Home-Brew Test Automation

bret@pettichord.com

www.pettichord.com

Test Automation Conference, San Francisco, March 2003

XP and Automated Testing

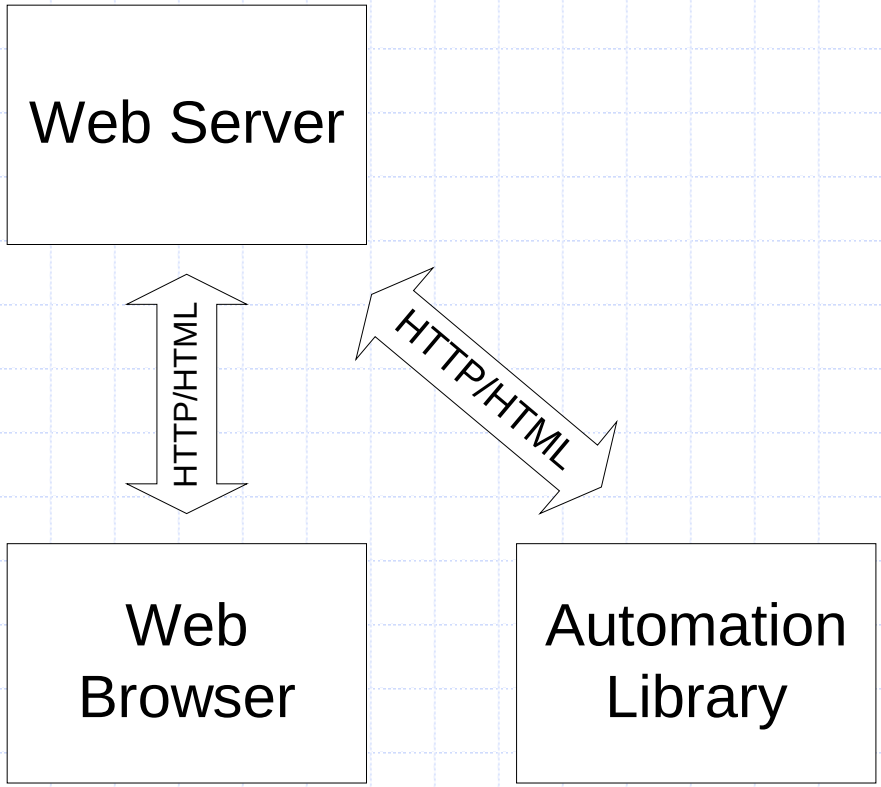
- Programmers write automated unit tests.
- Acceptance tests must also be automated.
- Programmers and testers *work together* on acceptance tests.

XP Teams Rarely Use Commercial GUI Test Tools

- Price
  - Too expensive to give copies to everyone on the team
- Tool Languages
  - Need something that everyone understands
  - Often weak and limited: “Heinous”

