Your Agile Blind Spot: 10 Attitudes That Limit Agility

Gil Broza
@gilbroza
Objective, Needs are met by Mind-Set

- Values
- Beliefs
- Principles

People before process
Adaptation
Frequent value delivery
Customer collaboration

Requirements get stale
Emergence is a better response to complexity than planning

Get to “done”
Self-organizing teams
Continuous quality

Attitudes are “low-impact” beliefs

Values support and satisfy Beliefs

Principles are manifested by Processes, Methods, Practices, Artifacts

3pVantage
Copyright © 2017 3P Vantage, Inc.
Be more agile
through *a change in attitudes*
Hold-over attitudes

(= they made sense pre-Agile)

• Discover which ones your team and leadership have
• Understand their effect on Agility and results
• Take away ideas for unseating those attitudes
• Consider Agile-minded alternatives
• Imagine we’re dropping in on an Agile team that’s building an HR Management System.

• Let’s observe the team going through an iteration (in a 2018-style silent movie 😊)
Our first stop: iteration planning
PO.: “It’s time to work on the Payroll Analysis feature. It’s estimated at a few weeks of work. How can we decompose it?”
Delivery team: “We could start with the results in text form; no graphs or trends. That’s about one sprint.”
*ScrumMaster:* “Filling up an iteration with a single large story is quite risky. Can you find an even smaller piece there?”
P.O.: “I wouldn’t bother. There’s no value in splitting it any further.”
Attitude #1

A deliverable must include a bunch of things to be valuable
#1: A deliverable must include a bunch of things to be valuable

- Value is delayed
- Feedback is delayed
- Changing course is hard
#1: A solution has to include a bunch of things to be valuable

Why does this attitude persist?

If we do X much later, we’ll have to throw out most of what we do now.

We’ll do X anyway, might as well work on it now.

If we don’t do X now, we’ll never get to it!

We’ve made commitments!

Users have expectations!

I don’t want to do half-baked work. It will reflect badly on me.
#1: A solution has to include a bunch of things to be valuable

Different perspective:
Not all pieces of the solution are equally valuable.

- Customer **outcome** (“job to be done”)
- Cost of **delay**
- **Testable**, then **usable**, then **lovable**
PO: “The next three items are very important. Can you work on those as well?”
Delivery team: “Even if we split them into sub-stories, there’s no way we’ll get enough done in the sprint.”
PO: “That’s fine.
Please, just do as much as you can.”
Attitude #2

Do as much as you can
#2: Do as much as you can

**Business**
- coding
- testing
- meeting

**Outputs**
- stories in ‘done’
- defects found
- deployments

**Outcomes**
- value delivered
- risks reduced
- assumptions [dis]proven
Workers also have this attitude!

Someone knowledgeable has done all the necessary thinking about the outcomes and the outputs.

We use sprints, and we split stories, but that doesn’t really change the work.

I’ll just work on it until I’m done.

It’s not my place to question the outcomes and outputs; I’m accountable to doing the activities and producing outputs.

Nobody can fault me for that.
Agile-minded alternatives to these assumptions

All of us take part in determining outcomes and outputs

The sprints are opportunities to revisit our decisions based on additional learning and feedback

Constraints, like sprints and WIP limits, force us to think deeply and have more options

Planning = Given our outcomes and constraints, what’s the best thing we can do?
ScrumMaster: “It looks like we have a task to enhance the authentication service ...”
“Dave knows all about that service, so I guess he’ll do this task. Dave, what’s your estimate?”
Attitude #3

Tasks should be done by experts
#3: Tasks should be done by experts

### Small consequences:
- Higher silos
- Lower lottery number
- Narrower tunnel vision
- Integration pains

### Bigger consequences:
- Slipped iterations
- Delays and waste
#3: Tasks should be done by experts

Instead, the team owns the deliverables.

How?
- Co-develop / swarm (“mob”)
- No individual “owner”
- Fewer deliverables at a given time
- Cross-train
- Smaller team / fluid roles

These techniques may not be enough unless you’re explicit and decisive about changing the attitude.
*Delivery team:* “We can reuse a lot of the Jabberwocky framework (with some adaptation). We need Mike, the architect, to make those changes for us.”
ScrumMaster: “Mike is already booked for three other initiatives, so this will just have to wait for the next iteration.”
Attitude #4

We can’t do anything about ...
Quick discussion with your neighbour

Where have you seen examples of this attitude in your organization?
#4: We can’t do anything about ...

Problems for the people:
• fosters resignation to a less-than-great state
• learned helplessness
• cynicism

Problems for the business:
• reduced engagement
• local optimization

Usual suspect: org and team structure
#4: We can’t do anything about ...

