SELECTING PERFORMANCE TEST TOOLING – PART 3

This post became a bit longer than I initially intended so here are some links to jump directly to the specific content:

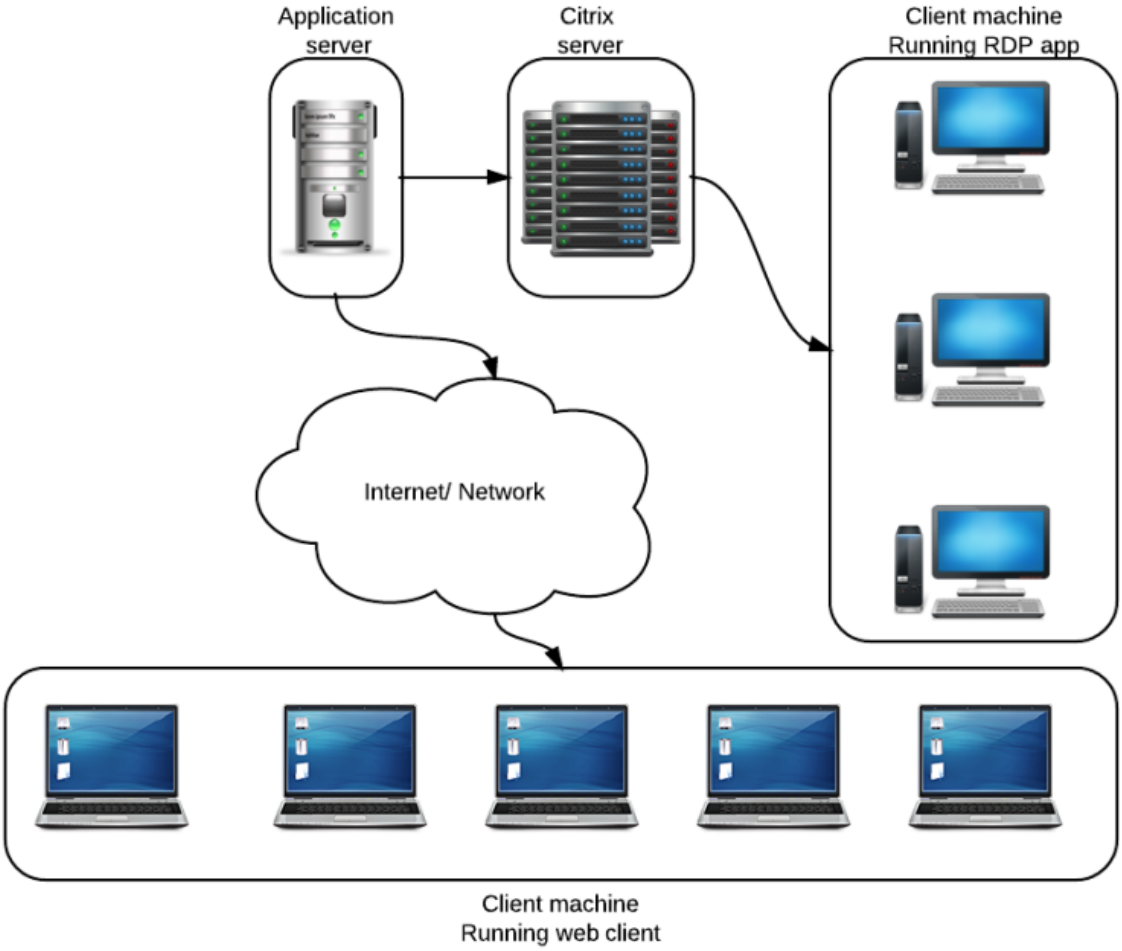
Some challenges in the PoC…

Following up on my previous posts (here and here) on this topic while executing the Proof of Concept I have run into some interesting challenges.

For starters, I am convinced I started the PoC the wrong way around; I started off with implementing some things in Sikuli rather than in SilkTest. Since SilkTest is, to me, less intuitive than Sikuli, since SilkTest tries to encapsulate a lot of different automation approaches in one tool, whereas Sikuli is focussed on one methodology only, I should have started with that one. However, I didn't and there is nothing I can do about that anymore now.

Secondly, the application we are about to put to the test is an application served from a Citrix platform, in other words, it is a remote application. The charter for this project is simple: Measure the performance of the application as the user would experience it. In other words, measure its performance via the RDP tunnel and not directly on the Citrix machine.

