AGILE IN THE FACE OF THE REAL WORLD

SHANE MCDOWELL
Sr. Business Analyst & Agile Guide, Red Hat IT
Agile & Beyond 2018
WE MEET OUR TRAVELERS

WHO WE ARE

Team of eighteen, including Development, Quality, Operations, and BA functions, in Red Hat's IT organization.

We are distributed across three continents and three US states

WHAT WE DO

Support the delivery of Red Hat's Products and Technologies, including subscription management and content delivery.

Primarily web applications and APIs sitting on top of a host of services, some supported by us, some not.
Red Hat is the world's largest open source software company. Red Hat, as a company and within IT, strongly encourages associates to spend time on different teams to make for a stronger overall culture.

Our team has varied from 12 to 20 members over the last five years, as teams have merged, split, and team members have moved to other departments at Red Hat.

Building apps on top of a number of services introduces a lot of complexity and dependency.
The year is 2013.

Thor: The Dark World and Man of Steel are making people question whether we've reached the limits of viable comic book movies.

Macklemore is making thrift shop fashion popular.

Your company is hearing about this Harlem Shake phenomenon and thinking about doing a viral video six months after the moment has passed.

A Red Hat IT team is practicing some very, very bad Scrum.
Background on the culture of Red Hat IT at the time to be fair to those who came before.

Agile transformations were being led by career project managers who were very good at managing budgets and requirements and all of the good PMP stuff, but had very little Agile experience.

As PMs would cycle through teams, various parts of the Agile transformation would lag or disappear, then come back, then disappear again.

It was almost purely about process. "Doing Scrum," not being agile.
WE ENCOUNTER VERY BAD SCRUM

What made this Very Bad Scrum?

- Multiple work streams for planning and standups
- No concept of velocity
  - Missing story points on at least half of the stories
  - Planning was done entirely by task hours
- Very few stories accepted
- Very few demos, very few releases
- Very little accountability outside of the team

We needed to get back to basics.
Every two-week sprint included five work streams and five sprint plannings. Most team members' time was split across more than one work stream.

Every stand-up included five rounds and each round was full of status updates, not two-way communication.

About half of the stories had no assigned story points because stories were not groomed prior to planning.

Story acceptance was around 40%

There was no accountability outside the team for sprint acceptance.
WE HIKE TO PASSABLE SCRUM

PLANNING & ACCOUNTABILITY

- Increased engagement with our Product Owner and other stakeholders
- Strengthened our commitment with fist-of-five
- Tied highest acceptance to Sprint naming rights
- Instituted regular Retrospectives

STORY IMPROVEMENT & QUALITY

- "As a, I can, so that" story form with Acceptance Criteria and a global Definition of Done
- Introduced the INVEST method
- Groomed every story before planning, including Planning Poker
- Ever-improving quality through increased unit testing and UI automation

After two years of improvements, we were getting acceptance up to 75%.
Planning & Accountability (additional notes)

Two week sprints let us do YYYY-A to YYYY-Z, an idea I stole from my previous team

Fist-of-five at the end of each planning, which exposed a lot of how the team really felt about our commitment

Spring naming incentive

- Team with highest acceptance % for the current sprint named the next sprint
- Avalanche, BOOM, Clarity, Dingoes, Hungry Hungry Hippo, Linux, Zombie Aardvark, etc.
- Sometimes put people in awkward situations, with pressure to finish a story in one workstream to get naming rights, even though he or she might also have a higher-priority story in a different workstream that is not yet complete.

After two years of improvements, we were getting acceptance up to 75%, which was still not great. We could get closer to 100%, but only if we severely under-planned.
WE REACH A CROSSROADS

We needed to be honest about our reality and why it took us two years to get to "passable."

