SOP for Managing User Stories on the MESH Agile Board

Table of Contents

Last Updated: 08/30/2024

1. Introduction

This SOP provides a clear and detailed guide on how to manage and track user stories on our agile board using the newly implemented process. The aim is to ensure that every team member, regardless of experience level, can easily navigate the board, understand the status of each task, and effectively contribute to the project's success.

2. Roles and Responsibilities

- **Product Designer (PD)**: Will follow their SOP as it relates to user stories during the sprint.
- **Technical Lead (TL)**: Oversees technical tasks and ensures they align with requirements.
- **Business Owner Reviewer (BOR)**: Verifies that the work meets business needs.
- **Software Testing Engineer (STE)**: Conducts testing to validate task functionality.
- **Scrum Master (SM)**: Facilitates task movements and addresses blockers.
- **Developers (DEV)**: Execute development tasks and ensure thorough testing.
- **Business Analyst (BA)**: Primarily responsible for writing all user stories.

3. Process Overview

Each user story is broken down into specific subtasks that represent the activities required to complete the story. These subtasks will move through three main stages on the board: To Do, In Progress, and Done. The goal of this process is to enhance transparency, improve tracking, and ensure that every aspect of a user story is addressed.

Status of User Stories

User stories will follow the same stages as subtasks: To Do, In Progress, and Done. The status of the user story reflects the highest priority subtask status. If any subtask is In Progress, the user story is considered In Progress. This approach avoids reverting to more complex statuses such as In Review or Approval, streamlining the process.

4. Step-by-Step Guide

4.1 To Do Column

In the To Do column, all the preparation work for the subtasks of a user story is conducted. This includes planning and ensuring everything is ready for execution.

- **Development**: Begin development work on the user story. Ensure all unit tests are planned and prepared.
- **Unit Testing**: Plan and prepare for unit testing to validate the development work.
- **Code Review**: Plan and prepare for an internal code review to ensure all development aligns with the set standards.
- **Deployment to TI**: Plan and prepare for deploying the code to the Testing Integration (TI) environment.
- **QA Testing**: Plan and prepare for Quality Assurance (QA) testing in the TI environment to catch any issues.
- **Deployment to UAT**: Plan and prepare for deploying the code to the User Acceptance Testing (UAT) environment.
- **Business Owner Testing**: Plan and prepare for the business owner's/PO's testing and validation in the UAT environment.
- **Approval**: Plan and prepare for final approval by the product owner.

4.2 In Progress Column

Once tasks are being actively worked on, they are moved to the In Progress column. This status indicates that the task is currently undergoing execution, review, or testing.

- **Development**: Actively work on development tasks, ensuring alignment with requirements.
- **Unit Testing**: Conduct unit tests and ensure that all tests pass before moving forward.
- **Code Review**: Perform an internal code review and address any feedback to ensure the code meets quality standards.
- **Deployment to TI**: Deploy the code to the TI environment and validate that the deployment is successful.
- **QA Testing**: The QA team tests the code in the TI environment, validating its functionality and stability.
- **Deployment to UAT**: Deploy the code to the UAT environment and validate that everything works as expected.

- **Business Owner/PO Testing**: The business owner reviews and tests the code in the UAT environment to ensure it meets business needs.
- **Approval**: The product owner gives final approval, ensuring that the functionality meets the acceptance criteria.

4.3 Done Column

When a subtask is completed, it is moved to the Done column, indicating that this part of the user story has been fully addressed.

- **Development**: Development work is completed and validated.
- **Unit Testing**: Unit testing is completed, and all tests have passed.
- **Code Review**: Code review is completed, and all feedback has been addressed.
- **Deployment to TI**: Deployment to the TI environment is completed and validated.
- **QA Testing**: QA testing is completed, and all tests have passed.
- **Deployment to UAT**: Deployment to the UAT environment is completed and validated.
- **Business Owner Testing**: Business owner/PO testing is completed, and all tests have passed.
- **Approval**: Final approval is given by the product owner, and the user story can be considered done.

5. Handling Blockers

Identifying Blockers

If a subtask or user story encounters an issue that prevents it from progressing, the assignee should immediately identify and document what is causing the blockage. Blockers could include dependencies on other tasks, lack of information, or external factors.

