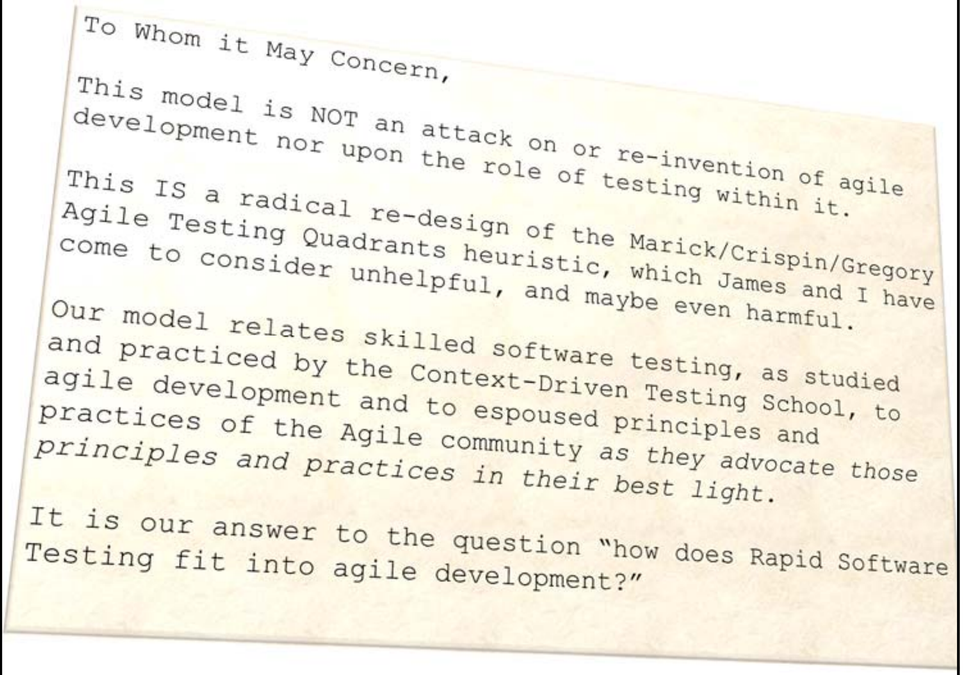
The ~~REAL~~ Agile Testing Quadrants
(as we believe they should always have been)

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

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

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


To Whom it May Concern,

This model is NOT an attack on or re-invention of agile development nor upon the role of testing within it.

This IS a radical re-design of the Marick/Crispin/Gregory Agile Testing Quadrants heuristic, which James and I have come to consider unhelpful, and maybe even harmful.

Our model relates skilled software testing, as studied and practiced by the Context-Driven Testing School, to agile development and to espoused principles and practices of the Agile community as they advocate those *principles and practices in their best light.*

It is our answer to the question "how does Rapid Software Testing fit into agile development?"

What is the testing role?

- To test is to evaluate a product by learning about it through 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 us up as scapegoats.
- Because this is already a very challenging portfolio and skill set. Adding anything to it distracts and dilutes tester 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!
- If I am a tester, can I make quality better?
- Sure!
- If I am a goalie, can I score goals, too?
- No rule against it!
- If I am a janitor, can I offer suggestions to the CEO?
- Why not?
- If I am not the driver of a car, can I grab the steering wheel?
- Feel free!

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, a 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?
- Why not? **But** his role is not necessarily to listen, or to comply.
- If I am not the driver of a car, can I grab the steering wheel?
- Feel free!—**if** the driver is incapacitated. Otherwise, ask nicely.

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

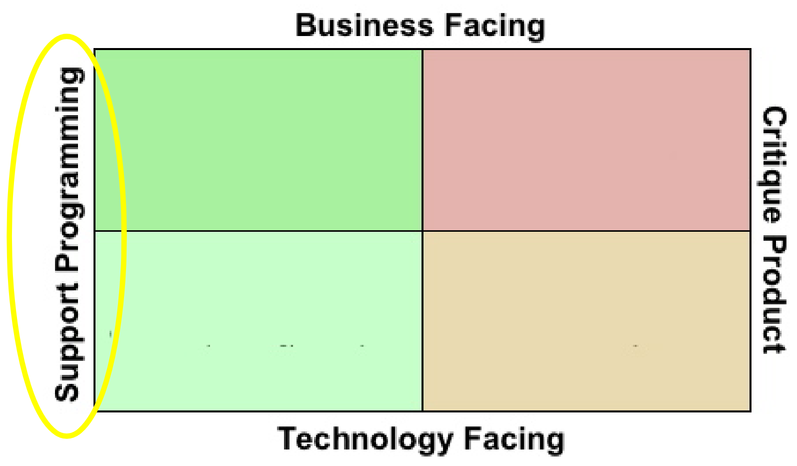
- If I am a father, do I HAVE to do fathering???

