Test Automation

SQDG
October 2008

Janet Gregory
DragonFire Inc.

Topics

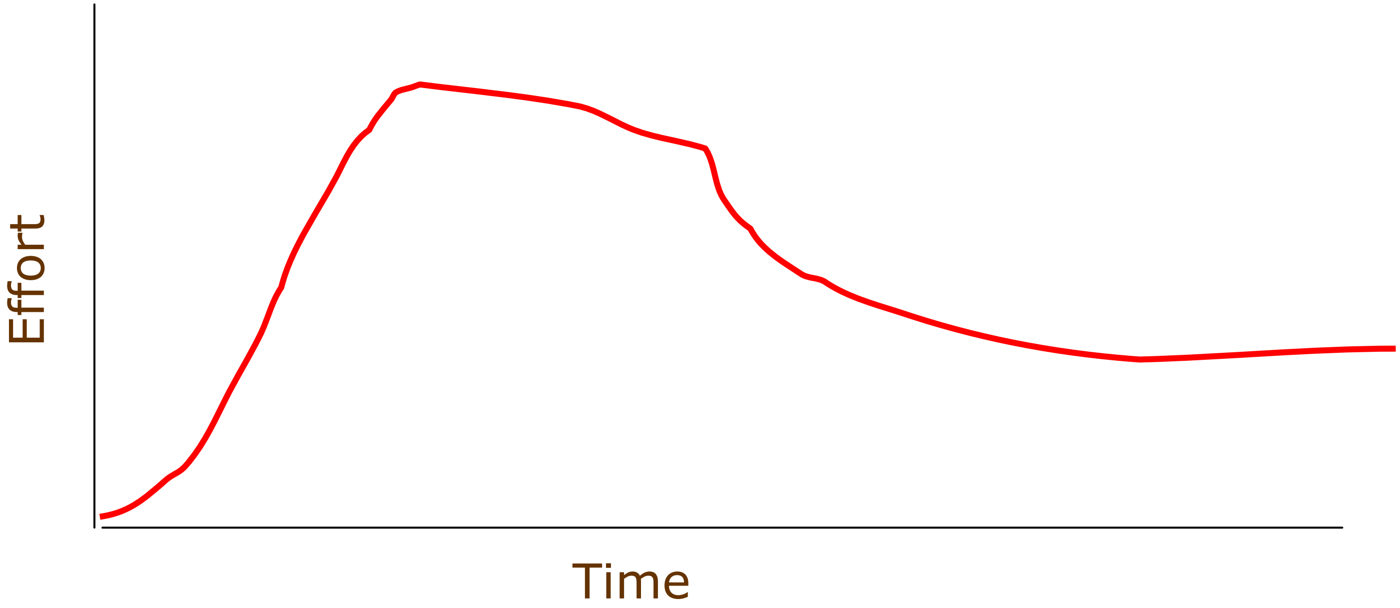
- Why Do We Automate?
- Barriers to Automation
- Selling to Management
- Evaluating Tools
- Getting Started
- Wrap-Up
- References

Why Do We Automate?

- Manual testing takes too long
- Manual tests are error prone
- Provide feedback, early and often
- Free people to do their best work
- Tests provide documentation
- Automation can be a good return on investment

Barriers to Test Automation c

- Test automation is hard!
- Requires a big investment
  - Time, money
- Payoff may not be immediate