PLANNING & ACCOUNTABILITY

- Too much work in progress, split across too many workstreams
- Too many interruptions, often for Support or Operations activities
- Value delivery was a mystery
- Some of our work didn't make sense in a sprint

STORIES & QUALITY

- Stories were becoming too small, not delivering value.
- Too much time on grooming things we weren't going to work on in the near term
- Frequently creating automation technical debt with carried over stories

The team felt stuck. Where do we go from here?
Speaker notes

Planning & Accountability (additional notes)

Too much work in progress, split across too many workstreams, including Support
- Feature of our team, not a bug
- Tried to limit the number of workstreams at any one time by working on one for a couple of sprints, then switching to another, but that only helped so much
- Creating a story in every sprint called "non-project support time" just to track support time

Value delivery: We had no idea of how long it was taking us to deliver working software to customers, only that we could likely develop and test a story of limited value in about two weeks and then release it at some later date

Not all of our work was created equally, with Development v Support / Operations, but we were trying to track and plan the same way for both. Releases were tracked as a story that would float across sprint boundaries, as needed.

Stories & Quality (additional notes)

Stories were becoming too small, not delivering value, so they could fit into a sprint.
- Fit = be comfortable committing to
- Almost every story would have "spikes" first so that the solution would be so nailed down that it could be committed to in the next sprint
- If we needed to do research before taking action on a story, we had an arbitrary limit on having to wait until the next sprint to bring in the implementation

Too much time on grooming things we weren’t going to work on in the near term, because each backlog owner wanted to make sure they had a healthy backlog without any good view across all of the backlogs
It was time to decide: Do we keep fighting against these constraints, or do we embrace them and find a process and framework that can help us find success and room for improvement?
It was time to decide: Do we keep fighting against these constraints, or do we embrace them and find a process and framework that can help us find success and room for improvement?

Of the seven bullets on the previous slide, three or four could be solved by eliminating sprint boundaries, two could be solved by collapsing workstreams into a single flow, and one could be solved with better end-to-end process metrics.

Management skepticism about estimates going away and relying on counting / consistent story size, but if we aren't able to hit our estimates anyway, what are we really losing? Let's focus on consistency first, then worry about converting that to planning.
WE TAKE THE PATH LESS TRAVELED

WHAT CHANGES DID WE MAKE?

- Removed sprint planning and story estimation
- Mapped our work into two workstreams: Development and Operations
- Resized stories as minimum marketable features
- Imposed WIP limits and established Classes of Service[1]
- Changed from velocity to lead time

WHAT PARTS OF OUR PROCESS DID WE KEEP?

- Review/demo/retrospective every two weeks
- Grooming on a weekly basis, or as-needed based on the number of items in the Analysis column

[1] Heavily inspired by Jesper Boeg's Priming Kanban
Development = Application software that we build, test, and deploy to Production for our end users. All applications were now rolled up to one backlog, forcing conversations across the workstreams and stakeholders to determine which story or stories entered our intake column next.

Operations = Platform support, continuous improvement, and technical debt. We treat this as a task board without many of the trappings of Kanban.

Lead time, WIP, and delivery rate are measured across three sections of our process - start to Production, start to PO Acceptance, Dev / QA start to finish

NOTE: These changes were done in such a way that they could be undone very easily. For example, I kept the conference room reservation for our sprint planning for the first month or two, just in case.
WE SET UP KANBAN CAMP

WHAT WE OBSERVED

- Over the first year of tracking lead times:
  - 70 days from commitment to Production delivery
  - 50 days from commitment to story acceptance
  - 30 days from Dev/QA start to story acceptance
- 40 stories in progress at any given time, ranging from 30 to 60.
- Delivery rate was incredibly variable, but generally on the high side of 0.5 stories / day.

WHAT WE LEARNED

- Obvious opportunities to reduce our lead time, including smaller releases and a tighter WIP limit on our intake column.
- We had no consistency with our execution, which highlighted how difficult it was for us to plan with any consistency.
20-day gap between stories being 'done' and stories being delivered into production meant that the average story was sitting on a shelf, not getting us any value, for nearly three weeks.

20-day gap between stories being committed to and stories starting dev meant we were grooming too soon and needed to lessen the WIP on our Analysis column.

