

Our Journey to Agile IT Operations

Author: Scherezade Khan

Claps: 65

Date: Jul 3

Hi! Iâ€™m the Program Manager for the Global IT team at Klaviyo, and in this post, Iâ€™ll explain how and why we now operate using Agile. While the Agile methodology is common for software development, people rarely think to use it for IT operations â€” and if they do use it, they implement it poorly or in name only. Iâ€™ll tell you how our IT systems and operations team used to work, what drove us to change, what we do now, and our plans for the future.

Problem: Scaling IT as Klaviyo Scaled

In the early days, our IT team handled all work the same. It didnâ€™t matter if the task was an escalation from our employee help desk (which we call SearchBar), the result of a security incident, infrastructure work, or systems administration duties. For the most part, we handled it **reactively**. While this reactive approach worked well when we were a smaller company, it broke down once we crossed a thousand people with hubs in three countries. We also started to outgrow our â€œgeneralistâ€ model, where everyone handled everything, regardless of scope or level of effort.



Noel Keady, me, Fernando Rosario, Wang Szeto, and Kyle Steinike around the corner from where IT sits in Boston!

These two ways of working caused us to accumulate tech debt, making it hard to plan resources and assess risk. At one of the first retrospective meetings I moderated for the team after I joined, we concluded that we were at a crucial inflection point. We needed to learn to identify and plan proactive work while budgeting appropriate time and resources for reactive work. We wanted our global IT team to solve complex problems with the best rather than the quickest solution.

Solution: “Diet” Agile

To address these challenges, we introduced a modified version of the Agile method that we call “Diet” Agile. Agile is a [project management methodology](#) with several implementations, most notably Scrum, which is popular in software development and is the implementation we selected for IT.

I was responsible for selecting Scrum, and chose it based on prior experience. Before coming to IT, I worked in Sales Operations. Like IT, Sales Ops is a technical services team that balances reactive work (e.g. user permission changes or unexpected incidents in the tech stack), and planned work (e.g. building API integrations or developing forecasting tools). Through my own experience as a jack of all trades in Sales Ops at a prior company, then in my first role at Klaviyo as a Business Analyst in Sales Ops, I saw the benefits of applying a methodology that emphasized prioritization, estimation and delivering value iteratively, rather than as a fixed “big bang” deliverable.

There is, however, always an inherent need for reactive work in teams like IT and Sales Ops – in those instances, Scrum can be limiting because it is not possible to estimate, prioritize, and define *all* work before the sprint begins. Simply put, stuff will come up mid-sprint, and things must be shuffled around. Hence the term “diet Agile”.

The first thing we did was migrate our IT Systems Engineers’ work tracking from a combination of Google Docs and Slack to Jira. This allowed us to scale into Scrum in a centralized way and, as an extra benefit, made it easier to collaborate with other groups such as Information Security.

Once IT was familiar with Jira, I coached my colleagues on how to write user stories for planned work. (We adopted user stories for both discrete, standalone increments of work and as parts of larger projects.) I introduced a simple framework for writing out business requirements, which entailed answering four questions:

1. **What** are we trying to do?
2. **Why** are we trying to do it?
3. **Where** is the problem observed?
4. **Who** is impacted?


Nothing was radical about that, but it forced a degree of deliberation and stepping back, which was hard when we worked in pure reactive mode before.

Next, I coached my colleagues on how to size (also called estimating or story pointing) user stories. Sizing is a practice from the Agile method where you assign a point value to a given issue, usually using the first few numbers of the Fibonacci sequence (1,2,3,5,8,13,21) since it’s an exponential rather than a linear scale.

Three key inputs go into sizing: the amount of work, task complexity, and risk (inputs for which can include the number of users impacted by the system change, dependencies on other projects or departments, etc.). The team breaks the work into increments (stories) and estimates the size of each story in points. This forces a shared understanding and encourages shipping value faster and iterating rather than producing a single, fixed deliverable at the end.

For example, we are currently making changes to further enhance our device authentication program, a multi-month project that is large and complex. We broke the project down into 16 stories, half of which were completed in June, half of which will be completed in the next couple of months. We are able to tell our internal customers when a project will be done, and provide early warnings when we’re off schedule.

