Test automation ready for innovation

By Allen Fisher

Pure black box testing is by definition immune to technology change. But can black box test execution be automated?

Allen Fisher explains how image recognition can solve long-standing problems of test automation and enable its application to new and future platforms

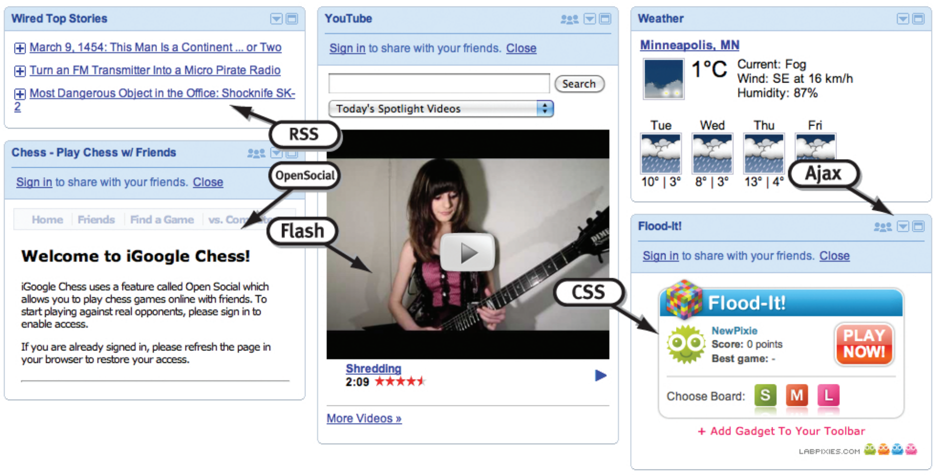
The best-known test execution tools run on Windows and are for testing Windows applications, or Web 1.0 sites optimized for Internet Explorer. But what about:

- other OSs such as Mac OSX and various Linux distributions which are gaining market share?
- smartphones, netbooks and tablets?
- automatic online updating of third-party software causing change?
- cloud applications, and users who want to use them with their choice of browser, a choice that's growing fast?
- Web 2.0 sites including Flash, Flex, Silverlight, Java, Rails, PHP, .NET, etc applications, often alongside one another on the same page? (see figure 1)
- highly configurable and skinnable user interfaces?
- closed systems which will not run alongside other applications or would then behave differently due to time dependencies? (see figure 2)

The solution to these challenges is a truly black box approach, which most test execution tools cannot achieve. To work they have to (i) identify user interface objects, or (ii) record events in terms of screen coordinates. In case (i) the tool is highly implementation-specific: testing a version of the same product for a different platform requires either a different tool which cannot use the tests already created, or reversion to manual execution. In case (ii), tests become invalid when the UI changes, whether as a result of development of the system under test or external factors: they fail to complete and/or return false incidents. Finally, traditional test tools run on the same computer as the SUT, requiring the OS to switch control continually between the two, so are often unstable and/or intrusive.

Achieving black-box automation: run the tool on another computer.

The test automation tool eggPlant runs on Mac OSX and Linux. But it can test applications on a vast range of other platforms. That's because it connects to and controls them using VNC (Virtual Network Computing). VNC is a remote control protocol similar to Apple/Windows



Wired Top Stories

March 9, 1454: This Man Is a Continent ... or Two

Turn an FM Transmitter Into a Micro Pirate Radio

Most Dangerous Object in the Office: Shocknife SK-2

Chess - Play Chess w/ Friends

Sign in to share with your friends. Close

Home | Friends | Find a Game | vs. Com

Welcome to iGoogle Chess!

iGoogle Chess uses a feature called Open Social which allows you to play chess games online with friends. To start playing against real opponents, please sign in to enable access.

If you are already signed in, please refresh the page in your browser to restore your access.

YouTube

Sign in to share with your friends. Close

Search

Today's Spotlight Videos

Shredding
2:09 ★★★★★

More Videos »

Weather

Minneapolis, MN

1°C Current: Fog
Wind: SE at 16 km/h
Humidity: 87%

Tue	Wed	Thu	Fri
10° | 3°	8° | 3°	13° | 4°	

Flood-It!

Sign in to share with your friends. Close

Flood-It!
NewPixie
Score: 0 points
Best game: -
PLAY NOW!
Choose Board: S M L
+ Add Gadget To Your Toolbar
LABPIXIES.COM

Figure 1: iGoogle page using multiple technologies

Remote Desktop, used for example by IT departments to manage server configuration or support users by temporarily “taking over” their workstation. eggPlant takes the role of a human operator, so can “see” the display, control the pointing device, and send keystrokes on the SUT device. VNC servers are available for virtually all important platforms including Symbian, Windows Mobile, Android and iPhone. An iPad version is sure to emerge soon.

Achieving test robustness: image recognition

So, eggPlant works by “seeing” the screen, and does not care what any of the items on it are called or what is their underlying technology. In that way, it's like a screen coordinate based tool. But it doesn't share their disadvantages because it records the UI objects as bitmap images of sectors of the screen rather than simply geometric points.