They would rather build their own testing frameworks

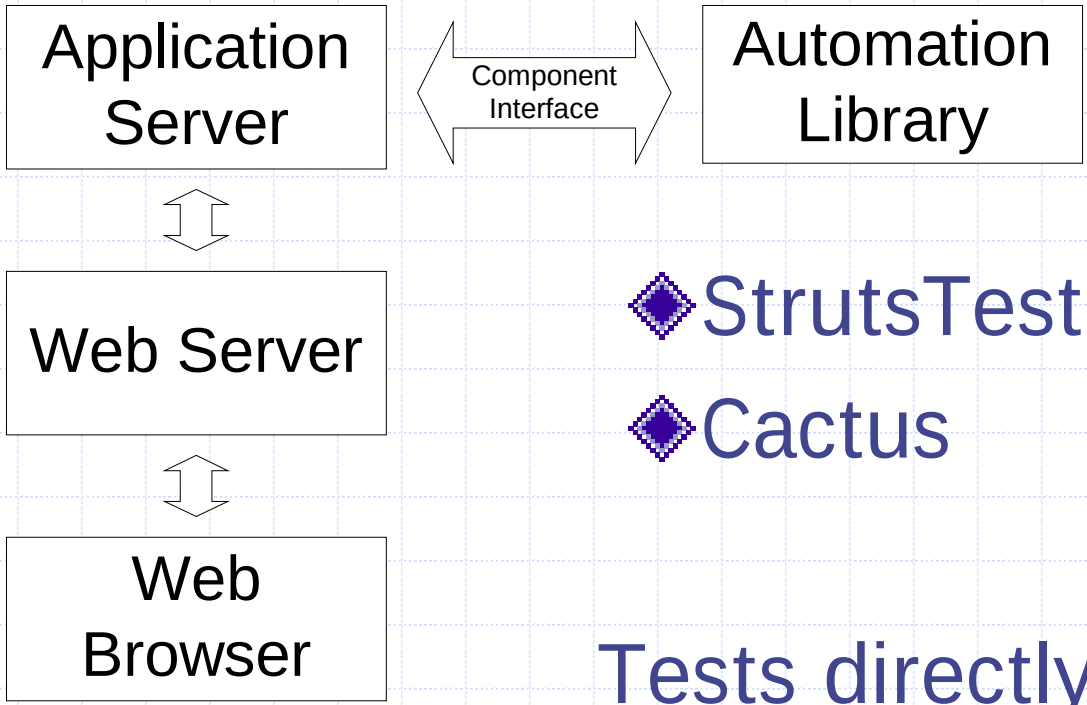
Example: Browser Simulation



- HTTP-Unit
  - Java
  - Perl
  - Ruby
  - Python

Tests execute directly against the web server

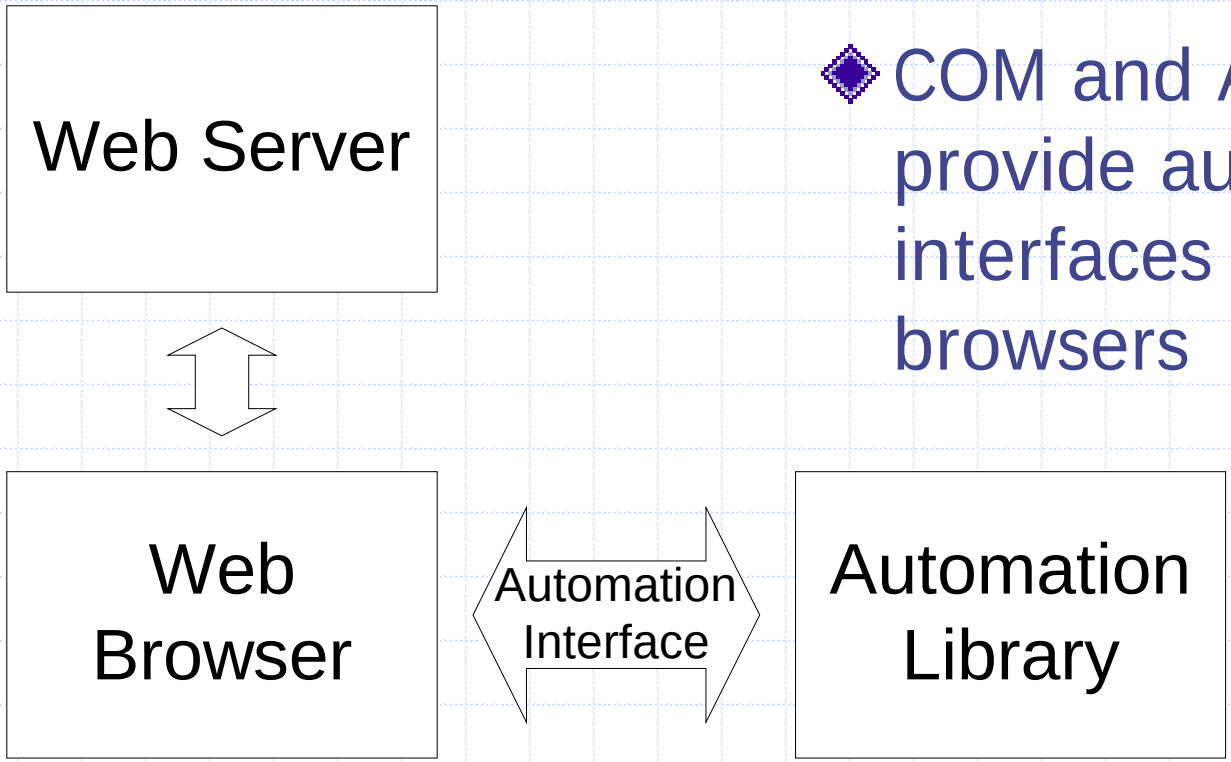
Example: Component Testing



- StrutsTestCase
- Cactus

Tests directly test the business logic, not the presentation

Example: Browser Automation



- COM and Applescript provide automation interfaces to browsers

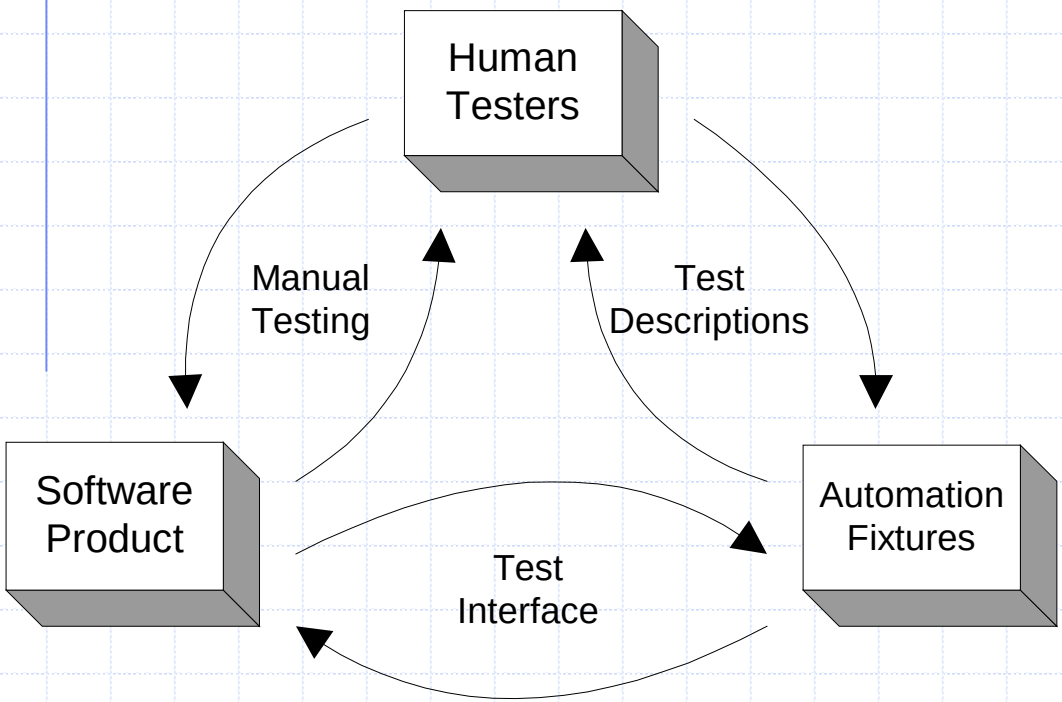
Tests execute against a browser

Home-Brew Strategies

1. Extending unit testing
2. Adapting the product for testing
   - Thin GUI
   - Test Interfaces
3. Building your own tool

These strategies are more effective combined

Test Interaction Model



- Customizing Testing Frameworks
  - For tester usability
  - For product compatibility

Agenda

- Beyond Unit Testing
- Scripting Languages
- Interface Drivers
- Building Your Own Tool
- Test Description Languages
