ENHANCED MULTIMEDIA NOTE-TAKING RESEARCH & SUPPORT APP DEVELOPMENT

AE2GRP INDIVIDUAL REPORT

29th April 2016

Group ID: GRP-DT
Name: Yichen PAN
Student ID: 6514897
Supervisor: Dave Towey

Group members:
Shihang WANG (6514904) Yiru JIANG (6514889)
Jiahui DENG (6514883) Livia FLORENSIA (6519837)

Contents

1 INTRODUCTION

In 2014, a number of leading researchers at The University of Nottingham Ningbo China (UNNC) started a project to examine how well the university will be able to support student learning in the future. Over 2014-2015, a team of UNNC Computer Science and Computer Science Management students worked as a software engineering (SE) team (called gp-dave) and investigated the challenge and implemented a solution for the research team. Now, in 2015-2016, we, a new SE team (called GRP-DT) is required to look again at the needs of research team, and then examine and evolve the software produced by gp-dave.

We are working together to complete a full cycle of the software engineering process, resulting in delivery of a revised tool, as requested by the research project team. We have gone through a complete requirements engineering process to identify the revised SE project requirements.

2 INDIVIDUAL CONTRIBUTION AND REFLECTION ON EACH ROLE

2.1 Team leader

At the beginning of the project, I was nominated as the team leader of GRP-DT. As the team leader, I efficiently led the whole development team to complete a full cycle of the software engineering process and delivered a complete cross-platform application at the end, together with detailed documentation including software requirements specification documentation, testing report and development report.

During the occupation, I tried my best to make the whole team self-organizing so that each team member takes collective responsibility for the work instead of distributing tasks blindly. By encouraging team members to pull work for themselves instead of waiting for the leader to assign the work, they are developing a habit of working corporately ely and actively. They could manage their work as a group rather than as a separate individual. According to the Agile Manifesto, the best architectures, requirements, and designs emerge from self-organising teams, which indicates the importance of the autonomy of each team member. However, it is not easy to envolve an inexperience development team into a self-organizing team because it not only requires competency, collaboration and motivation from team members but also requires careful management, credibility and control from the team leader.

2.1.1 Monitor and manage the progress

First of all, as the team leader, I helped set and monitor the direction for the team. For example, at each stage of development, I specified the stage purpose, organisational objectives or mission that spawn the myriad of smaller tasks. Then, after the tasks were distributed, I monitored the progress of each team member to make sure that the team is consistently delivering qualified software to requirements of stakeholders. It was also necessary for me to collect and analyze data about how the work is proceeding and initiate corrective action as needed. Moreover, I was also responsible for ensuring that each team member is aware of the priority of tasks and the team is always focused on the highest priority.

2.1.2 Create a positive working environment

As the team leader, it is one of my responsibilities to motivate team members so that they could contribute to the best of their abilities. Encouraging Motivation requires organizing the work and the working environment to encourage people to work as effectively as possible. It is essential since if people are not motivated, they will lose the interest in their tasks and thus become less efficient and productive. When they become inefficient, they tend to be more likely to make mistakes, and will not contribute to the broader goals of the team or the organization. Finally, this may lead to the team destruction.

As Maslow pointed out, people are motivated by satisfying their needs (Finkelstein, 2006). These needs are mainly divided into five parts including self-actualization, esteem, love, safety and physiological, as shown in Figure 1.

