Ascert

AUTOMATED END-TO-END TESTING & CERTIFICATION

Green Screen, in 2018, seriously?

It's hard to say what is most surprising. To be sitting in the last throes of 2018, writing an article about green screen applications. Or to have recently found myself deep in the bowels of 3270 and 6530 terminal data streams writing code to automate the testing of them. More accurately, that should really read 'porting code' - since my firm, Ascert, has had a TAL screen handling and testing library since I first joined (and worked on it) back in 1997. Even then, I think most of us assumed it was the last chance saloon for a technology that would soon go the way of the dinosaurs, disco, and flares. But time and the future makes fools of us all I guess. Especially those whose predictions of the death of the mainframe have proved somewhere between premature and wildly inaccurate.

At least the cause of this peculiar turn of events is simple to explain – a customer needed it. Rather less easy to explain is why this wasn't just one isolated customer, but a steady stream of requests from different sources over a period of months. Having barely had a single conversation over testing this style of application for the last 20 years, all of a sudden it was a hotter topic than contactless EMV testing, or Open Banking APIs. Being more of a practitioner than an analyst though, I parked my curiosity as to this resurgent interest and got on with the far more solvable task of understanding what was required and cracking on with delivering it. In so doing, I unwittingly uncovered some pieces that could help explain what was going on.

Modern Test Automation for Legacy Applications

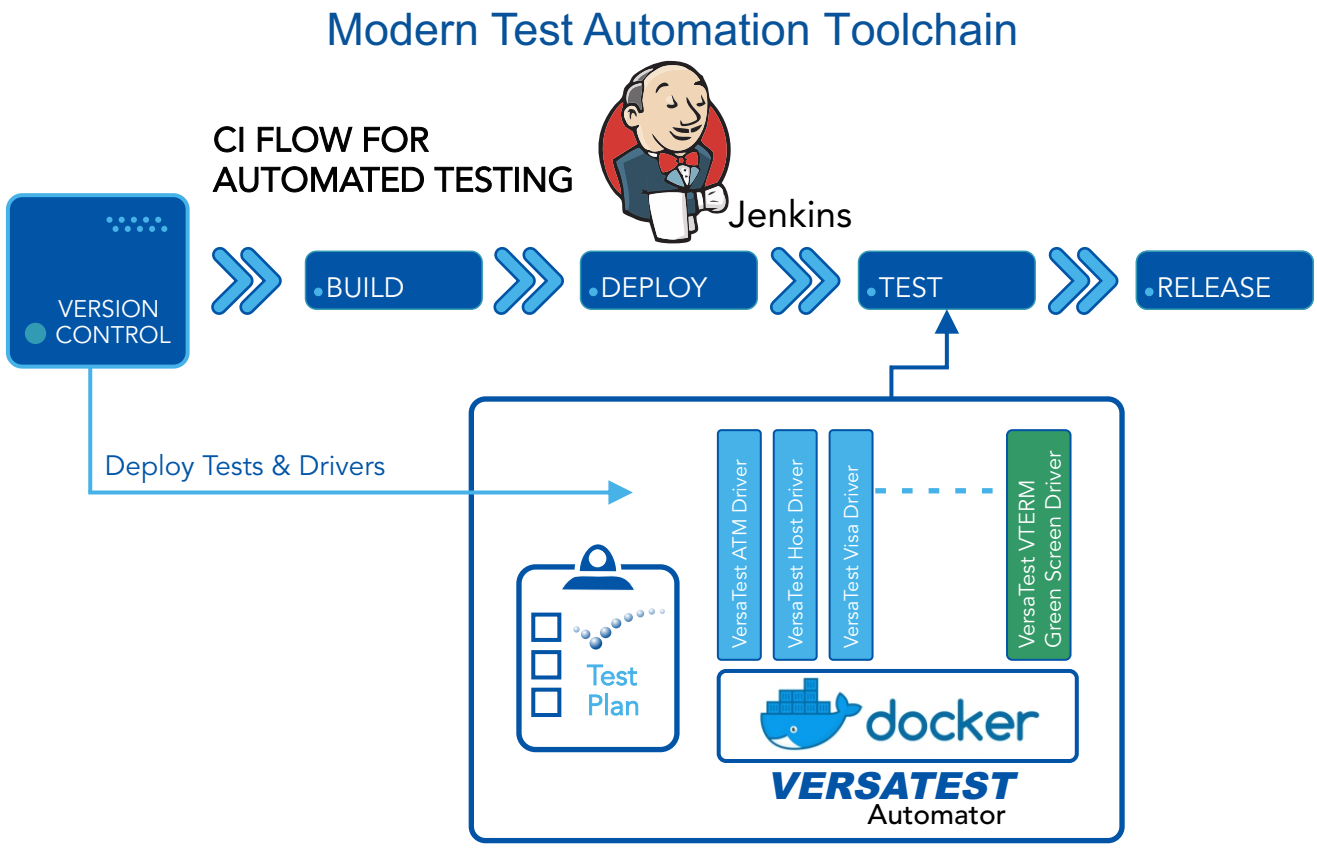
The first, and perhaps most unexpected piece, was that far from declining, according to at least one recent study mainframe use is actually growing again! It seems that the mass migration of applications away from mainframe data centers, that began with the client-server boom of the early 90s and continued through the cloud hosting and serverless computing trends, may have reached its zenith. Those applications, which could (arguably should) belong on cheap and widely available commodity infrastructure, have moved. Those which remain are too critical, risky, or expensive to consider migrating. Whether you consider IBM and HPE NonStop applications as classic, well engineered solutions built to last, or bygone relics that you are stuck with maintaining, the big iron systems we are left with today simply aren't going anywhere. There's no choice but to embrace the challenge of streamlining the management and testing of them within our modern, probably agile, practices.

