The REAL Agile Testing Quadrants

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

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

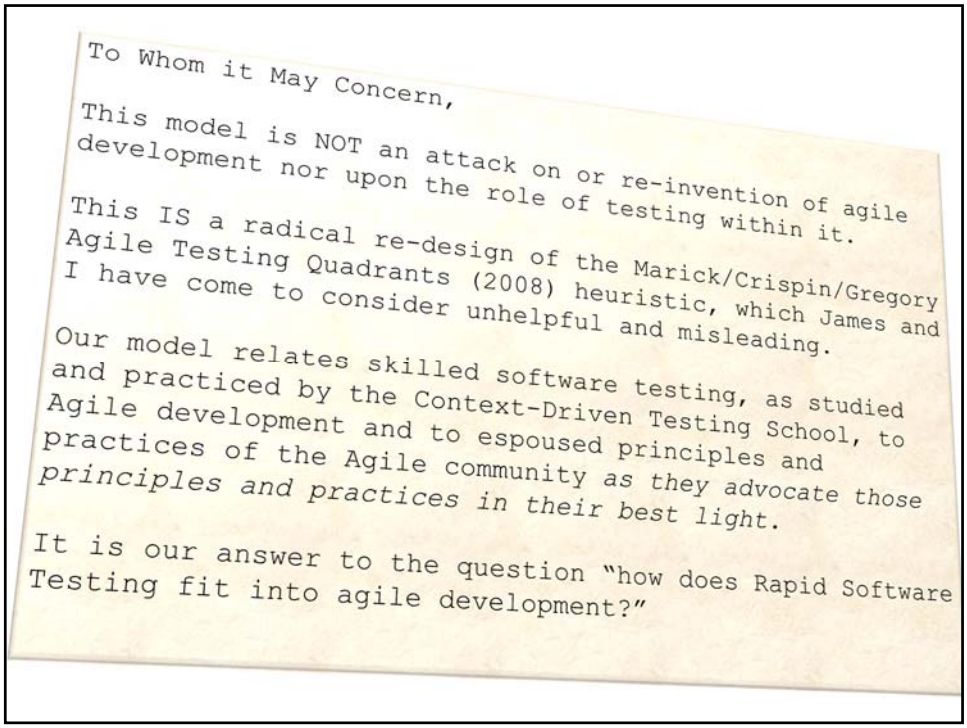
(with helpful comments from International Society of Software Testing members: Anne-Marie Charrett, James Lyndsay, Simon Morley, and Ben Kelly; and graphic design help from Mary Alton)

The ~~REAL~~ Agile Testing Quadrants
(as we believe they should always have been)

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

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

(with helpful comments from International Society of Software Testing members: Anne-Marie Charrett, James Lyndsay, Simon Morley, and Ben Kelly; and graphic design help from Mary Alton)



What is the testing role?

- *To test* is to evaluate a product by learning about it through exploration and experimentation.
- *A tester's role* is to develop himself as a tester, connect with the clients of testing, prepare for testing, perform testing, and report the results of testing.

Why is this the testing role?

- *Because that's the meaning and history of "testing".*
- Because to add quality policing or improvement to testing creates responsibility without authority; usurps management's role; and sets testers up as scapegoats.
- Because this is already a very challenging portfolio and skill set. Adding anything to it distracts and dilutes testing effort.

But wait! Does that mean that testers should ONLY look for bugs?

Relax… a role is a heuristic, not a prison.

- If I am a developer, can I do testing? Of course!
- *As a developer, you already do testing*. **And** you will have to sharpen your skills and cope with certain handicaps and biases if you want to do *great* testing.
- If I am a tester, can I make quality better? Sure!
- **And** if you do that, you will have adopted, at least temporarily, some kind of developer role. It's hard to wear two hats at once.
- If I am a goalie, can I score goals, too? No rule against it!
- **But** if you come forward, your team's goal is open. **And** the person covering it can't use his hands.
- If I am a janitor, can I offer suggestions to the CEO? Hey, why not?
- **But** *her* role is not necessarily to listen, or to comply.
- If I am not the driver of a car, can I grab the steering wheel?
- Go ahead… **but only if** the driver is incapacitated.

Relax… a role is a heuristic, not a prison.

- If I am a father, do I HAVE to do fathering???

Well, yes.

But wait! Does that mean that testers should ONLY look for bugs?

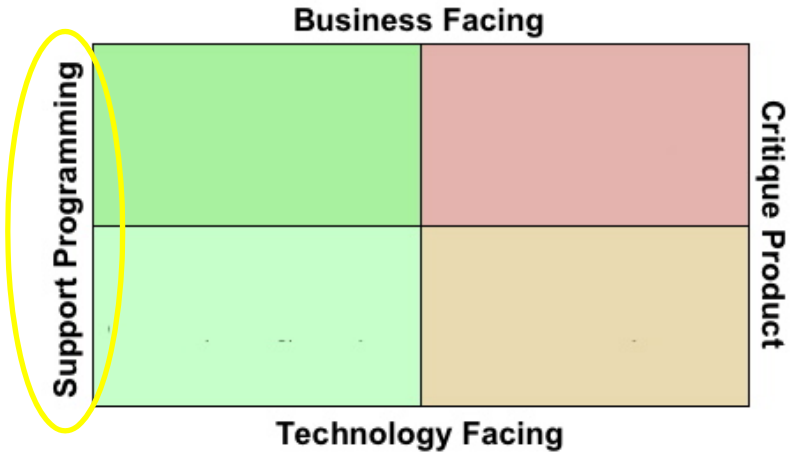
Of course not.
But finding problems that threaten the value of the product is an important task in creating a valuable product.
Testing focuses on that task.

