The Development Team and Quality

- Ole Rich Henningsen, Teamdriven.dk

- Question: “*Where to start?*”
- Answer: ”*It depends*”. ”*Not one answer*”

- What I have (partially) succeeded doing

- The context
    1. As a Scrum master
    2. Bottom-up (not top-down)

How I most often start (and the agenda)

1. The Development Team – “*it’s easy (they think*)”
2. Product Owner – “*the recurrent impediment”*
3. Back to the Team: Being Cross-Functional – “*it’s hard”*

1. The Development Team

it's easy (they think)

The Development Team

1. Pre-requisites
   - Some work to do, a purpose, and some people to do the work

2. Introduce the framework
   - Low-prescriptive nature

3. Form the team(s)

4. GO!
   - Typically short sprints

And then: Impediments, impediments, impediments

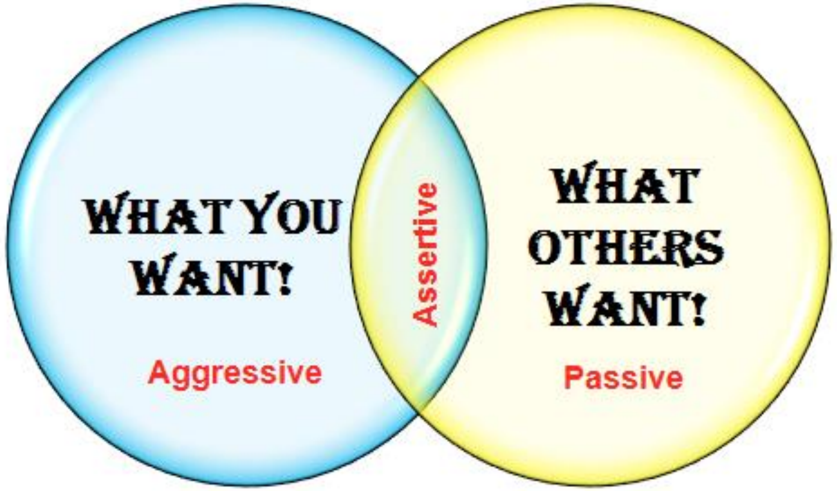
- “I don't have all the requirements”
- Team member monopolizes conversation
- Team member is too quiet, doesn't participate
- Lack of transparency
- Not dealing with mistakes
- Inappropriate refactoring
- Not dealing with impediments
- Blaming and displacing
- Bad estimation
- Not working at a sustainable pace
- Slackers
- Bad listening
- Missing skills
- Over reaching developer responsibilities
- Old waterfall habits
- Conflicts
- Heroes

Information Radiators

- Keep important information visible and transparent at all times and use them actively.
- Information radiator candidates:
  - Development Team plan / sprint plan
  - Build and Test automation status
  - Burn down
  - Definition of Done
  - Retrospective commitments
  - Etc.

Technique: Assertive Communication

- Wait until ”the storm is over”
- Go to your team member and tell them in a factual way
  - No judgement, attack or blame
- Formular: ”*When you [their behavior], then [results of their behavior], and I feel [how you feel]*”
  - Ex. “*When you arrive late, then I have to wait, and I feel frustrated.*”



Anti-Pattern 1: The Hero Developer

- Heroes resist Scrum as focus moves
  - to the team
  - away from the individual
- Heroes almost always ignore quality: Tests, Documentation, Automation
- Heroes are often “hackers”
- Needing a hero means the overall system is fundamentally broken

Anti-Pattern 2: Static Definition of Done

- DoD is the common denominator of quality for the product
- Revisit every Retrospective
    - Continuous Quality Enablement with DoD
- Remember that DoD works at various levels

“Agility has no end-state. Agility is a state of continuous improvement, a state in which each status quo is challenged, by our own will or by external turbulence.” (“Scrum - a pocket guide” - Gunther Verheyen)

Example

1. Code compiles
2. All tasks completes
3. Acceptance criteria from Product Owner is met
4. Refactored. No “clever” techniques and gold-platting
5. Unit test harness
6. Code coverage > 62%
7. Integration tested
8. Regression tested
9. User Acceptance Test passes
10. Deployment package is complete

Anti-Pattern 3: No automation

- Automate as much as possible
- “*DRY for tasks*”
- Plan 10% of the Sprint capacity of the Development Team to be spent on automation of
  - Build
  - Deployment
  - Test
  - Code metrics
  - …

Anti-Pattern 4: Poor Use of Retrospectives

A: “Everything went wrong that time.”

