Playbook Methods Repository

# **Defect Management Planning**

Design and implement processes to efficiently coordinate defect detection, root cause analysis, defect triage, defect resolution, and reporting across product team(s) and stakeholders. Effective defect management incorporates both the processing of defects and interplay with Agile Rituals to address systemic sources.

### Remote Agility: **•** High

### Linked Tactic(s): Quality Assurance

## Why we do it:

The tension between delivering new functionality and addressing existing defects is a constant in software development. Too often, teams have no set process for fixing defects in a sustainable, consistent manner, and instead employ ad-hoc emergency measures once a product’s defect backlog has become unmanageable, or customer pain points can no longer be ignored. Such measures may include “bug bashes”, where all feature work is paused for a set amount of time while the team works solely on defects.

To avoid scenarios like these, team members implement processes to ensure defects are analyzed, prioritized, and resolved regularly, resulting in a bug backlog that is consistently monitored and aligned with both the product roadmap and user feedback.

## 

## When to apply it:

* All responsible roles should meet before the start of development to determine the frequency of relevant meetings, and align on the initial measurements used to determine the number of defects addressed in each sprint. For example, are a certain percentage of the tickets in each new sprint required to be defects? Or should there be a minimum number of defects brought in (e.g., 3)?
* Once the project is into development, these measurements should be revisited as part of sprint retrospectives, and other team rituals.

## Best Practices & Considerations:

* Establishing Cadence: Typically the most difficult portion of Defect Management is addressing defects in a consistent and predictable manner. As mentioned in the previous section, establishing this cadence early not only ensures that the bug backlog is kept manageable during the course of development, but also normalizes having a portion of sprint tickets allocated to bugs. Using defect templates (see example below) also enforces a level of consistency.
* Bug Triages should be held on a frequent schedule, e.g. weekly or bi-weekly, to review the list of new defects. At least one representative from Product, Development, and QA respectively should be present to ensure that defects contain all necessary information for prioritization and investigation, so they can be immediately worked on once brought into a sprint.
* Root Cause Analysis is an important practice in defect management, as individual defects may describe only the symptoms of a deeper issue. QA can perform preemptive analysis during the logging of defects by executing scenarios related to the initial defect, to provide a wider view of the conditions that cause the failure. This avoids potential back-and-forth conversations in Bug Triage meetings, where this additional information would be requested after the defect has been logged.
* Product Dogfooding: It's a good practice to have a variety of users try to break the product and gather feedback that can be used to make high-impact improvements for the next release. It is slightly different from how QA does Exploratory and Functional Testing, (e.g: Designers, Product Developers and other QA members test the product and provide feedback). These defects are managed differently and considered as defects from user testing (Alpha Testing), where they will be created as new Stories or Epics, and broken down further into planned sprints as per their severity and priority.
* Summarizing the list of verified and resolved defects in a QA sign off document for each release of the product can help identify persistent regressions over time. Such regressions can then be added to a Smoke or Automated Test suite.

## Responsible roles:

* QA:
  + Creates defect tickets for new issues, and ensures that all required information is present on the tickets.
  + Schedules and facilitates Bug Triage meetings in partnership with Product.
  + Immediately informs the project team if there are any defects blocking QA.
* Product:
  + Collaborates with QA to determine a cadence of defects to address each sprint.
  + Schedules and facilitates Bug Triage meetings in partnership with QA.
  + Prioritizes new defects in project backlog, and determines which defects to include in new sprints.
  + Socializes the user/business impact of defects to assist with higher-level prioritization.
* Development Leads:
  + Collaborates with QA to determine Severity and Priority of the defects.
  + Coordinates and directs the development team efforts for defects.

## Tools:

### Online tools/platforms/services

* + [Jira](https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=144583515117&campaign=19313277967&creative=641924345505&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p74602868210&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=CjwKCAiAqt-dBhBcEiwATw-ggJ7g6brkt9a6ifxMo6aRFtPlOgMe8JC8fMxWyI2W9ZoVx6Eoka90HhoCXTsQAvD_BwE&gclsrc=aw.ds): Many individuals are no doubt familiar with using Jira to create defects, but it can also create Dashboards to help visualize different defect metrics and statistics.
  + [TestRail](https://www.gurock.com/testrail/?utm_campaign=gg_dg_us_can_search_brand&utm_source=google&utm_medium=cpc&utm_content=testrail%20%5BTestrail%20Website%20-%20Visited%20last%2030%20Days%5D&utm_term=test%20rail&gclid=CjwKCAiAqt-dBhBcEiwATw-ggFrKYlBq4zXgKwyuZKdBkPxpH57UEyaiEPGg03NJu1gMHM4iGvU2ThoCLh8QAvD_BwE): TestRail is a web-based test case management tool. Testers, developers and Project managers can use it to manage, track, and organize software testing efforts. It also can be integrated with Jira to track the defects. Dashboards can provide metrics including percentage of Test cases passed, failed, blocked, and number of defects discovered.

### 

### Log / Debugging Tools

* + ADB (Logcat): Command line tool to collect logs for android devices
  + [XCode](https://apps.apple.com/ca/app/xcode/id497799835?mt=12): IDE for macOS, used for development on all Apple platforms (iOS, macOS, tvOS, and watchOS)
  + [Flipper](https://fbflipper.com) (Debugger for Mobile apps & React Native apps )
  + [Charles proxy](https://www.charlesproxy.com/): Debugging Proxy for Web/Mobile Apps
* Other
  + [SoapUI](https://www.soapui.org/)
  + [Postman](https://www.postman.com/)
  + [Vysor](https://www.vysor.io) (Mobile app tool for Screenshots/Video Recording)

## 

## Thoughtworks Examples - Linked

### Client working docs, airtable, miro/mural boards

* + See templates below, based on client deliverables

### Client polished presentations/deliverables

* + See templates below, based on client deliverables

### Internal assets - clinic materials / guild docs

* + xx

## 

## Learn more: How we do this?

### Templates (docs, decks, sheets, miro, etc.)

* + [Defect Template](https://docs.google.com/document/d/15Zu_-Sn5LH0o_PJu7rwPIhAMz7fxgDzvfsJZqO46KbU/edit?usp=share_link)
  + [Bug Bash Template](https://docs.google.com/document/d/1IleturhKkSRP1YdYDR7KrVg_Oo7FKK5eASZYHir0qiA/edit?usp=share_link)
  + [Sample QA Result Summary Report](https://docs.google.com/presentation/d/1NlTFsf6qdEmQBMBeDKTEUqOq3bF9G5Qnn0UT4cX5jh4/edit?usp=share_link)

### How-To Resources (external or internal)

* + xx

### Outside References (articles, books, etc.)

* + [Defect Management](https://www.linkedin.com/pulse/defect-management-what-why-do-we-need-flip-nehrt)
  + [Defect Triage Meetings](https://www.linkedin.com/pulse/defect-management-triage-meetings-flip-nehrt?trk=article-ssr-frontend-pulse_more-articles_related-content-card)

### Sub-set Activities

* + xx