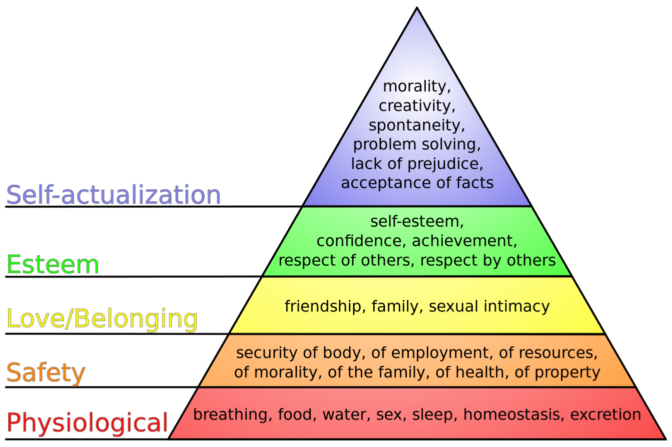

morality,
creativity,
spontaneity,
problem solving,
lack of prejudice,
acceptance of facts
Self-actualization
self-esteem,
confidence, achievement,
respect of others, respect by others
Esteem
friendship, family, sexual intimacy
Love/Belonging
security of body, of employment, of resources,
of morality, of the family, of health, of property
Safety
breathing, food, water, sex, sleep, homeostasis, excretion
Physiological

Figure 1: The branch model-Maslow's hierarchy of needs (Finkelstein, 2006)

For our team, the most important are esteem needs, which mean the needs to receive respects from others, and self-realization needs, which are the needs to feel improved and progressed during the work. To satisfy esteem and self-realization needs, I paid attention to showing group members that they are indispensable and valued by the whole team through appropriate praise and public recognition of achievements. I also gave them responsibility for their work and organized training programmes to help them promote so that they could gain new skills and knowledge.

2.1.3 Manage conflicts and promote group communications

Conflicts are common in a development team and if they are not handled appropriately, they are like icebergs and do harm to creating a non-threatening and nurturing environment. In this case, as the team leader, I carefully coped with each conflict occurred among group members and made sure they were resolved in an appropriate way, such as a face-to-face talk in a calm and neutral environment.

Group communications are also essential for the healthy evolution of the team. Appropriate internal group communications could avoid conflicts among group members resulting from lack of interaction between individuals. I often assigned two people for one task so that they could work together and got the chance to know each other further.

The informal meeting is another way I used to enhance internal communications of the team. During the informal meetings, I tried my best to prevent the meeting from being dominated by powerful personalities and let every member involve in the discussion and have the chance to give their thoughts and suggestions. Through informal meetings, group members could exchange ideas they have come up with, information on the status of their work, advice on work of others and potential changes to previous design decisions.

External group communications require that the whole team could exchange ideas with stakeholders frequently and get feedback constantly. I, the team leader, served as the bridge between the development team and the stakeholders. I represented the whole team to communicate with stakeholders, get immediate feedback, reflect problems from the development team and report progress.

2.2 Technical leader

As the technical leader of the team, I was mainly responsible for three things: underlying architecture for the software program, writing most of codes, and overseeing and directing the work being done by other group developers.

2.2.1 Configuration management

According to Sommerville (2011), configuration management is the process of managing a changing software system. It aims at guaranteeing that team members can access the project code and documents in a controlled way. It mainly consists of three central configuration management activities, including version management, system integration and problem tracking.

At the beginning of the development stage, I helped configure the Git in the working environment of each group member and set up an organization account on Github. By using Github collaboration platform combing with Git, we were able to work effectively. However, since we were not familiar with Git and Github, we often encountered various collaborative problems such as interruption from unstable or wrong committed codes and frequent commitments. In order to solve this problem, I adopted a scientific and efficient Git branch model (Driessen, 2010) and served as the Git maintainer to cope with usage problems and make sure every group member uses Git in a correct and efficient way, such as writing self-explained commitment descriptions, detailed issue report, setting clear and specific Git branches and merging two branches correctly. Under my careful maintenance, we developed more efficiently and cooperatively.

2.2.2 Configuring the underlying architecture for the software program

Due to the sudden and unanticipated change from stakeholders, we had to give up all we had developed and looked for a new development strategy. As the technical leader, I investigated all possible cross-platform desktop application development strategies, including PyQT, Adobe Integrated Runtime (AIR) and LAMP STACK, and made a careful comparison in terms of feasibility, learning curve of the new technique, benefits and drawbacks. After searching online and asking technically experienced professors, I made the decision to use NW.JS combing with MEAN STACK to develop a new web-based desktop application.

