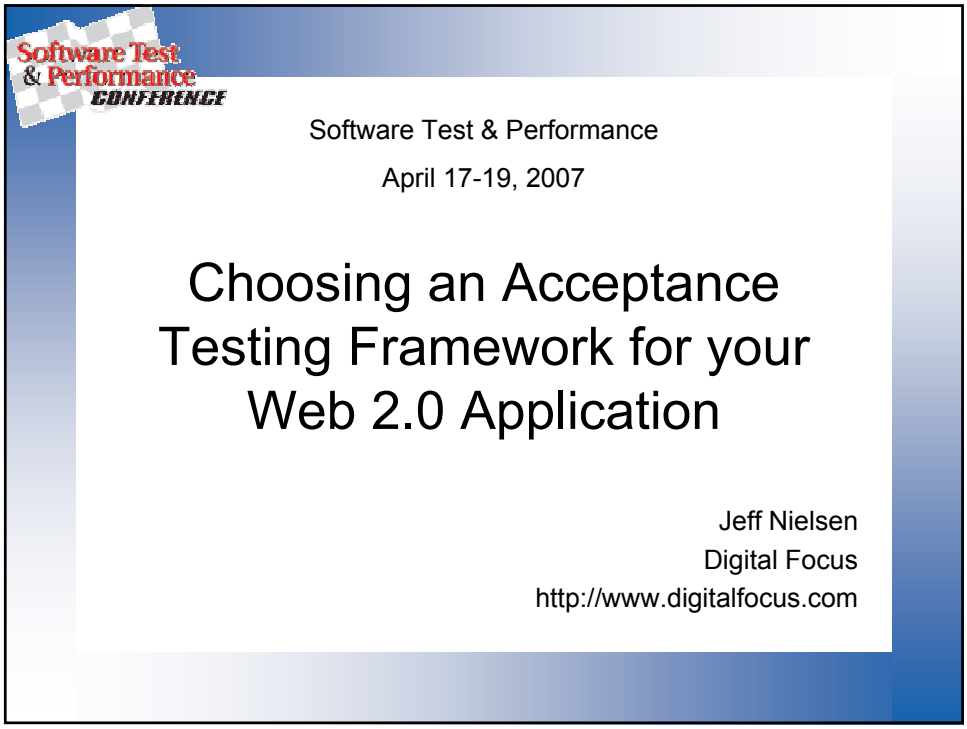
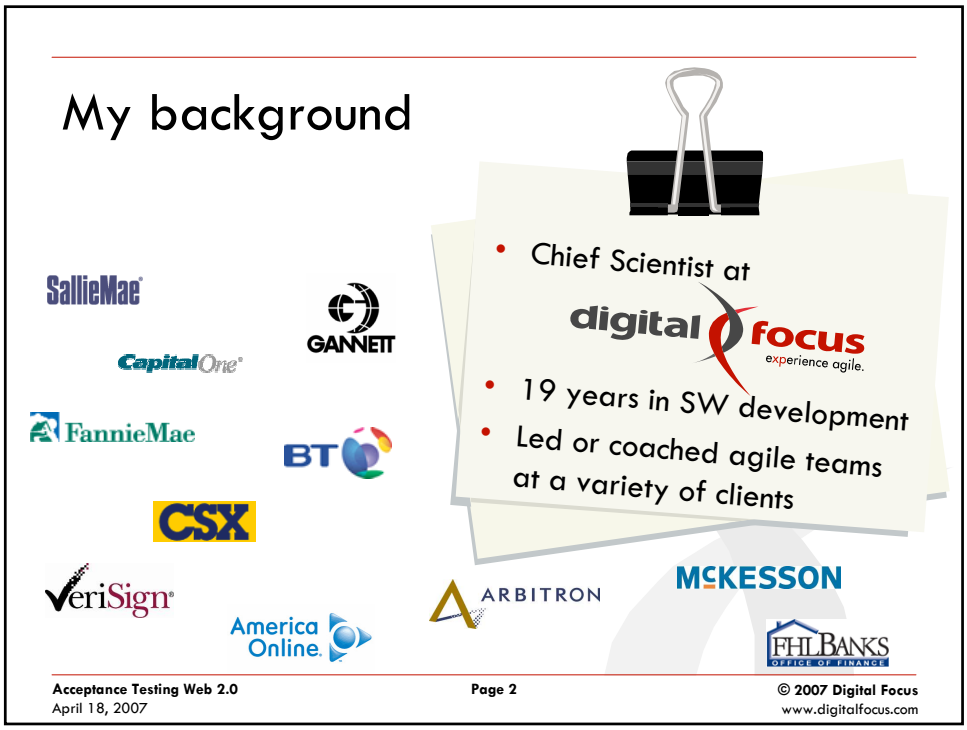
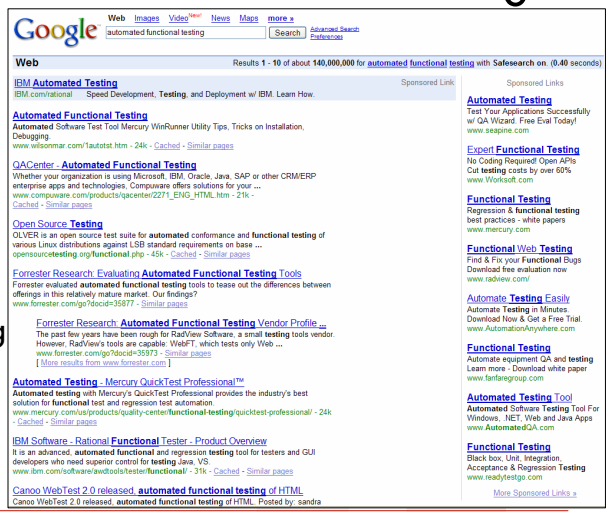

