Agile farming
Uku the PM

Designer-Engineer,
future farmer
Karl the CTO

Engineer, nerd, board game enthusiast, reader of books, geocacher, problem solver
What is eAgronom
Our mission
agile = ¬waterfall
Questions?

Wanna get hired?? 👍 us
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
Culture makes the company

- Farmer first
- Focus and Iterate
- Be frank
Engineering practices

- We are a Ruby shop on Rails
- Front-end HTML, SCSS, TypeScript, React/Redux
- Over 90% automated test-coverage
- Extensive and quick code review on PR-s
- 0-downtime deploys
- Deploy code to production multiple times a day through CI
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
Importance of harvesting
## Starting point

### Harvesting

#### OVERVIEW

<table>
<thead>
<tr>
<th>Crop (variety)</th>
<th>Hectares</th>
<th>Harvested area</th>
<th>Yield (t/ha)</th>
<th>Wet mass (t)</th>
<th>Dry grain (t)</th>
<th>Sold amount (t)</th>
<th>Seeds (t)</th>
<th>Residual (t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alskike Clover (Yet to be decided)</td>
<td>8.18</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Bean (Yet to be decided)</td>
<td>102.15</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Grasses (Yet to be decided)</td>
<td>2.53</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>K 6.1</td>
<td>2.12</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>K 9.1</td>
<td>0.41</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Oats (Yet to be decided)</td>
<td>274.99</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Pea (Yet to be decided)</td>
<td>167.19</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Red Clover (Yet to be decided)</td>
<td>268.02</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Wheat (Yet to be decided)</td>
<td>168.27</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Winter Colza (Yet to be decided)</td>
<td>116.35</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Winter Wheat (Yet to be decided)</td>
<td>35.03</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>1147.71</td>
<td>0.00%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>
Grab low hanging fruits

- Detect
- Prioritize
- Build
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
Why are full-stack developers valued?
Hardest thing in development
Pair programming

- Why
- How
- When
Finished in 4 minutes 43.9 seconds (files took 9.84 seconds to load)
317 examples, 0 failures
Operations: CORE-4052 separate load editing and adding
Kaarel Tinn - #5617, created 2019-07-17, updated 2019-07-17

Operations: CORE-4056 harvesting hectares progress
Kaarel Tinn - #5616, created 2019-07-16, updated 2019-07-14

Operations: Adds harvesting empty state for loads page

Operations: Progress styles
Andres Täht - #5592, created 2019-07-11, updated 2019-07-12

Operations: CORE-4055 add achievements to harvesting
Kaarel Tinn - #5609, created 2019-07-10, updated 2019-07-11

Operations: CORE-4037 progress section
Kaarel Tinn - #5598, created 2019-07-10, updated 2019-07-11

Operations: Harvesting tasks list
Andres Täht - #5564, created 2019-07-09, updated 2019-07-09

On-call: Move making silage to harvesting category
Andres Täht - #5577, created 2019-07-09, updated 2019-07-09

On-call: Remove yield indicators from the bottom of the page
Andres Täht - #5573, created 2019-07-08, updated 2019-07-09

Operations: Fine tune Harvesting UI in mobile
Uku Petters - #5564, created 2019-07-06, updated 2019-07-06

Operations: Create new lists for harvesting overview and varieties lists
Andres Täht - #5501, created 2019-07-03, updated 2019-07-04
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
Explore deeper problems
GOAL: Get more harvesting data

Projects (Harvesting 2019)
- Desktop load adding
- Crop vs variety
- Load adding improvements
- Desktop loads view and editing proposal
- Optional trash and moisture levels
- Reuse last load
- Yield algorithm improvements
- Define yield without loads
- Efficient yield calculation

Define the scope
Explore different ideas
Last week
- LV field diary
- Product amount
- Intercom mobile
- Users location words
- Biplan crash down future
- Offline support Beta
- Crop yield tech plan
- Mobile task ideas
- Mobile task add. discussion
- On-call on-site
- Mobile continued
- Offline indicator related
- Mobile feature test
- Field number research

This week
1. Mobile release procedure
2. LV & EE field diary
3. Mobile work indicator
4. Nationally crop yield
5. Intercom localization & scaling
6. Mobile feature spec
7. Testing webview on-map mobile
8. Task adding mobile QA plan

