How does the use of different estimation methods influence the perceived ease of the development process?

COMP150 - Agile Development Practice

1702208

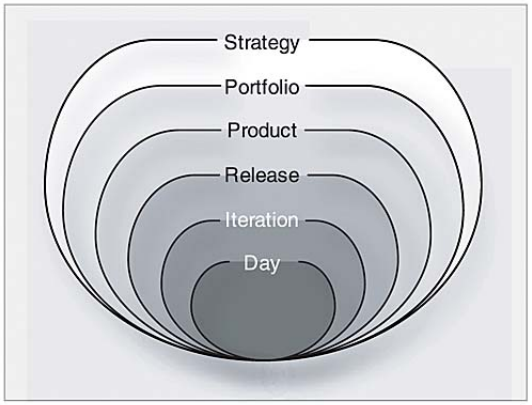
2017-11-08

1 Background

It is important to make the development process pleasant to work on. This way a team will be more engaged in the project, which makes it faster to accomplish. We all know that working without any will never ends well. Especially estimating and planning in the middle of the project rarely goes perfectly. In this case, different estimating techniques are useful. That is why in this paper I will try to talk about the most engaging ways of estimating. However, mentioning different levels of planning is a must.

2 Planning Importance

In this part, I will try to explain why planning is important for every development project. But before that, let's consider two variations, which teams respond to. They either refuse planning or spend too much time on it [1]. In the first case, it's clear that the team won't be able to answer such questions: "When will a feature be done?" and "When do we aim to release the product?" These are some basic questions teams have to have an answer to. Having no solutions to those will eventually lead to an unproductive process. And the second one is issued with overplanning. In this case, overplanning may also cause some major problems. The team bears in mind that their planning cannot be wrong [1]. This eventually leads to some unfortunate understandings after realizing that the plan was incorrect. Therefore, how thoroughly the plan might be it will rarely become more accurate than it should be [1]. There are a few bits of planning, which you can see in this planning onion.



The most important ones for the development team are daily, iteration and release. Daily are just casual stand-up meetings we all know well. But the iteration planning and release planning differ. First of all, iteration planning is

more detailed planning of some features team must do in 1 to 4 weeks. Whereas release planning is considered for 3 to 6 months. This includes what features should be implemented by the given time [2]. Although this type of planning shouldn't be left somewhere on the top of a shelf. It is recommended that release planning is overviewed and changed every time an iteration ends. So, after all this, it's time to make clear why planning is important. Planning is a way to optimally answer this one question: What should we build? [1] Moreover, a good planning should include such options: Reducing risk [3], reducing uncertainty, supporting better decision making, establishing trust, conveying information. To better understand these, see Mike Cohn's book "Agile Estimating and Planning" Chapter 1.

3 Estimating Backlog

So, once we have discussed planning, let's jump into estimating. In an agile development company estimating backlog has a unique place. Talking specifically about agile game development, estimating user stories is nonetheless important. An agile development team has to know how much a story is worth. By saying that, I mean whether a particular feature is difficult to implement or not. For different people, it might take different time. One could implement it in a 2-day length, whereas another can do that in 5 days. That is the main purpose of estimating stories. Although there are two types of estimating points: story points and ideal days [1]. Story points are preferred, but for novice teams, ideal days are easier. Ideal days mean working hours excluding breaks and other difficulties. On the other side, story points are just points, which should be considered as: Is this user story harder (or easier) to implement than the previous one? For example, a story was estimated with 4 story points. And the next user story is considered a little bit easier to implement, that means it's worth 3 story points. Advanced teams prefer this type of estimation because they can easily divide those equally for each day. It is a good way of planning iterations [4]. But

sometimes teams run into some problems, which eventually cause time waste. The first one is that not every member of a team is participating. This only means they are not enjoying the estimation. The other problem is that it takes too much worthwhile time [5]. But, there is a solution - planning poker!

4 Planning Poker

First things first, planning poker is an engaging way of estimating user stories. Some stories don't take much time [2] and discussions make clear why it was estimated exactly this way [1]. Moreover, there are a couple ways of playing this small game. User stories can either be estimated by non-linear numbers or poker chips. Each of these has its own unique level of engagement [6]. First of all, non-linear numbers - also called Fibonacci scale [7]. Those are 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100. The idea for Fibonacci scale is that people are biased taking the average. And the median of this scale is considered 8, whereas a linear scales is 20 [7]. When playing planning poker with this type of estimation, numbers usually represent working hours. It's somewhat helpful, but not as engaging as playing with poker chips. The way poker chips sound when placing onto a table makes people become emotionally engaged [8]. In general, stakeholders can buy a story which they want to be implemented in the next iteration. To understand it in more detail, see [8]. So, these types are definitely somewhat physically and emotionally engaging. Thus, team's mental engagement increases [8]. And how do we use this in agile game development? Simply as it is, after estimating user stories, the product owner can decide what stories are worth implementing first. Rather than discussing each story and getting to no conclusion, this is much easier.

5 Conclusion

So, does different estimating techniques provide a more easy way of development? It surely does; after all, it is hard to work continuously without having any fun. And that is why agile teams usually play planning poker to increase members'

engagement. Furthermore, as this paper discusses, it helps to plan iterations, which are nonetheless important. Let's briefly remember all the important things. First of all, iterations help teams plan what features should be done in a 1 to 4 week time. Iterations show features in more details, which, of course, helps the team to implement it faster. Whereas release planning is more for a business purpose that lasts for 3 to 6 months. This should answer such question as: "When will the product be released?" It can also be changed depending on what features the product owner wants to see. For example, it sometimes worth implementing more features, but a bit later... or the other way. About estimating I can only claim that it helps the team to understand a feature's importance. As it sometimes becomes boring, planning poker is a great solution to make it interesing. It surely increases team's mental engagement into the project.

References

[1] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[2] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, 2002.

[3] S. Grapenthin, M. Book, T. Richter, and V. Gruhn, "Supporting feature estimation with risk and effort annotations," in *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*, pp. 17–24, IEEE, 2016.

[4] N. C. Haugen, "An empirical study of using planning poker for user story estimation," in *Agile Conference, 2006*, pp. 9–pp, IEEE, 2006.

[5] J. J. Cunningham, "Costs of compliance: agile in an inelastic organization," in *Agile Conference, 2005. Proceedings*, pp. 202–211, IEEE, 2005.

[6] K. Molokken-Ostvold and N. C. Haugen, "Combining estimates with planning

poker–an empirical study," in *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, pp. 349–358, IEEE, 2007.

[7] R. Tamrakar and M. Jørgensen, "Does the use of fibonacci numbers in planning poker affect effort estimates?," 2012.

[8] J. Yip, "Hands-on release planning with poker chips," in *Proceedings of the 14th Conference on Pattern Languages of Programs*, p. 11, ACM, 2007.