During the whole winter holiday, I learnt all related techniques and distributed learning tasks among the whole development team so that one or two group members learnt one specific group of programming languages, such as CSS3 and HTML5.

At the beginning of the spring semester, I set up the basic architecture of our application: Node.js as the runtime environment in the server, MongoDB as the database to store data, Express.js as the application framework for handling routes and Angular.js as the client side framework.

2.2.3 Cooperative and collaborative learning

Because it takes a lot of time and energy for us to learn several same things at the same time individually, I held a series of training and knowledge sharing sessions to help other group members in some specific topics. Through holding sharing sessions, I encouraged each team member to learn in group, which is much more efficient and productive because of the knowledge sharing. It is also more likely that we could come to a more complete or profound understanding by comparing with others. Moreover, it promotes more sophisticated thinking by discussion of controversial topics and encourages healthy evolution of our team. Below is the list of learning sessions that I held:

- Introduction to Worktile and how to work e ciently in group
- Introduction to Git and GitHub
- Project management and tracking
- Practical application of the GitHub branch model
- Sharing knowledge of Java Swing
- Sharing knowledge of software engineering and requirement engineering
- Sharing knowledge of advanced application of Eclipse and eGit
- Sharing knowledge of MEAN STACK and NW.js
- Sharing knowledge of GUI design

2.3 Scrum master

As the scrum master, I was mainly responsible for coaching the team and facilitating team events to deliver potentially shippable increments of product at the end of each sprint. He also takes in charge of organizing the review meeting after each sprint, where the project is assessed against the sprint goal determined at the beginning of the sprint.

From my experience, the scrum master is not a traditional team lead or project manager, but serves as a buffer between the team and any distracting influences.

2.4 Product owner

As the product owner, I took in charge of linking the communication gap between the whole team and stakeholders. During meetings with stakeholders, I communicated team status, negotiated priorities, requirements, schedule and problems, and kept stakeholders interested and satisfied. Additionally, the most important task is to prepare a product backlog, which is a prioritized features list for the product. Through the product backlog, it is possible to convey a vision of individual task to the scrum team and track development progress.

3 REFLECTION ON THE WHOLE PROJECT

Overall, I carried out all obligations of my roles and successfully led the team to go through a complete cycle of the software engineering process, from requirements engineering, software design and product development to software validation, quality assurance and software testing, and finally released a functional cross-platform desktop application, Quick Note.

According to Reid (1995), reflection is a process of reviewing an experience of practice in order to describe, analyze, evaluate and so inform learning about a practice. Thoughtful reflections are useful and essential to software engineering because software development is an experiential learning process. Our performance can be improved a lot through iterative reflection.

3.1 Team leader is not manager

When I served as the team leader, I sometimes acted like a manager, assigning work directly and ruling by authority, because it was more efficient. Although the distinction between leader and manager may be confusing, the difference between the two is that a manager focuses more on organization and keeping the team on task while a team leader cares more on the big picture and integrity of the whole team. Management involves planning and estimating the progress of the whole team, collecting, analyzing and evaluating product data and charting, controlling and reporting on progress. By comparison, the leader should pay more attention to communicating with team members, coordinating work activities, deciding collectively and boosting morale. The team leader should lead the team by coaching and teaching team members instead of driving them for tasks. Leaders and managers both need to build relationships team members. However, this kind of relationship is significantly different. Managers may need to maintain a distance from team works in order to rule them by authority. On the other hand, leaders need to be very empathetic to other team members so that they are motivated to work and pursue a common goal held by the leader and the rest of the group. In a word, the relationship between the manager and team members may be more focused on managing and manipulating the team, but for the leader, the relationship is more related to create a more positive and nurturing working environment.

