Refactoring Space

Energy Drink for Your Codebase

michael.mai@valtech.com

valtech_


IBM
Home
Service
Products
Portfolio
Team
Contact
Nav Menu

00

section

Why?

Flexibility and Adaptability

- Why?
- How does it feel?
- How to stay yourself?
- What happens to our company culture?

NOKIA OSAKEYHTIÖ
NOKIA AKTIEBOLAG

NOKIA

NOKIA

valtech.

Survival

- Existence!
  - … in the face of competition

Changes in the market are the norm

Holding course is granting your **COMPETITION** the **WIN**

Disruptive brands

- Ford Motors
- Tesla
- IBM
- General Electric
- Patagonia
- Gap
- FedEx
- McDonalds

Refactoring business model

- Nokia
- Yamaha

Selection from https://fabrikbrands.com/25-disruptive-brands/

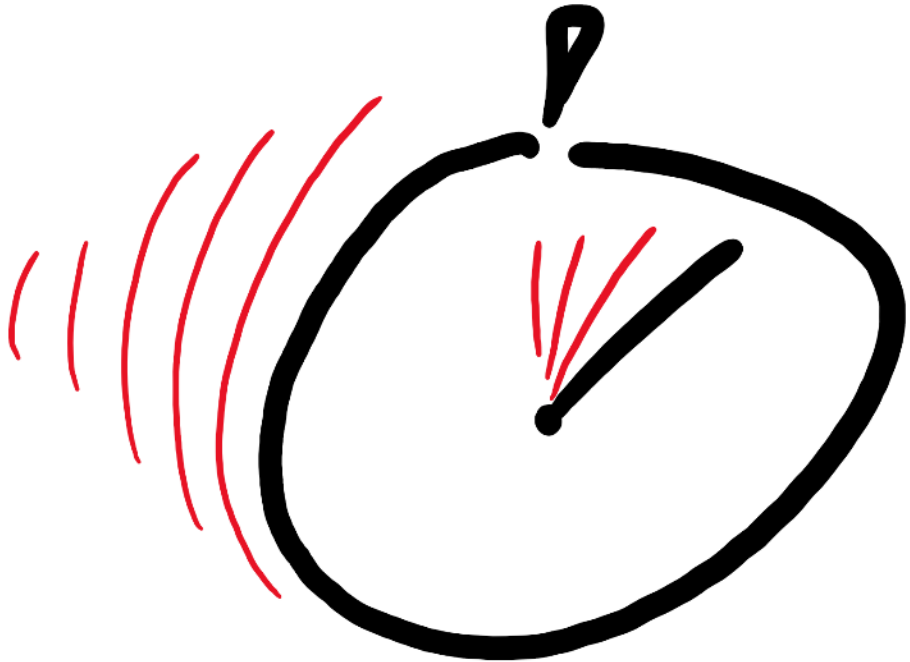
01

section

How?

Stable and fast build system

Technical foundation



Stable and fast build system

Technical foundation

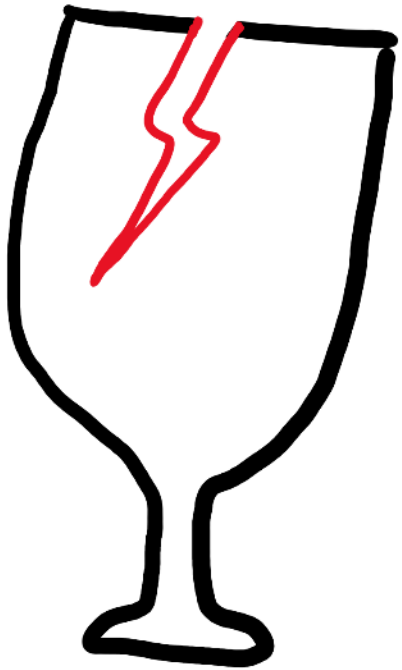
- Don’t have this?
  - **Great** → Gather volunteers and enthusiast and start building your Build
- Consider
  - CI is a development practice
  - Tools and process should be “helping hands” neither masters nor tyrants
- As your teams are IN …
  - The Build fits the needs
  - Include additional experts during refinement and eventually SP2
  - Necessary modifications can be executed directly
  - Potentials are faster identified
- … but …
  - Maybe you need to change your org-structure