Software Test
& Performance
CONFERENCE
Software Test & Performance
April 17-19, 2007
Choosing an Acceptance Testing Framework for your Web 2.0 Application
Jeff Nielsen
Digital Focus
http://www.digitalfocus.com

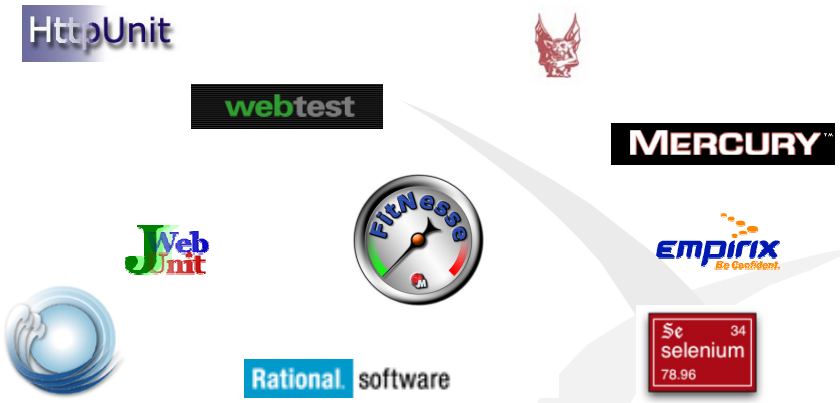

My background
SallieMae
GANNETT
CapitalOne
FannieMae
BT
CSX
VeriSign
America Online
ARBITRON
McKESSON
FHLBANKS
OFFICE OF FINANCE
Chief Scientist at
digital focus
experience agile.
19 years in SW development
Led or coached agile teams at a variety of clients
Acceptance Testing Web 2.0
April 18, 2007
Page 2
© 2007 Digital Focus
www.digitalfocus.com

There has been a recent resurgence of interest in automated functional testing

- Everyone wants to do it
  - From the XP'ers to the Waterfall community
- Everyone has a hard time getting it going



