nod

Production Process Book

Updated Feb. 22, 2021
Prepared by Alicia Silhavy

CONTENTS

OVERVIEW

The current NodMD production process was lacking in three main areas: ownership, process, and documentation. Onboarding new team members, communicating project status, and meeting deadlines has been challenging due to these reasons. This document details the previous process and upcoming changes to a number of areas in order to uphold nodMD's standard for product quality and team values.

- **Agile management strategy.** Development tasks are sorted onto Kanban boards. In addition to consolidating the Kanban boards, we can implement Scrum methodology to bring more structure and collaboration to the team. Scrum is known for its ability to motivate developers and to provide greater deadline and status transparency.
- **Feature definition.** New features, such as nodReferrals, do not have a structured lifecycle. We can implement a process with clear, iterative steps -- ideation, validation, user stories, design, development, testing, and deployment -- to cut down on redundancies in the final development process.
- **Developer communication.** Developer communication is currently sparse and difficult to trace. Going forward, we will be implementing more frequent, structured communication with our overseas developers in order to maintain project transparency, improve team morale, and decrease failed tests.
- **Jira issue organization.** The current board structure in Jira makes it extremely difficult to see what is in progress, to-do, and done -- one of the biggest advantages of using Jira to manage Agile projects. By restructuring our boards and issues, we can refocus our development team and more effectively communicate project status.
- **Issue creation and structure.** Many of the previously created issues have lengthy summaries (titles) with little to no details in the description. As reported by Vikram, this has left the development team without the details required to complete tasks. Restructuring issues in Jira with more detail and supplemental fields will solve this issue and make it easier to navigate, search, and filter in Jira.
- **Testing, and QA.** Our current testing process is unclear in Jira, making it difficult to understand what is really happening with tickets left in the "Testing" status.

Implementing a partially-automated testing board in Jira will bring clarity to the testing process and encourage multiple tests, cutting back on the number of reopened tickets.

- **Development Operations.** Although the developers' current process is overall solid, we can improve upon it by adding more Jira integrations and automations.
- **Documentation.** The only existing documentation is the unorganized backend/developer documentation in Confluence. We need to expand our Confluence documentation to include end-user support, a credentials knowledge base, and product definition in separate, organized spaces.
- **Client Support Management**. Client support is currently 100% manual and has no clear owner. Atlassian, the parent company of Jira, has a product called Jira Service Management that would allow us to digitize and automate our ongoing client success efforts with a public portal and knowledge base.

SCORECARD

In order to understand the need and reasoning for change, it is important to first understand the company's history. Below is a scorecard showing key responsibilities, their owner, and their score out of 5. Owners include the Nod Product Ownership (PO) Team (Brandi, Dr. Murthy, Alicia, Stephan, and historically Jodie and Dustin), External Developers (Thinkitive), External Designers (User Ten), Vikram, and Hari. This scorecard can be used in the future to evaluate nodMD's product progression and management strategies to further identify strengths and weaknesses.

Responsibility	Owner	Score	Notes
Idea Conceptualization	Nod PO Team	3	New features frequently lack detail and nuance.
Validation	Nod PO Team	2	Some product validation was done, but was lacking in user testing and market research.

Responsibility	Owner	Score	Notes
User Stories	Nod PO Team	3	Dustin created some user stories and requirements, although they lacked in detail.
UI/UX Design	User Ten	2	The wireframes created by external design teams did not last through development -- the designers and developers lacked the detail and testing necessary to create screens that held up over time.
Jira Task Creation	Thinkitive	0	Jira tasks have none of the required detail. The description is often left blank. This should be completed by the Nod PO Team, not Thinkitive.
Development Execution	Thinkitive	TBD	We are unsure of the quality Thinkitive can produce when given sufficient detail.
QA and Testing	Thinkitive	4	The process is messy but does meet basic requirements.
Project Architecture	Thinkitive/Vikram	3	Project architecture is generally documented and, so far, has remained stable.
Project Costs	Thinkitive	2	Although an initial cost estimate was given, it was not tracked through project production. There is a lack of understanding of the developer costs. The Nod PO Team should begin their own efforts to track project costs.
Project Timeline	Thinkitive	1	An initial project timeline was created by Thinkitive. This should be done with requirements given by the Nod PO Team. Project timelines since the original scope have not been created collaboratively with developers and are therefore unreliable.
Documentation	Thinkitive	1	The only existing documentation is backend documentation by Thinkitive. This should be an expanded Nod effort.