Addressing the four questions above in the Jira ticket works well for small projects. For projects with greater complexity, particularly ones with dependencies on other teams, we’ve found it’s worth forcing more formality. We go beyond the four questions and use the template shown below. The same business requirement questions must be answered, but there’s also additional guidance on writing out solution design, dependencies, risk, and identifying stakeholders.

Problem Statement	A 1-2 sentence summary of the problem you're looking to solve and why. If you need help arriving at this summary, go to the Arriving at the Why and What section below.
Project Sponsor	The person held accountable for the successful outcome of the project. They unblock bottlenecks, are the final decision maker, the point of escalation, and responsible for appropriately resourcing the project.
Project Manager/Owner	The person who is responsible for overall project planning, management, and execution.
Project Team Members	A person responsible for completing a specific task or task(s) in the project and/or a subject
Project Stakeholders	<p>A person who has a stake in the outcome of a project, typically because (but not limited to the following reasons):</p> <ul style="list-style-type: none"> • they or their team are directly affected by the project • they or their team are dependent on the successful completion of the project • they or their team will benefit from the outcome of the project
Project Link	Please provide the link to the relevant Jira project
OKR	<p>Please identify the relevant OKR that this project corresponds to, as applicable:</p> <p> Sign in to access Google Drive Spreadsheet</p>

Project Overview Section of the IT Project Plan Template. Filling out this section helps the IT person managing the project give a bird's-eye view of their project to all relevant stakeholders.

Solution Scope

Must have	<ul style="list-style-type: none">••
Nice to have	<ul style="list-style-type: none">••
Not in scope	<ul style="list-style-type: none">••

Dependencies

	Dependency	Point of contact
1		
2		

Risk Assessment

What could slow us down or cause us to miss deadlines?	Potential problems / proactive solutions
What are we currently uncertain or worried about?	Potential problems / proactive solutions
What hurdles have we encountered in similar projects?	Potential problems / proactive solutions

Solution Scope, Dependencies, and Risk in the IT Project Plan Template. Identifying solution scope, dependencies, and risk upfront is essential for project planning, especially for technically complex or high-impact work.

The template empowers members of IT to dig deeper into a project with stakeholders before accepting work and streamlines the project's visibility to all parties involved.

Limits of "diet" Agile for IT (and how we solve for them)

As mentioned, a significant portion of our work is reactive, and accounting for this is not possible with the Agile methodology in its purest form. To manage, we do three things:

1. Functional Groups – We subdivide our team into functional groups expected to balance their workload between the ticket queue in our helpdesk software and planned project work. This allows us to do capacity planning upfront and expect consistent story points each sprint.

Functional Group	Project work (%)	Ticket work (%)
SearchBar - The “help desk” team, primarily in the ticket queue.	25%	75%
Escalations - Systems engineers who support SearchBar with tickets they cannot resolve on their own (whether due to complexity or knowledge gap).	40%	60%
Systems - Work primarily on project work, the final escalation point if Escalations cannot resolve the ticket.	90%	10%
Operations - Work primarily on operational functions (no general issue tickets, though in rare instances, projects may originate as tickets).	100%	0%

Breakdown of our IT team by functional group and the amount of project vs. ticket work each group is responsible for.

2. Frequent check-ins – We use meetings (described below) and a Slackbot to prompt team members to share project status, blockers, and whatâ€™s at risk of moving to the next sprint. That way, if reactive work swells and threatens to throw off project work, we have an early warning and can be deliberate about reprioritizing or pushing back.

3. Documenting incidents and reactive work (even after the fact) – We expect that, at minimum, the four business requirements questions stipulated above must be answered on a user story to document reactive systems changes, even if it is after the change has been made or the incident has been solved.

Hereâ€™s our schedule for planning and status meetings:

MON 27	TUE 28	WED 29	THU 30	FRI 31
3	4 BACKLOG REFINEMENT AND PLAI	5	6 SYSTEMS SYNC	7
10	11 SYSTEMS SYNC	12	13 SYSTEMS SYNC	14
17	18 SYSTEMS SYNC	19	20 SYSTEMS SYNC	21
24	25 SYSTEMS SYNC	26	27 MONTHLY RETRO	28

Calendar for our planning and status meetings.

