Testing is Testing; Agile is Context

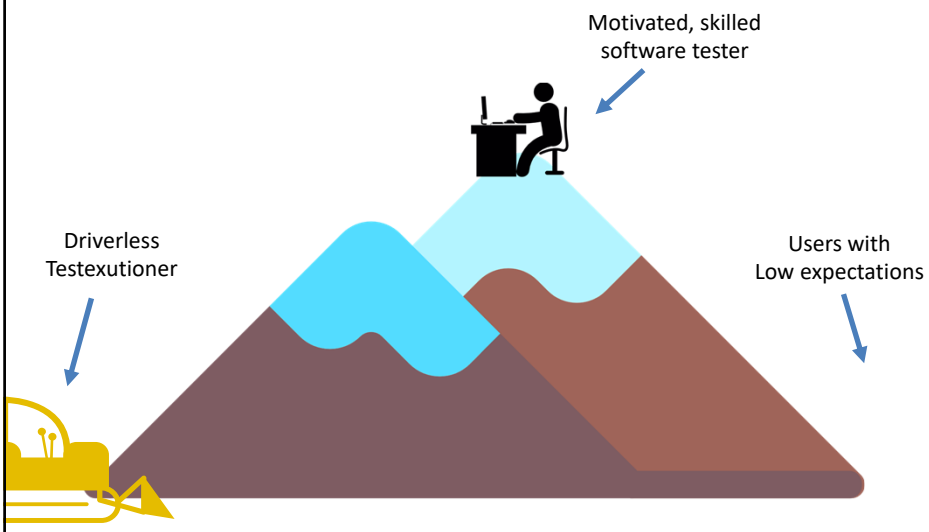
Michael Bolton
DevelopSense
http://www.developsense.com
@michaelbolton
michael@developsense.com

James Bach
Satisfice
http://www.satisfice.com
@jamesmarcusbach
james@satisfice.com

(with helpful comments from International Society of Software Testing members: Anne-Marie Charrett, James Lyndsay, Simon Morley, and Ben Kelly; graphic design help from Mary Alton….and credit to Scott Barber for the title)

I apologize in advance.

Demand for skilled testing is eroding.



TestingIsTestingAgileIsContext-2018-04 - 3

Demand for skilled testing is eroding
under assault by shiny automation on one side, and low quality standards on the other.

TestingIsTestingAgileIsContext-2018-04 - 4

Demand for skilled testing is eroding
under assault by shiny automation on one side,
and low quality standards on the other.
The entire industry has become dazzled by the values and practices appropriate for experimental, non-critical software.

TestingIsTestingAgileIsContext-2018-04 - 5

Demand for skilled testing is eroding
under assault by shiny automation on one side,
and low quality standards on the other.
The entire industry has become dazzled by the values and practices appropriate for experimental, non-critical software.
Meanwhile, testing culture, already weakened by ISTQB-style commodification, is stagnating.

TestingIsTestingAgileIsContext-2018-04 - 6

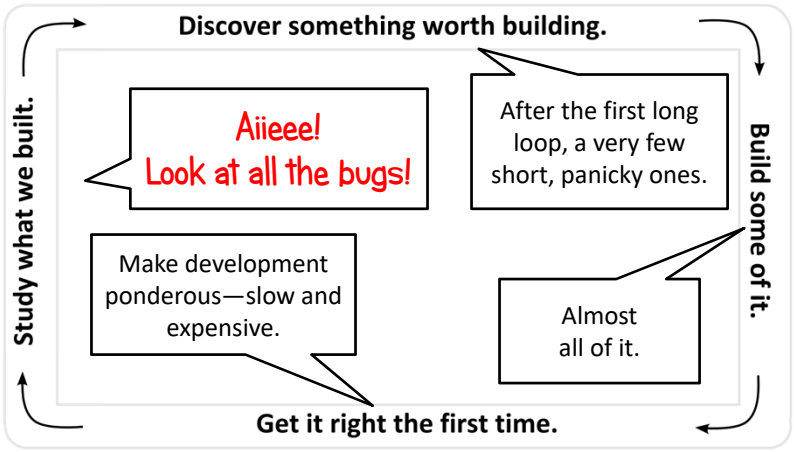
Why test *at all*?

There's probably some difference between the product we *think* we have, and *hope* we have, and the product we *really* have.

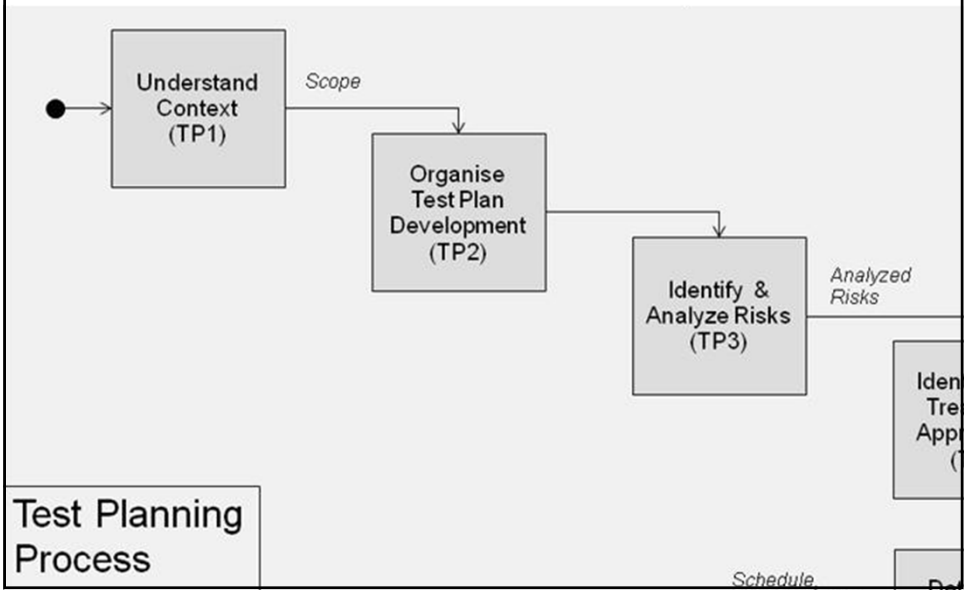
And if our product is important, and carries risk, it would be good to know about the differences.

