

Agile Test Automation

James Bach, Satisfice, Inc.

James@satisfice.com

www.satisfice.com

Examples of Agile Automation

- **CD test system** (300% improvement in CD package testing throughput in two weeks)
- **Auction Testing** (programmatically generated test; web site smoke test; server configuration checker; bid activity report generation; delivered and debugged in about 8 days)
- **Embedded Web-Server in Printer** (three test programs in 90 minutes)
- **Email Client Testing** (mail database dumper; installation checker; email blaster)
- **10 Minutes, One Tester** (InCtrl5; Spy++)

Typical Automation Formula

1. Purchase an expensive GUI test execution tool. (see Rational, Mercury, Compuware, etc.)
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.

**This can work if your product is
very easy to test and it doesn't change much.
Does that describe your product?**

Snap Out of It!

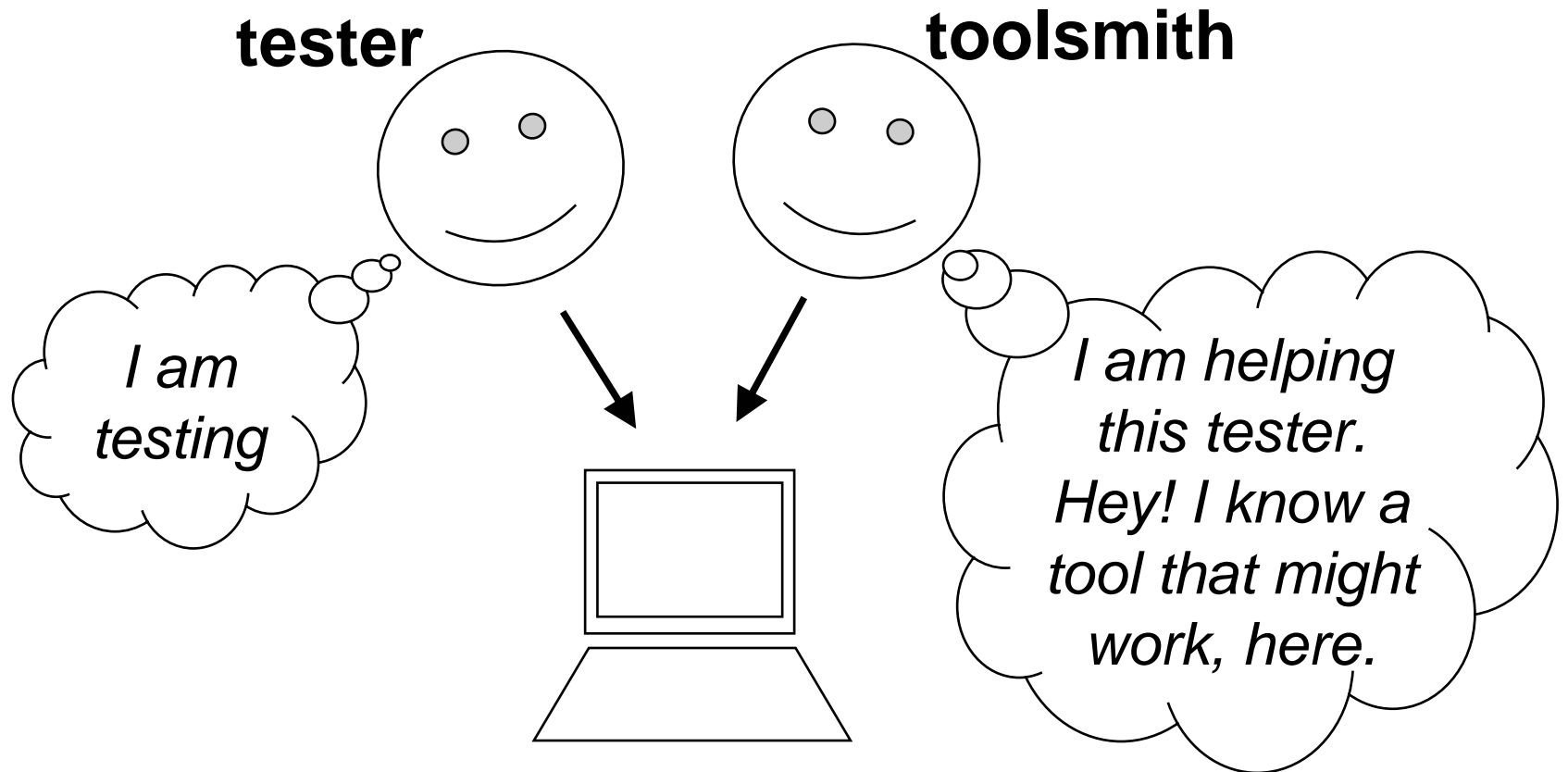
Why think so *narrowly* about test automation?