- Coaching Tests

Beyond Unit Testing

- XP has had a major impact on unit testing
  - JUnit has been ported to dozens of languages and extended for dozens of frameworks
  - Developers all over are actually writing unit tests
- XP has now set its sights on tools for acceptance testing

Unit Testing

- Units are functions, methods or small bits of code
- Unit tests are in the same language as the code being tested
- Unit tests are written by the same programmers as the code
- A test harness or framework collects tests into suites and allows them to be run as a group

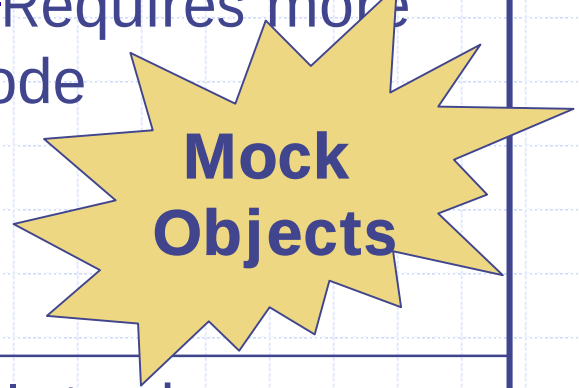
Test-Driven Development

- Developers write unit tests *before* coding.
  - Motivates coding
  - Improves design (reducing coupling and improving cohesion)
  - Provides regression tests
  - Supports *refactoring*
- Is a design approach as much as a testing approach
  - Specification by example

Unit Integration Testing

- How to test units that depend on other units?