B: “What’ll we do about it?”

A: “Forget that ever happened.”

B: “Good idea.”

Retrospectives

- Follow-up each sprint and retrospective
- Nice addendum to retrospectives
    - Is our DoD increasing in scope?
    - Is our quality improving?
    - Are we learning more from each other?
    - Are we hiding or ignoring anything?
- Vary retrospectives
    - Asking questions
    - Star fish
    - Etc.

2. Product Owner

the recurrent impediment

Anti-Pattern 5: Absent Product Owner (APOP)

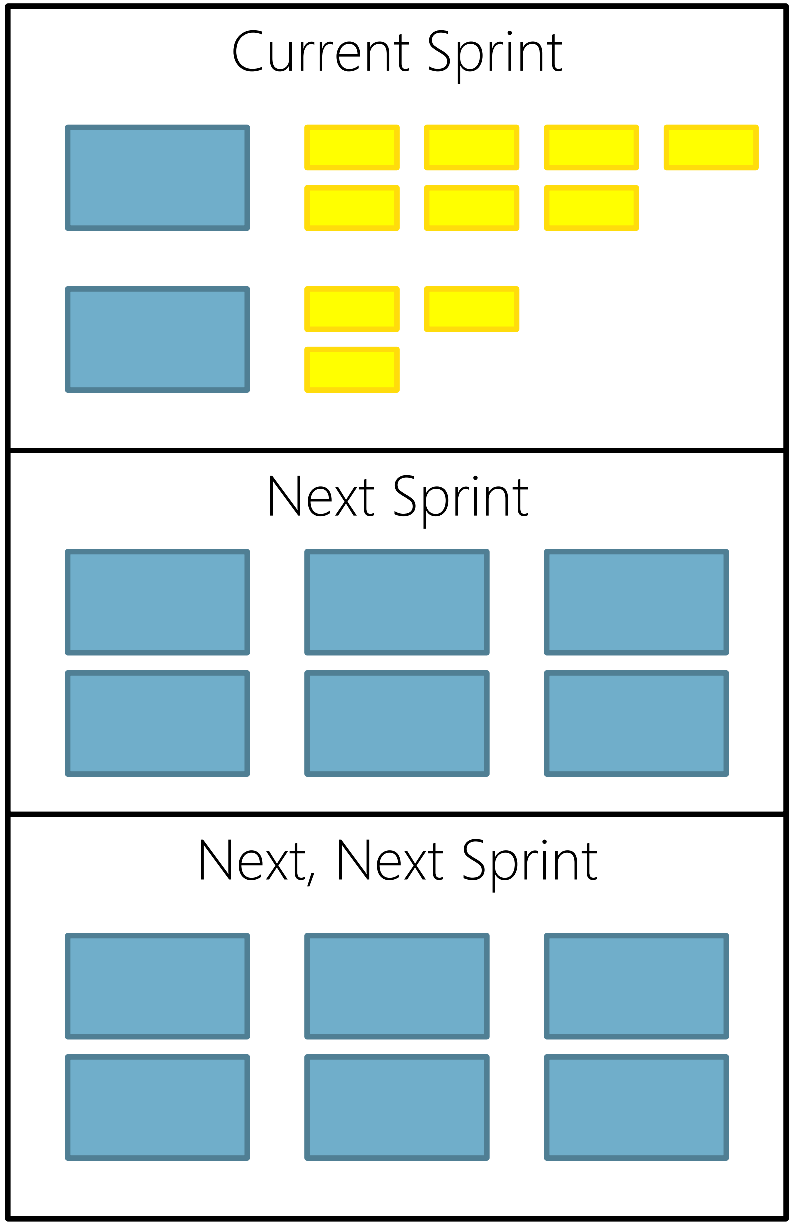
- Aka “the very busy PO”
- Very common and very destructive
- Increases wait time and creates waste
- A quarreling Product Owners is worse
- Feature decisions are often decided by those least appropriate to do so