Responsibilitiy	Owner	Score	Notes
Jira Organization	Thinkitive	0	Jira is extremely unorganized with boards being used in very non-traditional ways. It is nearly impossible to report on project status with the current setup. This should be managed by the Nod PO Team.
Client Support	Sales and Product	0	Client support is currently 100% automated and has no clearly defined owner. It seems that patients, facilities, and providers call in to either a sales representative or a product owner to find answers regarding the platform. Generally, the answers to these questions are unclear due to the factors above.
TOTAL		**21/65**	

AGILE MANAGEMENT STRATEGY

Previous Process

Previously, the production of nodMD software was managed under a loose Kanban framework. The production process was owned by Jodie, Vikram, and Tanji at Thinkitive.

Future Process

We will be implementing a new strategy that includes both Scrum and Kanban framework, introducing the concept of sprints. Kanban methodology makes sense for our support and bug work. These tasks, which are usually shorter, stand-alone tasks, change frequently in priority and often need to be worked on rapidly. It makes sense to organize all of these bugs in a single Kanban board.

For new features, adjustments to existing features, and any other ticket that does not require emergency-level support attention, adding in the Scrum methodology will benefit the team with structure and organization. The Scrum methodology focuses on organizing tickets into two groups: the backlog and sprints. The backlog consists of all tickets that need to be completed but do not have a set date, sprint, or position on a roadmap. Sprints are iterative chunks of work, generally lasting about two weeks.[1]

A key element of Scrum is the ability to estimate work during the sprint planning process. With overshore teams, I have found the following sprint process to be effective:

1. The product ownership team collaborates to identify priorities and requirements for the next sprint.
2. The project manager organizes those requirements into a sprint, assigning, prioritizing, and linking all tickets.
3. The development team estimates each of their tasks within the sprint. The estimation is the exact amount of time it will take to complete the task from start to finish. This is also an opportunity for them to ask questions on tasks without cutting into a sprint deadline.
4. The project manager collaborates with the development and ownership teams to narrow down the task list until all developers have about two weeks of work to complete. A deadline is assigned based on the estimations.
5. The sprint scope and deadline is cleared with all teams and work begins. The developers have until the deadline to finish all agreed-upon work within the sprint.
6. After the sprint is finished, the developers complete a retrospective (reflection period) and estimate a new sprint. This helps developers to set goals based on the past sprint's results.

There are numerous benefits to using the Scrum methodology. When the work is agreed upon by all teams, a commitment of expectations and respect is made. Scrum empowers developers, increases team morale, and cuts down on redundancies -- all leading to better work being done faster.[1] This framework also enables the product owners to better understand, report on, and communicate project status.

These iterative sprints also allow us to address Jira's organization. Rather than having a

[1] https://www.scrum.org/resources/what-is-scrum

new board for every feature or topic (see Jira Issue Organization), sprints can be themed around these topics and organized on one board. Developers only see the active sprint, allowing them to focus their attention to what's most important. Completed issues from previous sprints, however, can still be found and filtered in Jira Issues.

FEATURE DEFINITION

Previous Process

One of nodMD's largest pain points is the definition and communication of feature requirements. While the other points outlined in this document -- especially regarding ticket creation -- can and will help to communicate requirements to the development team, a better internal process can also be put in place. The previous process was owned by Dustin and Jodie.

Future Process

All features start out as an idea. These ideas can be found internally or suggested by users. They should be communicated to the product manager to start the production process. The first step in the production process is to validate the idea. This includes coordinating with other teams to conduct market and user research. The idea can then be narrowed down, revised, and defined based on the research.

After an idea is presented, revised, and validated, the product manager can write user stories. User stories are a mix of bullet points and sentences describing each and every specific function within the idea from the user's perspective. These user stories must be agreed upon by all participating teams as they will guide the rest of the product's function.

The user stories, once complete, should be broken down further into tasks. This intermediate step allows the product manager to identify any screens that may need to

[1]https://www.agilest.org/scrum/why-does-scrum-work/

be forwarded to a design team to create mockups. Mockups should be created for any user interface-centered tasks. There should be significant collaboration between the internal ownership team and external design agency in order to refine and finalize the mockups. Dedicating more time before ticket creation and development to refine user stories and mockups prevents blockers down the road, cutting down on production costs and time.

If possible, it is advised to conduct more research after mockups are created but before the development process begins. This can also save time and money later down the line by testing and validating designs with users. User testing in this phase may include role plays, A/B testing, and exploratory tests.

