Software Engineering Primer

Fighting complexity over time with peace

Yothin Muangsommuk
Data Engineer at LINE Thailand

Yothin Muangsommuk

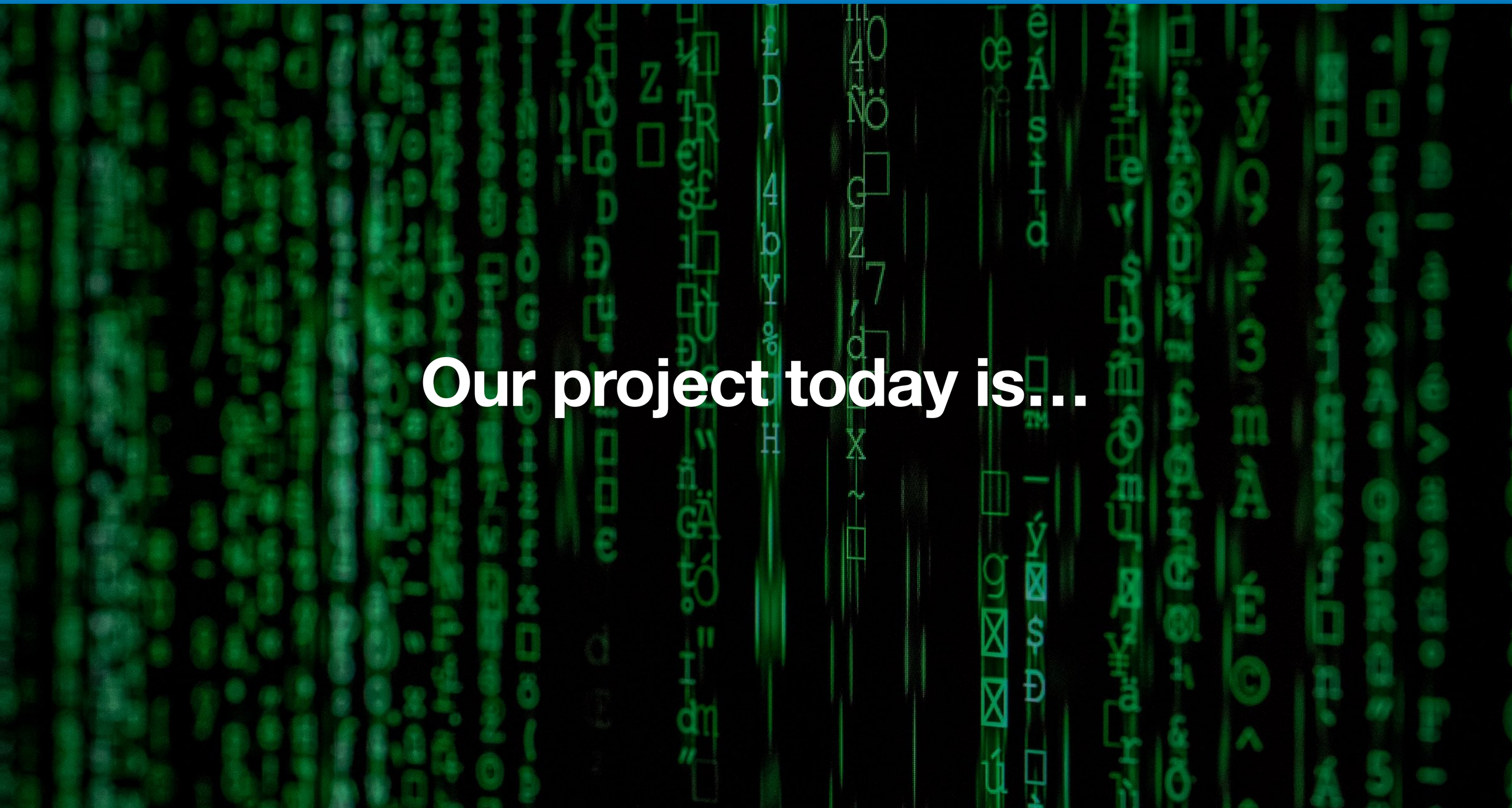
About Me

- An experience *Software Engineer, Tech Lead*, and currently *Data Engineer*
- Start Coding at 14 C++ as first programming language and be a **Pythonista** since 2010
- Falling in love with Linux and Open source software community since high school
- My focus has always been improving a sustainability and maintainability through Technical Excellence and Software Craftsmanship

Today Agenda

And we have around 3 hours together

- Build a project as our instinct told us
- Review the first implementation
- Let talk about Software Engineering and why it matter
- Understand Complexity
- Values, Principles and Practices
- Rebuild the same project with *engineering practices* in mind



Our project today is…


1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number

FizzBuzz

Believe me it can go wildly than we can imagine

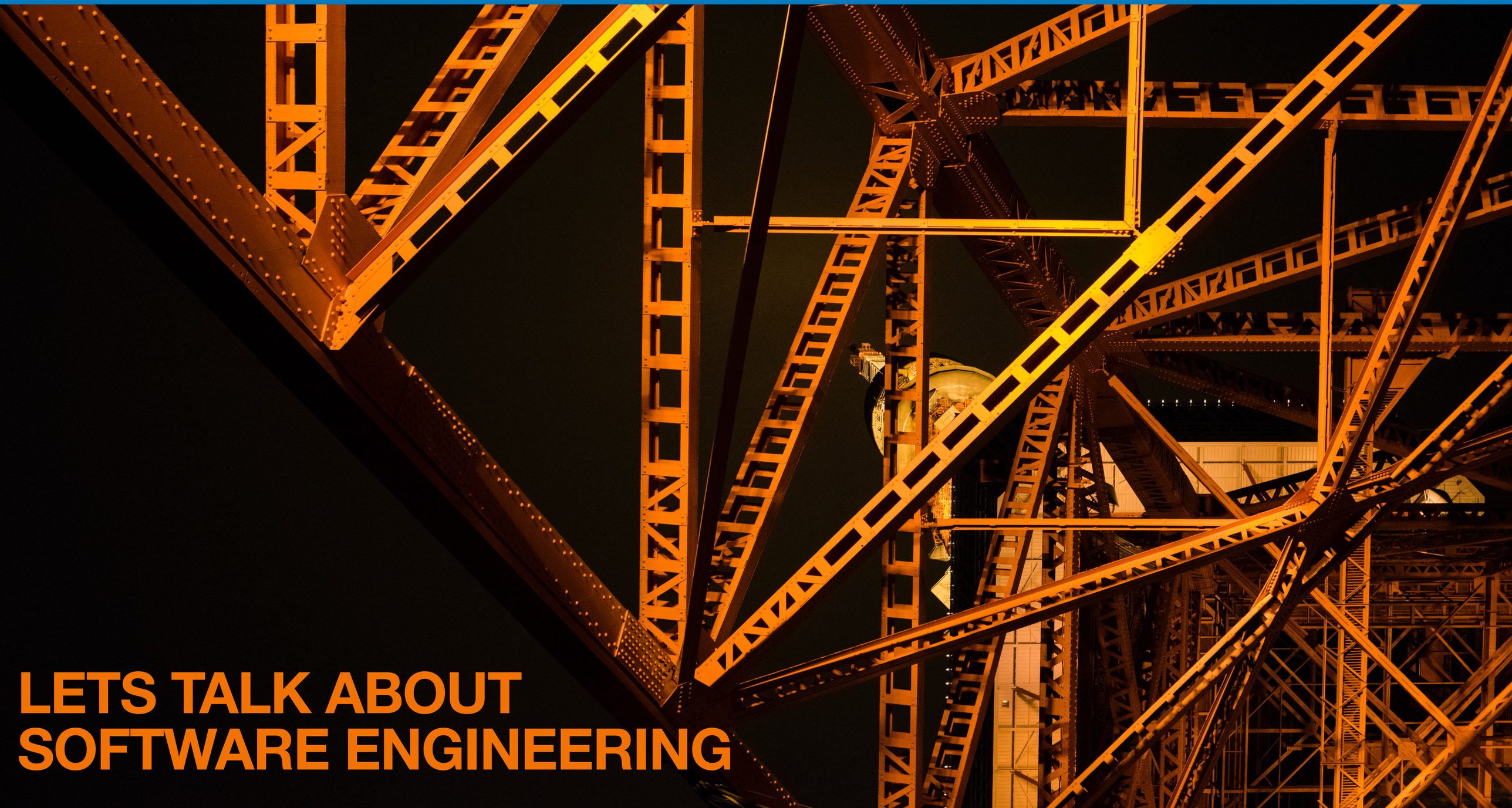
- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number
- The CEO said we can not compete with only FizzBuzz now let do FizzBuzzBang!. Where Bang is related to 7

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number
- The CEO said we can not compete with only FizzBuzz now let do FizzBuzzBang!. Where Bang is related to 7
- Our marketing team ask if we can make 35 another easter egg that return the result from 35 to 1 like the way we do with 21


Believe me now that even FizzBuzz can go wild

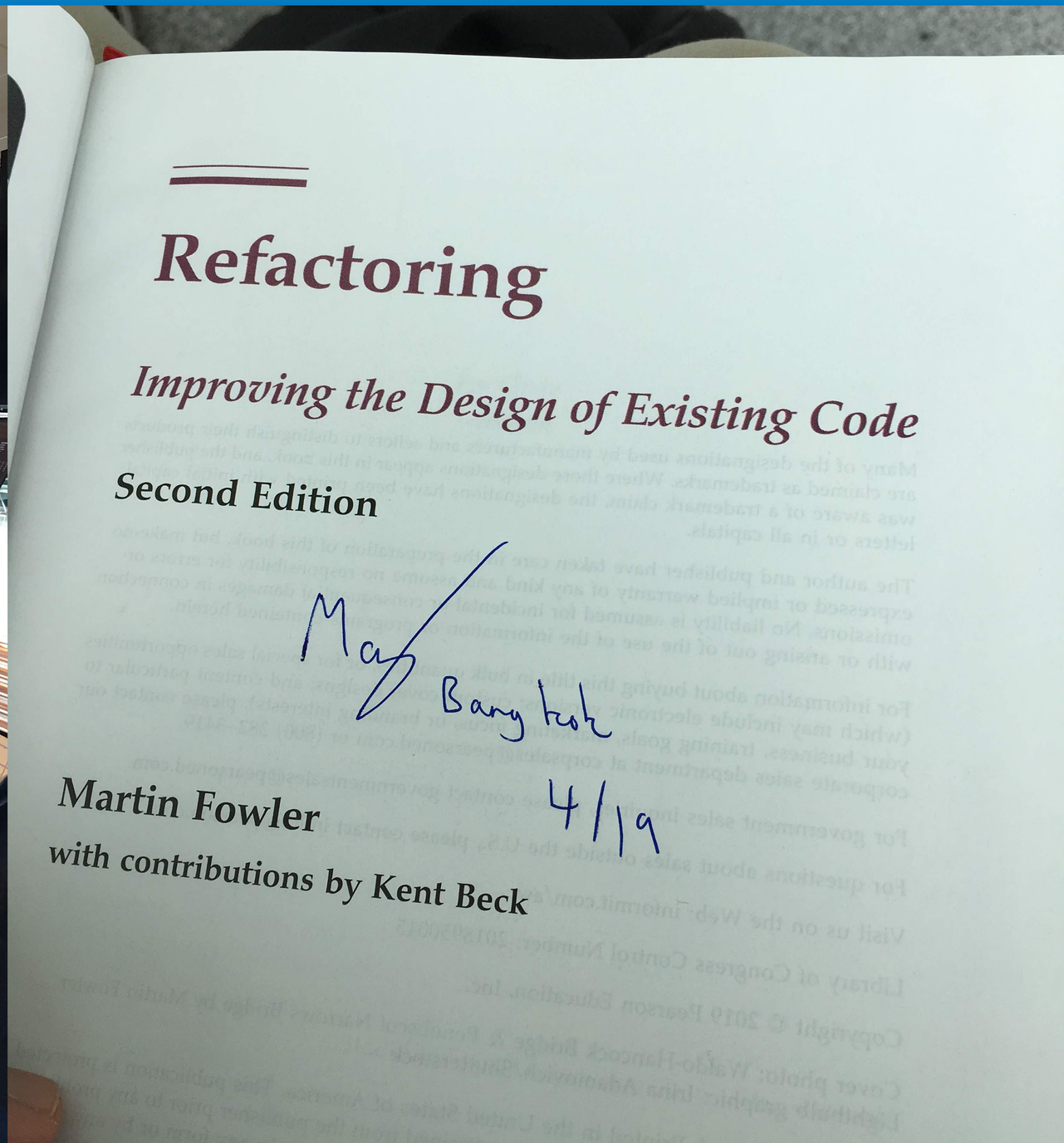


LETS TALK ABOUT SOFTWARE ENGINEERING

“Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.”

