When And Why To Split User Stories On A Kanban Board

COMP150 - Agile Development Practice

1807684

November 14, 2018

1 Introduction

Agile is a development philosophy that attempts to create a easier and stress-free work environment for developers and programmers. This philosophy exists with the ideals that if at any point during the development effort the requirements of the product change it allows the team working on it to adjust to the requirements more easily than before.

2 Sprint Programming

Sprint programming is an important aspect of development with scrum , and almost necessary for the use of kanban boards. It has many positive effects; it breaks a large task—developing an entire game—into shorter and much more manageable chunks, it encourages teams to face problems head on at an early stage allowing them to solve them before they become issues, and it encourages sharing of solutions for problems that may become an issue in the future.

3 User Stories

User stories are a valuable method of managing a team's task list, as not only does it provide an easy way of managing and prioritising a backlog, as well as clearly showing how much work is done and how mush is left to do; but most importantly, it's understandable even to those in the group that aren't programmers. As they're written from the user's perspective and in a user friendly lexicon, it means that those who don't understand technical "jargon" can still add, remove, and change them.

4 Story Sizes

Smaller stories are much more motivating; it's much easier to pick up a quick task, complete it, and move on. As tasks are self-selected in a scrum team, this means better efficiency and higher team moral. Also, as they should be completed quickly, it allows colleagues that can see how long someone has been on a specific story to elect to step in help if they feel someone has been taking too long to complete a particular task[1] .

However, they also create a quite a significant mess on the backlog. Having a multitude of tiny, one-hour tasks that have no obvious way of telling what relates to what can cause an unusably cluttered board, negating any positive effects of using the kanban technique.

5 Splitting Stories

This quandary can be avoided by creating much longer stories, often known as Epics. These are then split at the beginning of a sprint[2], in order to allow different people to complete different sections of a story. This also allows a team to complete a part of an epic in a particular sprint, if it's too long and/or convoluted to do so in a single sprint. However, in order to show that they are linked to each other in some way, the should have some identifying similar feature (such as by being the same colour, or by a letter or number in one corner).

A good way of managing epics is giving each one a set of ”conditions of satisfaction” (CoS). These are are a set of conditions that must be verified for the epic to be marked as complete. These should be extremely granular[3], as they need to be easily testable. An advantage of this is that these make for an easy place to split the epic, as generally each CoS, though linked, will have slightly different requirement to be considered completed.

6 Conclusion

So as you can see, each have merits. It's definitely worth having epics as they show which tasks are connected much more easily, and they keep the board clean and easily accessible; but splitting them down into smaller stories is generally necessary in order for completing them in a sprint to be feasible.

References

[1] N. Oza, F. Fagerholm, and J. Münch, “How does kanban impact communication and collaboration in software engineering teams?,” *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 125–128, 2013.

[2] C. Keith, *Agile Game Development with Scrum.* Addison-Wesley Professional, 1st ed., 2010.

[3] O. Liskin, K. Schneider, F. Fagerholm, and J. Münch, “Understanding the role of requirements artifacts in kanban,” *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 56–63, 2014.