

Software Quality Assurance Plan

Company 4

Author: Quality Coordinator

December 7, 2020

Version 2.4

| Name | Date | Change | Version |
|---|------------|---|---------|
| Della Baby | 2020-10-22 | Initial Draft | 1.0 |
| Della Baby | 2020-11-16 | Reworked on the whole document.Changed the content and added content based on quality assurance plan. Added quality guidelines for each processes.Added workflow description.Removed roles and responsibilities and SQA tasks. | 1.1 |
| Della Baby | 2020-11-18 | Added some more content to the development quality section. Added a new section for information flow and quality analysis. | 1.2 |
| Della Baby | 2020-11-26 | Revised document based on comments received after document review.Process workflow chart is updated. Added more content to process workflow. Added more content to section 3 by adding chart and metrics. | 2.0 |
| Della Baby | 2020-12-01 | Revised document based on comments received after document review. Added two subsections under section 3. | 2.1 |
| Della Baby, Uno Österman | 2020-12-02 | Added more descriptive content to process workflow. | 2.2 |
| Della Baby, Filip Brunander, Alexander Bois | 2020-12-04 | Revised the structure of the document, added references to other documents, added more information to document quality, made additions to requirement quality, changed the code quality about writing code, added information about Design quality and UX testing, Removed information that can be found in other documents | 2.3 |
| Alexander Bois | 2020-12-07 | Added last links to references, added content in the process workflow, to the first and last points, Corrected spelling mistakes | 2.4 |

Contents

| | | |
|----------|------------------------------------|-----------|
| 1 | Introduction | 4 |
| 1.1 | Definition | 4 |
| 1.2 | Purpose | 4 |
| 1.3 | Scope | 4 |
| 2 | Process Workflow | 4 |
| 2.1 | Traceability | 5 |
| 3 | Quality Action plan | 6 |
| 3.1 | Requirement Quality | 6 |
| 3.2 | Document Quality | 7 |
| 3.3 | Code Quality | 7 |
| 3.3.1 | Writing code | 7 |
| 3.3.2 | Code review process | 8 |
| 3.3.3 | Refactoring | 8 |
| 3.4 | Design Quality | 9 |
| 3.5 | Product Quality | 9 |
| 3.6 | Test Quality | 9 |
| 3.7 | Information Flow | 9 |
| 4 | Quality Analysis/Evaluation | 10 |
| 4.1 | Code Quality Analysis | 10 |
| 4.2 | Test Quality Analysis | 10 |

1 Introduction

1.1 Definition

In the context of software engineering, software quality measures the quality of the software design and, it is a degree of conformance to that design. It can also be defined as the ability of the software to perform as per customer requirement.

1.2 Purpose

The quality assurance plan aims to help the project team members understand how all can work together and deliver a good quality product. The plan will guide how the project's quality should be in each stage, like requirement analysis, design, development, testing, and deployment.

1.3 Scope

The quality assurance plan establishes the SQA activities performed throughout the development cycle of the software product. This plan's primary goal is to ensure that each product's stages, such as the requirement, document, code, design and development, and test, follow the quality guidelines to achieve the quality of the entire product. The quality assurance plan will focus on the code quality, and test quality using metrics specified in section 4.

2 Process Workflow

Here the company's Process workflow is described. During these different parts many different actions will be taken to ensure quality throughout the process. First we describe the process and later the actions taken to ensure quality in different parts of the workflow.

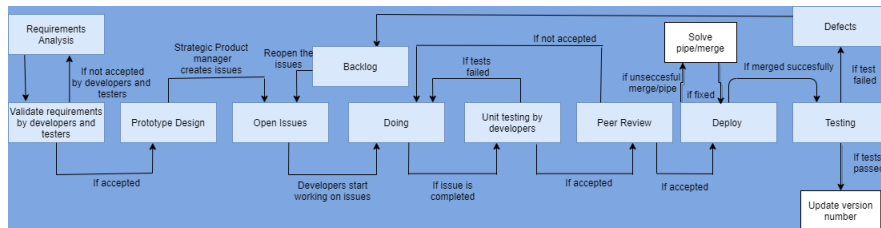


Figure 1: Process Workflow

1. Starting from the left the analysts will have a conversation with the customer, brainstorm and define user-stories, functional- and non-functional requirements to define and refine what should be included in the product. This is done both internal and in cooperation with the customer in different phases.

2. The strategic product manager, testers and developers then review these requirements in order to give an opinion on how it can be implemented and tested and how much time it will take.
3. If accepted the requirements and user-stories are concertized by the designers in the design prototype, which is reviewed after each sprint to match the requirements.
4. The architects, together with the developers and the strategic product manager, then works to divide the requirements into smaller tasks for implementation in line with the design prototype. The architects and developers defines the components and views in a document while the strategic product manager distributes the tasks between the cross-functional teams. This is done ahead of each sprint.
5. The cross-functional teams further break down the tasks into issues in GitLab, each issue is then assigned to developers in the team. After the issue has been implemented successfully the code is reviewed by another group member or developer.
6. If the issue is successfully implemented, accepted and goes through the pipeline, a merge request is made. The merge request is then reviewed and any problems that occur for the merge request are solved before merging with the master branch. If the merge request is successfully merged with the master branch it is automatically deployed, via GitLab pages in the pipeline.
7. At the end of each sprint the testers have a test session. The testers will then begin different testing types such as Unit, System/Integration, and User testing. If any defects or bugs are found they create a new issue that is labeled "Bug" or "Defect", these are then added to the backlog for the next sprint. If the code passes and fulfills the requirements it becomes verified. The Quality Coordinator perform a quality analysis of the tests and the code. Feedback on the product from the customer is received in the end of each sprint. That feedback goes through the process in the next iteration, starting in step 1 explained above.

