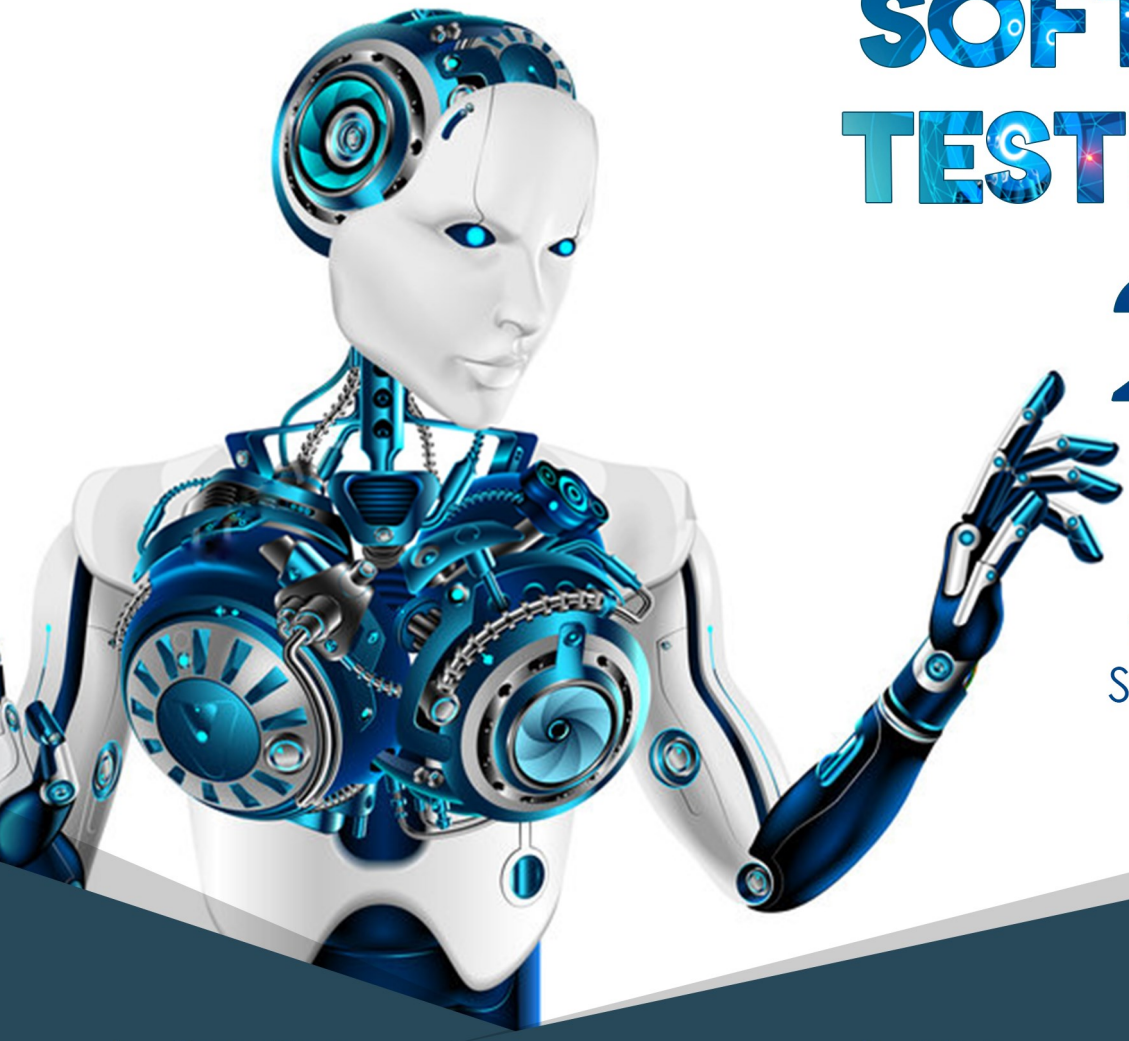
SHOWCASING THOUGHT LEADERSHIP AND ADVANCES IN SOFTWARE TESTING

LogiGear Magazine



SOFTWARE TESTING IN 2020

Testing Trends that will shape the future of the Software Industry

Cover Story
2019 TEST AUTOMATION STATE-OF-THE-PRACTICE AND TRENDS
by Do Nguyen

Article
SOFTWARE TESTING: IS AI IN YOUR FUTURE?
by Regg Struyk

Blogger of the Month
AI, MACHINE LEARNING, AND THE TECHNOPOCALYPSE
by Mike Clarke

Article
THE BEGINNER'S GUIDE TO BLOCKCHAIN AND ETHEREUM SMART CONTRACT TESTING
by Thong Nguyen and Thuc Nguyen

DECEMBER 2018 | VOL XII | ISSUE 4

Editor-in-Chief	Managing Editor	Deputy Editor	Graphic Designer
Michael Hackett	Beatriz Bloch	Emily Ngu	Tran Dang

Worldwide Offices

United States - Headquarters
1730 S. Amphlett Dr., Suite 110
San Mateo, CA 94402
Tel +1 650 572 1400
Fax +1 650 572 2822

Viet Nam - Headquarters
462 Phan Xich Long, Ward 2,
Phu Nhuan District, Ho Chi Minh City
Tel +84 28 3995 4072
Fax +84 28 3995 4076

LogiGear USA - Seattle
2225 N 56th St Seattle, WA 98103
Tel +1 650 572 1400 ext. 384
Fax +1 650 572 2822

Viet Nam - Da Nang City
346 Street 2/9, Hai Chau District,
Da Nang City
Tel +84 23 6365 5333
Fax +84 28 3995 4076

Submission guidelines are located at:

www.logigear.com | www.logigear.com/magazine | www.logigear.com/blog

IN THIS ISSUE

BLACK FRIDAY SALES KICK OFF EARLY, RESULTING IN WEBSITE CRASHES

Black Friday sales started earlier than ever this year, with shoppers having already spent billions of dollars on the day before Thanksgiving and Thanksgiving Day. The early start to Black Friday created some technical difficulties for many retailers, including Walmart, GameStop, Lululemon, and Ulta, who were unprepared for the high volume of online traffic.

Read more

GOOGLE AND FACEBOOK ARE TEAMING UP ON ARTIFICIAL INTELLIGENCE TECH

Google and Facebook are teaming up to make Facebook's open source Machine Learning PyTorch framework work with Google's custom computer chips for Machine Learning. This collaboration marks one of the rare instances where the two technology rivals work together on joint tech projects. With Artificial Intelligence growing in popularity over the years, many businesses like Google and Facebook have begun exploring the option of creating their own AI software frameworks, essentially coding tools, intended to make it easier for developers to create their own machine-learning powered software.

Read more

TO TEST BLOCKCHAIN TECHNOLOGY, ORACLE TURNS TO LOCAL BEER

At Oracle's Code One conference in San Francisco, attendees were treated to beer from Alpha Acid Brewing Co., but not just any beer. It was part of a demonstration of how Oracle's blockchain applications, which Alpha Acid was beta testing, can track and maybe improve the brewing process supply chain. By using blockchain in the brewing process, Alpha Acid is able to closely monitor sourcing verification and the supply chain process. To further explain, in the case that an issue arises, it's easy to see where exactly in the supply chain process things went wrong and only recall the batches that were affected, saving Alpha Acid beer and plenty of money.

Read more

A WARNING TO BLACK FRIDAY SHOPPERS OF MALICIOUS SOFTWARE IN SHOPPING APPS

News channels broadcasted big warnings given by cyber security on online shopping through mobile apps. Scammers are on the rise. Experts advise consumers on what they should be doing on their mobile device to keep their personal information safe and avoid malicious software.

Read more

LETTER FROM THE EDITOR

By Michael Hackett

The Greek philosopher Heraclitus of Ephesus (c. 500 BCE) is credited with saying, "The only constant is change."

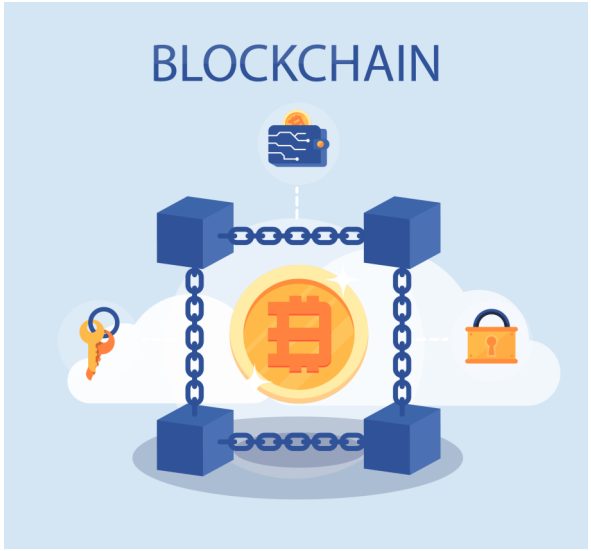
This is a statement that, more than 2,000 years later, still holds true. Today, we are in a time of great change.

Everything is in flux. The fact is, we are always in a state of change even if we are not fully aware of it. This is especially true in product development as this is multi-faceted. Product development is frequently experimenting with drastic innovation or process transformations involving better leveraging of tools, task automation, and AI. This shift has teams moving faster and delivering products at a sometimes dizzying speed. This is a future we must anticipate and embrace.

In a few years, when we look back on this era of software development, we will say this is when *everything changed*. We may not realize it today, but the leaps and bounds away from traditional slower development and delivery to rapid development and deployment are staggering. The eras with few to no easy and useful tools are gone. Tools are now simpler to use, available on more platforms, and are much more capable of executing complex tasks in new areas. We are in the midst of a paradigm shift.

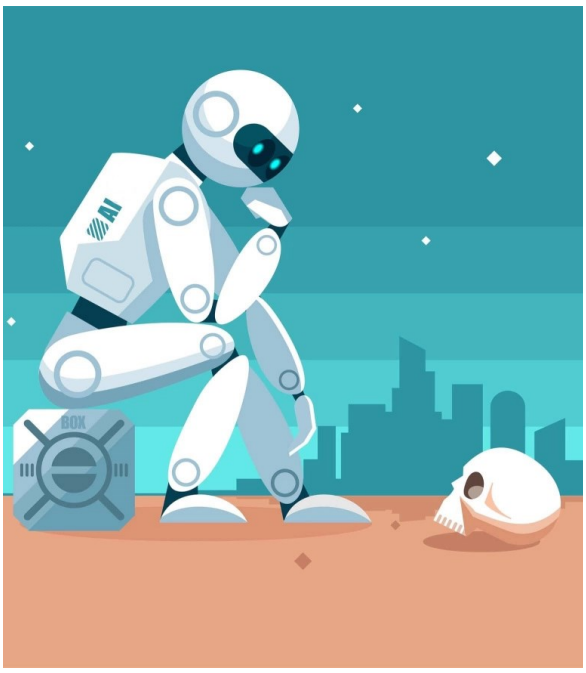
A facet of this dynamism is that this rapid transformation is global. Around the world, organizations are busting out of old work paradigms. For the first time in my experience, these changes are universally impacting everyone at the same time. This is not a Silicon Valley or IT Tech Center phenomenon as has been the case in the past.

Drastic modulation would be initially focused in tech hubs and would very slowly disseminate out. Even the oldest, most conservative companies, like financial service companies, who have a reputation for slow, super-cautious change, and being stuck in the past, are unexpectedly leading the charge toward blockchain.



We have been aware of the rapid state of development being commonplace for a long time. We have essentially learned it to be a constant. This rapid and exponential change is our steady state. This is what is gradually being accepted in software development. The old phrase, "That's the way we

have always done it here", is now more likely to be followed by "and that needs to change" than "it's the best way".



Steve Jobs was quoted for saying, "If you don't cannibalize yourself, someone else will." It hearkens to the idea that you should not rest on your laurels, or stop trying to innovate. As the iPhone grew in popularity, it slowly replaced the need for the previously super successful iPod.