3.2 Change management

For me, one meaningful aspect learnt from the project was that different kinds of changes often occur at different stages during the software development. In most cases, we need to respond to changes rather than stick to the plan or requirement specification rigidly. Having experienced the unexpected change that forced us to give up Java Swing and use NW.js, I understood that it is important to make constant deliveries to stakeholders to avoid the premature commitment to requirements for the whole system and thus allow changes to be incorporated into later requirements at relatively low cost. It is also necessary for me to embrace the changes positively and design the system to accommodate these changes flexibly. For example, we should not be afraid of changes or even provoke reaction against changes. Moreover, we also should not blindly accepte the change and adapt to it indistinguishably because not all changes are valid or beneficial to make.

Having reflected on these aspects, I gained a deeper understanding in change management process and agile methodology. In the later stage of the development process, after making SRS document, we put our focus on producing high-quality and readable code, instead of rigid documents. Whenever the change is requested, we check change requests because not all change requests require action. Then, we assess and analyze the change thoroughly.

3.3 Conflict management

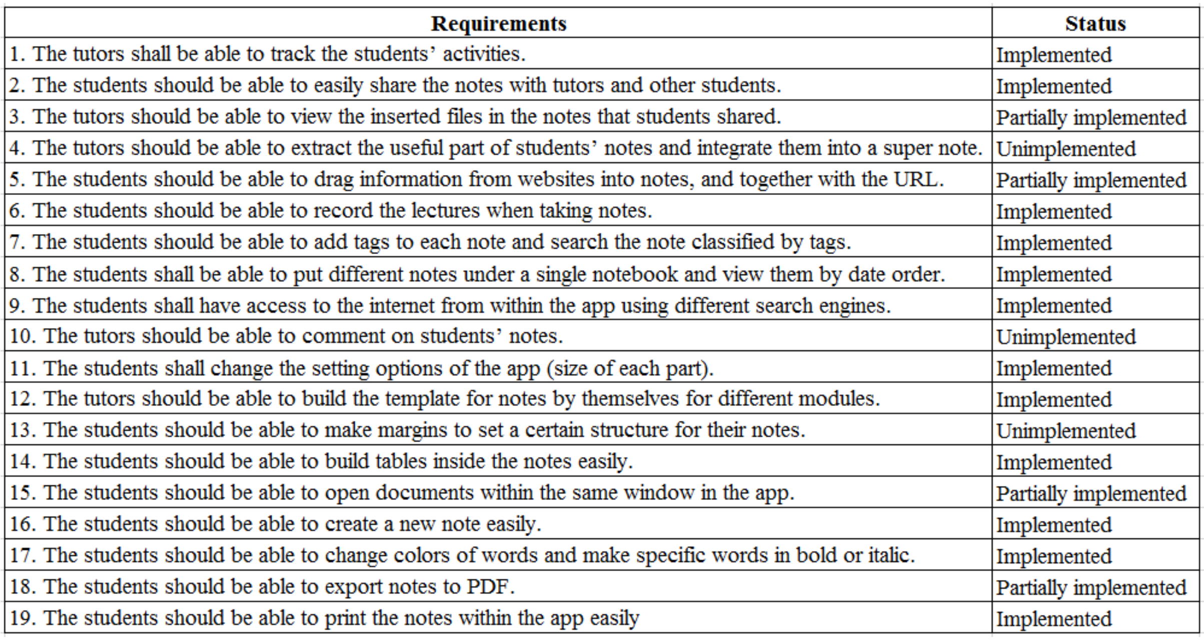
As the team leader, I came across several conflicts with one of team members. The most useful idea I learnt from those conflicts was a deeper understanding of conflict management and the role of a leader in the teamwork. First, I realized that conflicts are like icebergs, and if they cannot be resolved properly, they may have profoundly bad influence on the whole project. Direct aggressive behavior, such as approach and scolding, is useless but to make everything worse. In the conflict, I should behave in an assertive manner, which affirms my point of view without aggressively threatening others instead of an aggressive manner. Additionally, I also found that there are two kinds of conflicts: good or constructive one and bad or destructive one. Constructive conflicts can improve relationships, encourage corporation and promote open communication while destructive conflicts may damage relationships, create negative feelings and do harm to the teamwork. I also need to treat conflicts correctly, viewing them as an opportunity to think creatively, evaluate new choices and promote teamwork. Some conflicts are unavoidable but reflection on these conflicts after the crisis can sometimes result in new ways to work together, which is beneficial for teamwork.