LeSS

- “More with less” – Teams own their processes

Stable and fast verification

Business reliability



Stable and fast verification

Business reliability

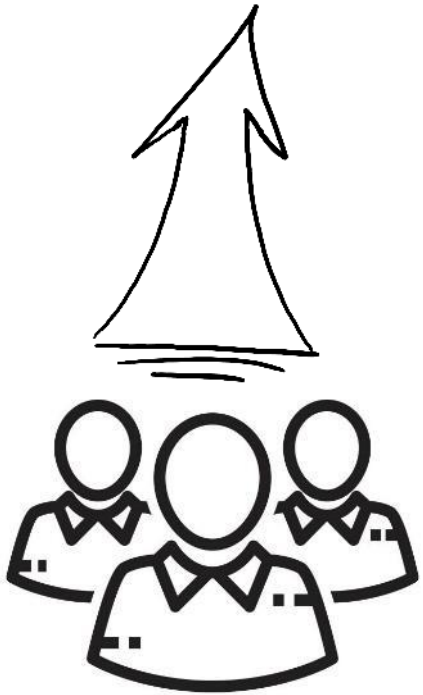
- Don’t have this?
  - **Great** → Gather volunteers and enthusiast and start building automated business verification and validation tool chain
- Consider
  - Proving your business case
  - Providing fast and valuable feedback to developers
- Reliability …
  - Precise and fast
  - Complete (may not be fast)
  - Exhaustive (definitely not fast)
  - Complaint
  - Acceptable
  - Desirable (by customer & target group!)
  - Ecosystem (also foreign ecosystems)

Agile Manifesto

- “Working Software”

Support by teams

Bottom-up



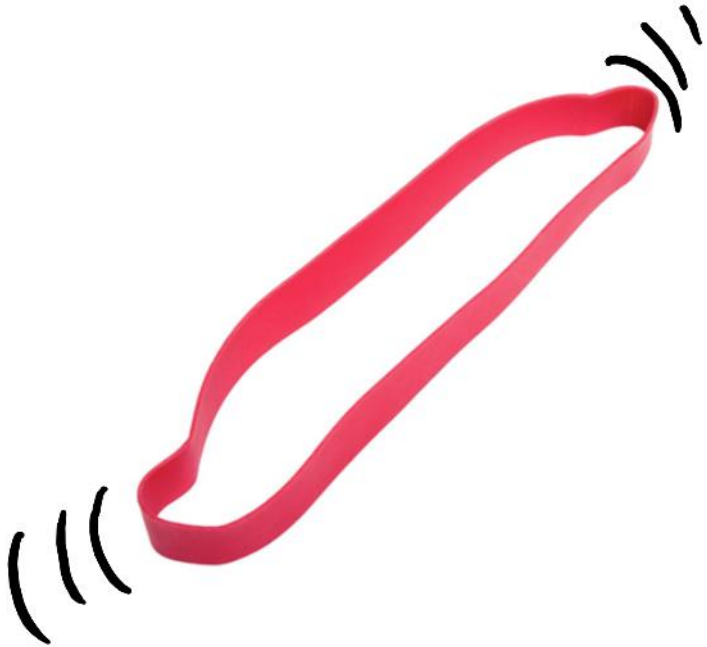
Support by teams

Bottom-up

- Don’t have this?
  - **Great** → Gather teams and coach on business case, life-cycle, technological-life-cycle, …
- Consider
  - A profitable business case runs the company
  - Profitable should not be limited to short-term view
- Understanding …
  - Technology
  - Business
  - Competition
  - Operation and service cases
  - Shifts and disruptions

Support by (middle-) management

Top-down / Middle-down + Middle-Up

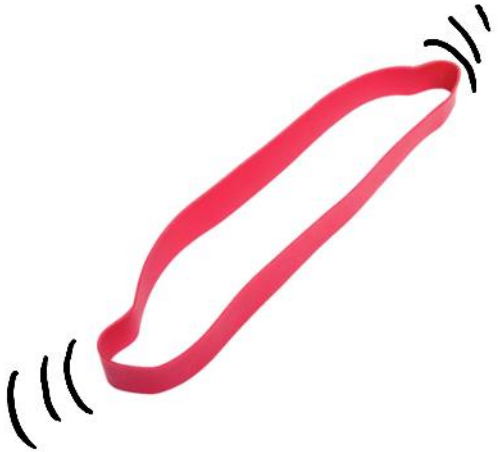


Support by (middle-) management

Top-down / Middle-down + Middle-Up

- Don’t have this?
  - **Great** → Collaborate with sponsor and coach on business cases and product vision

