

Test strategy and guidelines TCF Documentation

February 2022

Contents

[Test Management in Agile 3](#_Toc96536815)

[Recommended testing types 5](#_Toc96536820)

[Test Lifecycle & Management 6](#_Toc96536821)

[Test environments 7](#_Toc96536828)

[Defect lifecycle 8](#_Toc96536858)

[Sign Off 8](#_Toc96536861)

# Test Management in Agile

# Agile methodology

Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. Development and testing activities are concurrent unlike the waterfall model.

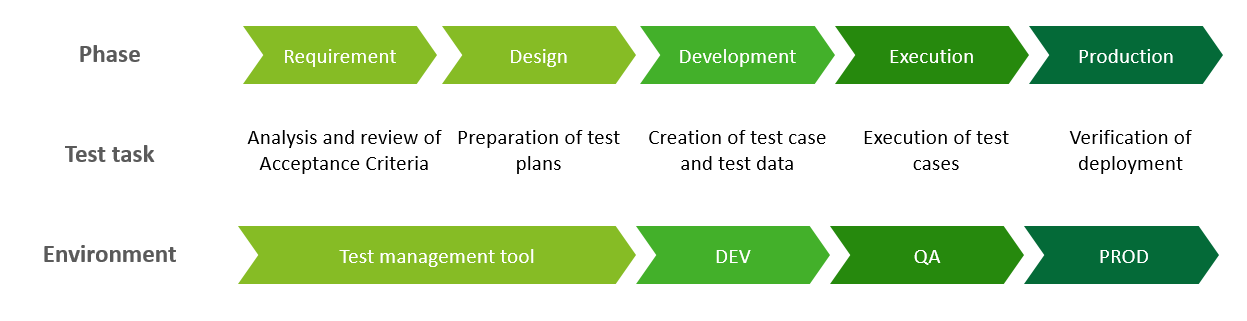
# Agile team structure

|  |  |
| --- | --- |
| **Role** | **Responsibilities** |
| Business Analyst | Sprint planning and initiation, addressing requirement clarifications, driving daily scrum meetings, sprint closure and sprint retrospective meetings. |
| Developer | Understanding business requirements, estimating user stories in the sprint backlog, and developing the product / deliverables / service |
| Test Lead | Developing a test plan and managing team members according to it, coordinating with other areas to accomplish the test effort, control the workflow of those who are team members. |
| Test Manager | Implementing testing processes, managing testing sign off and deliverables. |
| Test Engineer | Developing test cases, executing them, analyzing, and reporting results. Developing and maintaining the test framework and supporting scripts. |
| Product Owner | Defining the vision or goal of automation, prioritizing, and evaluating needs, works with internal stakeholders to ensure project deliverables and expectations stay on track. |

It is recommended that the tester: developer ratio in an agile team is **1: 2**

# Role of test team in the Agile process

The agile testing process begins at the inception of the software development life cycle. From the initial design meeting, through the development phase, to final testing and hardening of the application. This process is repeated in two-to-four-week sprints until the project is released.



# Testing guidelines

* **Time:**Testing should always start parallelly alongside the requirement analysis process
* **Risk Assessment:** Critical sections should be tested with the highest priority and as early as possible
* **Test Scope:** Determining the test scope at the beginning of the sprint keeping in line with resource capacity
* **Code Freeze:** Setting a specified duration where development activities are frozen to complete testing functionalities within the scope of that sprint
* **Test Coverage:** Ensure that all the requirements have been captured in the tests
* **Test execution**: Prioritize your test cases so that testing is effective and can be completed within the required time
* **Release delivery:** Time available for testing is not unlimited. An effective test plan is crucial before starting the process of testing
* **Negative scenarios:**Testing should be done with correct data as well as with flawed inputs to make sure the system is leak proof
* **Analyzing test results:**Quantitative assessment of tests and their results must be done
* **Feasibility of automation:** Evaluate existing test cases and assess their feasibility to be automated
* **Simulating production data:** Client data must be simulated in the testing environment to scrutinize issues specific to it
* **Business scenarios:** Incorporate business specific scenarios in your test cases which align with the tests executed by clients in UAT environment

Implementing above guidelines will help ensure deliverable of high-quality software in tight deadlines that enhance the confidence, trust, and positive perception for the company.

**Recommended testing types**

There are recommended types of testing that can be used to make sure that changes to the code are working as expected.