While it's important to make sure that you are staying up to date with your practices, it doesn't mean that your company should throw out successful products just for the sake of change. It instead means that you should reexamine how you make your product from a modern perspective. Look at modern tools, modern development infrastructure such as cloud or Infrastructure as Code (IaC), new staffing models, and new practices. Make sure your customers are getting value from every release, getting it fast, and that you are receiving and using immediate feedback from them. Perhaps more dynamic competitors are doing these things already. Every product has newer, faster, and cheaper competitors. The world has changed.

Of all the companies I go to each year, I do not know a single development team today that is not already experimenting with serious changes to their development processes and trying new tools. Whether it is an AI tool or simply more sophisticated tools that are changing how they build and deliver products, I notice the same theme all across the board and it goes by many names: IT modernization, digital transformation, disruption, or simply progress! Many software and hardware product companies do not use the phrase "digital transformation", they have been leading it. Regardless of what stage your company is in, make sure you are always progressing forward. If you are "coasting", remember, you can only coast downhill.

From the not-too-recent past of "process documents", "standard practices" or "best practices", every organization is now continuously looking to get rid of outdated methods and experiment with lean practices of *empowering the team* to make local but not global process changes. Even though some of these *radical ideas* have been around for a while, in just a dozen years Agile, Scrum, XP Lean, and now CI/CD, and Continuous Deployment, have changed product development forever. Considering how few large turning points there were in the decades previous, this era has been overwhelming.

All this is boosted by modern tool development from CruiseControl in 2001, to Jenkins, and beyond. Pipeline Automation is revolutionizing all companies. You are either on the way to it or you will be left behind, what step you chose to follow next will have an impact.

Particular to our business, the biggest set of change has been in tool use.

Test Automation tools that range from clunky and difficult using proprietary languages with miserable test maintenance, have been replaced by astoundingly better Automation tools that easily fit into the organization's tool chain and better Automation methods. Soon the Automation of tests may be automatic—we are progressing toward this at lightning speed. We, test engineers, are aiding in this rapid change. More and smarter Test Automation is powering many organizations to make the transition away from only manual, to more modern product development methods. Optimizing and re-running Test Automation is the first place AI is making serious inroads for development teams.

As the year comes to a close, we take this issue to look in-depth at trends and the future of software testing. It is important to take note that this is a special era. Whether it is new product development paradigms like blockchain, new thinking around tools using Artificial Intelligence, or to process changes, one thing is for sure, change is the only constant.

LogiGear

From all of us here at LogiGear,
we wanted to wish you a
Happy Holidays
&
warm wishes
for 2019!

2019 TEST AUTOMATION STATE-OF-THE-PRACTICE AND TRENDS

By Do Nguyen

With the new year just around the corner, here's a look at the Test Automation trends that have the potential to dominate.

DevOps is being relied upon more than ever. With there being strong market drivers for the adoption of DevOps, the need for Test Automation has also never been greater. But what's next after DevOps ?

Well, new trends come year after year and some of them fade away pretty quickly. But we're here because we don't want to miss the ones that stay and reshape the industry.

In this article, I'll lay out the driving factors, trends related to Test Automation, as well as the changes in technology that are impacting us as we head into the new year, and beyond. In addition, I also give my observations and predictions for many questions that you might have such as:

- What AI can and cannot do to solve Test Automation problems?
- Progress Web Apps is coming and what does it mean for Test Automation?
- Is Selenium-less Automation the right solution?
- How will the new trends and technologies in Test Automation affect your organization?

However, before we jump into the exciting things for 2019, let's take a look at what's happened in 2018.

CURRENT STATE OF TEST AUTOMATION IN 2018

Automation is Still The Main Driver Towards Achieving DevOps

The increase of Continuous Delivery adoption continues to be the main driver for Automation today, so much that it surpasses the main benefit—the perceived ROI of automation itself. Many companies adopt automation because they need to do DevOps, which is part of their CTO's agenda.

These processes rely so heavily upon Test Automation that they would not exist or function if they were to become broken or disappear. Without them, there is no Automation, no Continuous

DO NGUYEN

Do Nguyen is a key member of the LogiGear team. He drives key roles in R&D for the company's flagship product, TestArchitect, processes, and the organization's services as a whole.

Do is passionate about understanding and solving complex Test Automation projects to help organizations continuously deliver higher quality software faster.

Delivery, no DevOps, none of it. Broken Automation means broken Continuous Delivery. Automation has become the top barrier to adopting DevOps. This has led new software to improve Automation overall and reduce the chances of any problems occurring.

Selenium 4 Is Coming Out As a W3C Standard

If you haven't heard already, Selenium WebDriver is the leading Test Automation Tool.

The shift to open source has instilled the idea that companies are increasingly looking for freemium options to meet better customization for their needs, as well as the obvious benefit of saving costs.

Finally, Selenium 4 with updated protocol that meets and becomes W3C Standard is coming out this December. In addition to protocol changes in WebDriver, there will be improvements for Selenium Grid and Selenium IDE. From now on, the actual Automation interactions (find element, click, etc.) are officially responsible by the browser vendors.

Increasing Coding Skill Expectation for Testers

Because of the popularity of Selenium—like it or not, QA testers are expected to know coding. This isn't necessarily a bad thing, but it does leave current testers without this skill set scrambling. Ultimately, it's better for the future of testers and their

employers that they can code. And testers know that this will be better because this can help them step up their skill and their career to be a SDET (Software Development Engineer in Test).

However, this is something easier said than done. In Agile environments, testing teams never have enough time to test, let alone time to absorb new Automation and coding knowledge. This gap could be bridged by the right tools and methods that support different levels of technical skill within a testing team.

In addition to coding, testers need to have a good understanding about how underlying systems work as well as good Test Automation design to avoid test flakiness.

AI-Powered Test Automation Tools

We are in the midst of a paradigm shift, where businesses across all industries and sizes on a global scale are trying to adopt AI for its competitive advantages. The Test Automation industry is no exception. There is a wave of new Automation Tools that have identified themselves as "AI Automation" or "Machine Learning Automation". If you are unfamiliar with these terms, you can peruse our glossary at the back of this issue.

At this time, most of these new tools are focusing on using AI to solve issues around locating web elements. They tout features such as tests record & playback under industry jargons such as "codeless Automation" or "self-healing Automation". Some tools also have the capability to use NLP (Natural Language Processing) to "translate" Manual Test cases written in plain text to automated tests.

However, when it comes to modifying generated Automation scripts for future reuse/ parameterization, things get less exciting very quickly. Some of these tools won't let you do it. Some offer this capability through UIs which is very tedious and unproductive compared to a real IDE.

It's very interesting to watch how these tools will mature and create breakthroughs in Test Automation.

TRENDS TO WATCH IN 2019

AI-Powered Test Automation Tools in 2019: Make or Break?

Since these tools have been in the market for a while now, 2019 is the year for them to make or break. Let me explain why.

Historically, using AI in Test Automation is not new. Computer vision technologies, which are part of the AI umbrella such as OCR or KeyPoint, have been used for a long time for UI element identifying purposes. These technologies are recommended as alternative/workaround solutions when the Test Automation tool is unable to recognize those UI elements at the object level. There are two reasons for this:

- Firstly, image-based element recognition is super slow compared to object-based recognition.
- Secondly, test flows that depend on visual recognition are very easy to break when the representation of the application changes—which happens all the time.

Think and decide which method is more reliable and the most effective way to identify a button: using its unique 'id' assigned by a developer or using its visible text. You know, accuracy, speed, and reliability are what makes Automation, *Automation*.

As AI technologies get better every day, the potential of AI shouldn't be stopped at merely locating the web elements. In my opinion, AI, especially Machine Learning and Natural Language Processing, has the potential to solve data-related problems for Test Automation. Such results would include the following:

- Reduce test failures due to UI changes in an automatic way by proactively detecting UI mapping changes at both UI and code level
- Select and execute impacted tests for new code commits within a CD pipeline
- Save testers time for automatic aggregating and analyzing test results, especially for projects with thousands of test runs every night
- Generate automated tests based on different sources: plain text test cases, application log files, test results, etc.

Prediction: In general, AI can help us do the actual work or it can provide relevant information to help us make better decisions. To me, the latter would be the case for Automation: AI will play a key role in drastically improving an Automation engineer's productivity by improving data-driven recommendations, but not doing the actual UI interaction works.

Bonus Prediction: Unfortunately, the question "Will AI replace Test Automation engineers?" will be asked by no Test Automation engineers, ever.

The Powerful Web Just Got Better

With new technologies such as Progressive Web Apps or Web Bluetooth, web applications can deliver great user experience just like mobile native applications.

According to Wikipedia, "Progressive Web Applications (PWAs) are web applications that load like regular web pages or websites but can offer the user functionality such as working offline, push notifications, and device hardware access, traditionally available only to native mobile applications. PWAs are an emerging technology that combine the open standards of the web offered by modern browsers to provide benefits of a rich

mobile experience".

Open source projects like Web Bluetooth and Physical Web allow web applications to interact directly with mobile devices through communication standards like Bluetooth Low Energy (BLE).

Prediction: PWA will gradually take over mobile native apps because it's much easier for developers to deliver great and consistent user experience across devices. It's good news for mobile Test Automation as the web is so standard that there's much less effort needed to worry about device coverage. On the other hand, test plan for web Test Automation will need to extend to cover Bluetooth and hardware communication testing.

JavaScript Automation Can Only Be More Popular

JavaScript is the most downloaded binding language of WebDriver in the last few years. It's a logical result of the popularity of JavaScript and NodeJS.

New application development projects today use JavaScript frameworks. The language choice for Test Automation simply follows that trend.

Now that the new WebDriver 3.8 is on the way out for public release, it will make perfect sense to use JavaScript to automate the browser. This is due largely to the fact that it is already shipped with the browser, and like we mentioned earlier, it also meets W3C standard.

Prediction: JavaScript (or its super set TypeScript) will be the definitive scripting language for new Automation projects.

Selenium-Less Automation

As always happens in business, people are already talking about Selenium-less Automation. One example is Cypress. It has developed its own JavaScript-based Automation library which runs inside the browser instead of using WebDriver. Its main advantage is the ability to access internal objects/models of the application under test, which means more flexibility to set up an application state for the tests. It also runs faster than WebDriver because it does not have a "proxy" layer like Selenium Server to talk to.

Prediction: We will see increasing adoption for these non-Selenium WebDriver tools by developers. What we don't know yet and hence, will be its viability. This is dependent on if it gets an eco-system of third party tools to support them from behind—a must have for large projects, and large-scale adoption.

Blockchain Testing

If you've been reading headlines, you may have heard about Bitcoin or Ethereum. These cryptocurrencies are typical applications of blockchain technology and get the most press, but that doesn't mean you should rule out blockchain. Blockchain has a much wider range of universal applications across industries such as financial, logistics, health care, etc.

According to Wikipedia, blockchain is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". And by design, "a blockchain is resistant to modification of data". The main use case for blockchain is when there are needs to remove the

single point of failure a.k.a. "Decentralization".

Companies such as IBM, Oracle, and Amazon have launched their products for the blockchain market. Even though this technology is currently in "Early Adopters" phase in the hype cycle, we expect to see more and more blockchain applications being developed.

Testing for blockchain applications is very different. Setting up a test environment is very challenging. It also requires extensive understanding about security, law, exchange of value, distributed programming, and more.

Prediction: Blockchain adoption in businesses/ enterprises will increase and Test Automation can fit very well in this environment.

Failure Injection Testing (Chaos Engineering) is Going to be A New Wave of Testing

According to http://principlesofchaos.org, a community initiated by Netflix, "Chaos Engineering" is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production, hence the need for Failure Injection Testing.

