STAR CANADA

A TECHWELL EVENT

W1

Test Automation

Wednesday, October 17th, 2018 10:15 AM

7 Sure-fire Ways to Ruin Your Test Automation

Presented by:

Seretta Gamba

Brought to you by:

TECHWELL™

Seretta Gamba

Seretta Gamba has forty years' experience in development and fifteen in test automation. After going through all the usual developer roles, in 2001 she was put in charge of test automation for her current company. She developed a framework that enabled her company to quickly get excellent results. After having talked about the framework at a couple of conferences, she met Dorothy Graham and was invited to write a chapter in the book Experiences of Test Automation. With Dorothy she has been developing the Test Automation Patterns Wiki and is now writing a story book about the patterns.

Seven proven ways to ruin your Test Automation

Agenda

Introduce each method
Explain about possible defences against it
List efficient countermeasures
Rate it

Conclusion

TEST AUTOMATION PATTERNS

Testautomationpatterns.org

TEST AUTOMATION PATTERNS

Diagnostic

- *ISSUES*
  - *PROCESS*
  - *MANAGEMENT*
  - *DESIGN*
  - *EXECUTION*

- PATTERNS
  - PROCESS
  - MANAGEMENT
  - DESIGN
  - EXECUTION

Methods

1. Guillotine Methods

   Step 1: develop test automation

   Step 2: cut the head off

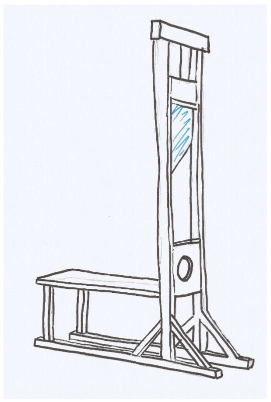
2. Boiled-Frog Methods

   Step 1: develop test automation

   Step 2: more and more effort to keep automation running

   Step 3: shelfware

Guillotine Methods

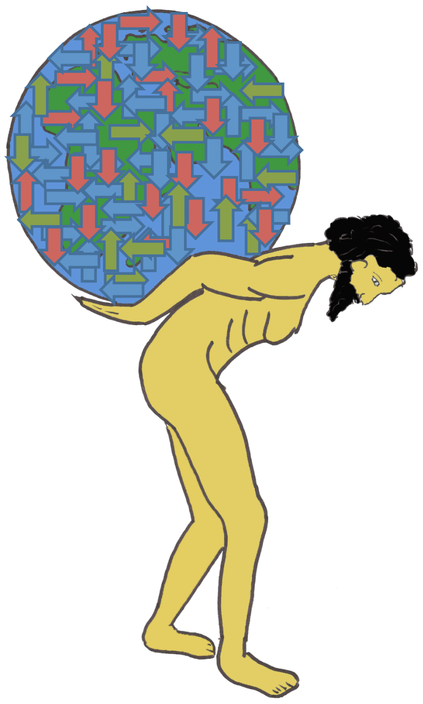


1 Fire Atlas

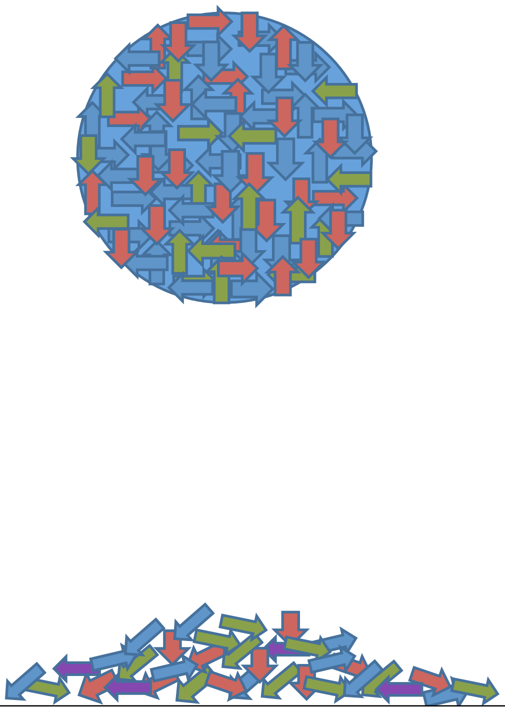
2 Impossible Goals

3 Change Tool

1 – Fire Atlas



1 – Fire Atlas



1 – Fire Atlas

Test Automator

1. develop test automation
   - no standards
   - no documentation
   - no specs
   - no backup
   - no team
   - an exotic automation tool or a self-programmed framework
   - tests that are only partially automated and need manual interventions at some point
2. leave for a better job!

