PIXELFEDERATION®
players to players

Automation of Game Testing with SikuliX and AirtestProject/Poco

Peter Miťko

October 10th, 2019
SANAE Software Testing BEER.EX

Speaker introduction

Peter „Mitec“ Miťko

- 10 years in Pixel Federation
- Client mobile programmer (3bot)
- Server programmer (Trainstation)
- Client & server programmer (Galactic Junk League)
- QA automation developer

Two Approaches

Image Recognition (SikuliX)

- SUT is a Black-Box
- position of elements is detected through image recognition
- each element is identified by file path to its screenshot
- requires screen
- value (text, number, ...) must be retrieved through OCR

Selectors (AirtestProject/Poco)

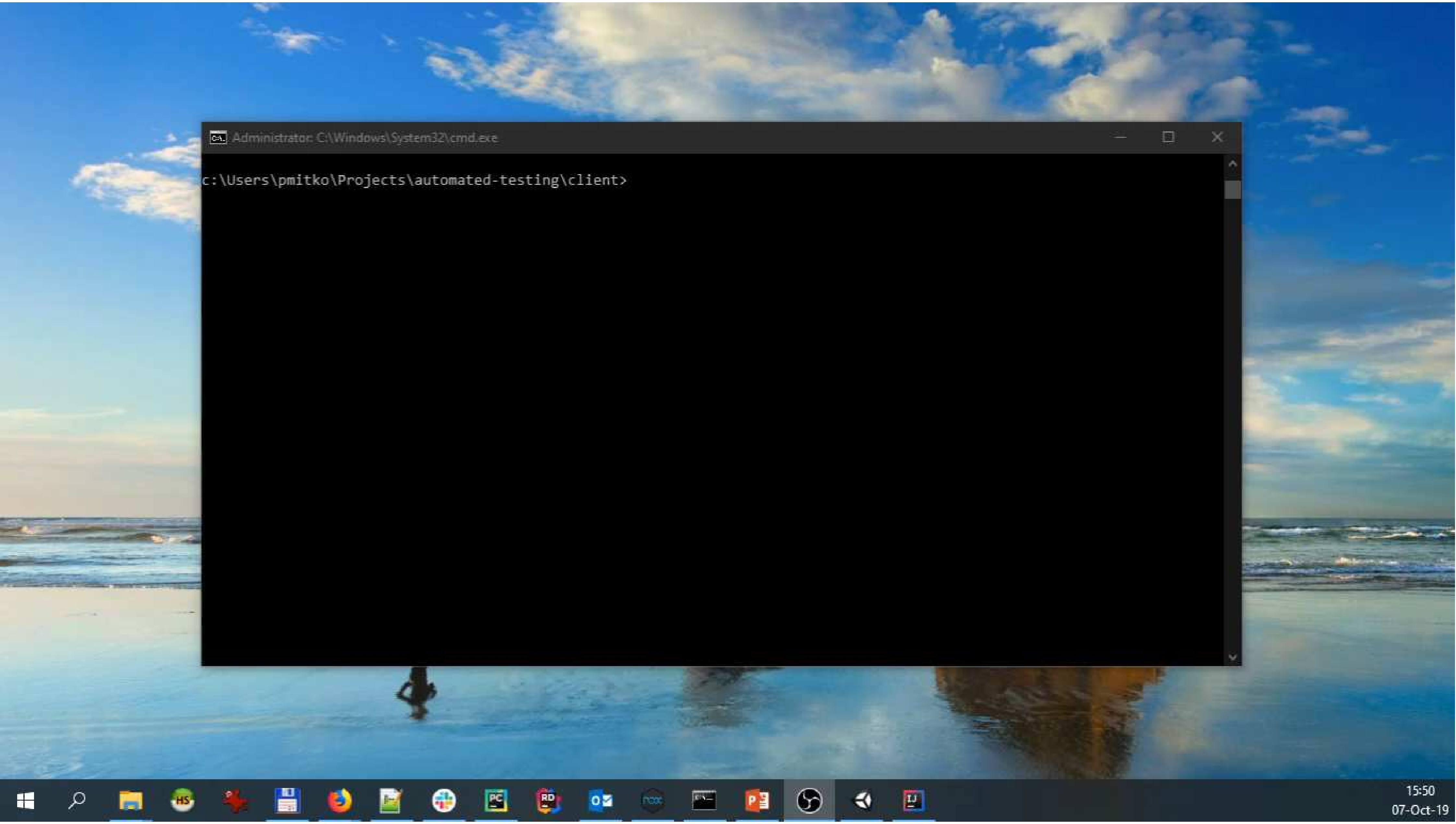
- SUT must be a White-Box
- SUT must contain test command handler
- position of elements is detected through selectors (by css like style)
- each element is identified by its path in hierarchy