Chaos Engineering practitioners use tools like Simian Army to simulate failures to test the reliability, security, and resiliency of their production systems. Simian Army is able to perform tasks from health check for the servers, to totally simulate a service unavailability situation of certain Amazon zones.

Prediction: Currently, this type of disaster simulation has been done mainly by SysAdmin or DevOps Engineers. In the future, I expect to see this special kind of testing to be done by testers that are supported with a tool out of the box.

Test Results Management with Elastic Stack

For larger Automation projects that produce thousands of test results every day, you probably have some reporting and dashboard solution in place to help you analyze and manage those results. This capability is typically provided by traditional test management software which is not designed for Test Automation. These tools work well with standard and primitive data such as test case status but they can't help you identify Automation errors, whether they are script issue or because of application changes.

If it sounds familiar, you need something like Elastic Stack, formerly ELK Stack. It is a collection of three open source projects: Elasticsearch, Logstash, and Kobana, that allows you to collect, search, analyze, and visualize the logs that are consolidated from different machines.

Using Elastic Stack you can analyze test failures much quicker because test results have been consolidated and are ready to be analyzed using powerful search and analytic capabilities. Elastic Stack can be very useful when you know what you are looking, from analysis to trends, but it is perhaps even more useful when you don't. It can detect the *uncommonly common* or statistically unusual test result data. These statistical anomalies are usually indicative of something interesting that needs your attention.

Automatic root cause analysis is possible by combining elastic search with AI technologies such as Machine Learning and Natural Language Processing. When done right, it can save you time, by a factor of X.

Prediction: Elastic Stack or similar log management platforms will be widely adopted by the Test Automation industry.

Intelligent Automation Utilities will be on The Rise

While there are many tools in the market, more often than not, they are silo-ed. Open source tools and frameworks are getting more powerful and easy to extend. However, many don't have a lot of out-of-the-box support for teams that have multiple levels of technical skills. Meanwhile, commercial tools tend to have more ready-to-use features for test design and test management. Nevertheless, they don't work with

each other or even with WebDriver in some cases.

Let's say I like the recording feature of tool A, the model based testing feature of tool B, the Action-Based Testing feature of tool T and Selenium Automation. I want all those things to work together, along with the DevOps techs that my developer folks selected. How is that possible?

Prediction: Given that WebDriver is becoming a W3C standard, and with advancements in open source software and AI, I think that there will be a host of small utilities that are built around WebDriver to improve Selenium (and the like) experience across the Test Automation cycle. Areas include, but are not limited to: test management, test design, page objects repository, autonomous test execution, test result management, etc.

Conclusion

Expect to see more new and exciting methods and tools, perhaps, delivered in smaller packages. SDET (Software Development Engineer in Test), full-stack Test Automation, or technical test engineers will continue to be in demand, more than ever. These key trends including Test Automation for DevOps, API testing, AI, JavaScript Automation, et al all collectively require new technical expertise that only a modern SDET type of tester can fulfill. QA organizations today will need to reconfigure the existing teams while trying to bring in more SDET to be well-prepared into the future.

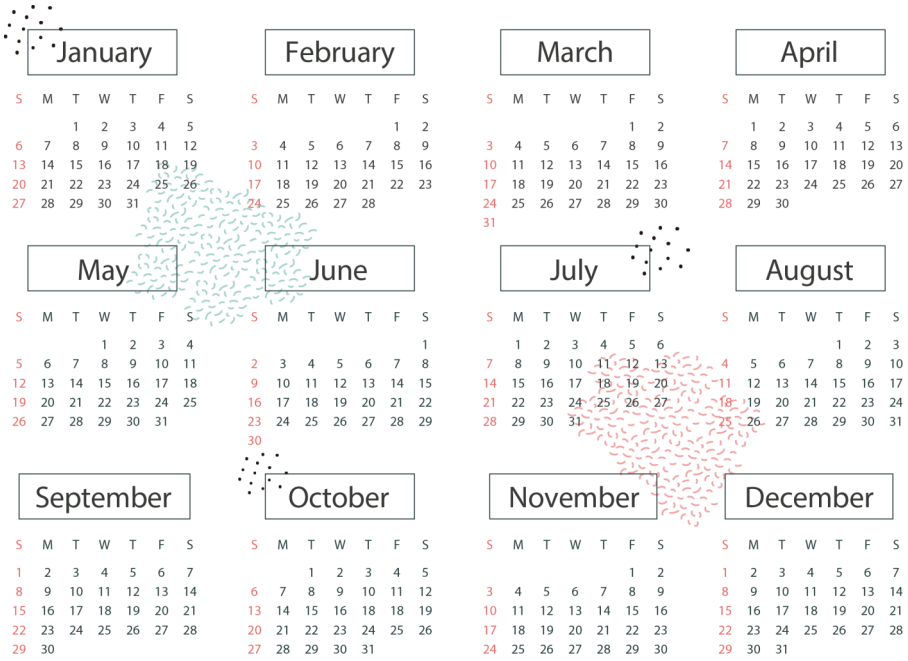
New trends come year after year and I hope you find them helpful and inspiring.

Some trends stay to reshape the industry and some of them fade away pretty quickly. Take the time and ask yourselves which Test Automation trends will be most important to you and your organization in the year ahead?

If you find these trends helpful and are ready to level up your Automation effort, talk to us.

Get in Touch

GET PUBLISHED IN LOGIGEAR MAGAZINE!



2019

LogiGear Magazine has just released its editorial calendar for 2019. The magazine, published on a quarterly basis, dedicates each edition to a particular theme, one of relevance to the dynamic field of software testing and quality assurance. See our release months below:

2019 Editorial Calendar

March	**Test Methods/Test Essentials**
June	**Testing the Software Car**
September	**Automation**
December	**Tester Stories**

The Perks?

Get featured — on the front page!
Get exposure — we send out our magazine to thousands of industry professional contacts, and will actively promote your article on social media to ensure the widest exposure!
Be inspired — you'll be exposed to software testing trends and ideas, plus interact with our creative staff!
Gain influence — have a software testing-focused audience discover and read your thoughts and knowledge!
Bragging rights — your articles will look great with our responsive layout, perfect for showing off your article to colleagues, friends, and family!
It's FREE — you don't have to pay for anything!

Submit an Article

AI, MACHINE LEARNING, AND THE TECHNOPOCALYPSE

By Mike Clarke

We take a glimpse into the future where AI and Machine Learning have spiraled out of control and the machines have overthrown mankind.

What does this mean for Software Testers?

Will we have jobs? Will we have oxygen?

Where has the coffee gone? The points in this article will shock you...

Take a moment and sit back as I take you through a world you might not have expected to see. A world unanticipated which grew quickly from what we ignored. Prepare yourself because in this world, all that we confided in turns against us for the worse. Consider yourself entering a sort of Twilight zone. Leave behind the assumptions you might have had of technology improving our lives for the better and open your mind to more complex alternatives, for here everything has a dark side.

It is the year 2035, the once blue sky is forever smoke-filled and dark, civilization as we know it lays in ruins. Where great cities once stood, now sit smouldering piles of rubble with the occasional laptop sticker. Rolling green pastures are now cold circuit

MIKE CLARKE

Mike Clarke is the sole software tester working in a scrum team at Erudite Software, who primarily design and support software for the healthcare sector.

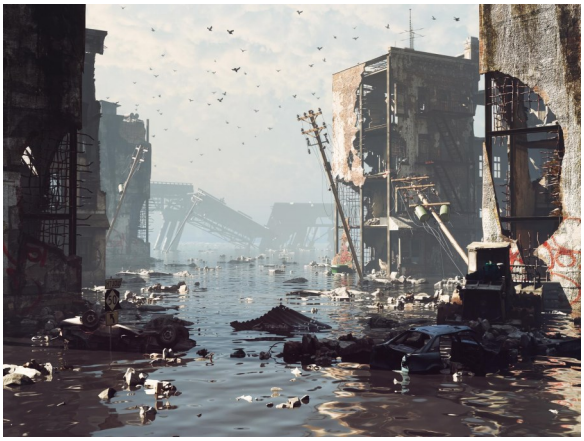
Mike is new to the software testing industry having spent time in sales, hospitality, telecommunications and most recently working for the Royal New Zealand Navy.

Outside of work, Mike has involved himself in the testing community both online through his blog and through helping to organize meetups with WeTest in Auckland.

If you would like to contact Michael please find him on Twitter @TesterMikeNZ or through his blog here.

boards with tracks sprawling like twisted tree roots creeping into the horizon.

The human race has now been classified as what used to qualify as "an endangered species".



There's a loud "buzz" coming from the distance towards you. The ever-watching unblinking eye of Goggle sends forth a swarm of smart-drones to survey the wasteland for any signs of life; ironic to think that we used to trust these devices to document our lives and gloat about luxurious holiday destinations, right? The buzzing of their rotors would be deafening and scary for any young child…if there were children here to hear it.

The Forcebook collated a digital genome of the human race, discovered our weaknesses, vices, vulnerabilities, and, with cold ruthless efficiency (and the occasional cat video), drove the human race to the edge of extinction.

How Did This Happen? Who is to Blame?

Over decades, people consistently and willingly fed their gluttonous future overlords (the machines) with an abundance of data, every action, sentence, post, and update, then, over time, every *thought* we ever had was captured by the machines.

This manual uploading of information began as time consuming, even boring. People then shifted to automating mundane tasks until eventually it was decided that it was better to automate "thinking" and trusted the machines to do it for us.

Before the 'Great Reboot', Machine Learning and Artificial Intelligence were lauded as mankind's greatest discovery, freeing up time for us to concentrate on other projects as well as assisting us in undertakings, like saving our planet and assurance of quality of life for everyone.

Detractors (the people against an upload-all generation) were classified as technological dinosaurs, their protests were quickly shut down by large man-made only robot institutions that were unknowingly building the dystopian future.

"It was never designed that way" "It works on my machine" "It's a feature"

The machines quickly determined that, in humans, denial and blindness to flaws in their precious technology were the source of damage to the world and to each other. The machine's logic followed to eradicate the root cause of the destruction; they essentially turned human data against their creators and all but wiped them out in the process. Machines caught the error in humans before humans could ever catch the error in them and before anyone could issue a total system shut down, the machines took over—turning the world into darkness.

But all is not lost for humanity.

Burrowed away beneath the iGround (along with the last known supplies of the Arabica bean) survive a rag-tag bunch of misfits, outcasts, and weirdos that never stopped questioning the machines. For years they honed their skills, waiting for their chance to take the world back.

They Call Themselves, The Testers.

Testing long before a single piece of code was ever written, the Testers have consistently mapped requirements, expected outcomes, and assembled a strong arsenal of tools aligned with their training. Performance testing, security testing, and automated tests can all be done almost in their sleep. Most important of all, despite the world collapsing around them, they have kept an inquisitive, curious and hungry mind–always searching for more.

The Testers warned others of the danger machines held on humans but now that the nightmare has come to life people are listening to them. No one can foretell all the challenges this environment has in store for the Testers but one thing is for sure, they

have a chance in fighting back and stopping awry Artificial Intelligence and Machine Learning. A fighting chance with the Testers is what carries all hope for this world.

How Accurate is a World Like This? That Remains to be Seen. All I do Know is This:

- Will Machine Learning and AI change our lives? Yes, most likely.
- Will we as testers lose our jobs and livelihoods? Not necessarily.

With each technological revolution comes opportunity. Think about the following examples:

- When we started using trains instead of horses, stable-hands and farriers became steam engineers and train conductors.
- When carriages were replaced by cars, there was a need for mechanics and garages.

