YES! You CAN bring test automation to your company!

Alan Ark
QA Manager
Compli
arkie@compli.com
http://www.linkedin.com/in/arkie

Abstract

No money? No support from your supervisor? No experience in test automation? No problem.

Even with these barriers to entry, you too can reap the rewards of automating test activities in your web application. Sometimes a formal framework is overkill. Shooting for the moon rarely works when there are so many unknowns. With a less than optimal implementation, bad test automation could also leave your project at risk.

This presentation will provide you with an idea framework to help your test efforts along. The key is to start small and learn as you go along. By utilizing Ruby (a free scripting language) and the Watir (Web Application Testing in Ruby) library among other tools, the hit to your budget is zero dollars. By building on cumulative successes, you can present the business case for supporting automated test efforts in your organization.

As someone who has brought in test automation to several organizations, I will share with you my keys to success, so that you may also reap the many rewards of automated testing on your projects.

Biography

Alan Ark is the QA Manager at Compli, in Portland, Oregon. Alan has gained tremendous experience working for Unircu, Switchboard.com, and Thomson Financial - First Call. Mr. Ark has previously presented 'Euro: An Automated Solution to Currency Conversion' at Quality Week '99, and 'Collaborative Quality: One Company's Recipe for Software Success' at PNSQC 2008. At Compli, he is using Ruby to solve problems both large and small. His LinkedIn profile can be viewed at http://www.linkedin.com/in/arkie.

Introduction

In the course of our day to day lives, we tend to work on the same areas again and again. In some cases, to the point of taking the exact same steps on newer iterations of the project. Isn't there a way that we can make the process a little less painful? Usually there are ways to accomplish this very idea.

The idea of test automation can be expanded to include ideas that are not test cases in themselves, but that need to be in place for running of the tests. Some examples are setup of test environments, clearing out log files, and creation of data. Creation of automated test cases is also key to helping you become more efficient in your job. The use of Watir and other Ruby libraries can help with the regression testing of various web applications. In fact, using a few other libraries on top of Ruby, you can even interact directly with databases!

We'll examine some of these concepts using my experiences with the goal being to help your company realize similar gains in efficiency as well. It is my hope that this session will be more give and take rather than a single voice speaking to a crowd. While I have worked in companies that utilize Microsoft tools, the ideas can be generalized to most other development toolsets.

The main idea of this paper is to present test automation in a different light. By showing how automation can be used not only the execution of test cases, you should be able to gain more leeway from the stakeholders in your company for more test automation activities in the future.

The section titles come from song titles by Jimmy Buffett. He happens to have a song for every occasion.

1. "If the Phone Doesn't Ring, Its Me"

The usual reasons I hear that test automation is not used in various companies.

Price
- The company does not have the budget.
- We can't afford the training.

Support from management
- The company doesn't want to buy the tools.
- My manager doesn't want to spend the time automating tests.

Experience
- We tried it once, but it failed miserably.
- We don't have people who know how to program.
- I can't do it.
- It's too hard.

2. "Bend a Little"

Pouting is not going to get you anywhere. Realize that when you give a little that maybe you'll get a little in return. You may not get all the resources that you are asking for, but by showing some success, that opportunity is closer than it was before.

Think about what you really want, and go explore some options. At first, you may have to do some of this analysis "off the books" until you have some solid gains to show. Take the time to really show how these methods can shine in your environment.

Fit the exploration into your day to day activities.
- Be flexible.
- You might have to do some of this research outside of office hours.

Recognize when things are not going as well as you had hoped.
- Prevent this activity from becoming a rabbit hole.
- If you are spending too much time on the research, maybe its not the right solution

Plan on getting interrupted
- Realize that this is not job number one for you.
- Document progress so that you can continue with the project as time permits.

3. "What If The Hokey Pokey Is All It Really Is About?"

Is it an elaborate tribal custom or something simpler? Sometimes, simpler really is better. Especially when starting with something new. Realize that all projects are unique in their own special way, but to recognize and leverage patterns when they appear.

What's the plan? People like to hear what has worked for others. This provides validation that someone else has achieved some level of success with their work. The real question that needs to be answered is "What are MY pain points?" By learning what YOUR pain points are, only then can a solution that works for your situation be crafted.