Unit isolation testing *Test each unit in isolation*	*Create stubs and drivers objects for external units*	Requires more code
Unit integration testing *Test units in context*	*Call external units*	Introduces dependencies. Test suites take longer to run



Home-Brew Ingredients

1. Test Harness
   - Collecting tests so they can be executed together
2. Language
   - Creating the automation fixtures
   - Providing a language for describing tests
3. Product Interface Driver
   - Giving access to the software product

Three Kinds of Languages

- System Programming Languages
  - Optimized for *performance*.
  - What your programmers are probably using.
  - **C, C++, Java, C#**
- Scripting Languages
  - Optimized for *ease of use* and *high productivity*.
  - Command interpreters facilitate learning and exploration.
  - **Visual Basic, Tcl, Perl, Python, Ruby, Rexx, JavaScript, VBScript, Lua**
- Data Presentation Languages
  - Optimized for *readability* and *structure*.
  - No logic
  - **HTML, Spreadsheets, XML**

Scripting Languages for Testing

- **Visual Basic & VB Script**
  - Popular
  - Integrate well with **Microsoft** technologies
- **Tcl**
  - Well-established
  - Compact
  - Popular with **embedded** systems
- **Perl**
  - Vast **libraries**
  - Integrates well with Unix/Linux
- **Python**
  - Concise support for object-oriented programming
  - Integrates well with **Java** (Jython)
- **Ruby**
  - Everything's an object
  - Principle of **least surprise**

- Tcl, Python and Ruby have *command interpreters*
- All but VB are *open source*
- All are well supported

The best language is the one your team knows.

Language Choices

What should you write tests in?

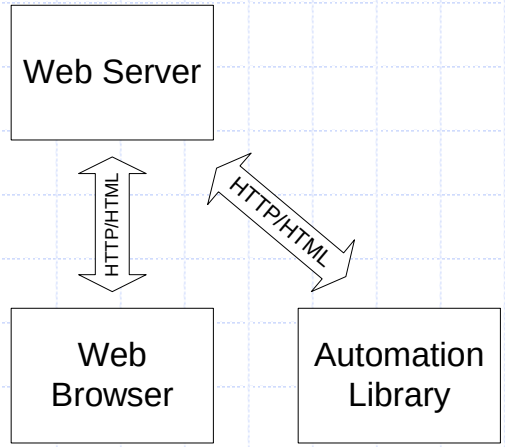
1. **System programming language**
   - Reuse unit test harness
   - May result in lower productivity
2. **Scripting language**
   - Requires interface to product
   - Allows most kinds of tests
3. **Data presentation language**
   - Requires support code in a scripting or system language
   - May improve understandability

- Options
  - 1 only
  - 2 only
  - 1 and 3
  - 2 and 3

Interface Drivers

- How do your tests access the product?
  - Simulation
    - Access the server in the same way as the client or browser
  - Automation
    - Automate the client or browser using automation interfaces.
  - Thin GUI
    - Access the business logic directly
    - Keep the presentation layer thin

Open Source Browser Simulation



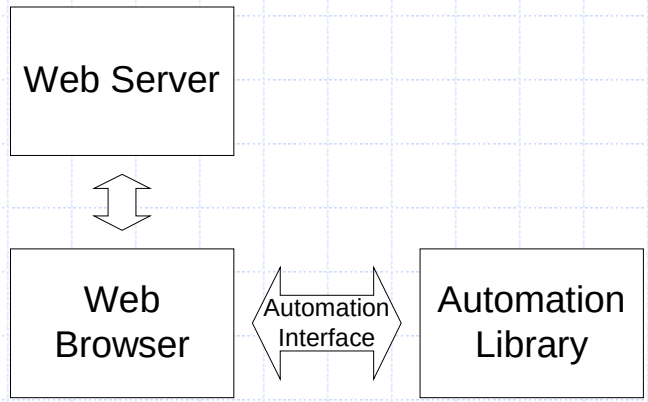
- **HttpUnit**, Russell Gold
  - Browser simulation in Java. Popular & well-extended.
  - http://www.httpunit.org/
- **jWebUnit**, Thoughtworks
  - A refinement on HttpUnit and FIT. Java-based.
  - http://jwebunit.sourceforge.net
- **Canoo WebTest**, Canoo Engineering AG
  - Java-based browser simulation with XML.
  - http://webtest.canoo.com
- **HTTP::WebTest**, Richard Anderson and Ilya Martynov
  - Browser simulation in Perl.
  - http://search.cpan.org/author/ILYAM/HTTP-WebTest-2.00/
- **WebUnit**, Yuichi Takahashi
  - Browser simulation in Ruby.
  - http://www.xpenguin.biz/download/webunit/index-en.html
- **Puffin**, Keyton Weissinger
  - Browser simulation in Python and XML.
  - http://www.puffinhome.org/

