Introduction to the Course

01219245/01219246
Individual Software Process
Spring Semester 2014

The original idea of the course

- This course is the first one in the series of two software process course.
- The second one is Team Software Process.
- Both courses play the same role as PSP/TSP (PSP – Personal Software Process, TSP – Team Software Process) under the umbrella of CMMi.

Personal Software Process

- PSP was developed in 1996 to apply the idea of CMMi to the level of single developer.
- The idea is to use **data** to **improve** your development process.
- "data" = something that you can measure.

A process?

- A **process** is *how* you work on something.
- Not following any process is, in itself, a process.

PSP

- Input: a *complete* requirement
- Output: a product (+ other artifacts)
- There are 3 maturity levels:
  - PSP0: start measurement
  - PSP1: estimation and planning
  - PSP2: quality and design
- A lot of data is collected along the way, using various forms.

Why don’t we use PSP?

- The main objective for using data to improve the process is timeless and remains valid.
- However, software industry changed a lot from the day PSP was developed.

diff

- We have "better" insights on how to develop software. (We seem to understand why it is hard.)
- New development practices and tools were developed.
- The business competition gets a lot stronger; this requires a cheaper, faster way to build products that satisfy the customer’s need.

OK, so what’s now?

- We will try to cover various skills you need as a developer.
  - There are tons of them, but we will focus on the ones you can use/practice even if you are a solo developer.
- What are the skills/practices that you need to learn?

A biggest picture

- Software is at the heart of business.
- What are things that business concerns?
- Let’s see another nice video on how to develop a business model:

  Business Model Canvas Explained

  http://www.youtube.com/watch?v=QoAOzMTLP5s

A big picture

- Customer segments and value propositions should match. This is **VERY HARD**.
- Lean startup
  - http://www.youtube.com/watch?v=i65PaoTIVKg

A smaller picture

- Let’s see how a team fits in the modern software development process.

