Practical experience: 2 yrs and ongoing, 1 project, release 2-3 month.

All meetings: planning (6h), daily (15min), retrospective (1h45), review (3 review because of 3 projects at the same time)

Team: 9 pers, One team working on several project at the same time.

The guy was Scrum master and “proxy” product owner (team leader was doing priorisation of the backlog)

Team was doing estimates, using story points.

Two violation: no cross functional team (team of only developer), no separation of scrum master and product owner.

Specification and testing were done before, resp. after in a waterfall mode. Test environment, and unit test were done during development phase. But then was another test department doing more testing.

Mascha asked what has been refined in the process. They didn’t had product backlog for all project (separate per project), no estimation with story point (but in working hour).

Masch ask what are the weak points: they’ve separated the product owner and scrum master, so the only one missing is the cross functional team. Which won’t change given that it goes against the organization (functional dept, and test dept). This however imply communication overhead. Team is growing, so they will need to split into a scrum of scrum.

Splitting role is better: it’s too much work for one, and leads to compromises, risk that quality (scrum master) suffers because of ROI (product owner).

What about tool support: board, + agilo plug-in for trac. Both were configured to their need. Product backlog with user story, split user story into task, and assign them to sprint. Sprint burndown for the team. Trac only accessible to developer. No detailed report to stakeholder. Progress was shown during review. Stakeholder were not interested in detailed report.

Negative point about the plugin: you can configure heavily, but usability is not optimal.

How far can you configure the tool? Quite heavily: type of issue, type of properties, story point or hours, etc. You can link issue together. \*\*\*\* You can assign taks to sprint, story to sprint, etc. and do the priorisation. Rather flexible, no restriction like other tools. \*\*\*\*\*

Roles are reflected in the tool, e.g. who can do that. Usually it makes no sense to restrict the view, it goes against the transparency. \*\*\*\*

Official bug tracking (test dep) done with another tool. Internal testing (dev) done with trac, and bug where entered as user stories. Dedicated timeframe for bugfixing (it was a special story with fixed time). They had user story, task, and special user story (not estimated, they had fixed amount of working hour).

Only user story were tracked on the board. Visibility is much better on the board.

Team was responsible for handling the board and the backlog and keep them in sync. Redundancy was well accepted, the board was a request from the team for visibility \*\*\*\*\*

Document (specification) were in Word on a shared file system. Attachment to user stories. Changes in doc were reported in the tool.

Mascha ask for update of doc. Answer: waterfall, spec were almost frozen. No bi-directinal link. Track change only. \*\*\*\*\*\*

Mascha ask about feature in trac and agilo (see picture).

Mascha ask about knowledge transfer. Answer: one developer as point of contact, pair programming.

Was it a problem that the team work on 3 projects at the same time? No, team would have been too small, plus they were all working on the same application, but yes, that complicates a bit. E.g. product owner was the same for 3 project, so they needed one person of contact per project so that he could get the insight.

Who selected the tools? Answer: I did. Why? Because he knew it beforehand. Trac is great because it’s integrated in the dev. Environment. Other tools were too restricted; you can not configure them, e.g. different project in the product backlog. Other thing, e.g. two point estimates \*\*\*\*\*\*

Mascha ask about reporting? Answer: there was none. Only a release burndown for internal tracking, done in Excel. \*\*\*\*\*

Every developer tracked its work done, need to report on how much was done per project (no usual with Scrum). Also one of the part missing in Agile fan (????).

Mascha ask about visibility. Not much to say.

Mascha ask about product vs. project level. Product was the web application, project was one feature or additional thing added to the app. Team of internal and external. Document was as less as possible, wiki for most important information. Rest was collective knowledge by the team (e.g. pair programming).

Other artifact, e.g. release notes, small UML, etc. in the wiki, It was too costly to keep information in the wiki as the application was changing all the time. Shared knowledge in the team. \*\*\*\*

Masch ask how Scrum was introduced. Answer: in smaller step, with the normal feedback process to improve. Retrospective was useful in this sense. One problem was separate backlog. Big change for developer to get used to the responsibility \*\*\*\*\*

It’s better to keep the tools small and simple. First the process, then the tool. He wants to be able to configure the tool. Tool that are easier to use are restricted to one way of doing it.

Estimate & work in progress. To work on one task, we don’t need tool support. If we need one, it means you hide the problem. For estimation, the best is card.

Non-functional requirements. Tasks are step to achieve a user story. Focus on visible functionality. Other NF are part of acceptance criteria for the story. That’s part of the definition of done. Refactoring or redesign is just normal user story. They are prioritized the same way.

Test and test data are part of the source code. Do we need test report? Continuous integration should be summarized as a flag red/green. Test coverage, there is no practical use for it. You measure as less as possible, measurement leads to the wrong action. Team is responsible for quality. \*\*\*\*\*

Transparency is the heart of Scrum.