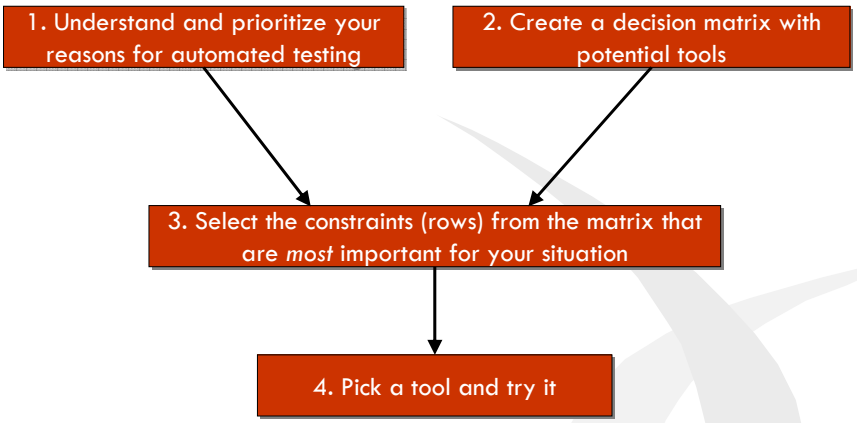
There are many tools and frameworks to choose from



Today we will discuss how to get value out of your automated acceptance tests

- Framework for selecting an automated testing approach
- Example of applying the framework for a "Web 2.0" application

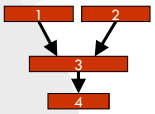
Four steps for selecting a tool



1. Understand & prioritize your reasons for doing auto. functional testing

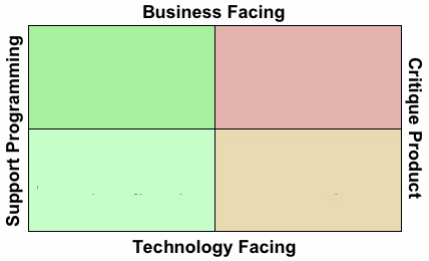
- Provide faster feedback to developers to improve quality
- Reduce workload/cost of manual testing
- Improve communication between dev. team and customer
- Prove to customer that application works
- Create a regression testbed for safety/ease of ongoing change

How do these potentially conflict?



You can't have it all

- There are fundamentally different philosophies about where automated tests provide the biggest value
- A tool/approach that supports one philosophy will not work well for another

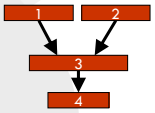


Brian Marick, *Testing Categories*
http://www.testing.com/cgi-bin/blog/2003/08/21#agile-testing-project-1

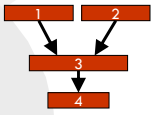
2. Create a decision matrix

- Who will be reading the tests?
  - developers, analysts/testers, customers
- Who will be writing/automating the tests?
  - developers, analysts/testers, customers
- Who will be executing the tests?
  - build system, developers, analysts/testers, customers
- What stage of stability is the application in?
  - prototyping, iterating, maintenance
- How complex/detailed are the tests?
  - support for abstraction, DSL
- How well does a tool support your domain?
  - support for specific features

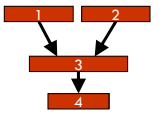
Starter questions



2. Add potential tools to the matrix

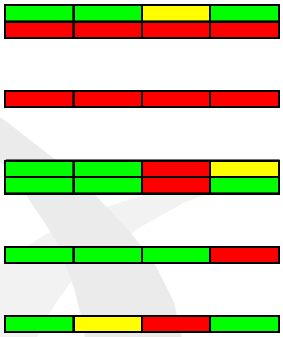


3. Select constraints (rows) that are most important for you



e.g., with prioritized reasons:

1. Improve communication
2. Regression testbed



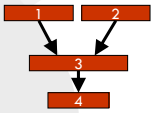
Accompanying constraints:

- Customers & devs. can read tests
- Customers can write tests
- Build system & developers can run tests
- Support for domain-specific abstractions
- Support evolving functionality

