Cognizant Softvision

Real-World Automation

What It Is And What It Should Be

@pgrizzaffi

Who Is This Guy?

Paul Grizzaffi - Principal Automation Architect at Cognizant Softvision

Career focused on automation

“Software Pediatrician”

Advisor

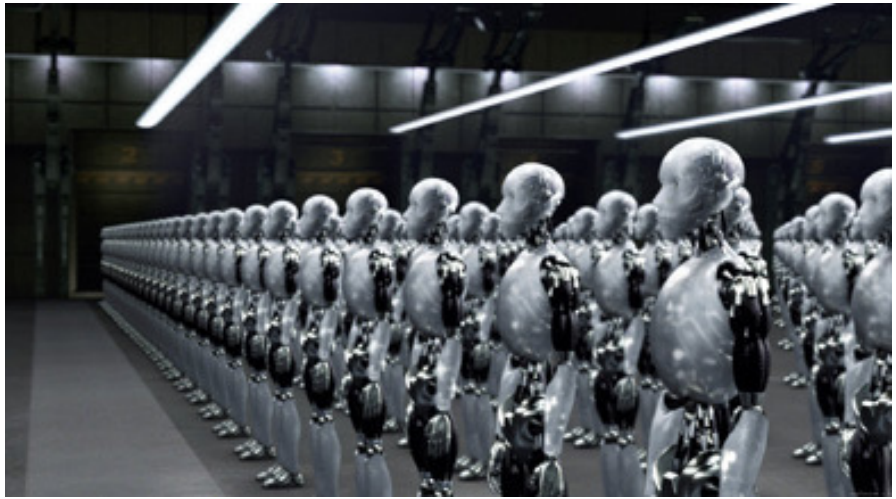
- STPCon
- Advanced Research Center for STQA at UT Dallas

Heavy metal enthusiast

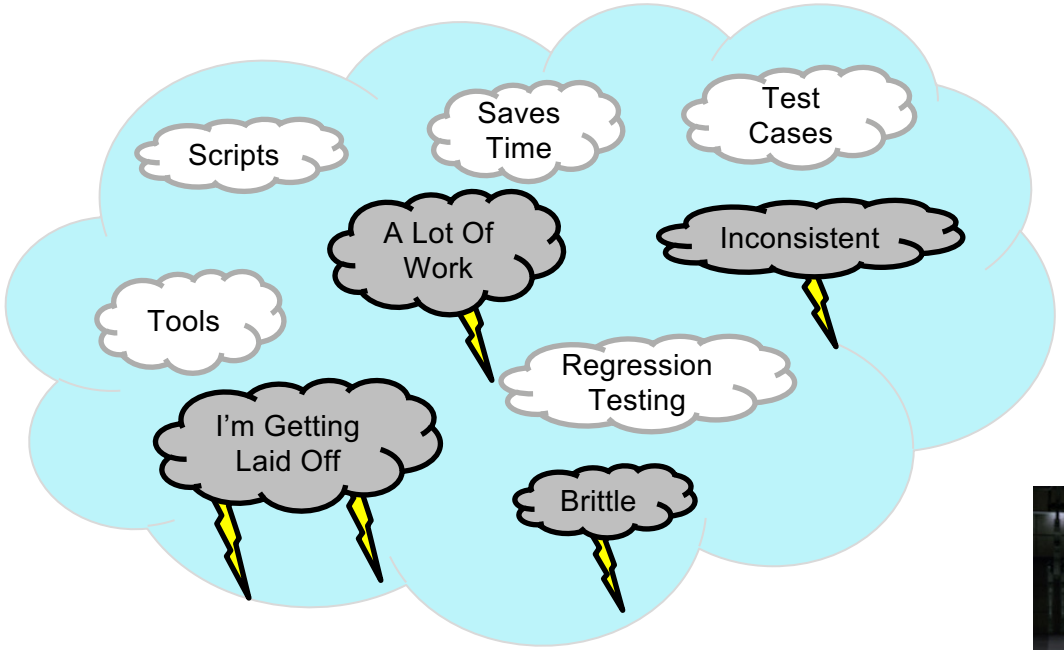
Where Is This Guy?