Martin Fowler


Refactoring
Improving the Design of Existing Code
Second Edition
Martin Fowler
with contributions by Kent Beck

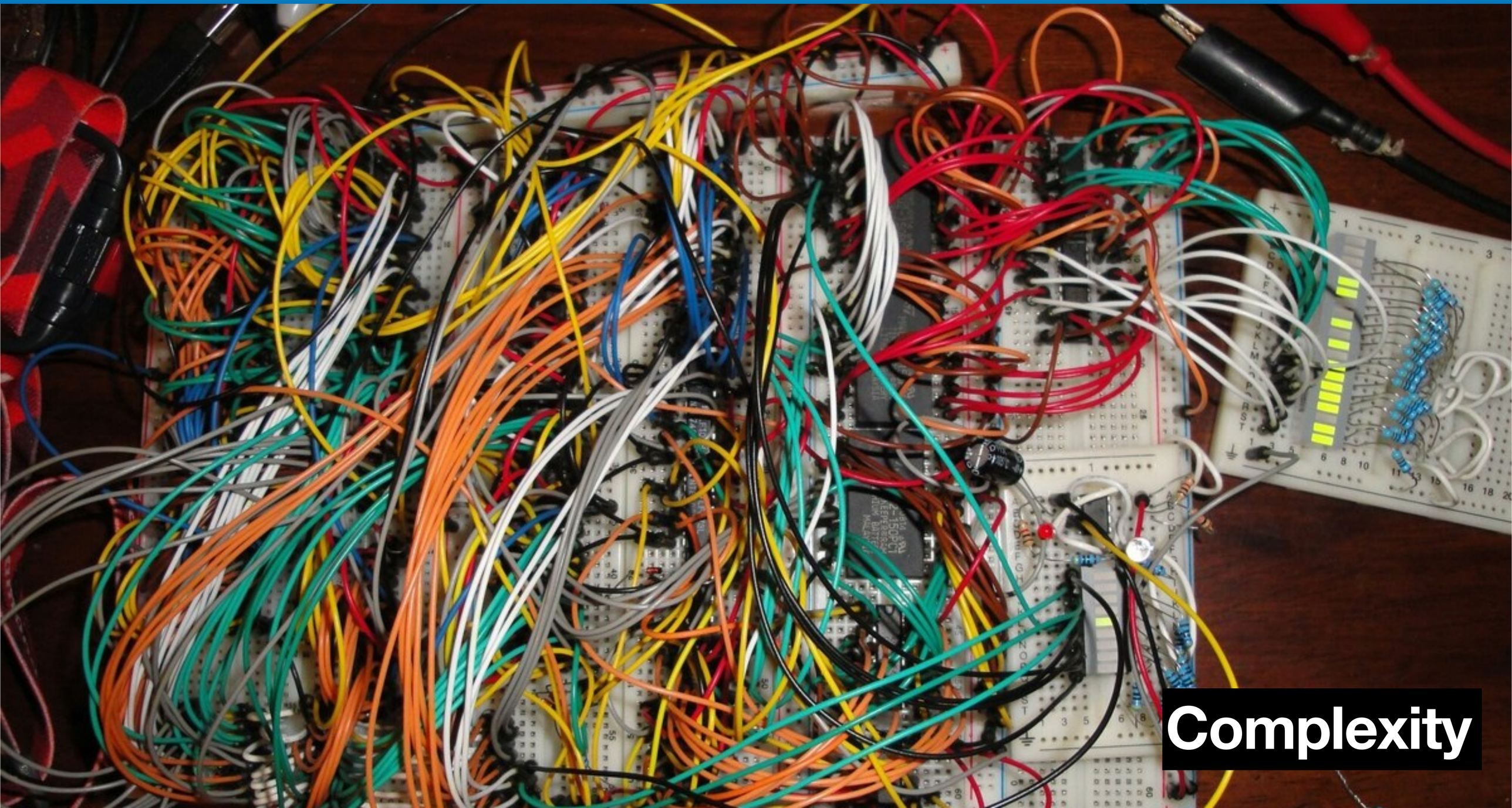
Building Software is not just solving the puzzle

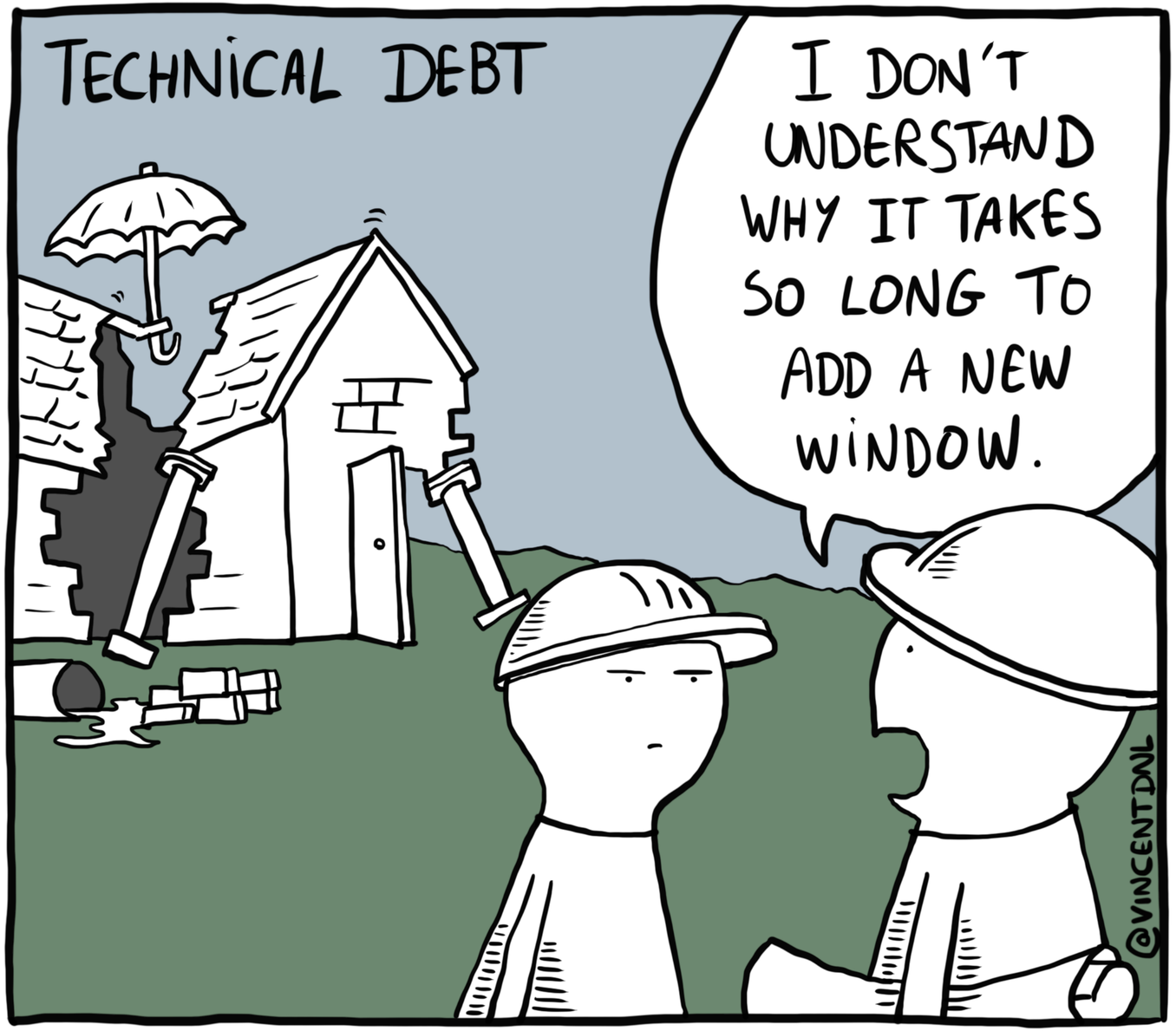

CHANNEL
UHF
VHF
ON
OFF
LO-TECH

It about maintaining it the same way as we growth plant


When we talk about
Software Engineering we
added time as factor

Software alway have enhancement and extension, without it mean no one using it

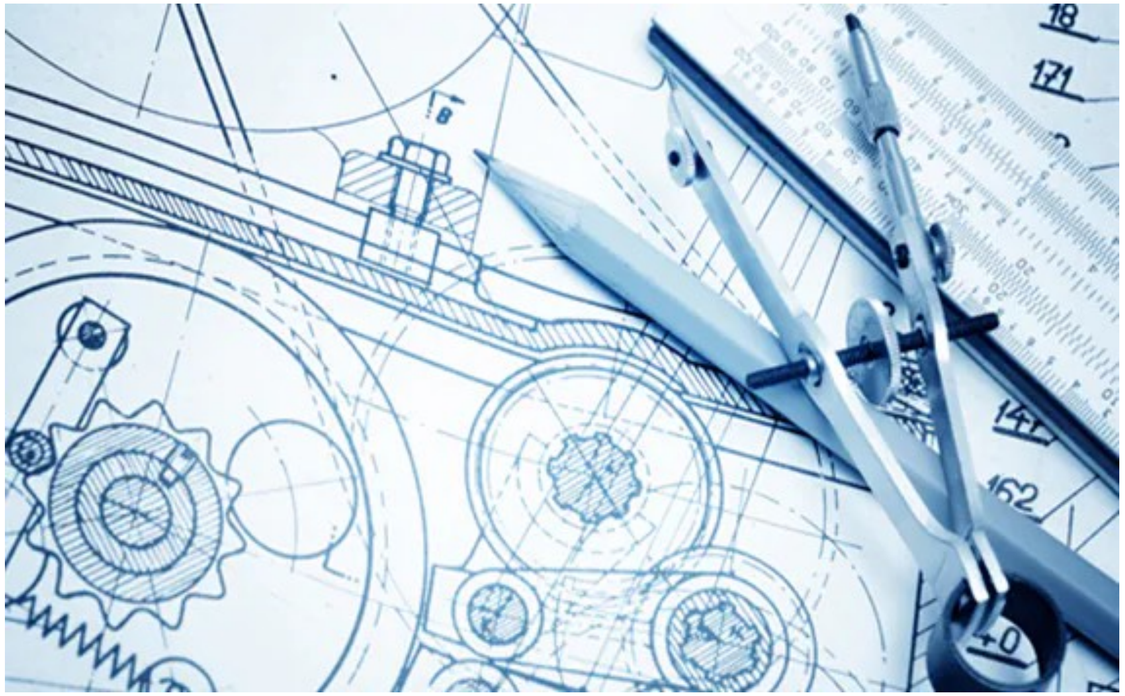

Complexity


TECHNICAL DEBT
I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.
@VINCENTDNL

Software Engineering

Where engineering discipline meet computer science

- Focusing on **fighting complexity over time** when building a software by Embrace **Engineering discipline**, Producing a good design, and Keep remind us out of harmful path
- Today we focus on **Software Design** and **Engineering Practices**
- There are also other topics that not covered today like Development Process, Risk Management, Software Modeling, etc…

