Staying the Course: How Rigid Processes Create Flexibility

Leona Parent Grzymkowski

Leona.Grzymkowski@gmail.com

Abstract

Processes – a developer's worst nightmare. So why did we work to add more of them?!

The answer is that processes give developers a structure to rely on. The processes are the common bond of all development tasks. If done well, processes give tasks a roadmap to completion, eliminating the need to decode and decipher them. In her current role, Leona Grzymkowski has taken on implementing structured quality assurance processes into an already established software development lifecycle (SDLC).

This presentation will outline the path First National Bank *of America* took to change the perspective on how development work is accomplished, who is involved in the process, and how to open the lines of communication between internal groups.

Processes are hard work, but by providing guideposts along the path to a formalized and rigid SDLC process, this presentation will help to reveal the benefits. Throughout this presentation, examples of what First National Bank *of America* has done on their journey to become as streamlined and productive as possible and how they handle unprecedented disruption in their day-to-day work will be highlighted.

Biography

Leona Parent Grzymkowski is a Quality Assurance Leader at First National Bank of America. Her interests include opening communication and fostering connections between developers and users, process documentation, and building clear and actionable processes. Her work experience has given her experience in roles like QA Tester, Development Support Help Desk, Process Analyst, and Data Analyst. She has a Bachelor's of Science in Management Information Systems from Michigan Technological University.

1 Introduction

This paper will walk through the past few years of process implementation and formalization for the software development group at my employer, First National Bank *of America*. It will discuss the state of the development department each year, methods and tools that we tried, and descriptions of our team format. The goal is to provide information on how to implement processes to allow flexibility when it is most needed and explore the benefits of flexibility and rigidity.

2 The State of Things: 2018

2.1 Software Development Lifecycle

At the beginning of 2018, First National was utilizing Phabricator (a web-based suite of development tools) for development task management. There were very loose guidelines on what a task should look like and the sizing of the task. Business Units were typically calling on 'their' developer to complete work for them. We did not have any broad view of how much work we were actually doing, and the recurring issues were masked by a lack of tracking and problem definition. Users had no clear definition of their role in the development process. Much of the business logic was owned by developers, as they had been tasked with designing the systems that supported all business operations.

2.1.1 Roadmap to Change

2.1.1.1 The Phoenix Project

In March 2018, as part of our commitment to staff education and advancement, our department read The Phoenix Project by Gene Kim, Kevin Behr, and George Spafford. It is a novel that discusses the relatable problems of an IT department and their relationship to the business. Reading this book inspired big changes to the way that we handled our development tasks.

2.1.1.1.1 The Kanban Board

As a department, we created a physical Kanban Board, which gave us a way to visualize the work we were doing. Every request that came in got its own card on the board. We divided the board into seven sections: Specify, Implement, Review, Validate, Staging, Done, and Track. Each section has a maximum number of tasks allowed in that status at a time, called a WIP (work in progress) limit. Tracking task progress in this manner allows us to identify bottlenecks in the process, and once we learn about a bottleneck, we can take direct action to reduce the blockage. In many cases, the bottleneck can come from the WIP limit being too low. For example, if code reviewers can only work on one task at a time, a bottleneck will occur for tasks in Implement Done status. To combat this, we would adjust the WIP limit to allow for three tasks to be in Review WIP at one time. We would use that WIP limit and determine if it needs to be adjusted again based on workflow. Over time, we have adjusted and found WIP limits that meet our workflow needs, but are always monitoring the limits and are able to make adjustments as needed. We utilize the WIP limits like smoke detectors – if there is a status that is over the WIP limit, we investigate the root cause of the problem. If there is a fire (a true problem with the WIP limit), we will make the appropriate adjustments to prevent the problem in the future. More likely though, it will be a false alarm, like a smoke detector going off because it is too close to the stove. In this case, the root cause may be that there are outside circumstance to the bottleneck, like someone being out on vacation or waiting for approvals from the business units.

The benefits of adding the Kanban board to our process include aggregation of the work being done across our team, finding the bottlenecks in our workflow, and creating a clear picture about where tasks are in the process for anyone to reference, including business users.

