Обретение навыков

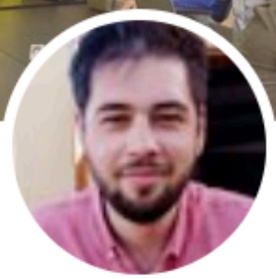
Орел
Апрель 2018
@nikitonsky

Никита Прокопов

FIRA CODE
≠ ⟶ ++ :=

Δeṭáʃč̭ri̊πt


(RUM)



Nikita

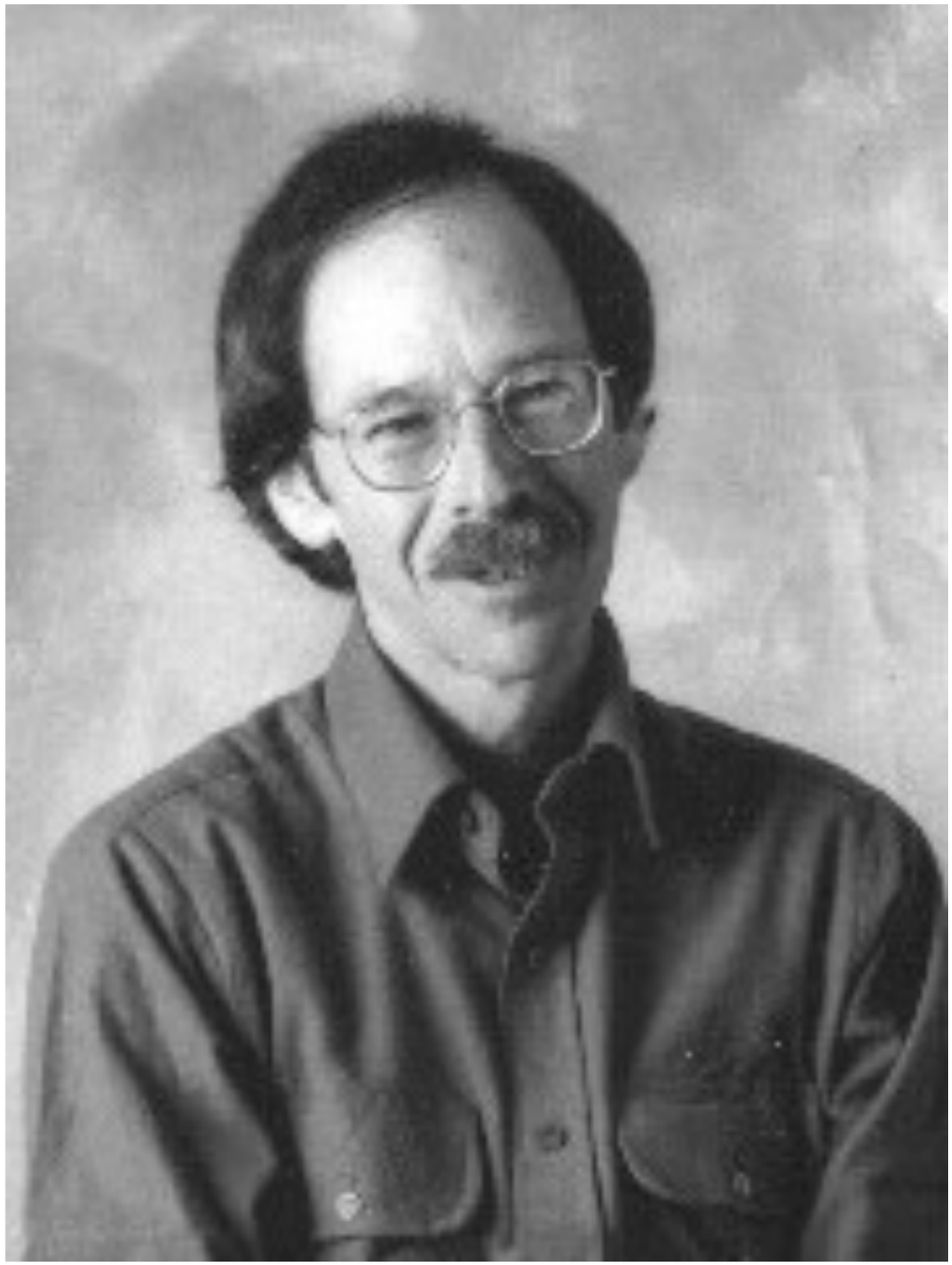
@nikitonsky