To get smaller releases, we needed to get better with our automation and enlist BAs to help with any manual validation so that they full burden of testing the release through the environment didn't fall on our QE folks.
OUR TRAVELERS BLOSSOM

- Ownership
  - Ask "What makes the most sense to you? Why?"
  - "There aren't that many of us. Let's just talk."
  - You do the work, you demo it
  - Environment of trust in retrospectives
- Quality
  - Quality is part of the hiring process
  - Every member of the team is responsible for quality
    - Collaboration between Dev and QA for proper depth of testing
    - BAs write test scenarios
    - Exploratory test parties
Teams are a group of people, not resources!! You can't physically split them. They have thoughts, feelings, preferences, goals. They have a stake in your team and in your company. They need ownership and accountability.

Ownership
- Trust in retrospectives - starts with me as the agile leader and admitting when I make mistakes

Quality
- Exploratory test parties are where everyone on the team, including those who aren't in the direct path of the implementation, like Ops folks, PO, UX designers, documentation writer, etc., are invited to a "free" lunch in exchange for trying their best to break a release through creative manual testing. It is a really awesome thing, both for increasing quality and building teamwork, that we hope to not have to do any longer, if our releases stay small enough.
OUR TRAVELERS BLOSSOM

- Teamwork & Communication
  - Use tools that bridge geographic divides
  - Be respectful of time zones
  - Celebrate your successes!
  - Quirks of our team
    - Name every iteration as a team
    - Every team meeting ends with "Go Team!"
    - Team Wave
    - "Anything to celebrate today?"
    - Chat bots with karma and sass

- Embrace Change in Team Membership
  - Connections on other teams
  - Fresh ideas from new team members
  - New hire documentation is living documentation
Teamwork

- Video Conference Stand-ups, so everyone has the same level of communication. Every single meeting we hold is via video conference, but we respect people who choose not to turn on cameras.
- As much as possible, hold team celebrations during work hours so you don't exclude those with evening obligations. For our team, remote team members can go do something similar and expense it. Example: Team in Raleigh went to Star Wars and paid for remote team members to see Star Wars at a theater near them.
- Naming Iterations - Brainstorm and vote!
- Celebrations in every stand-up
- We end every meeting with "Go team!"
- Team wave whenever we meet in the main video conference room

Embrace change: We can get mad and frustrated about people leaving our team, or we can embrace it, because it is our reality.
WE CLEAR NEW PATHS

Big Project #1 - App-wide Redesign

- Eighteen-month effort including a nearly complete rewrite of our flagship application to modernize and close feature gaps with legacy applications
  - Built new screens in parallel to existing screens
  - Provided opt-in access to the new experience
  - Gave customers a feedback button
  - Got much better at tracking usage patterns
- Process Improvement
  - First nine months: Six releases
  - Last nine months: Eighteen releases
  - Removed dependencies by building our own services
- Flipped the switch on the whole site with very little fuss
Soliciting customer feedback on every screen was very valuable for us. We had some features that we didn't think were used much, so we initially deferred them. As soon as we let users opt-in and give us feedback, we learned that one of those features was very desired. We were able to quickly drop that into our Analysis column and deliver quickly. On the other hand, we had a couple of features that we never implemented because no one ever asked for them.

Because these screens were a sort of "beta," we were able to push ourselves on releasing faster and faster by getting smarter and more targeted with our release testing, with more leeway if things broke. The more we did without issues, the more we built confidence to continue speeding up and releasing smaller chunks of features.
WE CLEAR NEW PATHS

