SID – The Smart Investment Dashboard

C. Malcolm Todd – T00232792
Saloni Saluja – T00608615

SOFTWARE EVOLUTION PLAN

PREPARED FOR:

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

CONTENTS

EXECUTIVE SUMMARY

This report serves to define the software maintenance procedures to be employed during the software maintenance phase of SID, the Smart Investment Dashboard. For investors that use a self-directed investment strategy, SID will function as a resource by helping to mitigate or remove that strategy's current disadvantages. 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, and a battery of unit tests, system tests, and release tests. To test SID, we will make use of an established framework named Allure that supports multiple programming languages and architectures. Allure will also allow for test automation to improve SID's regression testing and reporting.

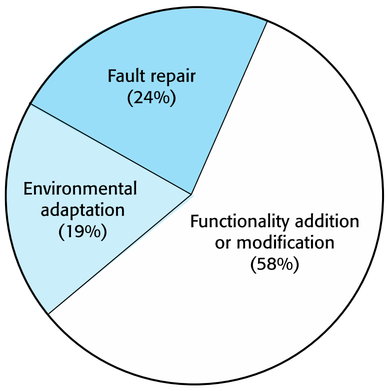
Software maintenance is a key stage of all software's functional lifespan. As such, it is critical to define a strong software maintenance process for all software projects. Additionally, there are three types of software maintenance responsible for fault recovery, environmental adaptation, and the creation of new functionality [1]. A strong software maintenance process allocates an adequate amount of time for each type and adheres to the appropriate allocation throughout the software's lifespan.

The implementation of a software maintenance process can prove challenging for small organizations that employ Agile development methods. This challenge stems from the unplanned nature of software defects which can result in missed sprint delivery goals. To mitigate this risk, we will employ select aspects of a strategy named 'Agile_MANTEMA' which combines Scrum project management mechanism with the MANTEMA strategy used in the maintenance of large projects. Agile_MANTEMA also allows to the categorization of maintenance tasks which will be valuable when scheduling maintenance activities within our limited developer resources.

SOFTWARE MAINTENANCE

After completing the delivery of SID, it is then our prime responsibility, as developers, to maintain SID in response to changes in order to extend the dashboard's functional lifespan. Throughout the lifespan, SID must remain in compliance with current industry regulations and trends. Additionally, customer requirements likely to change frequently as new features or functions become necessary over time [1]. Lastly, if any hardware or platform specifications were to change, such as a change in the operating system used by our clients, we must modify our software to accommodate the new specifications. To effectively cope with these sources of change, it is imperative that we develop a maintenance process for SID that addresses all types of software maintenance.

Types of Software Maintenance



There are three types of software maintenance techniques, namely, Corrective (Fault Repair), Adaptive (Environmental Adaptation), and Perfective (Functionality addition or Modification) [1]. A fourth type of maintenance exists named Preventative [2]; however, we will address this type when refactoring SID. The effort distribution for each of these accounts for 24%, 19%, and 58% respectively. As a small team with limited developer resources, we must implement good maintenance scheduling to ensure these relative allocations are followed.

Figure 1: Type Distribution; Source: Adapted from [5]

Corrective Maintenance

Corrective maintenance is concerned with fixing errors or defects that will be observed when the SID is in use. These errors, referred to as residual errors, prevent the software from conforming to the agreed specifications, and can result due to errors in design, logic, and coding [3]. Critical defects can even result in system failure due to an error [1]. In the event of system failure, actions must be taken to restore the operation of the software system and can consume valuable time.

An example of a software design defect in SID would be the design of a confusing and cluttered display of useless decorations that are not in line with user requirements. Logical errors could result from invalid tests, faulty logic flow, or incomplete test data. One of the potential logical errors for SID could occur if we incorrectly import investment portfolio data which would involve working with large data sets integrated from multiple sources. The need for Multiple data sources increases the chance for errors, so it is necessary to regularly test that SID is pulling data correctly.

The approach in corrective maintenance is to locate the original specifications in order to determine what the system was originally designed to do [3]. If the data represented in SID is not correct, we need to "clean" the data [4]. Cleaning data can take time. For that, we can use Excel-type formulas and macros to help identify errors in the data that might make it corrupt. In addition to avoiding duplicate data, we must perform cleaning of data by identifying outliers, incorrect data, missing data, or data that simply does not make logical sense [4].

