SELECTING PERFORMANCE TEST TOOLING – PART 2

In my first installment I wrote about how I got the requirements together. Based on those I wrote a plan on what will need to be done. In my second installment I wrote about the first considerations of what I need the tooling to be able to do. In this part I am going to discuss a few of the things I have done to come to the shortlist and what will I do as a Proof of Concept for the tools.

Load generator

First of all, let's get the easy part out of the way. We will need something to generate a (functional) load on the servers. That part i consider relatively easy, no big tools are needed for this since we have an extremely powerful opensourcetool at our fingertips: Apache's JMeter. 

The load will have to be generated based on both HTTP traffic and client/server traffic, neither of which should pose a problem for JMeter. The most difficult part for load generation is getting the numbers out of the system, e.g. figuring out what the average and peak load is on the system. For this we have thrown some lines out to application managers to figure out.

Long list

The longlist I started out with was not just any list, it was a set of several lists. Out of this initial set I picked a bunch to actually play around with a bit more. Some gave me fun new insights, some disappointed me from the beginning, just by reading the sites or white papers.

The list of tools I initially looked at somewhat seriously was the following:

- Telerik TestStudio

- Microsoft Visual Studio Test Professional
- IBM Rational Functional Tester
- SmartBear TestComplete
- Borland SilkTest
- Original Software TestDrive
- Sikuli
- Thoughtworks Twist
- RobotFramework
- Apache JMeter
- AutoIt
- Froglogic Squish
- Neotys NeoLoad

Shortlist

Quite a few of the tools I installed, just to see how they work and integrate with developer tools like Eclipse and Microsoft Visual Studio Express. The majority of the more expensive tools barely integrate at all, since I would need to have a full version of Visual Studio rather than the Express version. That is a full disqualifier for me in this phase.

Another strong disqualifier is if the tool simply refuses to run on a Windows XP Professional environment, such as Microsoft Visual Studio Test Professional. Within this company the majority of machines are still running Windows XP or XP Pro, so the tools need to work perfectly in that environment. Interestingly the only tool that flat-out refuses to be installed on it is a Microsoft own tool :).

After having considered the needs for the tool in the short term and possibly longer run two tools jumped out big time: *Borland SilkTest and Sikuli*.

What have been (some of) the disqualifiers for the other tools I looked at:

- availability of a downloadable and fully usable demo version, some tools have no demo version available or the demo is locked off.
- support of SAP for possible future use, the organisation is looking at a long road ahead of SAP upgrades and patches, so automated test support would be a welcome helping hand
- possibility to use the tool for more than just performance or load testing, for example for pure functional test automation
- organizational fit moving forward, e.g.
    - will the less technical people within the organisation be capable of using this tool for future runs of the tests built for this particular project?
    - will this tool be capable of supporting upcoming projects in both functional and non-functional tests?
    - is the learning curve for the internal users not too steep (or, how much programming is actually needed)
- price, is the price something that fits within the project budget and does the price make sense in relation to the project and capabilities of the tool

PoC

With this very short list of two tools a Proof of Concept will be made to see how the applications deal with several situations I will be running into during the performance tests.

One of the main parts to test is whether or not the tool is accurate enough in measuring and reading the state of the application under test. Since the application under test is two fold: a web-application and a remote desktop application.

The webapplication, as stated in the previous post, will not really be the difficult one to test. The remote desktop application however is more challenging to test. The application runs on a Citrix server and thus the object ID’s are not visible to the test automation tooling. The second outcome of the PoC should be to see how

well the tooling deals with the lack of object ID’s and thus with navigating the application based on other pointers. For Sikuli the challenge will be different resolutions, for SilkTest I will be focusing on finding a way other than navigating by screen coordinates.

Source : http://martijndevrieze.net/2013/04/29/selecting-performance-test-tooling-part-2/