- Consider
  - Product Vision need to inspire your employee first



Support by mind

… now this is difficult …

Support by mind

… now this is difficult …

- Don’t have this?
  - **Ooch**

- You may have seen …
  - Lack of volunteers
  - No team is raising for refactorings
  - No consideration of refactorings during SP1 and SP2
  - APO neglect technical improvements

“It is difficult to get a man to understand something when his job depends on not understanding it”

Upton Sinclair

But maybe this is
“Means and ends”
confused

~~Support by~~ mind

… now this is difficult …

ENABLE MINDS TO SUPPORT

- Don’t have this?
  - **Ooch**

- You may have seen …
  - Lack of volunteers
  - No team is raising for refactorings
  - No consideration of refactorings during SP1 and SP2
  - APO neglect technical improvements

> **“It is difficult to get a man to understand something when his job depends on not understanding it”**
>
> Upton Sinclair

But maybe this is
“Means and ends”
confused

Dynamics!

“We don’t want to refactor”

What to we really need?

- What do you really need?
  - Why are you in business?
  - For how long do you want to stay in business?
  - Why are your customer with you?

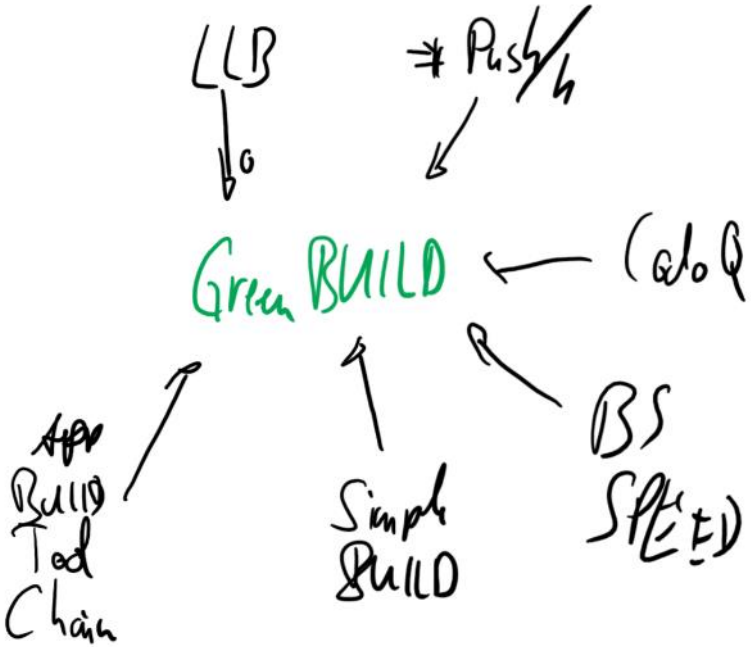
We want to attract more developers
- **So we can increase our capacity to outrun our competition**

We want to have releasable builds, even during a sprint
- **Release as the business sees fit**
- **No tyranny of (major) releases**

