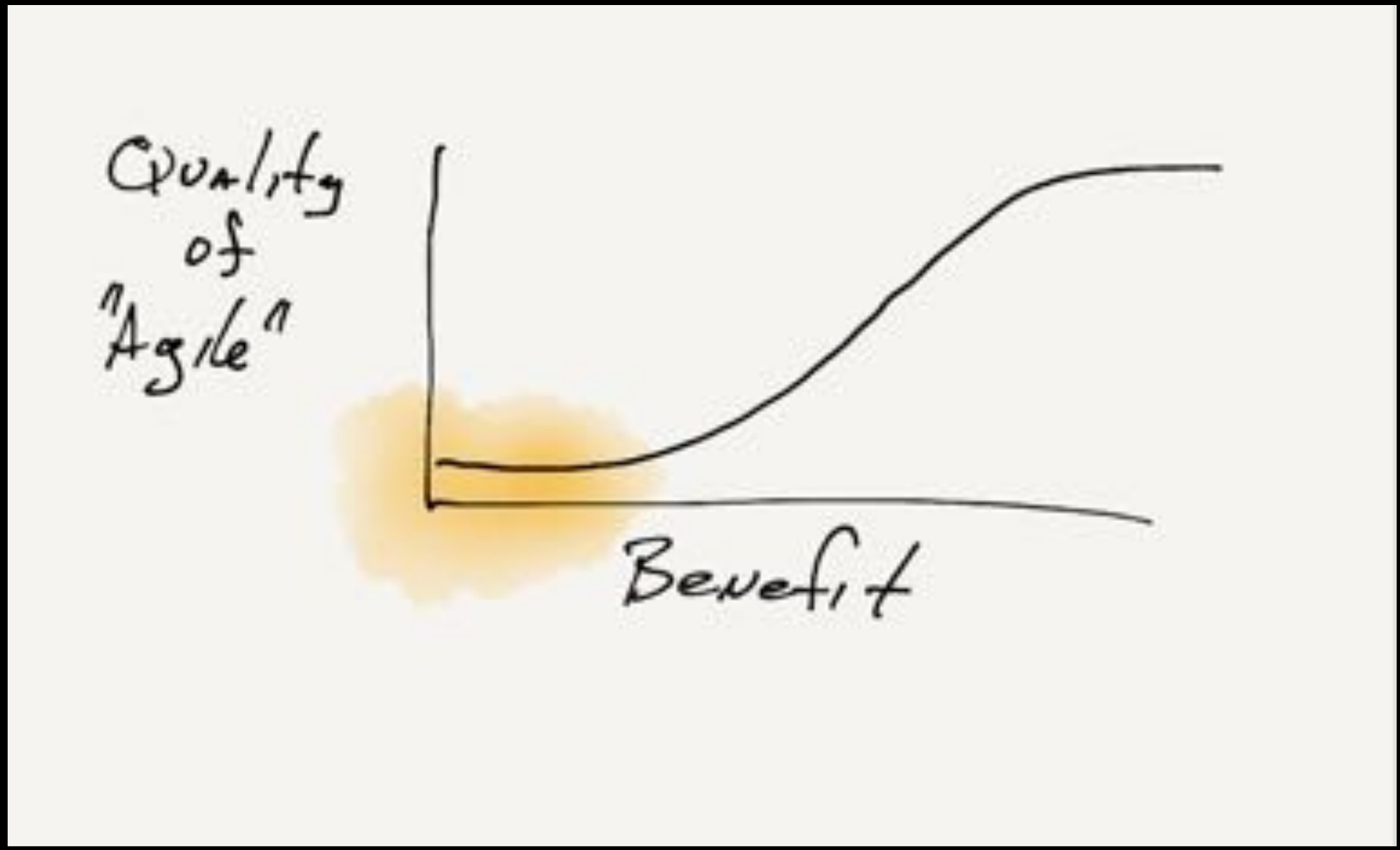
Developers:
Abandon Agile

Ron Jeffries
Chet Hendrickson

We have some questions

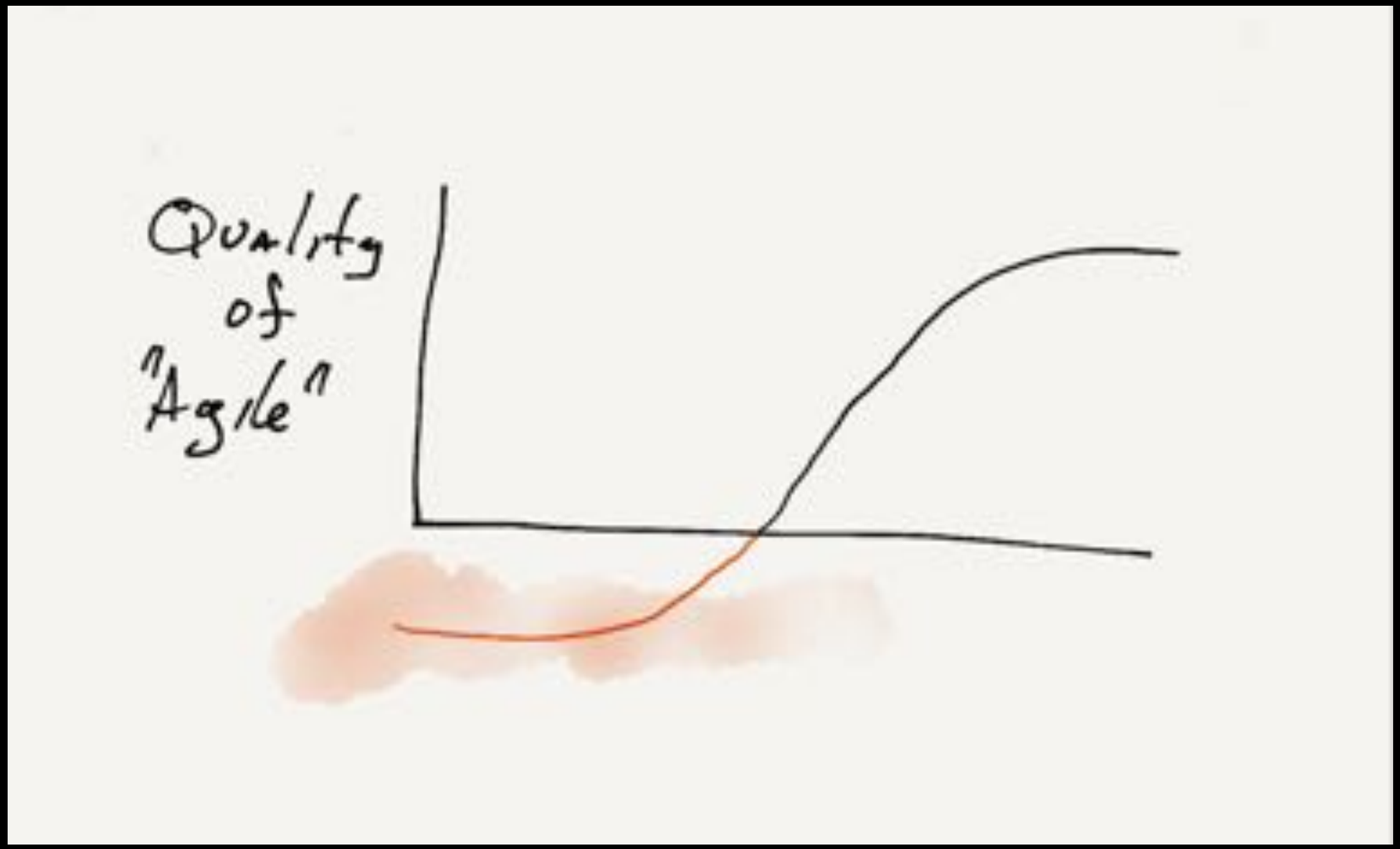
- What "method[s]" are you using?
- What problems and issues do you see around you?
  - Management
  - Team
  - Other

Answers?