After the product design is finalized, the project manager should create tickets for each individual feature. Granularity should be small, within reason. Each function should have at least one or two tasks to cover both the frontend and backend requirements. Dividing tasks out in this way helps to cover all bases when communicating tasks, ensuring all requirements are met, and helps the development team to find greater satisfaction as they are able to close tickets more frequently.

DEVELOPER COMMUNICATION

Previous Process

Currently, developer communication is managed through two ways: the separate Slack workspace and daily stand-ups.

Future Process

First, the daily stand-ups should be refined. Scrum meetings should last from 5 - 10 minutes and should only include important status updates. The meeting should not be about problem solving -- it should focus on bringing up important topics quickly so they can be further discussed later on.[1] We would benefit from the following action items:

- Starting meetings on time every time
- Holding scrum meetings every day (no cancellations)
- Encouraging developers to pre-write notes in a shared space to share during the meeting
- Sharing what each developer did that day, what their successes were, what their struggles were, and what they will do next shift
- Encouraging questions, but answering and problem solving after the meeting

Making stand up meetings more effective creates a more engaged and connected team. It is one of many Agile tools that leads to better work being completed faster through visibility, communication, and increased team morale.[2]

Developer communication overall should be habitized and increased. Currently, the only routine communication seems to be through a private Slack channel where the developers occasionally ask questions. This should be supplemented with a daily update from the project ownership team. Furthermore, adding more lines of communication can help to increase an awareness of the project status, blockers, and successes. Options here include sending daily reports at the end of each shift, logging daily Scrum notes, and adding routine comments to each Jira ticket. These options provide a starting point for the team to expand on communication in the future.

JIRA ISSUE ORGANIZATION

Previous Process

Jira is organized into Projects, which are then organized into Boards. Currently, each Board is a feature or topic within the software -- "Ongoing Issues," "Phase 2-Issues," "PCC_Errorhandling" (Fig. 1). Each of these boards is a Kanban board containing tickets from both the Primary "Nod - Current Version" project and the "Nod - Legacy Version" project.

[1] http://www.base36.com/2013/03/how-to-run-an-effective-scrum-meeting/

[12]https://www.mpug.com/why-are-daily-scrum-meetings-important-for-successful-project-management/

This process seems to be owned by Tanji at Thinkitive.

This structure makes it extremely difficult to gain an at-a-glance understanding of the developer workload and status. Because tickets are spread out across boards, developer focus is spread thin. Not only does this make it difficult for the project owners to understand the developer workflow, but it makes it challenging for developers to see progress and visualize deadlines -- both key aspects to motivating teams.[1]

Future Process

Having so many boards for the same team is a very non-traditional approach to Agile management. I believe that the team would benefit from consolidating the boards and projects. The two Nod projects can be merged into a single NodMD project. The boards can be merged into two boards, one for each Scrum and Kanban. Consolidating tickets into one board will allow a better understanding of all issues and priorities across the project without the need to track multiple boards.

I also believe that the general issue layout can also be improved upon. Right now, labels are being used to place issues within the single-topic boards. With merged boards, this will not be needed. It can be replaced with a more detailed labeling system that can denote the project phase, nod product (consult, referrals, etc.), task type (frontend, backend, etc.), and feature location. This would enable two things: first, it would allow for more detailed reporting and filtering within Jira; second, it would help to communicate and filter issues at-a-glance, speeding up the status communication process.

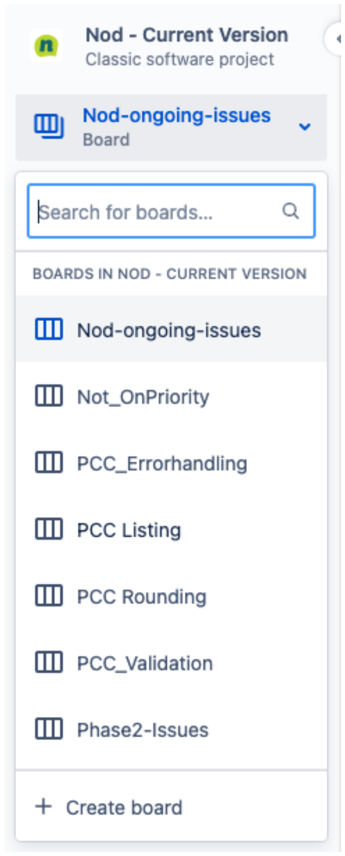


Fig. 1 - A sample of some of the current boards in Jira.