Adaptive Maintenance

Adaptive Maintenance becomes important when the environment of SID changes. The term "environment" in this context refers to the conditions and influences which act on the system from outside of the system [3]. This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system. Updating services, making modifications to vendors, or changing payment processors can all necessitate adaptive software maintenance [5].

Being a Dashboard, SID relies heavily on external sources for data, thereby exhibiting a software dependency. Therefore, we need to ensure that SID is playing nicely with all the data and with the software that is housing the original data. Connectivity between these two sources is the key. So, Adaptive Maintenance would be modifying these external data relationships that SID has with other data housing software over time to reduce their complexity. Ultimately, they should come out to be a simple as possible. Also, we plan to make SID an online Dashboard, instead of offline, so that it exhibits platform independence.

Perfective Maintenance

Perfective Maintenance would deal with the evolution of SID's requirements and features. It would involve making functional enhancements to SID in addition to those activities which proactively increase the dashboard's performance without having to have been suggested by faults [2].

As SID gets exposed to customers, they would think of different ways to expand the information displayed by the dashboard, which in turn can become future enhancements to SID. Perfective Maintenance would also include removing features from a system that are not effective and functional to the end goal of the system. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per our customer's changing needs.

SCHEDULING SID’S MAINTENANCE

Maintenance Challenges in Agile

Implementing a software maintenance process within an organization employing Agile methods can create scheduling conflicts [6]. Agile methods rely on a schedule of development iterations; however, the occurrence of critical software defects is unscheduled. Without careful planning, these occurrences can result in missed sprint targets [6], which in turn could damage user confidence in the organization or software.

Fortunately, there exists a maintenance strategy that can employed within our small organization to help improve our maintenance processes in an Agile environment named ‘Agile_MANTEMA’ [7]. Our complete maintenance process will be a combination of time management tailored to our organizational structure and borrowed elements from Agile_MANTEMA that fit our organization.

Useful Aspects of ‘Agile_MANTEMA’

Agile_MANTEMA is a conceptual software maintenance process that is designed to speed up the MANTEMA process, which is used in medium to large projects such as banking, to make it more agile by incorporating Scrums [7]. According to [7], Agile_MANTEMA is a strategy with the following advantages:

- Combines the Agile framework for managing projects with MANTEMA’s technical processes
- Continuous and early delivery of maintained software that aligns to SID’s incremental goals
- Close work between maintainers and users that is consistent with our other Agile processes

Another useful area within Agile_MANTEMA is the classification of maintenance into two categories which are plannable and non-plannable. Incidents placed within these categories can be further segmented into service levels based on their severity [7]. We can then use this classification and service level pairing to determine when to schedule the maintenance for completion by our developers.

For more details on MANTEMA and how Agile_MANTEMA differs, please consult Appendix A.

SCHEDULING CONT'D

Allocating Developer Time

As a small team of two developers, the available number of working hours is limited. As such, the importance of an enforceable maintenance schedule is critical. For this reason, each work day will be divided amongst the three types of maintenance in the appropriate allocations. Each morning the days maintenance workload will be determined during a Scrum to identify which corrective, adaptive, and perfective maintenance tasks are to be accomplished that day.

Additionally, our maintenance schedule will provide a daily buffer of time which can be allocated to any time sensitive or pressing maintenance tasks so as to ensure no missed software deadlines or to provide additional time in the event of a critical defect. According to [6],allocating such a buffer is a maintenance best practice within small organizations.

