a.venugopal@dxc.com

Day – 3

Date:28/02/2024

PRD + Solution Design/HLD/LLD + Each module --> User stories + Agile - Scrum methodology

Test-plan

https://www.testrail.com/blog/create-a-test-plan/

Test – case-template

https://www.softwaretestingmaterial.com/test-case-template-with-explanation/

[8:07 AM] noor shareef

https://www.browserstack.com/guide/verification-and-validation-in-testing

By Dibya Sir
What is release cycle?
What is sprint cycle ?
Is release cycle == sprint cycle
When should the retrospective happen?
Sprint Plan :
https://www.youtube.com/watch?v=75I_PWPG0-s

Burn-down chart?

a.venugopal@dxc.com

How To Create A Test Plan (Steps, Examples, & Template:

In software development, a test plan defines your testing team's test strategy, goals, and scope, which ultimately work together to ensure that all your software components are tested sufficiently before a release.

Follow these six steps to create an efficient test plan:

1. Define the release scope
2. Schedule timelines
3. Define test objectives
4. Determine test deliverables
5. Design the test strategy
6. Plan test environment and test data

How to create a test plan?

1. Define the release scope:

Before any test activity occurs, it's important to define the scope of testing for your release. This means defining the **features** or **functions** that need to be included in the release, considering any constraints and dependencies that can affect the release, and determining what type of release it is.

Examples of questions to ask when defining the release scope include:

- Are there new features being released in this version?
- What are the risk areas?

- Are there any particularly sticky areas where you’ve seen regressions in the past?
- What type of release is it? Is this a maintenance release that includes bug fixes? Is this a minor feature release? Is this a major feature release?
- What does being “done” actually look like for your team?

Example:

Software Application: Online Shopping Platform

Release Scope for **Version 2.0**:

Feature Additions:

Introduction of a "Wishlist" feature where users can save items for future purchase.

Implementation of a real-time chat support system for customer assistance.

Bug Fixes:

- Resolving an issue where the shopping cart occasionally shows incorrect item quantities.
- Fixing a login authentication bug that caused intermittent login failures.

Changes and Updates:

- User interface improvements for better navigation and a more modern look.
- Enhancements to the checkout process to make it more user-friendly.

Exclusions:

- Integration with a third-party payment gateway (deferred to the next release due to ongoing negotiations).

Dependencies:

- Dependency on an external address validation service for accurate shipping information.

Acceptance Criteria:

- Wishlist feature should allow users to add, remove, and view saved items.
- Real-time chat support should connect users to a customer support representative promptly.
- Bug fixes should be validated and tested with various scenarios to ensure they are resolved.

2. Schedule timelines

Specify release deadlines to help you decide your testing time and routine. Here are some pointers for determining timelines:

- Consult your project manager to understand the current release timeline.
- Look at past release times and schedules.
- Consider extraneous elements: Does the release need to coincide with outside variables, such as conferences or events? Factor those into your release date prediction.
- Consider the timeframes for development: Your development team might have a set schedule for finishing development work. Make sure you comprehend that timeframe so you can adjust the testing schedule.
- Add some extra wiggle room: It’s common to encounter unexpected delays. Including extra time for unforeseen events can help you stick to your plan.
- Review and update the schedule frequently to ensure the test timetable is attainable.

3. Define test objectives

A test objective is a reason or purpose for designing and executing a test. These objectives ultimately help guide and define the scope of testing activities.

Examples of general test objectives include:

- Identifying and reporting defects
- Testing new features
- A certain level of test coverage

Examples of objectives for specific types of testing include:

- **Functional testing objectives:** Ensure the software works as it should. Examples of goals for this objective include: Validating user workflows, data processing, and verifying input/output parameters.
- **Performance testing objectives:** Ensure the software is efficient and can handle various loads. Examples of goals for this objective include: Verifying software reaction time, throughput, and scalability.
- **Security testing objectives:** Uncover program security flaws. Examples of goals for this objective include: Verifying authentication and authorization features and identifying potential threats.
- **Usability testing objectives:** Concentrate on ease of use and user experience. Examples of goals for this objective include: Validating software accessibility, verifying user flow, and identifying user-related issues.