4

Our Problems with the Quadrants

- James encountered the quadrants first in 2003 or so, when Brian Marick explained them to him; I started to hear about them shortly after that.
- I participated in the Agile Testing Mailing list, which seemed to exalt processes and tools, but not talk about *testing* much.
  - There was lots of talk about checking, but they didn't call it that—but in fairness, back then, I didn't either.
- I abandoned the list in 2008 or so, after I got tired of the misrepresentation and dumbing-down of testing.
- **I feel that the quadrants helped to feed that misrepresentation.**
- We have learned much more about (agile) testing and how to discuss it since the quadrants arrived; it's time for an overhaul.

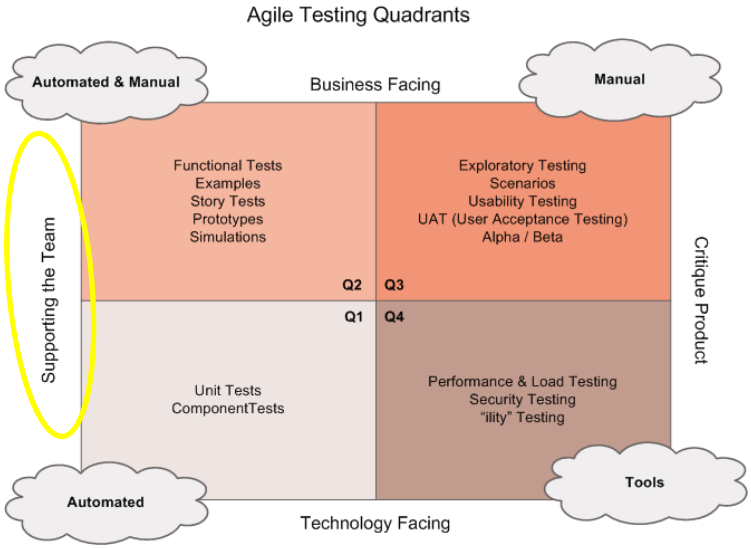
Marick's Original



See http://www.exampler.com/old-blog/2003/08/21/, http://www.exampler.com/old-blog/2003/08/22/#agile-testing-project-2, and subsequent posts.

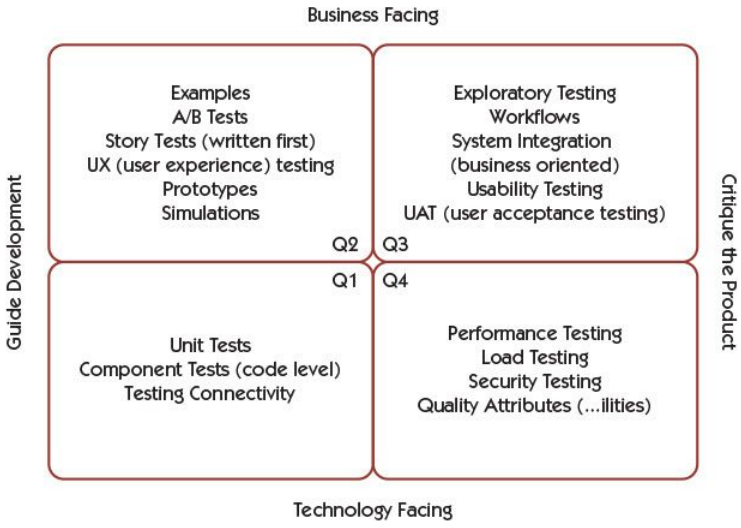
5

Crispin & Gregory's (Earlier) Version



See Crispin & Gregory, *Agile Testing*

Crispin & Gregory's (Newer) Version



See Crispin & Gregory, *More Agile Testing*

6

Supporting Programming or the Team?

- Marick's original and his comments on it frame simple output checks as more "integral" to the programming process than vigorous testing.
  - **Maybe he was talking about lower critical distance; okay.**
  - **Nonetheless, let's treat it all as connected together, in super-rapid feedback loops. It's *agile development*, right?**
- The Crispin & Gregory version implies that critique is not supporting the team, or not the work of programming. That in turn implies that that testers do not belong in Agile unless they write code.
  - **Testing—critiquing the product—IS supporting the team**
  - **Testers may *or may not* write code, use particular tools, or apply particular skills.**

Tools and techniques are everywhere

- The first Crispin/Gregory version made confusing and unnecessary distinctions about testing with and without tools.
  - Tools are not remarkable in testing. Good testers use them anywhere, *everywhere*, for lots of purposes.
  - There is no such thing as "manual" or "automated" testing, just as there isn't "manual" or "automated" programming.
    - See http://www.developsense.com/blog/2013/02/manual-and-automated-testing/
- Both versions pin certain techniques and approaches to certain quadrants.
  - We believe any test technique or approach may relate to any quadrant, for which there may be overarching tasks and goals.