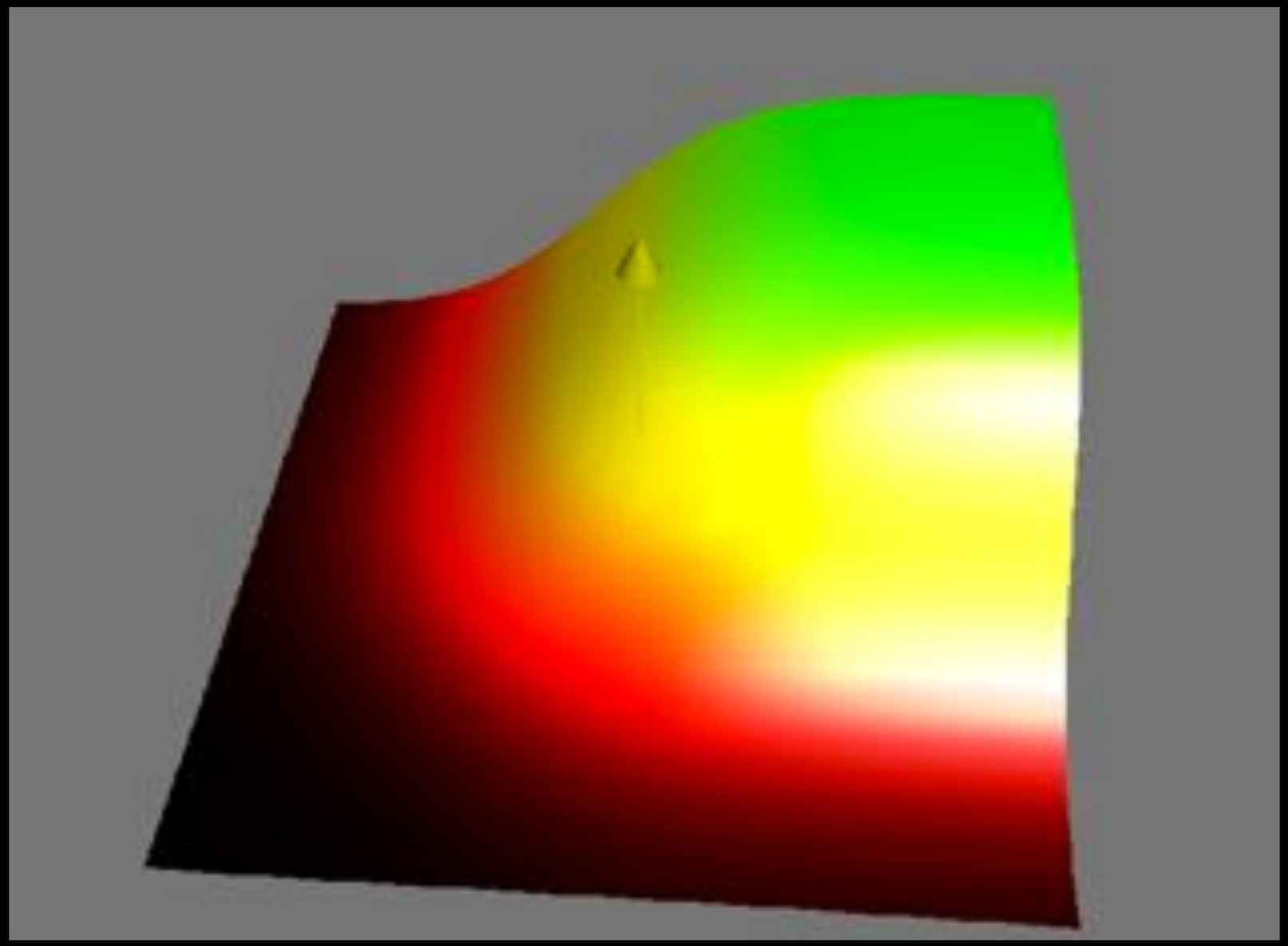
Agile Effect

on the business



Agile Effect

on the developers



Effect of skill

x = technical skill

y = business skill

Dark Agile

Recognizing Dark Agile

- You'll know it when you see it
- Pressure
- Defects
- No Increment
- Slowing Down


pressure
BLACK
GIRLS
CODE

MUST GO FASTER

"Whip the ponies harder"