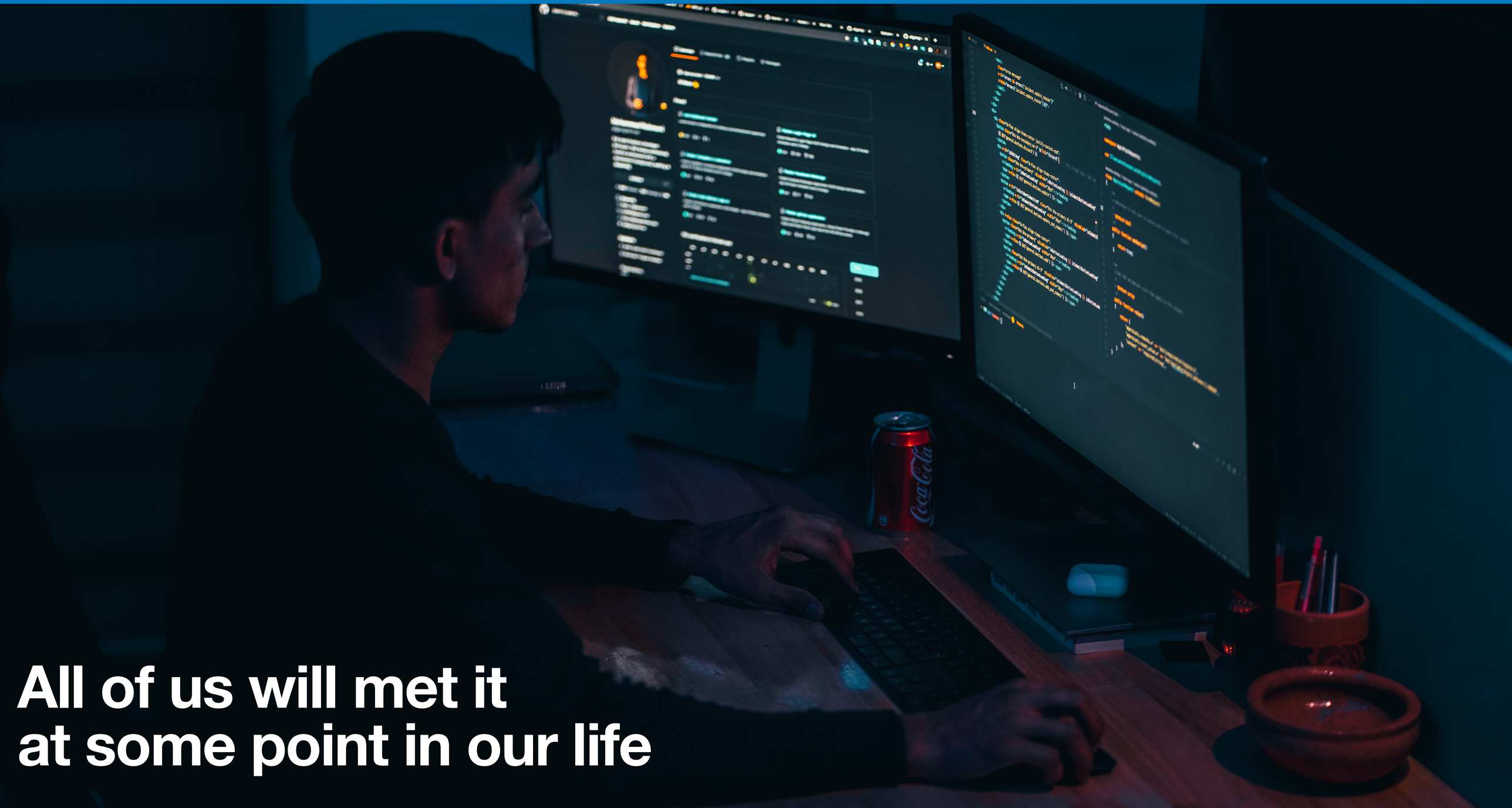


Understand Complexity

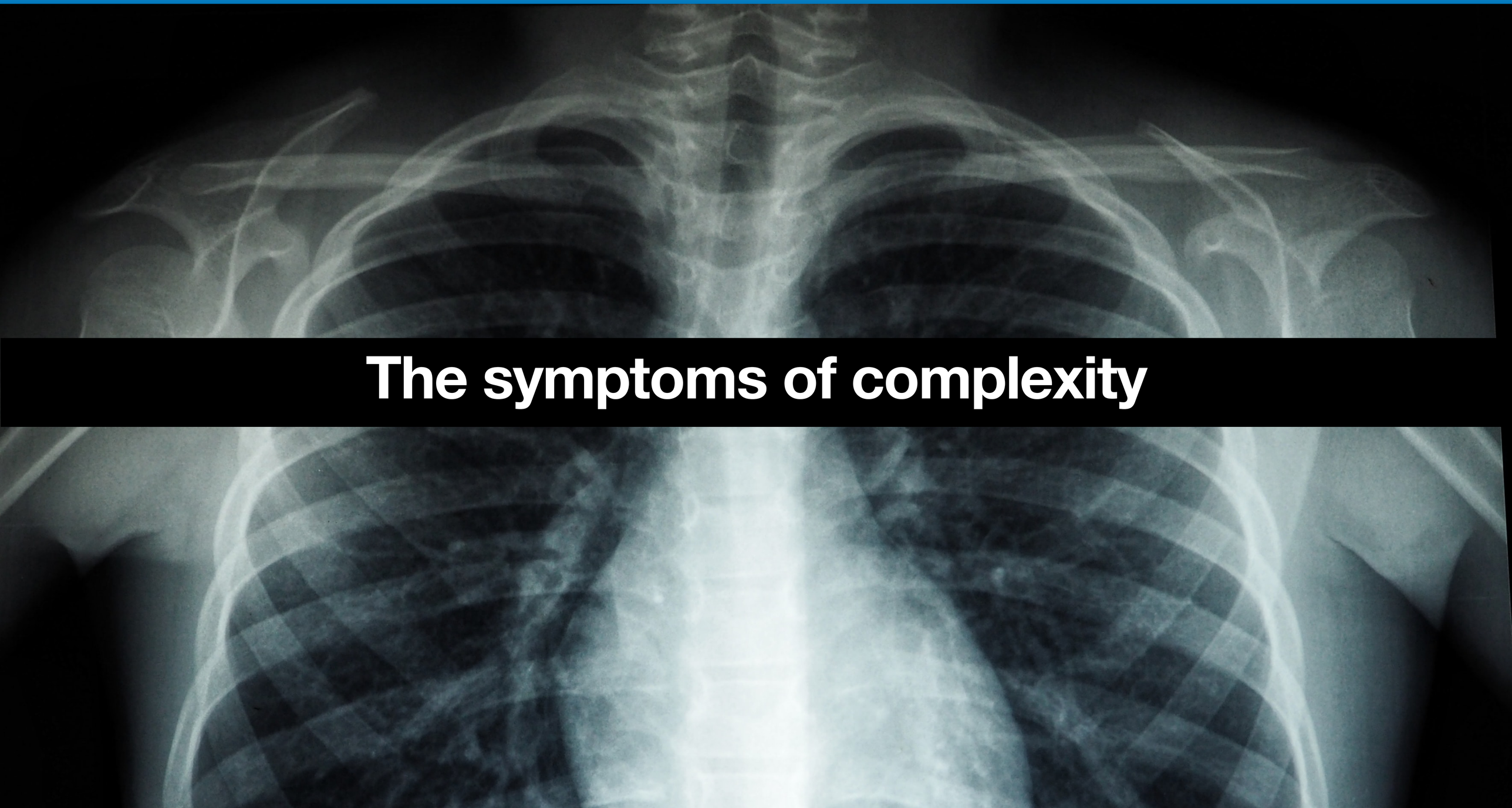
```
        if (!Strings.isNullOrEmpty(deviceId)) {
            ResponseEntity<AcmeServiceResponse<CustomerProfileStatus>> response = customersExpService
                    .checkDeviceNameStatus(deviceId);
            if (response != null && response.getStatusCode() == HttpStatus.OK) {
                AcmeServiceResponse<CustomerProfileStatus> Response = response.getBody();
                String crmId = Response != null ? Response.getData().getCrmId() : null;
                if (!Strings.isNullOrEmpty(crmId)) {
                    CustomerProfileResponseData customerProfileResponseData = custProfileService
                            .processCustProfile(crmId);
                    if (customerProfileResponseData != null) {
                        if (!Strings.isNullOrEmpty(Response.getData().getDeviceStatus())
                                && Response.getData().getDeviceStatus().equalsIgnoreCase("1")) {
                            if (!Strings.isNullOrEmpty(Response.getData().getEbCustomerStatusId())
                                    && !Strings.isNullOrEmpty(Response.getData().geAcmeUserStatusId())
                                    && Response.getData().getEbCustomerStatusId().equalsIgnoreCase("02")
                                    && Response.getData().geAcmeUserStatusId().equalsIgnoreCase("02")) {
                                customerProfileResponseData
                                        .setDeviceNickname(Response.getData().getDeviceNickname());
                                customerProfileResponseData.setCustomerStatus("Active");
                                ServiceResponse.setStatus(new AcmeStatus(ResponseCode.SUCESS.getCode(),
                                        ResponseCode.SUCESS.getMessage(), ResponseCode.SUCESS.getService(),
                                        ResponseCode.SUCESS.getDesc()));
                                ServiceResponse.setData(customerProfileResponseData);
                                return ResponseEntity.ok().headers(responseHeaders).body(ServiceResponse);

                            } else if (!Strings.isNullOrEmpty(Response.getData().getEbCustomerStatusId())
                                    && !Strings.isNullOrEmpty(Response.getData().geAcmeUserStatusId())
                                    && Response.getData().getEbCustomerStatusId().equalsIgnoreCase("02")
                                    && Response.getData().geAcmeUserStatusId().equalsIgnoreCase("05")) {
                                customerProfileResponseData
                                        .setDeviceNickname(Response.getData().getDeviceNickname());
                                customerProfileResponseData.setCustomerStatus("AccountLocked");
                                ServiceResponse.setStatus(new AcmeStatus(ResponseCode.ACCESS_PIN_LOCK.getCode(),
                                        ResponseCode.ACCESS_PIN_LOCK.getMessage(),
                                        ResponseCode.ACCESS_PIN_LOCK.getService(),
                                        ResponseCode.ACCESS_PIN_LOCK.getDesc()));
                                ServiceResponse.setData(customerProfileResponseData);
                                return ResponseEntity.ok().headers(responseHeaders).body(ServiceResponse);

                            } else {
                                customerProfileResponseData
                                        .setDeviceNickname(Response.getData().getDeviceNickname());
                                customerProfileResponseData.setCustomerStatus("Inactive");
                                ServiceResponse.setStatus(new
AcmeStatus(ResponseCode.INACTIVE_CUSTOMER.getCode(),
                                        ResponseCode.INACTIVE_CUSTOMER.getMessage(),
```



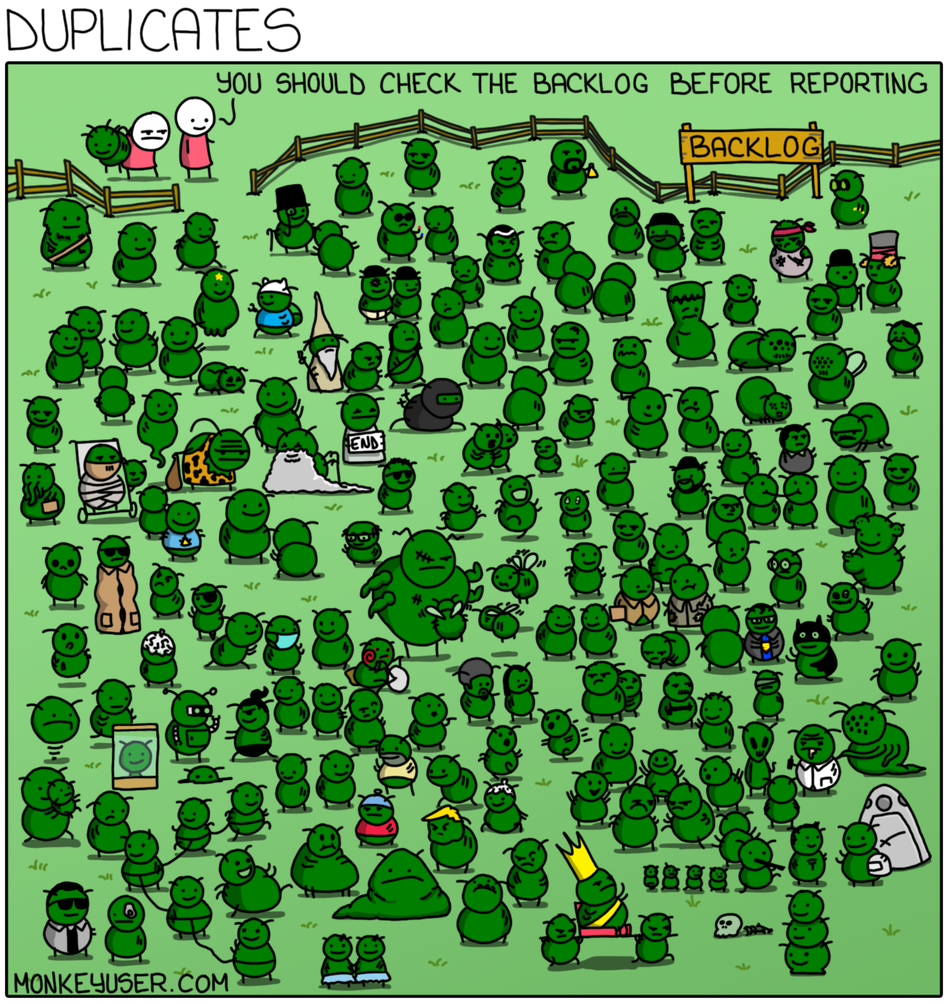
All of us will met it at some point in our life