Why You Might Like the Old Quadrants

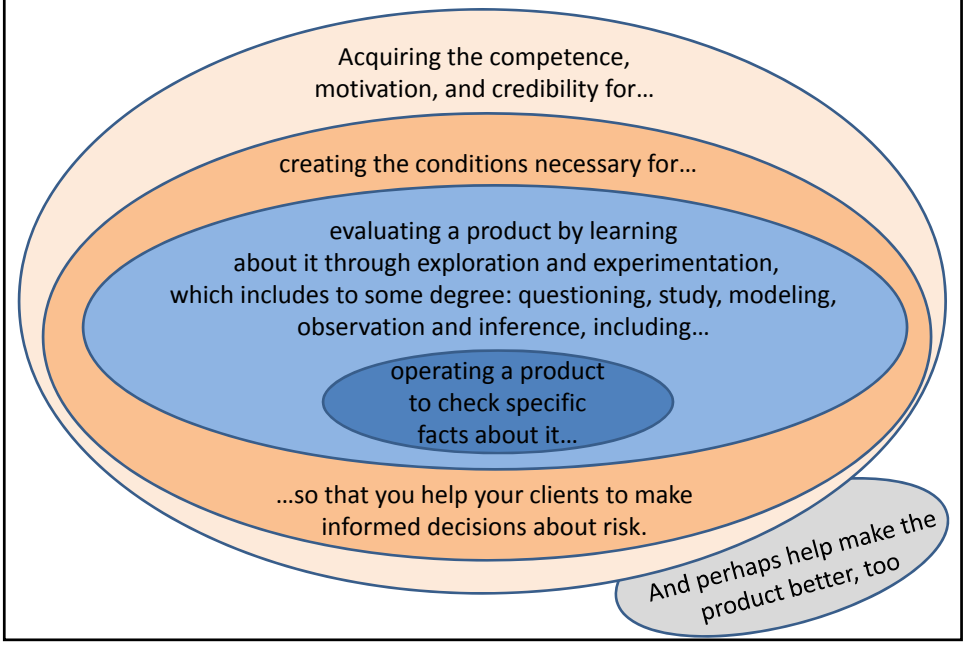
It's an example of a generic diversified test strategy!

- Our replies
  - We can *all* have generic test strategies that
    - better represent testing as we see it.
    - go MUCH deeper.

Our Primary Motivations to Revise

- People have been asking us for years how Rapid Testing fits with Agile Testing. We've needed a bridge for quite a while.
- The Quadrants are ten years old or so.
- Meanwhile, we've learned a lot about how to describe and frame our work.
- A few things that have been bugging us in the Agile Quadrants. For those who agree with those bugs, we offer some fixes.

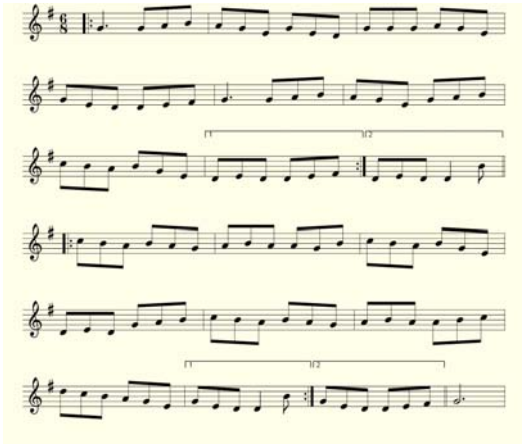
Testing is…



Testing is more than checking

- We should not confuse output checking (which is completely automatable) with testing (which is not).
  - Just like programming, testing is an active thought process. Checking lives *inside* it, just as compiling lives inside programming.
    - See http://www.satisfice.com/blog/archives/856
  - This message is not only for the Agile community!
  - To give Agilists their due, Marick refers to "checked examples", which he says he got from Ward Cunningham. Let's honour that!
  - The idea of "checked examples" likely influenced me, but Turing, McCracken, and Weinberg all referred to checking long before any of us showed up.

Ceci n'est pas musique.



This is not music; this is musical *notation*, a scheme for identifying what notes to play. Until a string is bowed or plucking, until someone blows through a mouthpiece, until someone whacks a drum… until *air starts to move*, there is no music.

Music is something that happens.

Removing Reification Fallacies

"test cases are testing" and "examples are tests"

- "To test" is a verb; a test is a performance, not an artifact*.
  - "**A** test" is an instance of testing activity.
  - "A test" is neither example nor script.
- Testing cannot be encoded.
  - Just as *programming* cannot be encoded; you cannot script the interpretation, invention, innovation, and problem-solving that happens in programming work. That stuff happens in testing too.

*See "Test Cases Are Not Testing: Toward a Culture of Test Performance", Bach & Hodder http://www.testingcircus.com/documents/TestingTrapeze-2014-February.pdf#page=31)

Removing Reification Fallacies

"test cases are testing" and "examples are tests"

- It is pointless to discuss whether "business people" can "read the tests"…
- …because *testing*—an activity—cannot be *read…*
  - we don't speak of business people "reading the code"
  - at best, *some* representations of *some* testing activity (checks) can be read in advance
  - test reports can be read retrospectively
  - for prospective purposes, it's probably more useful to think in terms of *examples*, but those are distinct from tests
- …and because if you try to communicate testing primarily through writing or other artifacts then you are probably underestimating the role of tacit knowledge
  - **and you are likely in conflict with Agile principles**
  - instead, prefer conversation, "knowledge crunching", description, demonstration, and interaction with products

"Facings" are beside the point.

