Craftsmanship

•••

The Codeborne way

Anton Keks @antonkeks

codeborne

well-crafted software

It all started in Hansapank

8 March 2010

9 people left Swedbank to start Codeborne

1 joined from HireRight

Codeborne

Locally owned old-fashioned business (not a startup)

3x Äripäev top 3 IT company in Estonia

Some present and past customers in Estonia:

Eesti Energia, Elering, Swedbank, LHV, Telia, Starman, Ericsson, Regio, ABB, Luminor, Big Bank, Coop Bank, SEB, Fontes, Astri, Qvalitas, Starship, Taxify

Plus others from Russia, Norway, Sweden, Japan, US, Czech

Sponsor of Devclub, Agile Saturday, Tartu Maraton

Codeborne in 2018

33 people

2 CEO & sales
1 office manager
2 designers/front-end
22 software craftsmen
6 software craftswomen
0 project managers
0 analysts
0 testers

Agile manifesto

Individuals and interactions *over* processes and tools

Working software *over* comprehensive documentation

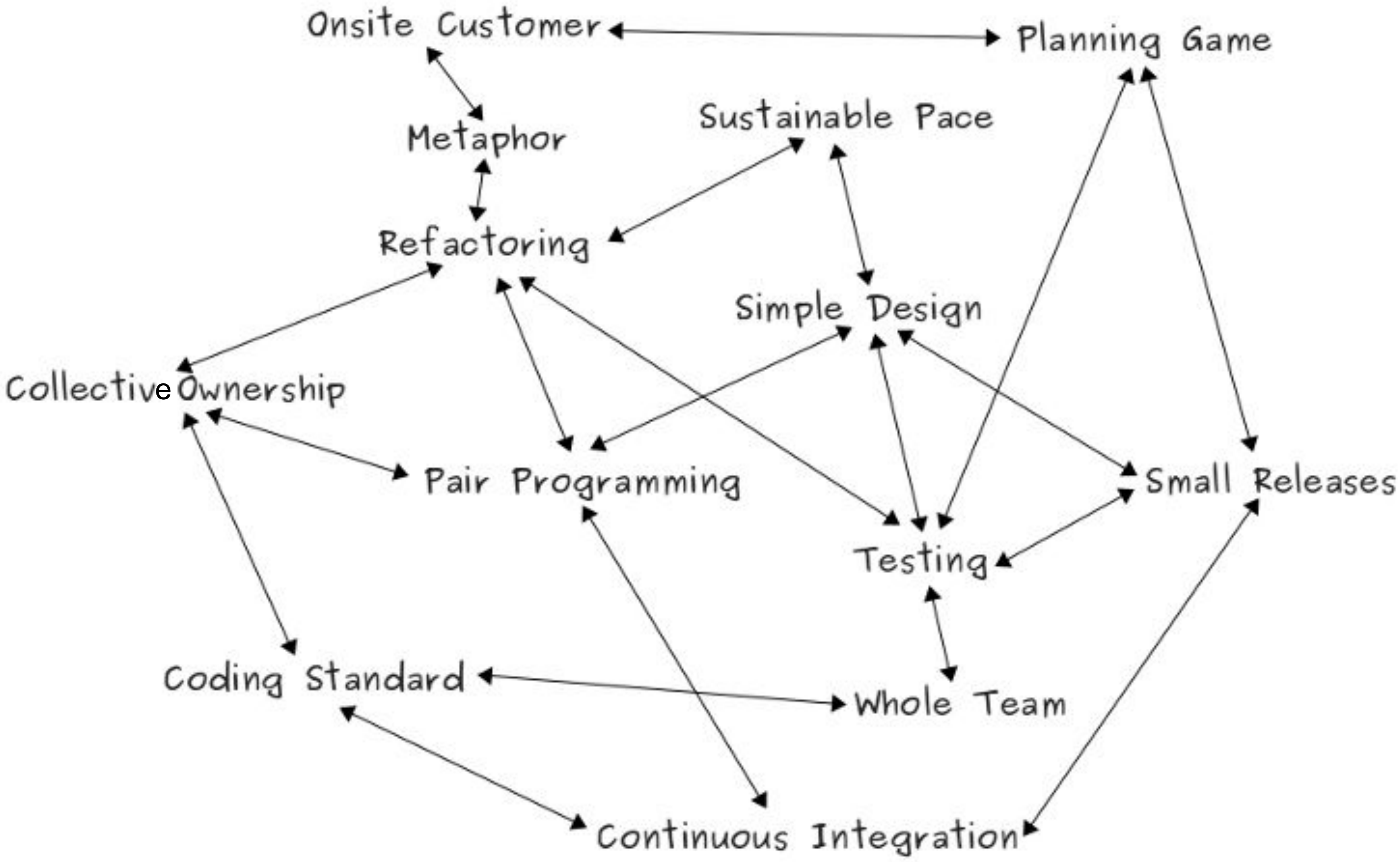
Customer collaboration *over* contract negotiation