1. **Functional testing:** Functional tests verify if a piece of software is acting in accordance with pre-determined requirements. It’s only concerned with validating if a system works as intended.
2. **Integration testing:** Integration tests verify that different modules or services used by the application work well together. E.g., it can be testing the interaction with the database or making sure that microservices work together as expected.
3. **Acceptance testing**: Acceptance tests are executed to verify if a system satisfies its business requirements. They require the entire application to be up and running and focus on replicating user behaviors.
4. **Performance testing**: Performance tests determine the stability, speed, scalability, and responsiveness of an application under a given workload. They are executed to identify and eliminate the performance bottlenecks in the software application
5. **Smoke testing:** Smoke tests are basic tests that check basic functionality of the application. They are meant to be quick to execute, and their goal is to assure that the major features of the system are working as expected.
6. **Regression testing:** Regression tests confirm that a recent program or code change has not adversely affected existing features. It is a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
7. **Security testing:** Security tests uncover vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss.
8. **Accessibility testing:** Accessibility tests ensure that the application is usable to as many people with disabilities, such as vision impairment, hearing disabilities, and other physical or cognitive conditions.
9. **Production verification testing:** Production verification tests confirm that a product meets its intended use and the needs of its users. This testing phase only takes a few hours and is concentrated mainly to ensure that the application is stable and in complete working order

**Test Lifecycle & Management**

### Functional Testing

It is defined as a type of software testing that validates the software system against the functional requirements/specifications. The purpose is to test functionalities which are developed in every sprint, by providing appropriate input, verifying the output against the functional requirements.

### How to test?

* Verification of the requirements specified
* Creating a detailed test plan
* Design and review test cases
* Executing test cases.
* Analyzing test results
* Retesting bug fixes deployed
* Sign Off by test lead/test manager

### Integration Testing

It is defined as a type of testing where individual software features that were developed during functional testing are integrated logically and tested as a group. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

### How to test?

* Preparing an effective integration test plan and identifying unit interfaces
* Identifying critical modules to test on priority
* Design and review test cases, test scenarios and test scripts
* Running the tests on the integrated modules
* Tracking bugs and record the test results of tests

### Regression Testing

It is defined as a type of software testing that ensures an application still functions as expected after any code changes, updates, or improvements. Regression testing is responsible for the overall stability and functionality of the existing features.

### How to test?

* Build your regression suite
* Select a regression testing approach (preferably automation)
* Select test cases for the regression suite
* Decide the frequency of the test runs (weekly runs are recommended)

**Test environments**

### What is a test environment? Let’s consider how a typical software application gets deployed and managed. There are times when developers make clones of the main codebase and deploy them separately to iron out bugs and glitches in different virtual machines. The location and state of these parallel applications are what we're calling test environments.

### Types of environments

We typically have four environments along any software's lifecycle.

* Development
* Testing
* Staging
* Production

Let’s look at the details of the testing done in each of these environments:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of environment** | **Owner** | **Basic tests done** | **Entry Criteria** | **Exit Criteria** |
| Development / DEV | Development team and Testing team | Unit testing, Functional testing | Test cases created, reviewed and test data prepared | Test cases of all testing type (Functional, UI, Compatibility, and Localization) are executed. |
| Testing / QA | Testing team | Integration testing, Regression testing | All the Functional and UI test cases are executed | All test cases passed, bugs fixed and verified |
| Staging / UAT | Client delivery / Release management team | User acceptance testing | Sign off from the testing team | All business scenarios are executed, and passed |
| Production / PROD | Client | Smoke-testing (**optional**) | Release package is ready for deployment | Release deployment successful with all smoke test passed |

# Defect lifecycle

Terminology of a defect may vary from team to team depending on the project configuration.

**Stages of a defect**

Following is a generalization of the states of a defect: -

* **New/To-do**: Defect is logged and documented for testing. It can also have this state when re-opened.
* **Open/In-Progress:** Developer starts fixing the defect.
* **Done/Verified**: Defect is fixed
* **Removed:** Identified defect is either rejected, deferred or duplicate.

**Process of managing defects**

* While testing the application, the tester identifies a defect and logs it in the product backlog.
* The defect is sent to the project manager so it can be prioritized & triaged
* The project manager decides whether the defect is valid and shifts it to sprint backlog depending on the severity (Critical/High/Medium/Low) and priority (1,2,3,4)
* If the defect is added to the sprint backlog, then it is assigned to the development team.
* When the defect is fixed, the testing team ensures that it is working as expected.
* Test lead signs it off making sure all the requirements are met.

# Sign Off

The formal stamp of approval, **declaring the completion of testing**by the test lead or test manager is called Sign Off.

**Functional Sign Off**

Once the test cases are drafted, a sign off is a declaration for the test cases to have complete test coverage for a specific functionality. After the test cases are executed, and bugs have been verified, the test lead/manager gives the final testing sign off declares the functionality ready for release. The test reports are communicated to all the stakeholders via emails, test management tool or a formal communication medium.

**Integration/Regression Sign Off**

After the product is thoroughly tested, a sign off task acts as a mark of completion of testing. A release dashboard is created containing graphical representations of all the tests that were performed and test reports are shared with all the stakeholders via emails, or a formal communication medium.

**Production Verification Sign Off**

It is done during the release deployment to ensure that the application that there are no critical issues. A basic smoke test on all the major functionalities on the client site to make sure everything is working as expected. There is no client data involved, and the results are stored for documentation purposes only.

# Appendix

|  |  |
| --- | --- |
| **Abbreviation** | **Definition** |
| DEV | Development |
| QA | Quality Assurance |
| PROD | Production |
| UAT | User Acceptance Testing |