Yes.

Our Problems with the Quadrants

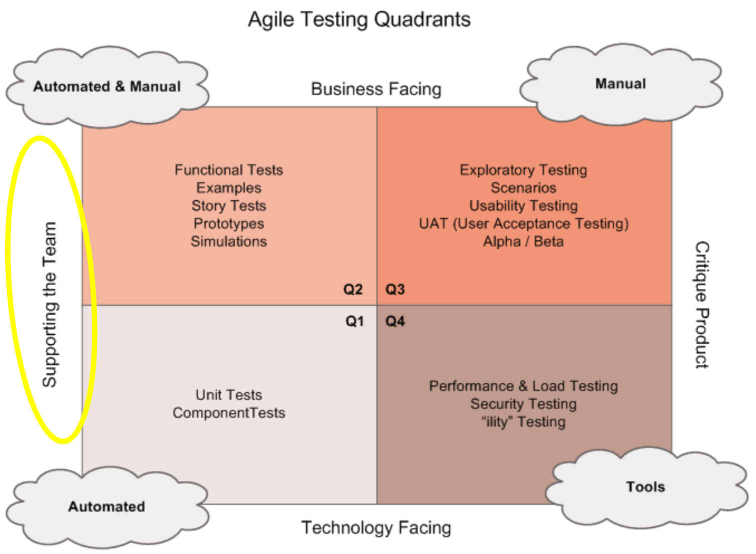
- James encountered the quadrants first in 2003 or so, when Brian Marick explained them to him; I started to hear about them shortly after that.
- I participated in the Agile Testing Mailing list, which seemed to exalt processes and tools, but not talk about *testing* much.
  - There was lots of talk 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.
- **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.

Crispin & Gregory's Version



See Crispin & Gregory, *Agile Testing*

Why You Might Like the Old Quadrants

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

- Our replies
  - We can have generic test strategies that don't misrepresent testing.
  - We can do **way** better than this—going much deeper.

Our Primary Motivations to Revise

- People have been asking us for years how Rapid Testing fits with Agile Testing. We need a bridge.
- The Quadrants are ten years old or so, and we've learned a lot about how to describe and frame our work.

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. This also 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.**

Testing is more than checking

- Both versions confuse output checking (which is completely automatable) with testing (which is not).
  - **Just like programming, testing is a live thought process. Checking lives inside it, just as compiling lives inside programming.**
  - **To be fair, this is a very common misconception, and not just in the Agile community; and to his credit, Marick refers to "checked examples", which he got from Ward Cunningham; honour is due there.**

Tools and techniques are everywhere

- The Crispin/Gregory version makes 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.**
- Both versions pin certain techniques and approaches to certain quadrants.
  - **(Any test technique or approach may relate to any quadrant– which represent overarching tasks and goals.)**

Reification Fallacies

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

- "To test" is a verb; a test is a performance, not an artifact.
- 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.
- It is pointless to discuss whether "business people" can "read the tests" because testing cannot be read; only partial representations of testing activity (checks) can be read.
- If you try to communicate testing primarily through writing then you are probably doing it wrong **(and violating Agile principles).**
  - Instead: prefer conversation, description, demonstration.

“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 guessing at the start of and freezing that guess til 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.)

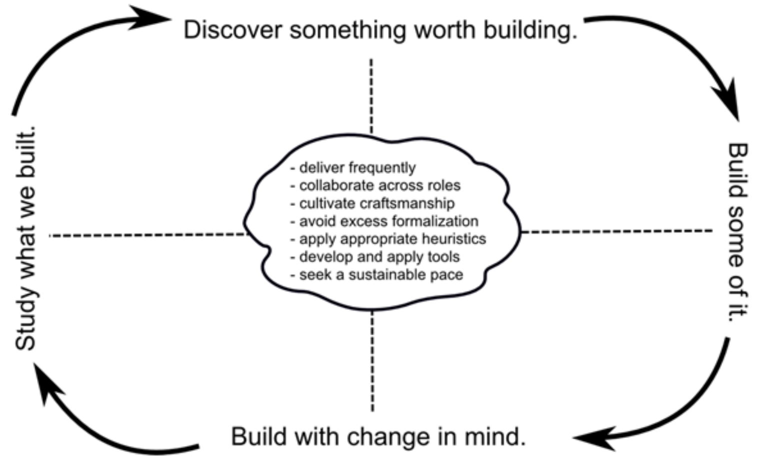
Instead, let’s look at the core heuristics of Agile:
- **continually re-focus on value (to keep producing value)**
- **ply our craft in ways that reduce the cost of responding to constant change (rather than denying change)**

What does it mean to do “Agile Development”?