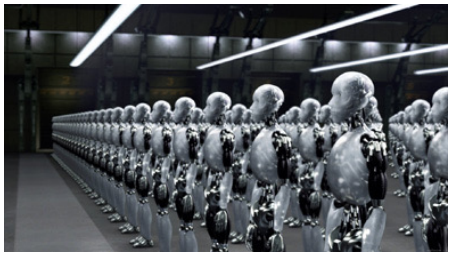
paul.grizzaffi@softvision.com
http://www.linkedin.com/in/paulgrizzaffi
@pgrizzaffi
http://responsibleautomation.wordpress.com

What Comes To Mind?



What Comes To Mind?





What's It Like Out There?

What's It Like Out There?

How many test scripts do you have?

How much of regression testing is automated?

Why aren't you using Selenium? (or Cypress.io? or UFT? or…)

Why do we need testers?

What's It Like Out There?

Primarily based on test cases

Big focus on smoke and regression

Big focus on UI (but that's changing)

Tool-centric

- Selenium (open source)
- QTP/UFT (HP)

Testing is dead (no I'm not)

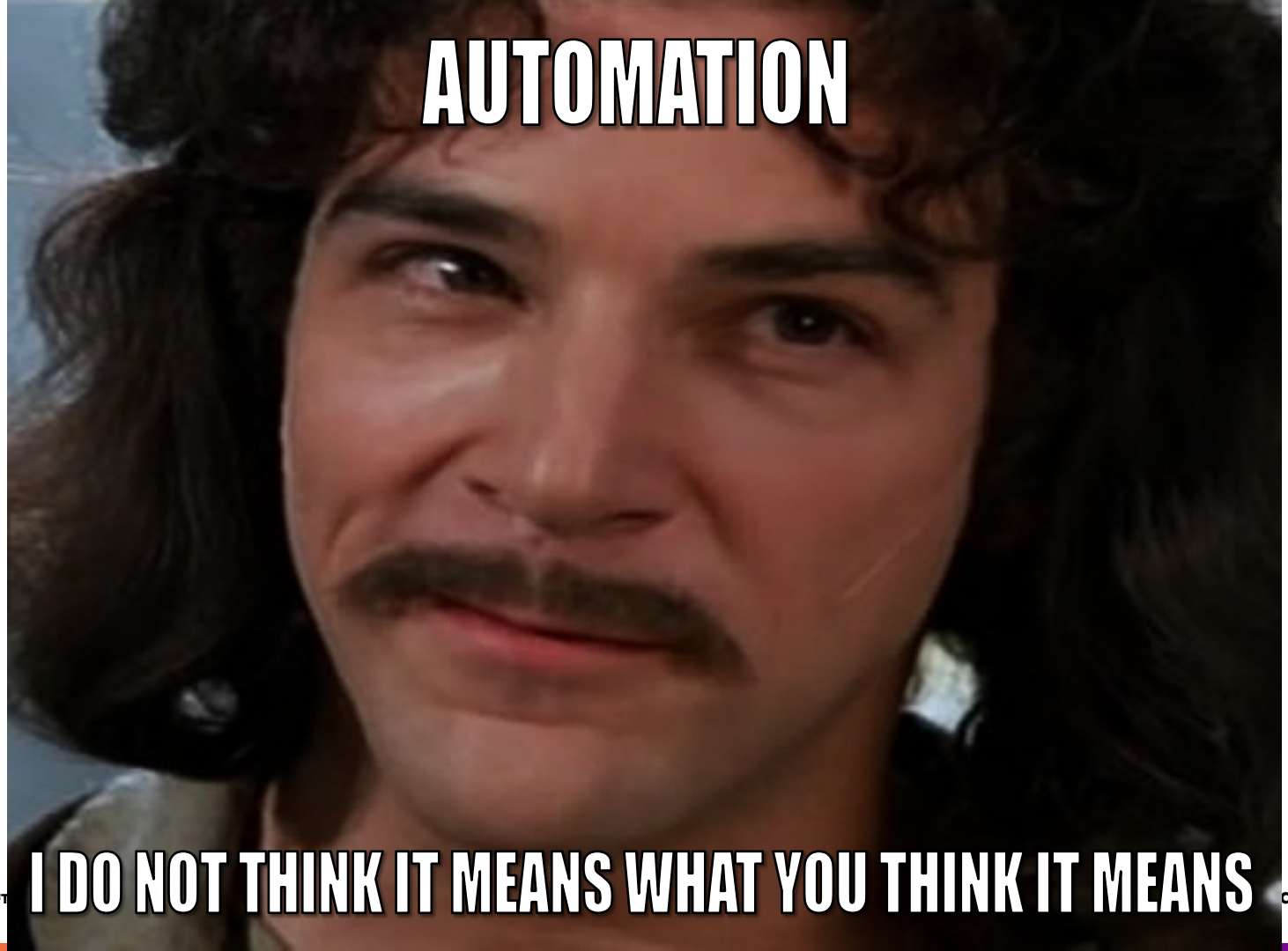
SDETs – we don't need testers

Process/Framework-Driven Automation

We automate because we “do Agile”

We automate because we “do Scrum”

We automate because we “do DevOps”



Traditions Can Be Important

Traditions Can Be Important

Traditional automation

- Detect behavior changes