The Universal Development Cycle

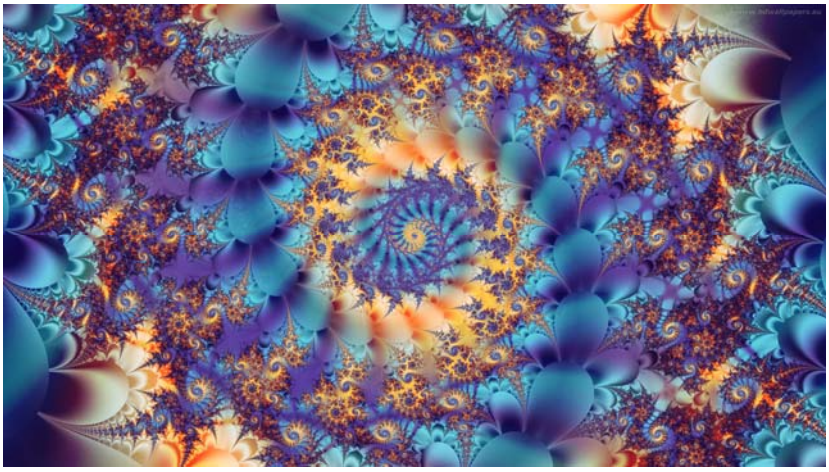
Traditional Development Cycle



Testing as an Assembly Line?



Development isn't linear…



development is a fractal!

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

http://www.agilemanifesto.org

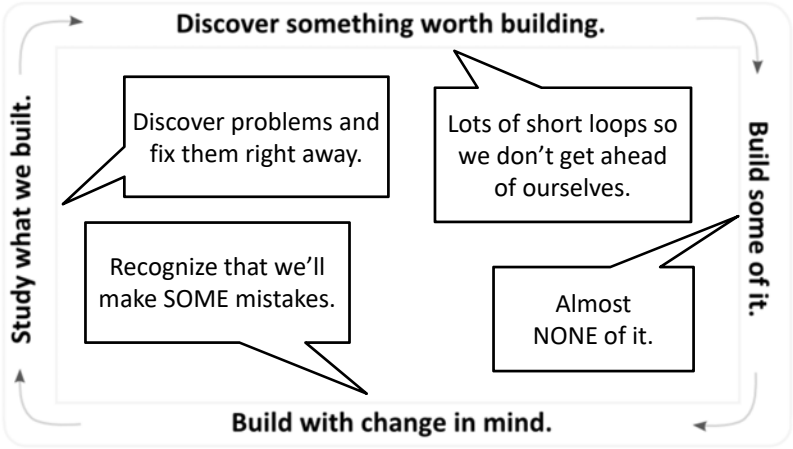
Principles of Agile Development

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Principles of Agile Development

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Development Cycle



Is This Agile Testing?

9.8.1 To verify P[illegible]racy

9.8.1.1 Connect the compo[illegible]s according to the *General Setup* docume[illegible]

9.8.1.2 P[illegible]r on and connect test jig (instead of the catheter)

9.8.1.3 Power on the Zapper Bo[illegible]

9.8.1.4 Power on the Control Bo[illegible]

9.8.1.5 Set default settings of **Temperature** and **Power** for the Zapper

9.8.1.6 Set test jig load to nominal v[illegible]

9.8.1.7 Select nominal duration and nominal p[illegible]er setting

9.8.1.8 Press the **Start** button

9.8.1.9 Verify Zapper [illegible]ds the power setting value [illegible]on display

THIS IS NOT TESTING

THIS IS OBSESSIVE DEMONSTRATION

Is This Agile Testing?

George is now an employee at Spacely's Space Sprockets, a manufacturer of "sprockets" and other high tech equipment. His job title is "digital index operator."

George's job primarily requires him to repeatedly push a single button (or on occasion a series of buttons) on a computer. In one of the episodes, George complained of his heavy work load—having to push a button for one hour, two days a week.

—Wikipedia

What does it mean to do “Agile Development”?

- Deliver often
- Collaborate across roles
- Develop craftsmanship
- Don’t be too formal
- Be prepared to try things, to fail, and learn
- Build and use tools
- Seek a sustainable pace

So what would Agile Testing look like?

- Individuals and interactions over processes and tools
  Focus on the skill set and the mindset of the individual tester
- Working software over comprehensive documentation
  Eliminate wasteful documentation; emphasize discovery and learning
- Customer collaboration over contract negotiation
  Answer the needs of the client and the team
- Responding to change over following a plan
  Respond rapidly to the ever-changing mission of testing.

Rapid Software Testing

Rapid Software Testing is a mind-set
and a skill-set of testing
focused on how to do testing
more quickly,
less expensively,
and more credibly and accountably.

RST is focused on how people learn and self-organize under pressure. We can apply it to any context.

What do managers and developers really want from testers?

An answer to this question:

Are there problems
that threaten
the on-time successful
completion of the project?

Two *Fundamental* Testing Questions

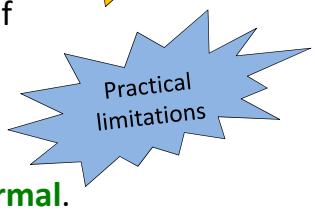
Is there a problem here?

Are we okay with this?

If you don't answer these questions,
people won't trust you.
That's when they start asking silly questions.

Premises of Rapid Software Testing

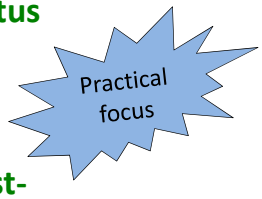
1. Software projects and products are **relationships between people**.
2. Each project occurs under conditions of **uncertainty and time pressure**.
3. Despite our best hopes and intentions, some degree of **inexperience, carelessness, and incompetence is normal**.



Therefore, software products and projects are fraught with risk.
It is our mission to investigate risk.

Premises of Rapid Software Testing

4. A test is an activity; it is **a performance**, **not artifacts**.
5. Testing's purpose is to **discover the status of the product and any threats to its value**, so that our clients can make informed decisions about it.
6. We commit to performing **credible, cost-effective testing**, and we will inform our clients of anything that threatens that commitment.
7. We will **not knowingly or negligently mislead our clients** and colleagues or ourselves.



Premises of Rapid Software Testing

8. Testers accept **responsibility for the quality of the testing work**, although they **cannot control the quality of the product**.

Rapid Testing Building Blocks

Rapid Software Testing uses a social and systems science approach informed and inspired by Jerry Weinberg, Herbert Simon, and Harry Collins

- **Context.** We listen and respond to the world around us.
- **Role and Self-Image.** Taking responsibility for our work.
- **Mission and Motivation.** Knowing what we are here to do.
- **Ethics and Integrity.** Rejecting waste and deception.
- **Diversity.** We need variety to cover complex products.
- **Relationships.** Working with ever-changing connections.
- **Skills.** Developing our abilities on the job.
- **Heuristics.** Fallible ideas and tools that solve problems.
- **Exploration.** Everything evolves; answers come over time.
- **Product Risk.** Constant danger of a bad bug hiding in the product.
- **Tests.** Not test cases… Actual tests!
- **Models.** Respecting both tacit and explicit knowledge.

What about…?
Quick Answers!

- **Management.** We focus on activities, not artifacts.
- **Metrics.** Never count test cases; maybe count time.
- **Automation.** We use tools. Tools are important.
- **Reporting.** Testers must learn to report and explain.
- **Documentation.** Concise! (Conversation is good.)
- **Speech.** Precise!