This is a software development project
- Multiple people can be working on separate issues
- Measure how much time the tasks are taking
- Get a good handle on how much time people are taking on the tasks - including yourself!
- Have a goal – What do you want to accomplish
- Have a high level plan – What steps are needed

Break out the pain points into smaller problems to solve
- Divide and conquer
- See if the problem breakdown is a well defined set of problems to solve
- If it is, it's a candidate for automation
- If it isn't, then you might have to rethink your plan of action
  - Reframe the question
  - Break the project up in a different way
  - Know when you are in over your head

See if automation can help move the testing process forward in different areas
- There are certainly cases where automation should not or cannot be used
  - OS and software installs.

Don't make things more complicated than they need to be
- Simpler really is better when you're first starting out

Example of something that is not truly "automation" but can help
- Configuration of registry settings
  - Importing them from a file works great
  - Big improvement over entering them manually
  - Able to rapidly configure multiple machines with the same configuration now

Types of things I do.

Seeding test data
- Use Ruby and the FasterCSV gem to generate comma separated value (CSV) files

Drive browsers
- Use Ruby and the Watir gem to interact with web sites

Push out web builds.
- Use command line scripts to take care of
  - Cleaning out directories
  - Pushing out the new files
  - Reregistering the libraries used by our product
  - Recycling Internet Information Service (IIS) on our servers
  - Turns a web rollout into one line at the command prompt

Create log files.
- Use Ruby and the File library to create useful log files
- Log errors
- Log status messages

Clean out old logs
- Can use Ruby and the File library
- Can use the command prompt directly

Interact with databases
- Use SQL Server Management Studio directly
  - Query Analyzer
  - Object Explorer
- Use Ruby with the following gems
  - Database Interface (DBI)
  - Open Database Connectivity (ODBC)

Interact with CSV files
- Use Ruby with FasterCSV to consume CSV files
- Use Watir to interact with the web site using the read data

Load testing
- Use OpenSta, an open source load testing tool
- Not for beginners, but listed here as a reference

Put some of the above tasks together
- Login/logout of multiple users
- Verify security permissions on multiple users
- Fill out web forms
- Verify data in various forms and formats
- Acceptance and Regression testing of our web app with full logging
- Performance testing

What do **YOU** need to do?

4. "Changes in Latitude, Changes in Attitude"

Sometimes reframing the problem statement can help you get to the final destination.

Creative brainstorming, problem-solving skills, and experimentation can all help achieve the wanted outcome.

- Price
  - Examine the free options.
    - Shell scripting
    - Scripting languages with libraries
      - Perl
      - Python
      - Ruby is my language
      - Watir - used heavily for web testing.

- Support
  - Gain trust
    - Don't try to solve all the problems in one swoop
    - Show a series of successes.
    - Leverage those small victories to larger ones

  - Show the ROI over time.
    - Keep track of how long it takes to do something manually.
    - Keep track of how long your coding activities take.

- Experience
  - Invest in what you know best - yourself.
    - Learn what you can from others
    - Google is your friend.
    - Check out the unit tests shipped with the Ruby libraries (gems)

  - Build on small victories
    - Build up your toolbox.
    - Try to recognize patterns.
  - Don't be afraid to try something new

Example of a creative solution to a task not easily “automated”

- Operating System and software installs problem
  - I used virtual machines running under Windows Server 2008 to help.
    - Manual setup once per OS/software install combination
    - Now a lot easier to spin up that combination in the future
    - Saves time and effort down the road

5. “It’s My Job”

In my opinion, one of the strengths of using Ruby is in that it is an interpreted language. You can use the Interactive Ruby shell (irb) and start investigating the language and its libraries (called gems in Ruby-ese). The following Ruby example can be cut and pasted into irb and used as the basis of your experimentation. You will need to install the following gems

- Watir
- dbi (Database Independent Interface)
- dbd-odbc (Database Driver – Open Database Connectivity)
- Faster-csv

Pointers to more information for the above gems are included in the references. There is also a link to a Watir tutorial contained within the references. The Watir example page referenced in the below code is a

good place to start to familiarize yourself with some of the things that Watir can do, and because Watir is built upon Ruby, you have the entire Ruby interface available to use. The possibilities are endless.

To use the SQL example, you will need to use a custom connection string to match your database and login credentials. You may want to modify the actual SQL query to something that would return results from your database as well.