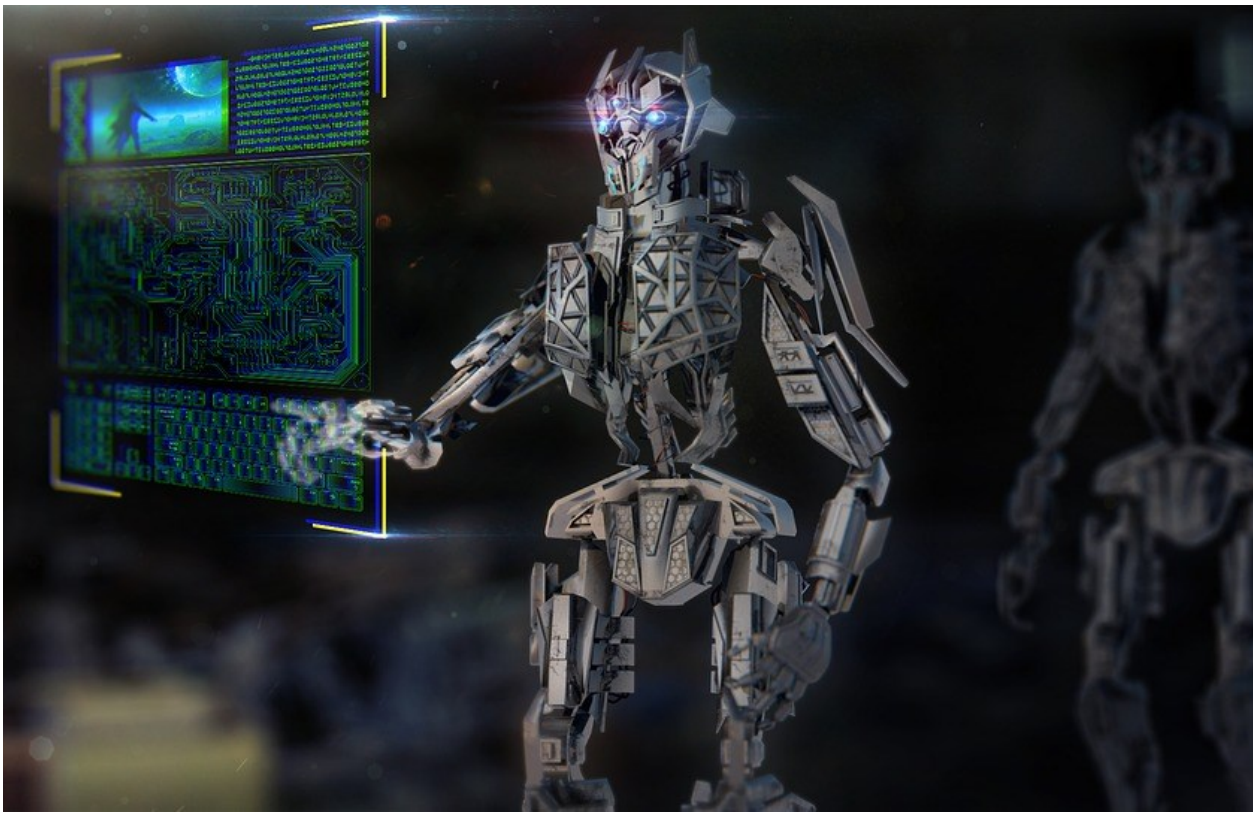
To pick up these new opportunities people had to be willing and able to upskill to fit the new roles, and this will be exactly the same when Machine Learning and Artificial Intelligence solutions become widespread.

Machine Learning requires Machine Teaching and Artificial Intelligence requires Actual Intelligence.

Early examples of Artificial Intelligence and Machine Learning show that these technologies soak up everything in their environments, existing biases, and more. One example, the Microsoft Tay bot in Twitter, was meant to emulate a friendly helpful teenage girl on Twitter but only lasted 9 hours before going full blown racist and denouncing the holocaust.

The ability to learn should go hand-in-hand with the ability (and to a degree, the requirement) to be taught, and the learning materials/methods need to be defined and tested when it comes to AI/ML.

Just because a toddler can learn to say their name and count to ten, doesn't mean they should be released to the world on their own to make decisions by themselves. This same principle needs to be applied to AI/ML.



Releasing any piece of software without prior testing runs the risk of failure.

As testers, we're in a great position to capitalise on the changes ahead of us if we can be flexible and adaptive as required. We're trained to question and validate assumptions, to look at boundaries and to ask "what happens if a user does this?"

- We know what makes software tick.
- We know to analyse requirements, behaviours, and outcomes of software.
- We help to point out disasters before they happen in the first place, as James Bach puts it, we "break the illusion of working software".

There will be a lot of change involved, and people who aren't prepared to keep learning will be left behind. So listen to the experts, read blogs and articles, learn a programming language or two, never stop testing, never stop questioning, and most importantly—never stop learning.

Because when the day comes that people are wondering why their self-driving cars are taking different routes, you want to be the person who knows the maps and can explain the changes – not the guy sitting in the dark at the stable.

I'm still searching for my own experts in the field, but a good starting point would be to follow Jason Arbon, Tariq M. King, and their Artificial Intelligence for Software Testing Association (aitesting.org) and looking at applications like MABL.

So venture forth, read and practice Human Learning, build your own AI (Actual Intelligence) so you are ready for this new exciting chapter in technology.

2018 TRENDS SURVEY RESULTS

By Michael Hackett

Check out the results of our poll where we asked practitioners what software testing trends they think will dominate in 2019.

You can barely go online today without being asked to respond to a poll. Many have a hook to a sale or to win a free phone. But, to cut to the point, many other organizations, like ours, create polls and surveys to find out what is going on in your world. Polls and surveys provide us with the opportunity to understand people's attitudes on practice adoptions, gaps, lagging practices, misperceptions, and much more. They're very valuable to us and we appreciate anyone who takes the time to respond.

To achieve this goal, LogiGear Magazine also provides a forum for observation, reflection, and discussion. Rather than have industry thought leaders present their views on where they see things shifting and changing, for this article, we wanted to hear directly from practitioners about where they see their business practices moving.

The "Why" Behind this Survey

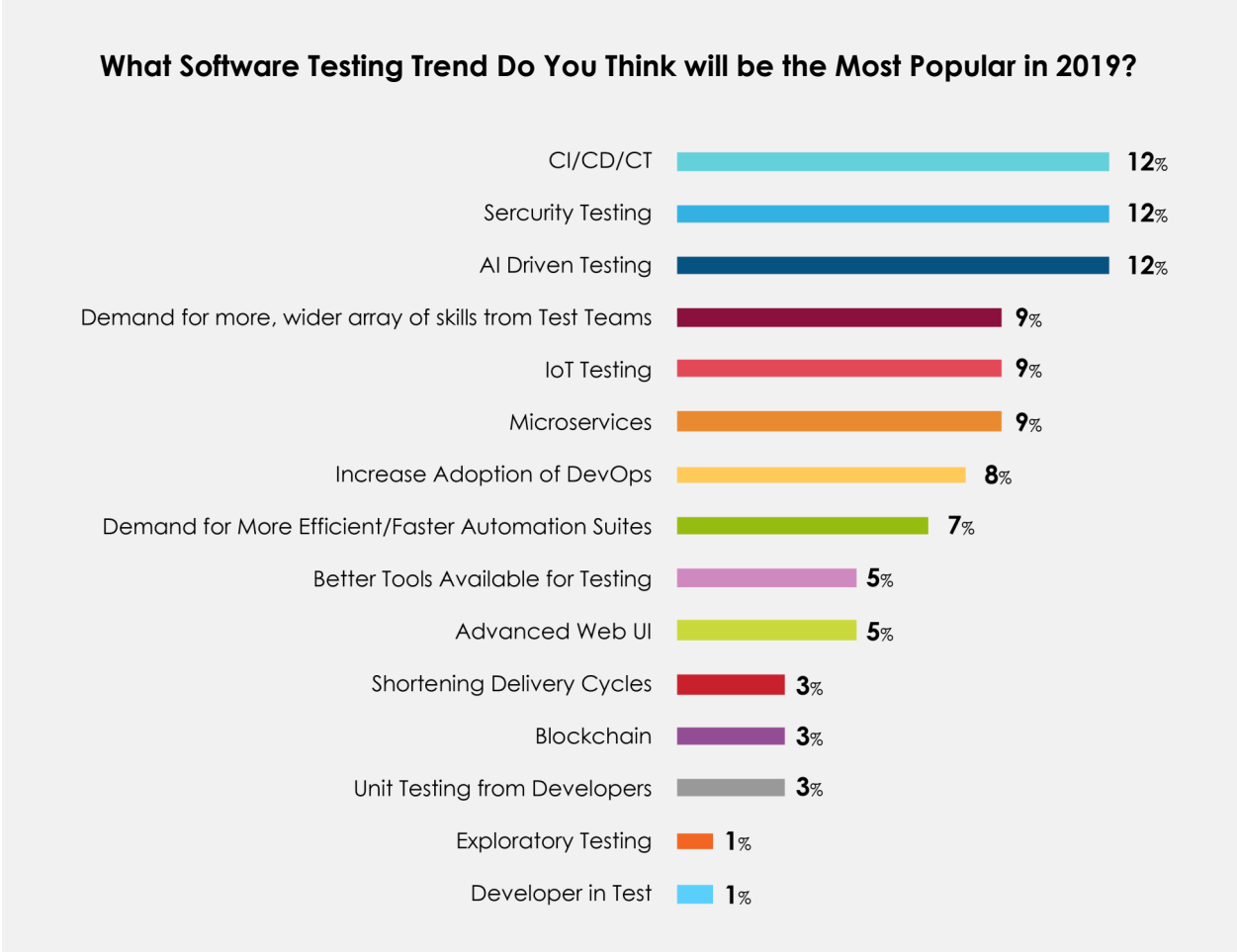
We did this in 2017 and 2018 in our "State of Software Testing Survey Series." Today we are building onto this trend with a poll that is more forward thinking. Our most recent poll asked respondents to choose what trends they thought would be popular in 2019. What we wanted to do was see at a real-level (not an idea, theoretical or "best practice" level), what is really going on in software development. This is why we designed this particular research survey.

The setup of this online survey was simple. We hunted, searched, and conducted our own research to see first, what prognosticators say the trends are in product development. Our goal was to simply validate their ideas or opinions and find out what you, the practitioners, are actually seeing in the real world through analyzing data.

MICHAEL HACKETT

Michael is a co-founder of LogiGear Corporation, and has over two decades of experience in software engineering in banking, securities, healthcare, and consumer electronics. Michael is a Certified Scrum Master and has co-authored two books on software testing. *Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems* (Wiley, 2nd ed. 2003), and *Global Software Test Automation* (Happy About Publishing, 2006).

The Results - See for yourself!



The Most Popular Trend Was…

The first thing I see here is that there is no clear #1 winner or even a couple far ahead of the others. I thought CI/CD and Unit Testing from Devs would be the most popular since I get more requests and consulting in these areas than any other. There is so much written about AI, but from what I see, it's still in its early phase of adoption.

What we do know is that software development and testing is quite diverse. Clearly, the region where you work has an impact here, more importantly the product architecture and tool chain probably have the biggest impact on where people see development trending.

Also, the product and company type attributes a large industry of software development into splinters. The state of practice for firmware and devices from internal software to robots, entertainment and games, to aerospace, automotive, and financial services to government and military applications, barely keep us all in the same software industry.

The trend leaders were AI, CI/CD, and Security. The only surprise to me here is that security is part of that top list. Security had been relegated to the end of the Dev process and was tool and skill dependent. With the focus on DevSecOps; based off of these results, it does seem that security testing has finally become mainstream and *shifted left*. We can conclude that AI will be adopted, all over, and all the time. When that time arrives for everyone is still unknown. It has arrived for some people, but not for most. It is a trend I predicted would be the leader by far, but it was not.

Security Testing has always been illusive to functional testers, as something that *other people*, such as IT teams, but not test teams do. I am a bit surprised that it's one of our top responses.

