The State of Automation

Going beyond the easy stuff with Quality Center

Rob Walker
Managing Partner
Ascert, LLC

Rob is a Managing Partner at Ascert LLC, based in Cape Town, South Africa. He has worked with NonStop for almost 30 years, starting in the financial industry and later specialising in the testing products sector. In 1997 he joined SoftSell Business Systems, which later became Ascert, and was instrumental in extending the testing product range from it's original NonStop heritage to an open toolset supporting testing of any hardware and software platform.

Walk around the QA and testing departments of most enterprises with large or critical IT systems and you're almost certain to find HP Quality Center screens on more than a few desktops. With a majority share of the market, whether you love or hate it, Quality Center has become the tool of choice for large organizations to manage their QA processes as part of their strategy for application modernization.

With such ubiquity you would be forgiven for assuming that Test Automation would be a baked-in, part of the standard solution. But that's where we enter the twilight zone – a dimly lit world at the edges of Quality Center's boundaries, where automation is often mentioned but seldom seen. To be fair to Quality Center, test automation is hard, with complexity that escalates rapidly in the face of the vast and disparate universe of modern and legacy systems, devices, user interfaces etc. The best that an off the shelf product can realistically offer is hooks as an entry point for managing test automation: a docking module to the outside world that enlightened customers will figure out how to use. And that's exactly what Quality Center offers, and has done since at least Test Director 8.0.

With such facilities available, you'd expect that on our imagined walkabout through Big Corporate QA, we would see very few of those Quality Center screens involved in manual testing activities. Sadly, that's not the case at present. Despite its widespread use, surprisingly few Quality Center installations take full advantage of that docking module to reap the full benefits of test automation.

Is this really a problem?

Whether you have a Quality Center automation gap or not will depend very much on the type of applications you are testing. If your testing revolves solely around interacting with and validating a standalone browser application, or a Windows GUI application at a PC, then you're already well served with automation tools. You can skip the rest of this article and get back to your day job.

Still reading? Don't be discouraged, you're amongst friends, a sizable community whose enterprise application testing needs don't fit such simple, one dimensional solutions. Even those cases where an application has a natural screen based user interface, chances are that the real work is being done by all manner of middleware and back-end interfaces, silently going about their business of exchanging messages and updating databases. The NonStop community will recognize this problem more than most, with its widespread deployments of ATM and POS systems, financial interchanges such as SWIFT and ISO 8583, mobile banking systems using SOAP messaging, cell phone switches using ASN.1, stock exchanges etc. All of these are large, message based, transactional applications where the system's real business value passes through internal, headless interfaces that are difficult to integrate with a Quality Center world.

With no window or HTTP form for a client side testing tool to hook into, message based applications don't fall into the group that can easily be tested by tools such as HP's Load Runner or Quick Test Professional. There may be tools to help with the testing of such interfaces, but not all integrate with Quality Center, and even fewer support heterogeneous tests involving a mix of steps across different devices or interfaces. Without a comprehensive level of integration and support, device and message based interfaces become relegated into the *"too hard"* category, adding to the already longer than desirable queue for manual testing and verification. Quality Center, for all its capabilities, ends up as little more than a structured documentation tool for some of the most vital interfaces and valuable transactions within an organization's IT systems.

There are, for sure, a number of areas where manual testing is still the most effective approach. And HP, with its Sprinter tool, offers ways to better support those remaining manual test cases. But should we fall back on that so easily for device and messaging interfaces? The truth is that with their well defined and precisely specified operations, these interfaces are just as worthy candidates for test automation as GUIs and web pages, even if they are somewhat trickier to connect into. And because they are critical interfaces to our business, we should be looking to manage and automate that testing through the same, well established and proven Quality Center processes that we use for our other areas of testing. We simply aren't doing our job as testing professionals if we leave these languishing in that *hard-to-automate* bin.

Getting down to the bare metal

By now, hopefully most readers will have been persuaded that test automation for applications with headless and legacy interfaces is both desirable and if not easy, at least possible. Which takes us on to the next problem - How?