More Tools

- http://www.junit.org/news/extension/web/index.htm

Open Source Browser Automation

- **Cliecontroller**, Chris Morris
  - IE and .Net Windows forms automation in Ruby
  - http://www.rubygarden.org/ruby?RubyForTesting
- **Samie**, Henry Wasserman
  - IE Browser automation in Perl
  - http://samie.sourceforge.net/



Open Source Java GUI Drivers

- **Marathon**, Jeremy Stell Smith et al, Thoughtworks
  - Java Swing GUI driver using Python scripts. Includes a recorder.
  - http://marathonman.sourceforge.net/
- **Abbot**, Timothy Wall
  - Java GUI driver and recorder using XML scripts.
  - http://abbot.sourceforge.net/
- **Pounder**, Matthew Pekar
  - Java GUI driver and recorder.
  - http://pounder.sourceforge.net/
- *More Tools*
  - http://www.junit.org/news/extension/gui/index.htm
  - http://www.superlinksoftware.com/cgi-bin/jugwiki.pl?TestingGUIs

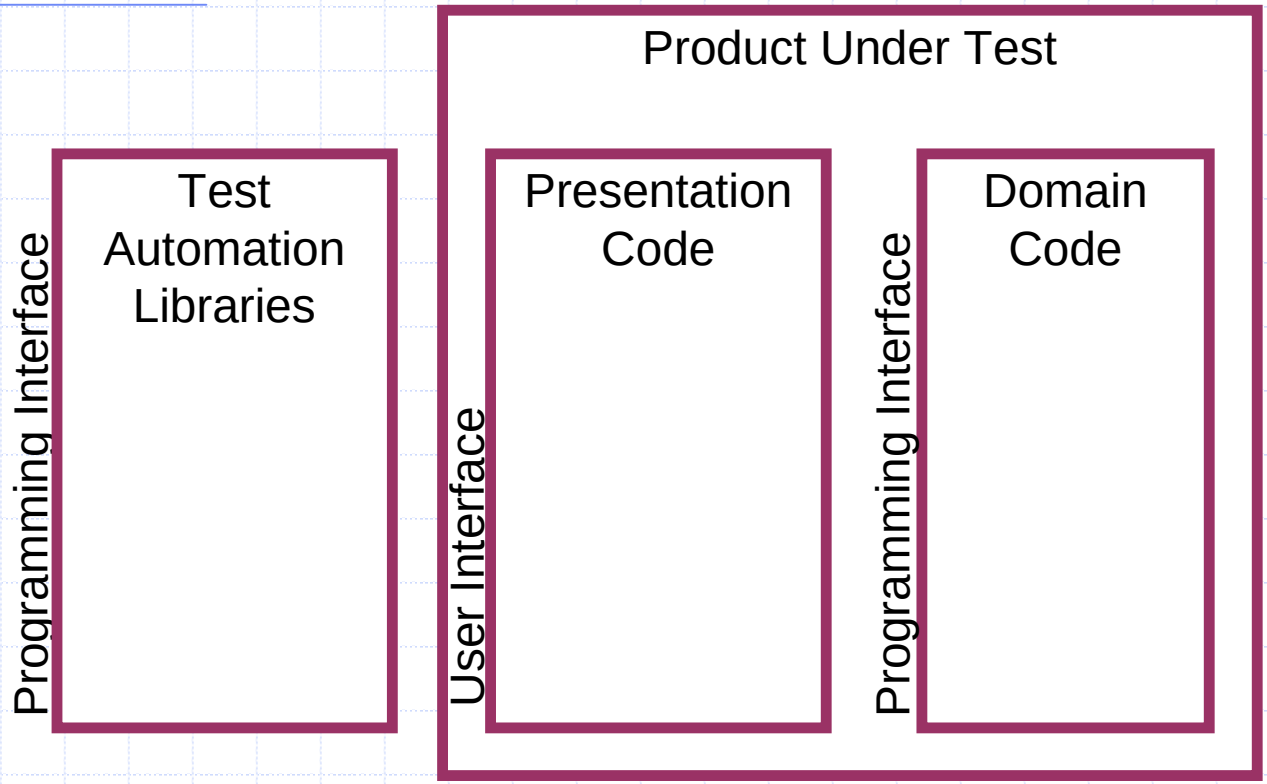
Other Open Source Test Libraries

- **Expect**, Don Libes
  - Command line driver in TCL. Long-established.
  - http://expect.nist.gov/
- **Win32::GuiTest**, Ernesto Guisado
  - Windows GUI driver in Perl. Popular.
  - http://search.cpan.org/author/ERNGUI/Win32-GuiTest-1.3/
- **Android**, Larry Smith
  - Unix/Linux GUI driver in Tcl.
  - http://www.wildopensource.com/larry-projects/android.html
- **JUnitPerf**, Mike Clark
  - Performance measurement for existing JUnit-based tests.
  - http://www.clarkware.com/software/JUnitPerf.html
- **OpenSTA**, Cyrano
  - Web performance testing driver and recorder. Uses a "proprietary" scripting language.
  - http://opensta.org/
- **Avignon**, NOLA Comp Services
  - XML integration with JUnit.
  - http://www.nolacom.com/avignon/
- **Framework for Integrated Test** (FIT), Ward Cunningham
  - Parses tests in HTML. Supports multiple languages.
  - http://fit.c2.com

Building Your Own Tool

- This is getting easier
  - Automation interfaces are now standard
  - You only have to support one technology
  - Use and extend open source tools
- Approaches
  - Capture/Replay
  - Scripting
  - Data-driven
- Maintenance is still a concern

Test Interfaces



- Which interface will be easier to test?

Test Interfaces

- Interfaces may be provided specifically for testing.
  - Excel
  - Xconq (a game!)

Any interface is easier to automate than a GUI.

- Existing interfaces may be able to support significant testing.