- Deliver frequently
- Collaborate across roles
- Cultivate craftsmanship
- Avoid excessive formalization
- Apply appropriate heuristics
- Develop and apply tools
- Seek a sustainable pace

Agile Development Heuristic

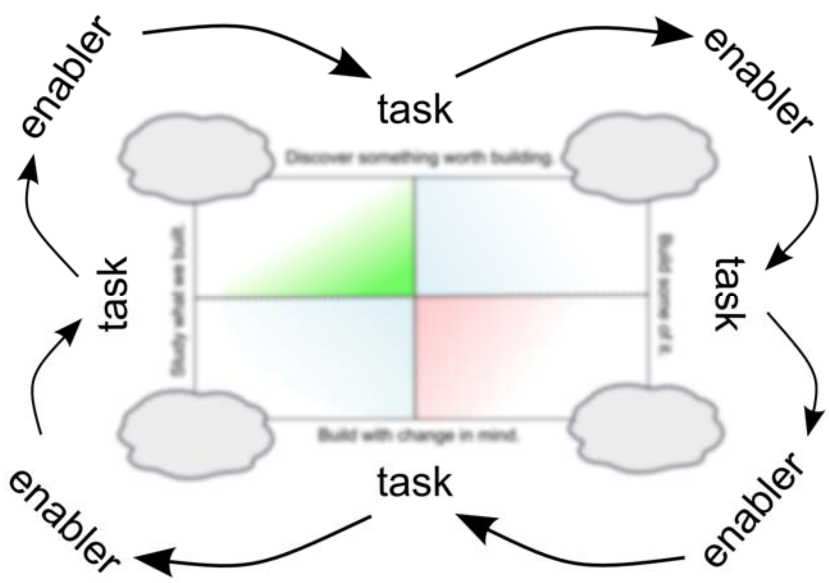
(which ought to be infused with testing)


Discover something worth building.
Build some of it.
Build with change in mind.
Study what we built.
- deliver frequently
- collaborate across roles
- cultivate craftsmanship
- avoid excess formalization
- apply appropriate heuristics
- develop and apply tools
- seek a sustainable pace

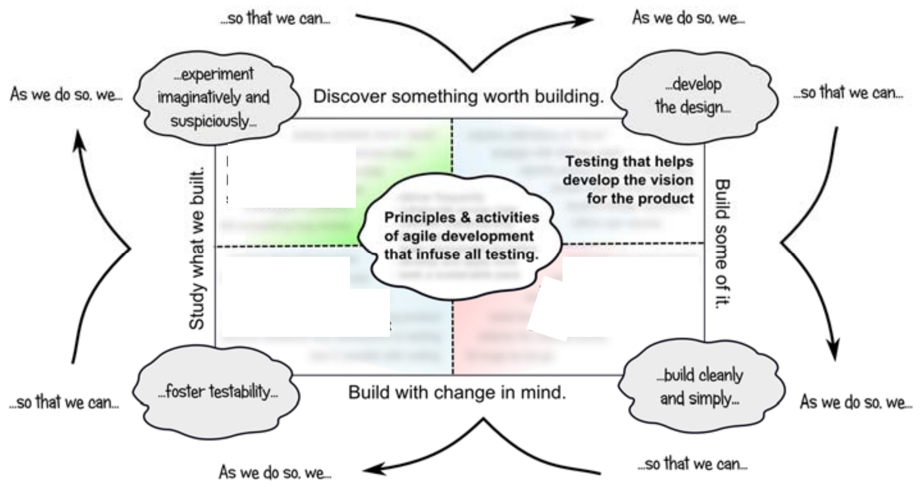
HOW do we do "Agile Development"?