- THE BUSINESS needs us to produce something of value.
- THE BUSINESS needs us to do that efficiently.
- THE BUSINESS needs to learn what it values over time, rather than guessing at the start of the project, and freezing that guess until the end.
- "Technology-facing" simply means doing things that help us **build with change in mind**—an activity our business clients need but do not *directly* care about (or sometimes even know about.)

11

Core Agile Heuristics

- "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."
  - therefore, continually refocus on value
  - and, in the testing role, refocus on *threats* to value
- "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
  - therefore, ply our craft in ways that reduce the cost of responding to constant change
  - and, in the testing role, provide relevant feedback about the product as rapidly as possible, to anticipate the risks and understand the effects of change

HOW do we do "Agile Development"?

- Discover something worth building
- Build some of it
- Build it with change in mind
- Study what we've built
- …and iterate!

12

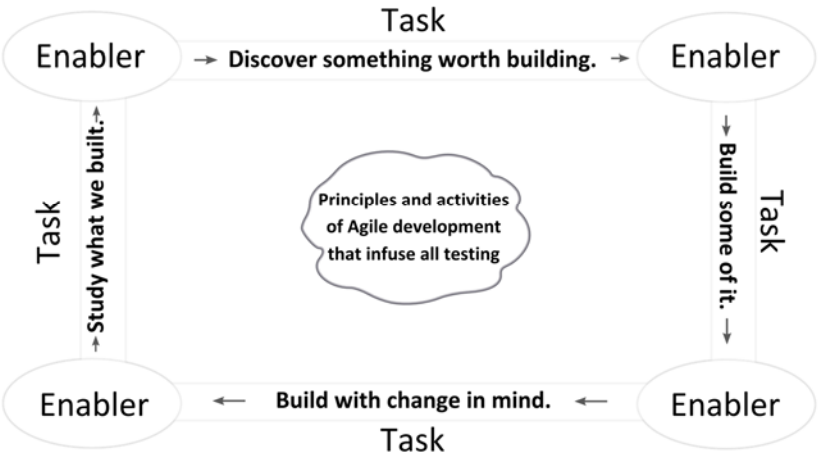
Agile Development Cycle
(which ought to be infused with testing)

What does it mean to do "Agile Development"?

- Deliver frequently
  - short sprints help prevent us from getting ahead of ourselves
- Collaborate across roles
  - building a quality product is a goal we all share
- Cultivate craftsmanship
  - "being Agile" won't help us without study, practice and skill
- Avoid excessive formalization
  - individuals and interactions over processes and tools; working software over comprehensive documentation
- Apply appropriate heuristics
  - be prepared for and tolerant of occasional failures
- Develop and apply tools
  - tools are everywhere in software development and testing
- Seek a sustainable pace
  - people who are overwhelmed tend not to do good work

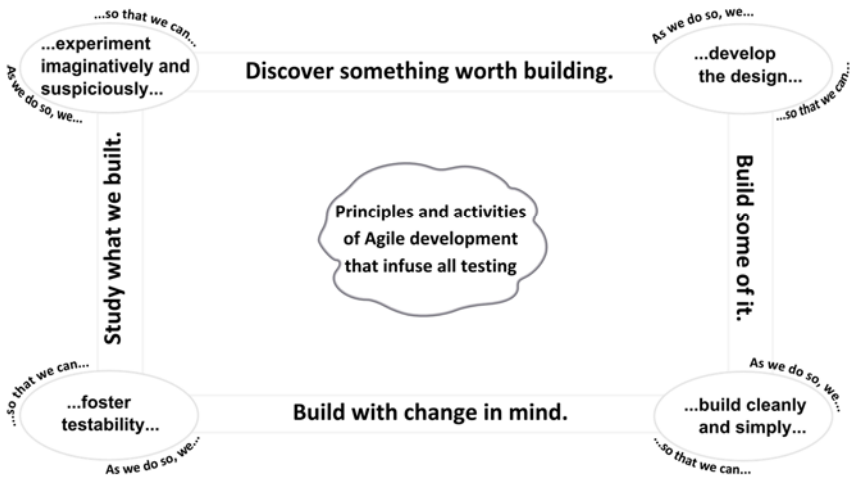
Agile Development Heuristics
(which ought to be infused with testing)

Tasks and Enablers



14

Tasks and Enablers



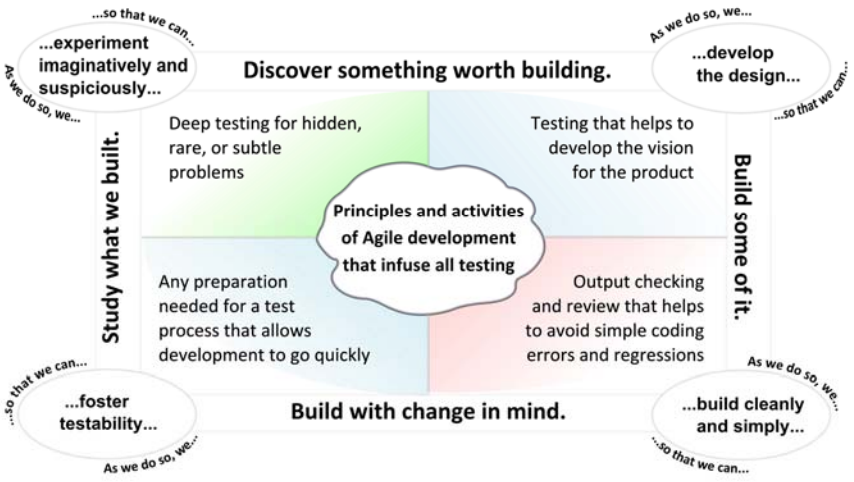
Testing in an Agile Context

- Testing that helps to develop the vision for and understanding of the product
  - discussion, review, thought experiments…
- Testing that helps to refine the design and prevent low-level coding errors and regressions
  - TDD/BDD, unit checks, higher-level output checking…
- Preparation for testing using a process that allows development to go more quickly
  - advocacy for intrinsic, subjective, and project testability*
- Deep testing for hidden, rare, or subtle problems
  - experimentation, investigation, exploratory scenarios, high volume/long sequence tool-assisted testing…