Big Project #2 - Completion of the Redesign

- Six-month effort to complete the final redesign of our flagship application
  - Continued our practice of building parallel screens and allowing users early access with feedback buttons
  - Increased collaboration with other Red Hat teams for feedback
- Process Improvements
  - Many developers focused on one part of our code forced hard decisions about branching / merging / communication
  - New change management rules allowed faster releases
    - More than one per week!
    - Oh no, we're going too fast!
- Flipped the switch on this portion of the site with very little fuss
The new change management rules removed lead times on change approval and reduced the amount of administrative overhead on each release. We decided to try releasing every story as it was complete, which resulted in four releases over two-and-a-half weeks. It was amazing to see the value delivered that quickly, but without any sort of build/deploy automation in place, a couple of team members spent most of that time just on release activities, significantly impacting our throughput for normal dev/qa activities.
WE REACH PRETTY GOOD KANBAN

WHAT WE'VE OBSERVED

- Over the last six months of tracking lead times:
  - 37 days from commitment to Production Delivery (was 70 days)
  - 28 days from commitment to Story Acceptance (was 50 days)
  - 16 days from Dev/QA start to Story Acceptance (was 30 days)
  - 18 stories in progress, ranging from 13 to 25 (was 40)
- Delivery rate is 0.5 stories / day.
- Very few defects reach Production.
- We are a trusted partner for our stakeholders.

"I've worked at a few places now, and this is the first one where the process doesn't suck."
Numbers are much better, but still plenty of room to improve. Specifically:
- 9 days from story completion to Production delivery will be significantly decreased when we complete our delivery pipeline work.
- The 2+ weeks to develop and test a single story means that our stories are still much too big and we need to look for more opportunities to split them down to their smallest valuable units, even if it *seems* less efficient, because we will get value and feedback sooner.

Delivery rate is still 0.5 stories / day, which is a bit lower than when we started, but much more consistent for planning.

Super-rough planning example: Lead time is 37 days, a new story works through every 2 days (1/2 story per day is one story every two days). A story that is fifth in the backlog should be in Prod in approximately 47 days. 5th story times 2 days per story to get into the workflow = 10 days + 37 days lead time.

We are a trusted partner for our business stakeholders, which we see in how they come to us for additional requests, choosing to fund initiatives through our team.
WE CLEAR NEW PATHS

Big Project #3 - Just-started Greenfield Project

- Completely from scratch, technology-wise
  - Take everything we've learned so far and apply it fresh
  - Early shared methods has forced even more discussions about branching / merging / committing directly to master
  - Pair programming
  - Build pipelines
- New lessons learned
  - Stories too big in the beginning
  - Change can be slow
Some of our initial stories for this project were semi-duplicating some existing functionality in one of our UIs, so our stories were basically, "do what the UI does," but we didn't think about how those screens evolved over time, pulling in data from different services, and how these new stories could do the same. Story 1, pull data from Service A, which is quick and well-understood. While getting feedback on Story 1, work on Story 2 to enhance it with data from Service B.
WE CHART OUR NEXT COURSE

- Team Growth & Accountability
  - Sharing agile responsibilities
  - 3 Amigos
  - Keep celebrating!

- Planning & Delivery
  - Tracking value at the story and feature level
  - Mid-to-long term prioritization and planning

- Technical Innovation
  - Building delivery pipelines for faster delivery
  - Ever-improving automation and proactive monitoring
  - Keep using Red Hat's awesome technologies

- Big Picture Process Questions
  - Do we still need separate Dev and Ops boards?
  - If we move fast enough, do WIP and Class of Service matter any longer?
Planning & Delivery (additional notes)
- Once we do a better job of noting the value of epics at a minimum, we can make better decisions about when to cut off one set of work for another set of higher value. We will be able to provide more context around changes in some of our metrics, too. For example, if our lead time decreases a bit, but it's because we have a couple of bigger, high-value stories go through our flow, maybe that's okay, or maybe we needed to break them up better, but at least we'll have the info to have that discussion.
- We currently write up one-page requirement documents to t-shirt size our upcoming initiatives, and then apply some formulas to get us a rough estimate of effort, but we still have a lot of work to figure out how best to build our roadmaps based on this and then how to plan in smaller work that doesn't reach the initiative level.
THANK YOU!

Shane McDowell
@mcdowellshane
smcdowel@redhat.com