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

Tasks and Enablers



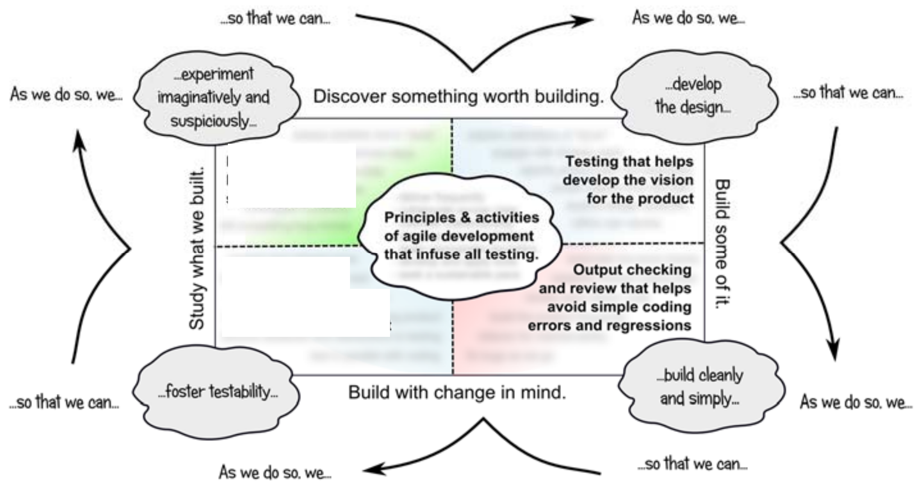
Enablers


...so that we can...
As we do so, we...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
...so that we can...
Testing that helps develop the vision for the product
Principles & activities of agile development that infuse all testing.
Study what we built.
Build some of it.
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

Developing the Design

- Explore definitions of done
- Engage with diverse users
- Specify product with rich examples
- Review reports from the field
- Explore design trade-offs
- Refine user stories

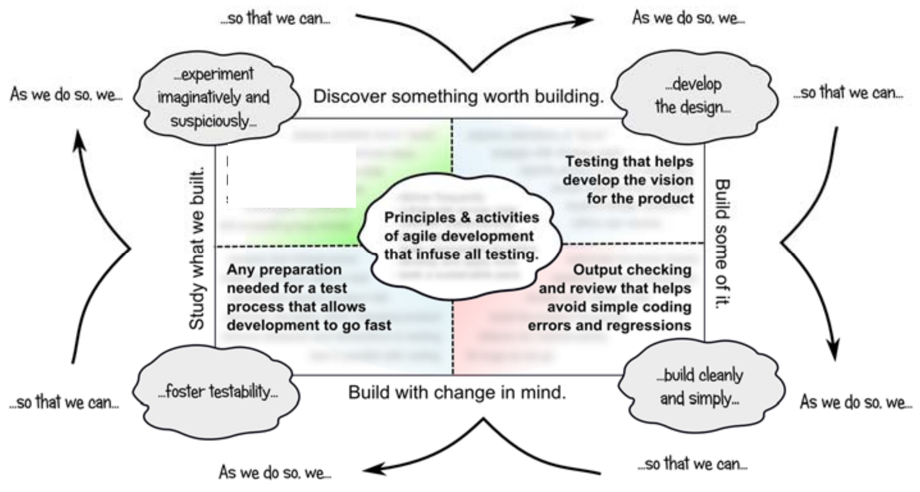
Enablers


...so that we can...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
...so that we can...
As we do so, we...
Testing that helps develop the vision for the product
Principles & activities of agile development that infuse all testing.
Study what we built.
Build some of it.
Output checking and review that helps avoid simple coding errors and regressions
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

Building Cleanly and Simply

- Automate low-level checks
- Establish shared coding style
- Review each other's code
- Build the product frequently
- Re-factor for maintainability
- Investigate and fix bugs as we go

