Agile Discovery - Aiden Burgess

Pros and cons of scrum, kanban, XP in a uni environment

Agile manifesto

> - ***Individuals and interactions*** *over* ***processes and tools***
> - ***Working software*** *over* ***comprehensive documentation***
> - ***Customer collaboration*** *over* ***contract negotiation***
> - ***Responding to change*** *over* ***following a plan***

XP Practices

> - *The Planning Game*
> - *Small Releases*
> - *Metaphor*
> - *Simple Design*
> - ***Testing***
> - *Refactoring*
> - ***Pair Programming***
> - ***Collective Ownership***
> - ***Continuous Integration***
> - *40-hour week*
> - ***On-site Customer***
> - *Coding Standard*

Relate my experiences with Agile and Scrum. My takeaways from agile and scrum. Talk a little bit about Kanban and how it is incorporated. Have a look at XP

Gentrack Internship

Gentrack was my first experience with an Agile environment. Although it was a bit limited as I was in a team of three interns who were working on a separate project. We created our own tasks as we discovered them and used a Kanban style of working, with assigned tasks and a deadline set for a set of tasks set by our manager.

SOFTENG 306 Project II (Refactoring)

For the course SOFTENG 306, I lead a team of seven in a refactoring project. This was my first experience leading a software project of such a large team. In my personal work I prefer having a backlog of items, which can be pulled from as tasks I am currently working on are completed. This clearly resembles a Kanban style of work, and our team used Trello to track the backlog and tasks. Initially, tasks were not assigned (self-assignment) and there weren't clear deadlines for tasks either.

After a few weeks, we found that the team completed less work than expected, so were behind in terms of progress, and there was an unequal workload amongst team members. Some people like to procrastinate so this style of Agile did not work well for our team. Also there was not much accountability as we held meetings infrequently (no daily stand-ups), so people didn't realise that others were getting work done.

Daily standups, which are a part of both Scrum and Kanban, are not feasible in a university environment. Student's have different schedules, and are not working full time on the project assigned to the team. Instead, this practice can be adapted to be held a few times a week, to allow for semi-regular updates.

As the project deadlines approached I recognised these issues and changed added clear deadlines and assigned people with specific tasks. We grinded and got the work done, although there were some team conflicts.

^ Learned adaptability (scrum) and a agile value. If we had followed the sprint retrospective maybe this change would have been implemented sooner. I think that sprint retrospectives are much more important earlier in a project to set good practices and reflect early on productivity. I should have fostered an environment to support my teammates and helped motivate them instead of taking such a hands-off approach. The development at the end of the project was not sustainable, so did not follow Agile principles.

Amazon Internship

Amazon internship experience in an Agile team. Initially following pretty strict scrum. Daily standups combined with ops talk was 1h+ each day. We had a retrospective, where we identified major issues. Estimates were not working out. The amount of time spent in meetings was too excessive. We debated between Kanban and a Kanban + scrum mix.

Amazon experience also integrated many XP practices, such as pair programming/debugging with my mentor, testing and continuous integration, collective ownership, and an "on-site customer". Almost every line of code I wrote had to be tested, there were unit tests, integration tests, end to end tests, "canaries", alarms, and monitors. The team also actively practiced collective ownership. As all the code had to be reviewed before being merged, no one person was responsible for any issues, the entire team took responsibility. We also received most of our tasks and issues directly from customers, they would send tickets with issues or potential features, and we would respond to each person individually and in a timely fashion. These customer tickets as well as some operational and team suggested tickets made up our product backlog (which was huge lol).

There was an internal anonymous survey to measure how the team was feeling about our work and projects. My manager noticed that satisfaction had been decreasing slightly over a month or so, and scheduled a meeting for the team to air any concerns. Here everyone spoke openly about issues such as excessive amounts of time in meetings, stress, and operational issues.

^ Even later on

Collective code ownership (XP) in industry allows each team member to make changes to any part of a code base. This may not work as well in a university context, as there are strong time constraints. Under these constraints the time needed to switch to a different module - reading documentation and understanding code - may be inefficient. I prefer the system of weak code ownership where modules are assigned to owners, but team members can modify other modules if they feel the need to do so. This means that there is always at least one person who has deep knowledge of some part of the codebase. (Can put experience here in softeng 306 part 1 where I switched from backend to frontend)