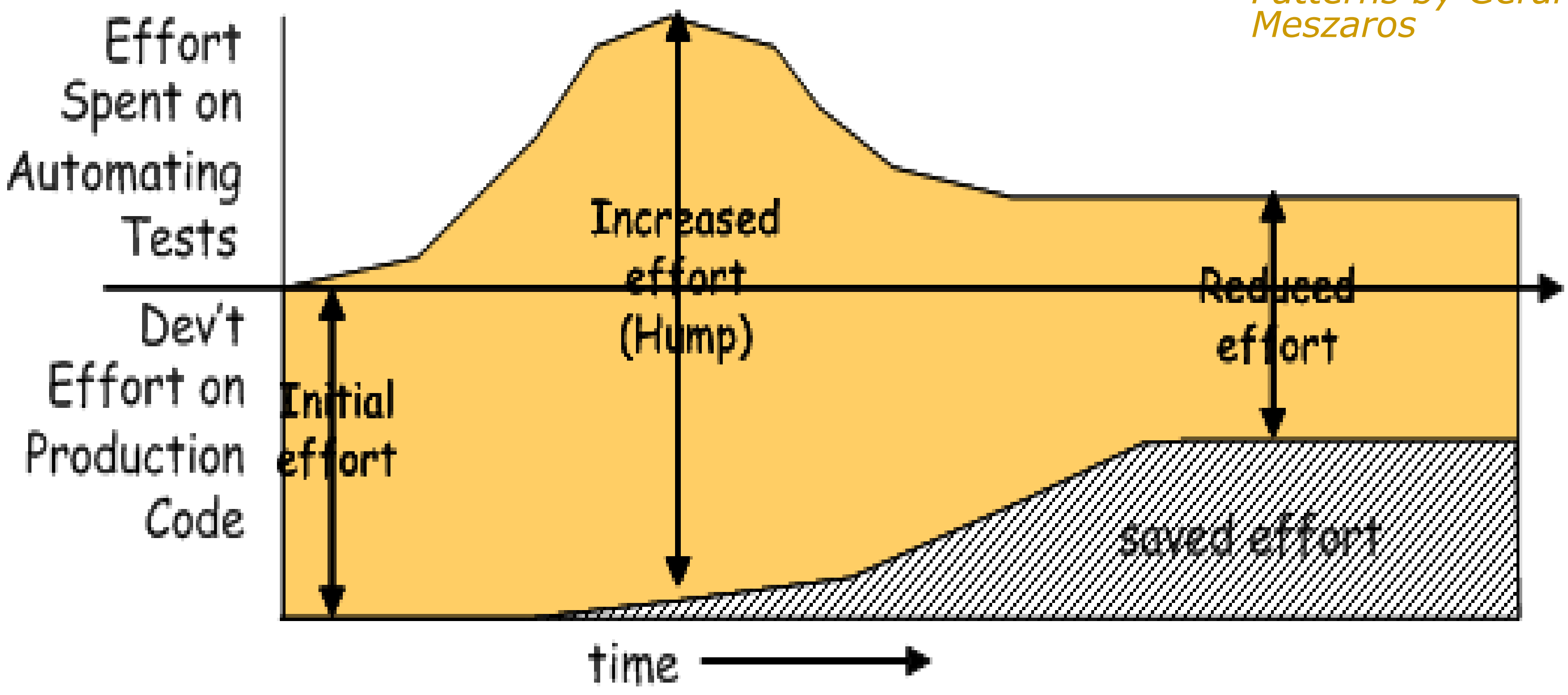
Barriers to Test Automation 2

- FEAR!
- Testers lack programming skills
- Programmers lack testing skills
- Rapidly changing code
- GUI design in flux
- Application not designed for testability
- Constraints - $$$

Selling to Management

- Acknowledge ROI takes time
- Not a “Silver Bullet”!

Source: xUnit Test Patterns by Gerard Meszaros



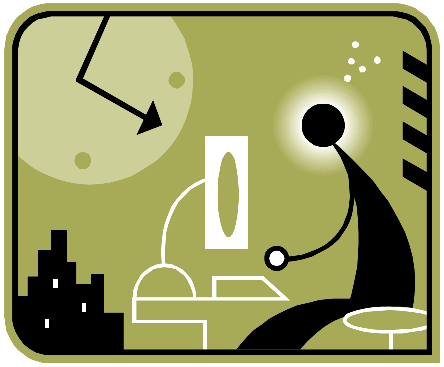
What Affects ROI

- Good test practices increase ROI
  - Simple, well-designed, refactored tests
  - Test resources improve over time

- Poor test practices reduce ROI
  - Tests are hard to understand
  - Tests are hard to maintain

Find Time for Evaluating

1. Get commitment for time to research
2. Budget time
   - Individual or group
3. Determine your requirements
4. Do some basic research
5. Compile a list of tools to consider



Determine Requirements 1

- Understand purpose
- What do you want to test?
- Criteria specific to type of tool
  - E.g.. SSL support for web testing tool
  - Link to example
- Reporting needs
- Integration with existing tools, infrastructure, hardware, software
- Test management needs
- Constraints - $$