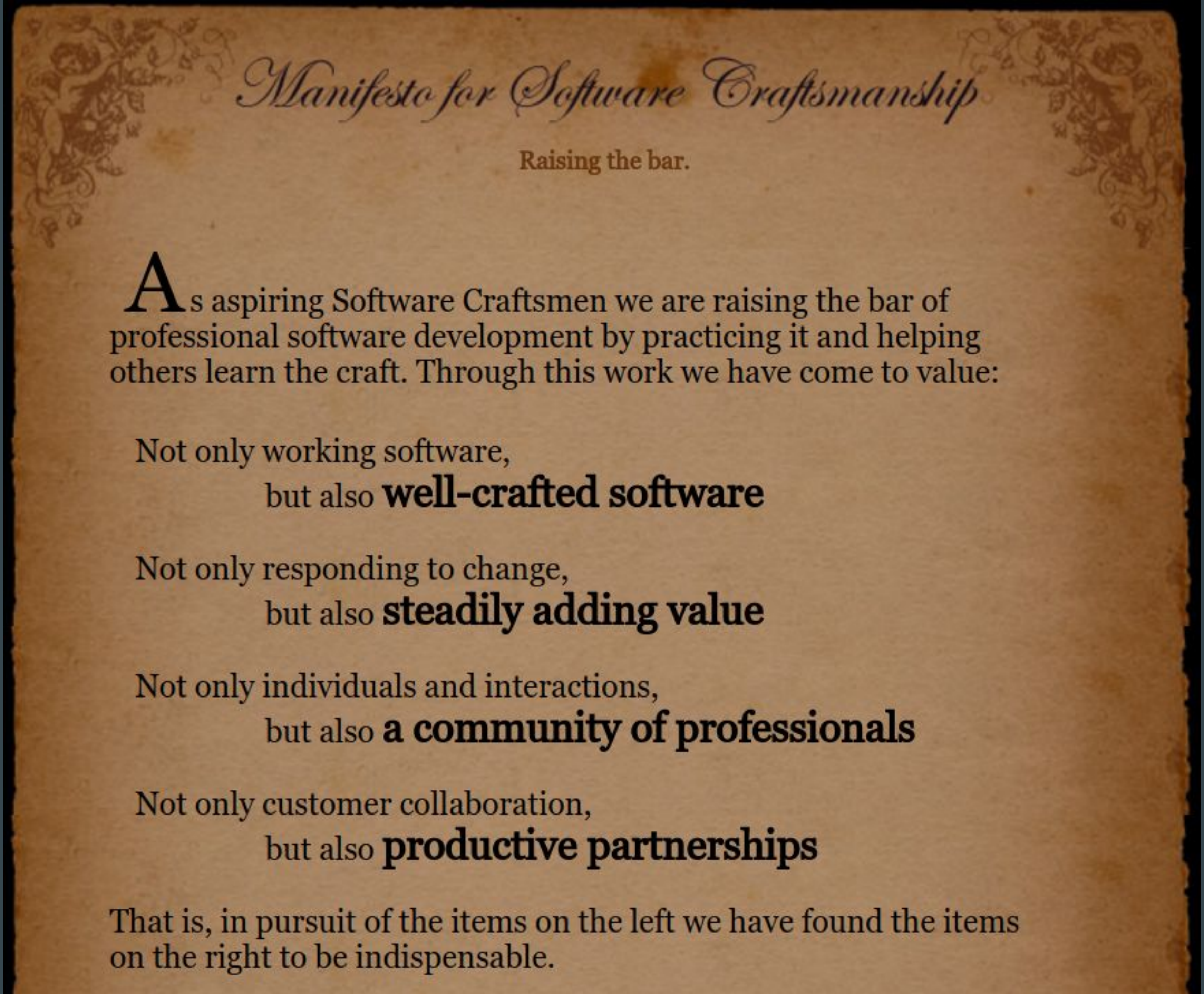
Responding to change *over* following a plan

That is, while there is value in the items on the right,
we value the items on the left more.
http://www.agilemanifesto.org/