Measure testing with the right metrics

Metrics assess the overall quality of a release, the progress of your testing, and the effectiveness of your testing (for a particular test cycle or the entirety of your testing).

They provide visibility into your testing process and overall product quality, ultimately helping your team decide if your release is ready to ship. Here are some metric formulas you might consider:

Defect Density

- Defect Density = Defect count/size of the release (lines of code)

Example: If your software has 150 defects and 15,000 lines of code, its defect density is 0.01 defects per line of code.

Test Coverage

- Test Coverage = (Total number of requirements mapped to test cases / Total number of requirements) x 100.

Defect Detection Efficiency (DDE)

- DDE = The percentage of defects detected during a phase / Total number of defects

Time to Market

- TTM = The time it takes for your company to go from idea to product launch

4. Determine test deliverables

Test deliverables are the products of testing that help track testing progress. Deliverables should meet your project's and client's needs, be identified early enough to be included in the test plan, and be scheduled accordingly. There are different test deliverables at every phase of the software development lifecycle. Here are important deliverables to focus on before, during, and after testing:

Before testing

- **Test plan document:** The scope, objectives, and approach of the testing endeavor are all outlined in the test plan.
- **Test suite:** Test cases illustrate how to run a test, including input data, expected output, and pass/fail criteria.
- **Test design and environment specifications:** The test environment outlines the hardware and software configurations used for testing.

During testing

- **Test log:** The test log records each test case's results, including issues and resolutions.

- **Defect report:** A defect report lists testing issues by severity, priority, and reproducibility.
- **Test data:** According to the International Software Testing Qualifications Board (ISTQB), test data is data created or selected to satisfy the execution preconditions and input content required to execute one or more test cases.
- **Test summary report:** The test summary report lists the number of tests run, passed, and failed, as well as open defects.

After testing

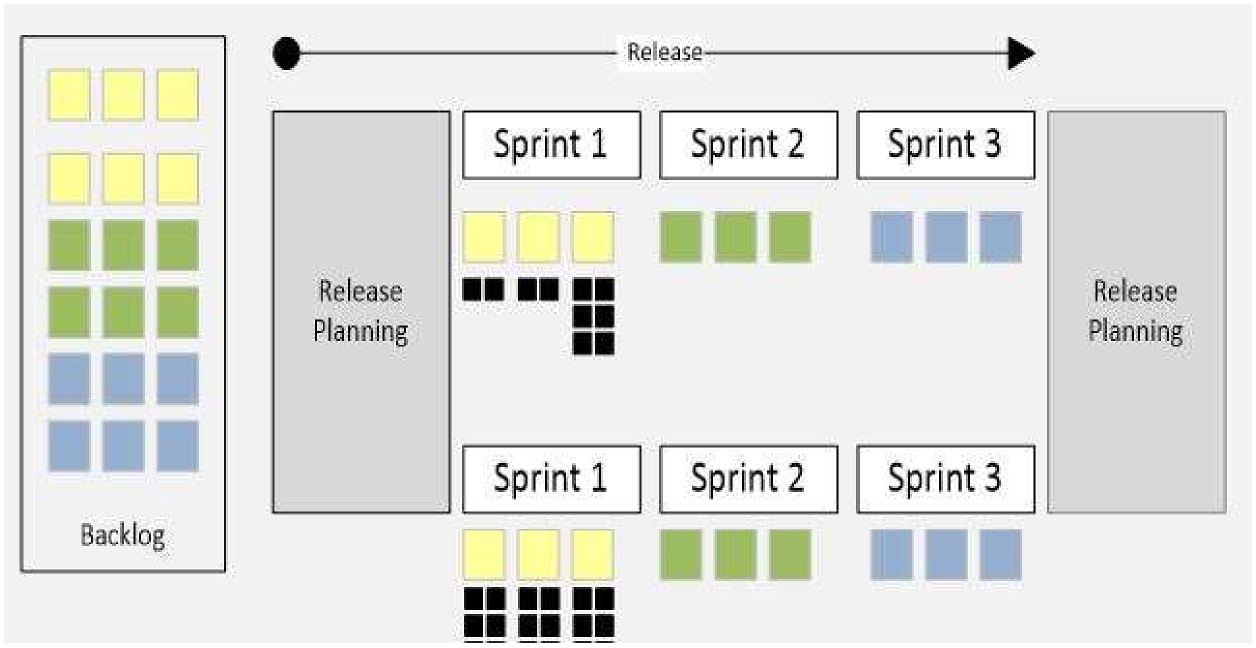
- **Test completion report:** Covers the testing scope, product quality, and lessons discovered.
- **User acceptance test (UAT) report:** Points to any issues found and fixed.
- **Release notes:** List information about what the release includes. Examples include any new features for development, advancements, or fixes.

