
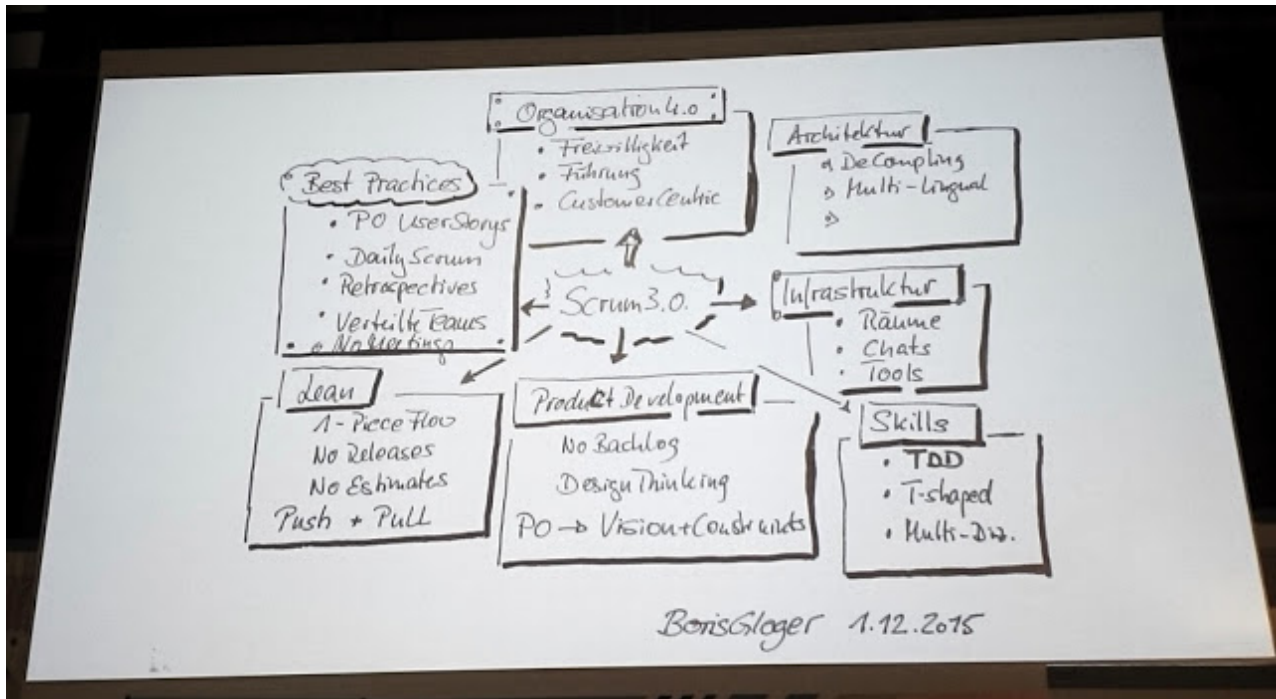


Scrum 3.0 and Organization 4.0 – impressions from a great evening with Boris Gloger at ImmobilienScout24

 ontheagilepath.net/2015/12/scrum-3-0-and-organization-4-0-impressions-from-a-great-evening-with-boris-gloger-at-immobilienscout24.html

By Sebastian Radics



Today I had the opportunity to join a great and inspiring presentation by Boris Gloger talking about Scrum 3.0 and organization 4.0 (thanks to ImmobilienScout24 for hosting a great event).

With this post I provide a short summary of my notes and insights and links to further posts I already wrote about some topics presented today.

Based on an initial blog post by Boris (DE), – we started today with a recap of the Scrum journey from Scrum 1.0, Scrum 2.0 and developed to today's Scrum status.

Scrum 1.0

- foundation by e.g. Agile Software Development with Scrum (Ken Schwaber)
- basic meeting artifacts, 3 roles (ScrumMaster as management role, Product Owner and team)
- retrospective was not yet part of it
- Backlog idea, but not yet that established
- focus on delivery
- sprint idea – a common way to think about what we would like to deliver together, but breaks in between sprints
- long Excel-lists with tasks and detailed task estimations

What did we learn?

- breaks between sprints don't make sense
- role of PO was still a business analyst role
- why 30 days and what does it mean – is it calendar days, what about Christmas
- sprint planning and commitments did not work

Scrum 2.0

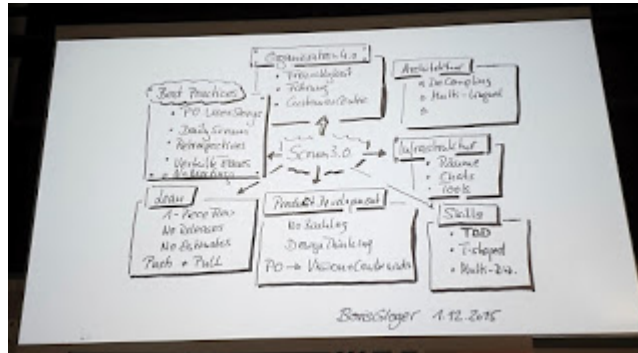
- roughly since 2004 – driving question, how could it really work?
- breakthrough for retrospectives on the Scrum Gathering in Vienna ... shortly thereafter commonly used practice
- more advanced ideas about the sprint planning
 - PO has to prepare the backlog and user stories
 - PO has to know what she wants
 - PO became the single wring able neck
- Sprint review pattern ... PO decides if the delivered is right or wrong
 - created a difficult situation for the PO
 - did the team fail when something did not get delivered (based on Waterfall-like thinking ... for sure the team failed))
 - followed by the PO shouting on the team

What did we learn?