Consider thinking of test automation as...

any use of tools to support testing

Agile test automation is the application of
agile development principles
to the test automation problem.

This is what it looks like...



Principles of Agile Test Automation

- Test automation means tool support for all aspects of a test project, not just test execution.
- Test automation progresses when supported by dedicated programmers (toolsmiths).
- Toolsmiths are directed by testers.
- Test toolsmiths gather and apply a wide variety of tools to support testing.
- Test toolsmiths advocate for testability features and produce tools that exploit those features.
- Test automation is organized to fulfill short term goals.
- Long term test automation tasks are avoided in the absence of specific approval based on a compelling business case.

To Clarify...

- **“Short Term”** 40 work hours or less. A longer task might be achieved in short term increments, as long as each such increment is a deliverable, useful unit of work.
- **“Directed By Testers”** the progress of test automation is measured in terms of how it solves problems for testers and test managers. Test toolsmiths pair with a tester “client” who has a problem and requests help. The toolsmiths render that help in terms acceptable to the tester.

Tool-Supported Testing

- **Test generation (data and script generators).** Tools might create specialized data such as randomized email messages, or populate databases, or generate combinations of parameters that we'd like to cover with our tests.
- **System configuration.** Tools might preserve or reproduce system parameters, set systems to a particular state, or create or restore “ghosted” disk drives.
- **Simulators.** Tools might simulate sub-systems or environmental conditions that are not available (or not yet available) for testing, or are too expensive to provide live on demand.
- **Test execution (harnesses and test scripts).** Tools might operate the software itself, either simulating a user working through the GUI, or bypassing the GUI and using an alternative testable interface.
- **Probes.** Tools might make visible what would otherwise be invisible to humans. They might statically analyze a product, parse a log file, or monitor system parameters.
- **Oracles.** An oracle is any mechanism by which we detect failure or success. Tools might automatically detect certain kinds of error conditions in a product.
- **Activity recording & coverage analysis.** Tools might watch testing as it happens and retrospectively report what was and was not tested. They might record actions for later replay in other tests.
- **Test management.** Tools might record test results; organize test ideas or metrics.

Test tools are all over the place.

- **MSDN Universal Library**
- **Any Microsoft development tool** (they always include useful utilities)
- **Microsoft compatibility toolkit** and other free tools (www.microsoft.com)
- **Web-based web testing resources** (HTML checkers, accessibility analyzers)
- **Windows Resource Kits** (available from Microsoft)
- **Scripting languages** (e.g. Perl, Ruby, TCL) and associated libraries
- **Shareware repositories** (www.download.com)
- **O/S monitoring tools** (www.sysinternals.com)
- **Open source testware** (www.opensourcetesting.org)
- **Spyware for monitoring exploratory tests** (www.spectorsoft.com)
- **The cubicle next door...** (someone else in your company has a tool for you)

Tasks of the Toolsmith

- Respond rapidly to requests for assistance from testers.
- Seek out test productivity problems.
- Investigate possible solutions in concert with testers.
- Apply technology to improve the test process.
- Advocate for specific testability features in products.
- Research available tools and learn how to use them.
- Gather tools that developers or testers produce.
- Review upcoming product plans to assess automation possibilities.

Questions a Tester Might Ask a Toolsmith

- How do I test this new thing?
- How can I see what's happening inside the product?
- How do I know if this test passed or failed?
- Is there a way I can automatically perform this operation?
- Is there a way to make this bug easier to reproduce?
- Help me investigate this bug.
- Here's a test I want to execute. Can you help me generate 1000 variations of it?
- How well have I covered this product?
- I want to stress test this product. Are there any tools that do that?

Managing Agile Automation

(experience alert: I haven't tried this, yet)

- **Request list.** This is a list of new requests from customers (testers).
- **Assignments list.** This is a list of what each toolsmith is currently assigned to do.
- **Delivered list.** This is a list of solutions that are currently in use by the test team. Each item on the list should include a brief description and statement about the positive impact that solution has on testing productivity.
- **Maintenance request list.** This is a list of solutions that need improvement. Consider dividing it into two parts: critical maintenance and enhancements.
- **Obstacles list.** This is a list of productivity problems in testing that remain unsolved because they require expensive new tools, substantial testability improvements, or more work than can be done in a short term.

How many toolsmiths do you need?

For a finite, arbitrarily large test team...

at least one*
*co-located w/testers

Why not have a test team consisting entirely of toolsmiths?

Two big reasons...

they think like toolsmiths

they're expensive