To use the Faster-csv example, you should have it point to a CSV file that contains a header row with at least two columns – 'TestName' and 'URL'. Or feel free to modify the code to use the headers in your own CSV file.

You may wish to enter commands into irb one line at a time. This way you can experiment with syntax and see the results of your handiwork right after you hit the <ENTER> key!

```
#  The hash mark at the beginning of a line marks the line as a comment.
# This script uses the following libraries – called gems in Ruby-ese
require 'Watir'
require 'dbi'
require 'Faster_CSV'

#---- Use Watir to fill in a web form
# Using the IE developer tool bar or Firebug to investigate your web control is very useful here
# The web page pointed to here is actually part of the Watir tutorial!
browser = Watir::Browser.new
browser.goto("http://bit.ly/watir-example")
browser.text_field(:name => "entry.0.single").set "Watir"

#---- Use Ruby to interact with SQL Databases
sConnect = 'DBI:ODBC:Driver={SQL
Server};Server=MyServer;Database=MyDB;Trusted_Connection=yes;'
hDBI = DBI.connect(sConnect)
sSQL=hDBI.prepare("select companyname from company")
sSQL.execute()
sSQL.fetch do |row|
      puts 'CompanyName  --  '+ row[0]
end
sSQL.finish()

#---- Use FasterCSV to interact with CSV files
# Use a CSV that has a header row
aList=FasterCSV.read('myFile.csv',:headers=>true)

# Iterate thru each line in the file and print out the test case name and target URL
aList.each {|aPage|
  sName=aPage['TestName']
  sURL=aPage['URL']
  puts sName + " – " + sURL
}
```

Pointers to the documentation for each of these libraries are contained in the References section. Also of use are the unittests that come bundled with each gem. The directory where you gem is installed should contain a directory named unittests. On my machine, I can find the Watir unittests at C:\Ruby\lib\ruby\gems\1.8\gems\watir-1.8.1\unittests. The unittests are all written in Ruby. Coupling the unittests with irb has given me great insight into how to better use the Watir gem.

6. "Stories We Could Tell"

Each company I have worked for has been a unique situation, each with its own set of challenges. Yet, the common thread with each is to figure out how to solve a problem with a suitable solution. Start off with a goal, and work your way towards it.

At Thomson Financial, test automation was not frowned upon, but it was not on the forefront of people's minds. Testing was thought of as a manual process. I found a project that could benefit from test automation and worked out a proposal for my bosses to why the project would benefit from automated testing for this data migration project. While the solution was written in perl, it could have been implemented in a number of different ways. I was familiar with perl and was confident that a solution could be crafted in a reasonable amount of time.

Thomson Financial

- Not originally slated for automated testing.
- Goal: Verify data conversion of historical stock quotes to the Euro.
- Not a web app, but a data migration that was under test.
- Original plan was to have someone randomly sample from the 40 million values
  - Figure out to convert one price point and compare it to the "official" number.
    - All the price points for a single stock.
    - All the price points for stocks that used that currency.
    - All the price points for all stocks against all currencies.
- Asked for month to go ahead and develop this from my QA Manager
  - Hesitant at first when I said a few weeks of effort needed
  - More receptive when we examined
    - Time taken to validate one value of one stock price
    - How easy it was to make a mistake in manual validation
    - Risk of inaccuracy encountered after the conversion went live.
- Wrote a framework in perl that examined **ALL** data points not just some of them
  - Found errors in the original conversion formula used
  - Found rounding errors in 2% of the conversion
- Entire effort took about 3 weeks
  - Design and implementation of the test harness
  - Test and retest phase
  - Included all logging and messaging.
- Key wins:
  - Earned trust at Thomson Financial
  - Presented results at Quality Week 1999

At Switchboard, some automation was already in place, but there was a project that accounted for 1/3 of our overall revenue. Making sure that this project was as bulletproof as possible was job number one for me. As such, I dove deep into my bag of tricks to craft a solution that would be reliable to preserve this income stream. Here, the automated solution also included the creation of a new web page that another test tool could drive. The web page I created was a simple web form that interacted with a known API. By using this web page, other tools could rapidly test the API's used by our customers by using web calls rather than low level C++ calls. I was able to leverage this web page to help craft the calls that would be used in load testing. In essence, a single page that I had created using HTML was an integral piece of the functional and performance testing portions of my job.

