Craftsmanship

... 

The Codeborne way

Anton Keks  @antonkeks
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

Present and past customers in Estonia:

Eesti Energia, Swedbank, LHV, Telia, Starman, Ericsson, Elering, Regio, ABB, Luminor

Plus many from Russia, Norway, Sweden, Japan, Czech Republic

Sponsor of Devclub, Agile Saturday, Tartu Maraton
Codeborne in 2017

32 people
1 CEO/sales
1 office manager
2 designers/front-end
25 software craftsmen
3 software craftswomen
0 project managers
0 analysts
0 testers
Follow your disciplines
Write your tests
Refactor your code
Keep things simple & clean
Do not RUŠHU!
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/
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
- Write automated tests to avoid regressions
- Deploy the system to the end users
- Evolve the system design/architecture by refactoring when needed
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
Product Backlog → Sprint Backlog → Sprint → Working increment of the software

- Product Backlog
- Sprint Backlog
- Sprint
- Working increment of the software

Time:
- 24 h
- 30 days
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, Ruby, Python, Scala, Kotlin, 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

- 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

Linux 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
Pair Programming

Nearly 100% of the time

Extreme code review

Two heads are always better than one

Knowledge sharing / Mentoring

Focus

TDD enforcement

Fun
Collective code reviews

For even bigger teams
Mainline based development

Single integration branch
   Good for CI
   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 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/

All development in feature branches, followed by Pull-Requests

CI builds all branches and PRs

Master is deployable to production at any time, contains only reviewed and accepted code
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 processes

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 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