[1] https://www.bluescape.com/blog/what-is-a-scrum-board-benefits-of-scrum-board/

ISSUE CREATION AND STRUCTURE

Previous Process

Issues in Jira currently lack detail. Many have long issue summaries with minimal to no supplemental details in the issue description. While part of this may stem from a lack of understanding of project requirements, the tickets themselves should provide all necessary details to completely understand the task requirements. This process was owned by a mixture of Jodie, Vikram, and Thinkitive.

Future Process

While writing paragraphs within the ticket description can become overwhelming, there is a middle ground that allows for sufficient developer detail. To solve this issue, I recommend restructuring issues within Jira:

- **Structuring the issue summary.** The issue summary should give a quick at-a-glance summary of the issue's contents. They should be easy to read and identify the feature, its location, and the topic of the ticket. Currently, issue summaries are lengthy without giving the required detail. Issue summaries in the future might be structured to say Nod Product: Feature Location - Brief Description. For example, NODMD-2434 "Create open tasks when patient actually gets in video call" would read "nodConsult: Video Calling - Create Task on Call Start." This allows for the Jira organization to be much more scalable.
- **Providing more details within the issue description.** Currently, the issue description section of each task does not provide many details, if there is any description at all. The example above does not have an issue description. A detailed description might read:

  > Currently, tasks are created when a video call is ended. This is an issue because some providers would like the option to view an appointment/video call as a task before the appointment ends. We need to change this workflow so that tasks are created as soon as a video call starts.

  This description provides all of the details a developer might need -- what, where, and how -- in a brief few sentences. Other important information to include are steps to replicate bugs and relevant mockups/graphics. These improved ticket descriptions will cut down on developer questions and uncertainty, helping to ensure work is done correctly the first time.

- **Providing a definition of done.** There are two types of a definition of done: a definition of done for a specific task, and a general checklist to meet definition of done requirements. The checklist of requirements, which includes testing, code reviews, and deployment methodology, is not detailed within the issue. The other type of definition of done is a clear, concise statement that gives the exact requirements that need to be met in order for the task to be considered complete. Providing this definition of done sets clear expectations across development, QA, and ownership teams.[1]
- **Giving more supplementary fields and labels.** Labels are currently exclusively used to place tickets on boards. Instead of the issue fields serving the board, the fields should serve the developer. Labels can be expanded to include information about the feature location within the software, and components can be utilized to better communicate the feature requirements and to better organize issues.

The ticket creation process should also be formalized. Tickets currently are written by many people, including the developers themselves. While it might make sense for developers to suggest tickets, it is most efficient to have the project ownership team own ticket creation.

TESTING AND QA

Previous Process

Nod's Jira boards currently have one column/status within each board for "testing." As with the concerns on the overall Jira structure, this makes it difficult to communicate status and priority, especially when multiple tests are required for a single task. Currently, testing is completed by one person for each ticket (unless the ticket was a bug reported by a source outside the development team, in which case, two tests are completed). This process was owned by Thinkitive.

[1] https://www.productplan.com/agile-definition-of-done/#:~:text=%E2%80%9CThe%20definition%20of%20done%20(DoD,Derek%20Huether%20of%20ALM%20Platforms.

Future Process

A dedicated testing board should be created. Test Scenarios, an issue type within Jira, can be created automatically. A specific testing board would encourage multiple tests and make it much easier to find, filter, and report on the test status of other tickets.

In order to ensure usability and to cut down on redundancies, each ticket should be tested by two separate people. These people can be anyone with access to our internal testing platform but did not develop the feature. This process can be managed within Jira and, when testing responsibilities are shared across the team, will provide significant value to the software at extremely little cost.

DEVELOPMENT OPERATIONS

Previous Process

Currently, developers work on tasks in their local branch, submit a merge request via Bitbucket to the rest of the team, then merge the branch with the development environment repository. The branch (associated directly with a ticket in Jira) can then be deployed to production. The code is briefly reviewed during the merge request approval. Larger features are reviewed in a code review meeting as a team. All code is hosted in Bitbucket, which integrates directly with Jira and our other Atlassian products. Deployment is communicated manually. Vikram and Thinkitive own this process.

Future Process

Our current development process is overall sound, but would benefit from additional code reviews. Thorough code reviews should be conducted by at least two non-participating developers for each merge request, no matter how small, to each server. Developer meetings can be scheduled for additional code reviews in the event of a larger request, although ticket granularity should prevent this.