The setup is basically as shown in this image (simplified of course)



Simplified picture of the Citrix application setup

Sikuli

For those not farmiliar with Sikuli, here's what they say about themselves:

Sikuli Script automates anything you see on the screen. It uses image recognition to identify and control GUI components. It is useful when there is no easy access to a GUI's internal or source code.

In other words, Sikuli is fully based on image recognition and pattern recognition rather than following the industry standard Object Model.

The good, the bad and the ugly.

Stepping away from the Object Model has some advantages, especially in this application setup, but I will get to that when discussing the Borland setup.

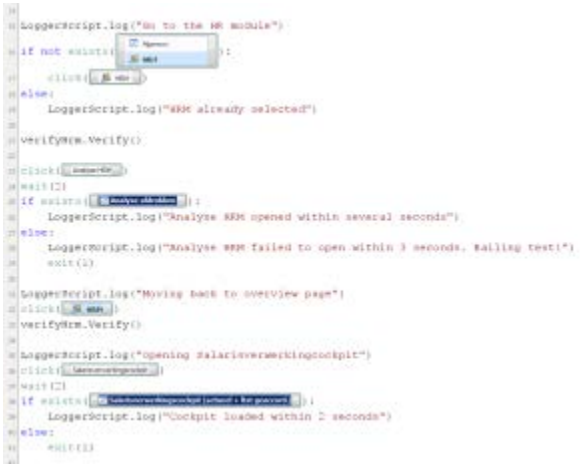
The good

Considering this is a Proof of Concept I have simply taken Sikuli out of the box, using Sikuli-IDE. The IDE works nice, simple and intuitive. It was very easy to start the RDP application and login without using any screenshots. The basic use of Sikuli is very simple and intuitive. Scripting in it is simple and logica, at least if you have a basic understanding of other scripting languages and/or programming.

Functionally stepping through the application was easy, just a few small screenshots were needed to load reports and verify that the report indeed is loaded successfully. In other words, the ease of use is excellent!

The bad and the ugly

I am mashing the bad and the ugly into one big pile since they are closely connected.



The first thing I disliked a lot is that Sikuli is 100% depending on Java 6, try running it on 7 and you have a problem (as in, it simply doesn't work).
Another bad part of Sikuli is that even if I wanted to, I cannot add Object ID's. This means that if I want to verify the existence of something, it needs to be done with screencaps and recognition thereof. Which leads me to the ugly. Screencaps are not the nicest way to identify objects, in fact they are ugly and not friendly to use,

since objects can occur, in a similar look and feel, several times on one screen. This results so now and again in the wrong button being clicked. It may look the same to Sikuli, but it is not the same functionally.

On top of that, I am now saving images in source-control (GIT) which I am not in favor of. Why would I want binary files in source control? I cannot do a diff on them anyway.

SilkTest

I have known Borland as a company for a long time yet in the past 10 years have not really worked with any of their tools. A short summary of how they see themselves:

With Silk Test, there's no need to understand coding so even non-technical people like your business analysts can build tests and get fully involved. This 13.5 release also breaks new ground by working with all the latest browsers, so a single script is all you need.

Well, there are some issues with that statement of course, cause a single script is always doomed to fail in the most horrid ways imaginable, but still, SilkTest is a nice tool to work with.

The good

The reason for looking at SilkTest was because I would like to have a tool now which is future proof for the organisatio. In other words, will this tool support further test automation on the end-to-end chains within this large organisation. One really important qualifier for that is solid SAP support. My Proof of concept on SilkTest started off looking into SAP support. The way Silk handles SAP I can simply summarize with one word: **good.** Out of the box it managed to select the correct SAP instance from the system selection popup, login without issues and after a few attempts execute a bunch of transactions. In other words, I was happily surprised! Most test automation applications I had on the longlist have serious issues in dealing with SAP.

The bad and the ugly

 The not so nice side of SilkTest in my opinion is that the recorded code is somewhat ugly, if not really ugly and not very friendly to read and through that probably also to maintain. This however is just a minor nuisance compared to the next issue.

Since the application under test is being served through an RDP tunnel I have no access to the object ID's. In other words, it is difficult to recognize objects on the application. In SilkTest it is not merely difficult, it is close to impossible. The only runnable way to do so I found is to record the tests based on the screen coordinates and then manually add assertions all over the place. However since SilkTest doesn't see what it is trying to test, getting the assertions in is really hard. What do you put the assertion on? There is no object to verify.

In other words, this is a disqualifier for SilkTest in this context.

Source : http://martijndevrieze.net/2013/05/02/selecting-performance-test-tooling-part-3/