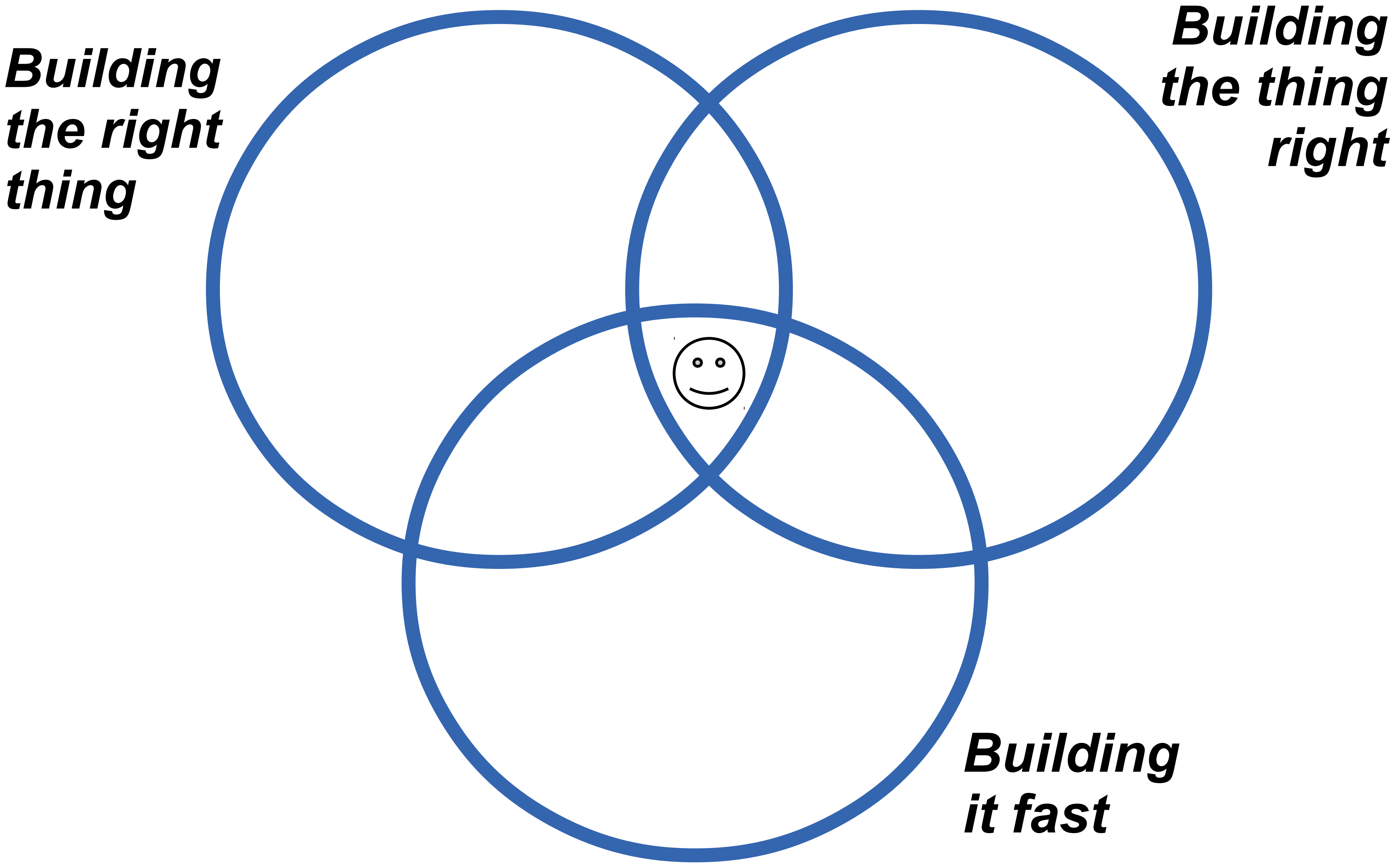
- Agile Product Ownership in a Nutshell

  http://www.youtube.com/watch?v=502ILHjX9EE

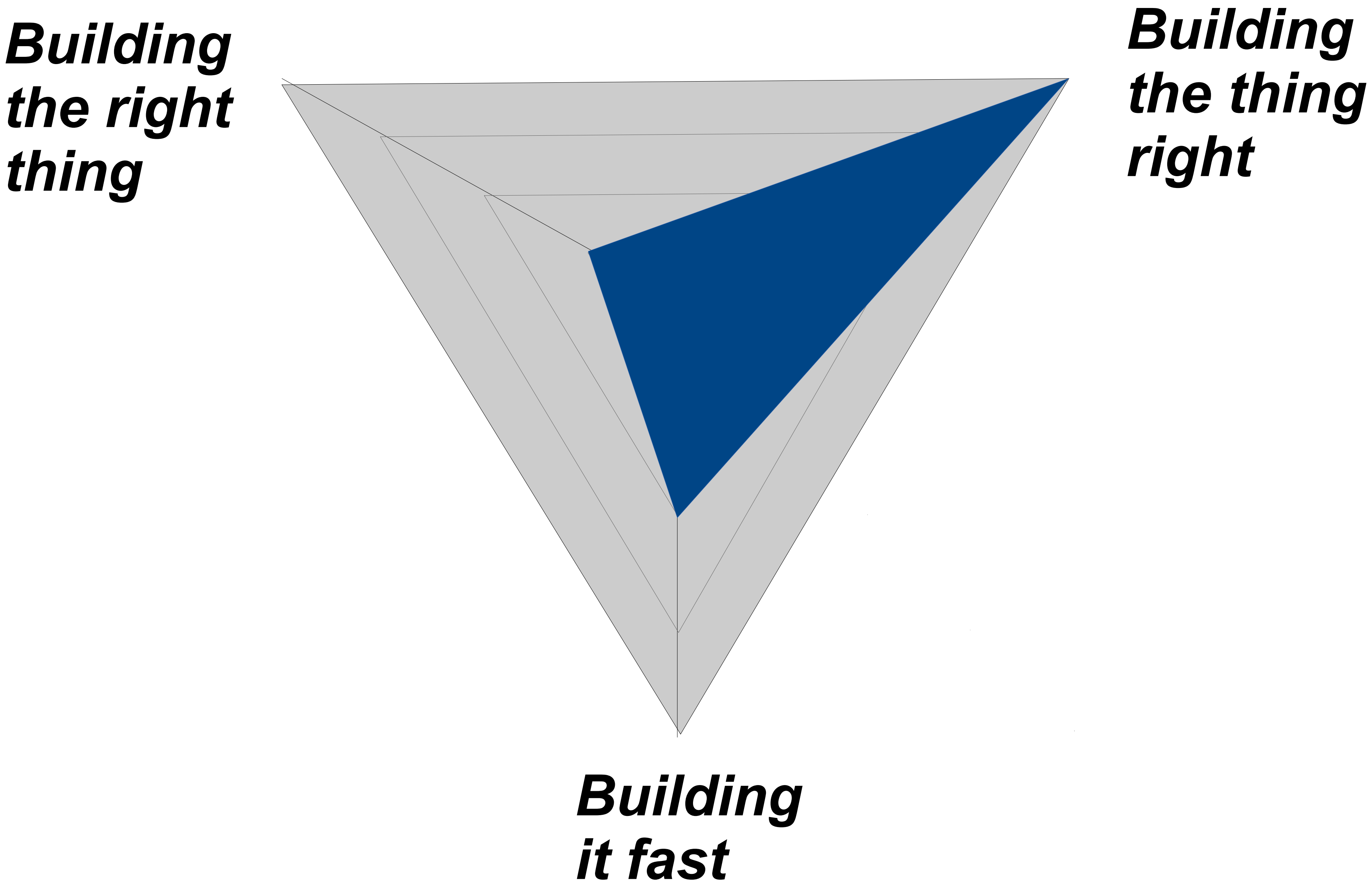
  or here:

  http://blog.crisp.se/2012/10/25/henrikkniberg/agile-product-ownership-in-a-nutshell