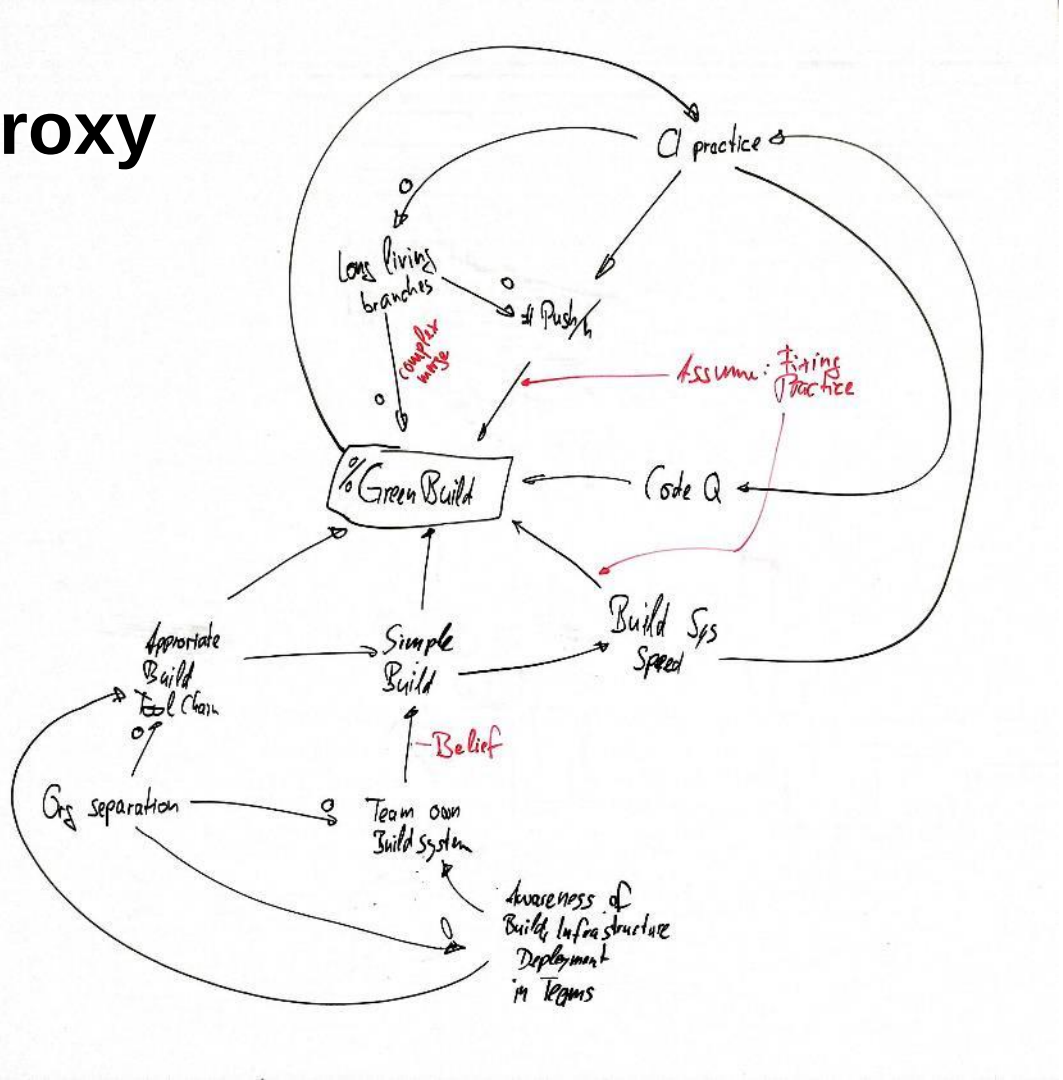
We want to be business flexible
- **So we stay alive as a company**

We want to develop junior developers fast to experienced/senior developers
- **We want to hire cheap “juniors”, get them fast to cheap “senior” developers**

Exploring a proxy



Exploring a proxy



Breaking the gordian knot

Entering …

Refactoring Space

- Providing a “safe” place to learn
- Providing a helpful space to fail and get direct coaching without hassle
- Space on high-level so failing is okay – What happens in Vegas stays in Vegas