The symptoms of complexity

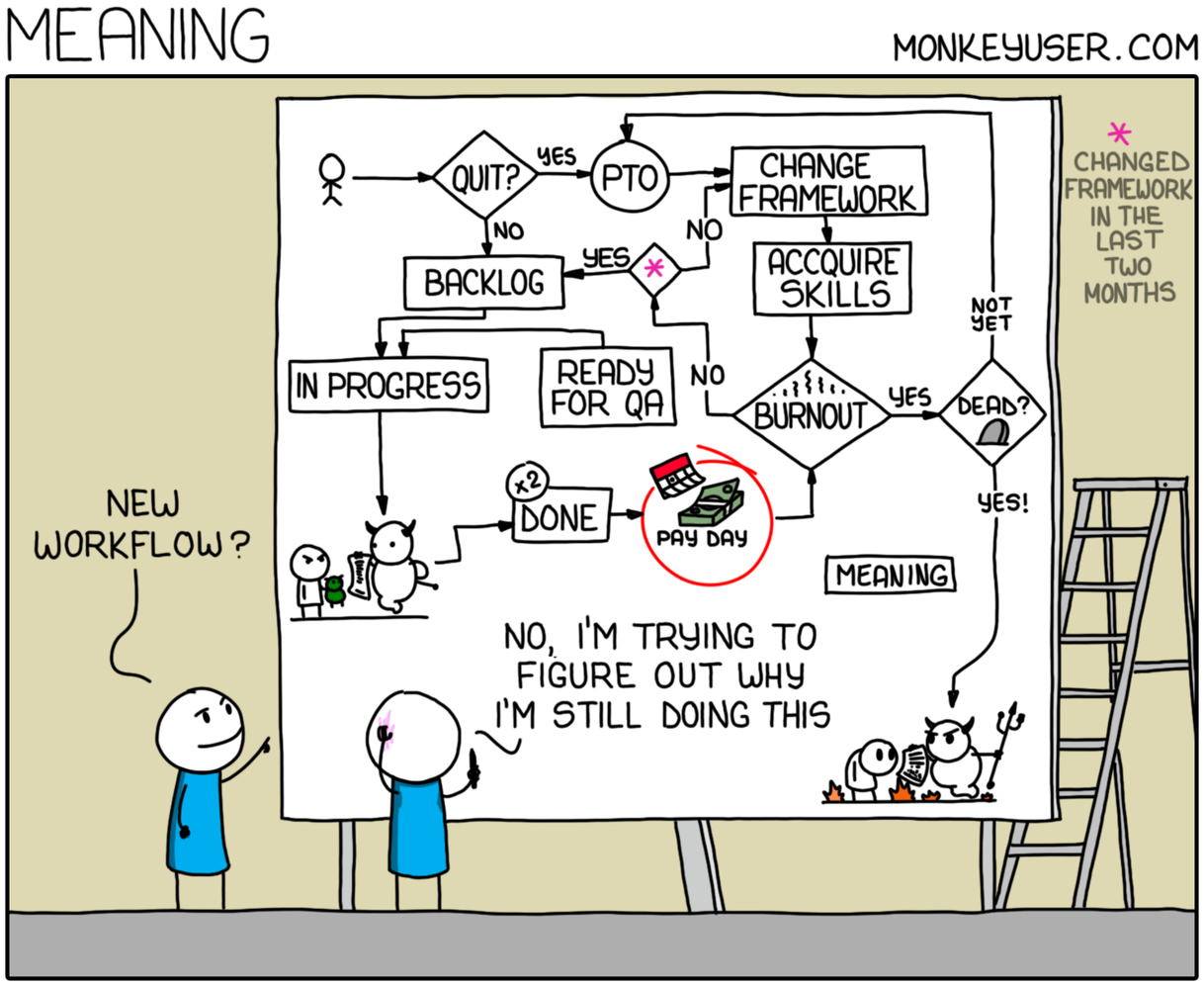
Change Amplification

Simple code require change in many places



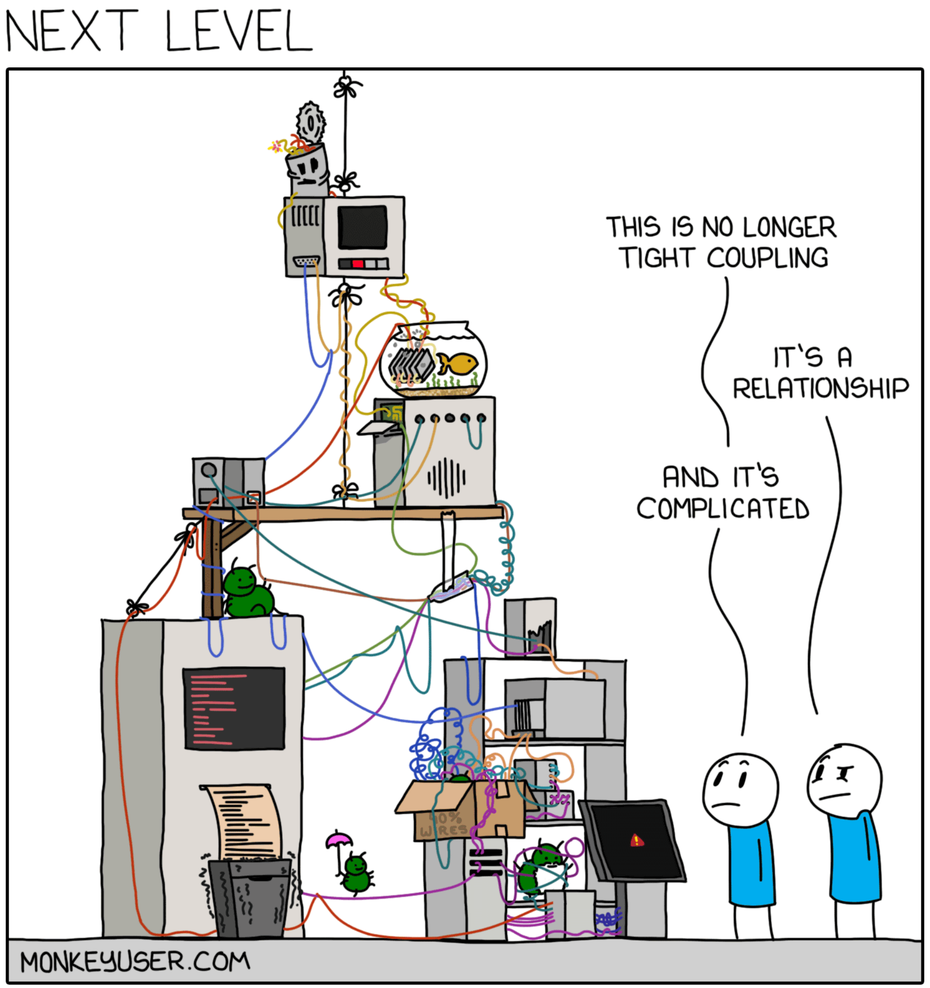
Cognitive Load

Need to know a lot of thing in order to complete a task



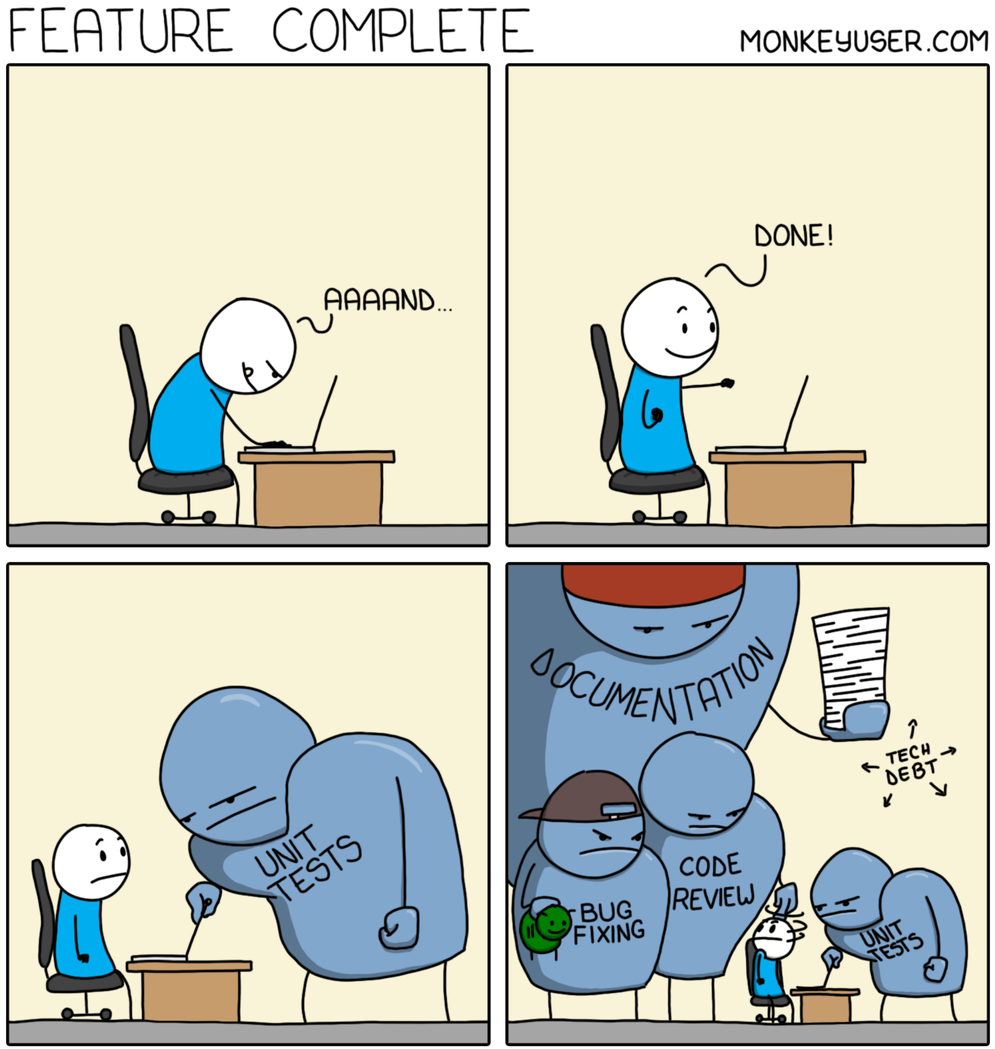
Unknown unknowns

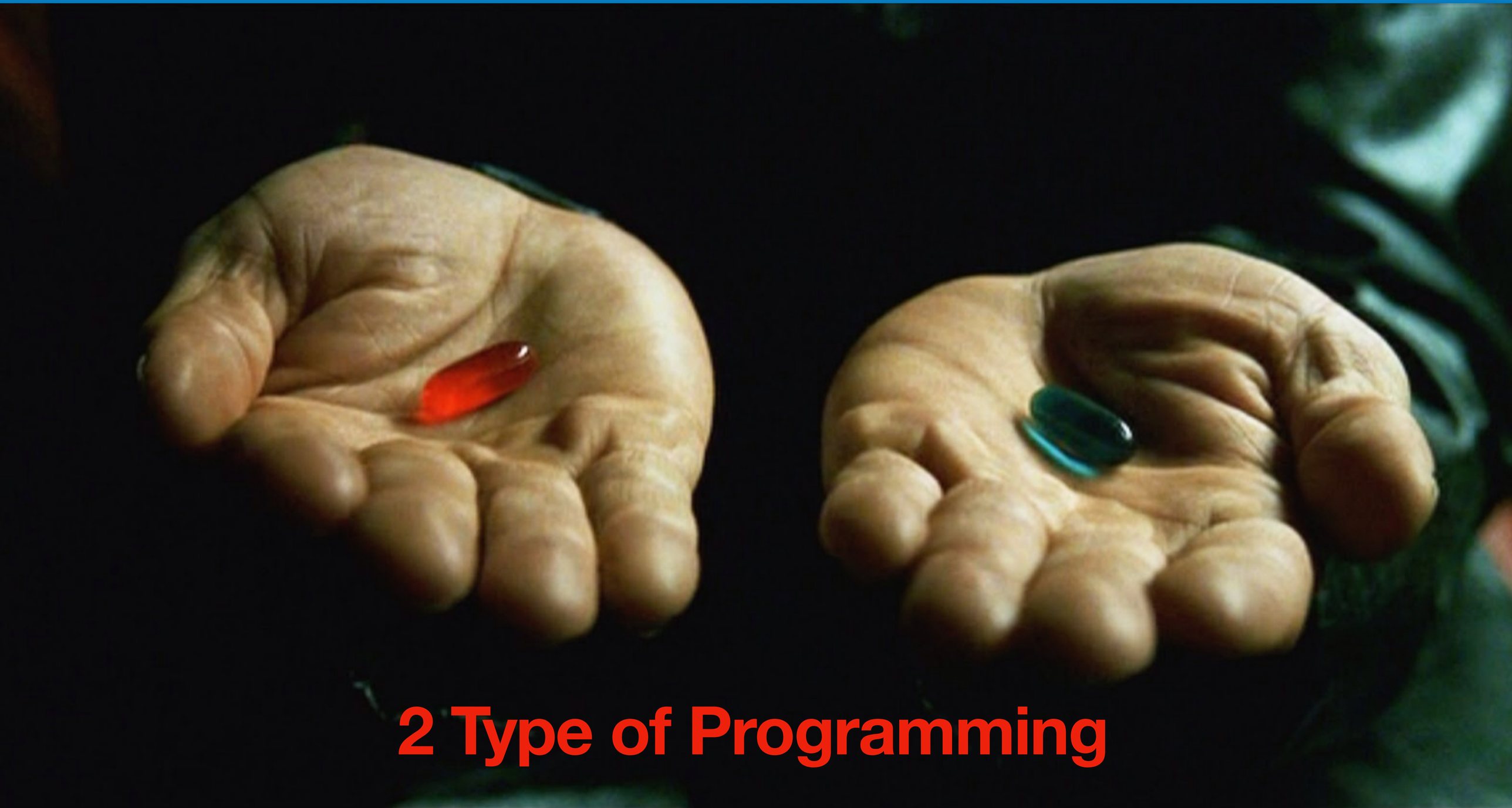
I don’t even know where to start to do this feature