2.1.1.1.2 Categorizing Work

We also took on categorizing our work in four ways: internal improvements, unplanned work, change work, and business work. Understanding where developers were spending their time in these four categories gave us a way to control our technical debt (internal improvements), and to ensure that we were working on the right tasks at the right time to add business value.

2.1.1.1.3 Three Ways

A big focus of The Phoenix Project was understanding the Three Ways as defined by Gene Kim and Mike Orzen. They say “We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.” (Kim)

The first way is understanding the performance of your entire system – understanding the whole puzzle and how it goes together versus just understanding the value of the piece in your hand. By implementing the Kanban board, we finally got a big picture view of what our department was dealing with as a whole.

The second way involves understanding where feedback comes from and how to act on it. The goal is to have short feedback loops, which is known as ‘shifting left’. Shifting left means that the feedback happens earlier in the SDLC, or, if you were looking at it written, closer to the left side of the cycle. The sooner a problem is identified, the less disruptive and expensive it is to fix. We worked to honor the second way by keeping a watchful eye on the Kanban board. If a task appears to be stuck, we worked to address it as soon as possible. Each day, we had stand up meetings to give a big picture view of how the department was doing. It also gave everyone a chance to understand the tasks in motion, and provide or ask for help from peers to complete a task.

The third way is a call to create a culture where it is okay to ask for help and also to experiment and take risks. In our team, there is a big emphasis to work with others and take on learning. For our department, learning new and better ways to do the work is an important part of the process. By always thinking forward, we are able to better serve our customers with safe, dynamic, and maintainable solutions.

2.1.1.2 Jira

In May 2018, our teams began using Jira to track our tasks. After operating with a physical Kanban board for a few months and working to get the hang of the workflow, we decided it was time to refresh the way we tracked tasks. Jira worked well for us because we were able to create a digital Kanban board that reflected all the rules and WIP limits we set up on the physical board. Utilizing digital technology meant that the information was now available to everyone without having to leave their desk to make updates or changes. It also gave us a flexible platform to document task details and do reporting. As we got comfortable with Jira, it expedited the feedback loops from one time per day to any time a person noticed an issue. This faster feedback led to fewer large changes, and more fine-tuning of the process.

2.1.1.2.1 Minimum Viable Product

Around this time, we also started to work towards making our tasks as small as possible. The industry term for this is *minimum viable product*, or MVP. By working towards the smallest change that delivers the functionality requested by the business, we create the fastest feedback loop for informing the next piece. Instead of tackling a task that could take weeks, we started breaking down large tasks into smaller pieces using epics. In Jira, we would create a task as an ‘epic’, which is a large task that can be broken down in to smaller pieces, called ‘child’ tasks. The child tasks are worked each an individually developed a piece of the epic request. This allowed a few things to happen: multiple developers could work on child tasks at the same time, feedback from the first tasks informed requirements for the later tasks, and in many cases, we found that the change was not as large or complex as first thought. All these benefits sped up the process by providing faster feedback loops and sparking more conversation between developers and the business units. Before introducing this concept, we would have tasks that would be hung up on non-crucial details, which could have easily been split into a new task. We also had larger and more risky code deployments, which resulted in large and risky code rollbacks if the change failed.

2.1.1.2.2 Workflow

While we got our Kanban board up and running with Jira, we needed to set up a workflow for how tasks moved across the board. One principle of Kanban is that the work cannot flow backwards. Another is that the work should not be pushed forward, but pulled ahead when the developer is ready to work on it. With these principles in mind, we set up a workflow that allowed work to be put into the next status by simply clicking a button. We also made it difficult to move a task backwards on the board. In setting up the workflow this way, we allowed it to become trackable – we can see how much time a task spends in each status, and how quickly it moved through the SDLC. One of the benefits of implementing this workflow is being able to track the average time for a task in each step of the process. Doing this allows us to focus on making small improvements to areas of the workflow that are significantly higher than others or that begin to take longer than they had in the past.

2.2 Team Structure

Our team of developers and support totaled 14 people in 2018. All developers worked under one manager, and there were no formalized teams for a system or business unit. Each developer, worked on 'their' systems, and supported others who were pulled in to work on tasks in that specialty. Once we implemented Kanban, we knew we would need to reframe our team structure to focus on providing value to our business users. Our department leadership began planning how we could do this, but in the meantime, work carried on as usual.