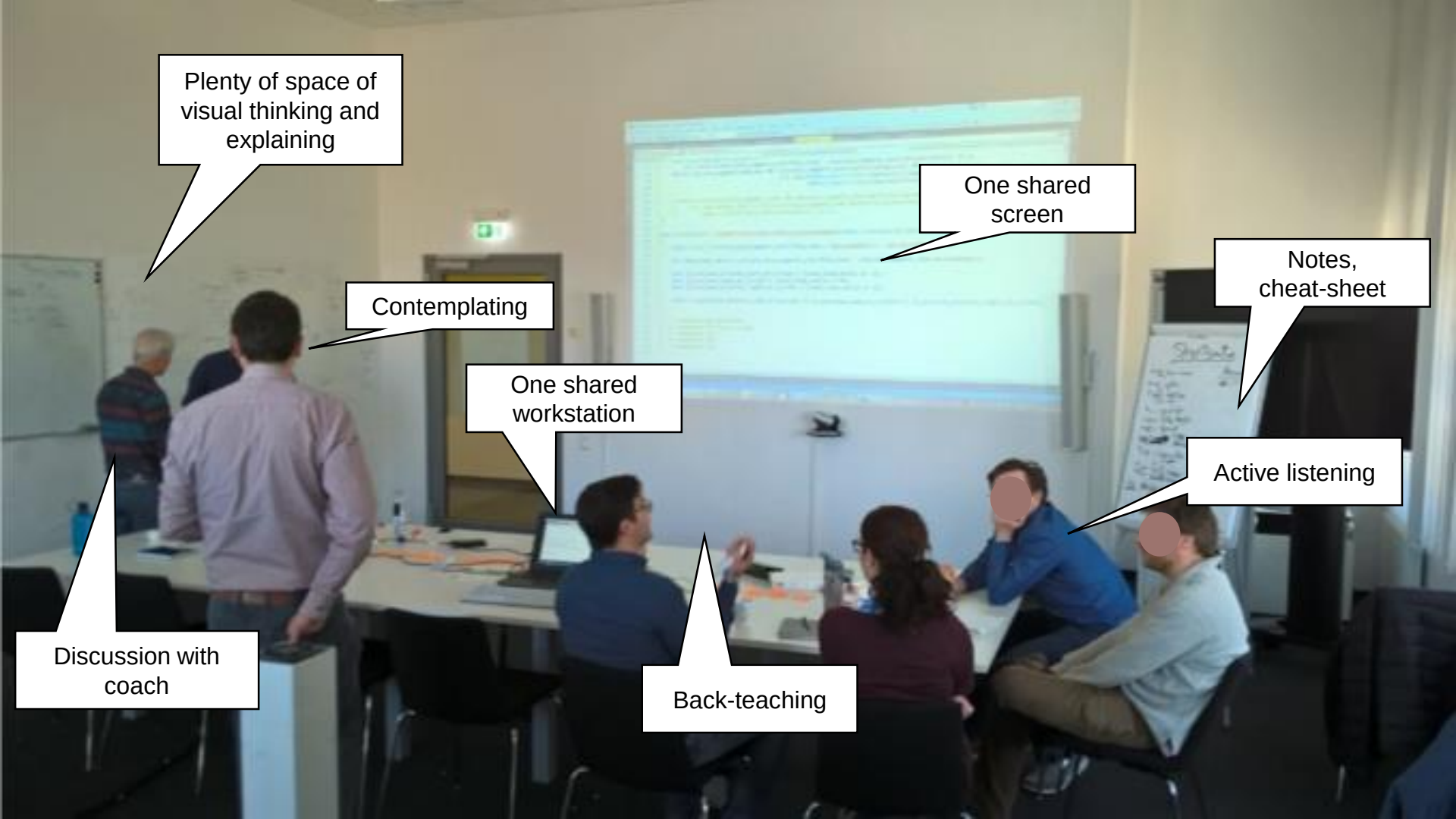
Coaching

Staying in the fire with your mates

03

section

What?


Plenty of space of visual thinking and explaining
One shared screen
Notes, cheat-sheet
Contemplating
One shared workstation
Active listening
Discussion with coach
Back-teaching

valtech_

From the trenches

Important outlines

- Implementation and design quality is a responsibility of the teams
  - Not related to any product backlog item required, but all
- Individual team members are welcome
  - Lower the entering barrier
- Vegas rule apply
  - Watch your company culture

Important outlines

- No preparation needed by joiners
  - Learning mind required, thou
- Developer machine (e.g. laptop) required
  - Easy application of learned practices in day-to-day work
- Dedicated room with whiteboards
  - Reduce complexity

Steps

- Ensure lateral support from disciplinary managers
  - Why lateral?
- Advertise to APOs
  - So they don’t block teams who wish to join a refactoring space
  - Most of the time PO does not interfere
- Promote through Scrum Masters
  - Connection makers

