Call us on: +919830995005
Email: contact@remoteprogrammer.com

Software Quality Management at RP

We are set to deliver our software applications better than our competitors at the minimum possible budget by adopting the well-defined quality management processes. We have definite quality assurance, quality planning, and quality control policies to deliver a software application with the right quality.

Quality Assurance

Quality assurance refers to the organizational processes to assure the quality of the software to be delivered.

At Remote Programmer, we attempt to assure quality in two simple ways.

1. **Build Quality In -** An proactive approach to add quality in the first place
2. **Post Dev Quality Improvement -** a reactive approach to rectify the defects which could not be identified in the first place.

Build Quality In

Let's start with an example to explain what is built-in-quality. An iPhone costs approx $1000 while a Mi phone just costs $100. Both provide similar functions. And you can never turn a Mi phone into iPhone by fixing defects. Because there is a difference is in the built-in-quality.

To add built-in quality in your software application you need to build it right in the first place. To build it right you need to define it right. Both the functional and non-functional requirements. And establish a proper validation and verification process to make sure that what is defined is delivered finally.

Here are the steps we undertake to add built-in quality

1. **Establish a Close Feedback Loop -** to add more customer value in the application quickly
2. **Deliver Fast to get Fast Feedback -** Choose 20% features that account for 80% value and can be defined well to reach the customer's fast. Avoid features that have confusion.
3. **Agile Requirement Management -** To remove mistaken ideas at an early stage
4. **Automated Software Modeling -** To remove mistakes in the design and architecture
5. **Test-Driven Development / Design (TDD) -** To code it right at first attempt
6. **Checklist Management for Verification -** Verification system to make sure all points are done
7. **Integrated Test Management System -** To make the testing & validation process manageable

Post Dev Quality Improvement

Whatever measures we take to make things right in the first place there will be some omissions or bugs in the system. As per software development dynamics, requirement changes with time at a rate of approx 1 to 1.5% per month. This is called requirement creep. So we must have a system to record the bugs, anomalies, omissions, and requirements creep such that these technical debts can change

impossible.

We normally adopt 3 continuous processes to take care of it

1. **Setup Defect Tracking System -** We have a built-in defect tracking system that can be integrated with the live project. One can submit a defect from the application.
2. **Setup Production Telemetry -** This is an automated proactive system that collects all the possible discrepancies in the system and sends notifications to the right set of people automatically - action can be taken before the actual problem happens.
3. **Continual Learning & Experimentation -** Explore better ideas, technologies, and techniques and apply them to achieve a better quality

Software Quality Planning

Quality planning is the task of determining what factors are important to a software project and figuring out how to meet those factors. A software quality plan is project-specific. That means it differs from project to project. Depending on the thrust areas, type, size, and complexity level of the project. We need to do the following to complete a project quality plan

1. Describe the project objectives and quality expectations.
2. Identify the organization’s quality policies such as ISO 9001
3. Identify other quality criteria or policies that may need to be followed, such as the requirements of clients or investors.
4. Define the acceptance criteria for the project deliverables
5. Define quality management roles and responsibilities.
6. Identify the standards that will apply.
7. Identify and list the quality metrics that need to be tracked.
8. Describe the monitoring, and reporting tools, procedures, and the process for delivering continuous improvement.
9. Describe the design and review tools & procedures
10. Describe the development tools, procedures, and processes
11. Describe the testing and quality assurance tools & procedures
12. Describe procedures for dealing with defects.
13. Describe the project acceptance processes and procedures
14. Describe code control procedures.
15. Describe change control procedures.
16. Set out any quality training requirements.

We have an in-house product development cum project management software that automates most of the above processes with standard practices.

Software Quality Control

Quality Control refers to the activities and techniques to verify that the developed product is in conformance with the requirements. The ultimate output of both processes is to deliver a quality product.

We at Remote Programmer attempt to add quality at multiple steps and often in an iterative manner agile values in mind. So there is a good chance that clutters are removed from requirements, bugs are identified and fixed and delivered product is developed with built-in quality. Here are the standard quality control processes.

MVP: Choose 20% of Features that account for 80% Value

Let me ask a simple question. How many features of MS word you normally use to do most of your regular works? I can bet, it should not be more than 10% to 20% of the total.

That is it. As per product management dynamics, 20% features adds 80% values. So if we can choose the 20% features of your product, it can be developed by 20% (maybe lesser too) of total time and launch in the market 80% ahead of people who attempt to develop all and launch. People who pick the lesser important features will end up with much lesser product value. We help our clients in this process.

Iterative Reviews of Requirement & Acceptance Criteria

We use the agile requirement management tool for developing requirements and managing the same. We also have an integrated test management tool where TDD Test Cases can be written ahead of development and exported as a checklist during coding. There is always a close feedback loop with the developers and product owners, so all the functional requirements, non-functional requirements, and associated acceptance criteria evolve towards perfection before actual development. Wrong logic and less important feature points are removed automatically.

