Chasing the giant rabbit:
A tale of software, automation, and outsize bunnies

Jonny Fletcher
ROQ

@jonnyfletcher84
Jonny Fletcher

✓ Test Automation Architect
✓ Started working at ROQ in October 2013
  ✓ Award-winning pure-play software testing services provider
  ✓ Founded in August 2009
  ✓ Operating from a purpose built Test Lab based in Chorley, Lancashire
✓ Responsible for the design, development, implementation and governance of automated test solutions
✓ Active speaker at software testing events throughout the UK
Chasing the Giant Rabbit:
A tale of software, automation, and outsize bunnies
What was the project?
Working with one of the UK's largest pet supplies retailers to deliver a test automation framework capable of functionally testing an EPOS system used daily in over 300 stores.

The software used by the retailer was developed by a third-party vendor as an off-the-shelf solution – with minor customisations added to meet the retailer specific requirements.

The test automation framework had to be capable of replicating a software testers’ interaction with a POS terminal – testing a .net desktop application hosted on a Windows operating system.
Why test automation?
Their decision to implement an automated software testing solution was primarily driven by the desire to reduce the time taken to release changes into production – while maintaining, or improving, the thoroughness of software testing conducted as part of new feature acceptance and regression testing.

For each new software release, a full regression test cycle would be carried out by 2 software testers and scheduled over a period of 2-weeks – this was a time-boxed exercise often forcing a risk-based approach to testing.

In order to respond to market changes quickly, the time taken to fully validate that new changes hadn’t adversely impacted existing code had to be reduced significantly.
What did we do?
We embraced Agile

The business was already delivering projects using the Agile software development methodology – legacy test cases hadn’t been updated to reflect this change in how requirements were captured, and tested

✓ Re-wrote the existing step-level test cases in the form of user stories and acceptance criteria
✓ Implemented a BDD-style test automation framework
✓ Used only open source tools
✓ Integrated the solution with Jenkins CI for test execution and reporting from a central dashboard
Given – When – Then

You & Your condition

What you see

What you do
Feature: Loose item basic sale

As a store customer,
I want to buy rabbit food,
So that I can feed my giant rabbit

Scenario: Purchase a loose item using exact cash

Given I have "£5.00" of "rabbit food" in my basket
And I proceed to the checkout
When I pay using "£5.00" of "cash"
Then the transaction is successful
Physical devices connected to the test machines posed the biggest challenge:

✓ Interacting with peripheral devices, including the card reader, could not be automated – forcing cash only transactions
✓ The till drawer opened after each cash transaction was processed
✓ The receipt printer would run out of paper

Constantly changing data relating to products, prices and promotions caused instability in testing – using hard-coded test data would be impossible
The successes

There were several key highlights:

✓ Reduced testing time from 2-weeks, 2 testers down to 7 hours overnight – significantly improving the flow

✓ Now executing all test cases during each regression cycle – no longer following a risk-based approach to testing – amplifying the feedback loop

✓ Implemented a data-driven approach, dynamically updating data tables contained within feature files

✓ Improved the quality of the software delivered by the third-party vendor through highlighting the lack of software testing soon after deployment to a test environment – shortening feedback loops

✓ The same approach to test automation has now been embraced by the whole business
The surprises

There were several unexpected findings along the way:

✓ Through generating test cases dynamically, this highlighted legacy data that had no purpose but was replicated to each POS terminal nightly
✓ Identified errors in promotions, e.g. those that could never be triggered or, more importantly, those that would never be honoured
✓ The automated test cases could be executed to generate large volumes of transactions that could be used to support basic performance tests
✓ We found that we could test not only the cashier facing display but the customer facing display too
✓ The automated test cases were not only saving time – they were responsible for protecting the clients’ reputation
Automated software testing is amazing...

BUT...

...software testers aren’t dead, yet
They’re empowered to explore
Any questions?
Thank You!

Jonny Fletcher
ROQ

@jonnyfletcher84

#EuroSTARConf