- Reduce effort on smoke and regression testing
- Earlier execution, earlier alerts
- Scheduled execution

My previous organizations have this and it’s valuable

Is there something else?

What If We Think Differently?

What If We Think Differently?

"Let's help the humans"

What makes us more efficient or more effective?

What's valuable?

What hurts?

Instead of *automation*, how about *assistance?*

Umbrella term for non-traditional automation

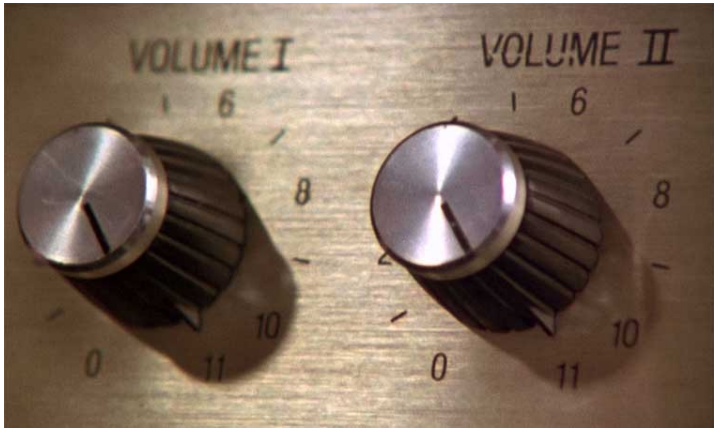
- “Words mean things” – Patrick Amaku
- Changing meanings is difficult
- New vocabulary for new concepts

Things that increase the value of manual effort

- “Off label” tool usage
- New tools, applications, scripts
- Tools not traditionally thought of as automation

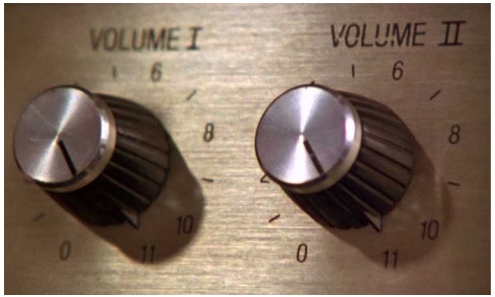
Let's Look At Some Examples

High Volume Automated Testing



“a family of testing a that enable the tester to create, run and evaluate the results of arbitrarily many tests”