The deployment process also has more opportunity for expansion within Jira. By adding in a field or feature to communicate when a ticket is ready to be deployed in Jira, we are able to gain a more clear view of the project's status. This also cuts down on the time and questions developers may have regarding deployment requirements. There are numerous ways to go about marking tickets for deployment and organizing releases in Jira that can further automate this process.

DOCUMENTATION

Previous Process

The only documentation nodMD has is the backend documentation stored in Confluence. This documentation is largely unorganized. This process is currently owned by Vikram and the Thinkitive team.

Future Process

Moving forward, additional spaces should be created in Confluence to house and organize documentation. The current nodMD space can be renamed "Backend Documentation." It is appropriate for the development team to add, edit, and organize this however they see fit. Further spaces can be created:

- **Credentials Knowledge Base.** A place to store all logins for all development-related accounts, including Sendgrid, AWS, Twillo, etc.
- **End User Knowledge Base.** A place to store all end-user documentation connected to Jira Service Management.
- **nodMD.** A place to house all agreements and processes, such as this manual, a working agreement, a high-level technology stack overview, project charter, roadmaps, etc.

CLIENT SUPPORT MANAGEMENT

Previous Process

No client support management software or process currently exists.

Future Process

As our client base grows, it is important that we are able to provide low-cost support solutions. Jira Service Management (JSM) is a fantastic way to do this. JSM, a part of the Atlassian suite that integrates directly with our existing project management tools, provides a public client support portal and knowledge base management solution.

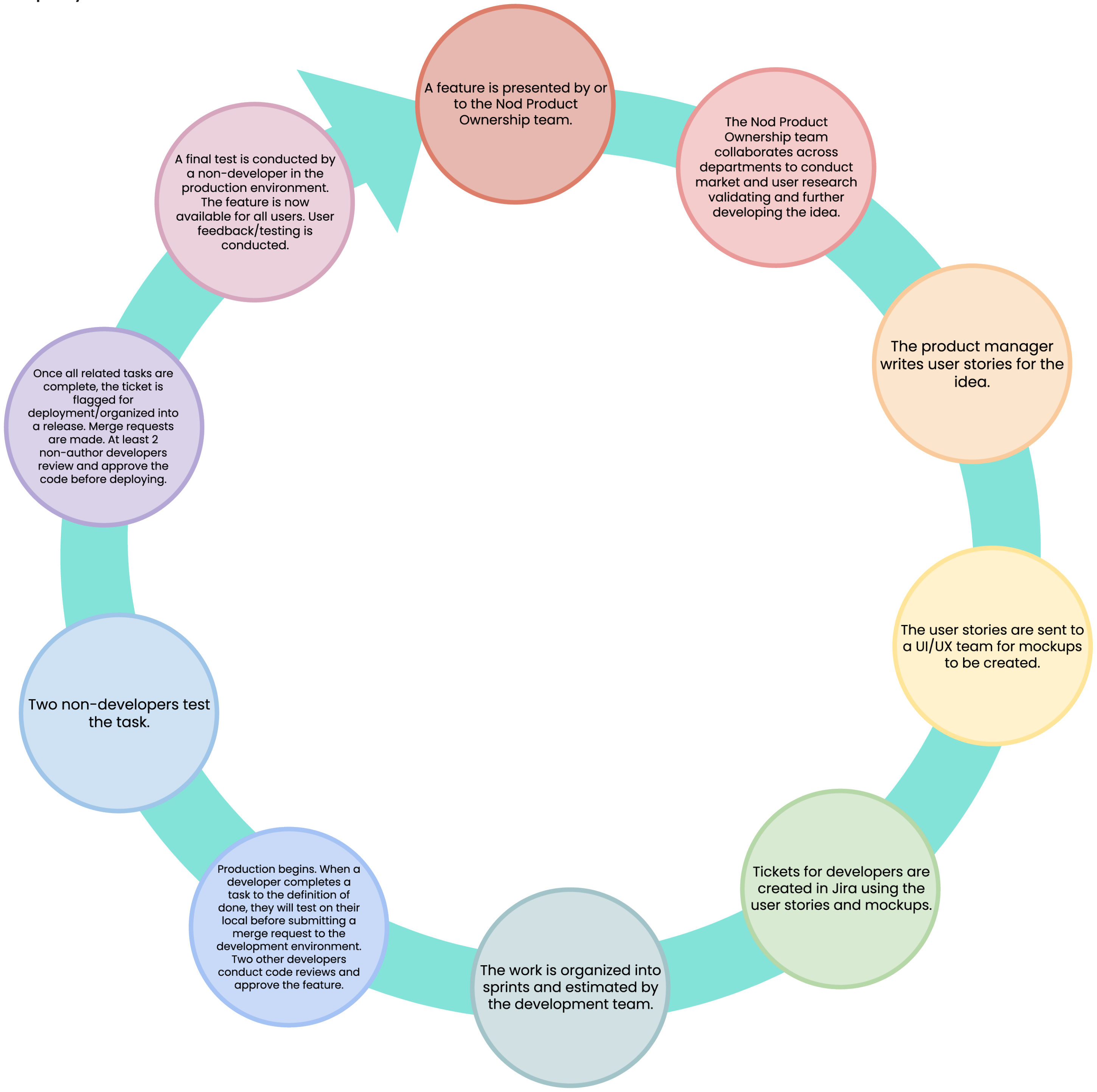
JSM's knowledge base syncs directly with Confluence to provide end-user documentation. In Confluence, anyone can create articles to be published in the knowledge base. Users can then find these articles by visiting our support portal (provided through JSM) and searching for an issue.