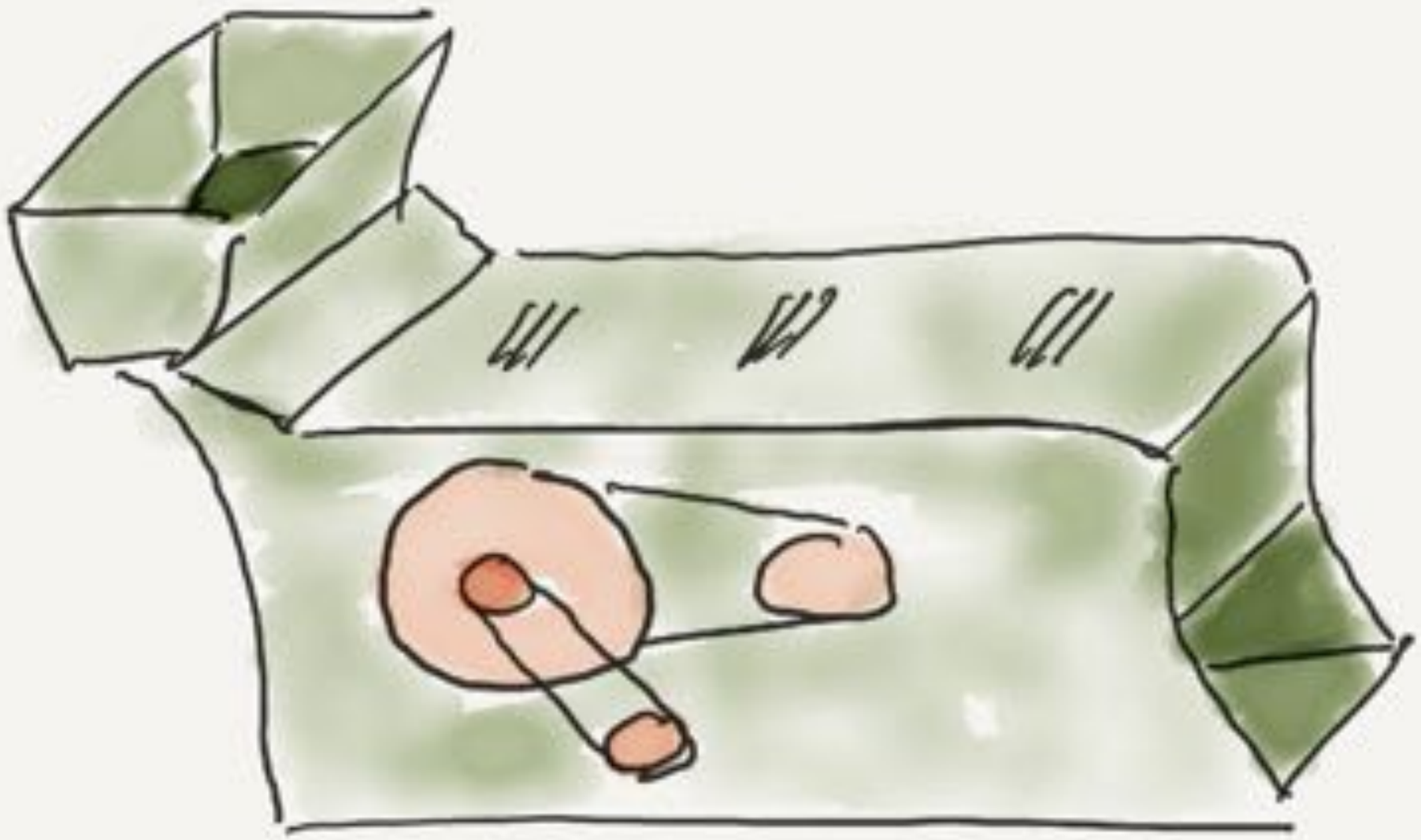
"Whip the ponies harder"