Ideal: a perfect balance



This course

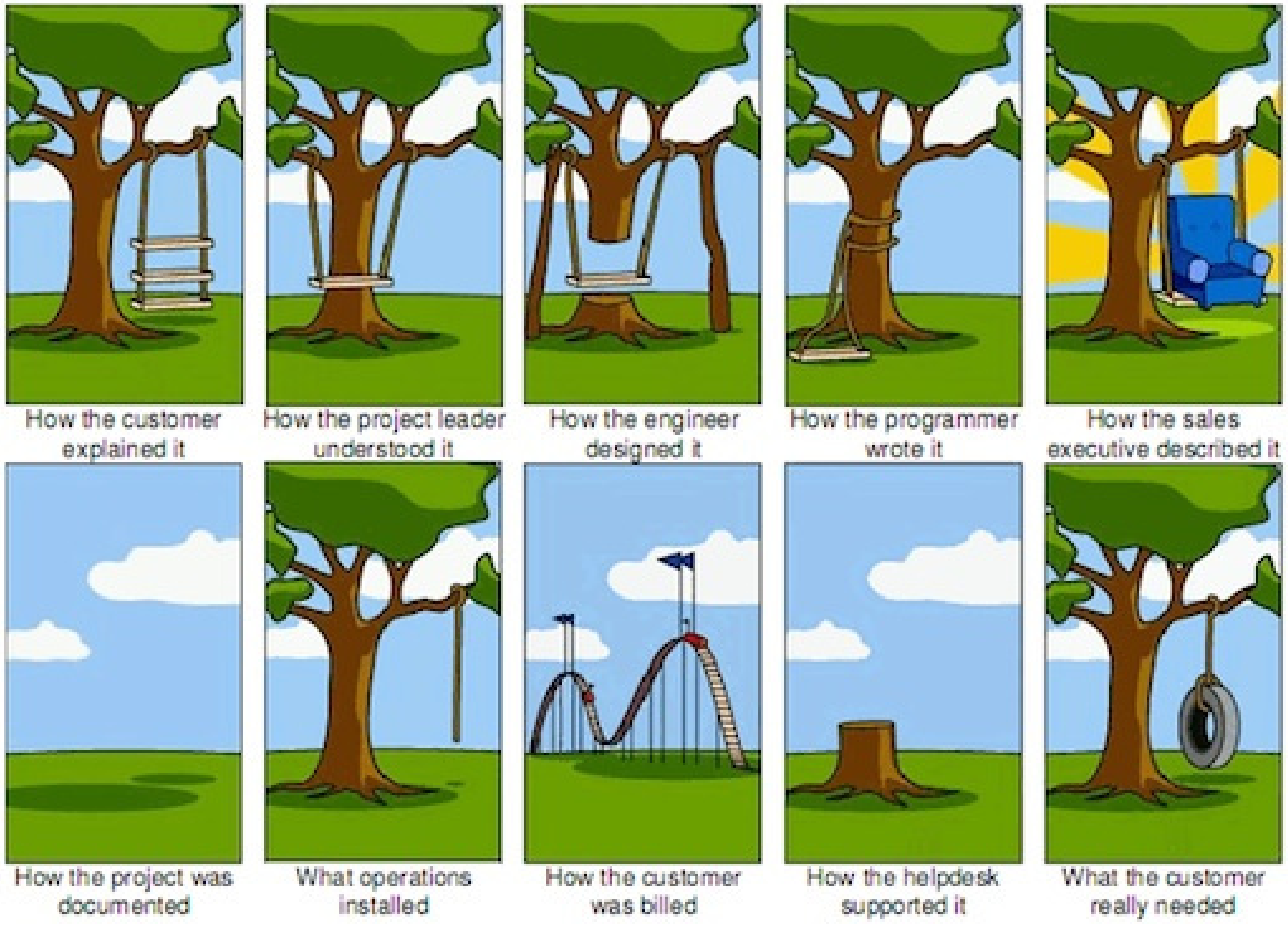

Building the right thing
Building the thing right
Building it fast

The goal of this course

- Short version:

  To learn basic software development skills needed for an individual developer

Why software development is hard


How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

from: http://www.projectcartoon.com/

Communication

- One of the biggest problems in software development is communication problems.
- But that’s not all...

Compare these

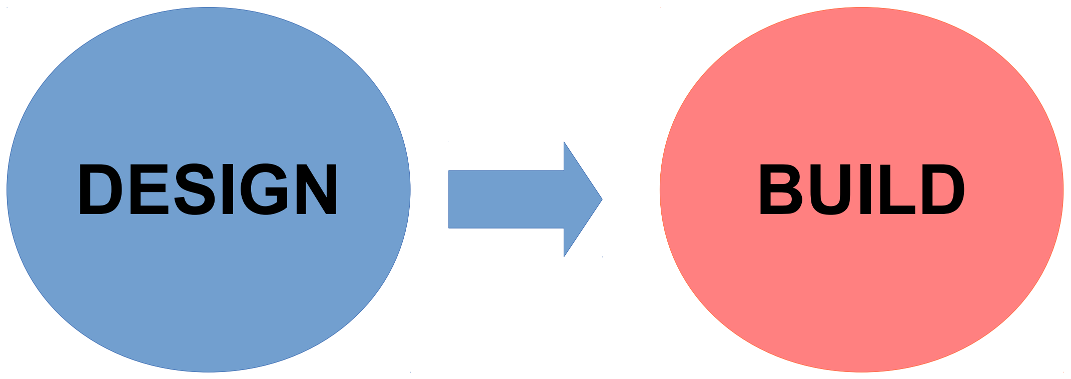
- I want to build a house

- I want to build a website for house builders

Which one is harder to imagine how the product looks like, and how it should work?

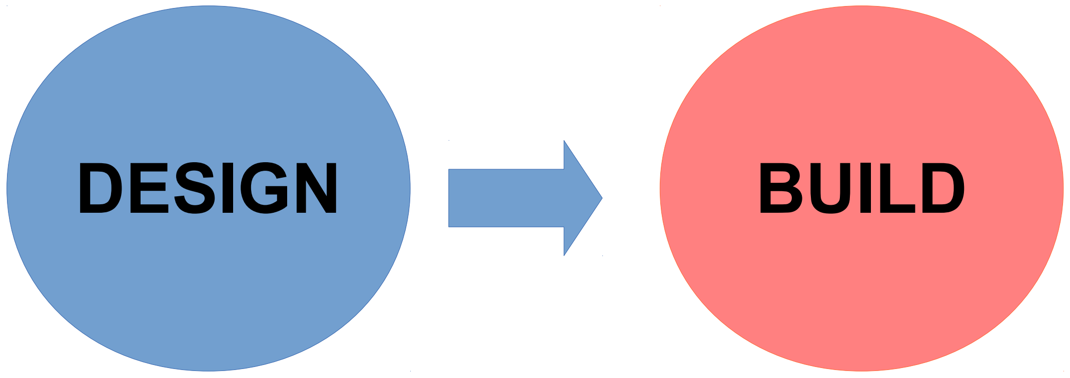
Building a house

- You can design and model a new house.
- When the model is ready, you can start building the actual house.