1 – Fire Atlas

Manager

1. put Atlas in charge of test automation, but be sure not to demand:
   - standards
   - documentation
   - etc.
2. fire Atlas because he or she has not applied standards, written documentation etc.

1 – Fire Atlas

Defenses & Countermeasures:

Issue *KNOW-HOW LEAKAGE*:

- DEPUTY
  - Test Automator: I'll have time after milestone x
  - Manager: no resources, but I'll keep it in mind
- DOCUMENT THE TESTWARE
  - Test Automator: no time just now
  - Manager: no need, just ask Atlas!

1 – Fire Atlas

Defenses & Countermeasures:

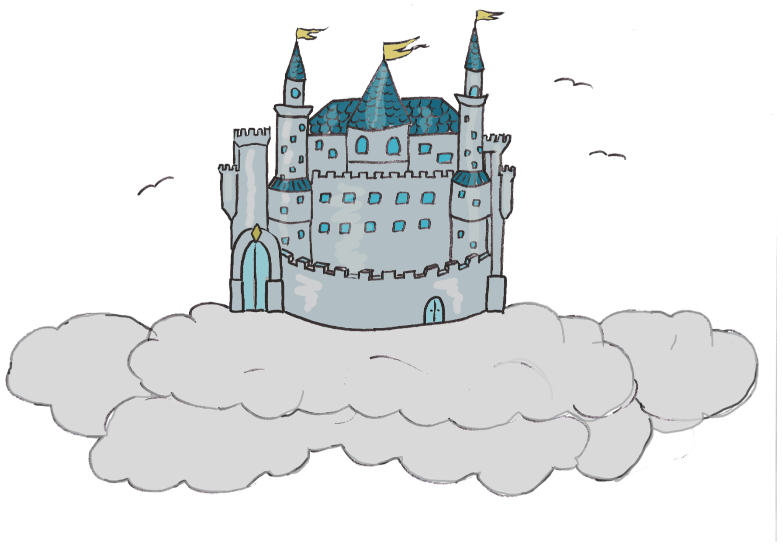
Issue *KNOW-HOW LEAKAGE*:

- GET TRAINING
  - Test Automator: go to all trainings
  - Manager: only Atlas ("save resources")
- PAIR UP
  - Test Automator: no need (1 person team)
  - Manager: good idea, but…1 person team!

1 – Fire Atlas

Rating:

2 – Impossible Goals



2 – Impossible Goals

- automate all manual tests
- any tester can do automation
- automated tests should find lots of bugs
- automating a test gives immediate Return on Investment (ROI)
- a test tool is all that's needed to start with test automation
- maintenance is not needed for test automation, only for 'real' development
- if automated tests pass it means that the software is bug-free

2 – Impossible Goals

Defenses & Countermeasures:

Issue *UNREALISTIC EXPECTATIONS*:

- SET CLEAR GOALS
  - *UNREALISTIC EXPECTATION* = goal
- DO A PILOT
  - waste of time! get going with automating all those manual tests!

2 – Impossible Goals

Defenses & Countermeasures:

Issue *UNREALISTIC EXPECTATIONS*:

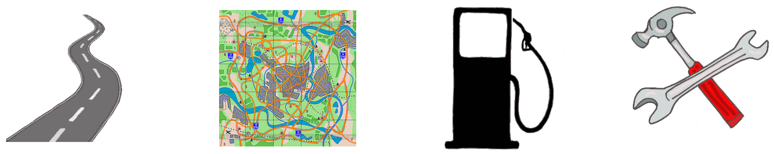
- SHARE INFORMATION
  - doesn’t fit with your outlook? ignore it!
  - conferences? people get weird ideas!
  - get on with automation instead of just chatting together

2 – Impossible Goals

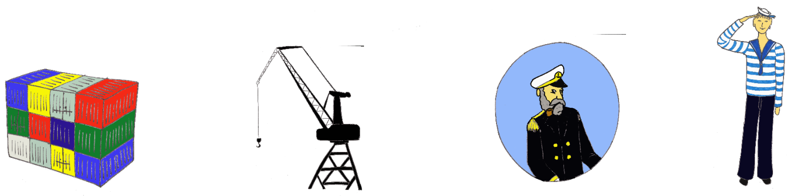
Rating: ☠☠☠

Excellent

3 – Change Tool



3 – Change Tool



3 – Change Tool

1. Use exclusively the tool’s functionalities
   - capture-replay
   - tool‘s own special script language
   - tool libraries
   - tool‘s own object recognition

2. Change tool

3 – Change Tool – Step 1

- Test automator: use exclusively the tool’s functionalities
- Developer: implement application so that the current test tool cannot support it.