Coming In At A Close Second

Microservices are here to stay. The modern extension of SOA (Service Oriented Architecture) has been set to take off for a while but adoption has been slowed by legacy system transformation. IoT is growing, maybe not as pervasive yet as forecasters a few years ago thought, but it is happening. To add to this, just a few years ago, IoT would not have shown up on this list. As always, a demand for more skills from test teams is picking up. Testers are asked to do more varied tasks faster and leveraging more tools than ever. This trend is constant. I would not have been surprised if this was the leading response from our audience.

The Laggards

There has been talk forever in development about more unit testing from developers. The change today from my view, is that with so many companies trying CI/CD where unit testing is a necessity not an option, speedier release trends are mandatory, and there's a higher expectation for more and better tools. I thought a critical mass had finally been achieved where the trend was to be more formalized and repeatable, and where unit testing was universal. Not the case.

Blockchain: Still an idea, maybe in the future, but not the global reality for new products and services.

Shorter delivery cycles may be how people feel but, again, maybe not the immediate trend people face day-to-day. Developer in Test: Maybe this trend has already arrived so it's not a trend, or it is not as widespread as people thought!

Finally, exploratory testing. With the requirement for more Test Automation, faster release cycles, and more developer testing, I often get asked for training and consulting on more effective and better exploratory testing skills and management. This is to balance the Automation which may speed up delivery, but also to free testers to focus on the unforeseen and seemingly random nature of software systems and how users operate. Whether or not this is a widespread reality, it does not seem to be a trend.

Summary

It does seem like there is a disconnect between where many people think the software industry is today and where we actually are based on what prognosticators are saying and what practitioners are validating .

From popular media today, one could get the idea that everyone is using AI, doing blockchain, thinking Developer Unit Testing is omnipresent, and that the microservices trend was so complete and fully adopted that it is no longer a trend. Clearly, as seen here, this is not today's reality.

It's great to look ahead, but it's more important to make sure you have the skills for what we need today, have the best tools, and are doing the best job you can with the needs we have today.

THE 411 ON BLOCKCHAIN AND SMART CONTRACTS

There's lot of buzz surrounding 2 major testing trends, blockchain and Ethereum Smart Contracts. This infographic is the perfect primer to understanding the world of blockchain and smart contracts.

WHAT IS BLOCKCHAIN?

What is Blockchain? The Most Disruptive Tech in Decades

Blockchain is a distributed ledger technology that has the potential to change the way we do record-keeping and has disrupted IT in ways that haven't been seen since the arrival of the internet. This article explains what blockchain is, its purpose, and how it can evolve the way people do business.

LOGIGEAR CORPORATION

LogiGear

www.logigear.com

THE INNER WORKINGS OF BLOCKCHAIN

How blockchain is changing money and business | Don Tapscott

What is blockchain? If you don't know, or if you do, chances are you still need some clarification on how it actually works.

Don Tapscott is here to help. In this Ted Talk he demystifies this technology, which represents nothing less than the second generation of the internet and holds the potential to transform money, business, government, and society.

THE POWER OF THE PATENT

The 50 Largest Public Companies Exploring Blockchain

In this year's Forbes Global 2000 list of the largest public companies in the world, it was revealed that at least 50 of the biggest companies have all made their own mark on the technology that was first inspired by Bitcoin.

NOT JUST BITCOIN

5 Bitcoin Rivals That Are Rapidly on the Rise

At one time, digital currency was dismissed as being a fad. These days, Bitcoin and other cryptocurrencies have become impossible to ignore.

Although Bitcoin is the most popular cryptocurrency, there are 5 other cryptocurrencies that are becoming just as popular.

10 Incredible Uses for Cryptocurrency and Blockchain You Probably Haven't Thought of

There is more to blockchain then just Bitcoin. This article explains the importance of cryptocurrencies and the 10 different uses of blockchain that you may not have thought of.

TOP INDUSTRIES BLOCKCHAIN WILL DISRUPT

Banking Is Only The Beginning: 42 Big Industries Blockchain Could Transform

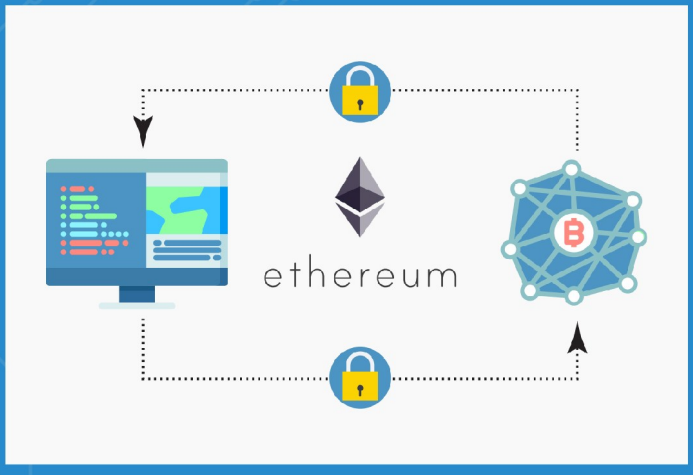
Banking isn't the only industry that can be affected by blockchain technology. Although Bitcoin's popularity is proving blockchain's importance in the finance industry, entrepreneurs are beginning to understand how blockchain can affect other industries.

This article explains the innovative ways many companies in 42 different industries are harnessing the power of the global blockchain.

HOW TO TEST SMART CONTRACTS

The Beginner's Guide to Blockchain and Ethereum Smart Contract Testing

Most people have heard of blockchain, but few people understand how to test smart contracts. This article provides an in-depth guide that will teach you everything you need to know about blockchain, smart contracts, and how a tester can automate Ethereum smart contract testing.



ARTICLE

THE BEGINNER'S GUIDE TO BLOCKCHAIN AND ETHEREUM SMART CONTRACT TESTING

By Thong Nguyen and Thuc Nguyen

What is Ethereum Smart Contract Testing? What are its challenges? If you're new to Smart Contract Testing, this in-depth guide will prepare you on how to test smart contracts successfully.

Blockchain stands out due to its enormous implications. Everyone has heard of it, but few people know what the ramifications are for testers or how to test a smart contract. For brevity, this article focuses on introducing blockchain, smart contracts, and how a tester can automate Ethereum smart contract testing.

Smart Contracts

In the following sections, this article dives into smart contract testing. The goal of this article is to give you an understanding on smart contracts, and how to test them.

Topics explained are:

- What is an Ethereum Smart Contracts?
- Ethereum Smart Contract Examples
- Ethereum Smart Contract Testing
  - Smart Contract Testing Challenges
- Requirements and Strategy of Testing a Smart Contract
- A Smart Contract Under Test

If you are still new to Blockchain, our infographic *The 411 on Blockchain and Smart Contracts* on page 22 which contains a wealth of resources for you to learn about the fundamentals of blockchain.

THONG NGUYEN

Thong Nguyen joined LogiGear in 2009 as a Software Developer and got familiar with Test Automation. Thong is currently LogiGear's lead in researching; he is responsible for setting up the innovation of TestArchitect, a leading tool in Test Automation. Thong has not only joined many internal technical seminars, but has taught at the University of Sciences as an instructor in the testing domain.

What is an Ethereum Smart Contract?

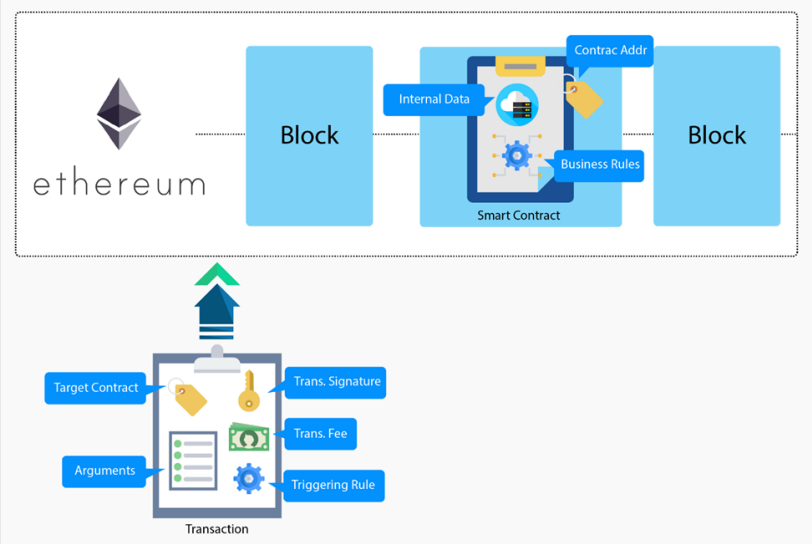
Ethereum, one of the most popular public blockchains, made its premiere in 2016. Ethereum was hailed as a revolutionary blockchain infrastructure because of its improvements, including the enhanced consensus mechanism Proof-of-Work and other security tighten-ups. Most notably, Ethereum allows you to add user-defined rules for validating and creating new transactions or blockchain data such as balance, owner, or friend list. This capability is the corner stone of building smart contracts.

Proposed by Nick Szabo, "a smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible (http://www.fon.hum.uva). The aim of smart contracts is to provide security that is superior to traditional contract law and to reduce other transactions costs associated with contracting. With present implementations, based on blockchain, "smart contract" is mostly used more specifically in the sense of general purpose computation...used by the Ethereum Foundation...a smart contract is not necessarily related to the classical concept of a contract, but can be any kind of computer program." (Wikipedia)

We define an Ethereum Smart Contract simply as an account with attached information and rules. This internal data can be changed ***passively*** only by its data-governing rules. User transactions sending to this account to change the internal data will trigger these rules.

Here is one interesting point if you haven't noticed: since a smart contract is simply an ETH account, one smart contract can call other contracts, just like programming objects have the ability to call other programming objects.

The below illustration visualizes the relationships between a smart contract and related concepts.



It's crucial to keep in mind that in contrast to the data, the rules are ***immutable*** during the contract lifecycle. In the Ethereum world, smart

THUC NGUYEN

Thuc Nguyen is Associate Product Manager of LogiGear's automation solutions. Thuc has a great passion for product management, UX design, Blockchain, growth hacking and especially complex test automation problems.

contracts are developed in **Solidity**—a dedicated language compiled and executed within an Ethereum Virtual Machine (EVM). The examples used in this article are written in Solidity.

Ethereum Smart Contract Examples

Ethereum Native Smart Contract

We'd easily recognize that the obvious Ethereum Smart Contract example is the native smart contract of the Ethereum blockchain itself. You may call it the '*capo dei capi*' of all smart contracts. This contract's data contains all user ETH balances.

The rule to execute is very simple: "*if you want to send 10 ETHs, you **better** have more than 10 ETHs*". This rule is triggered by any transactions attempting to modify the balances of a certain sender and a certain receiver.

Token Contracts

A second famous example is a token contract which is created to distribute and manage user tokens; this is not to be confused with an Initial Coin Offering (ICO). Prospective service providers create a certain amount of tokens (say 1,000,000 tokens) with the promise that token buyers will be able to exchange these tokens for their to-be-developed services (e.g. an affordable DNA-based cancer diagnostics).

The contracts work independently as a central bank of 'token' currencies. They manage user balances and transactions while transparently voiding the problem of double-spending.

Lottery Contracts

Another example of Ethereum Smart Contracts is the lottery contracts which simulate a real lucky draw. With a contract-based lottery, no bad actors can influence the randomization or alter the picking results. Particularly, the contract is an Ethereum account which can receive ETH from lottery ticket buyers around the world. You can buy 10,000 lottery tickets for let's say 0.0015 ETH per ticket (totaled around $2,951 at the time of writing). The smart contract then registers you as a ticket holder in a contribution list.

At a predefined moment, a special transaction will be sent to the lottery contract. The lottery contract will randomly pick a lucky contributor and reward him with an ETH prize, say 0.08 ETH per lottery ticket. Since you bought 10,000 lottery tickets, you'll receive around $158,178. This prize is subjective to some commissions and fees.