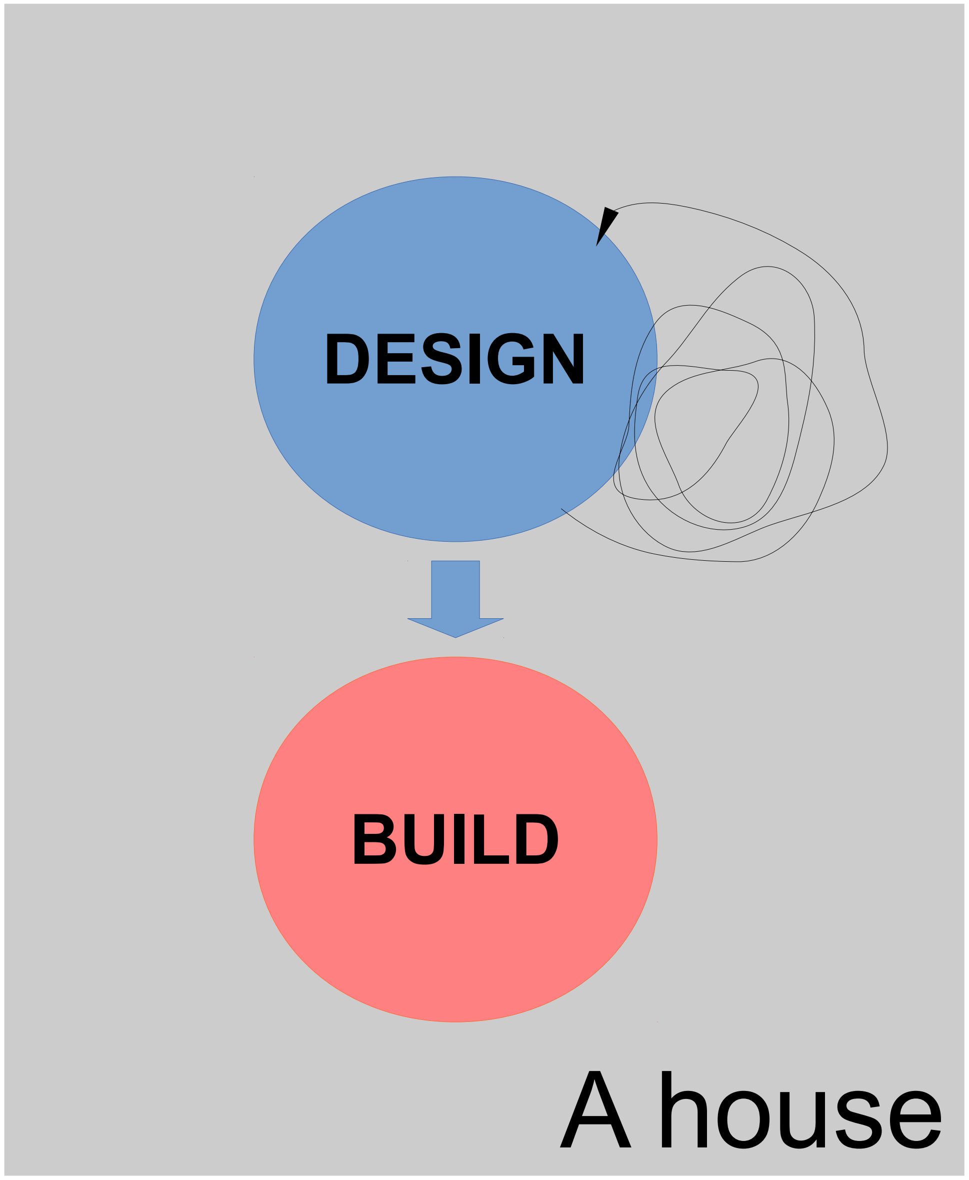
Building a software

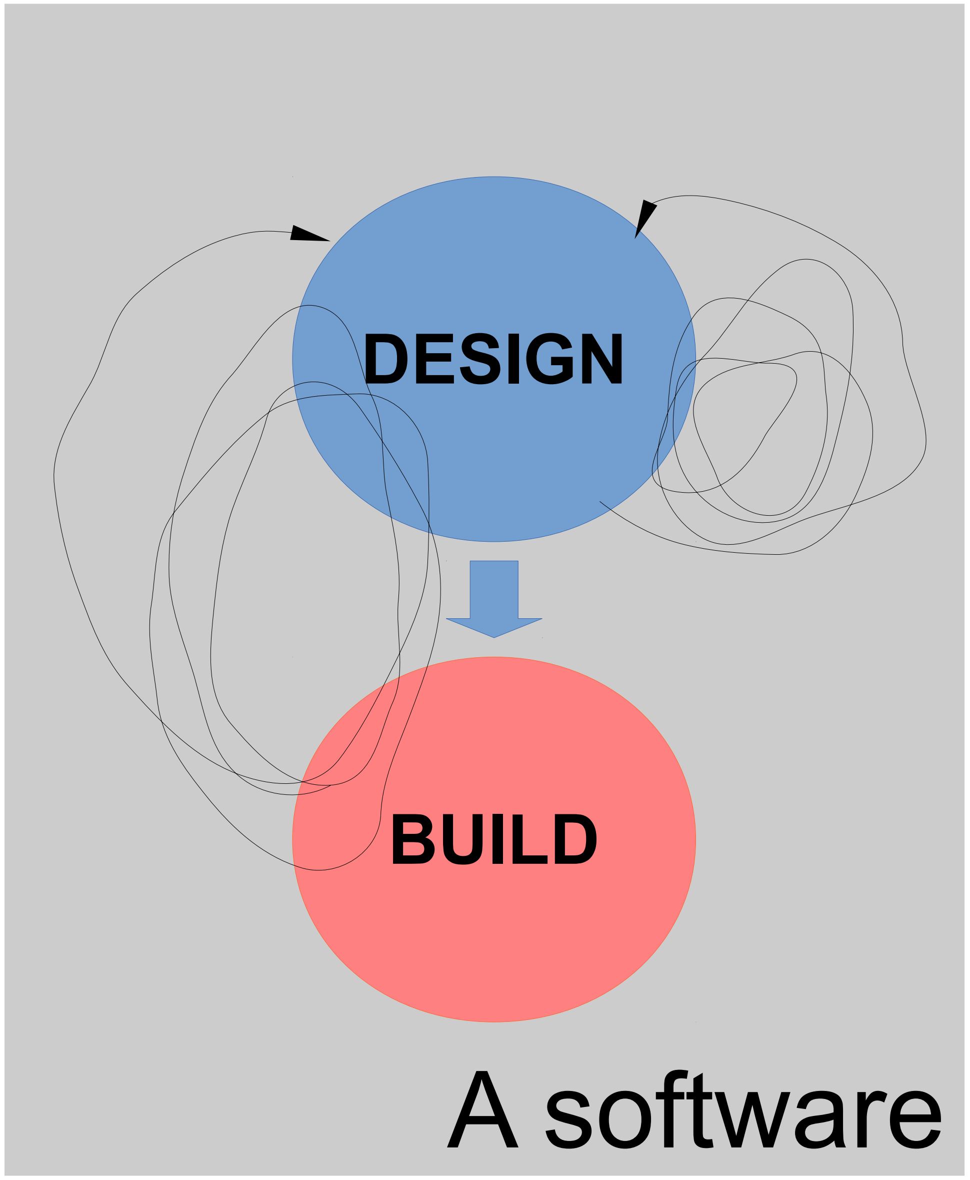
- You can design and model a new software.
- When the model and design is ready, you can start building the actual software.