Documenting the Blocker

Add a detailed comment to the blocked task, whether it is a subtask or the main user story. The comment should clearly explain what the blocker is and what is needed to resolve it (e.g., “Waiting on API credentials from the IT team”).

Applying the Blocked Label

Instead of moving the task to a separate blocked column, apply the blocked label to the task. This label is configured to turn the card red, making it immediately visible to the team that this task requires attention.

Reassigning if Necessary

If resolving the blocker requires input from another team member or department, reassign the task to the appropriate person. Ensure that you add a comment indicating that the task has been reassigned and why.

Regular Follow-Up

During daily standups, regularly review all tasks labeled as blocked to ensure they are being actively worked on and that the necessary actions are being taken to resolve them.

Escalating Blockers

If a blocker cannot be resolved in a reasonable time limit or is causing significant delays, escalate the issue to higher management or stakeholders. Document any escalations in the task comments.

Resolving Blockers

Once a blocker has been resolved, remove the blocked label and update the task status back to In Progress or its previous state. Add a comment noting that the blocker has been cleared (e.g., "API credentials received, resuming development").

Impact on Overall Progress

If the blocked subtask affects the entire user story's progress, document this impact in the main user story (e.g., "Development was delayed due to blocker XYZ").

6. Managing Issues and Bugs

Logging Issues

If an issue or bug is identified during any phase of the user story's progress, it should be logged immediately as a subtask or related task.

Documenting Issues

Provide a detailed description of the issue or bug, including steps to reproduce it, the expected outcome, and the actual outcome.

Assigning and Prioritizing Issues

Assign the issue to the relevant team member and prioritize it based on its impact on the overall project. Use labels or tags to indicate the priority level (e.g., Critical, High, Medium, Low).

Tracking Progress

Monitor the progress of the issue or bug fix just as you would for any other task. Ensure that updates are regularly added to the task to keep the team informed.

Resolving Issues

Once the issue or bug has been resolved, update the status to Done and ensure that any related subtasks are also updated. Add a comment to document how the issue was resolved.

7. Additional Training and Support

To ensure that everyone is comfortable with the new process, additional training sessions will be held. These sessions will include walkthroughs of the agile board, discussions of specific scenarios, and opportunities to ask questions. Example boards will be shared to illustrate how tasks should move through the various stages.

Here's a quick overview of the board – Board.

8. Summary and Best Practices

- **Consistency**: Follow this SOP consistently to ensure that everyone on the team is aligned and that the agile process runs smoothly.
- **Communication**: Keep all communications clear and concise, especially when documenting blockers, managing issues, or providing updates.

- **Review and Adapt**: Regularly review the process during retrospectives and be open to adjusting based on team feedback. This SOP is a living document and should evolve as the team's needs change.

FAQ

Q: What status should the user story be in?

User stories will follow the same stages as subtasks: To Do, In Progress, and Done. The status of the user story will reflect the highest priority subtask status. This streamlined approach avoids unnecessary complexity.

Q: How do I know what In Progress means?

The subtask title will indicate whether the task is in the development phase, testing, or awaiting review, giving you a clear idea of its status.

Q: Will this process make it harder to see the overall status of a user story?

No, the breakdown provides more clarity on where each part of the user story stands, allowing for better tracking and planning.

Q: Do I need to document the same thing in both the subtask and the main story?

No, subtasks should contain concise, task-specific updates. The main story should capture high-level updates, key decisions, and final approvals.

Q: How do I handle approvals?

Once an approval subtask is marked as Done, add a brief comment to the subtask. The main story should reflect the final cumulative approval in the comment.

Q: What if a subtask is approved but the main story is not?

Subtasks log detailed approval comments. The main story reflects the final approval status. If the main story is not approved, there may still be outstanding issues that need to be addressed.

Q: How can leadership quickly see the status of each ticket? Will they only see To Do, In Progress, or Done unless they open a ticket?

Yes, there is a way for leadership to view the status of each ticket. The board is designed to be transparent, and all subtasks are visible by default. This means leadership can see not only the high-level status (To Do, In Progress, Done) but also the detailed progress within each user story by viewing the expanded subtasks. Additionally, high-level comments on the main story provide an overview, while subtasks give detailed updates. Labels and tags are also used to quickly convey status or issues, ensuring that leadership has all the information they need at a glance.