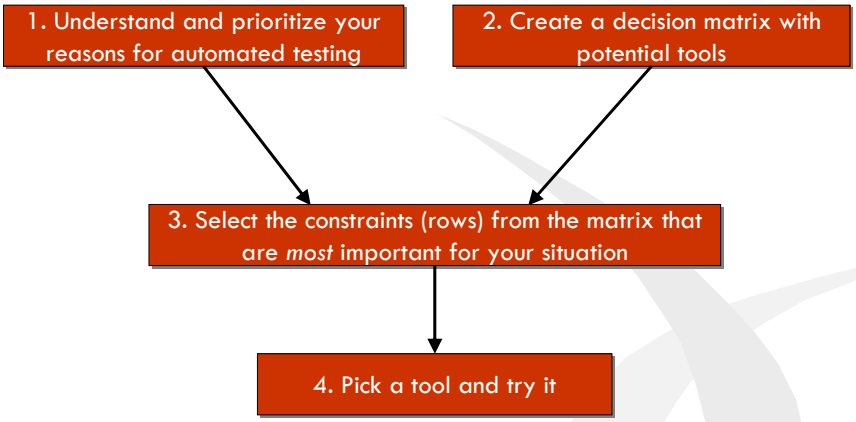
4. Pick a tool and try it!

- Don't expect to get it right the first time
- The devil is in the details
  - At some point the feedback we get from trying *something* provides more value than continued analysis

A sign along the Appalachian Trail
Photo: Chris Guiley



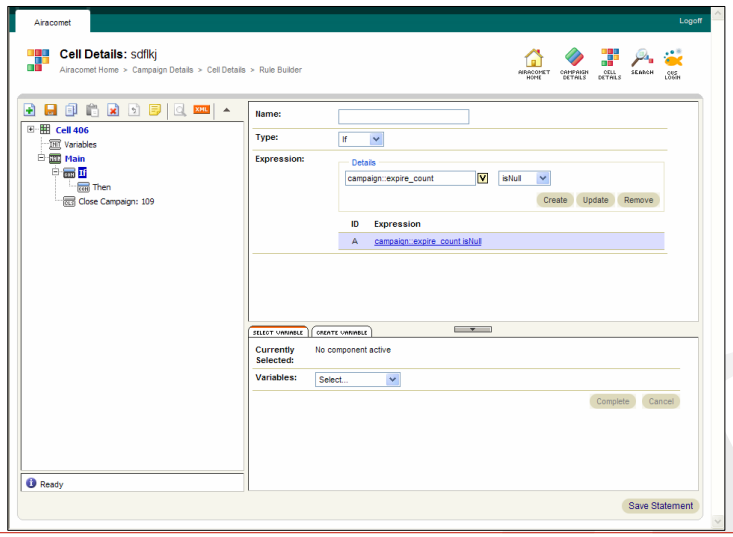
Let's apply this process to a Web 2.0 application



What is special about "Web 2.0"?

- Relies much more on built-in features of modern web browsers
- Heavy use of advanced JavaScript
- Asynchronous behavior
- Dynamic DOM for both input and output