Determine Requirements 2

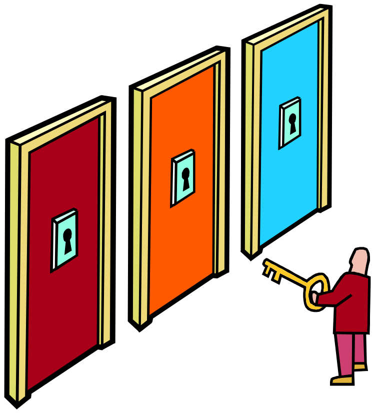
Inventory available skills

- Who will be automating and maintaining?
- What is the team’s skill set

Inventory tool needs

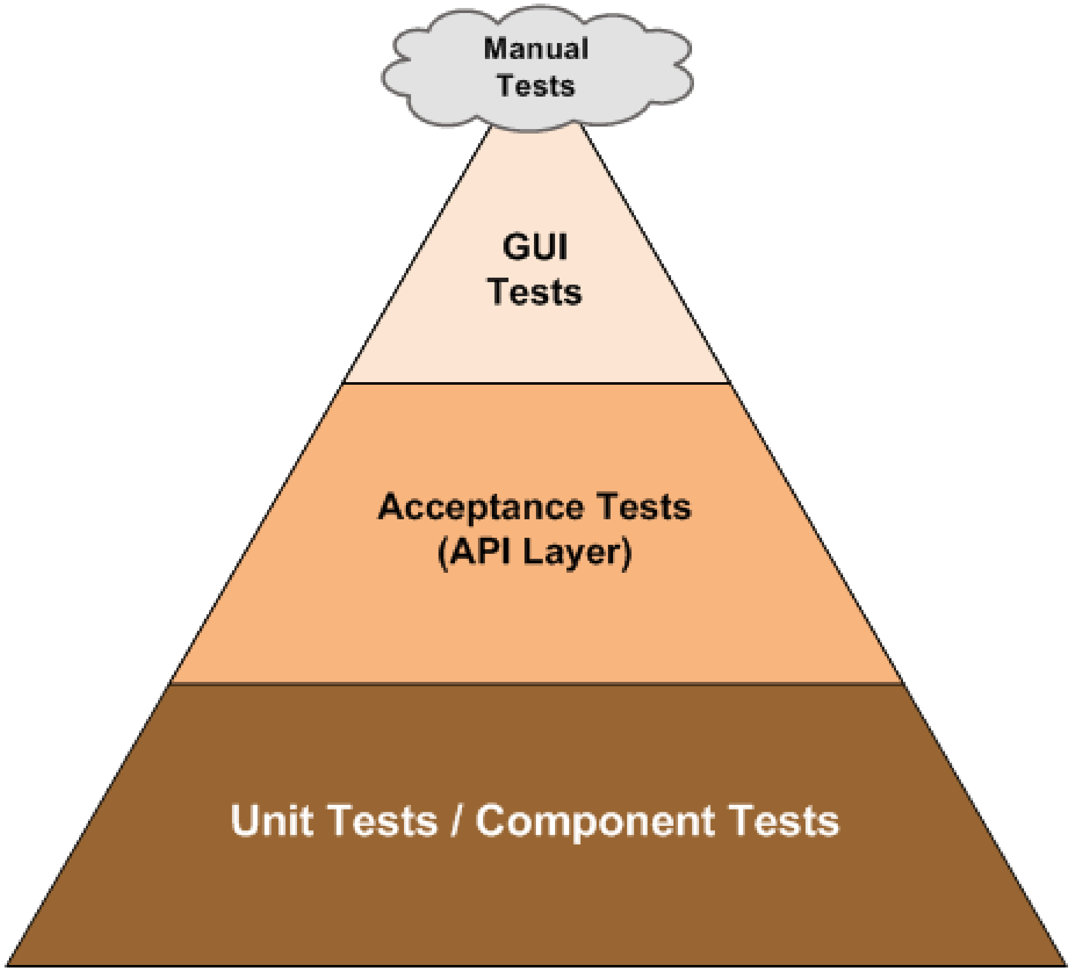
- Can non-programmers specify test cases?
- Programmer-friendly test tools?
- Tools to allow collaboration?
- Tools not directly related to testing?
- What is the most urgent need?

What are you automating?



- Unit tests
  - Best ROI
  - Big hump of pain
- Functional tests
  - Don't involve UI
  - May need programmer or specialist support
- GUI tests
  - Simple smoke tests?
  - Need more robust, data-driven, action-driven tests?
  - Can be expensive

Automation Test Pyramid



Evaluating Tools

- Home Grown, Open Source, or Vendor?
- Where to look
- Finding time to try tools
- Does the tool fit requirements?
- Judging the ROI

Match Tool to Requirements

- Make list of prospects
- Narrow down to one or two to try
- Go through demo or tutorial if available
- Pair if possible
- Do a spike: Try a simple but representative scenario (throw-away)
- Check results against requirements
- Pros and cons

Vendor Tools - Pros

- Perceived as a safe bet
- Likely to provide training, support, manuals
- Initial ramp-up may be easier for non-programmer
- May have complementary tools
- Some are robust, feature-rich
- Your company may already own one
    - But this shouldn’t be a primary reason!
- If you have an expertise in a vendor tool
- Tool only used by portion of team, or separate test team