If a user cannot find their solution through the knowledge base, JSM prompts them to submit a support ticket. This creates a ticket directly in JSM that can be filtered and completed like any other Jira task, allowing for the same reporting and process to be integrated seamlessly with the rest of the developer tasks.

JSM is an additional Atlassian product and does carry a price tag. Atlassian offers a free plan, which we would likely qualify for, a Standard plan (about $60/month), or a Premium plan (about $120/month).

UPDATED PRODUCT LIFECYCLE

Below is a visualized example of a potential updated product lifecycle/workflow. This follows a single feature or idea through its entire journey from conceptualization to deployment.


A feature is presented by or to the Nod Product Ownership team.
The Nod Product Ownership team collaborates across departments to conduct market and user research validating and further developing the idea.
The product manager writes user stories for the idea.
The user stories are sent to a UI/UX team for mockups to be created.
Tickets for developers are created in Jira using the user stories and mockups.
The work is organized into sprints and estimated by the development team.
Production begins. When a developer completes a task to the definition of done, they will test on their local before submitting a merge request to the development environment. Two other developers conduct code reviews and approve the feature.
Two non-developers test the task.
Once all related tasks are complete, the ticket is flagged for deployment/organized into a release. Merge requests are made. At least 2 non-author developers review and approve the code before deploying.
A final test is conducted by a non-developer in the production environment. The feature is now available for all users. User feedback/testing is conducted.

ACTION ITEMS

In order to improve our process and product without disturbing relationships with third-party vendors, the following phases can be implemented:

Phase 1

- ☐ Conduct one-on-one interviews with each developer to better understand pain points and trouble areas.
- ☐ Write user stories for top priority nodConsult fixes.
- ☐ Write tickets for top priority nodConsult fixes following the new ticket structure.
- ☐ Put together a new sprint in a new Jira board.
- ☐ Gather estimations from Thinkitive for per-ticket time.
- ☐ Reform scrum meetings (shorten, add consistency, add pre-written notes).

Phase 2

- ☐ Complete the test sprint with Thinkitive.
- ☐ Send the same sprint scope to other development agencies to gather estimations on cost and time.
- ☐ Write user stories for second priority nodConsult fixes and related nodReferrals setup/features.
- ☐ Send user stories to User Ten for estimations on UI/UX design.

Phase 3

- ☐ Evaluate the test sprint with Thinkitive alongside estimations from additional third-party vendors. Finalize a decision on a third-party vendor moving forward. Finalize plans for additional changes.
- ☐ Finalize mockups from User Ten.
- ☐ Create tickets in Jira for second priority fixes/nodReferrals setup.
- ☐ Write user stories for next phase of tasks.
- ☐ Update Jira based on the results found and previous suggestions -- merge boards, organize tickets, merge projects, etc.

Each of these phases should last approximately two weeks, although the timeline is subject to change with developer estimations and new insights. Should the transition period go as planned, a final, stabilized process can be fully enacted and practiced by no later than May of 2021.

CONCLUSION

Overall, the current production strategy lacks organization and clear process. Further data collection, including reports from sprints, interviews and surveys with developers, and user feedback, is necessary to create a truly effective and refined strategy, but is impossible to collect now. The changes outlined will help to create not only organization and process, but to further instill the culture of ownership, truth, and respect Nod strives to create.