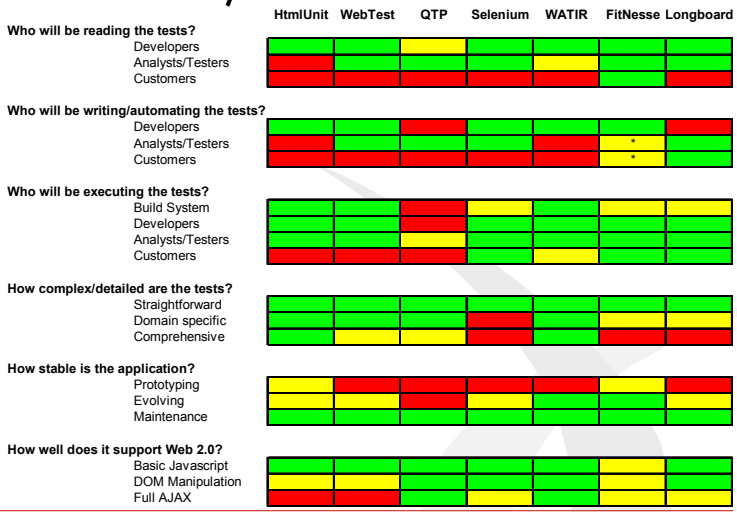
Airacomet



List of potential tools

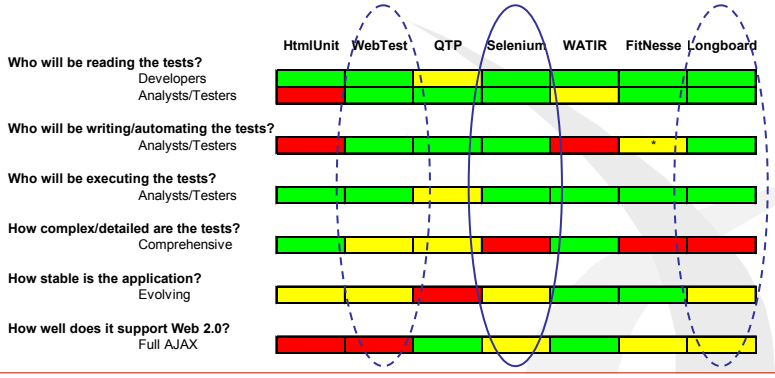
- HtmlUnit
  - Java-based extension to JUnit
  - Simulates browser
  - JS support through Rhino
- Canoo WebTest
  - XML layer on top of HtmlUnit
  - Add-ons for PDF, Excel, email
- QuickTest Pro
  - Mercury tool
  - Windows-based record/playback & scripting
- Selenium
  - Open-source JavaScript test framework
  - Runs inside frames in browser
- FitNesse
  - Wiki plus FIT framework
  - HTML Table-based tests
  - Programmers author “fixtures”
- WATIR
  - Ruby testing framework
  - Controls IE through COM
- Longboard
  - Bridges FitNesse to WATIR

Our decision matrix (best knowledge we have to start)



Selecting specific constraints based on our priorities

1. Create a regression test suite to support release #2
2. Provide better feedback to the developers



Trying out the tools

- Main goal at this point is to learn the specifics of testing *this application*
- Be prepared to be wrong
- Try to think about the matrix ratings that you are willing to compromise and those you are not
- How easily can you customize a tool to address its deficiencies?

Trying out the Tools: Canoo WebTest

- What worked?
  - Creating & deleting Campaigns
  - Nice reporting
- What didn't?
  - Doing anything else
  - JS support not adequate for the AJAX functionality
  - XML can be difficult to read
- Demo

WebTest/HtmlUnit Javascript support

Event	Description
onload	"onload" handlers on body or frameset elements will be triggered after the page has been loaded but before it is returned to the caller.
onunload	Not implemented yet
onclick	This will be triggered when the click() method of a ClickableElement is called.
ondblclick	Not implemented yet.
onmousedown	Not implemented yet.
onmouseup	Not implemented yet.
onmouseover	Not implemented yet.
onmousemove	Not implemented yet.
onmouseout	Not implemented yet.
onfocus	The method WebClient.moveFocusToElement(...) will change the focus to the appropriate element. This event will be triggered when an element gets focus.
onblur	The method WebClient.moveFocusToElement(...) will change the focus to the appropriate element. This event will be triggered when an element loses focus.
onkeypress	Not implemented yet.
onkeydown	Not implemented yet.
onkeyup	Not implemented yet.
onsubmit	This will be triggered when a form is submitted using a control on the form.
onreset	This will be triggered when a form is reset either by HtmlForm.reset() or via the javascript reset() method.
onselect	Not implemented yet.
onchange	This will be triggered when the appropriate setValueAttribute() method is called.

Trying out the Tools: Longboard

- What worked?
  - Easy to create Campaign & Cell
  - Very readable by customer (with name mapping)
  - Real browser
- What didn't?
  - Custom tree not supported in framework
  - Difficult to get Control Names from IE for dynamically generated inputs
- Demo

Trying out the Tools: Selenium

- What worked?
  - Most of the functionality in the app
  - Writing tests in wiki (custom functionality)
  - Exercising the Rule Builder!
- What didn't?
  - Needed to create custom Selenium tasks
  - Tests won't run well without "babysitting"
  - Problems hooking up to CruiseControl builds
  - Hassles working around "same-source" browser security
- Demo

In Summary

- Automated functional testing is important
- Web 2.0 currently challenges most of the test frameworks in some way
- Be prepared to compromise on your test strategy or application design
  - Change design to be "testable"
  - Test some features manually
  - Put stubs in to work around parts you can't test
- Plan to do some level of customization if you're serious about making automated testing work