3 – Change Tool – Step 2

Option 1: Just wait!

- your tool vendor will develop a new tool (of course not downward compatible with the old tool), stop supporting the old tool and you will have to switch to a new tool
- ➔done!

3 – Change Tool – Step 2

- Test automator:
  - none of the tests for the new functionalities can be automated
  - the tests in the current tool are too slow, too obscure or whatever
- Marketing:
  - the current application needs a modern makeover (*coincidentally* not supported by the current tool)!
- Change tool: ➔done

3 – Change Tool

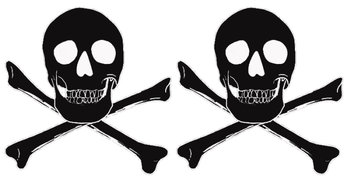
Defenses & Countermeasures:

Issue *TOOL DEPENDENCY*:

- TOOL INDEPENDENCE
  - Test Automator: rewrite everything in that old messy tool? no way!
  - Manager: we are not going to change the tool: no need to make such a fuss

3 – Change Tool

Rating:



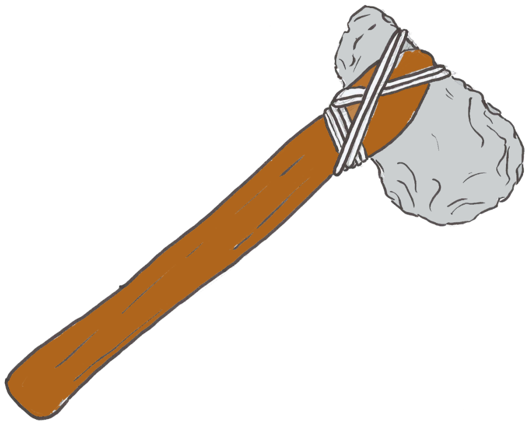
Good

After having changed to a new tool, go on using it exclusively: you will be able to apply this method successfully again and again!

Boiled Frog methods

4 Primitive Programming Practices
5 Manual Approach
6 Feigned Support
7 Ever-increasing Maintenance

4 – Primitive Programming Practices



4 – Primitive Programming Practices

Test Automator

- automate the test cases using capture-replay
- forget modularity, favour 'spaghetti code'
- avoid any reuse of code or data
- no documentation
- use 'crazy' names
- giant scripts (one script tests the complete application!)

4 – Primitive Programming Practices

Manager

- everybody should just automate more and more tests.
- make sure that test automators get no time for refactoring: "they can play around after they have automated all the test cases!"

4 – Primitive Programming Practices

Defenses & Countermeasures:

Issue *BRITTLE SCRIPTS*:

- GOOD PROGRAMMING PRACTICES:
    - DESIGN FOR REUSE
    - KEEP IT SIMPLE
    - SET STANDARDS
    - Design INDEPENDENT TEST CASES
    - use at least DATA-DRIVEN TESTING or, better, KEYWORD-DRIVEN TESTING

– Test Automator: mañana!

– Manager: automate all the test cases!

4 – Primitive Programming Practices

Rating:

Excellent

5 – Manual Approach

5 – Manual Approach

Test Automator

– long chains of tests that depend on each other
– one test executes practically every functionality in the SuT
– complicated GUI workarounds (instead of reaching into the database)
– compare results on the GUI
– too complicated? do it manually!

5 – Manual Approach

Manager

- every manual test has to be automated!
- now!
- refactoring is only needed for development and certainly not for test automation!

