Experience:

* 6 month
* experience: 0 project
* building the course
* scrum gathering
  + people come with problem
  + people discuss how to solve these problem
  + scrum picture
    - what kind of material
    - big note, post-it, etc.
* course is kind of a mini project
* couse: TFS and Visual Studio

Mascha draws the Scrum process on board.

Ask if there were other meetings. Answer: no process meeting. Knowledge transfer, pair programming. 1 sprint per day.

Scrum “Not everybody know everything, but 2-3 people know something together”. Collective knowledge.

The course trainer has the role of product owner, possible scrum master, that depends on whether somebody in the team knows already about scrum.

Review and retrospective were also “simulated”. First day: no TFS, workspace on the wall with paper. Then we improve the process, and on day 3 they switch to TFS, both way, TFS + board. At the end, they can decide what the like best. \*\*\*

Improvement from 1st day to last day? What were the diff? Answer: people start to think about process. Engineering where just hacking feature first, no thinkging about the process. Then they align, do more testing, synchronize better, teach each other more, they get the feeling of a team. \*\*\*\*\* (What is a process?)

That’s challenging. You see all cards, and want to do all to be the best team. Brute force approach. But then they realize it better so organize things. Then you introduce product steering, e.g. notion of release, build, etc. Scrum let you do mistake and then learn from it \*\*\*\*\*

Tools: board, and TFS. What did they prefer? Answer: most people see the point to track the work on TFS, but the interaction for the meeting is a problem. If you work with TFS, someone needs to enter the data, so one person need to have this precise role. Not necessary with the board. No necessary one single person 🡪 more interactive. Then during the project, responsibility to update is shared among everyone. You need discipline but then it works. (Does it work for real longer project) \*\*\*\* Using TFS is cool for remote workers, e.g. somebody sick at home.

Added value of such manually entered information: generate release not automatically. It’s backed up, not like the board.

Time tracking and estimation. The last group has problem to get away estimating in hour. And not follow story point (Scrum prohibit estimation in hour). But team developed the process: they decide to track in hour. \*\*\*\*

Mascha ask about visibility of progress with TFS, e.g. business view vs. dev. View. Answer: you can track code, work item aligned with the code, why a line of code has been inserted, etc. You can backtrack what the reason was. Product owner can work with Excel or Sharepoint, this is a big advantage. You can query user storie and taks, and edit in Excel in Sharepoint. You create, edit new items. Interesting for product owner, stakeholder, which are used to have internet + excel.

Disadvantage: the workspace is just data, list. It’s basically a database. There are add-on which make TFS more appealing, like card, etc. but it’s not been used. \*\*\*\*\*

Configuration of TFS: standard installation with agile process template, not yet the scrum template. This was a problem because the work item where not exactly matching.

Are the different roles reflected in TFS? Are there different views, or the same for all member? Product owner can get full access or limited access. It’s done through the queries. You can customize the visibility that way. \*\*\*\*

What about version control? All in TFS.

What are the weak points of the process? No many, the mini project is close the ideal. Time box model focus on short meeting is good, you see if project progress. The problem is more the output of the meeting. What’s the expected value of the meeting.

What are the good points of the process? Team is responsible of delivery. Team gets responsible and do it his own way. Another advantage: clear focus on feature “getting something to work”, not architecture or theoretic things. Information to stakeholder flows easily. (Why) \*\*\*\*\* Transparent process, you can see everything. No cheating. Same with scrum board. Immediate visibility. In usual project you have internal and external view. Here there is no point in doing it. Only one view.

Who defines the “definition of done”? Set by team and product owner together.

What were the meeting for knowledge transfer? Dev would meet after the scrum meeting, and people did exchange information, teach each other. Teaching is good to strengthen his own knowledge. This was not originally part of the course, it just happened spontaneously. Give the ability that everyone can be a teach at one point in the project. \*\*\*\*\*

Meetings with 10 people is no longer smooth, not longer productive. Too many opinions, start discussions with each other. Not happening with 5-6 people. Gap is around 9-10, as promoted by Scrum. (So how to do with more people?)

What was the difference between product level and project level? –

The room is important for the group feeling. E.g. if people are physically split in two. Better to have a team room where everbody fits in.

Linking of information & facilities in TFS? \*\*\*\*

* code
* user story
* test case
* bug
* change request ?

Weak links is not perfect. Work item in hierarchical position (e.g. tree) 🡪 no go for requirement engineering. We want an easier interface for linking. Text for user story is also too limited for req. engineering.

Report and query are visible on VS, and Excel. Default report in project template. Excel workbook.

Other kind of report, e.g. code duplication, etc. Reporting on the engineering side as well. Reporting framework.

Template need to be kept during the project. Impossible to update. Etc.

Document in sharepoint, or repository database

Doc are versioned, but not user story. That’s ok, because it should be handled with a change request work item. For complete req. engineering that would make sense. Doubtful.

The process is meta-process, you must do meta-thinking. To reach you goal, you have to work on your process.

TFS does limit a bit this vision. You don’t want to do too much on TFS and change your work item all the time. You don’t have good editor, it’s most “do it once”. Easy to make minor changes though.

Usuability and intuitivity of TFS is a major. Will we have TFS process engineer in the future?

Scrum board is not real multi-tasking with several people. There is a gap. You watch somebody write a card on the scrum board, while in the real work, you write your card on your desk, and can watch the wall on the meantime.