Unscripted Automation

By Paul Carvalho

http://www.STAQS.com/

Kitchener-Waterloo Software Quality Association

March 2009

What do *you* Automate?

- Tests? (if so, what kind?)

- Data setup?
- Performance measurements?
- Report generation?
- ...?

Interesting Set:
These Scripts usually created for a specific purpose

"Test automation is any use of tools to support testing" – James Bach

Can Computers Test?

- I don't think so.
  - Computers can't think
  - Computers don't understand test techniques
  - Computers can't observe beyond what they're programmed to observe
  - Computers can't interpret results

- Computers are good at repeating the exact same steps every single time (humans generally aren't)

➢ *Is there value in this?*

Computer-Assisted Testing

- Computers can *help <u>you</u>* test
- Questions to ask yourself:
  - What kinds of tests are worthwhile automating?
  - Can you compensate for the Minefield problem?
  - Can you write low-maintenance tests?

- How about if you let the computer drive while you observe and evaluate?
  (read Jonathan Kohl's article – available online)

Experience Report (ER)

- I can't tell you how to make test automation work in each individual situation, so I'll talk about how it's worked for me.
  - There might be something here that you may learn from or find useful on your own projects

- So, here's a tale of how I use test automation in a way that works for me .. right now .. on a specific application .. in my current team ...

ER – Project Background

- Application/System Under Test:
  - Web app: .Net, JavaScript, ASP, BizTalk, SQL db
  - Browsers: IE (mostly), Firefox, Safari and others
  - O/S's: Windows (mostly), Mac, Linux

- Industry = Financial
  - App does calculations
  - It takes inputs and produces reports & charts
  - Lots of fiddly bits (other functionality) too

ER – Test Automation

- Scripting Environment:
  - Ruby
    - WATIR (Web Application Testing in Ruby)
    - other libraries as required

- Case Study: Automation script most frequently used:
  - Smoke Test *(quick Functional test at the System level)*
    - Performs a quick walkthrough of the app to check that the main features/components are working
    - Finds lots of bugs
    - Low maintenance
    - Execution is different every time!

ER – How does the Script work?

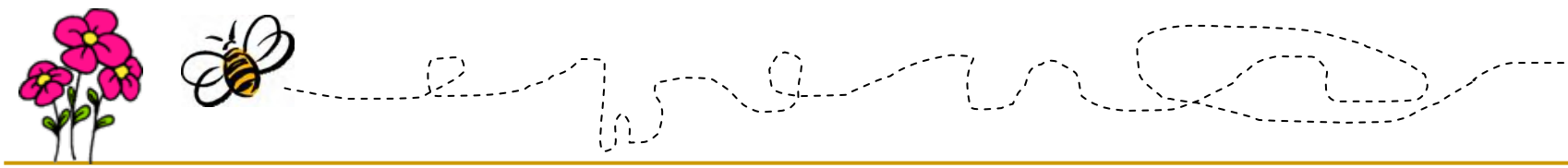
- Script decides which environment it wants to use
  - It looks up the valid user ID and password to log in
  - Starts up a web browser and testing begins

- The script has a basic mission to ‘walk’ through the application
  - It decides what data it uses as it goes along. It makes stuff up too. It looks totally random.
  - It logs any critical bugs found to output files. Other unexpected bugs are output to screen.
  - Additional bugs found if you watch it execute.

ER – Anatomy of the Script

- The automated script is modelled using these key principles:
  1. *Data Driven* – i.e. don't hard-code any specific values to use/input
  2. *Regular Expressions* – find things using the simplest possible expressions
  3. *Equivalence Classes* – all things being equal, let the computer decide what values in a range to use
  4. *Random* – Don't execute the same tests twice
  5. *Don't Repeat Yourself* (DRY) – avoid code duplication

Test Technique Alert !!



ASIDE: *Is this MBT?*

- No, this is not Model-Based Testing
- It's a good step in that direction though

- Reading Suggestion:
  - **Harry Robinson**'s Model-Based Testing articles & presentations

Honk if you love to crash software

ER – Script – Finds Bugs How?

- Key factors in script development:
  - Data-driven
    - Have the code make decisions based on the information displayed on screen
    - More closely resembles how a human would test
  - Teach the code to recognise Equivalence Classes
    - What are the valid & invalid inputs for a given field?
    - Teach the computer to figure it out and select values *randomly*
    - Similar to *Soap Opera Testing* – use interesting values
  - Random execution
    - Every time it runs, it follows slightly different paths *(considers the Minefield problem)*

ASIDE: Data Driven – Example

- Consider a Drop-Down List => contents are crayon colours.
- Before: (Scripted approach)
  - `ddl_array = [‘red’, ‘green’, ‘blue’, ‘yellow’]`
  - `crayon_picked = ddl_array[ 2 ]`
    - *What if the array changes?! (and why hard-code the selection?)*
- After: (Data-driven)
  - `ddl_array = drop_down_list.getAllContents`
    - i.e. never hard-code values!
    - Let the computer figure out what it has to work with
  - `crayon_picked = ddl_array[ rand( ddl_array.length ) ]`
    - Pick a random item in the list

ASIDE: Equivalence Classes – Example

- This Test Technique is important to understand
- When you look at an element, how do you decide which item to select in a list? What value to input in a field? ...
- e.g. Name input field: (Scripted approach)
  - `user_name_field.set( “James” )`
    - *What value is there in a test that repeats this input *every* time?*
- Name input field: (E.C.-aware)
  - `upper_limit = user_name_field.maxlength`
  - `user_name_field.set(random_input('char',rand(upper_limit)))`
    - ‘random_input’ method generates random chars of a set length
    - Put whatever you want into each field – change it up!
    - If the ‘maxlength’ property isn’t set, what will happen?

ER – Script – Low Maintenance?

- Key factors in script development:
  - Data-driven
    - You don't have to update the code every time the application content changes
  - Regular expressions
    - Makes your code less sensitive to changes in object names/labels
    - *These are important! Learn these!*
  - Avoid Code Duplication (DRY)
    - Reduces time spent updating code
    - *Develop good programming habits*

ER – Summary – *Unscripting* Tests

- We've *Unscripted* our test automation by:
  - Removing exact test steps from the code
  - Removing exact test inputs to use from the code
  - Programming path guidelines (decision models) through the app
  - Teaching the script to identify the data it needs as it goes along
  - Teaching the script to input data "randomly" based on field types and limits
  - Programming simple Oracles to help identify bugs *(knowing that the computer will miss many more)*

Summary – *How about you?*

- These principles can be applied to whatever tool/scripting language you currently use
- A good tester or programmer should be able to write good automated test scripts
- A good tester *paired with* a good programmer should be able to write ***great*** automated test scripts!

"Automating garbage produces fast trash" – (author unknown)

Recommended References

- Articles: *(there are many, here are a few I like)*
  - "Man and Machine: Combining the Power of the Human Mind with Automation Tools"
    - Jonathan Kohl, Better Software, December 2007
  - "Old School Meets New Wave"
    - Chris McMahon, Better Software, June 2006
- People: *(read their papers & presentations on Automation)*
  - Doug Hoffman - http://www.SoftwareQualityMethods.com/
  - Bret Pettichord - http://www.Pettichord.com/
  - Harry Robinson - http://www.geocities.com/harry_robinson_testing/

Demo

- Ruby & Watir in action
  - Automated scripts demonstrate the principles in action
- Test app = simple Bookstore web app