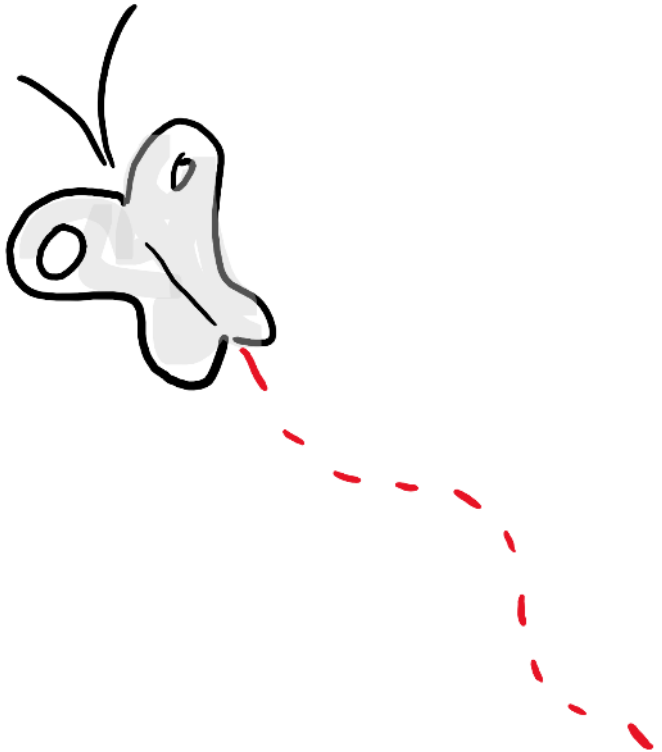
Steps

- Advertise in Communities
  - So they may spread the word within their enthusiastic members
- Hinting “prime” subjects teams
  - Offer additional support
- Visual

- No jingle ;)

How to get it fly?

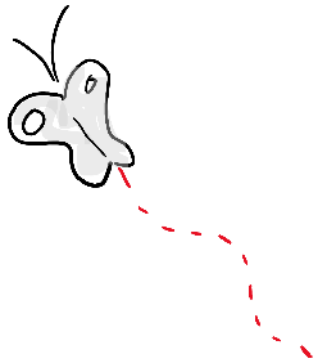
Many small refactorings keep you nimble, big ones adjust you to the market



How to get it fly?

Many small refactorings keep you nimble, big ones adjust you to the market

- Enthusiastic developers
- Spreading the word
  - Building success stories
  - Starting a movement
  - Conquering the code
- Let the results speak for themselves



05

section

Retrospect

Keep – Improve

- Keep
  - Small groups
  - Good ratio Coach / Developer
  - Open to current needs of joining induviduals
- Improve / Change / Experiment
  - Traction
  - “Doing good things and talk about this”
  - Getting clear of (contract) limitation whom to include in the session
  - Become first class in work schedule
    - No, its not slack
    - No, its not a separate item

06

section

Addendum

Q&A during session

Addendum

What to refactor first? Focus for refactoring space?

- Beside the foundation (build system, business reliability, …) there is no preferred sequence

- Avoid management-like direction
- Ask the question behind the call
  - E.g. Call: “We need better documentation”
    - Question behind: “Is the code to complicated to understand“ → consider: investing in Clean Code rather on documentation
    - Question behind: “Is the interaction between objects complex to understand” → consider: investing in Test Driven Development, Clean Architecture and/or Hexagonal Architecture, explanatory tests that suit as documentation and auto-generation of documentation from these tests and public API

Addendum

Time schedule? Question of scaling

- The concept of Refactoring Space is that of super charger for learning
  - First, establish learning
    - Once a week / twice a Sprint
    - half-day to full-day
    - Volunteers
  - Second, scale it
    - Everyday
    - half-day to full-day
    - Volunteers
  - Third, emerge into normal mode
    - No dedicated Refactoring Space needed any more
    - Expect spontaneous mob programming session between arbitrary teams and developers → support them

Addendum

Always with technical excellence coaches?

- Partial “Yes”
  - Yes, technical coaches should be present in the beginning to smoothen the kick-start of learning
  - Yes, technical coaches should be present for complex refactorings and restructurings in the first place. Students can learn to avoid pitfalls and dead-ends early on. So they are able to teach their colleagues tips, tricks and practices.

- Partial “No”
  - No, over time there is no need for technical coaches as the skills of the developers increase and are able to teach each other directly
  - No, over time a non-technical facilitator is able ask the “right questions” and therefore support students and developers in learning, refactoring and restructuring

Addendum

Should I wait for the next refactoring space to do refactoring?

- No, you shouldn’t wait for the next Refactor Space to refactor
  - The Refactoring Space is intended to kick-start the learn of “how to” refactor and restructure code
  - As soon as you see an opportunity, please do refactor and restructure

  - If you feel unsecure → pair or mob
  - If you feel unsecure → first improve test coverage (code and function/business case coverage)
  - If you feel unsecure → commit your change to a pull request for feedback before merge
  - If you like to socialize → pair or mob

Thank you

http://www.agilesoftwaredesign.de/posts/2019/refactoring-space-less2019/