**Tips:**
- Make empowerment and boundaries explicit
- Analyze what global optimization would take
- Look elsewhere in your company for ideas
Some time during the iteration
**ScrumMaster:** “Hey Tom, what are you working on?”
Tom: “I'm refactoring the Employee class, it's really hard to work with.”
ScrumMaster: “Tom, we’ve committed to work on several stories in this sprint. Please focus on that.”
Attitude #5

We need to keep producing
#5: We need to keep producing

Finished product matters

A lot of work, too few people

Work gets harder and slower

Keep those people producing

No time or appetite for investing in the product

Small problems turn into big problems
#5: We need to keep producing

Alternative attitude:

**Health** – of both product and people – needs ongoing attention.
Manager: “Guys, you’ve been saying for a while that the team is understaffed. so…”
“Meet Emma:
We’ve just hired her.
She’s joining the team today.”
Attitude #6

Only managers decide who’s on the team
#6: Only managers decide who’s on the team

Jeopardizes productivity, ownership, ... and happiness
#6: Only managers decide who’s on the team

**Instead**, involve the teams in determining their makeup.

Interview with target team

Empower the team to:

**VETO**

Self-selection
Our next stop: the standup (Scrum) meeting
Jen: “Yesterday, I did the back-end for the Taxable Benefit story. My part’s done.”
Attitude #7

Everyone should focus on their part
#7: Everyone should focus on their part
What does your team or company do that reinforces this attitude?
#7: Everyone should focus on their part

What typically holds this attitude in place

- I need to know what each team member is doing today
- I don’t need help
- I can’t appear to need help
- Collaboration is expensive and not really necessary
- I have enough on my plate
#7: Everyone should focus on their part

The Agile view:
Your users expect your product to make life better for them. **When everyone takes responsibility for the outcome,** users benefit, and its makers benefit.

The key shift
Dan: “ScrumMaster, I’m a free resource now. What should I work on next?”
Attitude #8

Someone else determines what I work on
#8: Someone else determines what I work on

Lots of perceived authorities:

- PO
- ScrumMaster
- Manager
- Technical lead
- “The Business”
- Coach
- Sponsors
- Steering Committee
How to start addressing this

- How different is decision-making in this team vs. the wider community?
- Who is uncomfortable with the prospect of greater collaboration, self-organization, and consensus?
- What’s going on that rewards collaboration, and what discourages it?
The review meeting
PO: “Are you guys done with the Payroll Analysis story?”
Abby: “It’s done.
The QA just needs to test it.”
All the subtasks are done
The QA just needs to test it
Still needs refactoring, but it works
We just need to finish the documentation
The rest is easy
We have it working on Chrome
Just waiting on a code review
The rest is easy
Attitude #9

It’s done (well, almost)
#9: It’s done (well, almost)

- Have you produced enough to make some users happy, or get useful feedback?  
  ... or ...
- Have you rushed, cut corners, or produced cruft?
- Have you done the heavy lifting, and expect others to take care of “the tail”?  
That’s Agile-friendly, though maybe “done” isn’t the best adjective.

That’s not okay.

That’s not okay.

These tactics hurt teamwork and productivity
#9: It’s done (well, almost)

Simplify, plan for less

Focus (limit multitasking!)

Everyone takes responsibility for the end product

Define “done” meaningfully
Our last stop: the retrospective
ScrumMaster: “Team, several cards here state that our deadline is impossible. How do you suggest we tackle this?”
Leanne: “You’ve raised that point with management how many times? Forget it, they’ll never listen to us.”
Attitude #10

Management ...

... will never allow it

... will never get it

... will never change

KEEP CALM
AND
JUST
FORGET IT
#10: Management will never...

Your organization’s agility is limited by its culture and management system. So work on that. Relentlessly.
The 10 attitudes

1. A deliverable must include a bunch of things to be valuable
2. Do as much as you can
3. Tasks should be done by experts
4. We can't do anything about...
5. We need to keep producing
6. Only managers decide who’s on the team
7. Everyone should focus on their part
8. Someone else determines what I work on
9. It’s done (well, almost)
10. Management will never...

Which attitude will you address first, and why?
Where do attitudes come from? Identity, character, values, personal history, fears, needs, and habits. The stuff that makes us *human*. There are no “best practices” for that. Instead: caring, curiosity, empathy, kindness.
To learn more from me

“Practical Agile Leadership” PracticalAgileLeadership.com
Comprehensive guidance for increasing Agility and improving delivery.
Toronto, October 18-19, 2018
15% discount with code AGILE2018

“Something Happened on the Way to Agile”
OnTheWayToAgile.com (free 20-day email-based program)
✓ How can we commit to dates and allow change?
✓ With so many meetings, how can we focus on work?
✓ How to foster product owner accountability?
✓ Aren’t two-week sprints too short?
✓ How to integrate code reviews effectively?
✓ ... and 15 other burning questions