Introduction

Agile and test-driven development is a theme that runs through the project construction. Practices based on these principles have been made during the past few weeks, experiences that gained from this process led to the formation of this reflective essay. This paper has been divided into four parts. The first part describes briefly the workflows, experiences and deliverables. The second part further focuses on the benefits and drawbacks of this development mode upon experience, leading to the description of the improvement measures taken and analysis of their effects during the next part. The final section summarises and discusses the limitation and expectations. The results of this reflection show that development upon agile and test-driven model is effective, whose benefits will increase with the accumulation of project management and coordination experience.

Practice Overview

The target deliverable of the project can be summarised as a webpage with a search engine that can give us, based on this data, descriptive statistics about authors, publications and collaborations. In this practice, agile development methods are first applied. This is reflected in the adherence to the requirements of the process and the principles of agile development.

In terms of the agile development process, the roles of the scrum team got assigned, as the five members of the scrum team are divided into a scrum master and a project owner (who will also be responsible for project manager duties). The remaining three are the main developers, but all five members are responsible for part of the programming work, including both testing and coding. The overall implementation is divided into 16 user stories, which are split into four sprints, each lasting one week. Before the start of each sprint cycle, the scrum master organises a meeting to discuss the week's requirements and the story points expected to be needed to complete the project. The project manager will update the schedule on Trello and the developers will pick their tasks. Considering that the project is also based on a test-driven development model, the testers will go through paper tests and write unitest suites in advance. During the development process, each developer works on their branch and the project owner maintains the main branch, resolving conflicts when developers committed their merge requests. Everyone keeps a brief log of the work they have done, including solution logs and time spent, on Trello. The completed and tested code is committed to the GitLab personal branch and, after merge information is carefully written, a merge request is submitted to the master branch. Before the next sprint, the project owner generates a burnup chart based on Trello and git history, a scrum master reflects on the feedback and optimises the design of the routine for the next sprint. In addition, the trunk-based development model is further adopted later in the project. Each sprint has several small periodic merges and iterations during which all development is restricted to a trunk-based branch. This greatly reduces conflicts when merging

code and the difficulty of finding bugs. In practice with the Agile Manifesto and Principles, the requirement for continuous delivery of projects is achieved through trunk-base development. In addition, the business people and developers maintained a close relationship throughout the process, providing feedback to the product manager when the developers were confused about specific details of the requirements, thus managing to maintain a constant pace indefinitely. More importantly, the scrum team always has the end product, not the documentation, as the ultimate goal. Last but not least, the team regularly reflects on how they can improve their results and adjust their actions accordingly.

Benefits Attainment

Agile development has brought many benefits. First and foremost, with scrum teams working together, the speed of development has been greatly increased, so that the goal of continuous delivery weekly has been achieved. In addition, complex projects have been split into less granular user stories and implemented in multiple iterations, resulting in less coupled programs and improved conflict issues with code merging. The close collaboration between the team has also led to a high level of enthusiasm among the members and timely feedback from testers and developers, so that many potential bugs are avoided at this stage, thus reducing the need for subsequent refactoring. At the same time, test-driven development has led to a significant improvement in code coverage and has effectively helped developers understand user requirements and standardise interface design.

As a case study, we compare sprint 1 with a traditional approach and sprint 2 with agile and test-driven approach. Figure 3-1 shows the burnup chart generated based on the entire development process. Sprint 2 is expected to take more time to implement than sprint 1. This is not only because of the complexity to implement, and the iterations and commonality that come with agile and test-driven development do increase the time spent. However, it also illustrated that sprint 2consumed 36% less time in error than sprint 1, meaning that even with the increased complexity, the overall project planning was more accurate and manageable through the use of agile and test-driven principles.