Am I a good person? No. But do I try to be better every day? Also no. // Clojure, DataScript, Rum, FiraCode, AnyBar, grumpy.website

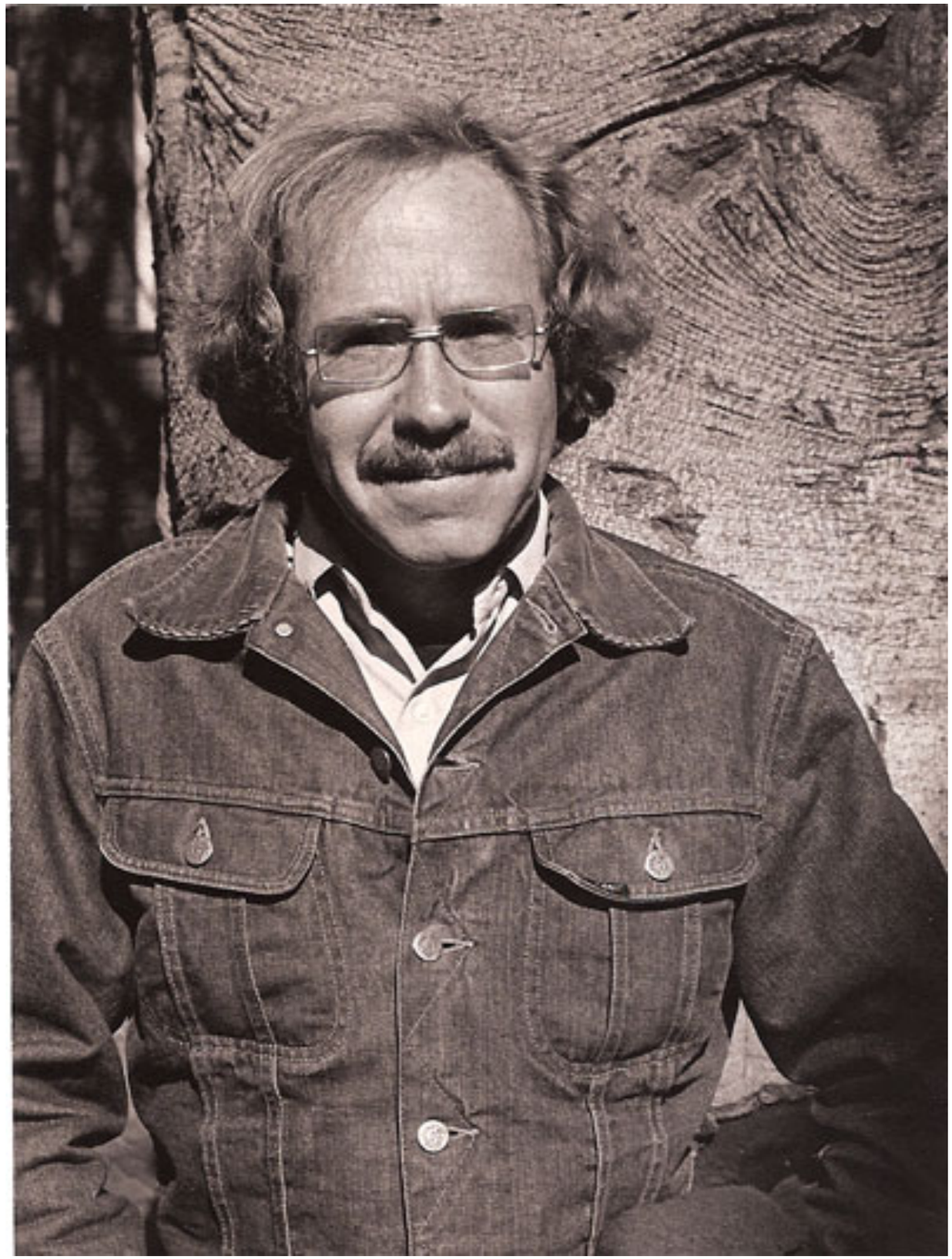
Tweets	Following	Followers
21.1K	304	4,939