New users of Quality Center quickly become familiar

with the *New Test* button in the *Test Plan* module, where as well as naming the test the user will select what *Test Type* it is. The test type chosen is the hook for automation (or not, in the case of the MANUAL test type) – it tells Quality Center which docking module to use to reach the outside world. Chapter 20 of the ALM 11.0 User Guide has a table listing available test types, some of which integrate with HP's other testing tools:

- LR-SCENARIO – will use HP Load Runner as the testing engine
- QAINSPECT_TEST – will use HP QAInspect as the testing engine
- QUICKTEST_TEST – will use HP QuickTest Professional as the testing engine
- SERVICE-TEST – will use HP Service Test as the testing engine

The automation capabilities and limitations of these tools are already well covered on the web, so we won't review them here. But before considering these cases as *"problem solved"* in terms of automation, there are a couple of points that should be noted:

- A *Design Test* can only have one *Test Type* - if a test has steps that need actions across multiple different tools, automation becomes a lot less straightforward.
- The model used by most of these tools is to hold the actual test definition in their own internal format, typically via the Test Script tab or as an attachment to the test. This disconnects the automated definition of what the test does from the *Description* and *Test Steps* fields within Quality Center. That's not ideal if you want to be sure the test actions performed automatically are actually the same as those the Quality Center description says should be executed.

These issues sound relatively minor at first, but their impact on real-world tests are too frequent and damaging to be considered boundary cases. Take, for example, the actions involved in a typical cash machine transaction:

1. request a cash withdrawal at an ATM device
2. look for an associated authorization request at the host interface or card interchange and respond back
3. look for the correct response arriving at the ATM
4. finally go check a green screen or GUI and make sure the transaction reflects correctly in the balance and logs

Even this most basic transaction needs us to hook into three different tools, even though only one test type can be allocated and the associated tool most likely needs it's own separate test definition in an external and proprietary format.

Beyond standard, single tool test automation

All is not lost though, that list of Quality Center test types has some extra tricks up its sleeve to help us:

- VAPI-XP – a powerful, *do-anything-you-like* test type with support for an automation script in Microsoft VBScript, Javascript, PerlScript, and PythonScript.
- Custom Test Types – a published Quality Center API open to anyone to create their own custom test type. Such custom test types are first class citizens in the Quality Center world, sitting alongside and with equal power as those supplied for HP and other testing tools.

There is a lot of good documentation on VAPI-XP, the current versions of which can be found in Chapter 27 of the ALM 11.0 User Guide. We won't repeat the information here, except to point out some noteworthy behaviors that are common to all flavors of VAPI-XP:

- They create an external test script, which although viewable and editable in the Test Script tab is actually held in a separate repository of flat files on the QC server. The script file is local to every test created – if you want a common automation script across tests you must create your own mechanism to duplicate a template script into every test, and a way to update all copies when the script changes.
- There is only one test script file. When you do a *Run Test* on such a test type, the entire script will be run once, regardless of whether the Quality Center test has actually been broken down into separate test steps. There is no concept of multiple entry points for the individual steps of a test.

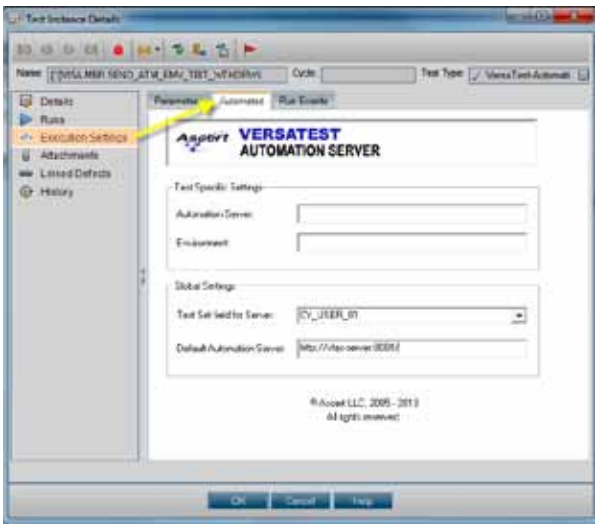
These pose some challenges to VAPI-XP as a candidate for creating a generic *"test automator"*. Don't be discouraged though. They can be worked around, and as a place to start exploring better test automation, VAPI-XP is both powerful and quick to get started with. One of the first full automation rigs created by Ascert for a customer in 2004 used a combination of VAPI-XP, Javascript, and VersaTest running on remote machines. Despite having a rather cumbersome infrastructure, it worked well and provided a high degree of automation.

The documentation on Custom Test Types is a little less comprehensive. Technical documentation and examples on packaging and creating them is there, but high level documentation on how all the pieces fit together is a little sparse. This is a shame, because despite being a tricky feature from a programming perspective, once you have it mastered it offers the most powerful approach currently available for test automation within Quality Center. Whereas it's possible to do most things in VAPI-XP, Custom Test Types provide the ability to package them up in a much simpler form for the user e.g. meaningful test type names and icons, custom panels for configuration, viewing etc.