All of these points are consistent with the Agile Manifesto and Agile principles.

Call this "Checking" not Testing

operating a product algorithmically to check specific facts about it…

Think "spelling checker"

means

Observe → **Evaluate** → **Report**

Interact with the product in specific, *algorithmic* ways to collect specific observations.

Apply *algorithmic* decision rules to those observations.

Report the outputs of the evaluations *algorithmically*.

A check can be performed…

by a machine
that *can't* think
(but that is quick and precise)

by a human
who has been told *not* to think
(and who is slow and variable)

Notice that "quick" and "slow" refer only to the speed of observable behaviours and algorithmic evaluations.
The machine is *infinitely* slow at recognizing unanticipated trouble.

Testing is…

operating a product algorithmically to check specific facts about it…

Testing

Scripted Testing

They say…

"Automate all the testing!"

They might have meant…

"Automate all the checking!"

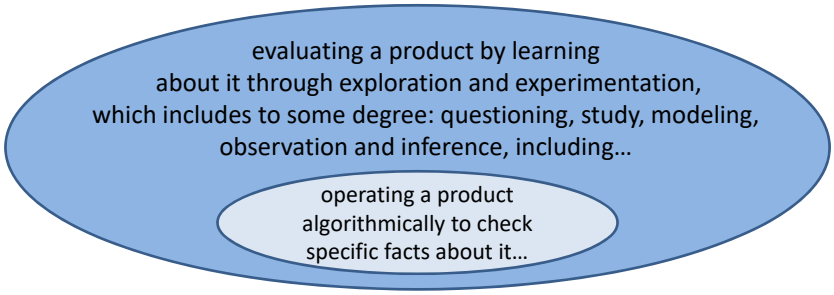
or…

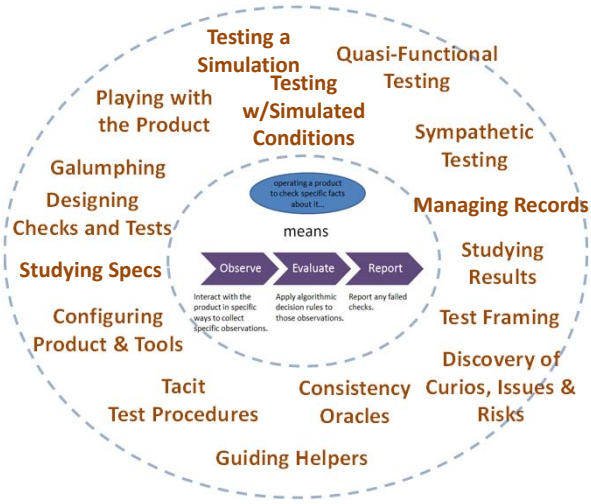
"Use tools!"

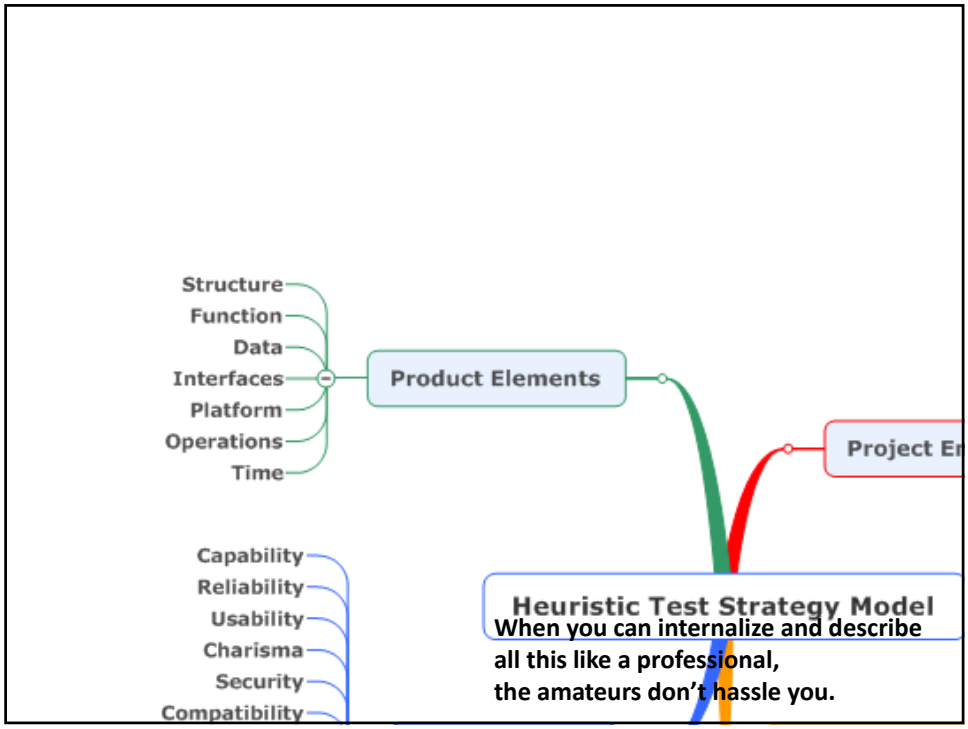
…which means

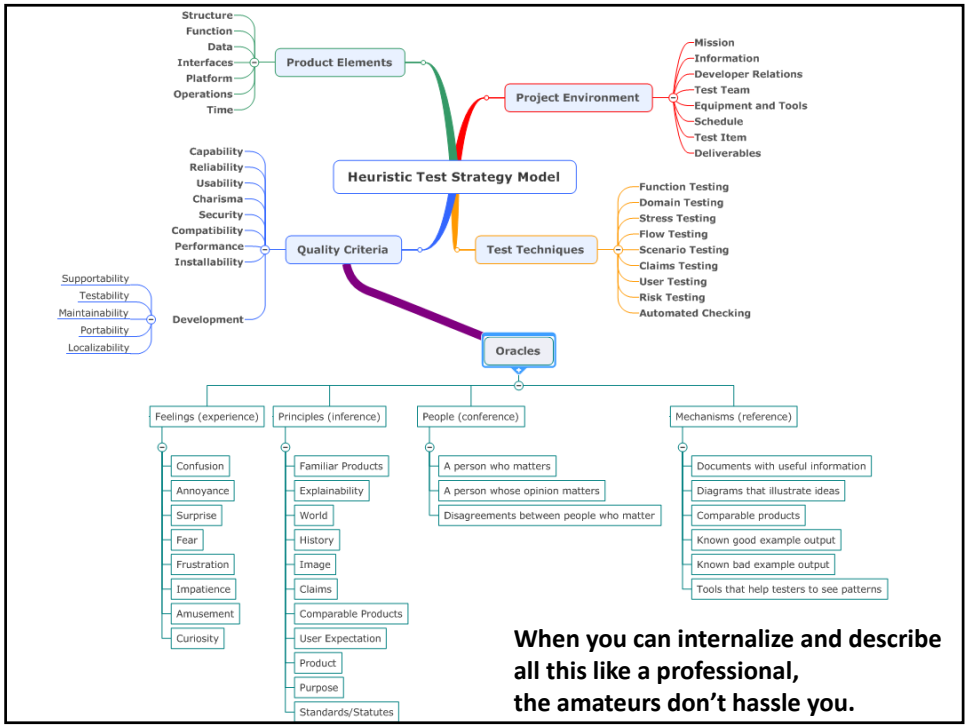
and risk assessment
and task prioritization
and coverage analysis
and pattern recognition
and decision making
and design of the test lab
and preparation of the test lab
and sensemaking
and test code development
and tool selection
and recruiting of helpers
and making test notes
and preparing simulations
and interacting with developers
and bug advocacy
and triage