Workshop on Teaching Software Testing (WTST) 2013



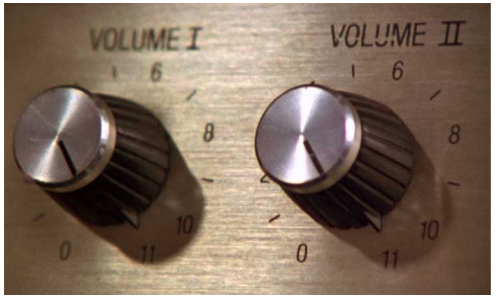
Aka HiVAT

Dr. Cem Kaner, J.D., Ph.D.: http://kaner.com/?p=278

Harry Robinson: http://wtst.org/?page_id=169

Interesting facets for the organization

- Many executions
- Random execution
- Results vetted by humans



Scud

The premise

- Product is large, complex, aging
- Not feasible to enumerate and follow all paths

The solution

- Random menu clicker – “Scud”
- Looking for things that “don’t seem right”
- Selenium-based Python script: 32 hours of effort
- Found four issues in the first week of use

Why that solution?

- Value shown in previous organization
- Selenium was the right kind of tool kit
- Broad license usage and not competing with other tool

ACH File Checker

ACH File Checker

The premise

- Need to test that ACH files are created correctly
- Funding was cut; project to end early
- Development would not be completed

The solution

- Create a command line app to perform part of the testing
- C# program: ~5 hours of effort
- Eliminated hours of testing effort

Why that solution?

- The code was there, but provided no current benefit
- Cheap, fast way to enable value
- Something's better than nothing

Data Server Regressor

The premise

- “Regression” testing between software versions with volatile stock market data
- Takes about 8 hours of effort

The solution

- Comparison tool; connect to both servers and compare
- Time to test reduced 1 minute
- C++ program: “Break Even point”: 6 weeks

Why that solution?

- Traditional test scripts not appropriate
- Existing API into the product
- C++ primary development language
- Product bug found during first execution

Account Creator

The premise

- User account creation is a pain in the…well…you know…
- UI is a slow, lots of typing
- Multiple tables to update

The solution

- Script created in Selenium IDE
- Effort: “a couple of days”

Why that solution?

- Saved ~20 minutes per account entry
- Easy to automate via UI
- Hard to automate via database interaction
- Coding was not forte

Zero Remover

Zero Remover

The premise

- Thousands of “golden files”
- In some cases DB now returns NULL instead of 0.00
- Manual effort estimate: 4 – 6 weeks

The solution

- A program to do the file transformation
- C# program: 5 hours

Why that solution?

- Disposable
- Windows/.Net shop
- Easy to distribute executable

Don’t forget about “record and playback”

Some Words About Effort

Effort calculation is a funny thing

ROI can be difficult to calculate

http://blog.smartbear.com/testing/there-aint-no-roi-in-software-testing/

Dawn Haynes

“Opportunity cost” can be easier and as useful

Think about value

Don’t do “Process/Framework Driven Automation”

Some Automation People I Read

TAU	@TestAutomationU
Angie Jones	@TechGirl1908
Richard Bradshaw	@FriendlyTester
Alan Richardson	@EvilTester
Rex Jones	@RexJonesII
Kristin Jackvony	@KristinJackvony
Paul Merrill	@dpaulmerrill
Melissa Tondi	@melissatondi
Joe Colantonio	@jcolantonio
Dorothy Graham	@DorothyGraham
Ministry of Testing	@ministryoftest
MoT Dallas	@MoTDallas
Software Test Professionals	@SoftwareTestPro

@pgrizzaffi

This is different

Nothing wrong with scripting test cases, but that’s an implementation

This is software development

Situational – the “knowns” help guide

Usage profile: running, distributing

Life span: disposable or long term

Questions!


paul.grizzaffi@softvision.com


http://www.linkedin.com/in/paulgrizzaffi


@pgrizzaffi


http://responsibleautomation.wordpress.com