2.1 Traceability

To ensure traceability we will use GitLab with the tool called "Issues" for listing and organizing tasks. The issues will be created according to the [Process Git Workflow](#) to ensure traceability to the requirements. This is the action taken in relation to item 5 in the list above.

3 Quality Action plan

The quality actions to be taken for different areas of the project artifacts are given below.

3.1 Requirement Quality

The analyst team will investigate the true needs of the customer and work on the requirements accordingly. The functional and performance requirements will be added to the Software Requirements Specification document. Anytime the analyst team considers a new requirement they shall fulfill the answers to the following questions.

| | |
|-----------------------|---|
| Is it necessary? | The requirement shall be required to fulfill the customers needs for the product. |
| Is it clear? | The requirement shall have a clear scope be unambiguous. |
| Is it understandable? | The requirement shall be written in a complete sentence consisting of predicate and subject. Do use words like "shall", "must" and "will" to leave out ambiguity. |
| Is it testable? | The requirement shall be testable. It shall be clear what is the input/action that triggers an output/response. |
| Is it small? | It is easy to estimate and work with a short and precise requirement. |
| Is it independant? | Every requirement can be implemented separately, which helps roll out projects faster and constantly deliver improvements |

Table 1: Requirement Quality Checklist

If the requirement fulfill the answers to the above questions they shall be ready for use. After that, the Strategic Product Manager shall validate that the requirement is necessary and independent before the start of a new iteration. The developers and the testing team will validate that the requirement is testable, small, clear and understandable when the requirement is assigned to them. This will ensure that the requirement satisfies the required level of quality. If not these members or teams will reach out to the analyst team with feedback on the requirement.

3.2 Document Quality

Document quality is the value of a document to its target audience. It is important that the content of the document should be of good quality.

In our company all the documents are authored by a person/team who has good knowledge in that area. For example, Software Requirement Specification document is handled by analyst team. When a document is considered ready to be published as a new version (from X.Y to X+1.0) by the author it must go through the [Document Review Process](#). This process will increase the chances of better quality of the documents. The Project Manager will select members with consideration to the difference of perspective and skill to conduct the review. After the review the author improve the document based on the feedback and publish the new version.

The document quality has several dimensions as follows.

- Structure
- Readability
- Clarity
- Correctness
- Completeness
- Information Density
- Precision
- Conciseness
- Complexity Hiding

The colleague who verifies the document will check that the document follows all the good quality dimensions.

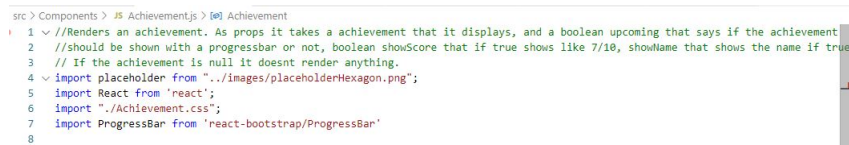
All the documents that is considered "Output documents" should conform to the document template of our company. The PDF documents together with older versions will be available in Microsoft Teams General channel under Files/Output, and their .tex-files is available in the company's git lab repositories.

3.3 Code Quality

3.3.1 Writing code

Top quality software make sure that the product is free from bugs and it maintains the code quality standard. It increases the efficiency of the product. The developers and testers shall follow set coding standards when writing new code.

- Clarity - The code should be readable, consistent in indentation and syntax, and as simple as possible. It should be understandable to anyone who has not written the code themselves.
- Comments - Add comments to the part that are not self-explanatory. The comments shall be clean and readable to explain confusing areas in the code further. The comments shall be brief and relevant. Inline commenting is the most common one. The most beneficial use is a simple-minded explanation for small functionality. Header comments are useful in source code for simple explanations of what to expect in that file.



```
src > Components > JS Achievement.js > [0] Achievement
1 //Renders an achievement. As props it takes a achievement that it displays, and a boolean upcoming that says if the achievement
2 //should be shown with a progressbar or not, boolean showScore that if true shows like 7/10, showName that shows the name if true
3 // If the achievement is null it doesnt render anything.
4 import placeholder from "../images/placeholderHexagon.png";
5 import React from 'react';
6 import "../Achievement.css";
7 import ProgressBar from 'react-bootstrap/ProgressBar'
8
```

Figure 2: Header Comments

- Naming Convention - The company does not follow a set naming convention. When naming the developer/tester shall consider if the names used clearly indicates its intention. File names shall be consistent, short and descriptive of what part of the product the content handles.