Switchboard

- Money not an issue, the challenge was testing against a known API that we defined.
- Company was willing to spend money to replicate the production environment for use of development and QA
- We had an expired license of SilkPerformer to update
- API could be construed as query string calls to a web server.

Goal: prove the functionality and reliability of an in-memory database used for searches

Many combinations of query string parameters to examine
I created a test page in HTML for manual queries for basic testing
Each query string parameter was represented on this HTML page
This formed the basis of a list of URLs that could be thrown against the web server manually
Used by both dev and QA.
Perl was used to create many more URLs for testing
Creation of the URLs
Persisting them to a file or files
SilkPerformer was used to drive the testing
The engine that read the files and threw them against the server
Used their built in logging and reporting tools.
Used for load, performance, stress, and failover testing.

Searches completed on the order of microseconds on the in memory databases.
Searches were returned on the order of subseconds on the webserver
Used by a customer that provided Switchboard with over 33% of all revenues.

Key wins:
Gained more trust from my co-workers
The customer never complained about the service offered by Switchboard.

At Compli, the challenge was different. There were no formal QA processes in place. There was no documentation about how the web application should behave. No money was available to buy tools. The C level officers were a little wary of automated testing. As the only QA engineer at the company, I had a challenge in front of me. How did I bring the use of automation into this company? It was at Compli that I dropped perl in favor of using Ruby. Ruby became my go to language because of the Watir gem. Watir comes bundled with a suite of unit tests that I used as examples to experiment with. It is a reliable tool that has become the cornerstone in al of my web testing efforts. Best of all, this tool fit the budget that I was working with.

Compli

Company had no real QA processes in place - ever.
Previous releases were painful on the company.
Buggy code released to production
Many phone calls with customers upon rollout
Many hotfixes created to address shortcomings
We had no money allocated for any testing tools.
Management didn't have a good idea of what could be accomplished.

Goals:
Make releases less painful for the company
Be able to turn around builds quickly
Define and implement a set of automated acceptance tests
Show the value of automated tests

I asked various people what was currently being verified with each release
- Customer support
- Internal channel support
- Basically one person from every department
Creation of a manual acceptance test plan
Went back to the people queried for input and comments.
Gained the trust of these people in the process
Used Ruby to implement the test plan

Initial implementation was only 40 scenarios with 200 verification points
Used Watir to drive the browser
Used ODBC to execute SQL queries as needed
Used Test::Unit as the test framework
Used the log4r library for logging purposes
Constructed a reusable automated testing framework that could be expanded

Key wins:

After the first rollout, both the president and CEO dropped by to say that this was the smoothest release ever in the history of Compli.
I gained more of their trust in the process of several releases.
I learned how to use Ruby.

Now we have an acceptance test framework that examines over 1200 test points
Takes about 2 person days to execute the test cases manually
Takes about an hour to run in the automated framework
Base regression test suite that verifies over 11000 test points
Takes about 3 hours to run to completion
More detailed regression tests also available
Multi day tests touching more edge cases
Run as needed.

More recently was a project that I was involved in this past May. We had a new integration with a 3rd party web site where we created reports based on the data that they provided. It entailed matching data against existing users in our system. The challenge was that sometimes the data didn't map to the users 100% of the time and we needed to be able to examine those unmatched data points to examine why we could not match the data. Was the problem in our matching algorithm or simply that the data really could not be matched.

The following project is my favorite example that illustrates the use of test automation in a non-traditional sense. The solution I crafted is not a 100% push button solution but illustrates how I used different tools to make the testing effort more automated than the straight forward (but error prone) manual verification. The solution implemented also did the verifications in a way that it would easy to verify new builds down the road.

Compli, week of May 16th, 2011

Goal: Verify our reports against the data generated by a 3rd party site (system of record).
Ideally: our reports would directly match the 3rd party reports.
Data Import
Users records to match against
3rd party incident reports from external site
Imagine our surprise when our reports didn't exactly match the 3rd party reports!
Reconcile report of unmatched data – one of two outcomes
Missing user records
Bad/missing drivers license data
Process every line on the exception report against our system
Reconcile our report vs. their report.