* See "Heuristics of Software Testability", http://www.satisfice.com/tools/testable.pdf

15

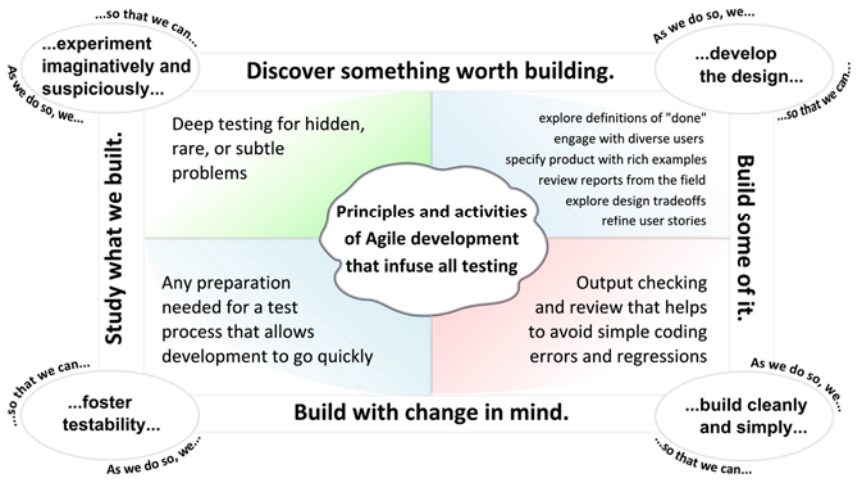
Testing in an Agile Context



Developing the Design

- Exploring definitions of done
  - and expecting that they will be re-evaluated on the basis of new information
- Engaging with diverse users
  - recognizing a broad interpretation of “user”
- Specifying product with rich examples
  - remembering that examples aren’t tests, but they help
- Reviewing reports from the field
  - informing design and testing with real-world problems
- Exploring design trade-offs
  - considering cost and value throughout
- Refining user stories
  - developing rich models of the product in its context

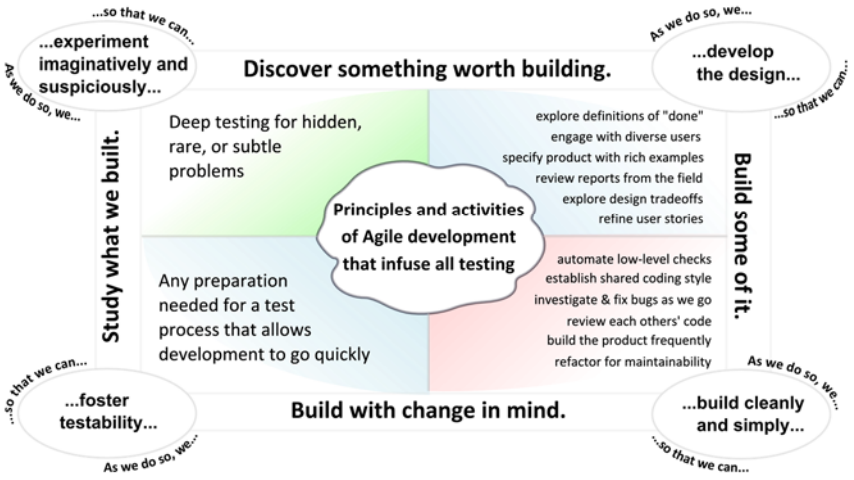
Developing the Design



Building Cleanly and Simply

- Automating low-level checks
  - producing a vastly more testable product
- Establishing shared coding style
  - reducing the overhead of interpretation
- Reviewing each other's code
  - swatting bugs before they even get into the house
- Building the product frequently
  - making it possible to obtain constant feedback
- Re-factoring for maintainability
  - simplifying while anticipating change
- Investigating and fixing bugs as we go
  - swatting more bugs before they hide behind the drywall