Truthfully, I'd rather have avoided the word 'agile' for the entire article if it were possible – just mentioning it can be riskier than announcing you're a Red Sox fan before noticing you're in a Yankees bar. But as the second of those pieces I stumbled upon, there's no way around it. Fortunately we're going to stay on the firmest of ground, because in amongst Scrums, Sprints and the plethora of other ethereal terms, there is one whose value is beyond reproach: Continuous Integration, or CI for short.

Even last-generation developers like me have no trouble understanding and seeing the benefit of CI. The idea that code which has been newly pushed to the central version control repository should be built, deployed, and tested as soon (and as automatically) as possible makes perfect sense. It's using computers to do what they do best – rapid, repetitive and mundane tasks. Tools such as Jenkins provide us a conductor able to orchestrate the required steps across heterogeneous platforms. And tools such as Docker give us a dynamic and lightweight model to spin up instances of those platforms to deploy our code and test tools onto. All of which works brilliantly until our automation chain runs smack into that big block of iron – the mainframe application. What if we need to login to a block mode screen to perform a transaction as part of our testing? Or run an enquiry into the mainframe and screen scrape the results to validate against the values expected by our other, distributed application components? Which was exactly the need articulated by our customers as we delved into their requirements for green screen testing. They didn't need a terminal driver rig that could run on a NonStop. They needed one which could run inside a headless Docker container, under the control of a full-scale Jenkins CI build and test process. Whether it's agile I'm not properly qualified to judge. But the end product they achieved was certainly impressive. A high degree of automation and use of modern tools and techniques to encapsulate the knowledge of how to test their old school applications.

That 'knowledge encapsulation' aspect was the final piece of the jigsaw that dropped into its rather uncomfortable place. Every morning when I wake up, for a few brief moments, there's that same youthful spirit of the 19 year

old who got his first programming job at Honeywell Information Systems. But one look in the mirror soon dispels that myth as the 54 year old lifer stares back at me. And yet, in terms of green screen terminal systems, I'm one of the younger developers who understands and has worked with the technology. The use of mainframe systems may be on the rise again, but that doesn't mean colleges and universities are churning out graduates to support them. The existing pool of skills is drying up rapidly as existing staff retire. There's a finite and shrinking amount of time to transfer their knowledge of how to maintain and test them. Encapsulating that knowledge within a CI rig makes perfect sense. Organizations are gradually realising that work has to be done now, or there's a risk it won't be possible to do it at all.



As shown in the diagram, the modern test automation toolchain is really a close parallel of the CI toolchain. Everything starts with the basic unit of change control – a distributed version control system (Git, Mercurial etc). Next comes an overall controller to co-ordinate the process (Jenkins, Atlassian Bamboo etc). Finally we need somewhere to deploy our test tools, environments, and scripts. Virtual Machines are a decent option for this of course, but the flexibility of Container solutions (Docker etc.) are a near perfect fit to the dynamic, rapid, and scalable server capacity required. Of course, when it comes to Green Screen, the challenge is finding (or in our case developing) a 6530 and 3270 simulation layer that is capable of running efficiently within such a headless container solution. With that in place, the Jenkins (or other) CI orchestrator can look after the entire test automation chain.

At this point I'm tempted to say this will be the last article, and code, I write around green screen applications. But I've said (or at least thought) that before. One part I am confident in though is that we have at least given those dinosaurs, which it turns out weren't extinct after all, a nice new home. They're living comfortably alongside their more agile descendants in the same, thoroughly modern, CI toolchain.

Modern Test Automation for Legacy Applications

Author Bio

Rob has been a developer of testing and simulation tools for 30 years, a majority of which has been with his current firm, Ascert, LLC. In the early days this frequently involved hand crafting automation rigs in system languages such as TAL, C/C++, and scripting languages such as TACL and Ascert's own VersaTest VTALK. With the widespread uptake of open platforms came fresh opportunities to learn new languages and techniques, Java and Javascript amongst others, to support the process of delivering server and cloud based solutions with web-enabled front ends. Although now one of the 'old-guard' of software developers, one of the aspects which Rob most enjoys about his daily work is the challenge of remaining current with technology trends.

About Ascert

Ascert is recognized as a leading provider of premier testing software solutions. Ascert was founded in 1992 to provide automated software testing solutions that help companies measure the performance, reliability and scalability of their mission-critical back-end servers and applications. With over 100 clients worldwide, Ascert's products and services are used at some of the world's most successful companies.

Advantages of VersaTest Automator

True automation of testing results in huge benefits. Test runs that previously took days or even weeks to execute can be completed in a matter of hours providing significant return on investment through time savings alone.

- No scripting required, enables use by a wide range of personnel
- Enables frequent regression and performance testing with more confidence in results
- High level pass/fail management view of testing with drill-down for detailed analysis
- Integrates with 3rd party products like Jenkins and HP Quality Center

Ascert

www.ascert.com/go/test

USA
Ascert, LLC
759 Bridgeway
Sausalito, CA 94965
+1 (415) 339 8500
info@ascert.com

EMEA
Ascert, Limited
6th Floor DBP House
63 Mark Lane
London EC3R 7NQ UK
+44 (20) 7488 3470
ukinfo@ascert.com