Iterative Design & Solution Plan Reviews

We have an integrated modeling tool where the flow diagrams (DFD and ERD) can be drawn dynamically. Any change in the relation logic automatically affects the diagrams which can be reviewed by the product owner easily and can find a flaw if any. We also use third party modeling or wireframe software to model the look and behaviors of the future application. These are reviewed by the experts where applicable. This way we can remove any design or architectural flaws if any at the early stage of the feature development.

Development with Verification Checklist

Each story or defect can be populated with multiple checkpoints. A checkpoint can a step or a TDD point (smallest requirement logic unit). This is just to reduce the chances of omission errors. These checkpoints are kept transparent with product owners such that he can see or edit any time.

Agile Testing Processes

Apart from iterative review and verification before coding or during the coding stage, we have processes for test management to validate the requirements and acceptance criteria. The testing can be done by the developer, by his peer, by the scrum master, by the product owner, or by a professional tester, depending on the situation.

Test Case Types

The most important step of testing is writing the test cases. In modern test-driven design practices test cases are requirements and acceptance criteria (specifications). This is part of the requirement development and management. These can be broadly divided into 3 categories

In broad senses, all test case types can be divided into 3 types

1. **Functional Test Cases -** directly connected to a software event or user action
2. **Non-Functional Test cases -** Defines the overall quality of the application or a component.
3. **UX Test Cases -** In between two. Not directly connected with user actions but affects the user experience

All tests (except exploratory) are done against test cases.

Unit Testing

It focuses on the smallest unit of software design. Such as the method of a class or component. Mainly functional teste cases are tested. But it can test performance like throughput too. This is done by the developer only based on the test case written as TDD requirements.

It can be executed manually or automated. Unit tests are in general quite cheap to automate and can be run very quickly by a continuous integration server.

We generally use the PHP Unit, or Codeception for this purpose.

Functional Testing

Functional tests focus on the business requirements of an application. They only verify the output of action and do not check the intermediate states of the system when performing that action.

This is done by the developer, by the peer, by a professional tester, or by the product owner against the functional test cases written. Normally in the agile method, it is done storywise or feature-wise.

Functional testing can be executed manually or using automated tools like **selenium drivers**. The best practice is to automate as more as possible for a large application.

Lean 5S Testing

5S special kind of testing to minimize the waste in using the application. The test cases focus on 5 practices

Sort - This is to ensure that a user can avoid seeing pieces of stuff that are not relevant to him/her. By giving proper filter options or by marking different kinds of stuff with different colors.

Set in order - This is to ensure a user sees data/information in proper order throughout the software. In the dropdown, search lists, menu, forms, etc.

Shine - This is to keep both the screens and remove any sort of clutter where possible.

Standardize - There should be standard rules for every type of software behavior. Unpredictable

behavior will simply kill the user's time.

Sustain - Practice above 4 thoughout the software application life cycle

Usability Testing

Usability Testing also is known as User Experience(UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use a software application to expose usability defects. Usability testing mainly focuses on user's ease of using an application, the flexibility of application to handle controls, and the ability of the application to meet its objectives.

Load Testing

Load testing specifically tries to identify how the application behaves under expected loads. Therefore, you should first know what load you expect for your application. Once you know this, you can start load testing the application.

Load testing is too important for an application that expects huge numbers of data and concurrent connections. If not tested in early-stage you may have to rewrite the full application.

We use Jmeter, A/B tools for load testing. This is to be done by an experienced load tester.

Security Testing

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

This is to be done by a person who is experienced in security testing. We use "sonarqube" automated tool for security testing.

Maintainability Testing (Code Quality Testing)

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

Acceptance testing

Acceptance tests are formal tests 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. But they can also go further and measure the performance of the system and reject changes if certain goals are not met.

Exploratory testing

EXPLORATORY TESTING is a type of software testing where Test cases are not created in advance but testers check the system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a "thinking" activity.