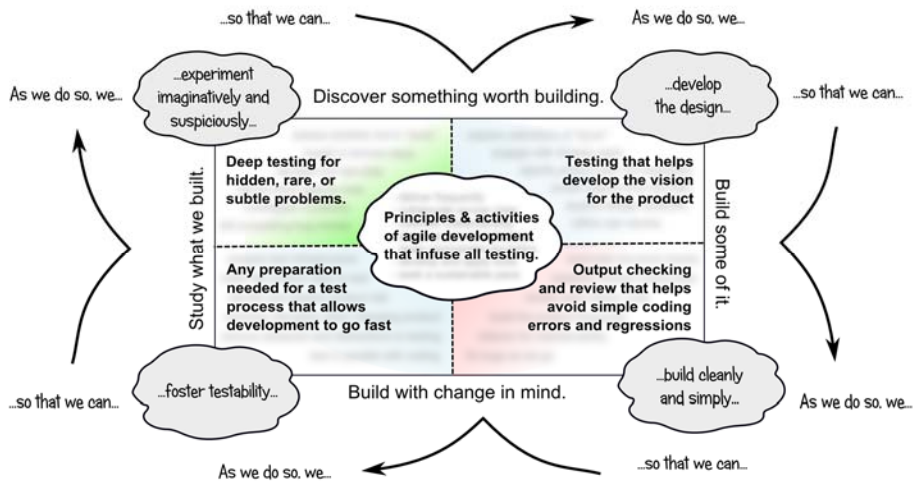
Enablers


...so that we can...
As we do so, we...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
...so that we can...
Study what we built.
Testing that helps develop the vision for the product
Build some of it.
Principles & activities of agile development that infuse all testing.
Any preparation needed for a test process that allows development to go fast
Output checking and review that helps avoid simple coding errors and regressions
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

Fostering Testability

- Prepare test infrastructure
- Make the product easy to test
- Identify and explore product risk
- Minimize disruption when changing product
- Remove obstacles and distractions to testing
- Test in parallel with coding

Enablers


...so that we can...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
As we do so, we...
...so that we can...
Study what we built.
Deep testing for hidden, rare, or subtle problems.
Testing that helps develop the vision for the product
Principles & activities of agile development that infuse all testing.
Build some of it.
Any preparation needed for a test process that allows development to go fast
Output checking and review that helps avoid simple coding errors and regressions
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

Experimenting imaginatively and suspiciously

- Assess whether we are done
- Model in diverse ways
- Develop rich test data
- Test and check against risks
- Investigate mysteries
- Tell compelling bug stories

It's not 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. →

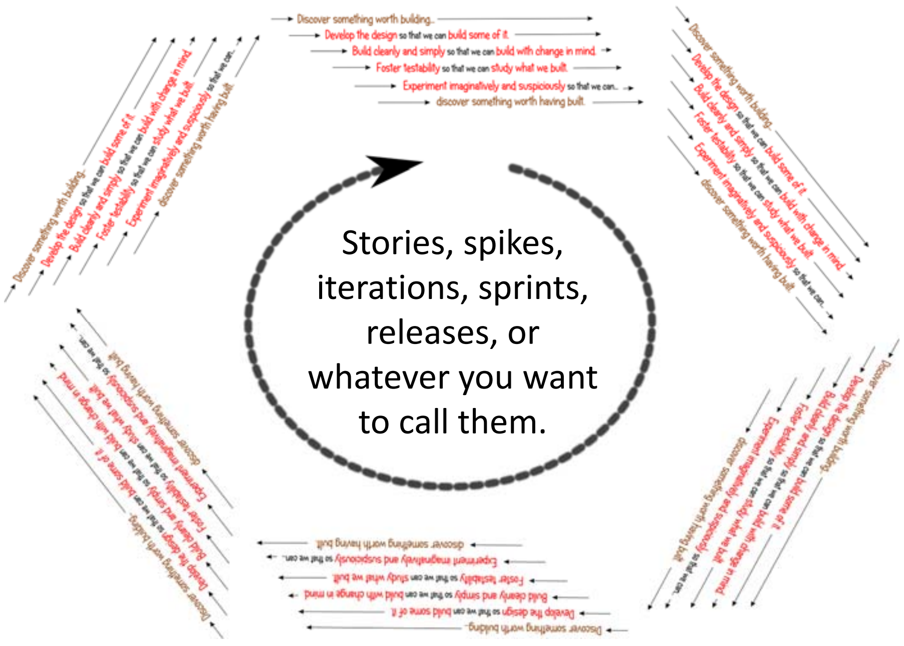
→ Experiment imaginatively and suspiciously so that we can... →

