Project manager @Zuehlke

.NET dev

TFS, application lifecycle tool

10 years experience

2 years ago, Scrum master certification

1 scrum project at the moment, 3 in the past two years

Project for Bundesamt fuer statistik (3dev, SM, PM=PO), 8 month

Sprint: 30 days (20 working days)

Daily scrum meeting, physically dev co-located, PO in another office but same building

Dail struc was timeboxed. Yes, except when PO was talking

It was difficult to separate the discussion between product backlog mgmt and ongoing progress – PO was frequently interrupting the dev report, and meeting went longer… \*\*\*\* split the meeting in two: dev / req mgmt \*\*\*

Sometimes customer in sprint planning

Sprint planning was time boxed. Priorization, work breakdown, estimation was done beforehand. PO w/ customer decides what goes in the sprint. Team could not change priority of item in backlog as set by customer.

Kind of product backlog items: user stories, bugs. Attributes of items were the same. Handled the same way. NF req. were not in backlog. A separate sheet for that. User stories were referring to NF req. Not strictly part of the definiton of “done”.

Product backlog was filled up to 80% of the capacity. 20% remaining were use for emergency bug, infrastructure work. Unplanned bug were in the sprint, not infrastructure work. Learning point: all work should be represented in sprint backlog, inc. infrastructure work. He would do that differently.

Review was 2h long. Retrospective was 1h with only team and scrum master, PO did not attend. Initially not time-boxed, then they changed.

What aspects of the process did you refine? No CI initially. Improve testing infrastructure to simulate client’s side. Issue with PO who couldn’t join every meeting – he wanted a feedback each two week. He had to learn how to participate more intensively in the project.

No training. No pair programming. Code review though. Other point of knowledge sharing? There was a wiki, very informal. Scrum master would request team member to document things, they would not do that spontaneously. Knowledge was in the head of the people.

Would the document be sufficient to continue the project, say, in 2 years? There would be lack of knowledge. But there is some knowledge loss. Even after retrospective, the documentation was always an issue.

Tools: wiki, TFS 2008, VS.NET, Toad.

Detailed requirement, statechart, seq. diagram were done with MS Office.

TFS for version control.

Scrum wall? At the beginning no Scrum wall. After 2nd Sprint it was introduced. It was a mirror of the TFS work item. Burndown chart generated by TFS. During meeting they were using the wall. Sync was done by developers – 15min per day. Scrum master would control that wall and TFS was consistent – again 15min.

It was not practical to talk around a tool or a beam in the meeting – wall was better. “For me TFS is very good tool in an environment where tractability is very high, compliance, medical, etc. or if a management dashboard is needed. Useful for the stakeholder, less for developer” \*\*\*\* interesting point \*\*\*

Missing the facility to add note or informal information in TFS. Very easy on the wall. Could be added in TFS as a text field, but would not be the same, e.g. red posit, highlighting. Informal information. Form vs. content.

TFS, conchango templates. Project portal was a web front-end. They used the standard TFS dashboard, no specialed front-end.

Custom reporting via excel integration; we consumed data from TFS. Reporting was remaining work estimation, a pie chart with remaining planned work.

Burndown chart at the beginning was useless because req were poor.

Definition of done and undone? 🡪 only of done. Definition of done was written on the wall. In TFS criterion for functional req. This was not an issue.

To substitute the scrum wall, we would need a very interactive GUI. \*\*\*\*\* that’s the main point \*\*\* Non-verbal communication.

Weak points:

Scrum knowledge was poor first 🡪 more training for another project

Customer was bypassing the PO 🡪 the PO was not present enough. He didn’t looked in TFS, and did not communicate in the right way.

Good points:

The ability to show and demo the product after every sprint. The feedback from customer (about the product) and team (about the process) \*\*\*\*\* meta \*\*\*\*.

Minor adaptation in TFS, e.g. category.

Granularity of user stories. To big at the beginning, needed to learn how to chunk them. Customer had to learn to break down also his needs. User story vs. tasks. \*\*\*\* could we do something about that? E.g recommendation system? \*\*\*\*\* Estimation were done in hour – the team was not use to story points, for another project.

Automation. Load tested could be automated, no performance tests.

As a scrum master or PO, a tool like TSF is very valuable. It stores historical data, no need to take picture of walls, etc. \*\*\*\*\* picture of wall & history \*\*\*\*. You don’t need meeting minute, you can pull data out of TFS and generate a report that acts as meeting minute and that would be stored in a folder. It was needed because the company was CMMI, and they needed to store the data into a project folder. \*\*\*\* adaptation \*\*\*\*

Linking from TFS was used very much, soft link like attribute of User Story (e.g. external resource). Naming scheme for issue, etc. to make weak links easier.

Visibility. Mostly used by developer, and PO. Dashboard for management was never used. Management was maybe not used to the burndown chart. Training might help in this area.

Testing is central to scrum, because by definition scrum mandate that the product is shippable at the end of the iteration. That’s part of the definition of done.