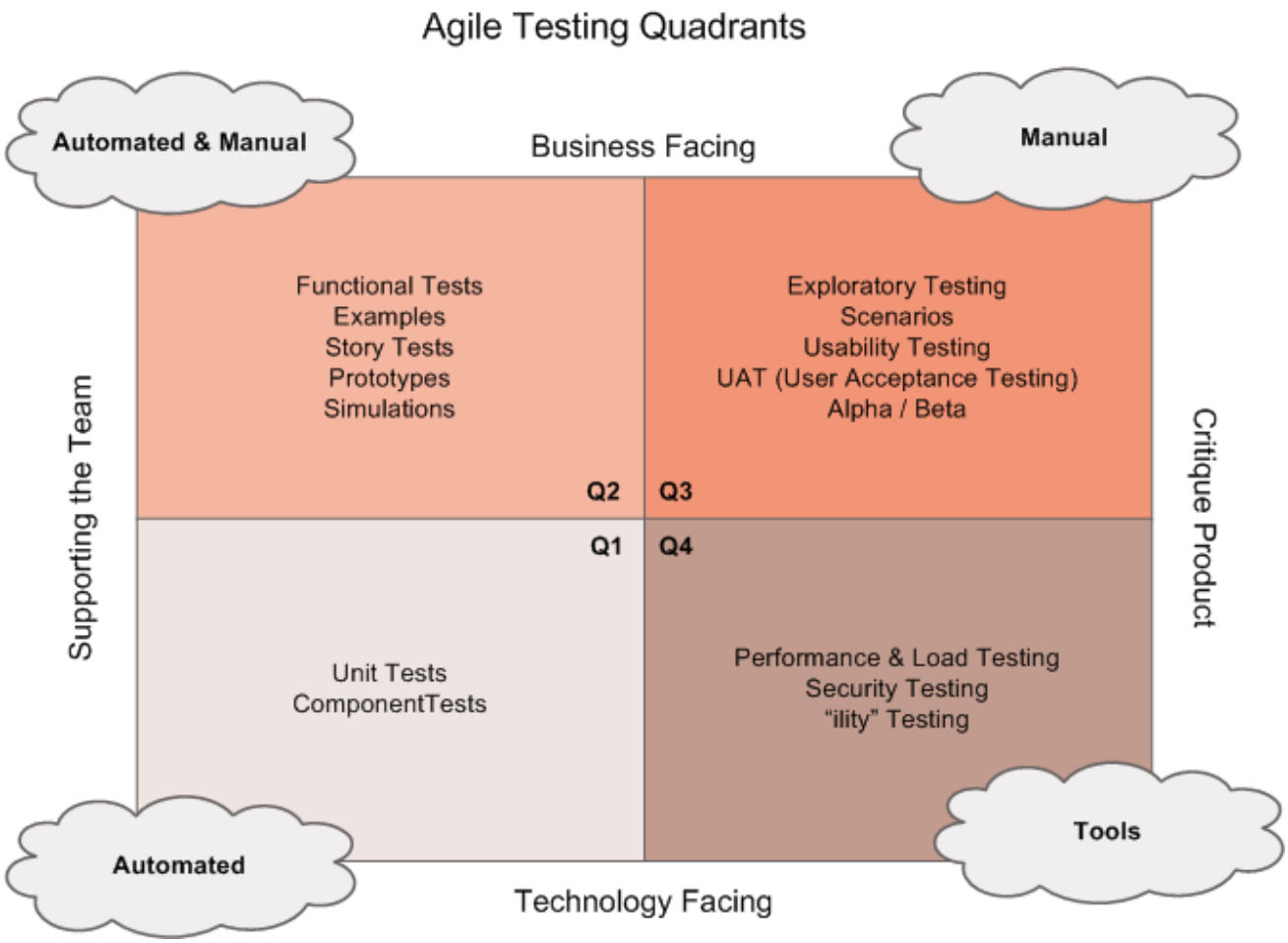
This is best to be done by the product owner as he is the final person to understand the requirement.

Defect Tracking and Fixing

A test either can pass or fail. If fails, a defect should be added in the defect tracking system which should go to the developer for fixing. A defect can be added during the time of exploratory testing too. Also, a defect can be added from the live application too.

We have an integrated defect tracking system and even it can be hooked in the live application such that anyone can report a defect from the live site. Once added, it can be moved as a task, subtasks, or a checklist and resolved accordingly.

Agile Testing Quadrants: What to test and when?



The Agile testing quadrant divides the entire testing process into 4 parts. This makes the Agile testing process easy to understand.

Among the 4 quadrants, the left 2 tell the testers which code to write and the right 2 quadrants help them understand the code better with the help of feedback to the left quadrants.

Quadrant 1

This quadrant focuses on the quality of the code. It includes test cases and test components that are implemented by the testers. These test cases are for automation testing to help to improve the code.

Quadrant 2

This quadrant contains business-driven test cases which are also implemented by the testing team. The main focus of this quadrant is on customer requirements. It improves the business outcomes of the software being created.

Quadrant 3

This phase provides feedback for the previous two phases. There are many iterations of reviews and feedbacks carried out in this quadrant which helps to strengthen the code. Usability tests, exploratory tests, user acceptance tests, and collaborative tests are performed in this quadrant.

Quadrant 4

The non-functional requirements of the code, such as performance, security, scalability, etc. are taken care of in this quadrant. Testing for stress and performance is carried out in this phase. Security and infrastructure test, data migration, and load testing. This quadrant makes sure that the code satisfies all the non-functional requirements.