SID – The Smart Investment Dashboard

C. Malcolm Todd – T00232792
Saloni Saluja – T00608615

QUALITY MANAGEMENT PLAN

PREPARED FOR:

Mr. Kevin O'Neil, TRU Computing Science
koneil@tru.ca

CONTENTS

EXECUTIVE SUMMARY

This report serves to define the internal quality management practices to be employed during the development of SID, the Smart Investment Dashboard, a software solution for self-directed investors to improve the process of self-directed investing. In developing SID, we will utilize an Agile software development process. We will also implement a thorough testing strategy using a combination of Test-Driven Development to create a battery of tests, including unit tests, system tests, and release tests, and with thorough regression testing. As a small organization, we will adhere to a strong software maintenance and evolution schedule that will be defined using aspects of the 'Agile_MANTEMA' strategy, which combines the Scrum project management mechanism with the MANTEMA strategy used in the maintenance of large projects [1].

Concerning SID as dependable software, our organization's selection of Agile methods and limited developer resources result in our inability to produce sufficient documentation as required by a dependable process; however, we will seek to provide SID with a measure of dependability. Our organization will build SID utilizing the reuse of select existing software to various extents where possible, such as application and component reuse, which includes use of the Wealthica application [2].As small Agile organization with a flat structure where responsibilities are divided equally amongst staff, our organization plans to deliver SID in 2 week increments with the delivery of SID's MVP at end of 12 weeks.

Concerning our organizations Quality Management Plan, we intend to implement certain quality controls within our process; however, as an agile organization many of these strategies will be implemented informally rather than as document-based procedures. The core of our Quality Management Plan will involve creating a quality culture within the team and to implement best practices such as Check before Check in, Never Break the Build, and to Fix Problems as we see them. Additionally, we shall employ a code review process which incorporates paired programming to help catch errors earlier in the process to improve our SID's code quality.

Regarding our quality evaluation and tracking efforts, we shall select key quality metrics which will allow us to track our process and output quality over time to help ensure quality software that builds user confidence in SID as quality software. To that end we will monitor our organizations Lead Time and Team velocity to ensure quality processes and SID's cyclomatic complexity to identify sub-standard components in need of improvement.

QUALITY MANAGEMENT

In order to ensure that SID can be developed with at an acceptable level of quality and better allow us to provide our customers with a sense of confidence in SID's quality, our organization must implement a sound quality management plan. However, due to our organization's Agile approach for the development of SID, the practices that we adopt for quality management are going to be primarily informal rather than being document based. First, our goal is to create a quality culture within the team, which should be relatively easy given our small size, where both the members feel equally responsible for delivering quality software and taking actions to ensure that quality is maintained.

Additionally, we plan to adopt several shared good practices as stated in [3] for delivering quality software to the customers such as the following:

I – Check Before Check In

As developers, both team members will be responsible for organizing their own code reviews with each other before the code is checked to the build system [3]. That is to say that each individual member of the team will be responsible for ensuring that they produce high quality work.

II – Never Break the Build

It will be our prime responsibility to ensure that the code compiles and runs correctly before integrating it with the system library. If the code doesn't produce the expected output, it may cause the system to fail. To support this, we will first perform the unit testing for the code to ensure that it works. Then, we will integrate it with the system and perform system testing so that the new code doesn't break the build of the dashboard that was running correctly before integrating the new functionality [3].

III – Fix Problems when we see them

As team members working together towards the development of the project, we will help each other discover problems or obscurities in the code that we write. The code written by one member will be cross verified by the other member. Any member who comes across any anomaly in the code will correct it, no matter whether he/she wrote the original code or not. This will help improve cohesiveness in the team and ensure that we produce quality code with the desirable outcome [3].

REVIEW PROCESS

The review process for the development of SID is also going to be rather informal. We will have a sprint review meeting after each iteration of the software has been completed to discuss problems and quality issues [3].

As per the Agile Project Plan laid out in the previous phase, there are certain tasks that require the collective effort of both the team members to produce the deliverable. To ensure that the deliverable is of utmost quality, we will take to Pair Programming. Being a team of two, this approach will be highly suitable for us. The code developed by either of the team members will be constantly examined and reviewed by the other member. Because of this, both developers will have to understand each and every single detail of the program in order to continue the development. With pair programming, we will also be able to find bugs that would not be discovered in formal inspections [3].