What's wrong with this approach?

Oh...



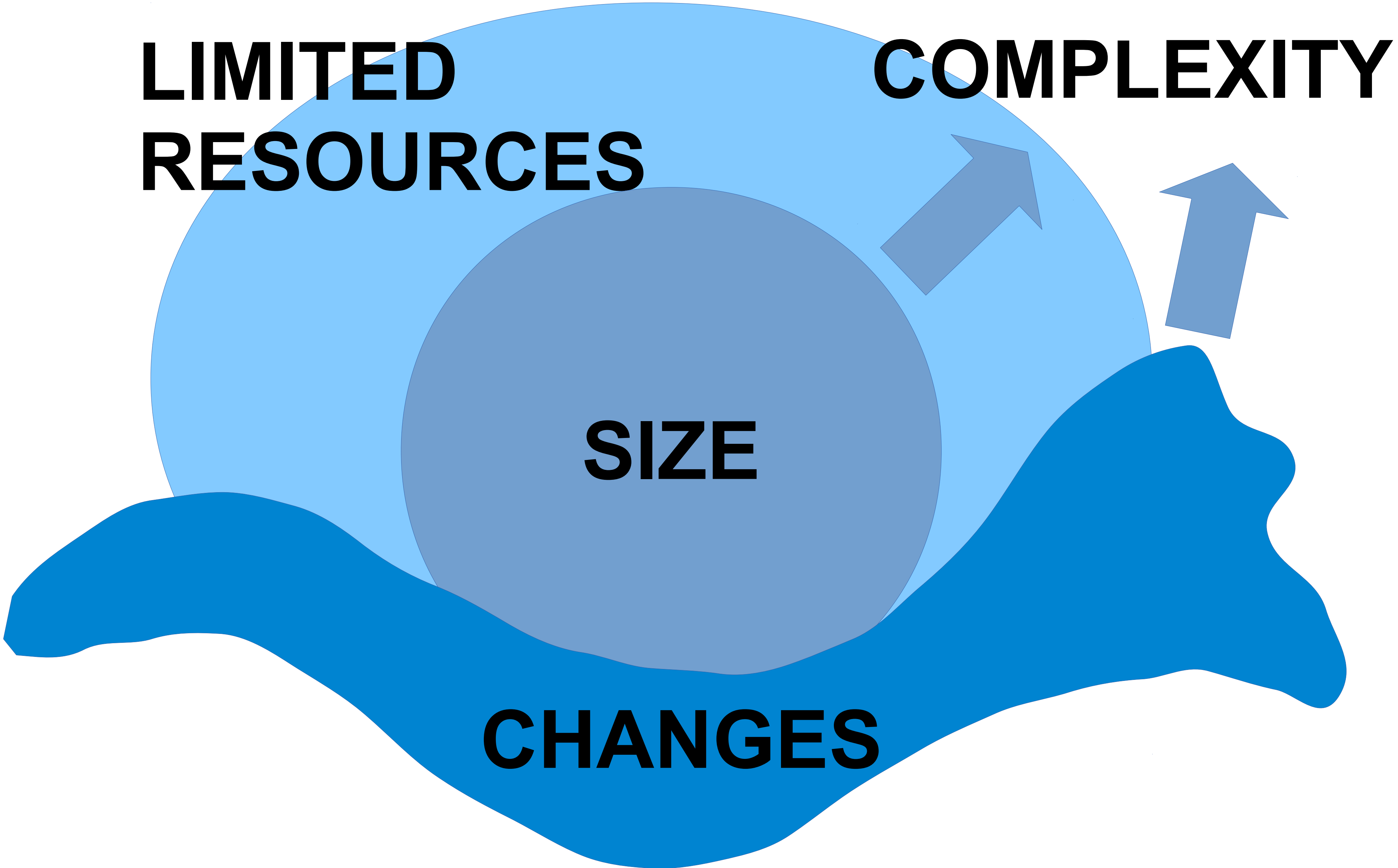


Perfect communication

- Even with perfect communication, we still have to fix the design.
  - See this by yourself, when you actually build a software project for yourself.