SikuliX

- Automates anything you see on the screen of your desktop computer running Windows, Mac or some Linux/Unix. It uses image recognition, powered by OpenCV, to identify and control GUI components. This is handy in cases when there is no easy access to a GUI's internals or the source code of the application or web page you want to act on.
- Custom IDE
- It has Java API and thus can be used in any Java aware programming/scripting language (Jython, JRuby, Scala, Clojure, ...)
- Supports several scripting languages:
  - *Python language level 2.7 (supported by Jython)*
  - *Ruby language level 1.9 and 2.0 (supported by JRuby)*
  - *JavaScript (supported by the Java Scripting Engine)*
- Comes with image text recognition (OCR) powered by Tess4J / Tesseract.
- Available at http://sikulix.com

SikuliX - test cases

- For web and mobile
- Only the most important test scenarios (happy path):
    - Registration
    - Login
    - Payment
    - Tutorial



```
c:\Users\pmitko\Projects\automated-testing\client>
```

SikuliX – takeaways

- SikuliX IDE is fine, but lacks stuff like type hinting, ... Java API + IntelliJ IDEA was for us much more productive choice (perhaps python + java API + PyCharm would be killer option).
- Documentation is OK, but a bit confusing at times.
- Detection of elements through screenshots was quite fast (250ms on average).
- Even screenshots from animated part of the screen worked well.
- SikuliX 1.1.4 has support also for image transparency - screenshots can be more fine-grained.
- Setting min. similarity correctly is crucial. If set too low, differences between screen and screenshot can lead to false positives, if set too high, tests will may become fragile.
- Debug mode (the red rectangle) is really helpful.
- Screenshots are more fragile then ids.
- If you need text/number values, OCR is crucial for you as well. We used Googles Vision, because it had better results then build-in Tesseract. Standard fonts worked very well, but not custom fonts. However SikuliX 1.1.4 comes with upgraded Tesseract integration and by using it with custom trained data, it could be the best option.

SikuliX – takeaways

- Developing/running end-to-end tests requires screen and system input (keyboard and mouse).
- SUT can be treated as Black-Box (no need to handle test command handler issues, product branches, …) and there is no need to handle multiple APIs (game/app, android/iOS payments, ads, …) - just the screen.
- Good VNC is crucial for testing on mobile. The most important capabilities are: frame rate, screen detail and stability.
- Testing in the could is perfectly possible, but the provider must support manual testing with good VNC capabilities.
- Scaled UI because of different screen resolutions is still problematic. Decreased min. similarity can handle it, but can also cause false positives.

AirtestProject/Poco

- Supports mainstream game engines (Unity3D, cocos2dx-js, cocos2dx-lua, Android/iOS native apps).
- Retrieves UI Elements Hierarchy in game's runtime.
- Has build-in image recognition (similar to the one used in SikuliX).
- Works on desktop and mobile (android, iOS).
- Is compatible with Python 2.7 and Python 3.3-3.6.
- Available at https://github.com/AirtestProject/Poco
- Airtest Project is its ”big brother” (https://github.com/AirtestProject/Airtest)

AirtestProject/Poco - test cases

- For mobile
- Only the most important test scenarios (happy path):
  - Main Screen
  - Resource buying (payments)
  - Tutorial
  - Other basic features

AirtestProject/Poco – takeaways

- Out of the box solution for Unity3D editor and also for game/app on android/iOS.
- Documentation is good and mostly in English, although there are still parts only in Chinese.
- Defining elements by their path is faster then making screenshots.
- Speed of test execution was satisfying only at the beginning. This may be caused by architecture of the game.
- Implementation of test command handler in the game/app can cause conflicts of libraries (e.g.: json). It also requires custom testing build of the game/app.
- Multiple APIs (app - Unity3D, payment window - android/iOS, ...) can be handled with the same code.
- You can use Python's unit test or pytest, but using PocoTestCase you'll gain access to enhanced test logs.
- These logs are viewable through NetEaseReplay as screenshots, video recording, logs and test code on timeline.
- Supporting tool PocoHierarchyViewer wasn't sufficient compared to Unity3D editor. However it was good enough for small android stuff.
- Because of support for python 2.7, type hints are not available (they'd be really helpful).
- Poco supports also image recognition, but it is slower compared to SikuliX.

Final Thoughts

- Both SikuliX and AirtestProject/Poco do basically the same thing, but with a different approach.
- They are not necessarily competitors – they can be combined.
- Image recognition has a big advantage in that SUT can be a Black-Box.
- Using selectors requires that SUT is a White-Box.
- Using selectors is easier and less fragile then using screenshots.
- Implementing test command handler into SUT requires custom testing build, merging production version into the testing build and may require handling of library conflicts.
- Image recognition often also requires OCR.

Thank you!

E-mail: pmitko@pixelfederation.com

LinkedIn: https://www.linkedin.com/in/peter-mitko

Q&A

Sources

https://www.deviantart.com/supuhstar/art/Browser-Icon-Mashup-474495059
https://sikulix-2014.readthedocs.io/en/latest/tutorials/masking/shot.png
https://thedistance.co.uk/ios/ios-android-app-audit/
https://portal.pixelfederation.com/en/
https://portal.pixelfederation.com/en/afkcats/about/
https://github.com/AirtestProject/Poco