XP

Extreme Programming


Onsite Customer
Planning Game
Metaphor
Sustainable Pace
Refactoring
Simple Design
Collective Ownership
Pair Programming
Small Releases
Testing
Coding Standard
Whole Team
Continuous Integration



Manifesto for Software Craftsmanship

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

Software craftsman should be able to

- Talk to customer directly
- Understand the underlying problem, not how customer proposes to solve it
- Propose solutions
- Break the problem into small chunks, write down as user-centric stories
- Design UI flow
- Write working code ← Old-fashioned software developer
- Write automated tests to avoid regressions
- Deploy the system to the end users (“DevOps”)
- Evolve the system design/architecture by refactoring

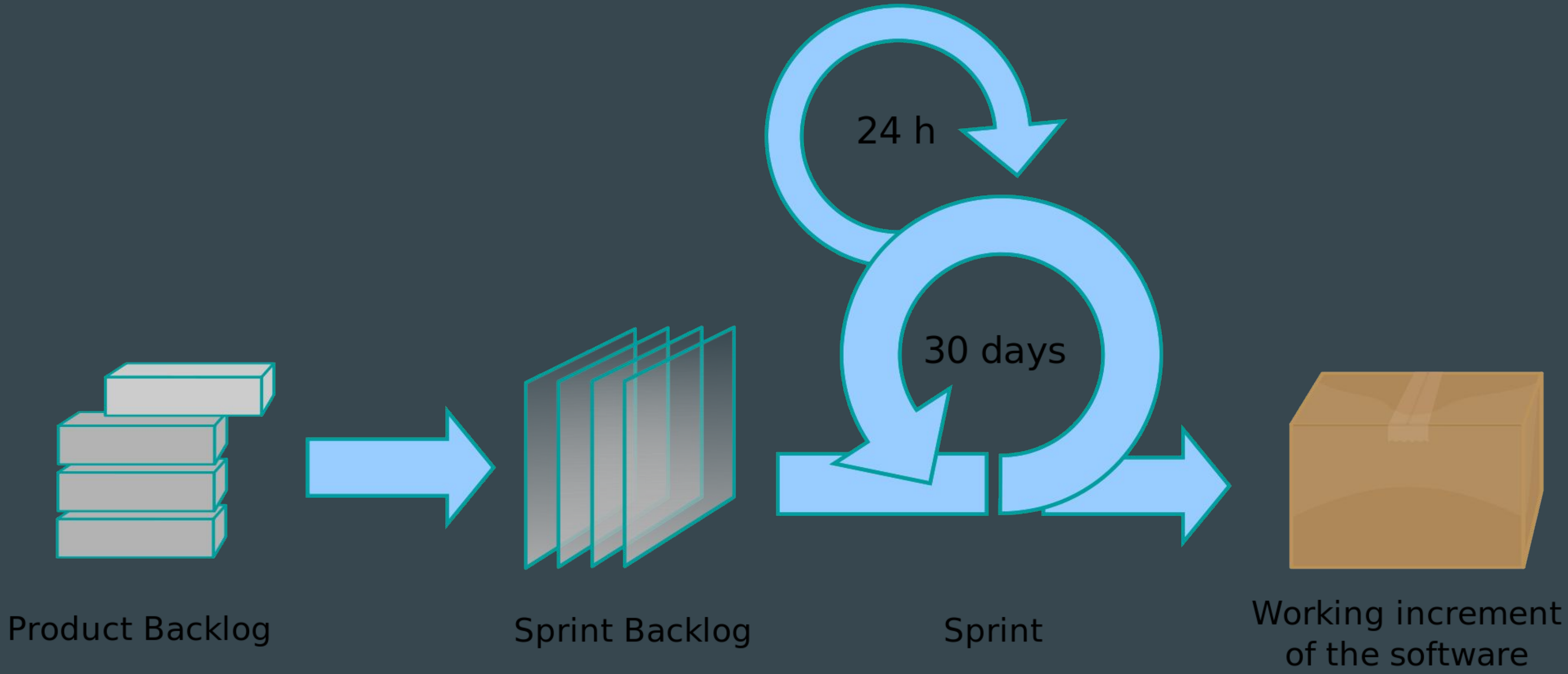
Projects

Usually staffed with 2-4-6 people

Who are responsible for everything

Project routine

- Before start: make sure we have **business** and **tech** contacts
- Kick-off meeting with them
  - Storytelling
- Iterations (1 week)
  - Stand-up meetings (mostly over video)
  - Developers focus on user stories
  - Continuous integration server builds and tests every change
  - Continuous delivery to a demo server
- Iteration planning
  - Demo
  - Prioritization & storytelling
- Until agreed deadline or can finish anytime for any reason


