Essential Patterns of Mature Agile Teams

This webinar explores the patterns of teams that are succeeding at "Being" agile as opposed to simply "Doing" agile. Bob Galen and Shaun Bradshaw draw on their vast experience and interactions to share the characteristics and behaviors of these teams.

The differences between "Doing" vs. "Being" Agile

- **"Doing Agile"** - focuses on tactics, ceremonies, and techniques.
    - **Stand-up:**
        - Looks like status meeting
        - People are late
        - May or may not attend every day

- **"Being Agile" -** focuses on team mindset, leadership mindset, behaviors, organization adoption, etc.
    - **Stand-up**:
        - Everyone shows up and engages
        - In addition to answering 3 questions (What they did yesterday, what they plan to do today, and if there are any blocking issues) team members add a 4th statement saying what they could use help with today. (For those familiar with Shu Ha Ri, this is the Ha stage)
        - Team members offer to pair with people offline to help them
        - Overriding notion of team

The Patterns:

1. Truly Emergent Architecture

- Team architects in "slices"
- Has Sprint 0's as appropriate
- Starts executing once you have part of the path in front of you
- Research Spike necessary if they don't understand what they are trying to deliver
- Comfortable with a demo of partially finished work.

2. Ruthless KISS (Keep It Simple Stupid)

- Attack documentation (plans, designs) in small slices and get feedback
- Continuously engage your Product Owner
- Set aside some negative tests until later sprint(s)
- Fight gold-plating developing (Just Enough) of everything
- For testing, more focused on testing to see what is working and holding off on more esoteric tests

3. Behaving Like a Team

- Includes Scrum Master and Product Owner
- Scrum Master creates safety
- Maximize strengths and minimize weaknesses
- Help each other out
- Succeed or fail – ***as a team***
- No Silos! For example, Development finishes up coding and could help with manual tests (Silos or hand-offs are a sign of "Doing" agile and not "Being" agile)

4. Truly Collaborative Work

- Co-located teams
- Avoid Scrummerfall-like dynamics
- Comfortable pairings
  - **3 amigos**: Developer, Tester, Product Owner
- Team listens to each other, has respect for each other
- Looks for opportunities to work together
- "**We are in this together**" mentality
- Non-technical people are not looked down upon
- Team members increase their knowledge of each other
- Efficiency gained by trust among team members

5. Lean Work Queues

- Limit WIP (Work In Progress)
  - Fewer things "in process" and small tasks
  - Visible workflow
  - Kanban is interesting variant of the "correct" team behavior
  - Allow team to work on 3 things instead of 10 things and get those **Done**

- Blending roles-individuals doing more themselves and handing off less
    - Team swarm to get things done (Highest priority first)

6. Quality on ALL Fronts

- Notion that testing is supposed to save us is not correct
- Build quality in at the individual level
- Focus on craftsmanship and professionalism within the team
- Doing the right things vs. doing things right
- May take longer, but if this is an important feature, we need to make sure to do it right
- Every step along the way, Development and QA are inspecting each other's work
- Just enough quality
    - Prevention vs. Detection mindset for the entire team

7. Testing is Everyone's Job

- Everyone should be willing to help with testing. Team willingly volunteers to do work. Know that feature is not done until it has been tested
- Take into account testing based on estimate. Best case: everyone on team considers both development and testing tasks that have to be done. Over time, developers will learn what needs to be tested in positive/negative scenarios
- Not all testing is functional. Some non-functional testing won't be considered in the estimate
- Test teams write defects and wipe their hands clean. That is not right. Testing is also about doing root cause analysis and verifying fix
- Do what is necessary for the team. No one whines about it, if they have to work on testing, because they know that they need to

8. Active Done-Ness

- Wild west development with no regard to how it really was
- As a team, decide what will make us comfortable releasing this. Should not just be that it passed. How good is the story? Have we met our goals for sprint? Ensure that everyone is as satisfied as they can be prior to the release
- We are all in it together to get the thing done
- Done, done, done-Everything is done

9. Product Ownership takes a Village

- Environment where the entire team “owns” the Product Backlog
    - All team members contribute
- Trust PO but also be willing to Challenge them

10. Righteous Retrospectives

- Make sure that only the team is invited to the retrospective (ask Scrum Master)
- This the place that team knows that they tried their best
- Safe environment
- Realize that at the point in time where they were, they did the best that they can do. Trust that they did the best that they could do
- Find actions that they can take in the next sprint
- Bite off critical thing each and every sprint
- Open and honest
- Ask what they have to overcome
- How can they try new things/experiment to change the landscape? (Quality, demo, planning, everything)
- Team drives energy and not Scrum Master