Testing is…





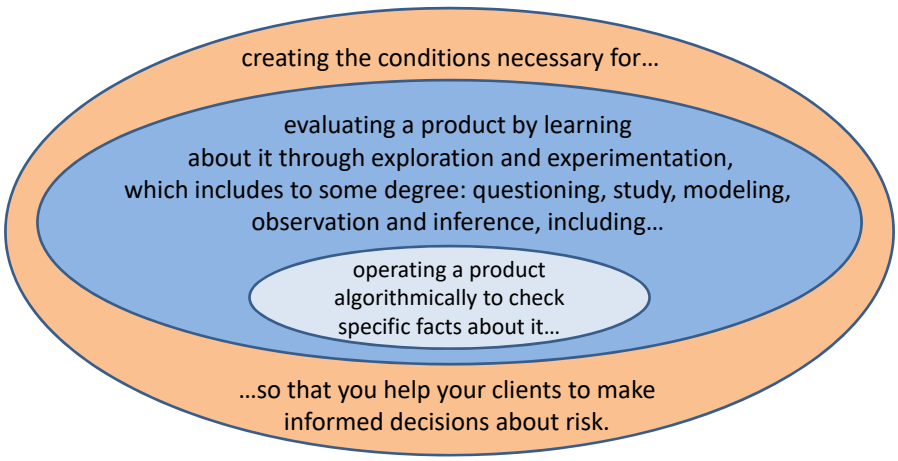


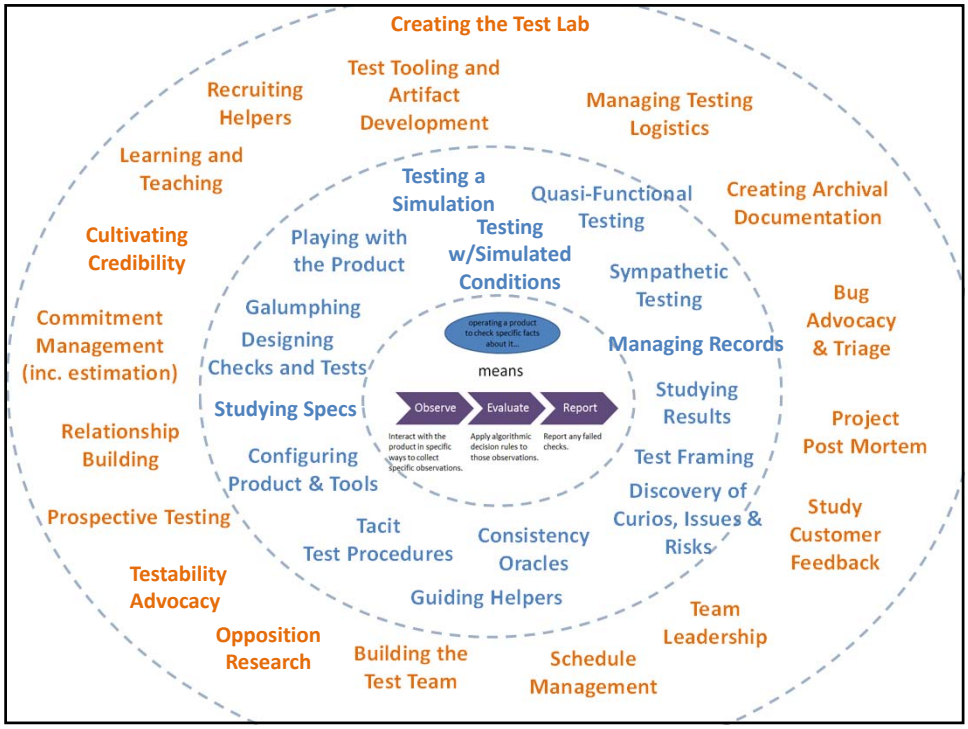


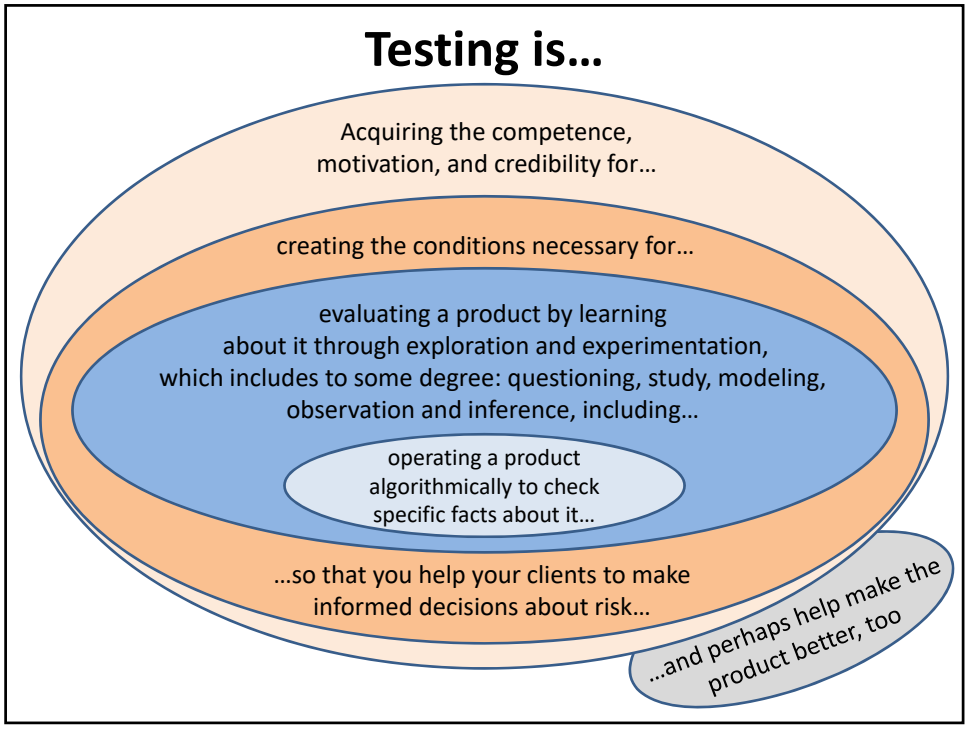
Why not say “exploratory testing”?