The screenshot below shows how a Custom Test Type's configuration panel is seamlessly integrated with the standard Quality Center user interface, in this case providing an easy way for users to select which servers and environments are to be used for executing automated tests.

Before we move on from APIs to automation models we must answer a question that will be troubling observant readers. Why haven't we discussed the REST API, introduced by HP in ALM 11 and significantly expanded in ALM 11.5? The answer is that at present, it doesn't offer advances for automation of tests from within Quality

▶



Center. That's not a criticism of HPs enhancements. Support for an open, XML based API is a welcome step that provides a cleaner and less platform specific way to access the Quality Center repository. But it isn't yet clear how HP intends to provide a similarly platform neutral API for the *Run Test Set* functionality needed to execute tests from within Quality Center's own user interface. An open approach to replace this functionality certainly won't be able to use the current model, which relies on launching agent programs on local or remote Windows based machines using ActiveX and Remote DCOM wiring. Until then, VAPI-XP or Custom Test Types are our best bet.

Towards a unified automation model

Although slightly different under the hood, both VAPI-XP and Custom Test Types provide a pathway towards a truly automated test environment within Quality Center. Both approaches provide a way to invoke custom automation code when a user clicks *Run Test Set*, and both allow automation code to use the Open Test API (OTA for short) to access the entity model within Quality Center, in particular the tests and test steps defining the actions to be performed.

That last part is so important to an effective automation model, we'll say it again – *"automation code can access Quality Center tests and test steps to decide what to do"*. Why re-invent the wheel and hold a separate and external automation script when Quality Center already has standard database fields that can hold it for us? Quality Center's model may not mirror how everyone thinks about testing, but it is sufficiently malleable that it can be made to fit most concepts of structured test organization. Using this model directly is such obvious best practice that it's a shame so few of the testing tools that do integrate with Quality Center follow this model.

By using standard Quality Center fields to store our automation description, we are forced to adopt a language that can be stored in text based fields. Rather than being an extra chore, this turns out to be a very good thing. Doing so creates an action definition that as well as being automatable, is also highly readable, as shown in the following simplified example:

```
action := send
type := cash_withdrawal_request
amount := 50.00
card := 4929 1234 0000 5678
```

The automation code invoked by VAPI-XP or within the Custom Test Type uses OTA to read definitions such as the above for each step, parses them, performs the required actions, and returns the results including supplementary logs and application files back into Quality Center.

We're now very close to our goal of a unified automation model, but there is still one problem remaining – how to handle test steps needing action at different interfaces. With all the pieces we now have in place, achieving this is easier than you might think. All that is missing is a qualifier telling our automation code the interface or tool that each test step involves, as highlighted below:

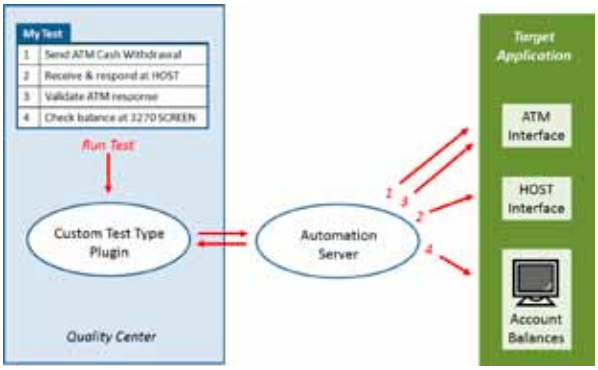
```
interface := ATM
action := send
type := cash_withdrawal_request
amount := 50.00
card := 4929 1234 0000 5678
```

With this in place, we have everything we need to use standard Quality Center *Test Steps* to contain interleaved sequences of actions to be performed automatically across multiple different application interfaces.

Before we move on to looking at a working implementation of these concepts, there is one last aspect to be noted about this automation approach. We've taken care to build a model where each *Test Step* in Quality Center defines an automated action to be performed at some interface. It is equally important that our automation component preserves this model during execution by creating result steps that when viewed in *Test Lab* will mirror the test steps defined in the *Test Plan*. As with the other parts, OTA provides access to all the necessary entities to achieve this.