Keep a Rolling Backlog Projection

- PBIs are estimated and ordered for approximately the next 3 Sprints

- The current Sprint is detailed
  - Broken into Sprint Backlog Tasks
  - Very granular detail

- Next 2 Sprints are understood by the entire Scrum Team
  - Estimated
  - Valued
  - Ordered
  - Loosely planned



PO Team

- “Dogfooding”
- Refinement with Scrum
- Same PO, <u>same rules</u>
- Development Team competency:
    - Business analysts
    - Conceptual designers
    - Graphics designers
- Is working 2-3 sprints ahead

- Waterfall smell?
    - No. Refinement and sprint planning is a parallel and on-going activity

Back to the Team

3. Being Cross-Functional

it’s hard

A tester has the heart of a developer

…in a jar, on his desk

No versus

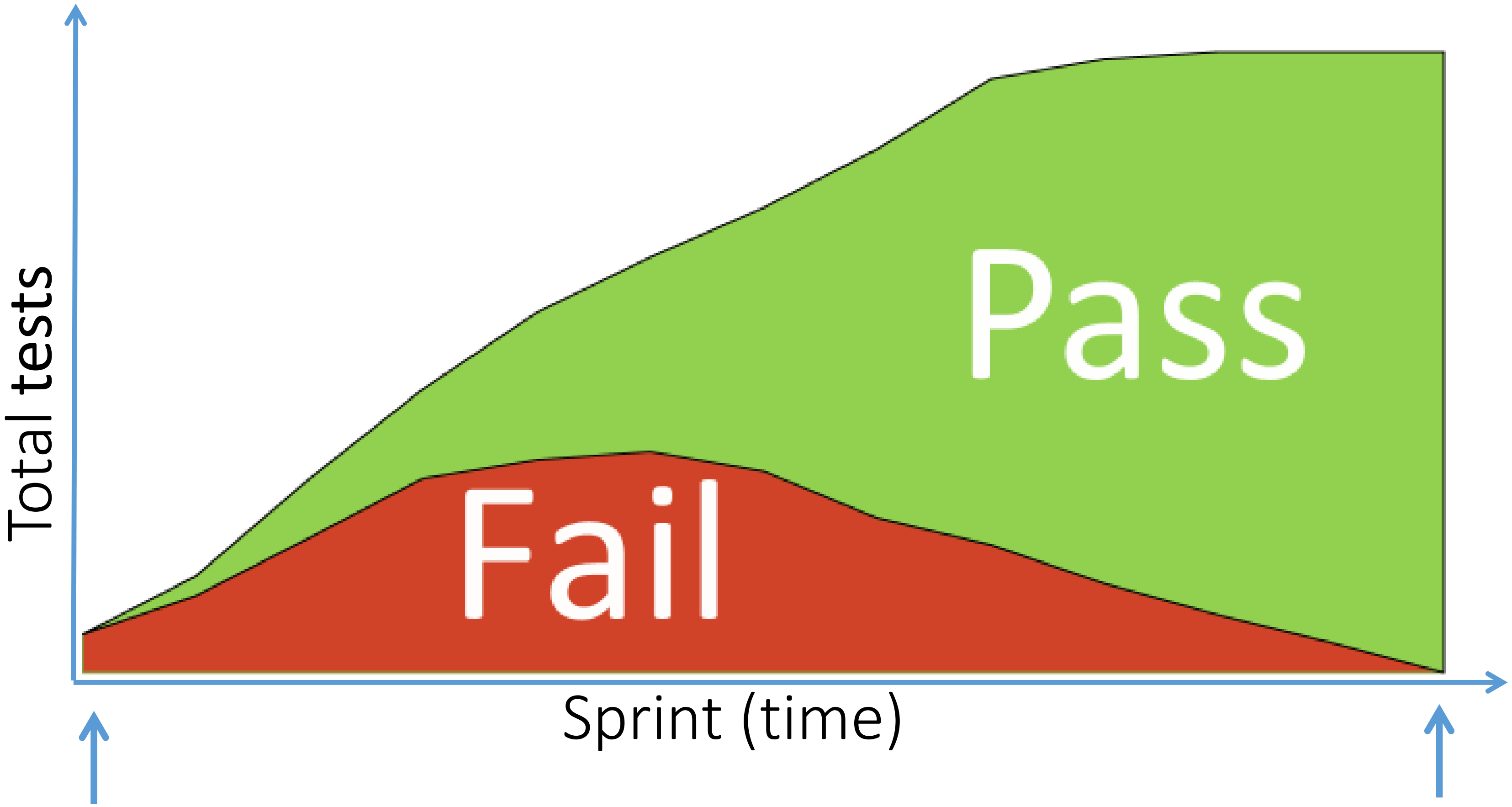
- No programmers vs. testers
- No architects vs. coders
- No business vs. IT
- No development team vs. PO
- No my team vs. your team
- Etc.