MUST GO FASTER

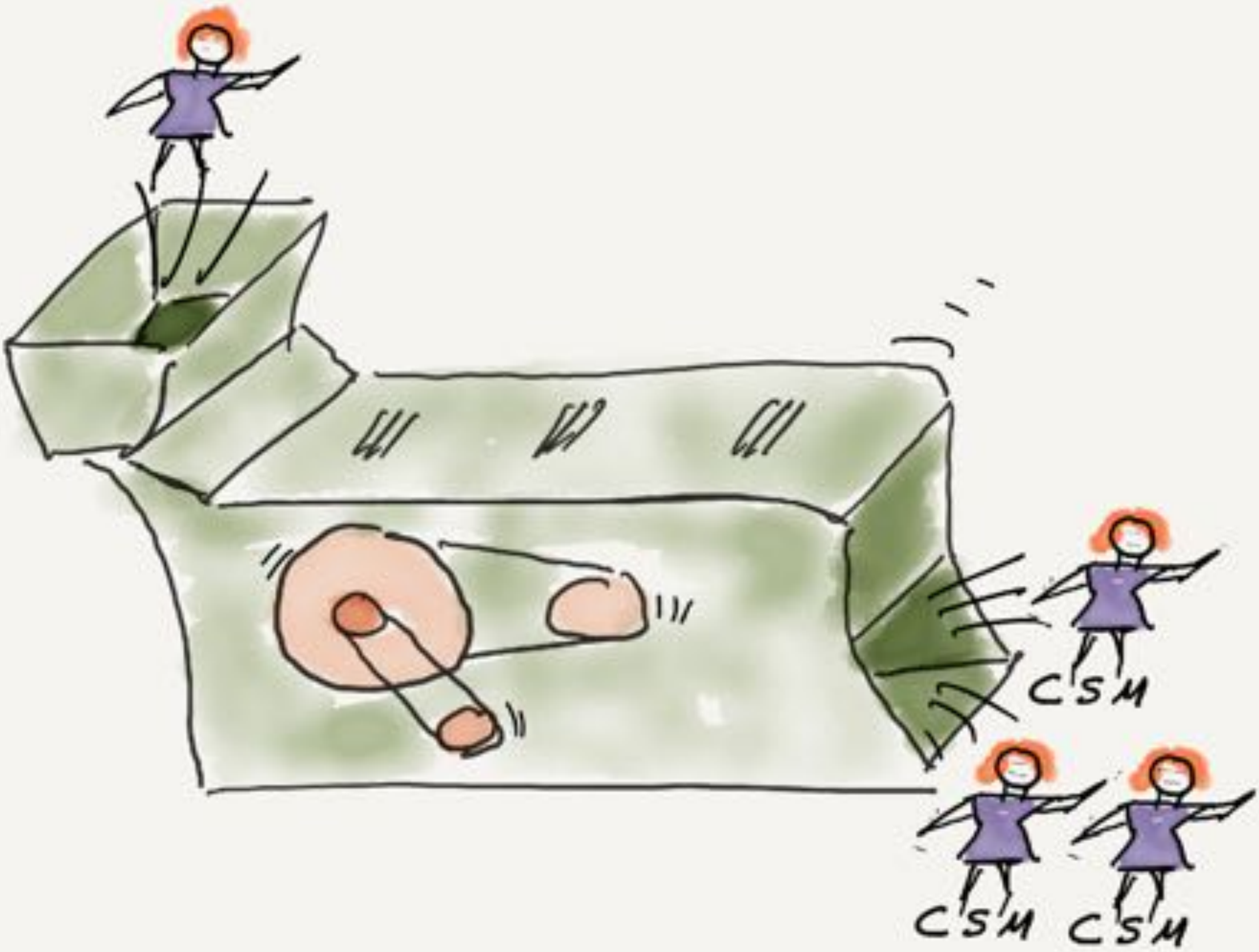
Make the world safe for developers

SAFe?

The "Agile Machine"



Agile-Industrial Complex Responsibility


CSM
CSM
CSM

They broke it
They should fix it

- Form over function
- 2-day easy certifications
- "Twice the work in half the time"
- JIRA
- Right-side vendors

No!

- It's not the Agile Machine's fault.
- They're not always helping, but they're not the root cause.
- If Agile's not working, it's down to us developers!

Yeah, no.

It's not up to them

it's up to us

Success with Agile

requires

Agile software development

What the Agile Machine sells
is not
Agile Software Development

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

The most efficient and effective method of
conveying information to and within a development
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly.

“It’s all talk until the code runs.”

–Ward Cunningham

The Software is everything!

- Turn conversation concrete
- Negotiate and educate
- Software development,
    - do you speak it?

To deliver the software every iteration, we need to be able to

- Develop Software Well
  - continuous in-team testing
  - continuous integration
  - incremental design improvement

TDD
ATDD
Refactoring

or equivalent,
if you know one

Do not create the scissors of bad design and then run with them!

Key Takeaways

(sounds like XP)

- Rise to software excellence
  - Our software is always available
    - Running, tested, usable
    - Keep promises
    - Good design all the time
- Separate ourselves from "Agile Scrum"
  - Remain independent of process,
  - Develop software with excellence

Abandon Agile

Do Agile Software Development

No method, just software excellence

Not even XP, though it's a great start.