Sample Maintenance Schedule

TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:30	CORRECTIVE MAINTENANCE	CORRECTIVE MAINTENANCE	CORRECTIVE MAINTENANCE	CORRECTIVE MAINTENANCE	CORRECTIVE MAINTENANCE
9:30					
10:30	UNPLANNED MAINTENANCE BUFFER	UNPLANNED MAINTENANCE BUFFER	UNPLANNED MAINTENANCE BUFFER	UNPLANNED MAINTENANCE BUFFER	UNPLANNED MAINTENANCE BUFFER
11:30					
12:30	LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
13:30	ADAPTIVE MAINTENANCE	ADAPTIVE MAINTENANCE	ADAPTIVE MAINTENANCE	ADAPTIVE MAINTENANCE	ADAPTIVE MAINTENANCE
14:30	PERFECTIVE MAINTENANCE	PERFECTIVE MAINTENANCE	PERFECTIVE MAINTENANCE	PERFECTIVE MAINTENANCE	PERFECTIVE MAINTENANCE
15:30					
16:30					

CONCLUSION

In conclusion, there are multiple forms of maintenance for which our team will be responsible during SID's functional lifespan. Each of these types of maintenance has their own importance on preserving the usefulness of SID and requires a schedule amount of developer resources to complete. These types of maintenance are corrective maintenance, relating to software defects, adaptive maintenance, relating to environmental change, and perfective maintenance, relating to the introduction of new functionality. To appropriately maintain SID, our organization will adhere to an allocation of 58% perfective, 24% corrective, and 19% adaptive in time spent on maintenance efforts.

To implement our maintenance strategy, we will borrow aspects of the Agile_MANTEMA process alongside small organization practices to ensure our ability to remain effective in developing new features for SID alongside any maintenance needs which arise. Using aspects of Agile_MANTEMA aligns our maintenance process to Agile development methods, and allows for effective classification of maintenance tasks for scheduling [7]. A weekly schedule of maintenance time will be utilized to handle both planned maintenance responsibilities, and a flexible buffer to protect against delays from any unplanned maintenance that would result in missed software delivery goals.

REFERENCES

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

[2] N. Quezada, "The 4 Types of Software Maintenance," Ender Technology Corp, 23 May 2019. [Online]. Available: https://endertech.com/blog/maintenance-bug-fixing-4-types-maintenance. [Accessed 5 February 2020].

[3] D. Thakur, "Types of Software Maintenance," Computer Notes and Technology Motivation, [Online]. Available: http://ecomputernotes.com/software-engineering/types-of-software-maintenance. [Accessed 5 February 2020].

[4] Cyfe, Inc, "10 Mistakes You're Probably Making in Preparing Data for Analysis," Cyfe, Inc, 2020. [Online]. Available: https://www.cyfe.com/blog/10-mistakes-preparing-data-analysis/. [Accessed 5 February 2020].

[5] CAST, "The Four Types Of Software Maintenance & How They Help Your Organization," CAST, 2019. [Online]. Available: https://www.castsoftware.com/glossary/Four-Types-Of-Software-Maintenance-How-They-Help-Your-Organization-Preventive-Perfective-Adaptive-corrective. [Accessed 5 February 2020].

[6] F. Rehman, B. Maqbool, M. Q. Riaz and U. Qamar, "Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology," in *21st IEEE Computer Society National Computer Conference*, Riyadh, Saudi Arabia, 2018.

[7] 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.

[8] M. Staron, "Dashboard development guide," Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, 2015.