-2. Commit
-1. cmd beta
0. Release notes
1. Biplan version
2. Commit
3. Cmd

R&D plan
1. Printable map
2. Always continue
3. Task idea
4. LV field diary
5. Product amount
Architectural problems
Architectural problems

- Option 1: Rewrite everything
- Option 2: Iterate
The why
Not all farmers harvest via varieties - others do it via crop. The goal is to make the harvesting module relevant for them.
Design spec: Crop vs variety

The how
Problems
Harvesting is strongly variety based. This document lays out problems and an out plan for combating this and making filtering by crop possible.
1. Harvesting loads schema does not support crop-based approach.
2. Harvesting loads are loaded via load_id
3. All harvesting forms, services work based off of variety_id or variety model as argument
4. Moisture levels are tied to a specific variety

Harvesting loads schema
The information here is denormalized: each load has a specific variety and all seasonal fields have a (the same) variety.
This poses a problem when adding a new load when grouping via crop - what is the variety_id?
The fix is to remove the dependency on harvesting_loads.variety_id - all calculations use specific load fields to calculate its variety/crop.
To do this we'll refactor all aggregation to work based off of load components. If a load component is in the correct crop/variety it's accounted for, otherwise ignored. This means that not 100% of load counts towards what you'll see in overview.
Once that's done, we'll deprecate the variety_id column and stop writing it.

Implications on loads page
Scenario: load with 2 fields with varieties A and B was added under a specific crop and then user switches to grouping by varieties.
Under loads page user sees when grouping by varieties for variety A):
Develop step by step 😄
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
How to ship half-done changes?

And why...
Feature flags

Created by Stenver Jerkku
Last updated Jan 25, 2019 by Karl-Aksel Puulmann

Sometimes you want to develop the feature without releasing it to end users. Reasons for this might be to avoid huge PRs, to test and release things internally or to make a feature country-specific.

For this purpose we have feature flags. You can check if a feature flag is enabled by:

```
is_beta? flow control
1   if Feature.enabled?(:new_feature, company)
2       call_new_feature_method
3   else
4       call_legacy_feature_method
5   end
```

or, it can be used in the views:

```
<% if feature_enabled?(:new_feature) %>
   <div> My new view code </div>
<% end %>
```
Feature flags are awesome, Feature flags are great. Sustainable dev processes They help us to create.

On the basic, is feature ready? Call 5 customers, for them turn it on. Once initial feedback is steady You'll know what works, what is wrong.

New feature has a bug? Great! No need to curse the fates. Rather, tell your teammates "Hey! Time to close that gate!".

Product needs payment tier or fifty? `set customer.access_flag to no` Replacing a service that is tricky? Gradually turn new one on, enjoy the show

Staying up late to deploy? Never again, I'm not a slave. I'll find a PM to annoy To turn it on, to start the game

Some rollout needs "business" approval? I would rather avoid meaningless debates How about a features removal? Flag it, measure, iterate.

There's so much more I could talk about short-term branching, A/B testing, experimenting and living without staging. However you got the point, you found out That feature flags are the best. I'll let Uku cover the rest
Enable or disable this feature for everyone with one click.

**Percentage of Actors**

- or -

<table>
<thead>
<tr>
<th>0%</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>100%</th>
</tr>
</thead>
</table>

Enable

Percentage of actors functions independently of percentage of time. If you enable 50% of Actors and 25% of Time then the feature will always be enabled for 50% of users and occasionally enabled 25% of the time for everyone.

**Percentage of Time**

- or -

<table>
<thead>
<tr>
<th>0%</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>100%</th>
</tr>
</thead>
</table>

Enable

custom (ie: 26, 32, etc.)

Enable actors using the form above.

**Groups**

- EE_companies

**Actors**

- User:4
Revalidate and improve
1. Culture and practices
2. Starting an agile project
3. Hardest thing in dev
4. Build the right solutions
5. Ship broken things
6. Show must go on
Are we done?

It’s a never ending messy world
Improve processes
Challenge your own system
Recap

- Culture makes the agile
- Communicate all the things
- “Never” do a rewrite
- Feature flags are awesome
- No plan is perfect, but plan regardless
- Only constant is change
Questions?

Wanna still get hired?? 🙏 us

(uku|karl)@eagronom.com

eagronom.com