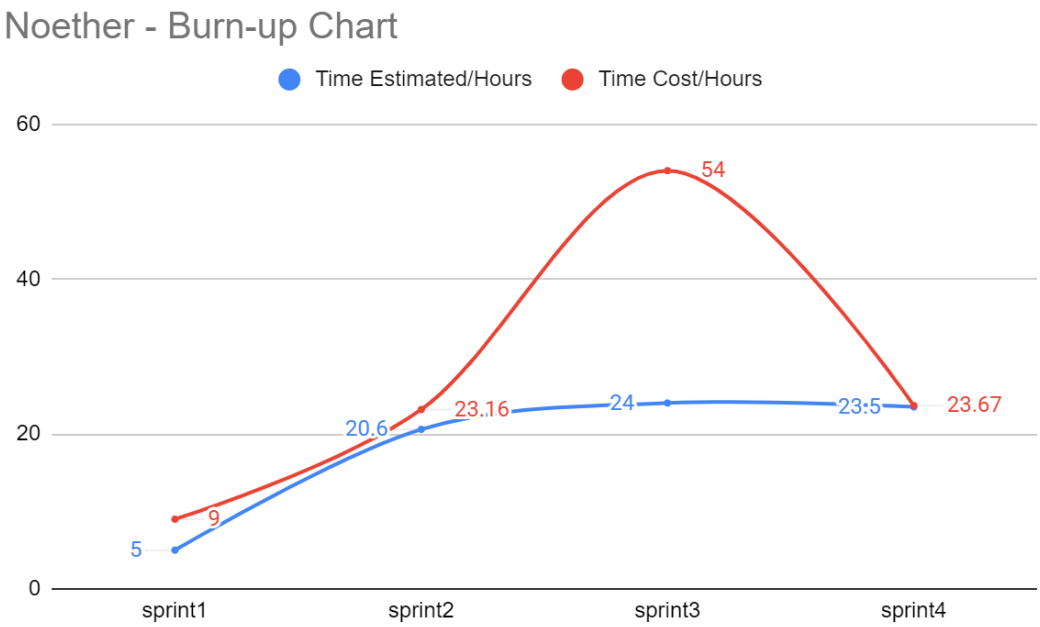

Fig. 3-1 Burnup Chart

Drawbacks and Improvements

In practice, however, in addition to objective drawbacks, such as the increase in time consumption, some unanticipated drawbacks were also revealed. Due to the inexperience and overconfidence of the scrum master and project manager, the story points were not divided correctly. Another exposed issue was the erratic iteration rate. Although the scrum team claimed to use the truck-based development method and had established rules and timetable of submission, this was not well adhered to due to unreasonable time planning and personal reasons, which might be different deadlines for different courses, of the developers. This led not only to a very high number of conflicts to be dealt with when merging code, but worse still to developers probably developing based on different versions of trunk branches. Besides, the team underestimated the importance of face-to-face communication. This was reflected in the fact that even when conditions permitted, communication remained online, via instant messaging software and development documentation. Moreover, the limitations of testers and developers' coding capacity led to the failure of test-driven agile development, as developers could not write code that should match the test suites. For example, for the implementation of the sorting function of a data table, the testers designed and wrote the API based on the back-end implementation as expected, but the developers switched to implementing it on the front-end via javascript for some reason, resulting in the testers having to modify the test code to suit. To add to the problem, the testers also have to try to get a code coverage report of the front-end code and integrate it with the generated back-end code coverage. More importantly, by not following the 'You Ain't Gonna Need It' principle, team members spent some time optimising and beautifying the front-end page layout, resulting in a compressed development time

associated with the project. Finally, as the importance of the regression test was overlooked, resulting in changes that broke the previous implementation not being detected in time and leading to new errors

Sprint 3, shown in Figure 3-1, can be used as the subject of problem analysis. Compared to sprint 2, sprint 3 required twice as many user stories to be implemented, and the dependencies and coupling between the different stories made it much more difficult to integrate them, yet it can be seen that the expected story points for sprint 3 were only increased by four hours, making the final time spent on the project seriously off schedule. This made the final time spent on the project seriously off schedule. Another point worth noting is that a large part of this time is wasting on tweaking and beautifying the front-end page layout.

Based on the reflections of sprint 3, the scrum team has made several improvements. Firstly, the frequency and length of offline communication have been increased. Debugging face-to-face has significantly accelerated the development process. In addition, the timetable for sprint 4 truck-based development got better designed with a full understanding of the developers' individual schedules. What's more, new tests have been added for regression testing. Finally, the development of test suites based on simultaneous front- and back-end testing was very smooth thanks to the previous research and accumulation of experience. As a result, the time estimated for sprint 4 matched perfectly to the actual time spent, as shown in figure 3-1.

Conclusion

Practical experience is gained through iterative implementation and reflection, agile-based and test-driven developments are progressively optimised in terms of implementation details, and the benefits of these advanced thinking are further revealed. Further work could be done to reduce the extra time consumed by iterations to further improve efficiency.