



Ten Ways to Fail in Agile Adoption

Scan-Agile 2009

Kristian Rautiainen
TKK/SoberIT

kristian.rautiainen@tkk.fi



Ceremony Without Understanding

- ❑ Doing everything as before, but calling it Agile
 - ❖ New terminology, but same routines
 - ❖ E.g. Weekly meeting => weekly Scrum...
 - ...or Daily Scrum that lasts as long as a weekly meeting

- ❑ Fallacy of allowing feature creep during iterations
 - ❖ "because we're supposed to be agile"



Underestimating Transition Effort and Time

- ❑ Organisational change or process improvement is always challenging and takes a lot of time
 - ❖ "This agile thing doesn't work for us. We tried it for a couple of months, but..."
 - ❖ Fallacy of not risking some time now in order to win a lot of time in the future
 - Transition to Agile slows down productive work for an unknown period of time
 - Business representatives tend to be impatient...



Throwing Away Existing Good Practices

- ❑ "Let's start from scratch with this Agile thing"
- ❑ "We used to observe users to elicit requirements. Now we invite them to iteration demos instead"
- ❑ "Something that works for us is not considered Agile. We have to throw it away."



Cultural Conflicts

- ❑ Project manager => Scrum Master
 - ❖ You often see this role transition in companies
 - But the differences in the roles cause trouble and confusion
 - (technical) project manager => (technical) product owner is actually an easier transformation

- ❑ Breaking down "silos"
 - ❖ Creating truly cross-functional teams is a big challenge...
 - ...but without them lots of effort for coordination is needed



Challenging Roles

- ❑ Product Owner
 - ❖ Need I say more...?

- ❑ Empowered, self-organised team
 - ❖ "Say what?"
 - ❖ A transition from a "clear roles and responsibilities + a project manager to tell us what to do" to self-organised is rocket science and a huge step for most people

- ❑ A lot of coaching and support is needed for the role transitions



Lacking Competence

- ❑ Getting rid of the difficult practices because they take too much time to learn
 - ❖ "TDD was really hard to understand and learn, so we stopped doing it"
 - ❖ You need to understand the trade-offs and underlying assumptions, otherwise you end up in total ad-hocracy

- ❑ Working at high velocity requires high discipline and skills



Minimal Plans = Minimal Planning

- ❑ Fallacy of planning only in Release or Iteration planning sessions
 - ❖ "I just saw the demo. Now I need time to figure out what you should do next"
 - ❖ Planning should be made "continuously" in a rolling wave fashion
 - Requirements refinement
 - 5% workshops have not been around in most companies (not for long, anyhow)



Just enough architecture = no upfront architectural work

- ❑ Working in short iterations may compromise the architecture
 - ❖ "The best architectures emerge"
 - Sure thing, as long as you have the best people
 - ❖ "Just enough arcitecture"
 - How do you know how much is enough?
 - ❖ "We have these 10 teams working on the release of this product"
 - "Architectural runway" is a must

- ❑ Architectural work is challenging no matter what you call the development process!



Anyone Seen the Onsite Expert User?

- ❑ Working with a simple backlog may cause losing sight of the "Big Picture"
 - ❖ Most backlogs only contain vague one-liners, not even proper user stories
 - ❖ If the onsite expert is not there to explain things, the end result might be completely off
 - The positive thing is that we know it pretty fast, but it is still waste



Agile Only Concerns R&D

- ❑ After all, the Product Owner is part of the team and is responsible for all "that other stuff"
 - ❖ Still, agile (and lean especially) does change the way *everybody* should be thinking about developing software

- ❑ Customers still want detailed plans and contracts!
 - ❖ Business/sales representatives force the teams to produce detailed plans upfront
 - ❖ *Who trains the customers?*



Thank You! Any Comments/Questions?



Contact info:
kristian.rautiainen@tkk.fi