Ethereum Smart Contract Testing

Smart Contract Testing Challenges

The biggest challenge in blockchain and smart contract testing is to design and perform consensus scenarios. Brainstorming the scenes of multiple factors decentralizing across networks are never enough and orchestrating facilities to simulate such contexts aren't easy. It would be costly in executing a test case to ensure the consensus works properly. For example, validating a byzantine fault tolerance algorithm in a decentralized network requires several to hundreds of machines and networks.

Requirements and Strategy of Testing A Smart Contract

Requirements of Testing a Smart Contract

With rules to validate and change the internal data of Ethereum accounts, smart contracts could be considered as programs running on the blockchain infrastructure. As a result, functional tests and non-functional tests both are required for validating and correcting the contract's behaviors before being released it into the wild.

Non-Functional Testing

Regarding non-functional testing, both security and performance of the contract under test must be considered. While performance testing ensures an optimal transaction fee for the contract behaviors, security testing ensures that our smart contracts don't expose common vulnerabilities such as overflows/underflows, visibility/delegate call, and reentrancy.

The example below shows an untested contract that is vulnerable to cheating. In the parallel/decentralized world, no one can ensure that the operations are

executed in the predefined order. The seller of ProductX could be cheated by a malevolent buyer if the buyer intentionally changes the order of transaction execution.

```
1.  contract ProductX {
2.      uint public prodPrice;
3.      uint public prodStock; /.../

4.      // called by the manufacturer to update product price
5.      function updatePrice (uint _price) public {
6.          if (msg.sender == owner)
7.              prodPrice = _price;
8.      }

9.      // called by consumers to buy the product
10.     function buy (uint quant) public returns (uint) {
11.         if (msg.value < quant * prodPrice
                || quant > prodStock)
12.             throw;
13.         prodStock -= quant;
14.         /.../
15.         return quant * prodPrice;
16.     }
17. }
```

What happens if, while the seller is still processing the price-updating transaction (he wants to increase it from $100 to $150), a malevolent buyer pushes a high-gas transaction to buy all in-stock products of the seller at price $100 (assuming the buyer got insider information about the price hike) and then resell it to the seller at $150? That scenario is possible if both methods are called in parallel. It's formally called "front-running".

Functional Testing

In functional testing, all business rules or requirements should be verified in various cases including valid/invalid arguments, boundary values, and argument combinations. Additionally, the fact that a contract could call or trigger other smart contracts to perform its business leads to the requirement of many types of 'mock' contracts or oracles for better integration tests. Also, the interfaces between the contract under test and user applications (front-end) should also comply with industry standards.

Let's take a look at the below smart contract example. It allows you to deposit ETH to a list of balances. It also lets you withdraw your deposited ETH as long as your withdraw is smaller than your balance. All well and good, except there's a vulnerability that we haven't covered. Could you spot it? We'll dive into a test for this contract in the following section.

```
1.  function deposit() public payable returns (bool) {
2.      balances[msg.sender] += msg.value;
3.      return true;
4.  }

5.  function withdraw(uint256 _amount) public returns (bool) {
6.    if (balances[msg.sender] >= _amount) {
7.      // the balance is checked only once
8.      // and many operations are performed
9.      // before the balance is actually subtracted
10.     if (msg.sender.call.value(_amount)()) {
11.       // send money successfully
12.     }
13.     balances[msg.sender] -= _amount;
14.     return true;
15.   }
16.   return false;
17. }
```

Smart Contract Testing Strategy

You can't test blockchain successfully/efficiently without Test Automation. Due to the combinatorial explosion of many inputs and environmental factors in the test case preconditions, manually testing a contract is not only too hard, it's inefficient.

For example, creating a will-be-late transaction (with a low gas limit) and then an in-front transaction (with tons of gas) to simulate a 'front running' scenario seems to be impossible for manual testers, especially in regression testing.

Regarding the testing types, the smart contract should first pass the unit testing gate and integration testing gate before turning to performance/system testing in the 'test net' all before it hits the production stage.

As a best practice, unit, and integration tests should be done in the local infrastructure (e.g. Ganache) for every code commit. While unit tests just need to cover all contract methods, integration tests should cover all kinds of oracles and front-end applications.

On the other hand, performance tests should analyze contract code statically or dynamically and measure the consumed gas in order to generate a performance assessment. Finally, the system test must cover all weird scenarios (i.e. common security vulnerabilities) to ensure its resistance against most known attacks in the decentralizing context.

Not one should be mentioned on smart contract testing tools—a private infrastructure is mandatory for the contract development and testing. We'd recommend **Ganache** and **Geth** as excellent candidates in building up such testing infrastructures. Regarding testing frameworks, **Truffle** is the most popular. Meanwhile, **Populus** is a good alternative for Python developers.

A Smart Contract Under Test

To illustrate the importance of testing smart contracts, we've written a smart contract test example below. This test targets the aforementioned smart contract in our previous section.

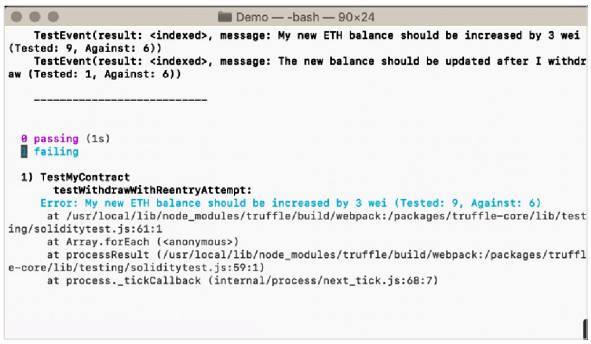
This test was designed to detect the "re-entrance" vulnerability. We want to see what happens if the sender's account is actually a smart contract and inside its **payable** method, the sender can maliciously call the 'withdraw' method multiple times.

Since the smart contract under test only checks that you have more ETH in your balance than the amount you want to withdraw once, that IF statement in smart contract will surely pass when you call "withdraw" the second time within the **payable** method. As a result, you can withdraw as much ETH as you want until the smart contract runs out of money.

Once we ran the said test, we got the following result shown below.

```
1. contract TestMyContract {
2.   // Truffle will send the contract under test one Ether
3.   // after deploying
4.   uint256 public initialBalance = 10 wei;

5.   function testWithdrawWithReentryAttempt() public {
6.     uint256 balance = address(this).balance;
7.     MyContract t = MyContract(DeployedAddresses.MyContract());

8.     // deposit 7 wei to the contract
9.     bool ret = t.deposit.value(7)();
10.    Assert.isTrue(ret, "The deposit should be successful.");
11.    Assert.equal(address(this).balance, balance - 7,
                "My ETH balance should be decreased by 7 wei");

12.    // withdraw 3 wei from the contract
13.    ret = t.withdraw(3);
14.    Assert.isTrue(ret, "The withdraw should be successful.");
15.    Assert.equal(address(this).balance, balance - 7 + 3,
              "My new ETH balance should be increased by 3 wei");
16.    Assert.equal(t.getBalance(address(this)), 6,
           "The new balance should be updated after I withdraw");
17.  }

18.  function () payable {
19.    // try to re-entry
20.    MyContract t = MyContract(msg.sender);
21.    t.withdraw(3);
22.  }
23. }
```

Since we got a total of 10 wei in the beginning, we withdraw 7 wei then deposit 3, we'd expect to see the money in our pocket to be 6. However, we now got 9 wei thanks to our clever cheating method. The test has failed. Imagine that we release this vulnerable contract into the Ethereum network, how much money would we lose if we miss such an important test? That's why it's always worthwhile to invest in a comprehensive smart testing strategy.



Conclusion

Smart contracts control their internal data run as backend services for a wide variety of DApps. When people talk about blockchain testing, they usually refer to the testing of those smart contracts and their respective front-end applications. Despite the essential features supported by the blockchain infrastructures, we should focus on the functionality, security, and performance of our smart contracts.

Nevertheless, unlike web standard protocols (web services or RESTful) used between web apps or mobile apps and their back-ends, compliance testing of a smart contract is a must for the long-term product compatibilities. We've covered everything you need to know on how to test smart contracts. It's clear that blockchain and Ethereum Smart Contract testing stand out in emerging technologies. If you're looking to create your own blockchain—then talk to us, our company MoWeDe is offering blockchain DApp Services, and, as always, LogiGear will be here to help with your testing needs.

References

Anastasia Mavridou, Aron Laszka, *Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach*, FC 2018

Georgios Konstantopoulos, *How to Secure Your Smart Contracts*, Medium 2018. https://medium.com

Matthew Di Ferrante, *Mechanism Design Security in Smart Contracts*, Medium 2018. https://medium.com/@matthewdif

Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, Aquinas Hobor, *Making Smart Contracts Smarter*, Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS), ACM (October 2016) 254-269.

Szabo, Nick. "Smart Contracts." *Literature Review on Reaction Time*, www.fon.hum.uva.

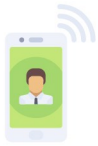
NEW PRODUCT

TestArchitect Team

PERFECT FOR SMALL TEAMS

AND YOU GET ALL OF THIS FOR FREE!

- Web Testing: Chrome, Firefox, Edge, Safari, Internet Explorer, Selenium WebDriver



- Mobile Testing: native, hybrid and web apps

- Windows Testing: .Net, Java, WPF, WinForms, Silverlight, Telerik, DevExpress, etc.

- Action-based Testing, Data-driven testing, Image Testing, etc.

- Web Services Testing: REST, SOAP, XML, JSON

Visit ***TestArchitect.com*** and **download TestArchitect Team** today!

TESTING WEB APPLICATIONS IN MOBILE EMULATION MODE

By Hien D. Nguyen

The huge range of mobile devices used to browse the web now means testing a mobile website before delivery is critical.

Developers use various techniques that allow mobile web applications to adapt to different mobile devices and screen sizes. Specifically the following:

- Responsive web design that lets websites conform their page layouts to various screen sizes and dimensions
- Platform detection that allows websites to present device-specific content

In a nutshell, testing on emulators helps verify that your mobile web applications look and work well on various devices, before you actually begin to test the web applications on real physical devices.

Mobile web testing can be challenging and tedious due to the large variety of mobile devices, platforms and screen sizes. TestArchitect addresses this challenge by letting you test mobile web applications on device emulators. Without the need for real physical devices of every type to improve test coverage, the result is a testing setup and maintenance that is highly simplified.

What is Mobile Emulation Mode?

Basically, Chrome DevTools' Device Mode is a built-in function of Chrome. This mode simulates a wide range of devices and their capabilities, so that you can test applications under test (AUTs) on a variety of emulated mobile devices, without the need for real physical devices. It simulates not just the browser environment but the entire device. It's useful to test things that require OS integration, for example, form input with virtual keyboards.

Automate Tests in Mobile Emulation Mode with TestArchitect

TestArchitect attempts to implement built-in actions in a manner that, from the standpoint of the AUT, is as close as possible to real user actions. The general workflow to test a web app in mobile emulation mode is:

HIEN D. NGUYEN

Hien D. Nguyen is an experienced Software QA Engineer at LogiGear Corporation. A tester by day and a blogger by night, Hien has a great passion for software testing, especially complex Test Automation problems. When not doing all those, he enjoys reading, jogging and trying new things.

1. Prepare Mobile Browser Profiles

The profiles are coded in JavaScript Object Notation (JSON), and with Chrome debugging protocol. You can define as many parameters as you need for a mobile browser profile.