A test plan's content and structure differ depending on its context. Although there isn't one cookie-cutter way to write a test plan, following best practices for test plan development can help companies deliver high-quality software.

TestRail is test plan software designed to make it easy to follow best practices for test plan development. In TestRail, you can enter test cases with preconditions, test instructions, expected results, priorities, and effort estimates.

What is Release Cycle?

In the quest to build better products faster, a release cycle is the life cycle of developing, testing, and deploying a product update or a new feature into production.



Importance of a Release Plan

The significance of a release plan in Agile cannot be overstated. It bridges the gap between the daily activities of the development team and the long-term product goals. Here are some of the key reasons why a release plan is crucial in Agile:

1. Alignment: It ensures that the team's efforts are aligned with the product vision and customer needs.
2. Transparency: A release plan provides stakeholders with visibility into the product development process, fostering trust and collaboration.
3. Prioritization: It helps prioritize features based on their value to the customers, ensuring that the most critical functionalities are delivered first.
4. Flexibility: Agile release planning accommodates changes in customer requirements or market conditions, allowing teams to adapt their plans to deliver the most value.

5. Forecasting: It enables teams to forecast release dates and manage expectations internally and externally.

Is Release cycle & Sprint cycle are same?

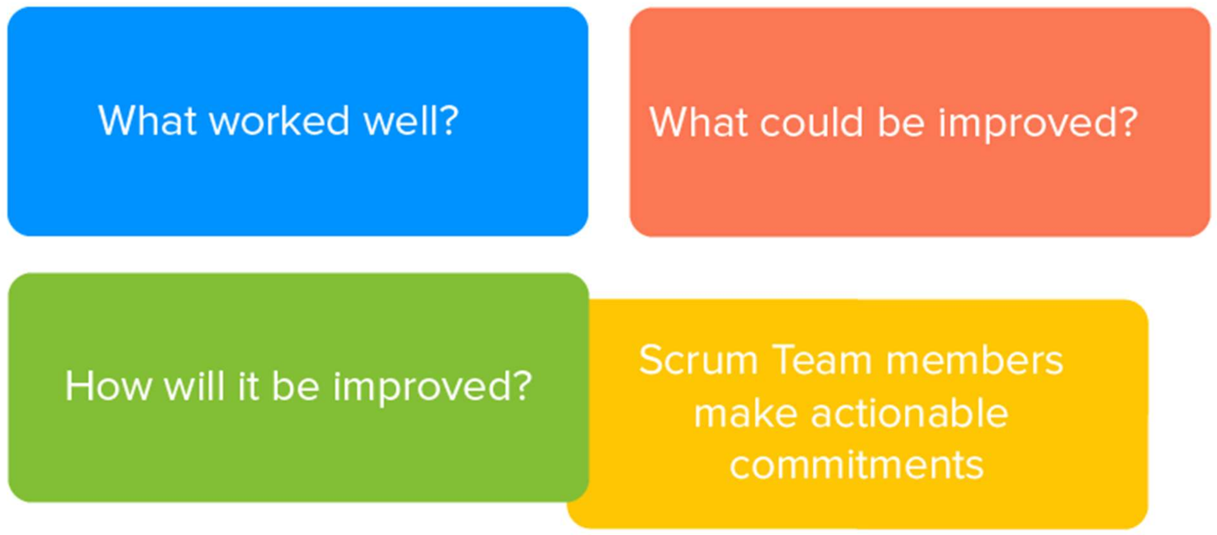
- A sprint should not be confused with a release. A sprint is a time box for completing a defined set of work, whereas a release brings a new product experience to market once it is ready to be delivered.

