AUTOMATING A LEGACY SYSTEM

In the past I have worked a lot with fairly new, web-based applications. So here and there I would run into some “old” desktop applications, but the majority of my professional career has been, working with web-based technologies.

In my current assignment however I have hit a nice new technology, at least new to me: COBOL over TN3270

When I started on this assignment I had one big fear: being forced to use one of the bloated commercial tools out there which claim they support TN3270 off the shelf. Oh, did I say bloated? That sounds like I am biased towards the so called big commercial tools.

This sense of bias is partially true, I am in favor of using small drivers like WebDriver or White where possible over using a full environment such as QTP or Rational Functional Tester. The big tools quite probably have their own unique selling points, however for a small IT organisation they are a huge investment and quite often overkill compared to what is actually needed.

When we started looking into tools we almost instantly dismissed the “big ones”, not so much due to their cost, but mainly due to the amount of customization still needed to make them work “out of the box” with TN3270 and the rest of the environment we’re automating. When talking to the vendors they all claimed that their tool is excellent to use for an environment for this, when asked however whether the tool will be pluggable out of the box the answer was consistently no, you will need to do some customization (or better yet, one of their consultants would need to do some customization).

This same amount of customization will need to be done with a small driver as well, so why spend a fortune on something that is no better than a small, cheaper or even free tool?

For the Proof of Concept we decided to take two different drivers: Jagacy and s3270.

Jagacy is in their own words:

"… our award winning 3270 screen-scraping library written entirely in Java. It supports SSL, TN3270E, and over thirty languages. It also includes a 3270 emulator designed to help create screen-scraping applications. Developers can also develop their own custom terminal emulators (with automated logon and logoff). Jagacy 3270 screen-scraping is faster, easier to use, and more intuitive than HLLAPI. It excels in creating applications reliably and quickly."

I cannot but agree with especially the latter part: it is indeed faster and easier than HLLAPI. In terms of more intuitive, I am not quite convinced (yet).

s3270 is a lot more simplistic than Jagacy:

" opens a telnet connection to an IBM host, then allows a script to control the host login session. It is derived from *x3270*(1), an X-windows IBM 3270 emulator. It implements RFCs 2355 (TN3270E), 1576 (TN3270) and 1646 (LU name selection), and supports IND$FILE file transfer."

In other words, s3270 can be used as a layer between an application and a legacy TN3270 system.

Talking directly to s3270 is not the most intuitive thing, this requires the code to start a process, keep track of the state of this process and throw input into this process stream. All nice and doable,

however it was already invented before by others and they quite likely did an adequate job at making this work, considering TN3270 is a fairly old protocol.

In order to make life for us easier while using s3270 or the PoC, we searched around a bit and found a wonderful open source tool h3270. Put simply, h3270 is a program that allows you to communicate with tn3270 hosts from within your browser. This open source tool helped us quickly hook up our test code to s3270. We dit not however need this browser functionality but are thankfully using the object orientated abstractions within h3270 for our own purposes

One of the wonders of working with tn3270 is that a whole new world of states opens up. In a web-browser for example, something is either there and visible or it is hidden from the user. Keyboards are never locked, when opening a new page the browser will tell your driver when it is done loading. With tn3270 it is not all that simple. First of all, there are features like a locked keyboard, it works with function keys that have disappeared from our physical keyboards years ago, the state of a screen is unknown most of the time, etc.

On the other hand, a terminal emulator is still nothing other than a form of sorts, just a bit more difficult to read and write.

Basically one of the things we once more saw proven, is that in test automation the basics are always similar. It is just the way you need to talk to the system that may be different.

Source : http://martijndevrieze.net/2012/03/16/automating-a-legacy-system/