→ discover something worth having built. →

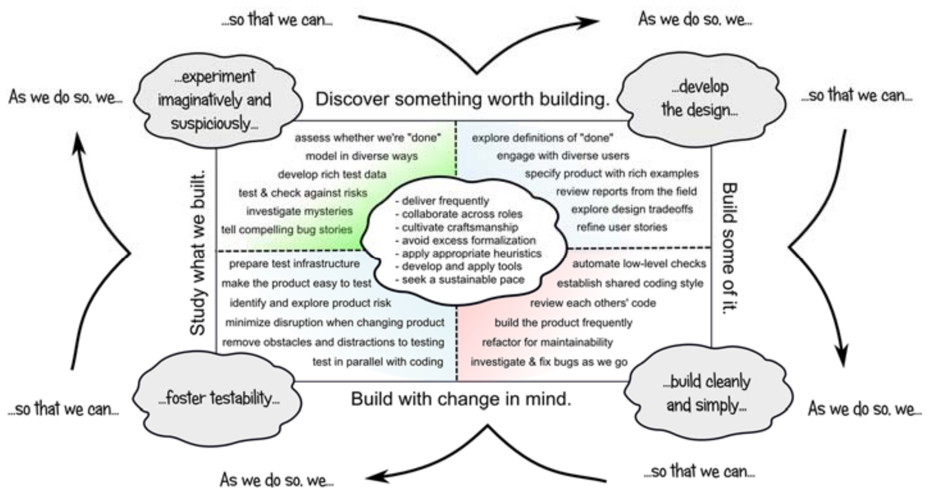
Although there is a cyclic tendency to these activities, they also overlap, combine, and support each other, in big loops, small loops, sudden turns, and epicycles.

Like swirls from stirring a cup of coffee…

…not like being tied to the hands of a clock

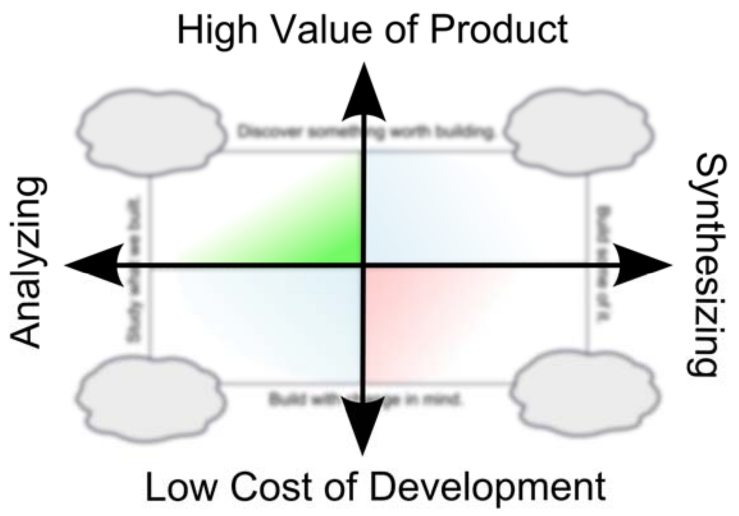

Discover something worth building...
Develop the design so that we can build some of it.
Build cleanly and simply so that we can build with change in mind.
Foster testability so that we can study what we built.
Experiment imaginatively and suspiciously so that we can...
discover something worth having built.
Stories, spikes, iterations, sprints, releases, or whatever you want to call them.