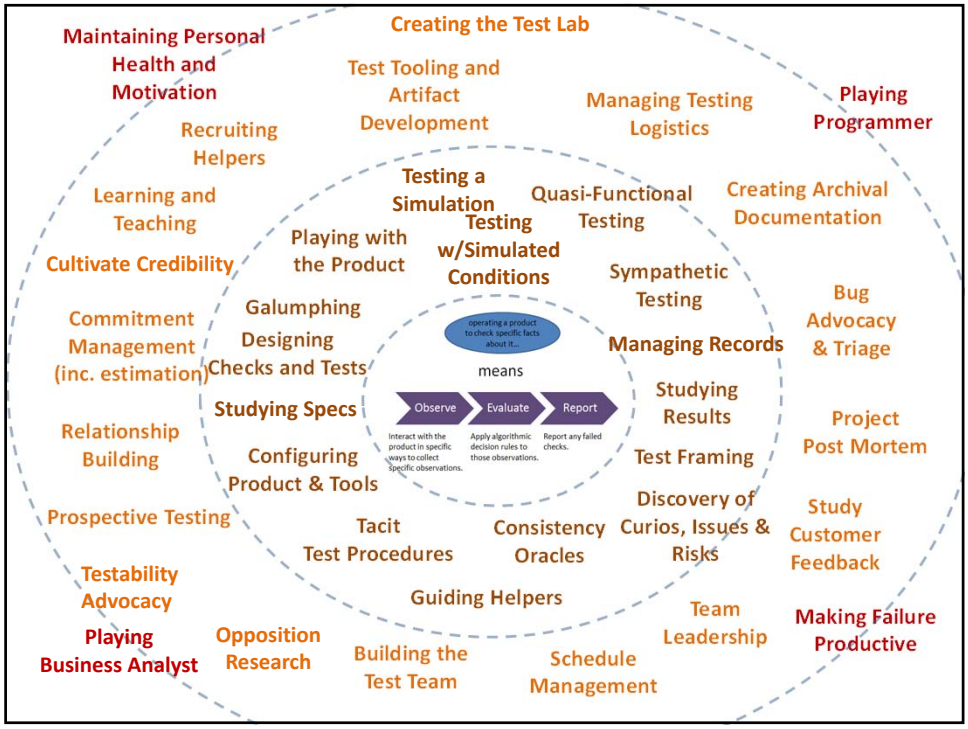
Why not say “vegetarian cauliflower”?

Testing is…









Why not just eliminate roles?

You have probably experienced this.

Why do some people think a role is a prison or a fortress?

This happens when "role" is defined as **the only things you do** and **what no one else does**.

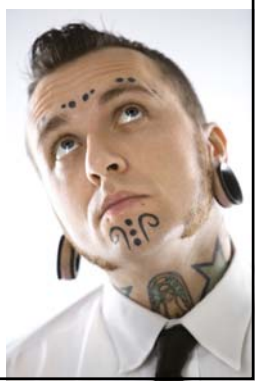
HIGH SOCIAL DISTANCE

What a Role Is…

- a commitment to perform some service(s)
- an idea to focus commitments
- a way to help organize effort on a team
- a heuristic for explaining or defining work
- like a hat

What a Role Is NOT

- Not a declaration of the only things you are allowed to do (not a prison)
- Not a declaration of the things that you and you only are allowed to do (not a fortress)
- Not permanent and unchanging
- Not like a tattoo



We like to think of roles this way.

A role is like a villa. It is a semi-private space. Someone dwells in it. Someone is responsible. But visitors may come and help.

- Developers help testers.
- Testers help developers.
- But testers are **ACCOUNTABLE** for testing work and processes.
- *Flexible* social distance

A Healthy Role Institutionalizes…

- **Competence:** Increases skill over time.
- **Focus:** Marshals energy and concentration to solve difficult problems well; economy of scale.
- **Anticipation:** Identifies future needs and potential problems before its too late.
- **Accountability:** Accepts responsibility for outcomes within scope of the role.
- **Coordination**: Minds the interface with other roles.

See “On a Role”
http://www.developsense.com/blog/2015/06/on-a-role/

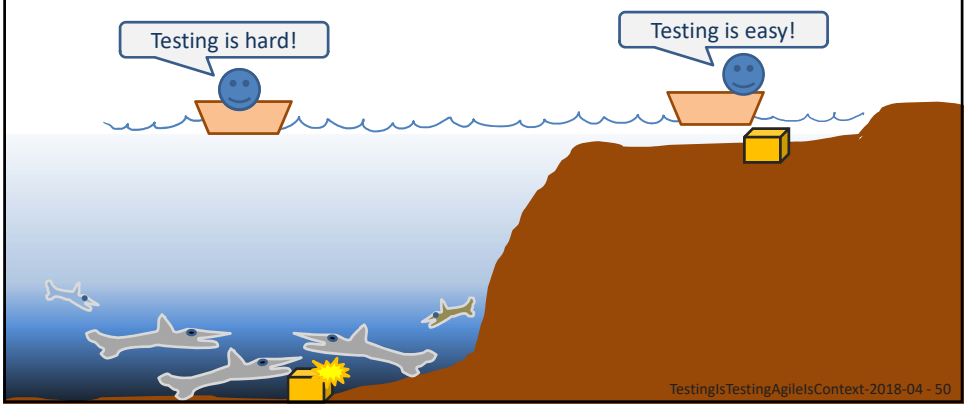
What is the testing role?

- *To test* is to evaluate a product by learning about it through exploration and experimentation.
- When someone is testing, that person has adopted (if only for that time) a testing role
- *A tester's role* is to
  - to develop one's self as a tester
  - connect with the clients of testing
  - prepare for testing
  - perform testing
  - report the results of testing.

What's Unique About Real Testing Specialists?

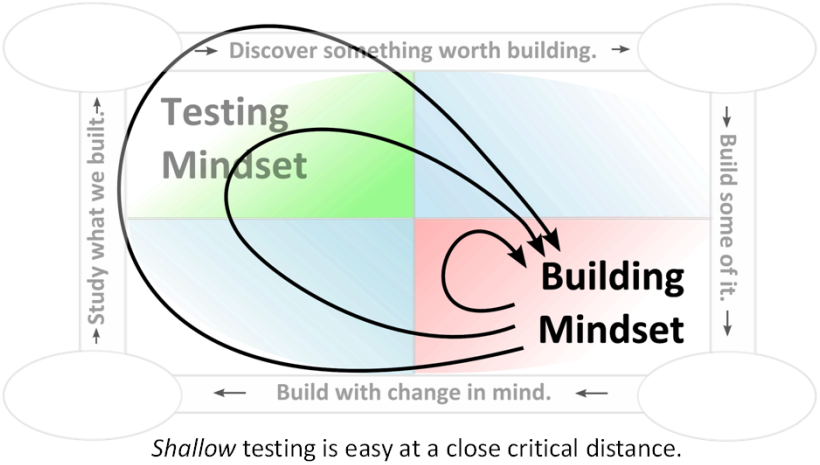
We test more deeply and reliably.