  - InstallShield
  - Autocad
  - Interleaf
  - Tivoli

  Web Services Interfaces are ideal!

Build a Tool or Adapt the Product?

- Depends on how difficult each is for your situation
  - Getting started
  - Maintenance
- Assess and decide as a team
- Consider doing some of both

Test Description Languages

What is the best test description language for expressing tests?

- Tables
  - Readable to more people
  - Require fixtures & parsers
- Scripts
  - Allow variables
  - Allow looping
  - Require test interfaces

```
Timeclock> start 'misc'
Timeclock> pause
Timeclock> start 'stqe'
Timeclock> jobs
misc, started 02002/08/30 4:32 PM, is paused.
stqe, started 02002/08/30 4:33 PM, is recording time.
```

Example courtesy Brian Marick

FIT tests

- Scrape tests from HTML docs
- Keep requirements & tests together
- Check automatically
- Browse results online
- Understandable by everyone

Division shall work with positive and negative numbers.

eg.Division		
numerator	denominator	quotient()
1000	10	100.0000
-1000	10	-100.0000
1000	7	142.8571 *expected* 142.85715 *actual*
1000	.001	1000000 *expected* 999999.94 *actual*
4195835	3145729	1.3338196

Coaching Tests

- Use tests to drive development
- Tests provide:
  - Goals and guidance
  - Instant feedback
  - Progress measurement
  - Health check of the project
- Tests are specified in a format:
  - Clear – so any one can understand
  - Specific – so it can be executed

eg.ArithmeticFixture					
x	y	x + y	x - y	x * y	x / y
200	300	500	-100	60000	0
400		420	380	8000	20

Specification By Example

- Defining requirements is difficult
  - Often ambiguous, rarely complete
  - Often harder to specify requirements than design
  - Often become out of date
- Defining them by example is incremental, and easier
  - Concrete
  - Run them to see if they are current
- Examples...
  - Ground concepts
  - Define expectations
  - Demonstrate progress & conformance

Test Automation in the Silo

- Traditionally test automators have worked in a separate space from developers
  - The code is separate
  - The teams are separate
  - *Ex post facto* GUI automation
- Reasons for change
  - Tool/application compatibility (testability)
  - Maintenance when GUI changes
  - Testing needs to be everyone's concern

Home-Brew Test Automation

bret@pettichord.com

www.pettichord.com

Test Automation Conference, San Francisco, March 2003