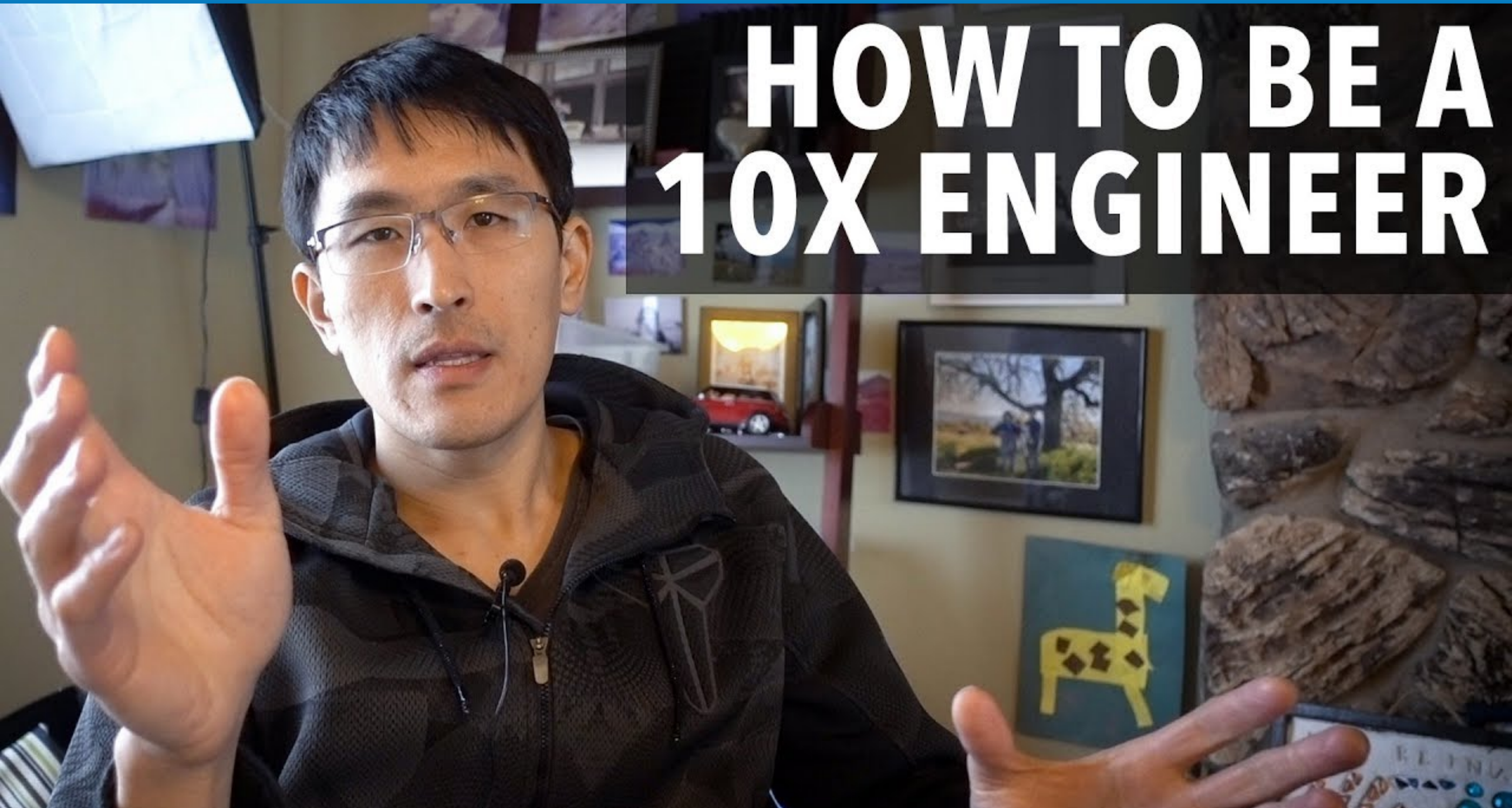
Working code is not enough

That is the root of complexity creep

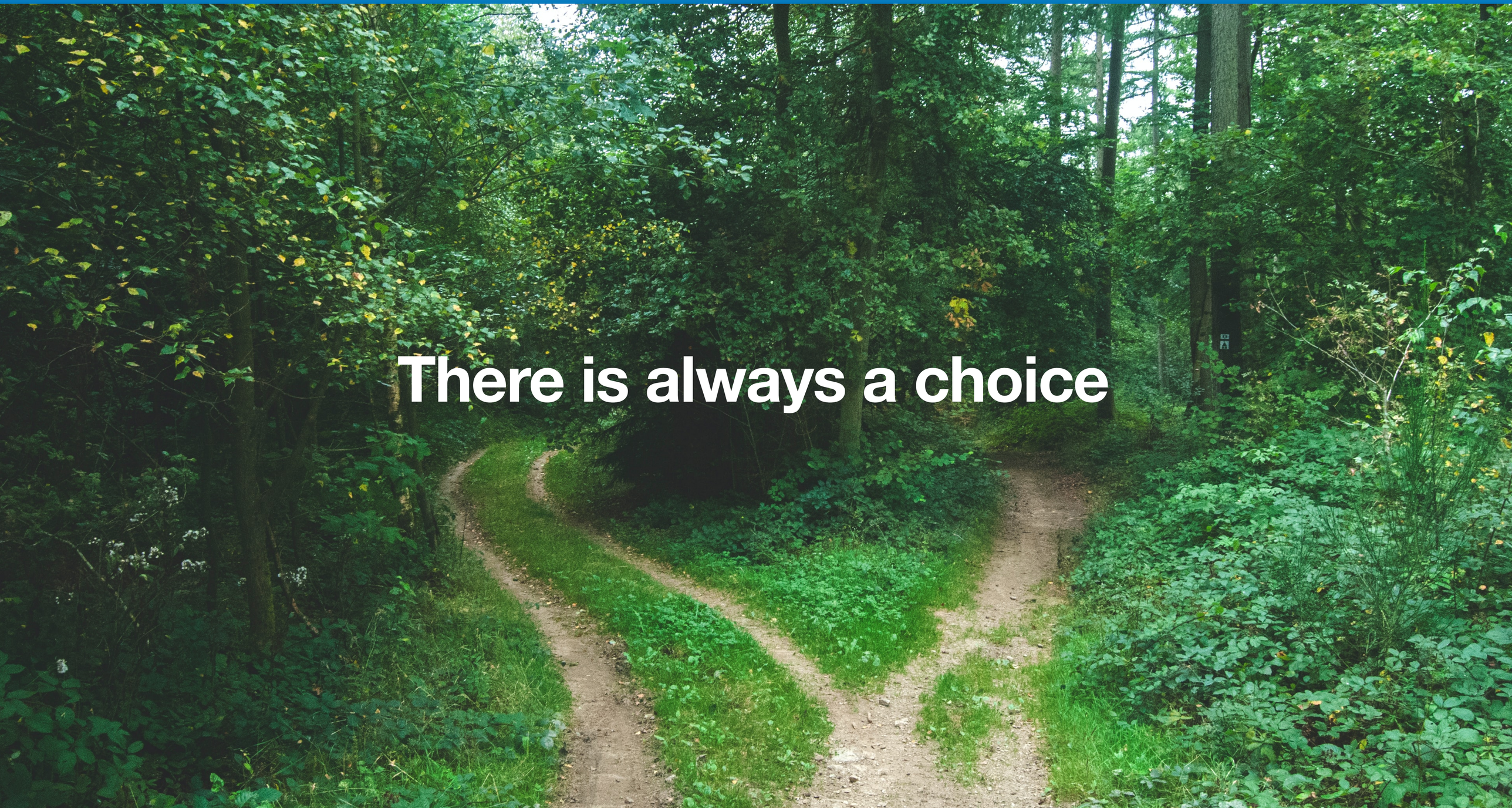



2 Type of Programming

Tactical Programming


HOW TO BE A
10X ENGINEER

Strategic Programming

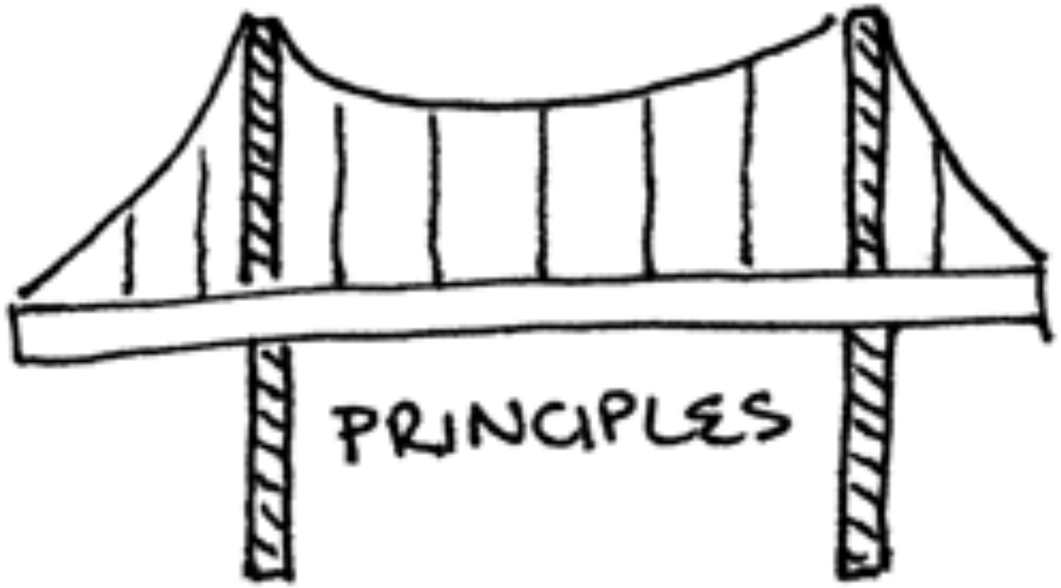

There is always a choice

Simple and Obvious

1 + 1 = 2

Encapsulate the detail via interface

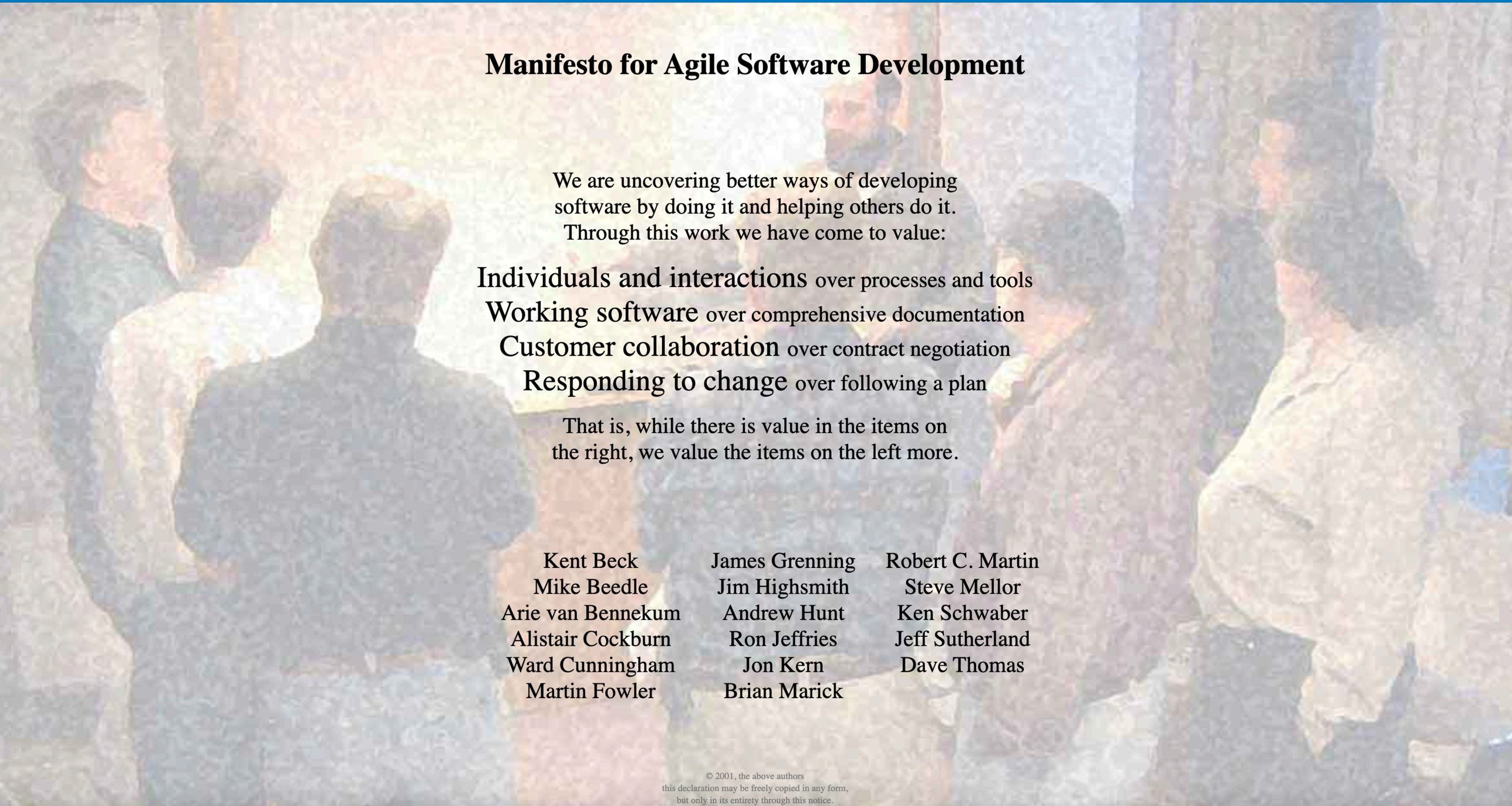
VALUES



PRACTICES


KANTOOR


package com.ds.ucd.be.becore.solr;
import ...
public final class LocationUtils {
/**
* Parses Point from it's String representation.
* @param locationString - String that represents location, as 2 double values split with coma. Accepts space after/before
* @return org.springframework.data.solr.core.geo.Point instance
*/
public static Point parseLocation(String locationString) {
Preconditions.checkNotNull(locationString, errorMessage: "Location String should not be null");
Preconditions.checkArgument(locationString.contains(","), errorMessage: "Location must be split with coma");