APPENDICES

Appendix A

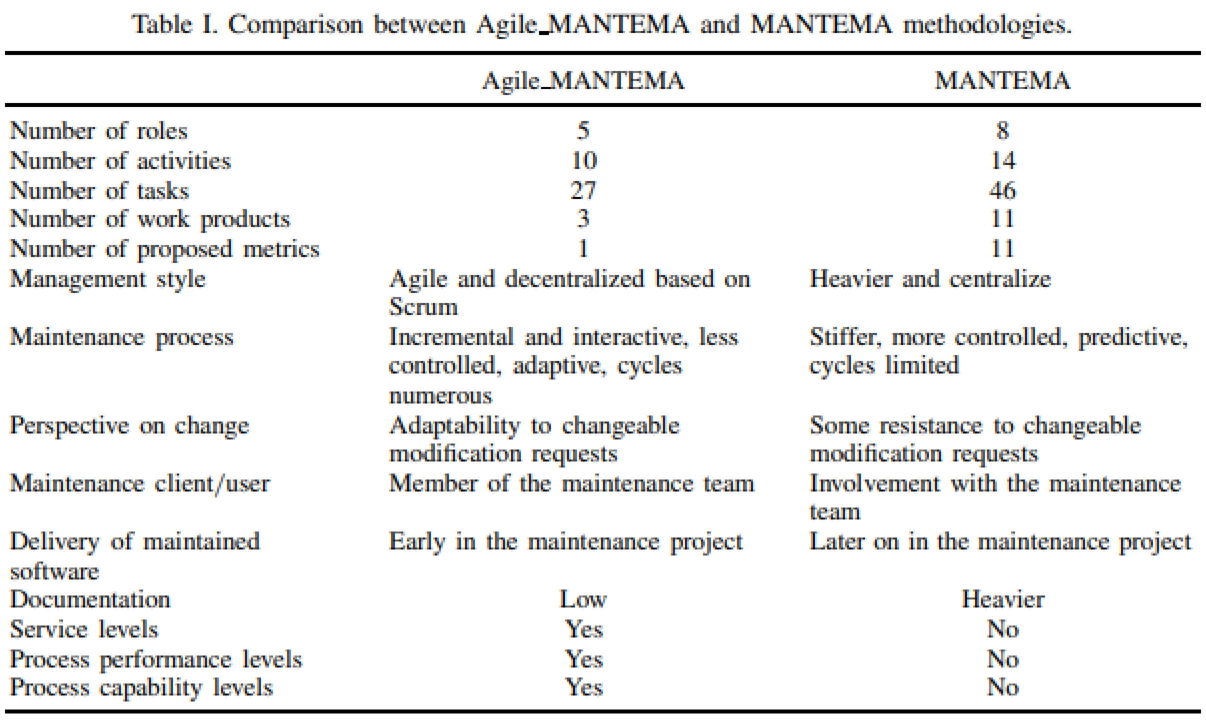


Table I. Comparison between Agile_MANTEMA and MANTEMA methodologies.

	Agile_MANTEMA	MANTEMA
Number of roles	5	8
Number of activities	10	14
Number of tasks	27	46
Number of work products	3	11
Number of proposed metrics	1	11
Management style	Agile and decentralized based on Scrum	Heavier and centralize
Maintenance process	Incremental and interactive, less controlled, adaptive, cycles numerous	Stiffer, more controlled, predictive, cycles limited
Perspective on change	Adaptability to changeable modification requests	Some resistance to changeable modification requests
Maintenance client/user	Member of the maintenance team	Involvement with the maintenance team
Delivery of maintained software	Early in the maintenance project	Later on in the maintenance project
Documentation	Low	Heavier
Service levels	Yes	No
Process performance levels	Yes	No
Process capability levels	Yes	No

Figure 2: Agile_MANTEMA vs MANTEMA, Source: Adapted from [7]

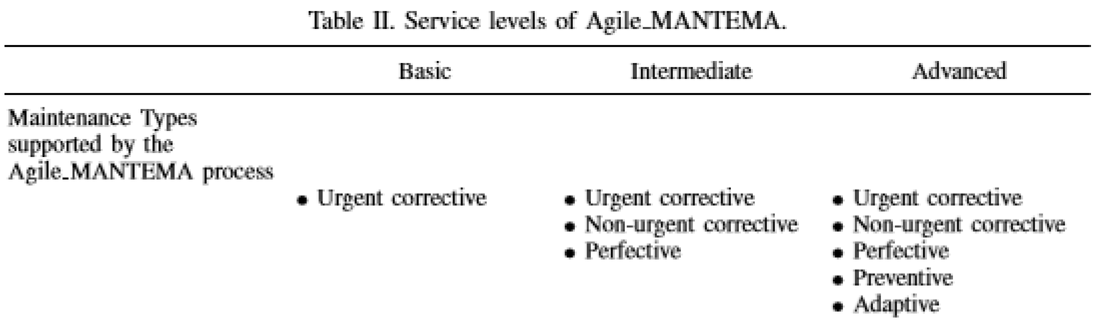


Table II. Service levels of Agile_MANTEMA.

	Basic	Intermediate	Advanced
Maintenance Types supported by the Agile_MANTEMA process	• Urgent corrective	• Urgent corrective • Non-urgent corrective • Perfective	• Urgent corrective • Non-urgent corrective • Perfective • Preventive • Adaptive

Figure 3: Service Level Classifications in Agile_MANTEMA: Adapted from [7]