PRODUCT QUALITY MEASUREMENTS

Another area of our quality management plan will relate to the ability to quantifiably track select well-chosen attributes that serve as reliable indicators of the quality of our organization's processes and outputs over time [3]. By monitoring the changing trends in these metrics, we can better ensure our team's productivity and the quality of our outputs [4]. Through quality processes and outputs, we can better adapt SID to our customer's requirements and maintain our planned delivery schedule. In providing these benefits in both processes and deliverables, our goal is to provide SID's users with the desired sense of confidence in SID's quality. However, we must be very selective in the number of metrics we wish to employ given our limited resources. This section outlines our selected metrics and how they can be used to track software quality.

Chosen Quality Metrics

Our chosen quality metrics can be segmented into two varieties based on the target to which they relate. As an Agile organization, our goal is to ensure we have both quality processes and outputs; therefore, we shall start by employing a selection of each type as defined below:

Process Quality Metrics

These metrics are important because they can serve as indicators of how quickly our organization can respond to changing requirements and new functionality requests [5]. They also serve to measure the agility of the organization and will help in future scheduling and costs estimation.

Lead Time

Measure of how long it takes the organization to go from an idea to delivered software functionality which can serve as a measure of the organization's responsiveness. This metric can be improved by simplifying the organizations decision-making processes [5].

Team Velocity

Measure of how many units of work the team is able to complete within a development iteration and serves as a useful metric for planning purposes; however, it should not be used to benchmark against other teams as it includes certain non-objective measurements [5].

Output Quality Metrics

This type of metric can be used to identify components or areas of source code that are below our desired standard [3]. Due to our need to focus on incrementally development of new functionality, we will measure code complexity so as to identify these types of components that will be hard to test and change in future Sprints.

Cyclomatic Complexity

Measure of the number of linearly independent paths through a section of code or a component and can be illustrated graphically using a Control Flow graph [6]. This metric is a code complexity metric that is correlated to a number of common coding errors [6].

CONCLUSION

To conclude, this report outlined the planned measures and metrics within our organization's Quality Management Plan. This plan will incorporate informal strategies to align with our Agile process and includes building a quality culture within the organizations, and the use of best practices such as Check before Check in, Never Break the Build, and Fix Problems as we see them. Our approach will also employ a code review mechanism that incorporates paired programming to catch errors earlier in the development process.

We will also employ the use of select reliable metrics to track the quality of our organizational processes and outputs to help build our customer's confidence in SID and the organization. These metrics include tracking Lead Time and Team Velocity to improve our processes and will help the organization to adapt to new customer requests while maintaining the agreed delivery schedule. To quantify code output quality, we will track the Cyclomatic complexity within SID's components to identify those that fall below our quality standards and could pose challenges for future change as SID is incrementally developed.

REFERENCES

[1] F. J. Pino, F. Ruiz, F. Garcia and M. Piattini, "A software maintenance methodology for small organizations: Agile_MANTEMA," *JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE,* pp. 851-876, 2012.

[2] Wealthica Financial Technology Inc., "We are Wealthica Financial Technology Inc.," Wealthica Financial Technology Inc., 2019. [Online]. Available: https://wealthica.com/about-us/. [Accessed 12 March 2020].

[3] I. Sommerville, Software Engineering (10th Edition), Pearson, 2015.

[4] Intellectsoft US, "15+ Useful Agile Metrics in Scrum & Kanban: Measure Quality, Productivity & Performance," Intellectsoft.net, 10 December 2018. [Online]. Available: https://www.intellectsoft.net/blog/agile-metrics/. [Accessed 6 April 2020].

[5] S. A. Lowe, "9 metrics that can make a difference to today's software development teams," Techbeacon.com, 2020. [Online]. Available: https://techbeacon.com/app-dev-testing/9-metrics-can-make-difference-todays-software-development-teams. [Accessed 6 April 2020].

[6] Tutorials Point, "What is Cyclomatic Complexity?," Tutorialspoint.com, 2020. [Online]. Available: https://www.tutorialspoint.com/software_testing_dictionary/cyclomatic_complexity.htm. [Accessed 6 April 2020].