locationString = locationString.trim();
if (locationString.contains(" ,")) {
locationString = locationString.replaceAll(regex: " ,", replacement: ",");
}
if (locationString.contains(", ")) {
locationString = locationString.replaceAll(regex: ", ", replacement: ",");
}
String[] location = locationString.split(regex: ",");
Preconditions.checkArgument(expression: location.length >= 2, errorMessage: "Location should constist at least 2 Double parameters");
double lat = Double.parseDouble(location[0]);
double lon = Double.parseDouble(location[1]);
return new Point(lat, lon);
}
}



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Manifesto for Software Craftsmanship

Raising the bar.

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.

Manifesto for Software Craftsmanship

Raising the bar.

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:

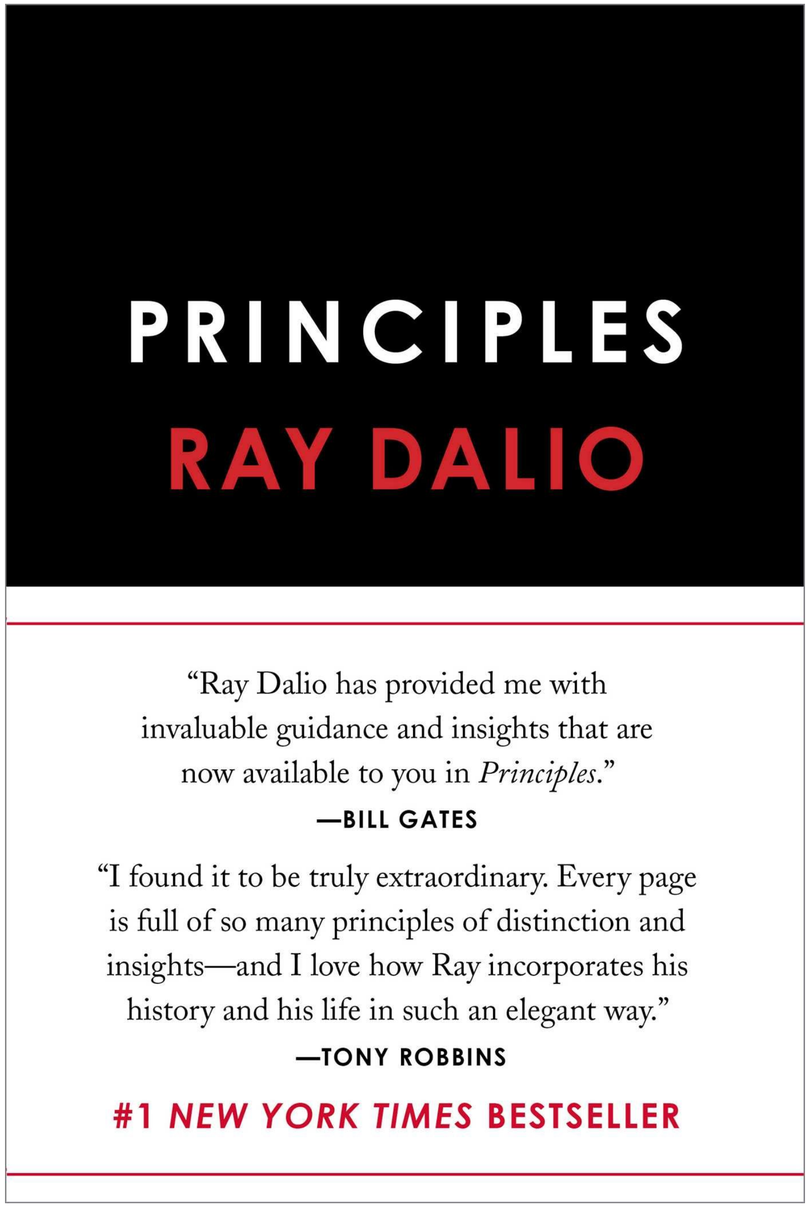
Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

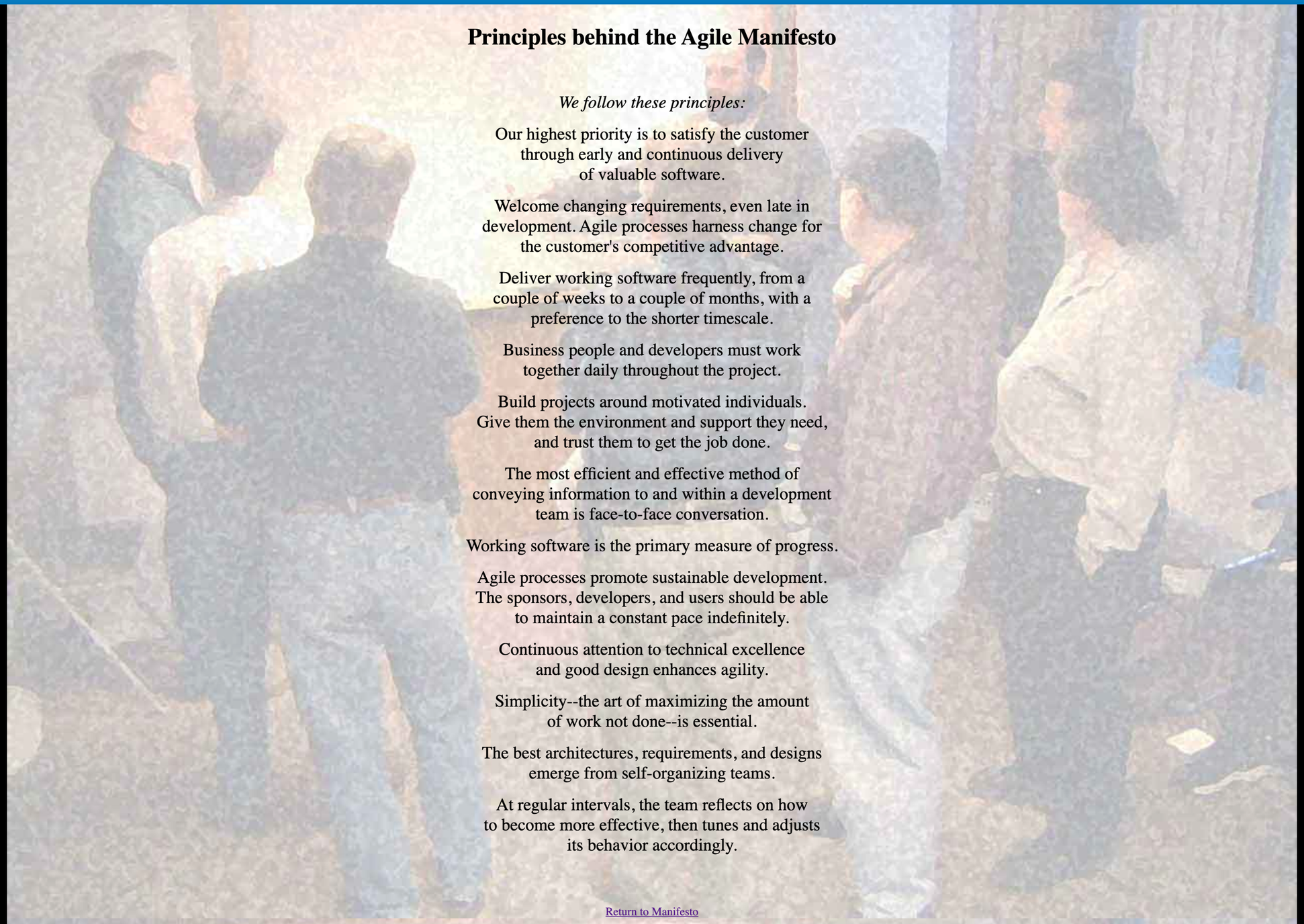
Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.