Vendor Tools - Cons

- Tend to be programmer-unfriendly
  - Proprietary scripting language
  - Or just a language your programmers don't use
- Tend to be heavyweight
  - Tests can be brittle, expensive to maintain
  - Especially capture-playback

Open Source Tools - Pros

- Created and maintained by people facing your same testing challenges
- Tend to be lightweight
- Usually appeal to programmers as well as testers
- Easily customized, since open source
  - if you have the resources
- Price is right
  - but remember purchase price is not main cost

Open Source Tools - Cons

- Training may be an issue
  - Often available at conferences, seminars, user groups
  - Some open source tools have tutorials, good manuals

- Look for active developer and user community
  - New features added often
  - Support available through mailing list or bug tracking system

Grow Your Own?

Pros

- Unique needs
- Can be customized

Cons

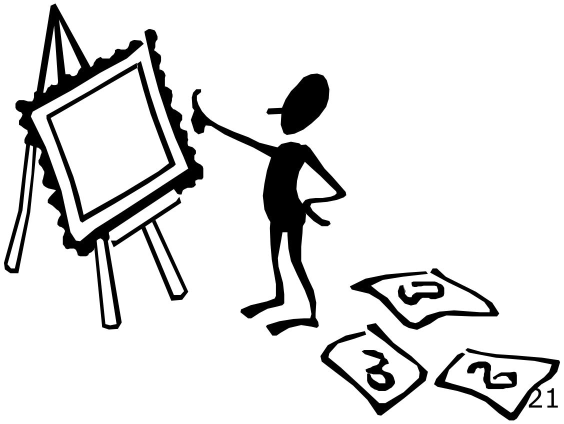
- Programming expertise needed to write test framework
- Consider the time to write
- Consider the time to maintain the tool

Presenting Tool Evaluation

- Recommendation - reasons
- Costs:
  - Training
  - Purchase, licences
  - Resources – people, computers
- Payback:
  - Time
  - Quality

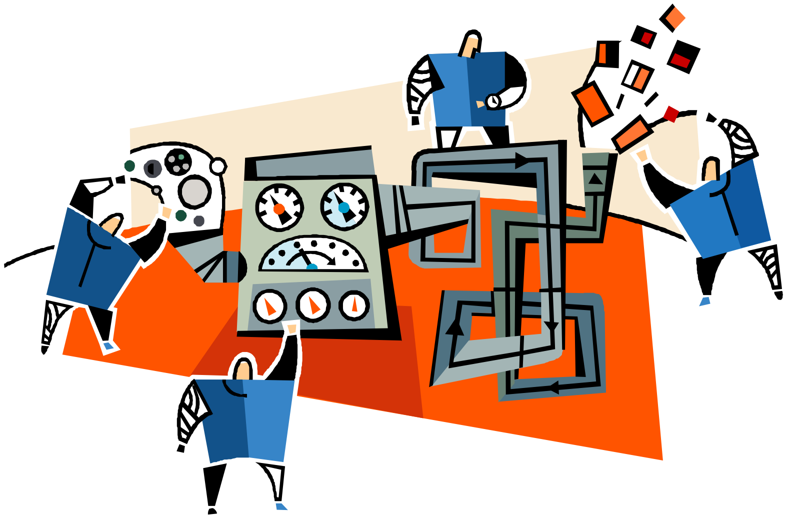
Judging the ROI

- Effect on productivity, velocity, risk
  - What will it allow you to do that you can’t now, long term
- Commit to trying top choice for short period of time
  - But long enough to gain competency
- Do retrospective, what worked and what didn’t?



Getting Started

- How much should we automate?
- What can we automate?
- What shouldn't we automate?
- What might be hard to automate?



What Can We Automate?

Automate tedious or repetitive tasks

- Testing-related or otherwise
- Continuous integration, builds
- Deployments
- Checking for updates
- Parsing, comparing output
- Automating tasks for business
- Set up for exploratory testing

What Shouldn't We Automate?

- Look and feel
- Usability
- Exploratory testing
  - Use automation to facilitate (set-up)

Note: If regression tests aren't automated,
….. There is no time for these others

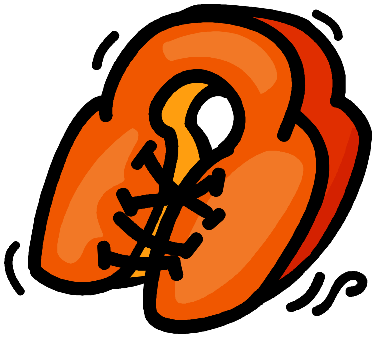
What To Consider Carefully

Automating end to end tests

- Manual end to end testing can be critical
- What’s the risk?
  - Example: safety critical systems
- Don't go overboard, automating every path
- Expensive to maintain

Push testing down as low as you can

  - Highest ROI in unit tests



What Else Shouldn't We Automate?