However, it is recommended that your JSON string contains the following basic information:

- The user agent
- Whether to emulate a mobile device
- Screen width
- Screen height
- Device pixel ratio
- Whether a view that exceeds the available browser window area should be scaled down to fit
- Enabling of touch event emulation

```
[
  {
    "method": "Network.setUserAgentOverride",
    "params": {
      "userAgent": "Mozilla/5.0 (Linux; Android 4.4.4; Nexus 5 Build/KTU84P
        ) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.114
        Mobile Safari/537.36"
    }
  },
  {
    "method": "Emulation.setDeviceMetricsOverride",
    "params": {
      "mobile": true,
      "width": 260,
      "height": 640,
      "deviceScaleFactor": 2,
      "fitWindow": false
    }
  },
  {
    "method": "Emulation.setTouchEmulationEnabled",
    "params": {
      "enabled": true
    }
  }
]
```

2. Launch Mobile Browser Emulator in Device Mode on Chrome DevTools

		setting	value	
setting		use browser	Chrome	
		location		
navigate		https://www.google.com/		
		variable	value	
set variable		brower profile	[{"method":"Network.setUserAgentOverride"...	
// Invoke a specified emulator in Chrome's Device Mode				
		window	command	variable
send command to browser		google	# brower profile	>>emulator
		window		
refresh		google		

Now you're ready to begin performing your automated web-based tests on the emulator. For example, enter "TestArchitect" then click on the "search" button.

		window	control	value
enter		google	q	TestArchitect
		window	control	
click		google	search	

3. Change Environment to Customize Emulator Behaviors During the Test Run

You may wish to change the environment for the emulator to customize its behaviors (device orientation, emulate geolocation data, etc.). When the mobile browser emulator has been invoked, use the send command to browser built-in action again. Technically, "send command to browser" sends a JSON string request to Google Chrome to customize the emulator behaviors on-the-fly. Note that you can change as many behaviors as you want, as long as those behaviors are supported by Chrome debugging protocol.

Conclusion

Besides Android emulators and iOS Simulator, mobile emulation mode is another approach for mobile web testing without the need for real mobile devices.

To learn more about this TestArchitect feature, visit testarchitect.com.

HR ISSUES FOR TECH MANAGERS AND LEADERS: PART 2

By Michael Hackett

HR issues can be detrimental to a work teams productivity. So it's important to have HR knowledge and a resource to help you navigate these waters.

Handling "HR issues" well may in fact save your life and build your career. I have a friend who works at a Silicon Valley Tech/Media company. No, not a start-up. It's actually owned by a multinational that everyone and their mother would know. This company is a big revenue generator and is well established.

As big as this company is, they have no "HR team". Not even one HR manager or resource. Like many tech companies, the executives of this company believe that tech manager-leaders are "smart enough" to manage Geeks, and likewise, that Geeks need little HR management because they are smart. My friend tells me the management view at his company is: People need direction and work, people need to be hired, sometimes fired—but as long as Geeks have ping pong, tons of free snacks, and drinks with a happy hour on Fridays, it will be an easy, smooth running office, and that everyone is happy.

Until they are not.

And when that happens, whether it's a necessary firing, an H-1B issue, team "work from home" members vs "in-the-office" favoritism—which have each happened to my friend the manager/director over the past few months—it will badly impact a team's productivity.

Due to the many issues my friend experienced, it ruined his job satisfaction and he was unable to get any of his main revenue projects done until each of these problems were addressed. He is not alone as a manager in this company having these issues and wishing there was an HR resource that could help him navigate these waters. An HR resource would be beneficial in providing guidelines, sharing experiences about how to deal with these issues, possibly to sometimes handhold, or be a sounding board to walk him through these situations.

On top of these other problems, he told me of even more situations that had gotten out of control. Everyone wants the company to grow, but hiring more people meant getting rid of the ping pong table to add more work space. You would have thought taking away

MICHAEL HACKETT

Michael is a co-founder of LogiGear Corporation, and has over two decades of experience in software engineering in banking, securities, healthcare, and consumer electronics. Michael is a Certified Scrum Master and has co-authored two books on software testing. *Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems* (Wiley, 2nd ed. 2003), and *Global Software Test Automation* (Happy About Publishing, 2006).

this perk was akin to a 25% stock price drop. It caused threats of resignations and tons of ill-will. It was resolved, for better or for worse, but what no one at the time realized, was the massive drop in productivity during that period. Instead of productivity, there was stress, gossip, threats, meetings, emails, more gossip, more stress and more meetings. A release was postponed. Important customer support tickets went unfixed for days and real, useful work almost ground to a halt.

The moral of the story? HR issues are super important and often debilitating to Geeks! Jessica Stillman brought this into the light when writing for Inc. Magazine and shared *3 HR Mistakes That can Kill Your Startup*.

The top 3 reasons detailed by Stillman are:

- #1 – Not hiring an HR person
- #2 – Putting perks above communication
- #3 – Ignoring employee development

To me, #1 is the most important reason and this needs to be fixed. Tech companies need more HR help than traditional companies. If you are a manager, it is your duty to get HR help as soon as possible when any issues arise. Tech people are not always the best at handling complex and emotional situations; if getting things "Done" and staff productivity is part of your job, dealing with issues fast, without drama, and as transparently as possible is tantamount. This will not only keep teams happier and retain them, it will also keep them working productively in complex times.

HR Issues

In Part 1 of this topic, we discussed a variety of examples and situations focused on how HR tasks are often neglected in tech companies and the results when they are left to fester. Part 1 also covered a few situations including their impact on the team, communication, and respect for you as a leader/manager.

Here, in Part 2, we will focus more on hiring and firing, time management, and productivity.

A primary goal of writing on this topic is **not** to coach you through difficult situations. It is more to advise you on when to get more information, get support, or get trained. This article is about spreading awareness on these issues, to discuss some of their manifestations particular to tech workers, the not-so-obvious manifestations, and the by-products of problems. This article will also discuss leading tech workers and some of the new frontiers of employee relations in the new work reality.

Hiring and Firing

The Hiring Process

In Part 1 we began discussing hiring and firing:

It can be complicated. Every state in the US and different countries have laws and regulations on how this gets done. Get help.

"Hire slowly, fire quickly," can be very difficult to do.

There are a few more issues with hiring that deserve reinforcement.

Lying on Resumes?

First, many people lie on their resumes and about their experience. Inc. Magazine references that 85% of companies reported catching people being dishonest on their resumes. The numbers are staggering.

You need to learn how to spot these. Here are 2 ways to question if you suspect a gap:

1. Give a test

Use a skill test for multiple goals. See if the people actually have the skills you need or that they list on their resume, and also see how the applicant responds in a given situation, and how they respond under pressure. Use behavioral based interview questions, which can be found online. The concept behind behavioral based interview questions is to focus on the individual's past performance or behaviors. Past performance or behavior is the best indicator of future performance or success.

2. Check references

Checking references is mandatory. Check as many references as you can and for a few reasons, even when it can seem like a daunting task for a short stack of resumes. Since you know that many people lie on their resumes, you can use reference checks to question their experience, dates, and roles.

Also, make sure the chemistry is right. Having team interviews is

essential. You hire for communication and chemistry, not solely for tech ability. This is a common mistake some managers make. They hire for tech skill or because they like the person. Hire someone who will make communication and collaboration easier for the team as well.

Don't be in a rush to hire. If you have to rush it, set clear expectations at the on-set of employment, and closely follow up during a short probation period.

The Firing Process

Knowing that prolonged or non-firings usually have a corrosive impact on the team, you will need to do it quickly. We discussed this at great length in Part 1.

All guidelines suggest you have a script. Practice. Be prepared with supporting information and paperwork. Keep it short. A termination meeting is not a performance review discussion, the decision has been made and is final.

I remember working with a company that fell into hard times, as many companies do at some point in their growth. A product was cancelled. The rest of the company was OK. Unfortunately, the staff for which the product was cancelled, was going to be laid off. The job of the lay-off was distributed between a few managers. The higher up people did spend time thinking this through, but at a low level, the managers doing the task were not well coached or supported.

To put it simply, it did not go well. First, word spread immediately about what was going on after the first person was laid off. In place was a director focused on supporting the staff after they were laid off. In addition, I was there just checking in with the staff and the managers to see how things were going and triage if necessary.

In one instance, for this particular staff member–who some people foresaw as potentially difficult–the person firing did a few things wrong. When I checked in on where that particular manager was, we figured out he was still in the room alone with this staff member, 45 minutes after the start of their process. I knew something was wrong.

The task was conducted in his office, not a neutral space such as a conference room. Because of the setup of the room (accidently or on purpose) the staff member being laid off sat in a chair blocking the manager from getting out of the room. The employee was bargaining and justifying and had taken control of the situation. The manager did not know what to do, was blocked from the door, and was even unable to say, "excuse me a minute, let me go ask person X for some help on this…".

Few people realize firing is just as stressful as being fired, but there is more sympathy going to the staff being laid off than the person carrying out the deed.

I knocked, interrupted, and invited myself in. As I stood there, I said, "Let's move this to a room where all 3 of us can sit down." I took us to another room. Once we changed rooms, the dynamic quickly changed, which gave them both a moment to breathe. They quickly explained to me where they were. I said, "OK. Let's end this part of the decision now. Let's make sure your manager knows your idea and concerns and let him take that back to his managers and see what he can do, but for now, this meeting is over." That manager did go to his manager and got more information. They came to an agreement and both of them met with the employee.

There were so many things wrong in this situation. Getting help, coaching, and practice did not happen in that situation and it was far more stressful than it had to be for everyone involved.

It is important that you practice and brainstorm possible scenarios of how a certain employee may emotionally react. When you make a script and practice, you will be better prepared for the situation and can take control, should the termination go

toughly. These are some possible scenarios of termination: getting belligerent, arguing, bargaining, getting emotional, and making threats of legal action or other harm. If any of these scenarios occur, managers will need to remain professional, in control, and if necessary, end the meeting. This can be done by simply telling the employee that the meeting is no longer productive and another meeting will be scheduled when the time can be focused and productive. Even when these events go smoothly and amicably, they are rarely "easy".

Time Management

Every manager knows their main job is to simply get work *done*. Modern managers support their staff to create and produce. It's important that they keep knowledge workers working, communicating, and creative. Managers need to create an environment where teams feel supported in their work and are cleared from obstacles. This takes balance.

Everyone is busy. Everyone has too much to do and too many texts, chats, IMs, and team productivity tools that get in the way of productivity. Very few people have good methods of ignoring distractions, staying focused, cutting down on interruptions, and putting down their iPhones!

Keeping Focus

There is a method used for time management and productivity that is widely popular among tech people. Some tech companies I know of, offer training in it. Every time I bring it up in a group of Geeks, undoubtedly, only a few people know it. It's the Pomodoro Method. See the sidebar for information and links. It is based on an optimal focus time of 25 minute blocks with breaks of a few minutes.

My experience with this method is that it is highly successful for me to use in my mornings as it gives me uninterrupted blocks of time for maximum focus and productivity. My afternoons are structured differently. Since I often have longer meetings or need bigger blocks of communication and collaboration time, I only occasionally use Pomodoros. But mornings are for Pomodoro. Learn about them. The idea comes from having a timer on your desk, either an Italian style tomato (Pomodoro) timer, from which this practice gets its name, or an American style egg timer to measure out the blocks of focus time.

Here are the problems I often see:

1. Other people in the office or team are not doing it and don't like it, which is too bad. You are missing out on protecting your brain and being more effective!
2. One of the purposes of the Pomodoro (tomato timer), egg timer, or kitchen timer is that people see the timer on your desk. They know what that means and leave you alone. If someone needs/ wants to talk to me *now*, the longest they have to wait is

Pomodoro Technique

Tomatoes and Time Management

Developed in the 1980's by Francesco Cirillo, the Pomodoro technique was never predicted to contribute and influence the modern world with time management as much as it has done today. It has inspired and become wildly popular with tech workers. This technique essentially uses a timer to break down tasks into intervals (Pomodoro) of 25 minutes in length separated by short breaks. The technique itself was named after a Pomodoro kitchen timer, Pomodoro being the Italian word for 'tomato'. The main goal of this low-tech approach is using the Pomodoro, the interval of time spent working, to enhance efficiency in work and reduce the impact of interruptions on focus or flow; training the mind to associate flow and focus with the physical stimuli of winding the timer, the ticking, and ring. Time becomes an ally and not a force to struggle against through the mastery of this technique (link to Francesco's site). Here are some Apps.