3.4 Agile development

Since we adopted Scrum methodology in the software development stage, we came across different kinds of problems and thus I developed deeper understanding of adopting agile Scrum approach in practice and teamwork. First of all, the design of backlog of Scrum is essential because if a task is not well defined, it would be difficult and inaccurate to estimate project costs and time. In such a case, the task can be spread over several sprints, which leads to scope creep. Secondly, the original design of Scrum requires experienced team members, and otherwise, it is highly possible that the project cannot be completed in time. Furthermore, in this case, the daily Scrum meetings and frequent reviews require substantial resources and even become waste. Thirdly, a successful sprint depends on the maturity and commitment of all members, and their capability to maintain a consistently high level of morale. Finally, it is important to trust team members and defensive or overzealous management may result in a failure in team management.

4 CONCLUSION

Overall, we have realized most functionalities listed in the SRS Document. However, due to the lack of the development time and the difficulty of implementation, there are some functionalities unfulfilled as shown in the checklist below.



Requirements	Status
1. The tutors shall be able to track the students' activities.	Implemented
2. The students should be able to easily share the notes with tutors and other students.	Implemented
3. The tutors should be able to view the inserted files in the notes that students shared.	Partially implemented
4. The tutors should be able to extract the useful part of students' notes and integrate them into a super note.	Unimplemented
5. The students should be able to drag information from websites into notes, and together with the URL.	Partially implemented
6. The students should be able to record the lectures when taking notes.	Implemented
7. The students should be able to add tags to each note and search the note classified by tags.	Implemented
8. The students shall be able to put different notes under a single notebook and view them by date order.	Implemented
9. The students shall have access to the internet from within the app using different search engines.	Implemented
10. The tutors should be able to comment on students' notes.	Unimplemented
11. The students shall change the setting options of the app (size of each part).	Implemented
12. The tutors should be able to build the template for notes by themselves for different modules.	Implemented
13. The students should be able to make margins to set a certain structure for their notes.	Unimplemented
14. The students should be able to build tables inside the notes easily.	Implemented
15. The students should be able to open documents within the same window in the app.	Partially implemented
16. The students should be able to create a new note easily.	Implemented
17. The students should be able to change colors of words and make specific words in bold or italic.	Implemented
18. The students should be able to export notes to PDF.	Partially implemented
19. The students should be able to print the notes within the app easily	Implemented

Figure 2: The Checklist of Requirements

In conclusion, after experiencing the one year project, going through a full cycle of the software engineering process and practicing the team leader, scrum master and other team roles, a deep understanding in software engineering is developed. Separate peer assessment for each other group member could be found in **Appendix A**.

References

Driessen, V. (2010) *A Successful Git Branching Model.* Available at: http://nvie.com/posts/a-successful-git-branching-model/ (Accessed: Dec. 5th, 2015).

Finkelstein, J. (2006) *Diagram of Maslow's Hierarchy of Needs.* Available at: https://commons.wikimedia.org/wiki/File:Maslow%27_hierarchy_of_needs.svg (Accessed: Dec. 16th, 2015).

Github. (2016) Available at: https://github.com/UNNC-CS-GRP-DTGroup/Notebook-NW.js (Accessed: 27 April 2016)

Pan, Y. (2015) 'Reflection Report On Experience In Software Engineering', AE2SEM: Software Engineering Methodology. The University of Nottingham Ningbo China. Unpublished assignment.

Pan, Y. *et al.* (2015) 'GRP-DT Group Interim Report', AE2DGRP: Software Group Project. The University of Nottingham Ningbo China. Unpublished assignment.

Reid, J.M. Learning Styles in the ESL/EFL Classroom. Boston: Heinle & Heinle Publishers, 1995.

Sommerville, I. (2011) *Software Engineering.* 9th edn. Addison Wesley.

APPENDICES

A. Peer Assessments