2.2.1 Red Time

Developers had a rotation of on-call during business days. Each developer was on the schedule to be red-time. For us, red-time was the point person fielding production issues throughout the day. Any emails or phone calls directed to the development group by IT escalation or direct contact were handled by the on-call person for that day. If the problem was a bug fix, the developer would work on that task throughout the day. If the task was deemed to be a long-term problem, a Phabricator ticket would be created to address it in the future. Red-time work sometimes carried over into the next day for developers, since there was less communication with business users, and often times, the process to get a task user tested and scheduled for release was not achievable in one day.

2.2.2 Yellow Time

Similar to red-time, yellow-time would spend their day in the rotation as developer operations. This could involve code-reviewing their peers work, assisting with questions or techniques that other developers wanted input on, or researching new methods, training opportunities, and process improvements for the department. The developer on yellow-time would also work as the red-time backup, in the case that the red-timer was overwhelmed with help requests.

2.2.3 Green Time

For developers, green-time was when they were working on the tasks that they had picked up. Ideally, this time would have few to no interruptions, allowing for continuous focus. Any interruptions or requests received by a developer on green-time would be redirected to the red- or yellow-time developer.

2.3 Roadblocks and Complications

2.3.1 Defining Measurable Goals

The first challenge our team faced in changing our workflow was gaining buy-in from each team member and management. Because software development is not a tangible good, it was important to show what a real, physical impact a perspective shift could have on quality and productivity. We set out to find ways to show the cost of the way we were working, which included time tracking with Toggl, and also worked to quantify the number of hours we spent on tasks and rework. This gave a measurable number: the cost of each task (time spent in hours * developer rate).

2.3.2 Leadership Buy-In

An important part of making these changes to how our department accomplished work was gaining approval from department leadership and company leadership. In our case, we were fortunate to have IT leadership on board, but still needed to create our case for company leadership. As with any new process, there was a learning curve that we needed to account for, but were able to show the efficiencies that would be gained by tracking our work and identifying patterns to solve large problems.

2.4 Benefits Gained

Overall, 2018 was the start of major changes for our team. The benefits we gained included a large scale picture of the type of work that we were doing, a way to measure how long the work took, and the ability to identify bottlenecks in the process. All these things laid the foundation for moving forward into 2019, and for us to identify places where we could improve our processes to be more efficient and to provide clarity to our business users.

3 The State of Things: 2019

3.1 Software Development Lifecycle

In 2019, we pressed forward with our Kanban method of completing tasks. As we became more comfortable, we worked towards our target conditions of small tasks (MVP) and one week maximum in the software development lifecycle. We added a new Jira workflow for Code Review in the last quarter of 2018, and also added two new formal development support teams in 2019: quality assurance and release management.

3.1.1 Release Cadence

In 2019, our development team released an average of 7.44 tasks per week. This number is slightly lower than our 2018 results[1] of 8.03 tasks released per week on average. However, we found that more tasks were making it through the entire process. This means we were selecting more viable and important tasks to complete.

Our department found that it is important to be able to release new code to production throughout the workday. We have no set date or time for weekly releases. Rather, our Release Management team works to deploy code on demand. We are able to release changes during business hours for all repositories except one. We chose to exclude this one repository due to its size – it would take the production system offline for almost every user for around 5 minutes. We deploy this repository off business hours only.

[1] *2018 results only include June 2018 – December 2018. We began using Jira as a service at the end of May 2018.*

3.2 Team Structure

Our teams went through some big changes in 2019. We split our release management role out from the DBA role, added our first glimpse of a development operations team, created a quality assurance role, revamped and grew our code review team, and defined our development teams as distinct business unit support groups.

3.2.1 Business Unit Teams

Business unit teams are comprised of developers who support one designated discipline of the organization. Each business unit team has a team leader, who guides a daily stand up for the group. These teams are responsible for working tasks on the Kanban board, and for communicating and collaborating with the business users who are stakeholders of each task.