Building with Change in Mind

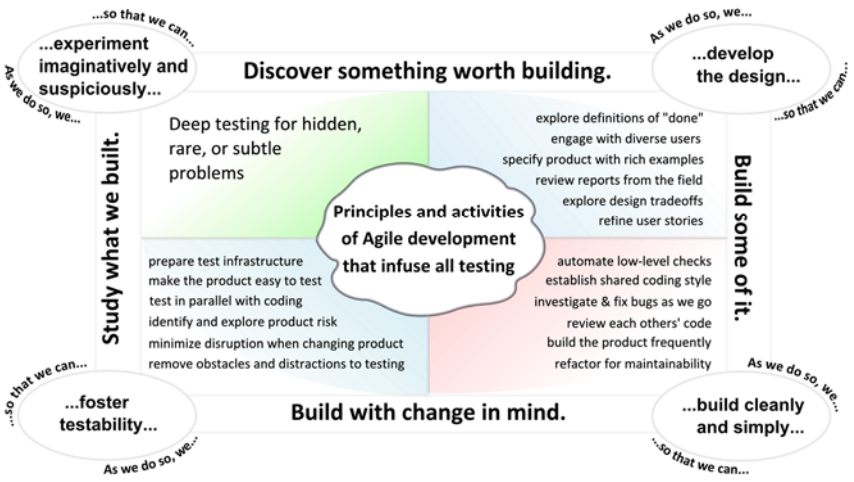


Fostering Testability

- Preparing test infrastructure
  - selecting and developing tools and environments
- Making the product easy to test
  - advocating for visibility and controllability
- Identifying and exploring product risk
  - digging up buried assumptions
- Minimizing disruption when changing product
  - making testing a service, not an obstacle
- Removing obstacles and distractions to testing
  - addressing issues so testing can go quickly and smoothly
- Testing in parallel with coding
  - accepting anything our clients want tested at any time

18

Fostering Testability

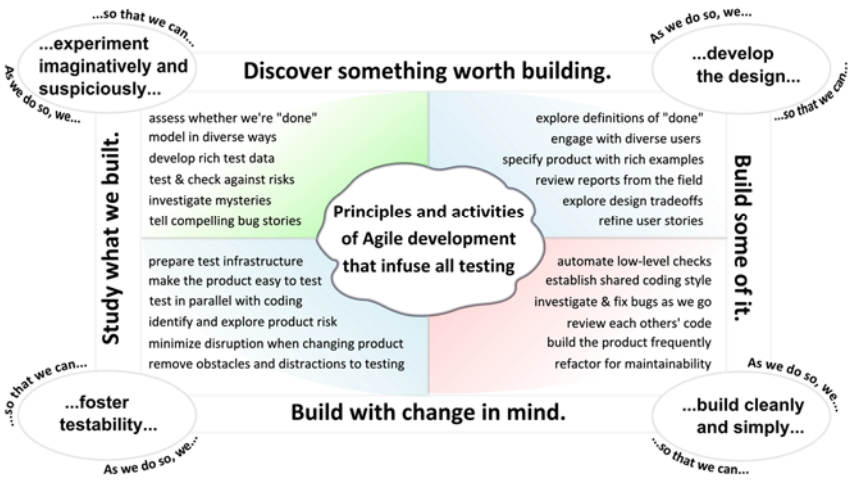


Experimenting Imaginatively and Suspiciously

- Assessing whether we are done
  - recognizing how we *might not* be done *yet*
- Modelling in diverse ways
  - covering the product from many perspectives
- Developing rich test data
  - testing for adaptability, not just repeatability
- Testing and checking against risks
  - focusing testing on what matters
- Investigating mysteries
  - probing hidden, subtle, or rare problems
- Telling compelling bug stories
  - providing context to show how bugs might threaten value

19

RST's Agile Quadrants in Detail



Development isn't linear!

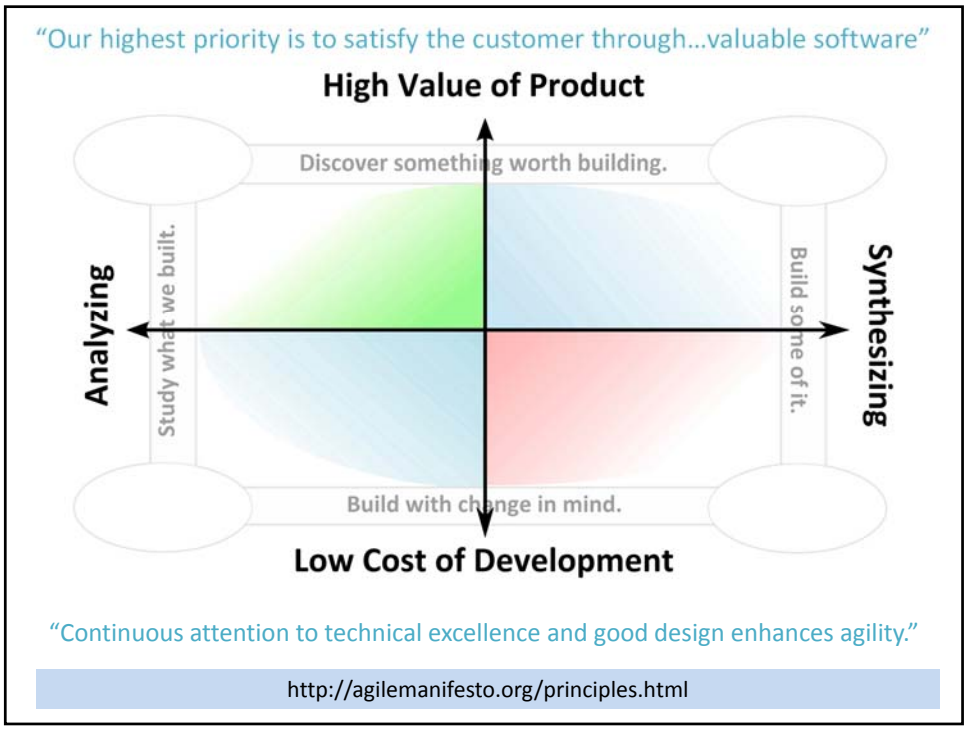
Discover something worth building...
Develop the design so that we can build some of it.
Build cleanly and simply so that we can build with change in mind.
Foster testability so that we can study what we built.
Experiment imaginatively and suspiciously so that we can...
discover something worth having built.