When scripting a test that includes, for example, a click on a button, the tester captures an image of that button (see figure 3). The image is given an arbitrary name which is used to refer to it in eggPlant's scripting language, SenseTalk (figure 4). It is necessary to capture it only once; subsequent interactions with the object are scripted by selecting it by name. As well as text, images can form part of the expected outcome for the test, ie the screen output: they are captured and named in the same way.

When the script is executed, eggPlant finds the image wherever it is on screen using image recognition algorithms and performs the scripted action. So the test runs even if the UI objects and outputs are in different locations on the screen, for any reason. If an image cannot be found, SenseTalk's `ImageFound()` function returns FALSE, so the script can raise an incident.

Other than if navigational structure of the SUT changes, the only time an eggPlant script needs maintenance is if the appearance of an object changes. It's achieved by executing the script with the *Image Doctor* feature. When eggPlant cannot find a required object, it pauses. The tester then selects the object as it now looks. Execution continues, pausing again if another object cannot be found so that the operation can be repeated. When the script finishes, the test has been repaired; it is now correct for the current build of the SUT.

Figure 2: air traffic control: a closed, real-time system

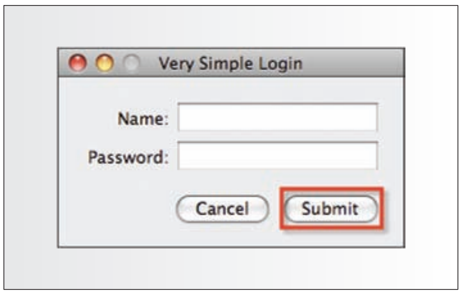
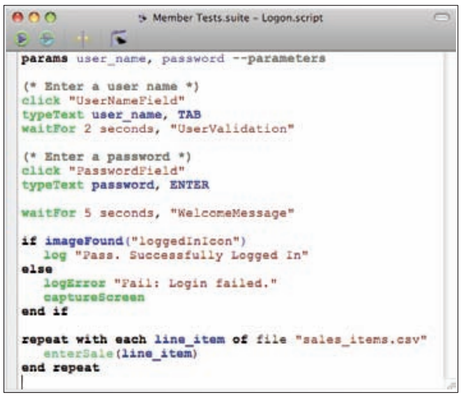


Figure 3: capturing a UI object as an image



```
params user_name, password --parameters

(* Enter a user name *)
click "UserNameField"
typeText user_name, TAB
waitFor 2 seconds, "UserValidation"

(* Enter a password *)
click "PasswordField"
typeText password, ENTER

waitFor 5 seconds, "WelcomeMessage"

if imageFound("loggedInIcon")
   log "Pass. Successfully Logged In"
else
   logError "Fail: Login failed."
   captureScreen
end if

repeat with each line_item of file "sales_items.csv"
   enterSale(line_item)
end repeat
```

Figure 4: SenseTalk

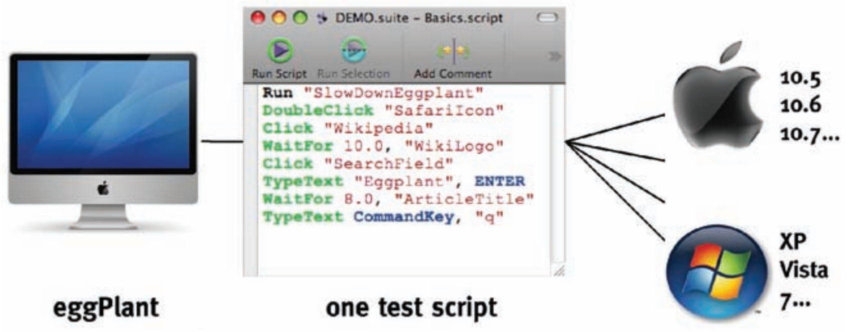
Achieving test reuse: image collections

When an image is captured using Image Doctor, it does not by default replace any previous image of that object, but is added to a “collection” of valid images for that object. During subsequent execution of the script, eggPlant searches for all of them.

This means the script can be enabled to test any other version of the SUT, including those running on other platforms, by exactly the same procedure: eggPlant is connected to the new test item and the script is executed. Images of how the objects now look are added to their respective collections (images in embedded objects such as Flash and Java applets will often look similar enough to be found without user intervention).

Abstracting the UI from the test procedure in this way means the same script can be used for multiple builds, versions (including language translations) and ports to new platforms (figure 5), multiplying the return on investment in automating the tests ■

Allen J Fisher is a Senior Systems Engineer with TestPlant, the producer of eggPlant.



```
Run "SlowDownEggplant"
DoubleClick "SafariIcon"
Click "Wikipedia"
WaitFor 10.0, "WikiLogo"
Click "SearchField"
TypeText "Eggplant", ENTER
WaitFor 8.0, "ArticleTitle"
TypeText CommandKey, "q"
```

Figure 5: script reuse for multiple platforms and versions