3.2.2 Developer Support Teams

3.2.2.1 Code Review Team

The code review process has existed in our department for a while, but in the past, there was no dedicated team. Each developer would be assigned a day to review code for tasks that their peers were working on. The schedule was a rotation and included all developers. Now, with a formal code review team, we have a dedicated team of developers who support the three business unit teams. The developers on the code review team also work as part of a business unit team. Each business unit has dedicated reviewers, and there is a Kanban board lane dedicated to Code Review. Tasks in the Code Review lane can only be set to done by a member of the Code Review team.

3.2.2.2 Quality Assurance Team

The quality assurance team is considered a development support team. This team formed in July 2019 with one person working to implement some quality control processes in the existing SDLC. The first order of business was to implement standards around writing specifications for tasks. To do this, the team provided written guidelines and tips for writing strong requirements. In doing so, we created clearer paths to success, and worked with the business units to make sure they were framing their requests as desired outcomes. We also worked with developers to change the way that they wrote specifications to exclude technical terms that would not be easily interpreted by the business users. By creating specifications that both parties could easily understand, we built trust and rapport between the two groups.

3.2.2.3 Release Management Team

The release management team was, for a long time, one person's responsibility. In 2019, the department leadership divided this role from the database administrator role. At this time, our team worked to make the largely manual and undocumented process more transparent and easy to understand. In just under a month, we created written documents outlining the process of taking a task from Validate Done status through staging and release to production. The next piece of creating a transparent release process was to create a way to digitally track where tasks were, not only in the larger SDLC, but also where they were in the release management process. Once again, utilizing Jira, we worked to create a release management board, which allows developers and users to check where their task is in the process, who is working on it, and when the task is scheduled to be released to production. This board also gives the release manager flexibility for when they are out of the office or pulled into another issue; the backup can easily review the board and pick up where the schedule left off.

3.2.2.4 Business Analysts

In 2019, one business user group has provided a business analyst. We worked to train them on the Kanban board, SDLC, and new process teams. This person has been an excellent advocate for their business unit group. They have worked to create methods of ranking and prioritizing the group requests, which, hopefully in the near future, will help other business units prioritize their work requests to maximize the value of tasks that are accomplished for them. We have a long way to go with the business analyst role, but are hoping to quantify the benefit and provide an example for how a business analyst can streamline the business units' communication with the development team.

3.2.2.5 Additional Support Teams

You may have noticed that none of these new groups directly address production issues (red-time, as it was called before). In April 2019, our team made the decision to have one person triage user requests from help desk, phone calls, and emails. This way, we are able to organize and group similar problems and work toward solving the root causes rather than just putting a temporary fix on them. This also reduces the interruptions developers experience throughout the day. It also helps us understand the cost of unplanned work within our group. Understanding these costs drives efforts to improve initial code that make it to production.

3.3 Roadblocks and Complications

3.3.1 Peer Buy-In

Over the course of the year, we added many new pieces to the development support team. It was important to build trust with both developers and business users that these new processes, although a little more work, would be worth it. The new support teams needed to show that the process changes could provide more accurate task descriptions, better quality code, and constructive feedback for both developers and business users. We worked to open the lines of communication with all task stakeholders, and offered guidance and coaching for the new processes. These measures helped develop the necessary and important buy-in of our peers.

3.3.2 User Buy-In

Another big test of the new processes and methods we implemented was the users. Because the majority of the changes we made in 2018 were internal to the development group, there was little change to the user experience. However, in 2019, we worked to empower the business units to own their domains and to take more ownership in development tasks by collaborating to write the specification for each task. We also looked to business users for explicit approval on testing plans. As we rolled specification requirements out, we worked to guide and teach business users what we were looking for. As with the development team, it was important to build trust and provide open communication. We are still working on providing documentation so that the software development process is as transparent and defined as possible.

3.4 Benefits Gained

In 2019, we continued to improve the software development process. By utilizing the Kanban board, we were able to identify some ways to improve our process. We chose to break the release manager role out as a defined role, which lead to improvements around the clarity and communication of the release process. We also chose to define a code review team – previously, developers were asking any peer on the team to review their changes. Now, there is a small team, still made up of peers, but each dedicated to a particular business unit. This brings a more focused review by someone who is familiar with the existing code base. One last improvement was the addition of Toggl. Toggl is a time tracking tool, and by creating categories for each department, we are able to show more accurately how much time the development team spends working on tasks for each business unit. We are also able to show how much time we spend during each Kanban phase by utilizing labels for the Toggl entries.