(Why? Because we aspire to do so; we enjoy doing it; and we learn how.)



Critical Distance

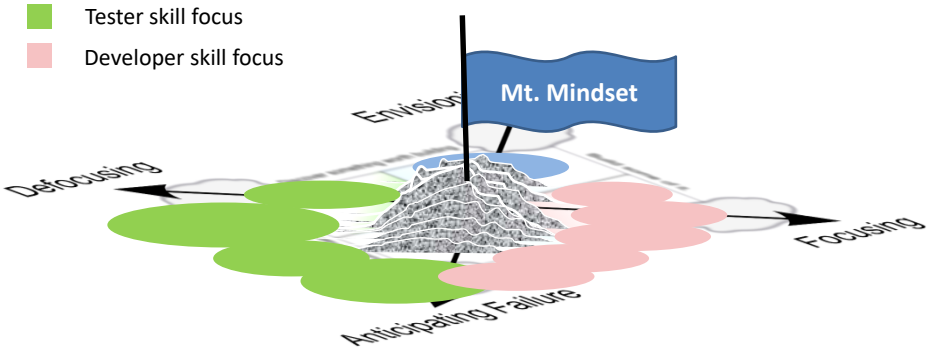
"Critical Distance" refers to the difference between one perspective and another.
Testing benefits from diverse perspectives.



Shallow testing is easy at a close critical distance.
Deeper testing, more like users, or more focused on more diversified risk, tends to require or create more distance from the builder's mindset.

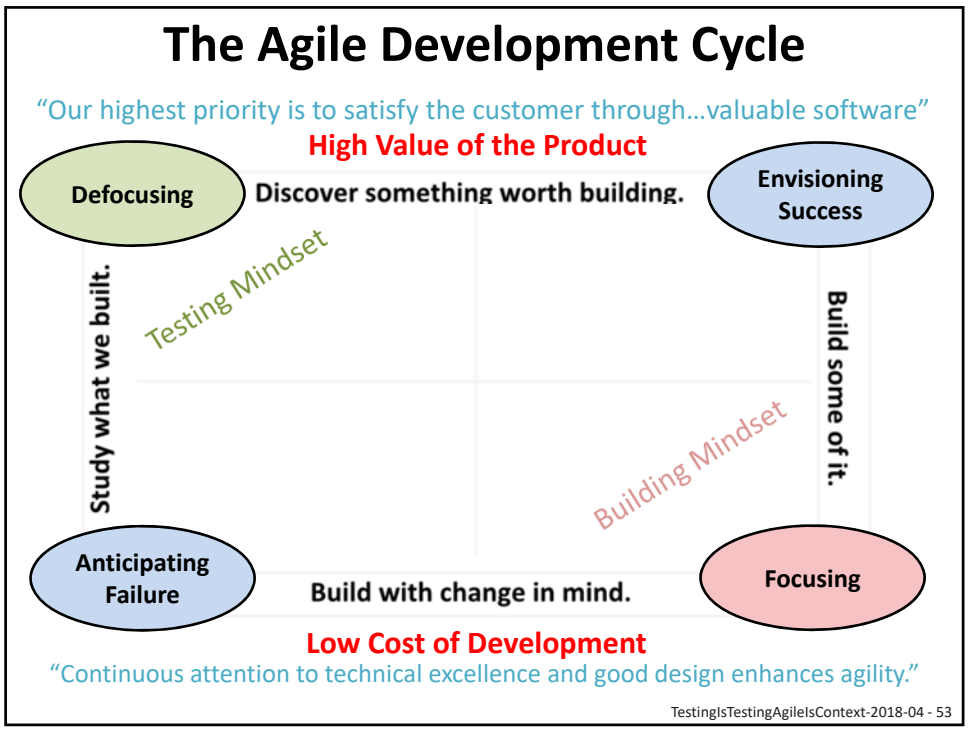
Why roles? Because changing mindsets is HARD.

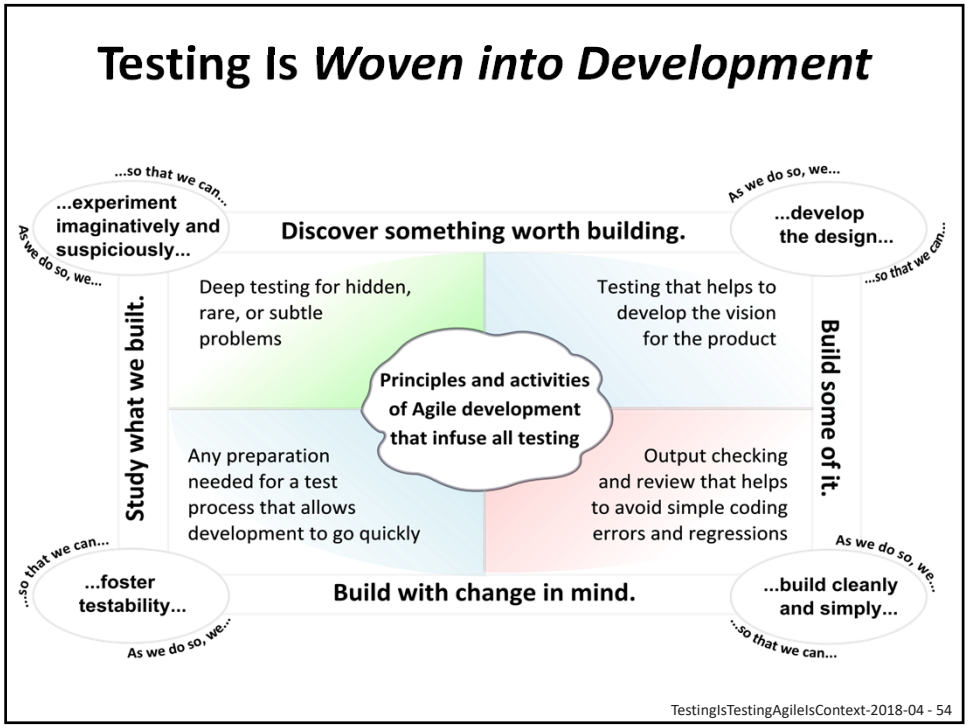
- Business analyst skill focus
- Tester skill focus
- Developer skill focus



NOTE: We do NOT claim that different kinds of work *must* be done by different people, or that the people *must* have different titles.