A Rapid/Agile Testing Ecosystem

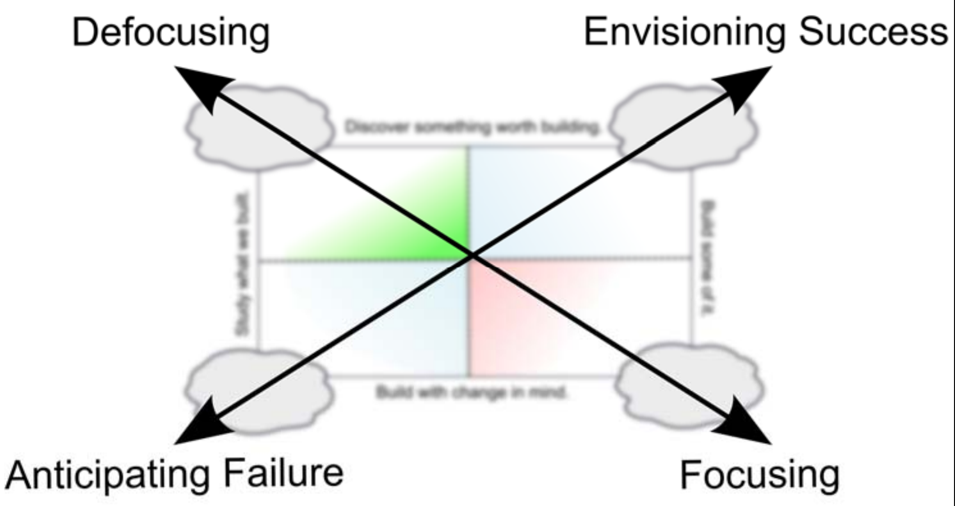

...so that we can...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
...so that we can...
As we do so, we...
assess whether we're "done"
model in diverse ways
develop rich test data
test & check against risks
investigate mysteries
tell compelling bug stories
explore definitions of "done"
engage with diverse users
specify product with rich examples
review reports from the field
explore design tradeoffs
refine user stories
- deliver frequently
- collaborate across roles
- cultivate craftsmanship
- avoid excess formalization
- apply appropriate heuristics
- develop and apply tools
- seek a sustainable pace
Study what we built.
Build some of it.
prepare test infrastructure
make the product easy to test
identify and explore product risk
minimize disruption when changing product
remove obstacles and distractions to testing
test in parallel with coding
automate low-level checks
establish shared coding style
review each others' code
build the product frequently
refactor for maintainability
investigate & fix bugs as we go
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

16

“Our highest priority is to satisfy the customer through…valuable software”



“Continuous attention to technical excellence and good design enhances agility.”

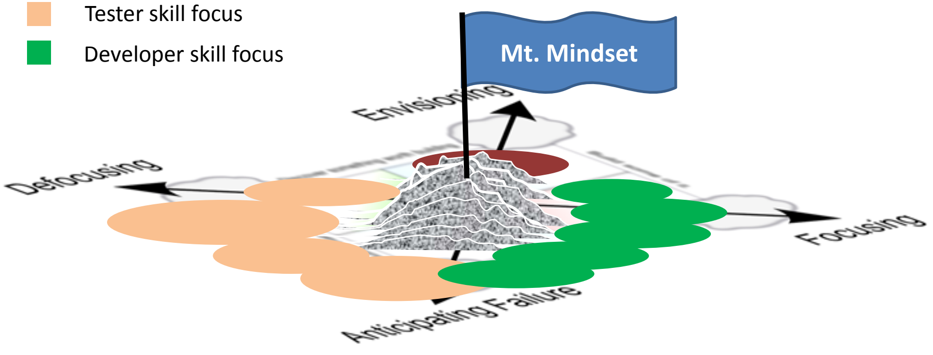


A Central Obstacle Divides Work

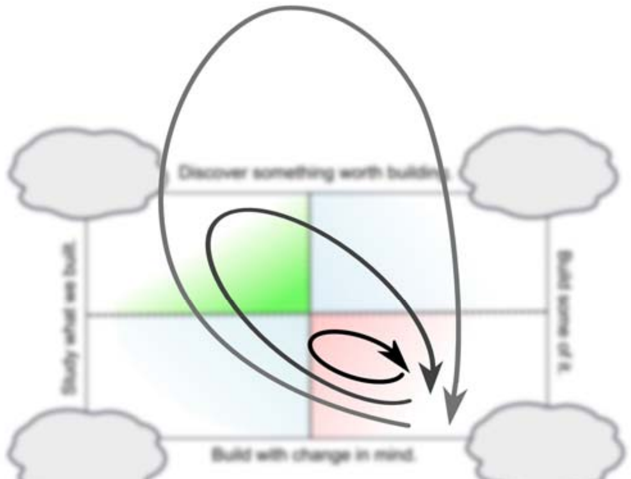
🟥 Business analyst skill focus

🟧 Tester skill focus

🟩 Developer skill focus

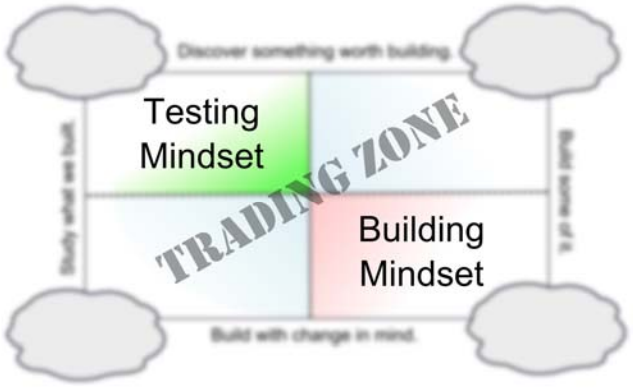


NOTE: We do NOT claim that this work *must* be done by different people, or that the people *must* have different roles. We DO claim that roles on an agile team (collaborating with each other) are a powerful heuristic for solving the mindset switching problem.