Q: What goes into the main ticket versus the subtasks? If I have the testing in UAT subtask, will I have the full context of the ticket or just a test script or summary?

The main ticket should contain high-level updates, key decisions, and approvals. It is the place where you document the overarching progress and major milestones of the user story. Subtasks, on the other hand, should include detailed task-specific information such as test scripts, steps taken, and results. This ensures that each subtask has all the context necessary to complete that specific piece of work, while the main story tracks overall progress.

Q: What happens if I find an issue while I have the testing in UAT subtask that needs the developer to fix it or rework it?

If an issue is found during the testing in UAT subtask that requires rework by a developer, follow these steps:

1. **Identify and Document the Issue**: Document the issue clearly in a comment on the relevant subtask. Provide details such as steps to reproduce, expected versus actual results, and any screenshots or logs that may help.
2. **Create a Test Issue Subtask**: If the issue is related to the current user story, create a new subtask called “Test Issue” and assign it to the appropriate developer. This subtask should include the documented issue and any additional context.
3. **Update the Status**: Move the original testing in UAT subtask back to “To Do” or “In Progress,” depending on the severity of the issue and the rework required. The newly created test issue subtask will follow the usual flow through the board.
4. **Discussion and Stand-Up**: Raise the issue during the daily stand-up to ensure visibility and to coordinate the next steps with the team. If the issue is unrelated to the current user story, create a new bug work item and prioritize it based on its impact. Severe bugs should be addressed in the current sprint if possible; otherwise, they should be planned for the next sprint.

Q: Who is responsible for managing and moving the tickets? Would a developer need to go to Jira every time they need to promote the work to a new environment?

Each team member is responsible for managing and moving their assigned subtasks on the agile board. When a task is completed, the assignee should update the status and move it to the next appropriate column (e.g., from In Progress to Done). For tasks that involve multiple steps, such as promoting code to a new environment, the developer will indeed need to update the status in Jira. However, to streamline the process:

- **Assign to Next Responsible Person**: Ensure that the task is assigned to the next responsible team member for review, testing, or approval.
- **Use Comments**: Add comments to indicate the completion of your part and to notify the next person in the workflow.
- **Automation Tools**: Automation can help reduce manual effort. For instance, setting up automated notifications or status updates when specific conditions are met (e.g., code deployment success) can make the process more efficient.

Task Transitions Without Pre-Assignment

While pre-assigning QA tasks can help streamline the workflow, it is understood that in the initial stages, the team may not yet know who will take on specific QA tasks. Therefore, an alternative approach to manage the handover between development and testing, as well as from testing to product owner or business owner (BO) approval, can be implemented through effective communication.

Key Points

- **Use of Team Chat for Communication**: When a development task is completed and ready for QA, the developer should use the team's communication platform (e.g., Teams chat) to notify the QA team. For instance, the developer can post a message like "Story XYZ is ready for QA, can someone pick it up?" This ensures that all QA members are aware of the task's readiness, and the first available team member can assign themselves to the task.
- **Tagging for PO/BO Approval**: Similarly, when a task is ready for product owner or business owner approval and the specific approver has not been pre-assigned, a comment should be added to the main story or the relevant subtask. The comment should tag both potential approvers (e.g., @Stephanie, @Chevelle, "Story XYZ is ready for your approval"). This way, either one of them can pick up

the task based on availability. If the task remains unassigned after a reasonable time, this tag serves as a prompt, ensuring that the approval process does not stall.

- **Daily Stand-Ups and Continuous Communication**: While daily stand-ups are critical for discussing ongoing tasks, this real-time communication via the team chat complements those meetings by allowing immediate updates and responses. Developers and QA should actively use this chat to flag when tasks are ready for the next phase, keeping everyone informed and enabling swift transitions.
- **Managing Unassigned Tasks**: If a task remains unassigned, whether it is a QA task or a PO approval, the tag comment acts as a gentle nudge for team members to pick it up. This approach ensures flexibility while maintaining accountability and visibility across the team.

This method of using team chat for communication, combined with tagging relevant team members when tasks are ready for the next stage, ensures that no task falls through the cracks. It also allows for adaptability in early project stages when roles may not yet be firmly established or pre-assigned.