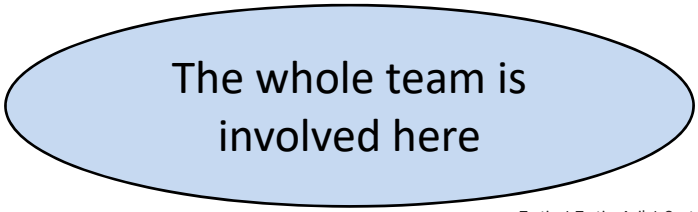
We DO claim that having skilled people *focused* on testing is a powerful heuristic for addressing the mindset switching problem.





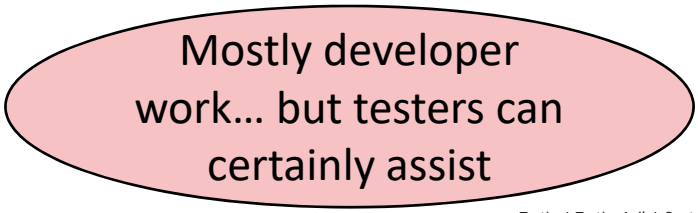
Developing the Design

- Exploring definitions of done
- Engaging with diverse users
- Specifying product *with* rich examples (not "by")
- Reviewing reports from the field
- Exploring design trade-offs
- Refining user stories



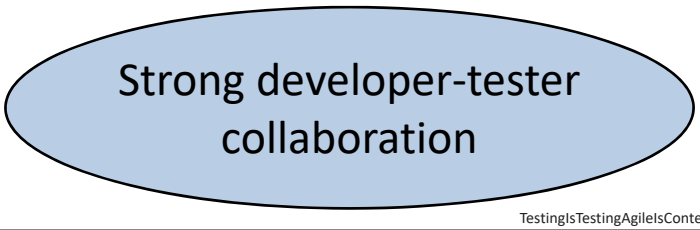
Building Cleanly and Simply

- Automating low-level checks
- Establishing shared coding style
- Reviewing each other's code
- Building the product frequently
- Re-factoring for maintainability
- Investigating and fixing bugs as we go



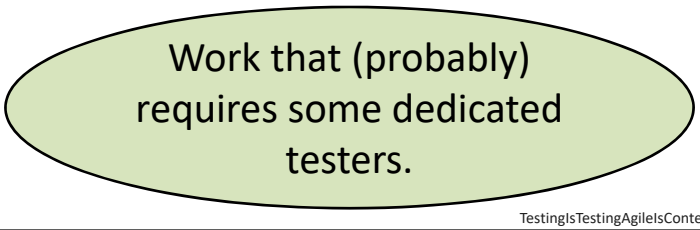
Fostering Testability

- Preparing test environments and tools
- Making the product easy to test
- Identifying and exploring product risk
- Minimizing trouble when changing the product
- Removing obstacles and distractions to testing
- Testing in parallel with coding

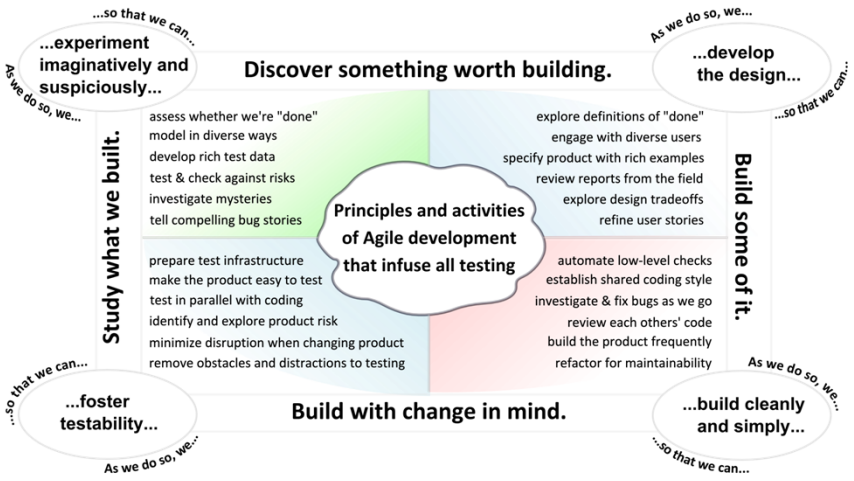


Experimenting Imaginatively and Suspiciously

- Assessing for done—or not done *yet*.
- Modelling in diverse ways
- Developing rich test data
- Testing and checking against risks
- Investigating mysteries
- Telling compelling bug stories



RST's Agile Quadrants in Detail



Foreseeable Problems

- Problem:
  - Programming time and testing time are asymmetrical.
  - Deep testing may take time, effort, tools, skills to perform.
- Answers:
  - Focus on testability.
  - DO treat sprints as appointments for a review.
  - DON'T treat sprints as commitment to finishing something, but as commitments to try finishing them—and to see what we learn in the process.
  - Notice when pace becomes unsustainable.

Foreseeable Problems

- Problem:
  - Agile (and especially) DevOps means that we're getting new builds all the time—no time to test!
- Answers:
  - "No time to test" *literally* means "no time to study the product and investigate it for potential problems". Is everybody OK with that?
  - DO treat continuous integration as an opportunity to do deep testing on periodic stable builds and quick checks on incremental builds.
  - DON'T let frequent, frantic building mislead you into thinking that a checked product is a tested product.

Technical Suggestions

- Resist test cases and scripts; focus on test activities and the testing story.
- Let risk guide your testing
- Test in short, uninterrupted sessions; review and discuss them; seek and provide feedback.
- Avoid premature, excessive formalization.
- Keep documentation concise.
- Use recording tools like an airplane "black box".
- Emphasize exploratory scenario testing.
- ASK FOR TESTABILITY!

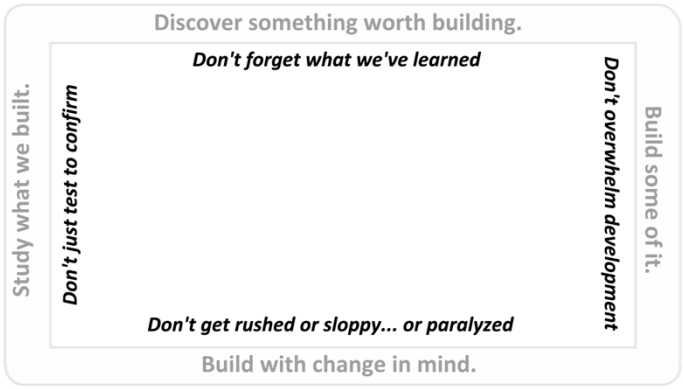
Tools?