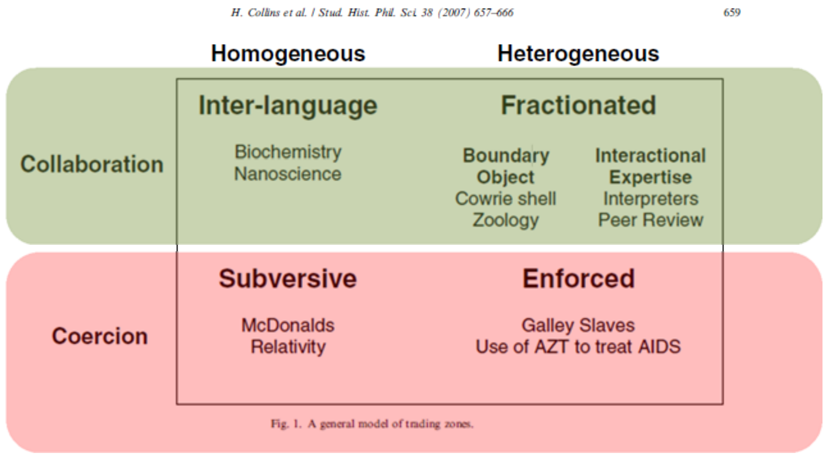
Critical Distance

"Distance" here 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.



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.

Collins' Trading Zones Model



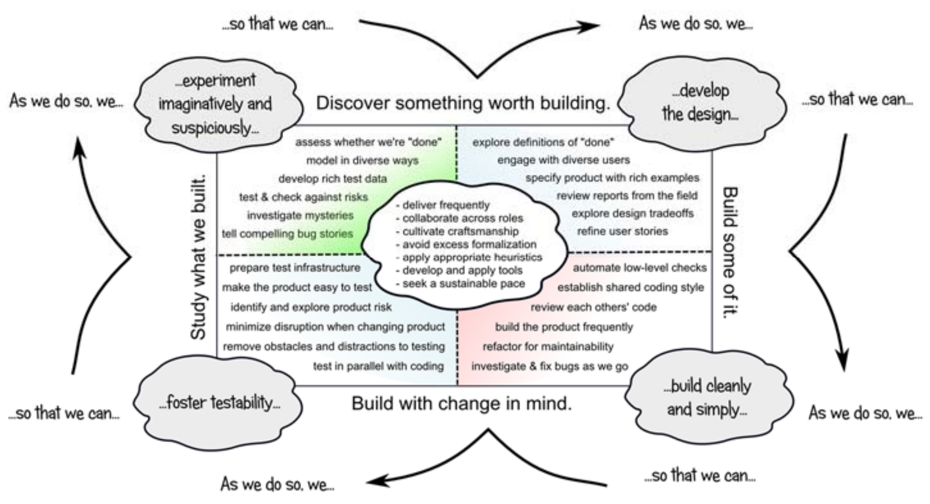
H. Collins et al. / Stud. Hist. Phil. Sci. 38 (2007) 657–666 659

	Homogeneous	Heterogeneous	
Collaboration	**Inter-language** Biochemistry Nanoscience	**Fractionated** **Boundary Object** Cowrie shell Zoology	**Interactional Expertise** Interpreters Peer Review
Coercion	**Subversive** McDonalds Relativity	**Enforced** Galley Slaves Use of AZT to treat AIDS	

Fig. 1. A general model of trading zones.

The RST Agile Testing Ecosystem V1.0

(James Bach and Michael Bolton)


...so that we can...
As we do so, we...
As we do so, we...
...experiment imaginatively and suspiciously...
Discover something worth building.
...develop the design...
...so that we can...
assess whether we're "done"
model in diverse ways
develop rich test data
test & check against risks
investigate mysteries
tell compelling bug stories
explore definitions of "done"
engage with diverse users
specify product with rich examples
review reports from the field
explore design tradeoffs
refine user stories
- deliver frequently
- collaborate across roles
- cultivate craftsmanship
- avoid excess formalization
- apply appropriate heuristics
- develop and apply tools
- seek a sustainable pace
prepare test infrastructure
make the product easy to test
identify and explore product risk
minimize disruption when changing product
remove obstacles and distractions to testing
test in parallel with coding
automate low-level checks
establish shared coding style
review each others' code
build the product frequently
refactor for maintainability
investigate & fix bugs as we go
Study what we built.
Build some of it.
...so that we can...
...foster testability...
Build with change in mind.
...build cleanly and simply...
As we do so, we...
As we do so, we...
...so that we can...

20