Putting it all together

The following diagram builds on the concepts discussed to show our original 4 step multi-interface example with the automation components in place



The diagram follows the concepts discussed here, but takes them a stage further in implementation by splitting

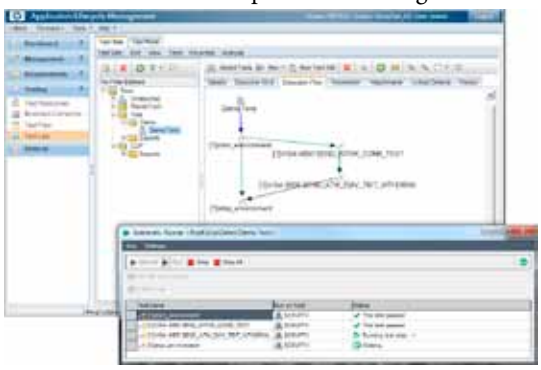
continued on page 46

The State of Automation

Continued from page 22

out the *"automation intelligence"* into a separate Automation Server. One of the drawbacks of Quality Center's custom test type architecture is that it requires plugins to be installed on the Quality Center server. At sites with large numbers of users, this can become a maintenance nightmare when a plugin update needs to be re-issued to every client in the organization. Moving the bulk of the automation logic out of the plugin and into a separate server lowers the frequency with which the plugin needs to be changed, and also creates a level of platform independence. The Automation Server is no longer restricted to only running on Windows platforms.

Another area of refinement that can prove useful is giving real-time feedback during test execution. Although Quality Center has no concept of individual test steps in the *Run Test* panel, it is possible to feed these back in status text messages to give the user a clear indication of test progress. Whilst on the subject of real-time feedback, we should also mention the importance of returning immediate pass/fail statuses back to Quality Center as each *Test* in a run completes. These form an integrated link between your automation model and other Quality Center features such as the Dashboard and the Execution Flow facility within Quality Center. The screenshot below shows both of these aspects in action together.



Is the effort worth it?

If everything we've talked about in this article sounds like a lot of effort, you'd be partially right – it can be. Which brings up the question of whether it's all worth it, or should we just allow those tests to remain in the too-hard bin, and leave our manual testers to execute and verify them.

The good news is there are ways to reduce this part of the modernization effort. HP has an active community of Quality Center partners, and offer tools that implement the approaches and practices discussed here, such as Ascert's own *VersaTest Automation Server Plugin for HP Quality Center*. Such plugins may not actually write your automated tests for you, but they provide you a framework to get up and running quickly.

Whether you start small with your *Smoke Test* or *Top 50* transactions, or go for a full Business As Usual (BAU) regression suite from day one, there are a lot of benefits to be had in return for automating your regularly run test packs:

- Automating not just the execution of tests, but also the far more time consuming and error prone aspects of validating and recording the results, even for tests with steps spanning different interfaces and test tools.
- Being cost and time effective to run a complete regression pack for every change and new release, rather than relying on a *"we don't think anything else should have been affected"* risk strategy simply because you can't manually run all of the tests you'd like.
- One single, common definition of your tests, held entirely within Quality Center, with far less chance that an external *"automation script"* can get out of date with the description held in Quality Center.
- Using Quality Center as the primary platform for definition and execution of automated tests reduces the number of staff to be trained in other tools.
- Subject Matter Experts (SMEs) are typically involved when changes are first implemented, but have usually long since moved on when an application reaches a BAU stage. A properly implemented automation model lets you capture the testing knowledge from these SMEs before they leave for other projects, and ensures that nothing is lost when repeating tests in months and years ahead.
- And last but not least a full, automatically logged, auditable record-of-fact for every test ever performed. Not just a vague and disputable list of pass/fail marks based on a manual inspection of some no longer available screen or log - but a complete record of every message field and interaction down to the finest level of detail that might be later required as evidence that diligent testing was performed.

In the end, it comes down to individual organizations to quantify the monetary value of these benefits. A large user of BASE24 and NonStop systems is on record as making savings of £1m year on year after working with Ascert and its partners to adopt the approaches described here as part of the user's application modernization project. The fact that they also significantly increased their test coverage in the process would have sent them singing all the way to the bank – except of course, they are a bank.

Ascert was founded in 1992 as a supplier of advanced testing software and services for the NonStop platform. Ascert's native and off-platform solutions allow a widerange of testing activities for the NonStop from functional through performance testing, managed directly or via HP Quality Center as part of an enterprise testing environment. Solutions built on Ascert's VersaTest technology are used for testing payments systems throughout the world,. Ascert is an HP Partner and member of HP Software's Enterprise Management Alliance Program (EMAP)