Change is inevitable

Challenges

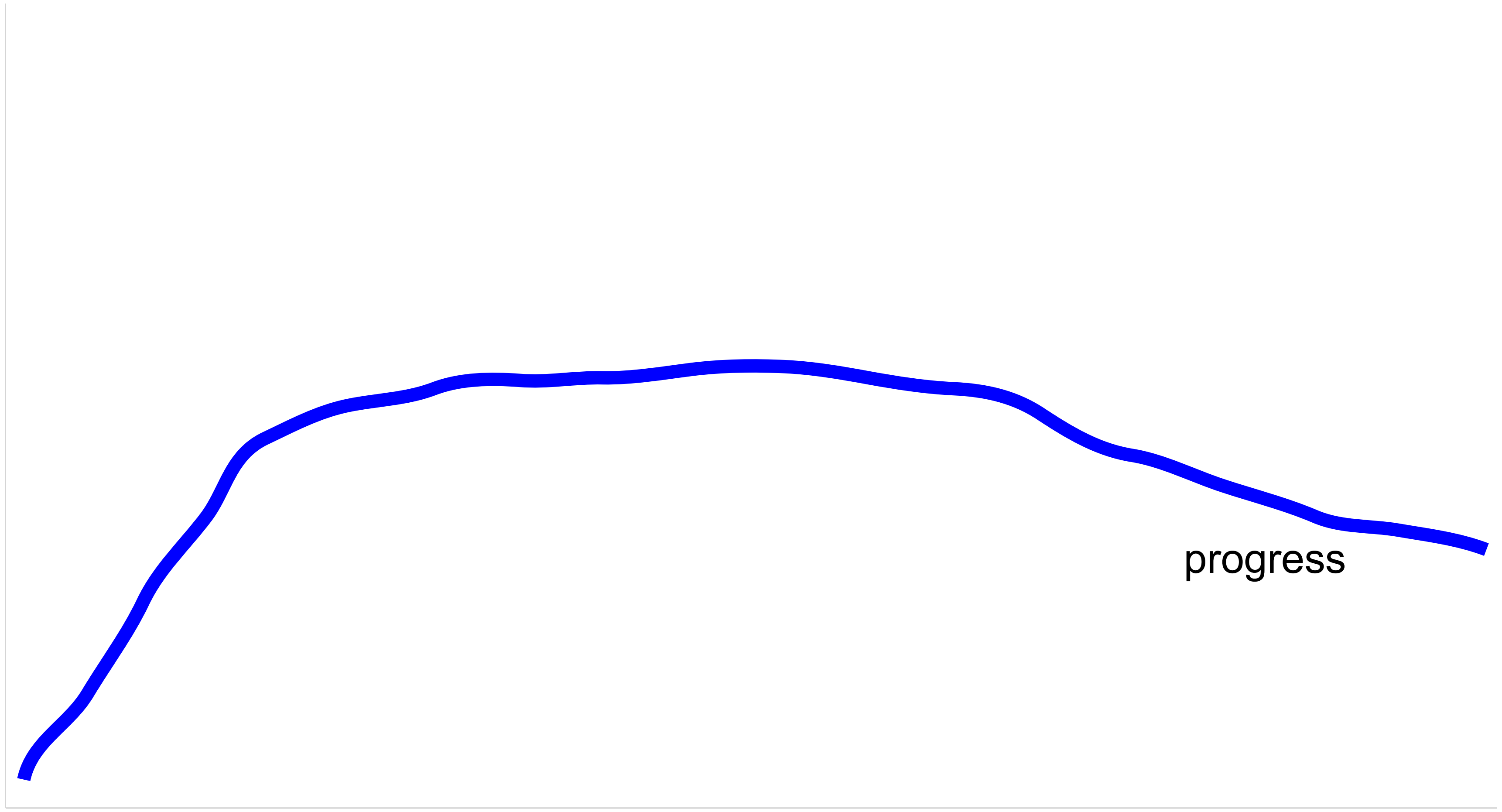


Skills

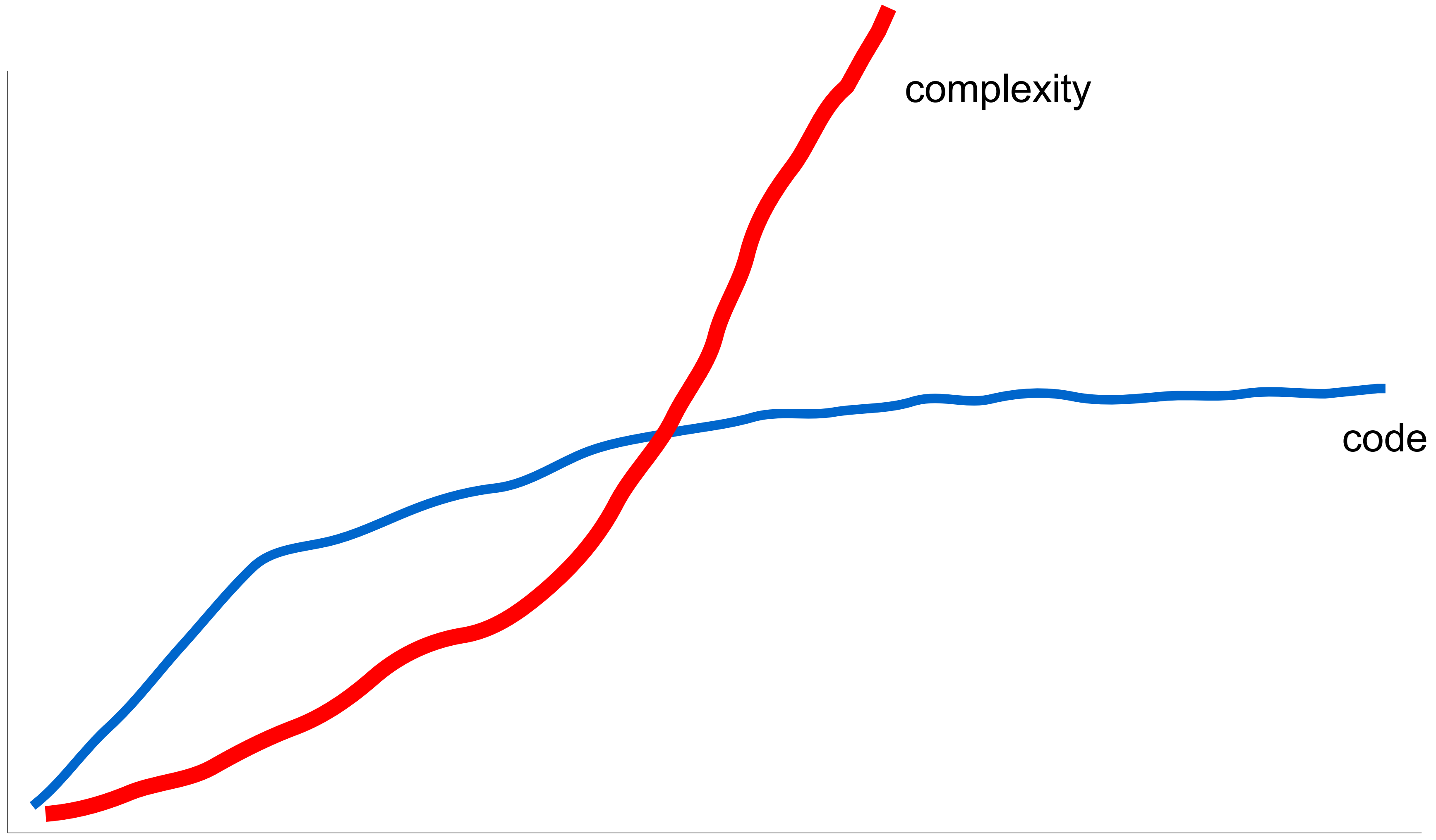
- Basic programming skills
  - Coding
  - Debugging

ONE DAY WORK

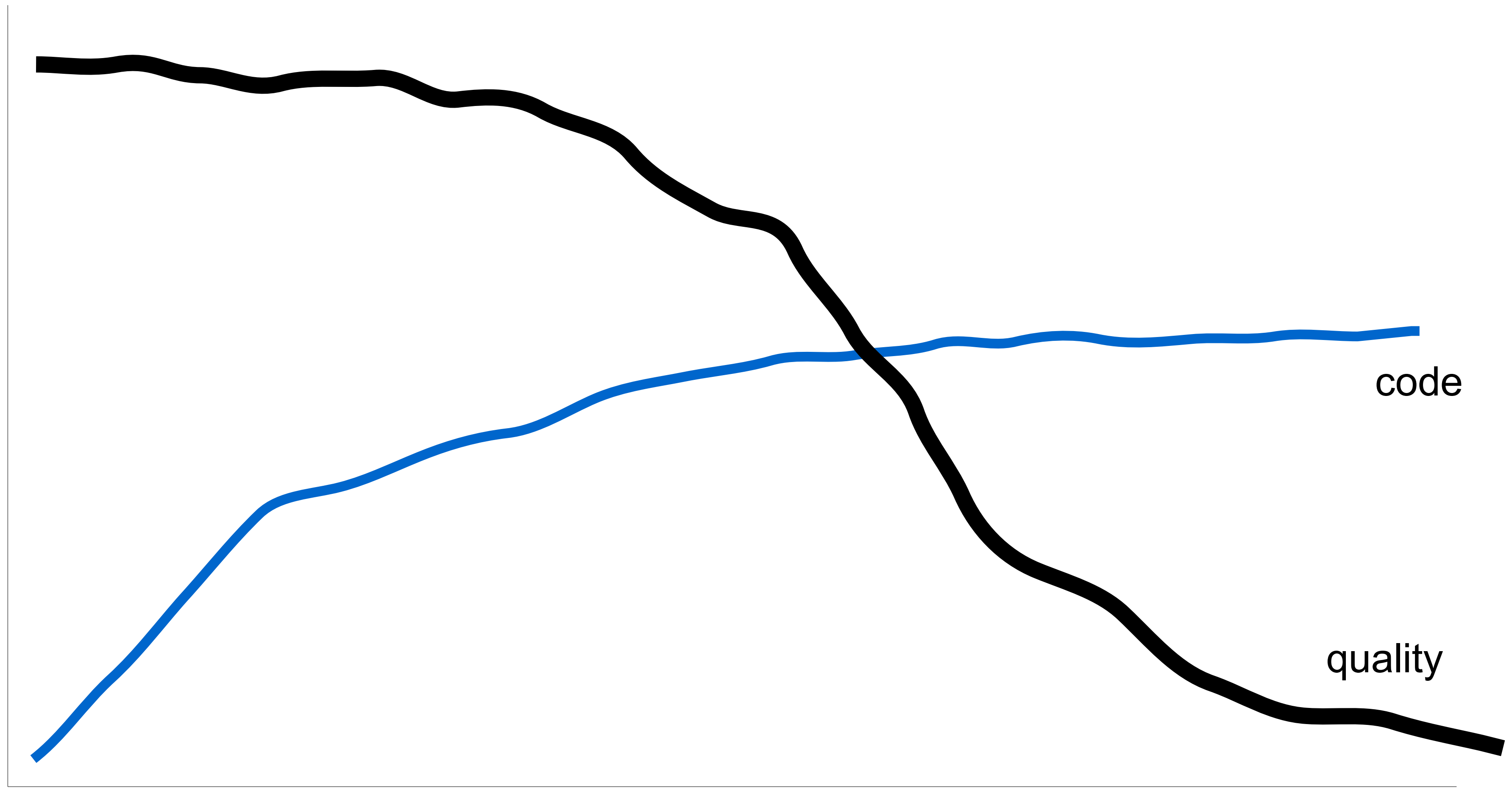
Basic skills: Progress over time



Basic skills: codes/difficulties



Basic skills: code quality rating



Skills

- Basic programming skills
  - Coding
  - Debugging

ONE DAY WORK

- How to keep making progress while
  - the software gets larger
  - the software gets more complex
  - changes are arriving

SCALING UP

Is it possible?