@nikitonsky

Модель



Stuart E. Dreyfus



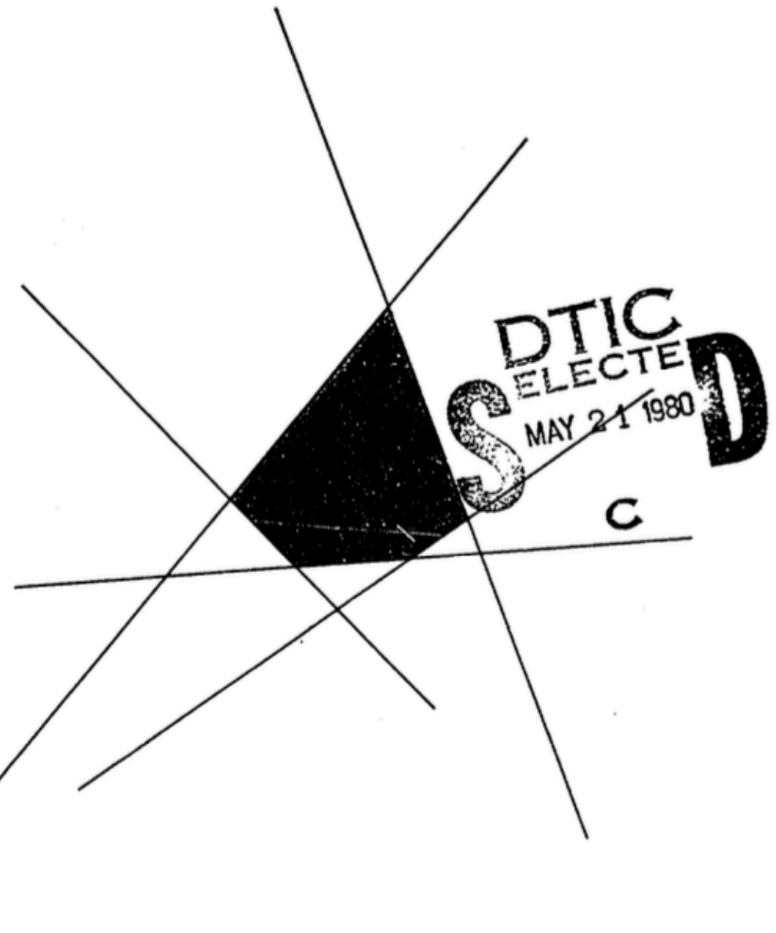
Hubert L. Dreyfus


THE·UNIVERSITY·OF·CALIFORNIA·BERKELEY·
LET THERE BE LIGHT
·1868·

ORC 80-2
FEBRUARY 1980

A FIVE-STAGE MODEL OF THE MENTAL ACTIVITIES INVOLVED IN DIRECTED SKILL ACQUISITION

by
STUART E. DREYFUS
and
HUBERT L. DREYFUS