- DON'T use them to "do" the testing. Tools don't do testing; YOU do.
- DON'T become fixated on tools.
- DO prefer lightweight tools, in general.
- DO use them to **support** testing.
  - setup and configuration management
  - data generation
  - probing the product
  - visualization
  - logging and recording
  - automated checking (most efficiently at the unit and integration levels; not so much at the GUI)

Social Suggestions

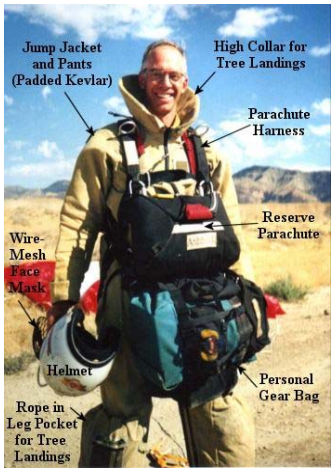
- Practice explaining testing.
- Declare your role and commitments.
- Don't accept responsibility for the quality of the product.
- Embed yourself (or your testers) with the development team as
- Ask for testability.
- Watch where time and effort are going.
- Note the advantages of developer testing.
- Resist bureaucracy.
- Be a service to the project, not an obstacle.

Some Caveats

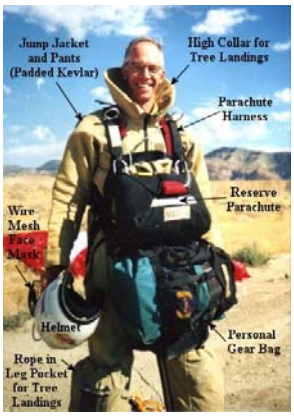


Career Path: Test Jumper!

- may spend weeks on one project, acting as an ordinary responsible tester.
- may spend a few days on one project, organizing and leading testing events, coaching people, and helping to evaluate the results.
- may spend as little as 90 minutes on one project, reviewing a test strategy and giving suggestions to a local tester or developer.
- may attend a sprint planning meeting to assure that testing issues are discussed.
- may design, write, or configure a tool to help perform a certain special kind of testing.
- may coach another tester about how to create a test strategy, use a tool, or otherwise learn to be a better tester.
- may make sense of test coverage.
- may work with designers to foster better testability in the product.
- may help improve relations between testers and developers, or if there are no other testers help the developers think productively about testing.



Test Jumper Requirements



- The ability and the enthusiasm for plunging in and doing testing right now when necessary.
- The ability to pull one's self out of a specific test task and see the big picture.
- The ability to recruit helpers.
- The ability to coach and train testers, and people who can help testing.
- A wide knowledge of tools and ability to write tools.
- A good respectful relationship with developers.
- The ability to speak up in sprint planning meetings about testing-related issues such as testability.
- A keen understanding of testability.
- The ability to lead ad hoc groups of people with challenging personalities during occasional test events.
- An ability to speak in front of people and product useful and concise documentation as necessary.
- The ability to manage many threads of work at once.
- The ability to evaluate and explain testing in general, as well as with respect to particular forms of testing.

Talking More Clearly About Testing

Replace	**with…**
Verify that…	Challenge the belief that…
Validate	Investigate
Confirm that…	Find problems with…
Show that it works	Discover where it *doesn't* work
Pass vs. fail…	Is there a problem here?
Test case	Test conditions and test ideas
Counting test cases	Describing coverage
Automated testing	Programmed checking
Test automation	Using tools in powerful ways
Use cases	Use cases AND *mis*use cases AND *ab*use cases AND *obt*use cases…
KPIs and KLOCs	***Learning from every bug***

Talking More Clearly About Testing

Replace…	**With…**
"The environment's down. We're stuck. We can't test."	"What can we test, review, or analyze now… and *are you OK with this situation*, dear client?"
"They didn't give us good requirements documents!"	"Let's write down what *we* know—and then they'll tell us when they think it's wrong!"
"It's too hard to test this!"	"What can we do in the product and the project to things more testable?"
"We don' t have enough time to test!"	"What testing shall we do—what shall we cover—in the time we *do* have?"
"We have to…!"	"We choose to…"

Getting started…

- **Take advantage of resources**
  - Email me
  - See James' book *Lessons Learned in Software Testing*
  - Browse the appendices of the class
  - Join forum: software-testing@yahoogroups.com
- **Do some focused ET**
  - Try a three-hour chartered bug hunt with a group of testers. *See what happens.*
  - Do some ET sessions by yourself and practice noticing how you think and practice taking notes
- **Practice solving puzzles**
  - Solve jigsaw puzzles, logic puzzles, Sudoku, cryptograms, or lateral thinking puzzles.

Getting started…

- **Try using guideword heuristics**
  - Use the models from this class
  - Modify the models from this class to fit your context
  - Much better: build your own models
- **Defrost your procedures**
  - Pick some procedures and simplify them.
  - Generalize some of your procedures.
  - Include design information to help test executors contribute.
- **Start a lunch-time test coaching meeting**
  - Pick a part of the product each week and talk about how to test it. Get the whole team to brainstorm better ways.
  - Do a risk brainstorm.
  - Review bug reports and talk about how to write better ones.

Getting started…

- **Practice your systems thinking**
  - Read “The Art of Systems Thinking” or “An Introduction to General Systems Thinking”
  - Try the exercises in those books.
  - Take any part of your product and model it in terms of its most important dimensions.
- **Put a coverage/risk outline inside your test documentation**
  - You don’t have to throw out your old test documentation, but how about putting a new section in that consists of a coverage outline, a risk outline, or a combination of the two?
- **Question value vs. cost for all your testing**
  - Is any of your testing not giving you much value relative to its cost? Why are you doing it, then?

Keep in touch!

James Bach
Satisfice, Inc.
james@satisfice.com
www.satisfice.com
+1 (360) 376-4367

Michael Bolton
DevelopSense
mb@developsense.com
www.developsense.com
+1 (416) 656-5160

Further Reading

- In addition to the bibliographies in Agile Testing and More Agile Testing, have a look at…
- "Test Cases Are Not Testing: Toward a Culture of Test Performance" by James Bach & Aaron Hodder
  - http://www.testingcircus.com/documents/TestingTrapeze-2014-February.pdf#page=31)
- Testing Problems are Test Results
  - http://www.developsense.com/blog/2011/09/testing-problems-are-test-results/
- Testers: Get Out of the QA Business
  - http://www.developsense.com/blog/2010/05/testers-get-out-of-the-quality-assurance-business/
- Testing and Checking Refined
  - http://www.satisfice.com/blog/archives/856
- RST Methodology: Responsible Tester
  - http://www.satisfice.com/blog/archives/1364

Further Reading

- Test Jumpers: One Version of Agile Testing
  - http://www.satisfice.com/blog/archives/1372
- Done, The Relative Rule, and The Unsettling Rule
  - http://www.developsense.com/blog/2010/09/done-the-relative-rule-and-the-unsettling-rule/
- The Undefinition of “Done”
  - http://www.developsense.com/blog/2011/07/the-undefinition-of-done/
- At Least Three Good Reasons for Testers to Learn to Program
  - http://www.developsense.com/blog/2011/09/at-least-three-good-reasons-for-testers-to-learn-to-program/