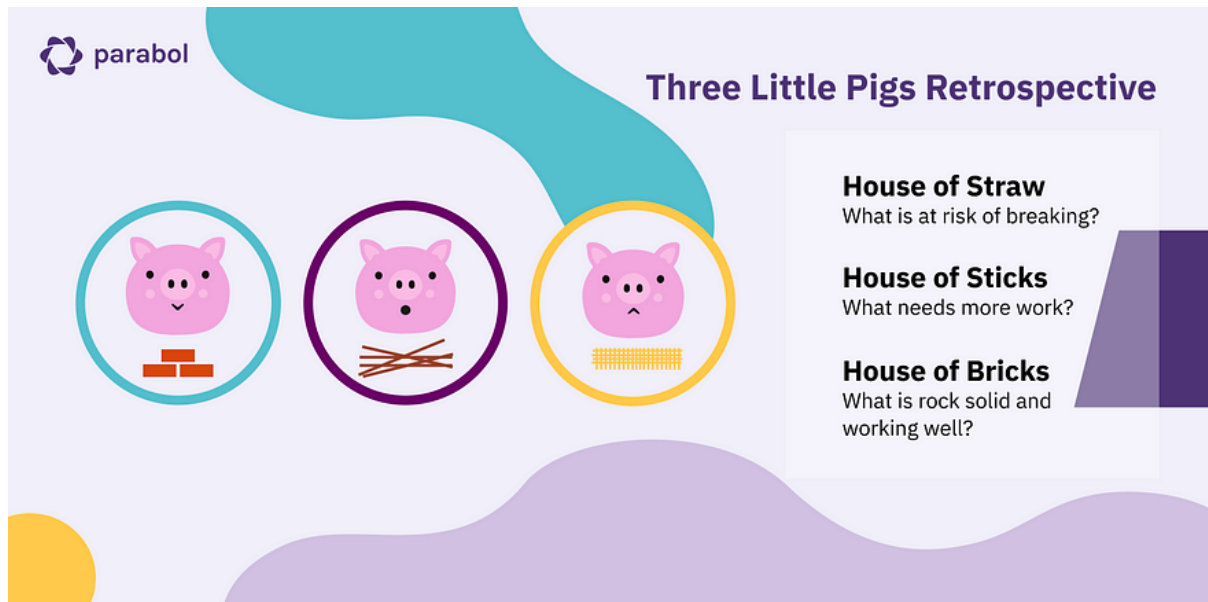
Backlog Refinement and Planning is a monthly meeting used to confirm that work for the next sprint/month (or subsequent sprints) has appropriate business requirements and is ready to be sized, assigned to a user, and pulled into a sprint. The goals are to:

1. Maintain a prioritized backlog of work ready to be pulled into a sprint.
2. Allow team members to ask clarifying questions, break out work that is too complex into multiple stories, and identify roadblocks that could put projects or stories at risk.

During the Systems Team check-in, we answer the following questions:

1. What are you working on this week?
2. Do you have any technical challenges you need help with?
3. Are you waiting on anyone to respond?
4. Do you want to demo something youâ€™re working on?

At the end of each month, I host a retro to review our work for the month, check in on how weâ€™re performing against our commitments, celebrate our successes, and take learnings into the next sprint. I am a big fan of [Parabolâ€™s Retro structures](#), particularly this one!



As someone who in a past life coached sales and customer success reps in process and product adoption, strong change management is close to my heart. We rolled out these changes over several months, onboarding the IT team onto these skills one at a time. My colleagues have proven to be agile (pardon the pun) and quick to adapt, all while offering great feedback to keep improving how we work together.

Moving Forward

We are in the early stages of adopting Agile, and it's already paying off. As an example, we configured SCIM (automated user provisioning) for several high-priority applications more quickly than similar projects in the past because we were more deliberate about gathering requirements upfront, assessing the complexity of the configuration, and setting timelines and communications with the relevant stakeholders. Another example: We automated the offboard IT workflow (as well as several others), which relieved our SearchBar team of many administrative tickets, letting them focus on troubleshooting important issues with users.

Next, I'd like to make two improvements to our adopting Agile program for Global IT. The first is better integrating our help desk software, Freshservice, with Jira so that escalations for system changes are less manual and redundant. The second is modifying our workflow for any system changes that impact above a threshold number of employees so that these changes get more eyes on them.

If anyone has suggestions for either of these or other tips for using Agile within IT, I'm open to them!