OPERATIONS RESEARCH CENTER

UNIVERSITY OF CALIFORNIA · BERKELEY

Эксперт

Специалист

Компетентный

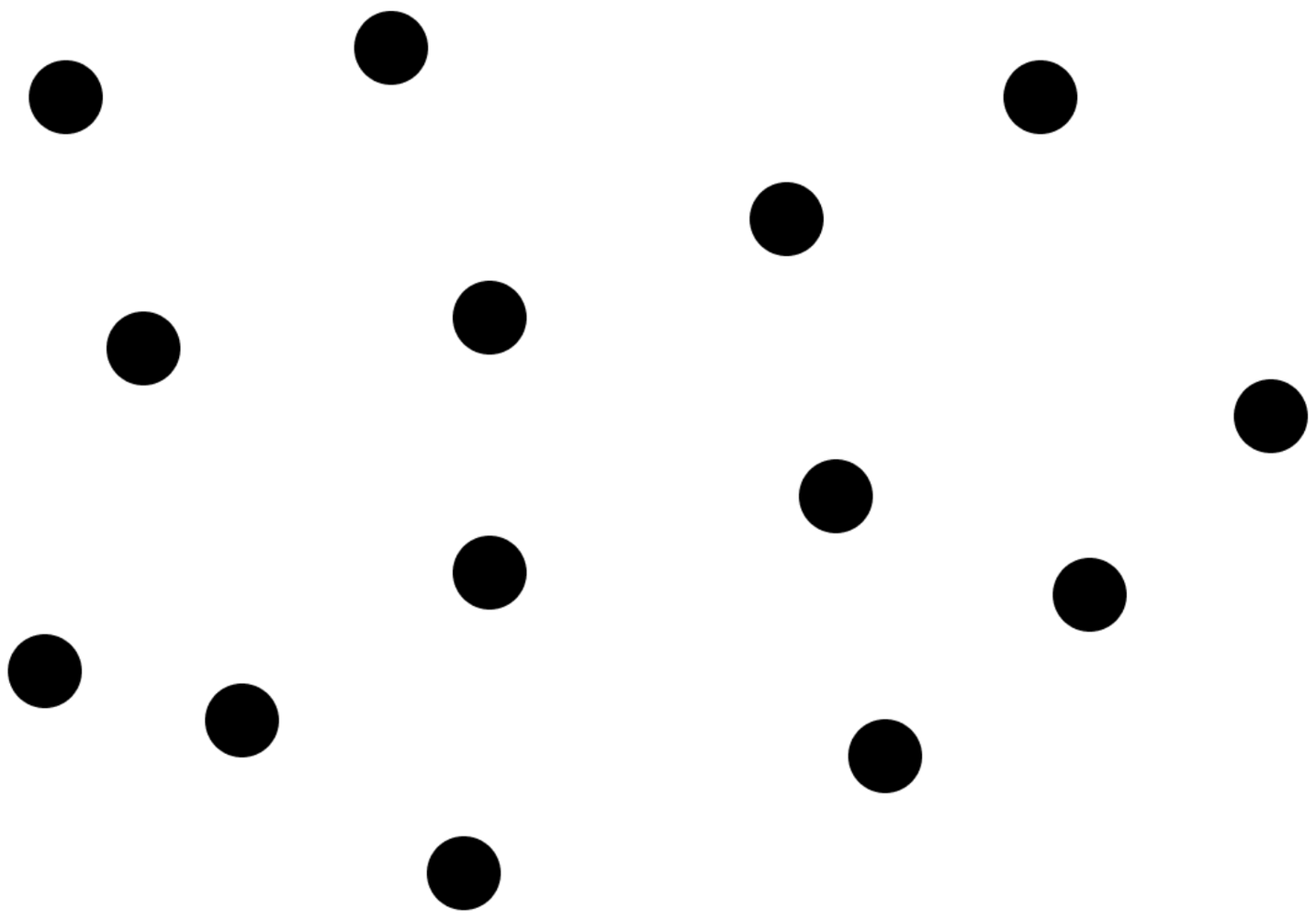
Продолжающий

Новичок

1. Новичок

1. Новичок

- Только начинает
- Нет практического опыта
- Видит изолированные фрагменты
- Учит правила
- Применяет буквально
- Требует контроля