3.3.2 Code review process

Cross-functional teams are handling the development of issues in our product. After completing the development process, the code is peer-reviewed by another member of the team. By having developers look at and critique each other's code, issues can be found and documented, leading to improved code quality once the issues are resolved. The developers are encouraged to share experiences and best practices through this process, with the goal of generating higher-quality code.

3.3.3 Refactoring

To improve the readability and maintainability of the code base. The development team plans for "code cleaning"-events where they refactor the code base. They remove unused code and branches that are no longer used in the repository. The latter to ensure that time is not wasted on maintaining code that is no longer relevant. The unused branches are deleted automatically from git after merging it to the master branch. If one never merges the branch, the configuration manager will manually go in and delete the branch after confirming with the person who created the branch. These activities will take place as needed, but approximately every other sprint.

3.4 Design Quality

To ensure quality of design the company is using two actions. The first is Customer meetings where the customer presented with the latest design prototype and able to give feedback. This ensures that the customer can provide opinions to guide the design to be more in line with their vision. The second action that is taken is to conduct UX testing sessions with possible users. These tests follow the [Test plan](#). With the results from the UX testing the design is updated to conform to the feedback received.

3.5 Product Quality

To ensure product quality we take a number of actions that are defined in the [Test plan](#). This includes all the testing effort made. These actions are taken by the testers continuously during the development. The testing process ensures that the product is free from defects that lead to improve the product's performance with good quality.

3.6 Test Quality

The testers create the document [Requirement Traceability Matrix\(RTM\)](#), test cases and defects list based on the actions in the [Test plan](#). Based on the RTM provided by the testers, the quality of the tests will be done by using some test metrics after each sprint by the quality coordinator. The quality coordinator will make a quality analysis report by using the test quality metrics after each sprint. The quality analysis document can be found under General/SQA.

3.7 Information Flow

The structure of the company is that we have three cross-functional teams with a scrum master each. We also have a Testing, Developing and Design Team, see the [communication map and clarification](#) for more information. The information flow between all groups is made in several ways. We are using Microsoft Teams to communicate with other members of the company. We have one channel for each cross-functional team as well as for other teams such as managers, analysts, developers, testers, designers and scrum masters. We spread the information as a meeting, chat, or call.

Every week the company has several meetings. Managers meet every week to discuss our situations and how we should act accordingly. Also, we have a scrum masters meeting where information from the core management to the teams and vice versa is discussed. Twice or thrice a week the cross-functional teams have scrum meetings to discuss their progress and what they need to do next. At the beginning of each sprint, the cross-functional teams will have sprint planning and sprint retrospective meetings. In addition to this, the whole company is scheduled for a company meeting once a week which is mandatory.

The meetings are helpful to understand the status of the entire project, to make an evaluation of the quality of each process and to make decisions based on the information collected by the involved members.

4 Quality Analysis/Evaluation

After each sprint, the quality coordinator will do a quality analysis. The quality of the code and test with the help of code and test quality metrics.

4.1 Code Quality Analysis

The code quality will be analyzed by using complexity calculation and number of lines of code. To write, debug, and maintain code is necessary for high software quality. Moreover, high code complexity brings a higher level of code defects, making the code difficult to maintain. By using CodeMetrics tool in Visual studio code, the complexity of code in each functions can be calculated. The following graph shows the code complexity in functions.

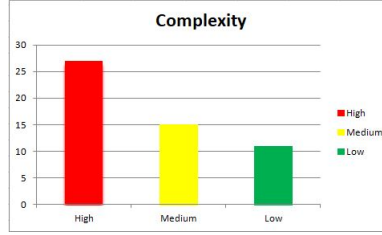


Figure 3: Code Complexity

The red bar indicates that the number of functions with higher complexity of code. The yellow bar indicates the medium and green bar indicates lower complexity of codes. The quality coordinator will inform the developers about the code complexity, and the developers will rework on the functions to reduce the complexity to provide good quality code.

4.2 Test Quality Analysis

The test quality is analyzed using the following metrics with the help of RTM.

1.

$$PassedTestCasesPercentage = \left(\frac{NumberofPassedTests}{TotalNumberofTestsExecuted} \right) \times 100$$

2.

$$FailedTestCasesPercentage = \left(\frac{NumberofFailedTests}{TotalNumberofTestsExecuted} \right) \times 100$$

3.

$$BlockedTestCasesPercentage = \left(\frac{NumberOfBlockedTests}{TotalNumberOfTestsExecuted} \right) X100$$

4.

$$FixedDefectsPercentage = \left(\frac{DefectsFixed}{DefectsReported} \right) X100$$

5.

$$CriticalDefectsPercentage = \left(\frac{CriticalDefects}{TotalDefectsReported} \right) X100$$

6.

$$TestExecutionCoveragePercentage = \left(\frac{NumberOfTestsRun}{TotalNumberOfTestsToBeRun} \right) X100$$

7.

$$RequirementsCoverage = \left(\frac{NumberOfRequirementsCovered}{TotalNumberOfRequirements} \right) X100$$

The quality analysis report will help to understand the improvements required for the project. The detailed report of code and test quality analysis can be found under General/SQA.