Test attempt 1

800 unmatched user records were found
Timed 3 records manually to estimate how long verification might take.
Entailed going thru the UI to verify the problem.
30-60 seconds per row to hand verify
Estimate of 38400 seconds (10.6 hours) to verify all 800 records

"Automated" Solution
- Export the unmatched data report from our app to Excel - manual
- Convert the Excel report to CSV – manual
  - Includes running an Excel macro to handle double quoting all fields
    - Excel save to CSV doesn't do this automatically for all fields
    - Poses problems when "numbers" are not interpreted as strings
- Formulate SQL queries to run – scripted in Ruby and ODBC
- Compare **every** line on the export file to a SQL query – scripted in Ruby

- Spent about 6 hours writing and testing the above process to help the verifications
  - Converting Excel to CSV issue
    - Data with leading zeros got munged and interpreted as integers
  - Add in
    - Debug lines
    - Output Logging
    - Better Comments
  - Made it more generic to allow greater reuse
    - Allow company name to be passed in
    - Made the SQL handle this passed in parameter

- Script took less than a minute to run

Test attempt 2
- Bug fixes came in to QA.
- 250 records to verify after bug fixes
- Guess how long it took to verify this record set.
- Now we were able to show the customer how many data issues were present in the system of record
  - Missing or extra leading zeros in ID fields
  - Missing or extra spaces and dashes on ID fields
  - Spelling input errors on text fields.

7. "Don't Bug Me"

Watch out for cases where you get so wrapped up on things that you can lose sight of what you're actually trying to accomplish. Watch out for the following traps

Don't lose sight of your day job
- If your day job is to create automated tests, you'll be in better shape.
- If your job is to manually test, don't forget what you're getting paid for

Don't lose sight of the original goal
- Focus on what needs to be done.
- Don't get sidetracked by things that are "cool"

Don't take comments personally.
- Now you're in the position of developing software.
- Others will give valuable input.
- Listen to them

Don't build something you don't really need just because it's cool.
- Create tools that are useful
- Create tools that accomplish the stated goal

KISS

Don’t get stuck on a problem
- If you do get stuck, you need to get unstuck
- Maybe someone else has a different perspective on things
- Comments from others can give you a different perspective to the issue at hand
- Maybe a problem reframe is needed

Don't bite off more than you can chew
- Recognize potential problem areas
- Ask for help when its really needed

Don't just cut and paste code
- Understand what the code is doing and why.
- If you don’t understand the code, you should not be using it.
  - How will you maintain it when problems arise?

Don't try to automate something that should not be automated.

Candidates for the “Do not automate” pile
- Cost/time savings will not appear
- When the script maintenance cost will be too high
- A really complex one-off test scenario
- Unstable user interface in use
- Testing tools unavailable or difficult to use

8. "Cheeseburger in Paradise"

Hopefully these ideas will serve you well in breaking down the barriers to getting automation into your company. Trust, communication, and documentation are some of the keys to getting what you need to accomplish this.

Figure out what the goal is
- If you can’t define what you want, how do you know what to do?

Examine how you can break up the problem space into discrete pieces that you can solve
- What do you already know how to do?
- What will you need help with?
- What new things will you have to learn?

Use whatever will solve the problem.
- Using the same language will be easier, but it's not the end of the world
- As you become more successful, settle on the technology side of things

Recognize patterns when you can.
- If you are seeing something being done over and over again, try to generalize the steps for reuse.

Be sure to comment the scripts!
- It helps with maintenance.
- I used to be a perl guy. Now my tricks have been reimplemented in Ruby
- By commenting the perl scripts, I knew what I needed to accomplish in Ruby
- Makes it easier to hand off to someone else for use
- Commented scripts can be used for training purposes.

Show the ROI

- Time taken to create the scripts vs. the time to run them manually
- Show your managers that this is time well spent

Build on your success

- Success will beget more success
- Build up the trust factor and things will get easier down the road

References

Ruby - http://www.ruby-lang.org/en/

Ruby Quickstart - http://www.ruby-lang.org/en/documentation/quickstart/

Watir - http://watir.com/

FasterCSV - http://fastercsv.rubyforge.org/

DBI - http://www.tutorialspoint.com/ruby/ruby_database_access.htm

ODBC - http://www.ch-werner.de/rubyodbc/

Microsoft Powershell Scripting - http://technet.microsoft.com/en-us/scriptcenter/dd742419

Microsoft Command line reference - http://technet.microsoft.com/en-us/library/bb490890.aspx

OpenSta - http://opensta.org/

Selenium - http://seleniumhq.org/

Jimmy Buffett - http://www.margaritaville.com/