PRINCIPLES
RAY DALIO
"Ray Dalio has provided me with invaluable guidance and insights that are now available to you in *Principles*."
—BILL GATES
"I found it to be truly extraordinary. Every page is full of so many principles of distinction and insights—and I love how Ray incorporates his history and his life in such an elegant way."
—TONY ROBBINS
#1 NEW YORK TIMES BESTSELLER

“Principles are the domain specific guidelines for life”

Kent Beck



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.

Return to Manifesto

The Zen of Python

19 guiding principles That influence the design of Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one—and preferably only one—obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than right now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea—let's do more of those!

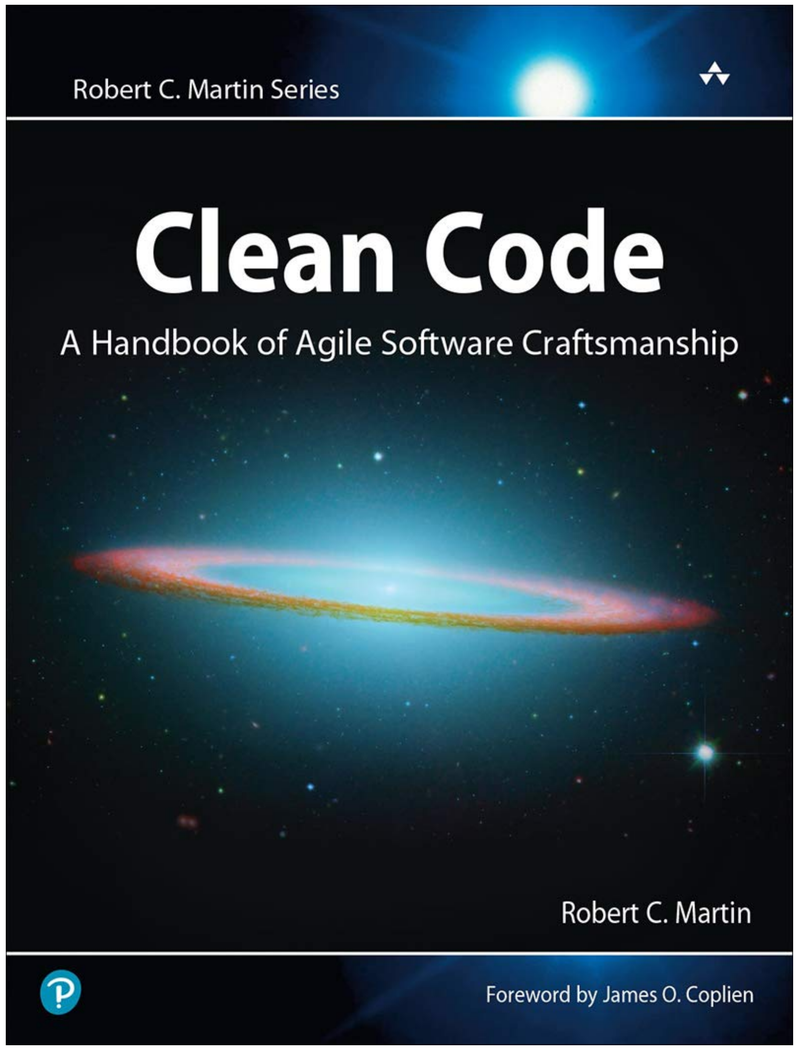
Practices

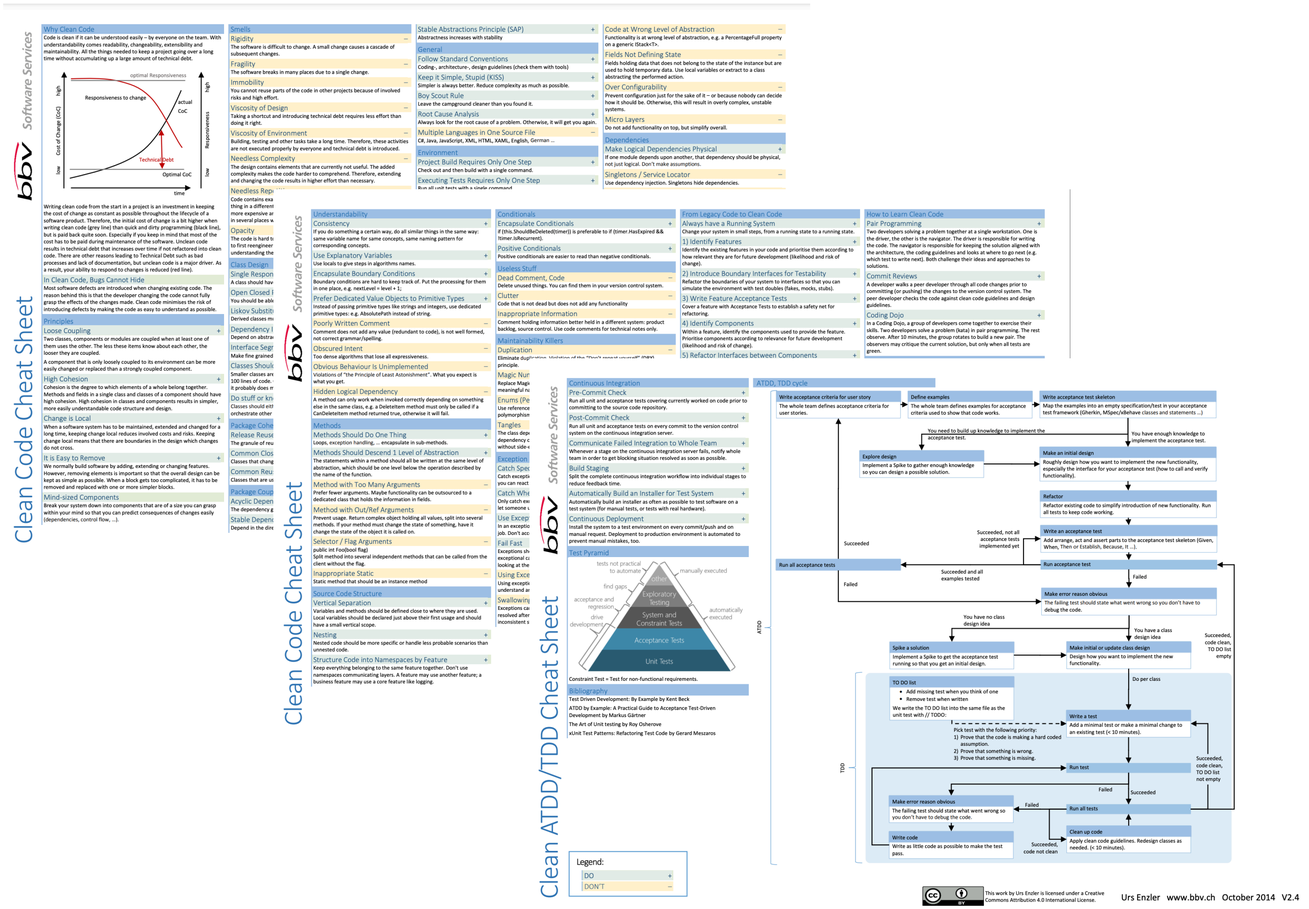
An actionable that bring accountability to value


FIRST
BDD
Single Responsibility
Boy Scout Rule
Rule of Three
TDD
Act Locally
Loose Coupling
DRY
Don't Reinventing the wheel
High Cohesion
KISS
YAGNI
Law of Demeter
SBE
No Silver bullet
Continuous Integration
Self Documented Code
Fail Fast
OCP
Catch specific Exceptions
Interface Segregation
Take small step
SoC
Inversion of Control
Reuse before use
Composition over Inheritance

Clean Code

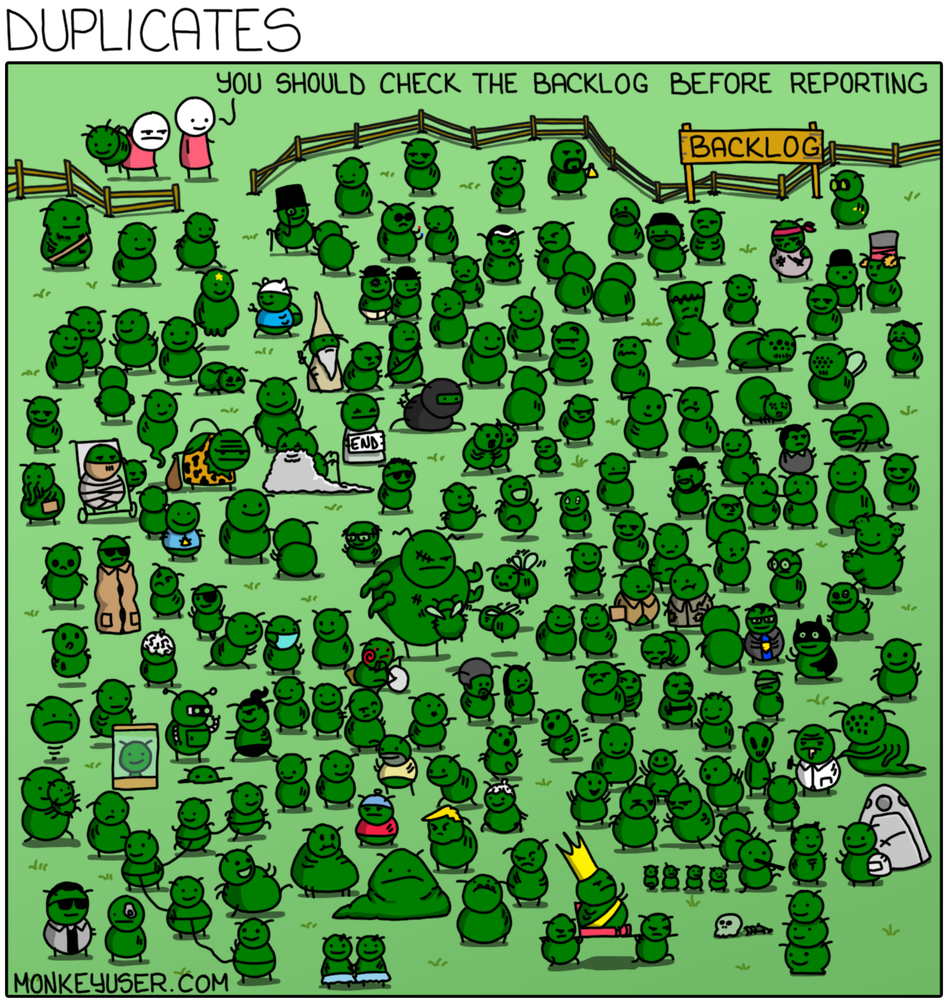
A classic set of principle & practice in software development





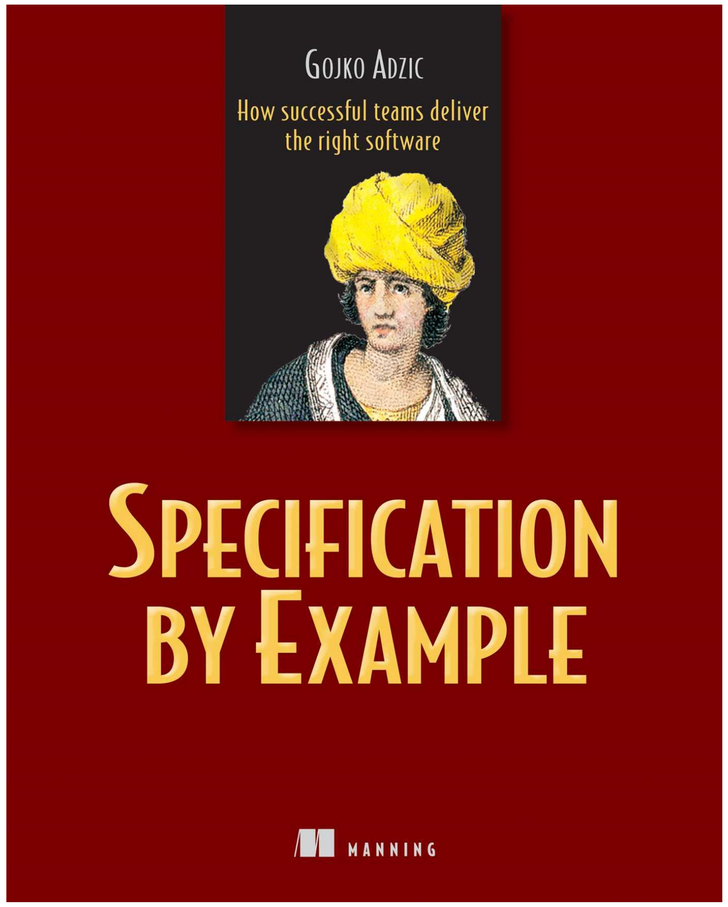
Don’t Repeat Yourself (DRY)