- PO mega busy
- we created a really stressful environment
- things were not really clear
- but many best practices arose
 - requirements were articulated using user stories
 - dailies – post it moving sessions
 - one can build a huge amount of trash following best practices
- Scrum and the process ... Scrum as the Silver Bullet
- great selling argumentation for Scrum
- it worked somehow on methodical level but did not address several problems e.g.
- scaling
 - approach to use Scrum of Scrum
 - collocated teams e.g in 2010/11 – huge teams 18, 18 coaches, 18 POs ... highly stressful, not that much fun ... and the organization killed the initiative shortly after the project was delivered
 - heavy meeting load for the PO – Daily, SOS, PO-Daily, Review ... does not scale
- architecture ... topic commonly shared infrastructure – addressed via communities of practice
 - team delegation and architects, but slow and often no decisions leading to the best people leaving the community

- today called guilds
- **process, process, process**

Scrum 3.0



- ideas collected from the last 2-3 years
- all methods elaborated
- new best practices

Product Owner

- it is not her duty to write stories, its the team's responsibility
 - team has close contact to the customer ... and **developers write stories**
 - PO is responsible for creating and transporting the **product vision** ... the **WHY becomes the central question** to answer
 - team – includes everyone necessary to really build the product/system

Dailies

- major goal: **progress**
- everyone shows their progress on the product instead of moving tickets around
- Mob programming – all work TOGETHER and show themselves the results (pairing next level)
- no more PO dailies
- distributed teams – reduce amount of necessary communication by through an intelligent architecture with clean interfaces and restructure your organization accordingly
- company example – one product one team, teams build that product like they think and its ok if there are differences among products (you can drive some level of standardization using guilds if necessary)

Recommended reading: Scrumban [R]Evolution of Scrum

NoMeetings

- reviews and dailies are removed or completely changed
- **cancel all regular planned meetings**
- establish communication on different channels e.g. chat
- ad hoc session to discuss next steps on your product development
- optional meeting attendance
 - if someone does not attend, its his duty to get up to date afterwards. Its a shift of responsibility back to the individual
- pair programming – (Manlow Innovation) – that really established pair programming in a tough manner

One piece flow

- people just work on one story at a time – all together (e.g. using Mob programming)
- differences really get transparent

Recommended reading: [#NoSprints](#)

No estimates

- who still needs story point estimations?
- its enough to count things that get delivered in a given amount of time
- story points were an interesting idea back in 2003, aiming to remove estimation in hours
- using Kanban one tries to optimize flow and throughput
 - reduce backlog size
 - PO has to learn to say NO
 - best backlog size is 1
 - communicate and establish that we do one thing at a time and not more ...
FOCUS

Recommended reading: [#NoEstimates](#)

No releases

- get it live immediately and receive real customer feedback (not management and PO can decide what works for the customer, it the customer who decides)
 - user stories are no laws but a way to foster communication
 - working with releases created delays – lets work on removing these delays
 - embed deployment in the team – DevOps – the team builds it, the team is shipping it
-

Product development

- not longer with backlogs but using design thinking approaches, hypotheses and data

- driven by thinking ... how do I get to the needed/right functionality
- design thinking ... I don't really know what to build
 - based on assumption, mini prototypes and/or fast and cheap development
 - to learn whether we're moving in the right direction
 - learn to think what the user is thinking
 - important early link with the real user
- cost estimation?
 - use probability approaches and forecasts based on delivery time and needed scope
 - ROI and budget response to the teams – and POs have to take this new responsibility
 - measure ROI increase

Recommended reading: [Awesome product management](#) and [Discover ideation](#)

Conclusion

The implementation of Scrum 3.0 is not easy. Teams don't like to question their way of working. Continuous delivery needs new skills and knowledge gaps become more transparent.

Implement voluntariness!

Check your level of agility by watching:

- politics in your company – how many discussions are inward focussed (between departments and hierarchies)
- it's not about self organization – it's an instrument – but the real goal is that people behave in a way that it is useful for the product to be developed. And therefore its of high importance that it is voluntary.
- the main task for a ScrumMaster – how can I help and guide others to contribute and have fun working on it.
- focus not solely on the process but on the purpose of doing something

Maybe I missed some important points? Please share your thoughts and insights with your highly welcome comment 😊