- A product release can occur at the end of a sprint or after several sprints.

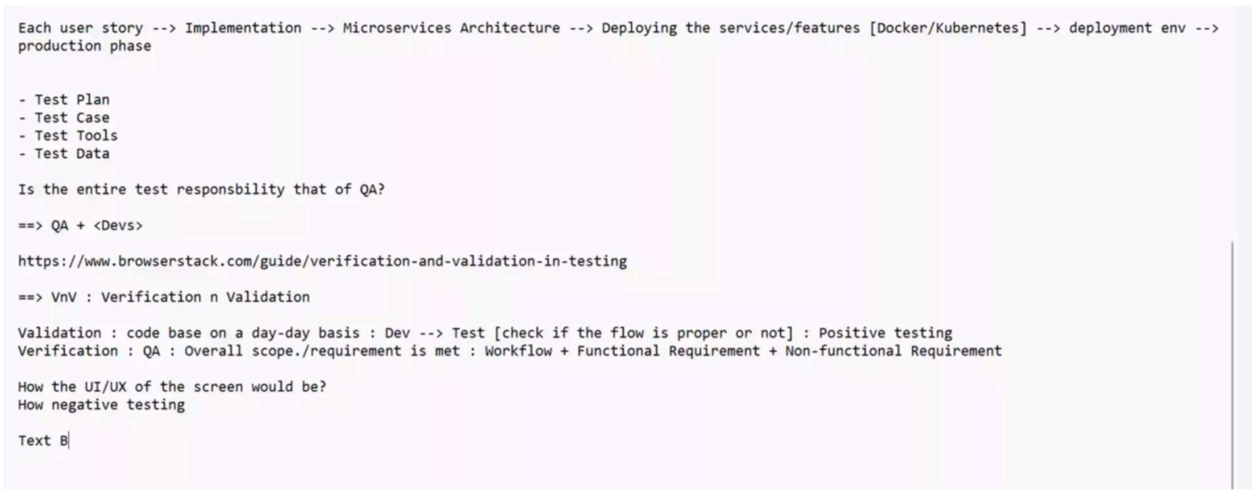
When should the retrospective happen?

If a Sprint traditionally takes 1-2 weeks to complete, then a retrospective should be held every 7-14 days. If your team is struggling, you may want to consider holding Sprint Retrospectives more often. However, if you think your team is doing well, you can probably stick to the once-per-Sprint schedule *a.venugopal@dxc.com*

The Sprint Retrospective Model



Scrible



```
Each user story --> Implementation --> Microservices Architecture --> Deploying the services/features [Docker/Kubernetes] --> deployment env -->
production phase


- Test Plan
- Test Case
- Test Tools
- Test Data

Is the entire test responsbility that of QA?

==> QA + <Devs>

https://www.browserstack.com/guide/verification-and-validation-in-testing

==> VnV : Verification n Validation

Validation : code base on a day-day basis : Dev --> Test [check if the flow is proper or not] : Positive testing
Verification : QA : Overall scope./requirement is met : Workflow + Functional Requirement + Non-functional Requirement

How the UI/UX of the screen would be?
How negative testing

Text B
```