24 h
30 days
Product Backlog
Sprint Backlog
Sprint
Working increment of the software

User stories

Systems evolve implementing user stories

“As a bank’s customer I can pay for my mobile phone”
“Customer can pay for mobile phone”

Every new functionality must benefit someone

This benefit should be verifiable (definition of done)

A story must be smaller than the iteration (1w)

We don’t waste too much time estimating, S/M/L is enough
- see #noestimates

Pivotal Tracker

www.pivotaltracker.com

(Forget JIRA!)

Launching big things

- Pilot / POC (Proof of Concept)
- Internal launch for big/existing organizations ASAP
- MVP (Minimal Viable Product)
  to the end-users ASAP
- Iterative improvement based on user feedback

Technologies

They come and go

Every team chooses programming language, frameworks, libraries

We use Java, Kotlin, Ruby, Python, Scala, C
    + JavaScript and HTML/CSS, etc

Clean code is more important than else

“Leave the campground cleaner than you found it”

Technical excellence

No Agile methodology works without this (hi, Scrum)

- Collective code ownership
- Continuous integration
- Repeatable builds
- TDD - Test Driven Development
- JED - Just Enough Design, Refactoring, YAGNI
- Simplicity, KISS
- Automation (no repetitive tasks, DRY)

Small steps for everything

- Small and frequent commits
- Small and completable user stories
- Small (short) iterations
- Small and regular releases

Good tools matter

Ubuntu workstations

Git

Intellij IDEA and other JetBrains IDEs

Pivotal Tracker

Make your own

e.g. CI alert screen, Git author selector for pair programming, Selenide for concise UI testing

Kronograaf
amserv
amserv-offers
amserv-services
bspb (andrei)
bspb-android
codeborne-tv
codeborne-web
coreless
edrk
ibank
inwise
panda
pixamo-partner-site
primefinance
slice-project
wizener-reader-java
wizener-web

Pair Programming

Nearly 100% of the time

Extreme code review

Two heads are always better than one

Knowledge sharing / Mentoring

Focus

TDD enforcement

Fun (Ping Pong Programming)

(Advanced) Pair Programming styles

Rally: driver and navigator (+switching of roles)

    The best, when levels/experience are close to each other

The tour

    One is showing another the way around, not effective

Backseat driver

    Driver is not touching the keyboard, navigator types

    Best for getting new people aboard

Disconnected pair

    If happens - stop immediately, take a break and “reboot”

http://blog.xebia.com/practical-styles-of-pair-programming/

Collective code reviews

For even bigger teams

Mainline based development

- Single integration branch (e.g. **master** in git)
  - Good/simple to setup Continuous Integration server
  - Nobody is working on old code
- Feature branches only when absolutely necessary
- Release anytime
- Feature toggling if necessary
  - e.g. new features are first visible to a small group of users

or GitHub Flow (usable w/o GitHub)

For our largest team (12+6), to save time on code reviews

https://guides.github.com/introduction/flow/
Simpler than **git-flow** - no special tooling required

All development in feature branches, followed by Pull-Requests

CI server builds all branches and PRs

Branches merged to master **after** installing to production
If not rolled-back for any reason

Arvuti luuletab ja maalib
Unistad tarkvara loomisest, mida kasutavad sajad tuhanded?
Praegune amet on muutunud igavaks?
TULE CODEBORNE'i TASUTA
TARKVARAKOOLI
oma ala parimate juurde õppima!
PARIMAD SAAVAD TÖÖPAKKUMISE!
codeborne

Tarkvarakool - Software School

Facebook works better than newspapers

School for non-IT people experienced in something else

IT in Estonia really needs many more good specialists

Many existing ones are spoiled with ineffective habits

Many students want to become managers instead
(there are too many useless managers everywhere)

e-Estonia image needs to be maintained

Has inspired Vali-IT government program

3rd level of professionalism

1. You are very good at doing it
2. You are so good, so you can innovate
3. You are so good, so you can teach others to innovate, too

   (then your innovation becomes the new norm)

More company culture

Daily company standup

TEX (TEchnology EXchange) weekly meetings

Month’s Last Friday lunch + retro

Fullmoon pub crawling

Out of office working week in Summer

Thanks!

codeborne.com/2015/08/10/working-agile.html