5 – Manual Approach

Defenses & Countermeasures:

Issue *MANUAL MIMICRY*:

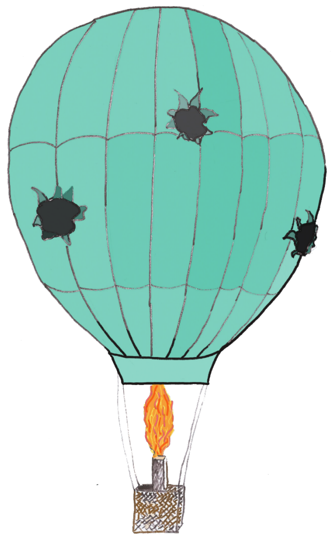
- THINK OUT-OF-THE-BOX
  - Tester: write un-understandable test specifications
  - Manager: tests are to be automated exactly as is

5 – Manual Approach

Rating: ☠☠☠

Excellent

6 – Feigned Support



6 – Feigned Support

Manager

- promise 100% support, but never really give it.
- don't plan support for test automation (especially not from testers)
- test automators work part time on automation

6 – Feigned Support

Defenses & Countermeasures:
Issue *INADEQUATE SUPPORT*:

- MANAGEMENT SUPPORT
  - Manager:
    - assert again and again that you support test automation 120%!
    - keep a list of possible excuses why you cannot give the requested assistance just now

6 – Feigned Support

Rating: ☠☠☠

Excellent

7 – Ever-increasing Maintenance

7 – Ever-increasing Maintenance

Developer

- ‘minimal’ changes to the object hierarchy in .net
- GUI-Components not supported by the tool
- changing texts or their sequence in logs or outputs
- change databases or database-qualifiers
- never give notice that you have changed something

7 – Ever-increasing Maintenance

Test Automator

- avoid using variables, use constants whenever possible
- never reuse scripts or data.
- define the GUI-Objects to be driven with their pixel positions or run-time indices
- give non-descriptive names to objects, variables or parameters
- never save scripts or data

7 – Ever-increasing Maintenance

Defenses & Countermeasures:

Issue *HARD-TO-AUTOMATE*:

- TESTABLE TESTWARE:
  - ASK FOR HELP
  - SHARE INFORMATION
  - DO A PILOT

– Developer: ignore the needs of test automation: you are working for the customers!

– Test Automator: as soon as you have time, you will do some refactoring!

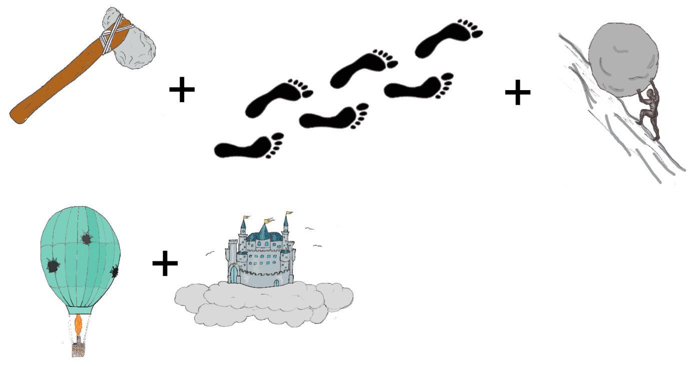
7 – Ever-increasing Maintenance

Rating:

Excellent

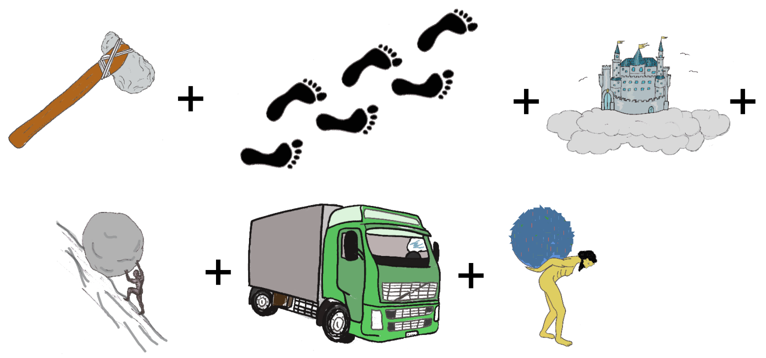
Conclusion

- Best results mixing the methods!
- Thousands of combinations!



Conclusion

- Best results mixing the methods!



7 Ways to Ruin Your Automation

1 Fire Atlas: depend on one person
2 Impossible Goals: management favourite
3 Change Tool: tool drives automated testware
4 Primitive Programming Practices: no standards
5 Manual Approach: automate manual tests „as is“
6 Feigned Support: promise, don‘t deliver
7 Ever-increasing Maintenance: build technical debt

I wish you success in applying these methods to destroy your test automation!

srttgmb@yahoo.com