Cross-Functional Team

- At least one developer who is capable of performing each type of task in the Sprint Backlog
  - Ideally there is more than one person with this skill
- Cross-functional teams != cross-functional individuals
  - High-performance Scrum Development Teams endeavor to have cross-functional individuals as well

“The opposite of a cross-functional Development Team is a dysfunctional one”

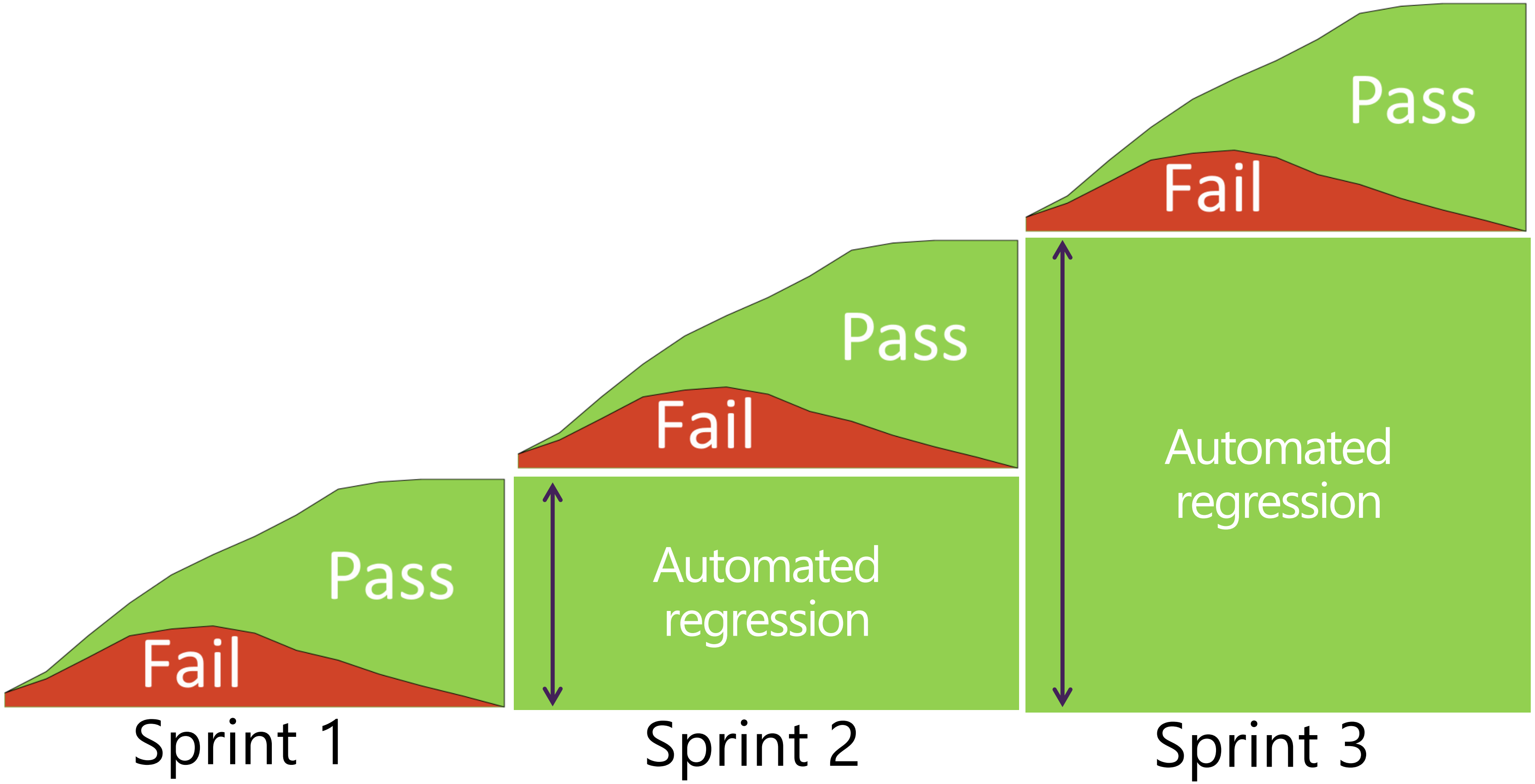
Test competency



Identify acceptance tests at Sprint Planning. These will initially be failing tests.

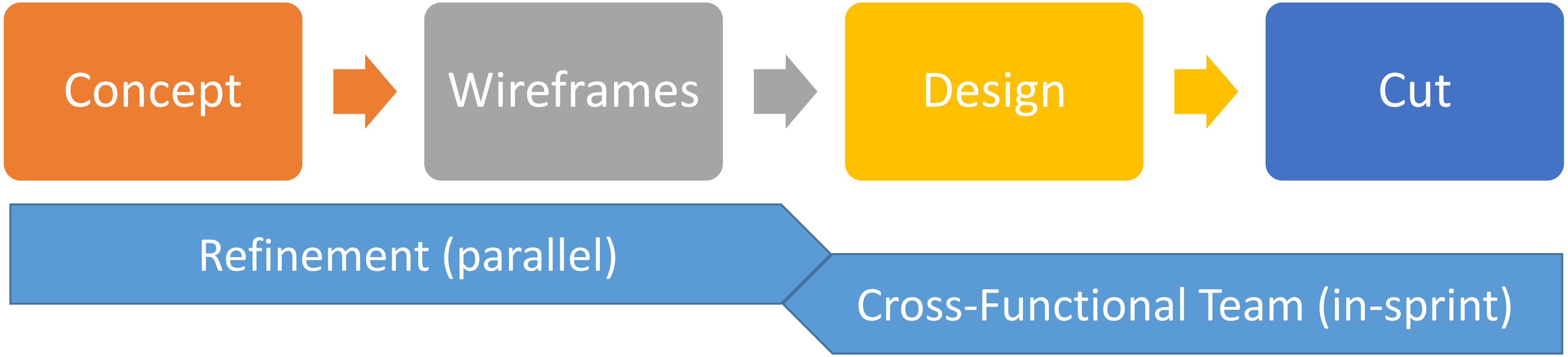
All acceptance tests for all requirements must pass by the end of the Sprint

Test competency



High-performance development teams strive for automated regression tests

Graphic Design competency



Concept

Wireframes

Design

Cut

Refinement (parallel)

Cross-Functional Team (in-sprint)

Spikes

- The Development Team can’t be expected to know how to develop every current and future PBI in the Product Backlog
  - This makes development and estimation difficult/impossible
  - Learning, or at least becoming familiar with, new product domains, components, frameworks, or languages will be required
- Spikes are just such learning opportunities
  - Spikes are investigations, proofs of concept, or experiments
  - The outcome of which is to gain just enough knowledge to be able to give the Development Team some confidence in their ability to estimate
- Most spikes are small and executed as needed during the Sprint
  - Larger spikes should be created and handled as a PBI

Anti-Pattern 6: Reverting to Bad Behavior

- The low-prescriptive nature of Scrum is the foundation of it’s success
- Scrum is simple, <u>but</u> hard
- Giving up when it feels hard undermines everyone else
- Scrum Teams need time and support to adopt the successful disciplines
- Old waterfall habits seems easy