- Engineering practices and cultures help facebook to keep building new features while the code base grows at an increasing rate.

Skills for not-so-small software

- Project breakdown
  - so that you can develop incrementally
- Complexity reduction techniques
  - so that you understand what you have done
- Automation
  - so that you do not have to do repetitive work
- Work tracking
  - so that you understand how well you work
- Planning/Estimation
  - so that you can make commitment

How to acquire these skills

Yes!

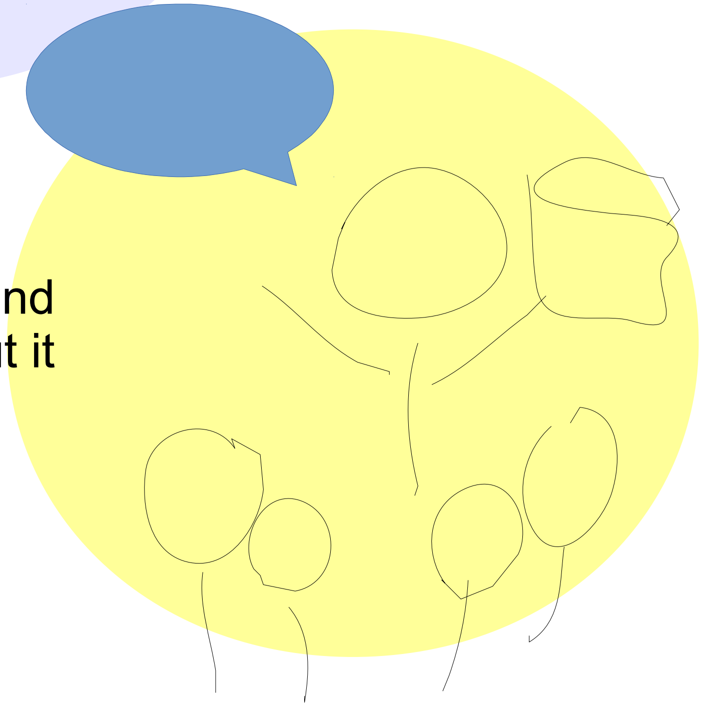
- We will have to do

quests

How to train your ~~dragon~~ self?

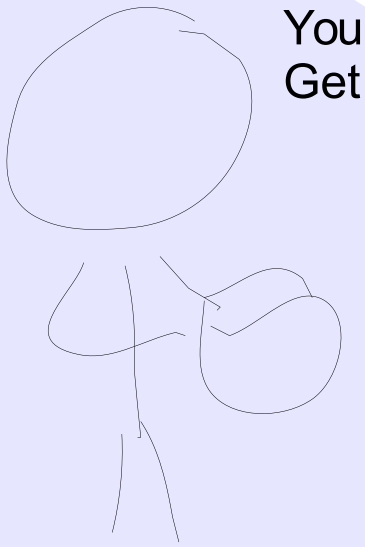
You work on your project.
Get some increment.

We plan on what to do next.

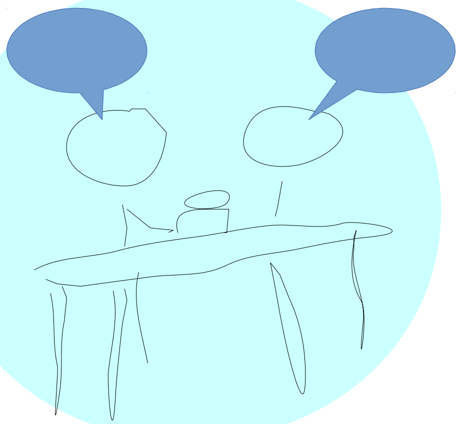


You tell your friend
about it

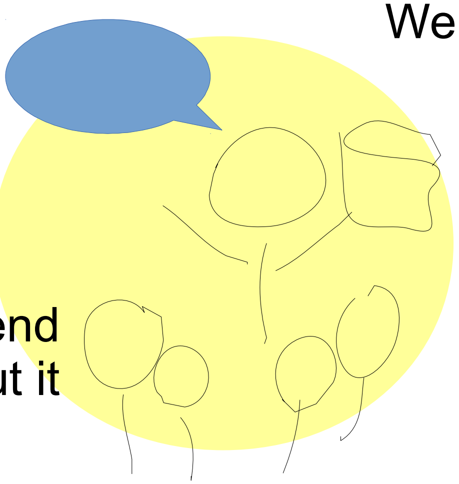
How to train your ~~dragon~~ self?



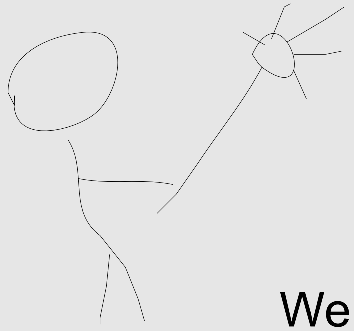
You work on your project.
Get some increment.



We plan on what to do next.



You tell your friend
about it



We will learn new techniques
that you can try to apply to your project

And do that again, and again...