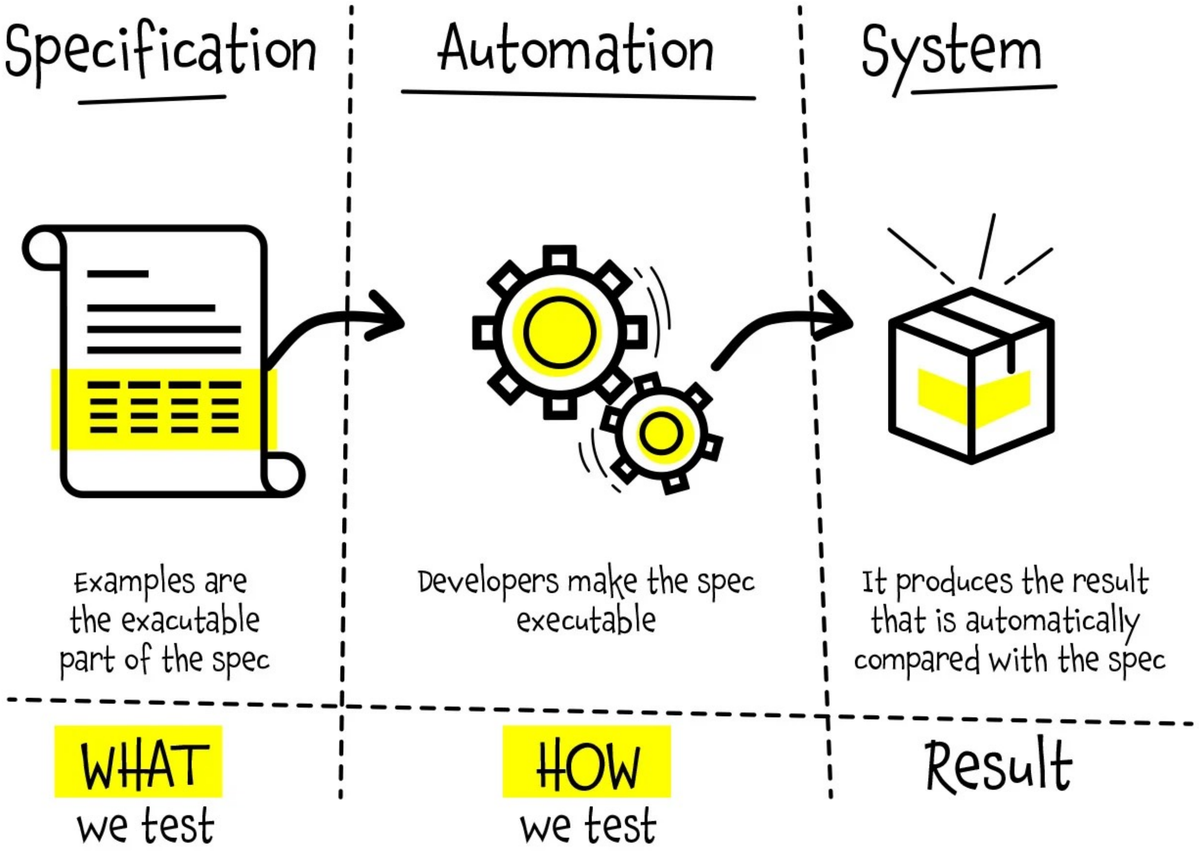
Against all duplication which is the root of complexity



Specification by Example

When you know the answer before you went to exam

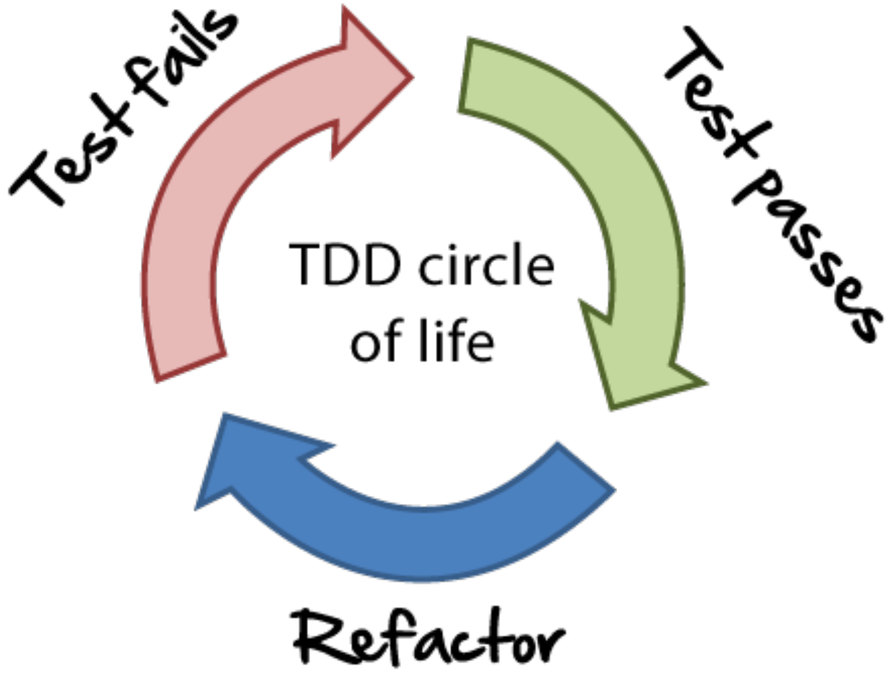


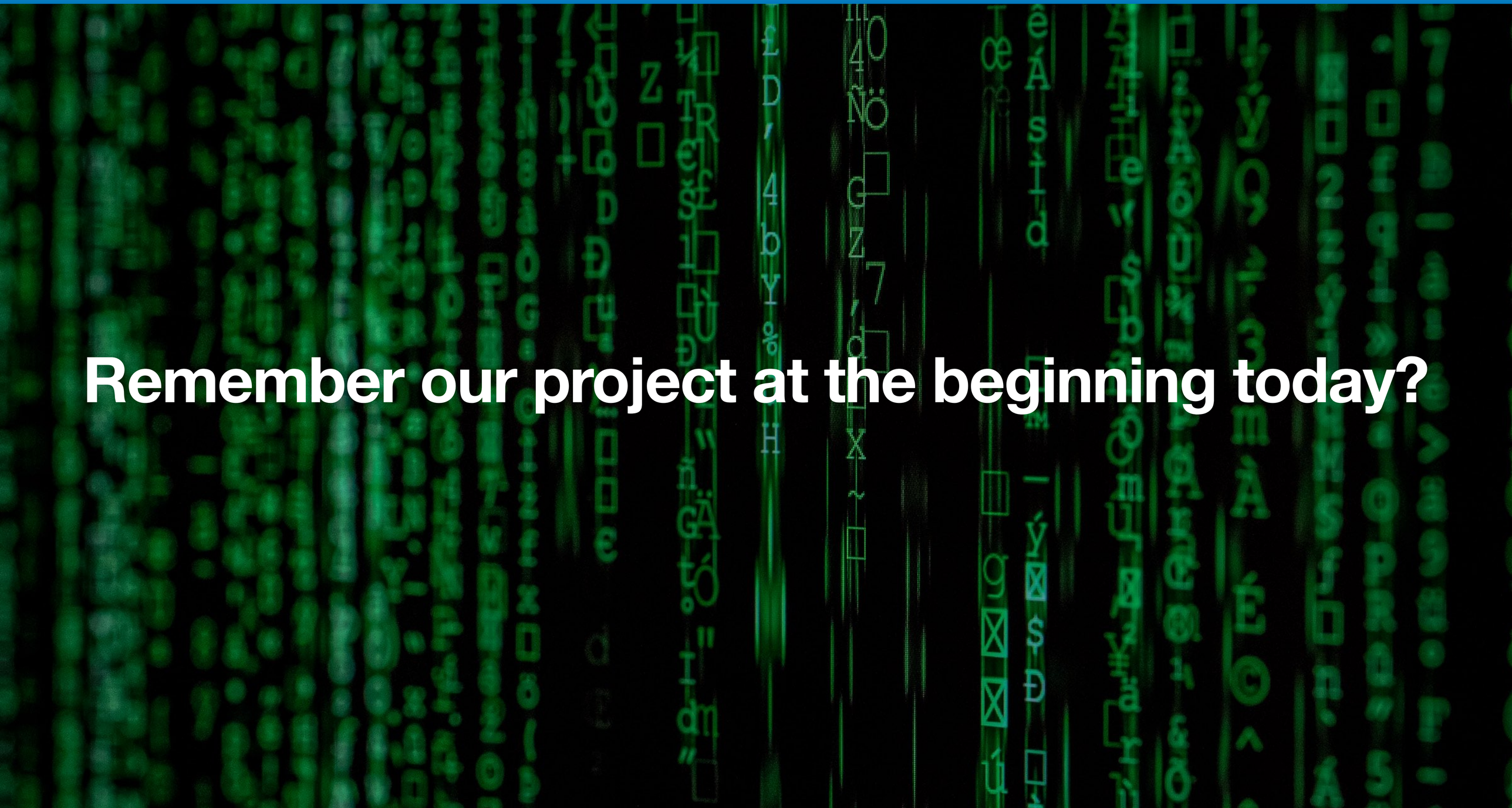


Test-Driven Development

One small step for code, One giant leap for architecture

- Originate from Extreme Programming way back in 90s
- Divide to 3 simple step:
  - Adding test and execute the test to **failed** from current program
  - Adding a simplest code to make the test **pass**
  - **Refactor** for future extendable and the test still pass
- Actually a design process not just testing





Remember our project at the beginning today?


1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number
- The CEO said we can not compete with only FizzBuzz now let do FizzBuzzBang!. Where Bang is related to 7

FizzBuzz

Believe me it can go wildly than we can imagine

- We want a program that accept arbitrary number and return the result of FizzBuzz things
- Our marketing team ask if we could make 21 as our easter egg that return all the result from 1 to 21 instead of just the entered number
- The CEO said we can not compete with only FizzBuzz now let do FizzBuzzBang!. Where Bang is related to 7
- Our marketing team ask if we can make 35 another easter egg that return the result from 35 to 1 like the way we do with 21

Let recap what we learn today

I know its a lot that why I create this slide

- Building software is not just solving problem, it consist of **Engineering Practice** and **Engineering Discipline** to fight the complexity through it life span

Let recap what we learn today

I know its a lot that why I create this slide

- Building software is not just solving problem, it consist of **Engineering Practice** and **Engineering Discipline** to fight the complexity through it life span
- **Change Amplification**, **Cognitive Load**, and **Unknown unknowns** are the root cause of complexity and don’t forget that **working code is not enough**

Let recap what we learn today

I know its a lot that why I create this slide

- Building software is not just solving problem, it consist of **Engineering Practice** and **Engineering Discipline** to fight the complexity through it life span
- **Change Amplification**, **Cognitive Load**, and **Unknown unknowns** are the root cause of complexity and don't forget that **working code is not enough**
- There are plenty of **Values**, **Principles** and **Practices** related to Software Engineering which will help you building software with discipline in mind

Let recap what we learn today

I know its a lot that why I create this slide

- Building software is not just solving problem, it consist of **Engineering Practice** and **Engineering Discipline** to fight the complexity through it life span
- **Change Amplification**, **Cognitive Load**, and **Unknown unknowns** are the root cause of complexity and don't forget that **working code is not enough**
- There are plenty of **Values**, **Principles** and **Practices** related to Software Engineering which will help you building software with discipline in mind
- **FizzBuzz** can go wild than you can imagine 😏