The Secrets of Faking a Test Project

Jonathan Kohl
jonathan@kohl.ca
www.kohl.ca

kohl concepts

The Secrets of Faking a Test Project

This presentation was adapted from James Bach's "Guide to Faking a Testing Project"

Portions of this presentation were created by James Bach and are used with his permission.

Content by Jonathan Kohl and James Bach, 2007.

The Case of the Faking Tester

Faking It

- Intentional faking
  - Am I deliberately fooling others?
  - Charlatan
- Unintentional faking
  - Am I accidentally fooling others and myself?
  - Naïve, inexperienced

Deliberately Faking It

A charlatan is:

- A person who pretends to more knowledge or skill than he or she possesses; quack.
  - charlatan. (n.d.). Dictionary.com Unabridged (v 1.1).
- A person who makes elaborate, fraudulent, and often voluble claims to skill or knowledge; a quack or fraud.
  - charlatan. (n.d.). The American Heritage® Dictionary of the English Language, Fourth Edition.

The Challenge

You want to release bad software, but you have to make it look as if you really tried to test it well…

Here's how you could fake it!

Establish Credibility

Pick a field others have little expertise in

- Software testing is an ideal field for charlatans
  - Less chance of people knowing what you are really doing
- Difficult to verify your actions
- **Key Idea:** Provide easy answers to their concerns.

Establish Credibility

Use an outside body to gain “qualifications”:

- Volunteer at a local user group, certification exam or academic institution
  - No one needs to know if you were a presenter or set up the chairs
  - The organization’s name is what is important
- Make up qualifications
  - For the least effort, just invent them

Establish Credibility

Visibility == Credibility

- Provide metrics reports
  - Print out and post charts and graphs of counts:
    - Bugs reported
    - Number of test cases
    - Lines of code in automated test cases
- Spend as much money as possible
- Walk around looking busy
  - Be sure to have a concerned frown, and a stack of papers

Establish Credibility

Be a Knowledge Parasite

- Always look for anything that might impress managers
  - Ingest and regurgitate buzzwords
  - Plagiarize
    - Internal sources
    - External sources
  - Take credit for other's work

2007. Aaron West. *Knowledge Parasites*.
http://westaj.livejournal.com/2653.html

Self-Presentation

- Call yourself an “Engineer” and talk a lot about “engineering discipline”
- Or call yourself “Quality Assurance” and talk a lot about “best practices” and “process maturity”
- Of course, you also *declare yourself an expert*. It’s easy!
- Process maturity lets you defend a slow and expensive process by featuring as a virtue its very slowness and expensiveness!

Faking it at Work

You could just lie, of course…

Testing is hard to supervise

- Your boss probably doesn't watch you closely.
- Say you tested it, but spend most of your time playing Spider Solitaire, instead.
- Report a few minor bugs to keep the heat off.
- But what if you were going to be audited?

Test Case and Pass Rate Metrics!

- Test cases are just containers, easily manipulated.
- Make your tests easy to pass, and all similar.
- It should not be difficult to produce thousands of them, just by using copy and paste.
- You need more than 1000 tests. Make the pass rate climb slowly.
- If necessary, restrict the oracles so that more tests pass.
- **Golden Rule:** Make the graphs fit expectations.

A Basic Strategy

- Behave Conventionally (don't worry, conventional testing wisdom is empty)
- Squander Energy (so that you can't test)
- Focus Narrowly (don't make eye contact with bugs)
- Deflect Scrutiny (don't avoid it, co-opt it)
- Minimize Humanity (humans are too good at testing)
- Blame Complexity, Ambiguity and Volatility (argue that no one can cope with these things)

Behave Conventionally

Testing folklore is popular:

- Detailed scripted test procedures with specific expected results
  - Executed after each build by unskilled testers!
  - This is the gold standard of testing fraud.
  - Real expected results are impossible to document fully, so it's hard for people to accuse you of doing too little.

Behave Conventionally

- Most managers think any intellectual process can and should be written down, so you are going with the flow.
- Create simple function tests so that they are unlikely to find problems even the first time through.
- DANGER: Testers may accidentally find bugs because they don't follow the scripts precisely.
- SOLUTION: Accuse them of lacking discipline and maturity.

Squander Energy

- Do little actual testing, but lots of process policing
- Follow a highly-visible, ritualized process that is heavy on form, light on function
  - Extra points for a fad process that sounds impressive. *(“Agile” anyone?)*
- Keep occupied by wandering around
- Spend most of your time creating and maintaining test plans and test cases

Squander Energy

- Let your time be dominated by creating and maintaining documents
- At least 20% of your time should be spent in paperwork, with little time for testing
- The other 80% needs to be dominated by meetings
  - Be sure to make a lot of pomp and ceremony about how hard the testers are working at any meeting.

Thick Official Documents!

- Thickness discourages scrutiny.
- Templates give appearance of analysis.
  - *Use downloaded templates for extra points.*
- IEEE 829 is a faker's best friend!
- Contrast your handsome docs to the crude ones you receive.
- Make a big show of keeping them up to date.
- The time you spend on these documents will prevent you from testing.
- Consider computer generated docs! Cool!!

Thick Official Documents!

Be sure to include in every document:

- Title page
- Approvals page
- Version history
- Table of contents
- Introduction to the project
- Purpose of the document
- Document reference list
- Acronyms and definitions
- Chatty tutorial text to discourage review
- LOTS OF FORMATTING

Little useful content, but plenty of excuses for including it.

Focus Narrowly

- If you squander enough energy, you won't have any choice
- Only test based on what the requirements docs say, and what the programmers say.
- Only test from pre-recorded test scripts, run at project end.
  - You can do this in a "test-first" style as well.

Focus Narrowly

- Refuse to expand your focus from requirements-verification testing
  - Blame a lack of staff
  - Cite something you might have read in a book or call upon other authorities
  - Use “tradition” as an excuse
  - Play the “tester independence” card
  - Any other kind of testing is “someone else’s job”
  - Say "That should be in the spec"
    - (ie. make it someone else's problem to define your focus)

Deflect Scrutiny

Appeal to official sounding "authorities"

- Conventional testing wisdom
- Experience
- Intuition
- "Best Practices"
- Use flavor of the month process dogma
  - "Agile" or "Lean" anyone?
    - *"We're Agile, so we have to do it this way!"*

Deflect Scrutiny

Blame:

- The programmers (they are writing buggy code)
- The requirements (they are lacking, not arriving in time)
- The process (that's the *real* problem)
- Management (for not hiring more staff)
- Competent people (they could be a threat to you)

Minimize Humanity

- Use low skilled testers
- Punish anyone who goes outside the process
- Discourage productive, skilled testers
  - If you're lucky they will just quit
- Constantly blame lack of success on a lack of testing tools
  - Especially expensive ones you can't afford

Expensive GUI Test Automation!

1. Purchase an expensive GUI test execution tool.
   - Take as much time as possible with the purchase decision so the purchase itself is the goal and reward
   - There should be at least three documents: one describing tool need, a decision matrix, and one full of the marketing material from the tools you reviewed.
2. Define a lot of paper test procedures.
3. Hire an automation team to automate each one.
4. Build a comprehensive test library and framework.
5. Keep fixing it.
6. BONUS: Test Management Software