- Tests that will never fail
  - Something so obvious nobody will forget and break
  - But if high risk or safety critical, automate anyway!
- Tests covered elsewhere, eg, unit tests
- ROI not there, eg, one-off tests

What Might Be Hard to Automate?

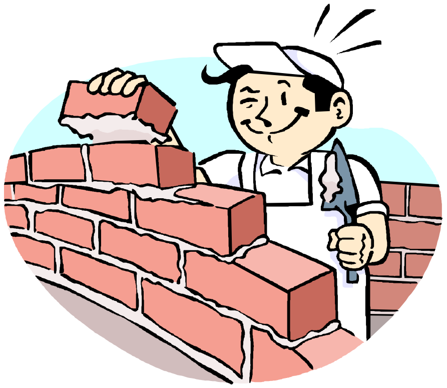
- Legacy code
  - GUI, business logic, database, IO layers intertwined
  - Hard to write unit, functional tests
  - Or 'strangle' the legacy code
  - But this is beyond our scope right now!
- New code not designed for testability
- May have to test through GUI or API
  - But team needs to solve testability problem
  - And find a way to write unit level tests

Where Do You Start?

- What's the greatest area of pain?
- Master one tool, then see what else you need
- Multiple tools for multiple needs
- Write simple tests
  - One condition per developer test
  - One business rule per customer test
- Tools can be as simple as a spreadsheet
  - Retrieve data and perform same calculation as app

Small Chunks

- Ask business to prioritize critical areas
- Write simple smoke tests in priority order
- Commit to automating regression tests for new features
- Whatever fits your situation!



“Whole Team” Approach

- In agile, the entire development team, not only testers or QA, is responsible for testing and quality
- Automation is a team responsibility
- Anyone can sign up for test automation tasks
- Programmers may automate tests specified by testers
- Programmers and testers may collaborate to automate tests
- Team designs code for testability

Remember….

Automation is hard

Payback takes time

Tackle it in small chunks

Don’t tie yourself to one tool

Not every team is the same!

Where to Look for Tools

- http://www.softwareqatest.com/qattls1.html
- http://www.testingfaqs.org
- http://www.opensourcetesting.org
- groups.yahoo.com/group/agile-testing

Coming in January 2009!

Agile Testing: A Practical Guide for Testers and Agile Teams

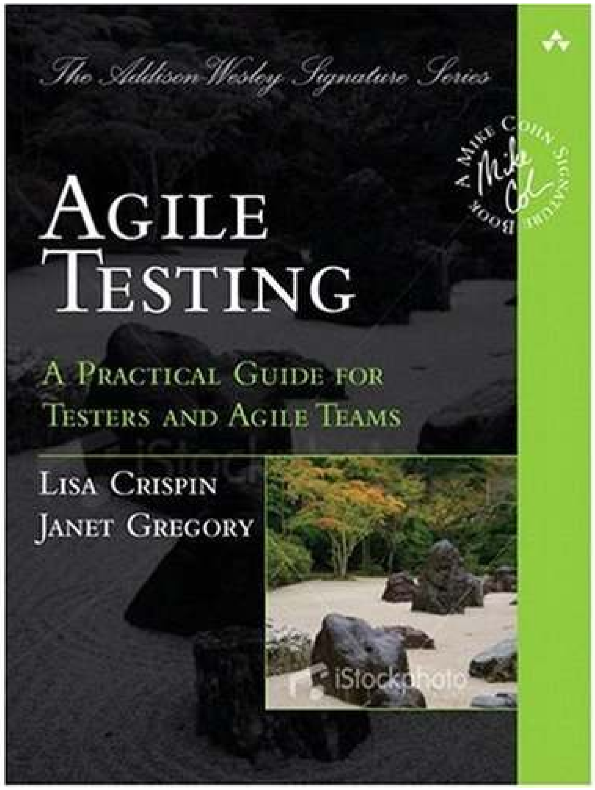
By Janet Gregory and Lisa Crispin

Available on

- Amazon.com
- Amazon.ca

www.agiletester.ca

www.janetgregory.ca


The Addison-Wesley Signature Series
A Mike Cohn Signature Book
Agile Testing
A Practical Guide for Testers and Agile Teams
Lisa Crispin
Janet Gregory
iStockphoto