming, we hope this story demonstrates that there is an alternative to code-based test automation. Unfortunately current market-dominating tools do not meet these fundamental requirements and only serve to reinforce the approach of testing in a silo with a team of coders not testers. Leave the coding to development and allow testers to do what they do best.... TEST!

The future of software test automation should not depend on a scripting language. It should still address application complexity and the dynamic nature of today's applications. At the same time, it should empower the software delivery team as a whole.

And what happened next in our Grinch story?

Well, in Testville they say - that the Grinch's small heart grew three sizes that day.

The true meaning of testing came shining through, hooray!



About Original Software

With a world class record of innovation, Original Software offers a solution focused completely on the goal of effective application delivery through quality management. By embracing the full spectrum of application quality management across a wide range of applications and environments, the company partners with customers and helps make quality and efficiency a business imperative. Solutions include a quality management platform, manual testing, full test automation and test data management, all delivered with the control of business risk, cost, time and resources in mind.

More than 400 organizations operating in over 30 countries use Original Software solutions. Current users range from major multi-nationals to small software development shops, encompassing a wide range of industries, sectors and sizes. We are proud of our partnerships with the likes of Allianz, Bimbo Bakeries, Costco, CertainTeed, Delta Dental of WI, Euronet. IAT Insurance, O'Reilly Autoparts, Cayman National Bank, Topcon, and DSC Logistics.



Original Software