Although there is a cyclic tendency to these activities, they also overlap, combine, and support each other, in big loops, small loops, sudden turns, and epicycles.

Like swirls from stirring a cup of coffee…

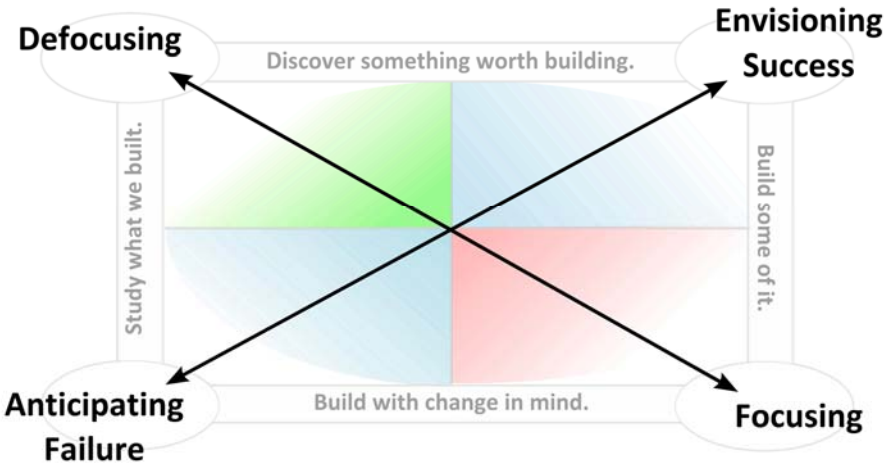
…not like being tied to the hands of a clock

20

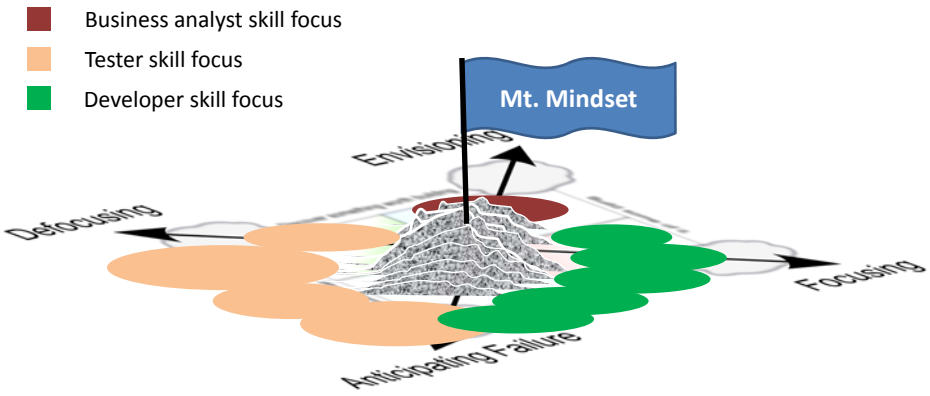


21

Productive Polarities



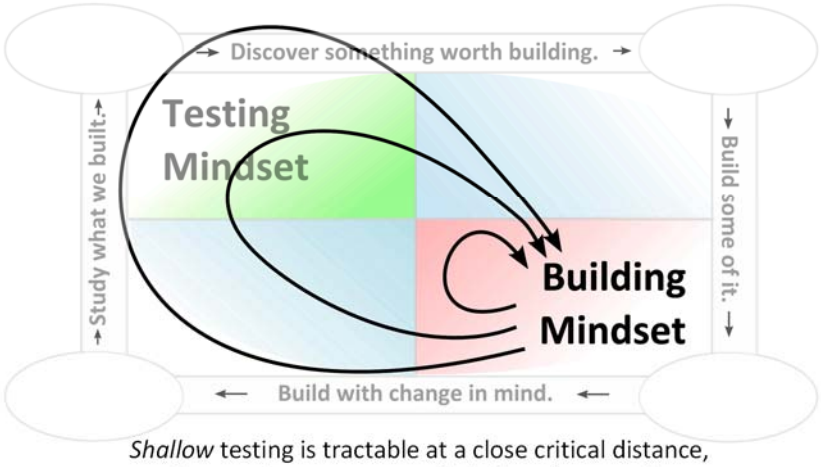
A Central Obstacle Divides Work



NOTE: We do NOT claim that this work *must* be done by different people, or that the people *must* have different roles. We DO claim that roles on an agile team (collaborating with each other) are a powerful heuristic for solving the mindset switching problem.