4 The State of Things: 2020

4.1 Software Development Lifecycle

In 2020, our SDLC remains largely the same as it was in 2019. We continue to intake tasks after the business unit has prioritized them, and work through each stage on the Kanban board. We have stabilized throughput of tasks quite a bit, and have settled into Kanban WIP limits that allow an optimized workflow. At this time, it is not likely that we will make large changes, but we will continue to utilize feedback and metrics to make small changes and tweaks. The next change that will be implemented is a formalized test plan requirement for all tasks, and a review and approval method for testing plans. Our team also has a goal of creating a robust regression testing suite to further ensure the success of new code entering production.

4.2 Team Structure

4.2.1 Development Operations (DevOps) Team

In 2020, we have begun to work toward the formalization of a development operations (DevOps) team. This team will include many of the teams and functions that already exist, including quality assurance, release management, production support, and even some other groups like database administration. The DevOps team is still a work in process at this time, but combining the efforts of these groups should really drive change and increase our ability to support developers and business users. We are also looking into how we can apply these efforts to make improvements for our system administrators, help desk, and backend support teams. Unifying the whole Information Technology department will allow us to provide a better service and experience for our business users overall.

4.3 Roadblocks and Complications

4.3.1 COVID-19

In March 2020, COVID-19 tore across the headlines. Gradually at first, and then all of a sudden, it was knocking on our door. Michigan public universities switched to online instruction, public schools closed down, and soon a shelter in place order was issued for our state. In all of this, our helpdesk and IT teams worked aggressively to outfit our many in-office workers to become remote. Overall, they achieved preparing 92% of our workforce to work remotely. Of course, everyone being remote all at once could be a big problem – how could our development group prepare? Fortunately, because of the effort and investment we have been putting in for the past two years on improving processes, creating development support, and creating a reliable and predictable SDLC, we have been able to continue serving our business users without much interruption at all. In the first week of remote work for all employees of the bank, 14 tasks rolled out to production with no rollbacks. We continue to follow our processes, collaborate with business users, and provide clear and consistent support for developers and business users alike.

4.4 Benefits Gained

By implementing a DevOps team, we have offloaded much of the work involved with Jira and Confluence maintenance, process improvement, and environment maintenance from the development team. The DevOps team is focused on providing the development team with consistent, optimal performance so they can focus on developing code. In the first half of 2020, the DevOps team was able to implement consistent retrospective meetings, update the priority scheme across the organization, pay down tech debt by implementing software upgrades for developers, create a clear and concise definition of each step in the SDLC, and much more. The DevOps team is actively working on engaging developers, IT, and business users to continuously improve our processes for all who are involved in them.

5 Flexibility in Process

Having invested the time and energy into making rigid processes, we are now seeing the benefit of flexibility where it really matters. Now, with the majority of our company working remotely, we are truly putting our processes to the test. We are not flailing, not taking shortcuts, but rather, finding it possible to maintain the same accuracy and communication as when we were all available in the office. Our specifications, code, and test plans continue to be reviewed, business users continue to participate as the drivers of getting tasks to development, and testing continues to be completed. We are able to lean on these processes as a guide, and now, more than ever, we are relying on the communication and trust that we built up to complete necessary work to bring the most value to our teams and customers. If a developer or user has questions about the process there are resources available and clearly documented. If there are questions about the status of a task, they can easily be addressed by looking up the virtual Kanban board on Jira.

In the end, by creating and enforcing rigid processes, we have created a stronger team who works well together and produces high quality software consistently for our business users. By continuing to fortify our processes, we will be able to continue providing exceptional value for our business users and customers.

References

Books:

Kim G, Behr K, Spafford G. The Phoenix Project: a Novel about IT, DevOps, and Helping Your Business Win. Portland, OR: IT Revolution; 2018.

Websites:

Kim, Gene. “The Three Ways: The Principles Underpinning DevOps.” IT Revolution, 22 Aug. 2012, itrevolution.com/the-three-ways-principles-underpinning-devops/.