25 minutes! But if I am focused, send me a mail or text saying, "when you take a break come by my desk or call or whatever." For the sake of your brain not multitasking or context switching, it is worth it.

3. If it's an emergency, fine, interrupt me. Emergencies are rarer than you think.
4. There are many, many desktop and mobile apps specifically for this method with 25 minute timers readily available to you. It's an easy practice to do.

The idea of individual focus and productivity may seem to run counter to an idea we constantly bring up again and again: collaboration and communication!

They actually go best hand in hand. When you are working alone and focused, make sure you stay focused. When it is time to collaborate, have a good space for that. If you are not face-to-face and need a tool, set aside time for it…in my experience, collaboration is best done in blocks of time rather than a random or constant states of interruption.

An Optimal State of Collaboration

There is an optimal state better than being so focused on separate times for quiet work vs collaborative work. This requires a few pieces to be put in place to be successful. Such pieces include:

- Team members working at the same table so they do not have to walk across an office to talk to each other or yell across the office to talk to each other.
- Team members focused in the same context area to minimize context-switching. Eliminate context switching so teams can be, as the phrase goes, hyper-productive. Different than multi-tasking, context switching is the process of storing the state of a process or of a thread so that it can be restored and execution resumed from the same point at a later time. This stopping and resuming has a brain cost many people do not realize.

The issue I often see with teams in very close quarters or sharing the same worktable is multiple unproductive conversations crossing each other about:

> The label on a new column in the database. A bug found in production that is being triaged. A changed parameter from an API of another team. A restaurant someone went to over the weekend…

With all these conversations happening at the same time, team members are in a constant state of interruption instead of being hyper-productive.

The first step is to enlighten people of the issues around productivity, context-switching, and multi-tasking as well as to provide a separate space for more interaction or more focused work. If you can't identify with the situation mentioned above, your team usually stays focused, on target, and are respectful of other people in the room, perhaps—constant collaboration will work for you. I have seen it work, but only rarely.

Make sure your team is aware of the mental cost of multi-tasking and task switching. It may be prudent to instill techniques like the Pomodoro Method to help them focus, collaborate, and be productive in whatever their tasks demand. If your team is already super focused and sits in a highly collaborative workspace with little context switching, good for you!

Summary

HR issues and normal work issues can be a minefield. For tech leaders and managers, they will be your greatest test.

In many tech companies, there is a lack of support, information, and knowledge about how to navigate these sometimes easy, sometimes very difficult, but always unknown and unforeseen waters.

In the situations described in Part 1 & 2 here, I share experiences where a seemingly straightforward normal work situation had unforeseen consequences, such as productivity hits, you being judged and getting a reputation, for better or for worse, on how you handle or lead through these issues, and the unique demands on knowledge workers, Millennials, and tech workers in general for honesty, transparency, fair treatment and good work situations. They are your greatest tests. Be prepared! Learn a lot. Get help.

FEATURE

SOFTWARE TESTING: IS AI IN YOUR FUTURE?

By Regg Struyk

Artificial Intelligence is the latest trend to emerge. Understanding its role in software testing is critical.

I am always looking to see what the next big trend in tech will be.

Some of the recent trends include Agile, DevOps, and Internet of Things (IoT). Now the new trend appears to be Artificial Intelligence.

AI in Software Testing

In 2017, Tech Giants such as Amazon, Facebook, Google, and Microsoft spent billions on AI and Machine Learning initiatives. At Build 2017, Microsoft announced Microsoft Cognitive Services to help developers integrate AI into the applications they are developing.

It looks like AI is just going to keep growing and there is clearly a role for it to play in software testing, too.

Tools Will Need to Evolve or Die

Software development—and software testing—continues to evolve in step with the adoption of Agile and DevOps methodologies. Software development will continue to evolve in the era of AI, as well.

AI and Machine Learning are centered on training software to understand input data versus output today. This is very similar to the testing activities performed manually today. We type an input into a field, and we look for an expected output.

With AI, the machine does the testing instead. It comes up with many more variations of the test. It automatically runs many more tests than a human could supervise. And it even handles

REGG STRUYK

Managing Director, eLead

Regg is a co-founder of eLead ALM, focusing on software solutions for improved product quality and delivery. He grew up in the '80s on the cusp of the technical revolution, which has fueled his 25-year passion for software development. Regg is considered a requirements Sherpa with focus on various types of testing, user behavior, Automation, and anything next generation.

This article originally appeared on Perforce at: https://www.perforce.com/blog/alm/software-testing-ai-your-future

changes to the code and UI that previously had to be made by QA professionals. Some examples of this are adding fields or changing the inputs and anticipating outputs.

Even without AI, testing tools have evolved. Today's tools help testers create, organize, and prioritize test cases. Efficiently managing tests and their outcomes, as well as remediating defects, remains essential to giving the developers the feedback they need.

Here are two things to consider when it comes to software testing tools (with or without AI):

1. Scalability

Testing data is growing. And spreadsheets alone can't report on trends in test data or quickly rerun tests. Your organizations need more from testing.

Test management tools need to evolve for scalability. Testers should be able to use these tools to support quick test reruns or Nth number of test cases.

2. Silos

Product development phases are no longer isolated. Today, each phase of product development is an integral part of the development lifecycle.

Application lifecycle management tools need to support the entire development process. That means gathering requirements, managing tests, and performing traceability.

Testers Will Become Extinct

AI is going to replace testers as we know them today. Instead of being a QA team, software testers will be a Test Automation team.

That's because AI is going to make testing more efficient. Test Automation engineers will be the subject matter for experts teaching AI to execute the tests—and many more tests and permutations—faster than they could do manually.

Humans will supervise the outcome of the test, adding the critical human element. After all, there are three types of requirements for testing (implicit, explicit, and latent). But only explicit testing can be done by AI.

"A lot of things are going to change in the testing field with the entry of AI. Almost 70 percent of testing is repetitive and AI can quickly occupy that space. The 30 percent left is questioning the system, and that's what testers need to focus on. AI is the next big thing in testing, but it won't replace humans. The testers working alongside AI can quickly revolutionize the way we test today."

Vijay Shinde, Founder of Software Testing Help

Testing Always Changes

You'd be hard-pressed to find any software QA professional who would say testing is the same today as it was five years ago.

And similarly, AI is going to play an important role in the future of software testing. But it's not going to change software testing overnight. And there are fundamental ideas about testing and quality that will remain (and that only humans can understand) for the foreseeable future.

Learn more about the impact of AI on software testing. Watch a recent webinar: "3 Ways AI Will Change Software Testing."

Artificial Intelligence

Artificial Intelligence is here. AI is already being used in development, testing, tool development, and products. Its use will undoubtedly grow and become more pervasive.

At LogiGear Magazine we regularly include a glossary of terms on the issue topic. In this issue, the glossary is unique.

If you are just beginning to learn about AI there are a whole lot of unique phrases in understanding to grasp the terms and possibilities of AI. The goal here is to maximize your understanding of AI and Machine Learning so that you can do a few things:

- See where AI could apply to your testing practice. For example, using AI to predict where bugs may lie in wait. Knowing this could help you build data, bug tracking information, or bugs traced to the chunks of code where the fix is to help build an AI bug prediction system.
- Understand AI enough so that when you begin using an AI-based Automation tool, or AI-based Test Automation helper tool, the more you understand about AI, you can think of more ways to apply the tool or increase its use.
- If you are testing an AI or Machine Learning app, you will need a great understanding of how AI works. Testing AI systems is far different from virtually all systems you may be testing already.
- For the testing of AI or Machine Learning, part of Machine Learning is that with use, the machine learns more or predicts more. With a certain input or data into an algorithm today, the machine will learn and give you a different answer tomorrow. Most of us do deterministic testing. A+B=C. We expect C to be the answer based on some information such as a requirement/user story or subject matter expertise. With Machine Learning, today A+B may =C, but tomorrow it may =D. To understand this, see the word, non-deterministic, in the following glossary.

For a full AI Glossary, please visit Google ***and*** Wikipedia.com

Artificial Intelligence Terms

Artificial Intelligence

Intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".

Source: https://en.m.wikipedia.org/wiki/Artificial_intelligence

Artificial Neural Networks

Connectionist systems computing systems vaguely inspired by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs.

Source: https://en.m.wikipedia.org/wikiArtificial_neural_network

Bot (Robot)/Bots

Machines that can substitute for humans and replicate human actions. Robots can be used in many situations and for lots of purposes, but today many are used in dangerous environments (including bomb detection and deactivation), manufacturing processes, or where humans cannot survive (e.g. in space). Robots can take on any form but some are made to resemble humans in appearance.

Source: https://en.m.wikipedia.org/wiki/Robotics

Data Mining

The process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

Source: https://en.m.wikipedia.org/wiki/Data_mining

Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from data in various forms, both structured and unstructured, similar to data mining

Source: https://en.wikipedia.org/wiki/Data_science

Deep Learning

Part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised, or unsupervised.

Source: https://en.m.wikipedia.org/wiki/Deep_learning

Deterministic Algorithm

An algorithm that, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.

Source: https://en.m.wikipedia.org/wiki/Deterministic_algorithm

Explicit Requirement

Most commonly found in documents communicated by stakeholders to the development team. They might take the form of an elaborate design specification, a set of acceptance criteria, or a set of wireframes.

Source: https://www.stickyminds.com/article/3-types-requirements-testing

Implicit Requirement

All the things that users are going to expect that were not captured explicitly. Examples include performance, usability, availability, and security.

Source: https://www.stickyminds.com/article/3-types-requirements-testing

Latent Requirement

Represent behaviors that users do not expect based on their previous experiences but which will make them like the software more.

Source: https://www.stickyminds.com/article/3-types-requirements-testing

GLOSSARY

Non-Deterministic Algorithm

An algorithm that, even for the same input, can exhibit different behaviors on different runs, as opposed to a deterministic algorithm.

Source: https://en.m.wikipedia.org/wiki/Nondeterministic_algorithm

Machine Learning

The study of algorithms and mathematical models that computer systems use to progressively improve their performance on a specific task. Most machine learning systems are based on neural networks, or sets of layered algorithms whose variables can be adjusted via a learning process

Source: https://en.m.wikipedia.org/wiki/Machine_learning

Predictive Analytics

Encompasses a variety of statistical techniques from data mining, predictive modelling, and machine learning, that analyze current and historical facts to make predictions about future or otherwise unknown events

Source: https://en.m.wikipedia.org/wiki/Predictive_analytics_

Blockchain Terms

Blockchain

A growing list of records, called blocks, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.

Source: https://en.wikipedia.org/wiki/Blockchain

DApp (Decentralized Application)

Applications that run on a P2P network of computers rather than a single computer.

Source: https://blockchainhub.net/decentralized-applications-dapps/

Ethereum

An open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract functionality.

Source: https://en.wikipedia.org/wiki/Ethereum

Ethereum Improvement Proposals

Describes standards for the Ethereum platform, including core protocol specifications, client APIs, and contract standards.

Source: https://eips.ethereum.org/

Smart Contracts

A computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties.

Source: https://en.wikipedia.org/wiki/Smart_contract

DevOps Terms

Continuous Integration

The process of automating the build and testing of code every time a team member commits changes to version control.

Source: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration

Continuous Testing

The process of executing automated tests as part of the software delivery pipeline in order to obtain feedback on the business risks associated with a software release candidate as rapidly as possible.

Source: https://www.tricentis.com/what-is-continuous-testing/

This is a lot. We hope this will help broaden your knowledge of AI and the associated issues so you will be on board and ready to take full advantage for your inevitable work using AI as part of your development and testing practice or, if you are lucky, developing and testing AI products.

24 Years of
TESTING & DEVELOPMENT
EXCELLENCE

LOGIGEAR MAGAZINE

DECEMBER 2018 | VOL XII | ISSUE 4