Critical Distance

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



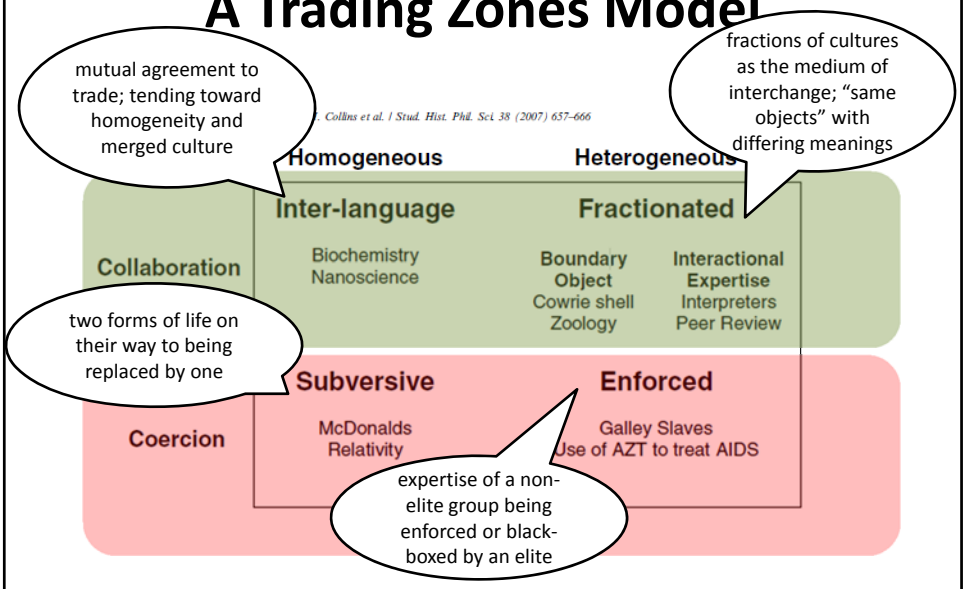
Shallow testing is tractable at a close critical distance,
whereas deeper or naturalistic long-form testing
tends to require or create more distance from the builder's mindset.

Trading Zone

Peter Galison introduced the notion of a trading zone in science as a situation
wherein people from different disciplines try to work together
despite their very different and incompatible concepts and language.

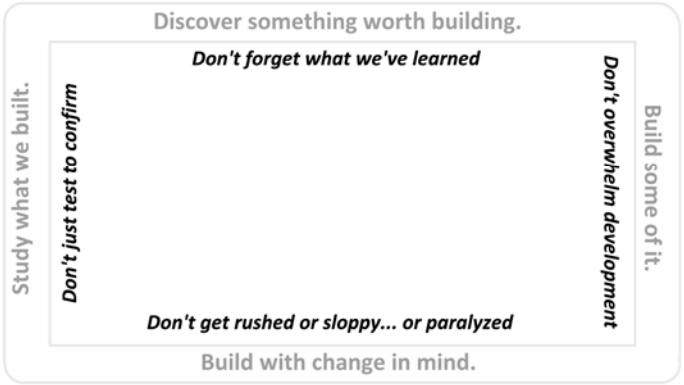
23

A Trading Zones Model

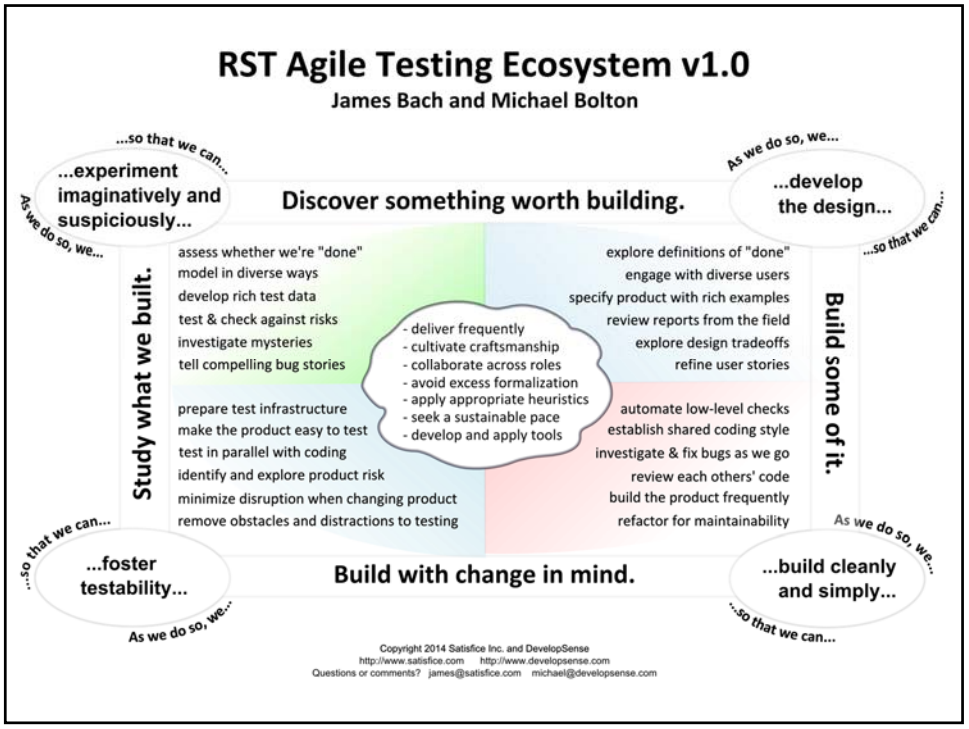


Collins, Evans, and Gorman, "Trading Zones and Interactional Expertise", *Studies in the History and Philosophy of Science*, 2007.

Some Caveats



24



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

25

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/

The ~~REAL~~ Agile Testing Quadrants

(as we believe they should always have been)

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

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

(with helpful comments from International Society of Software Testing members: Anne-Marie Charrett, James Lyndsay, Simon Morley, and Ben Kelly; and graphic design help from Mary Alton)

26