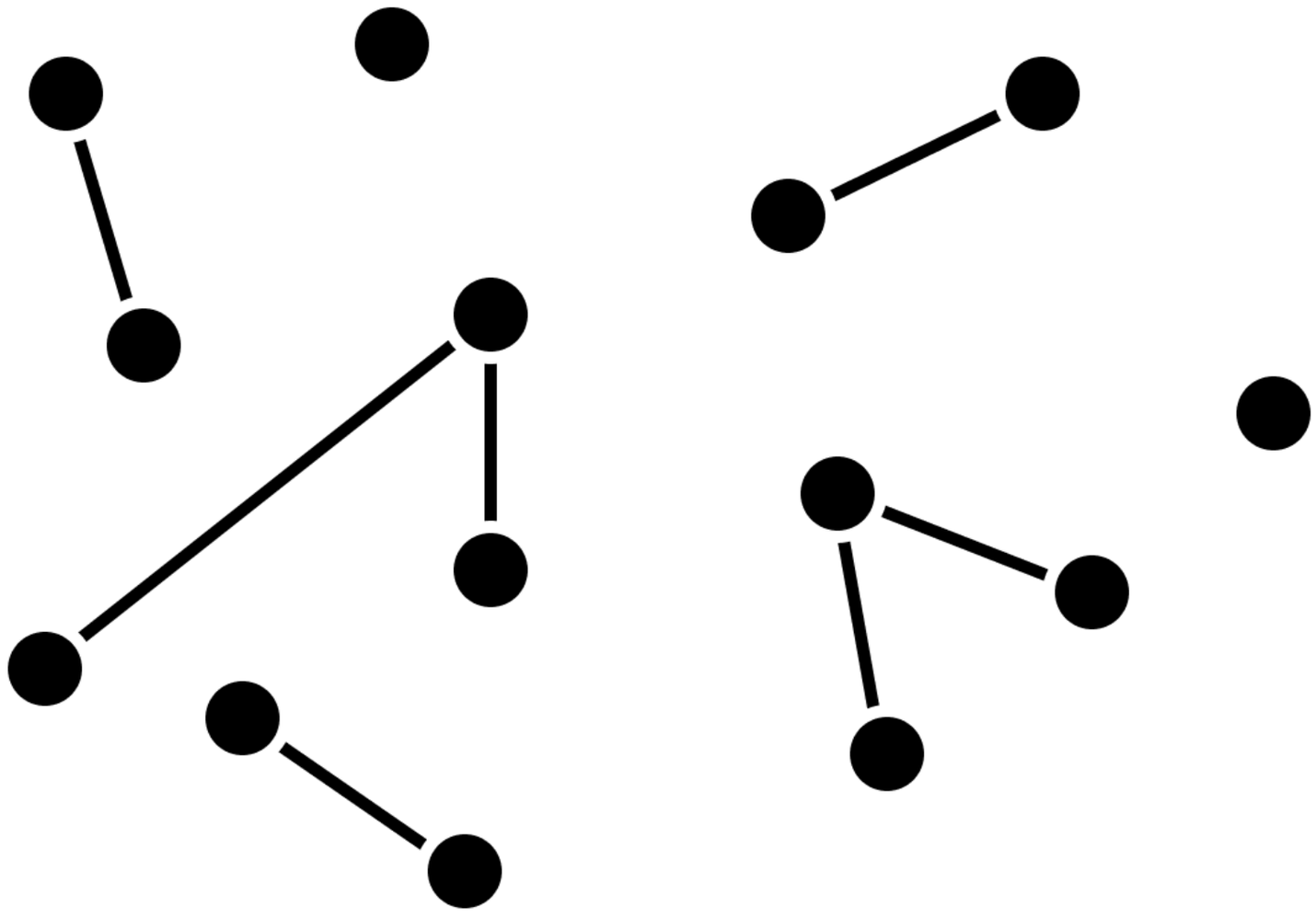

8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H

2. Продвинутый новичок

2. Продолжающий

2. Продолжающий

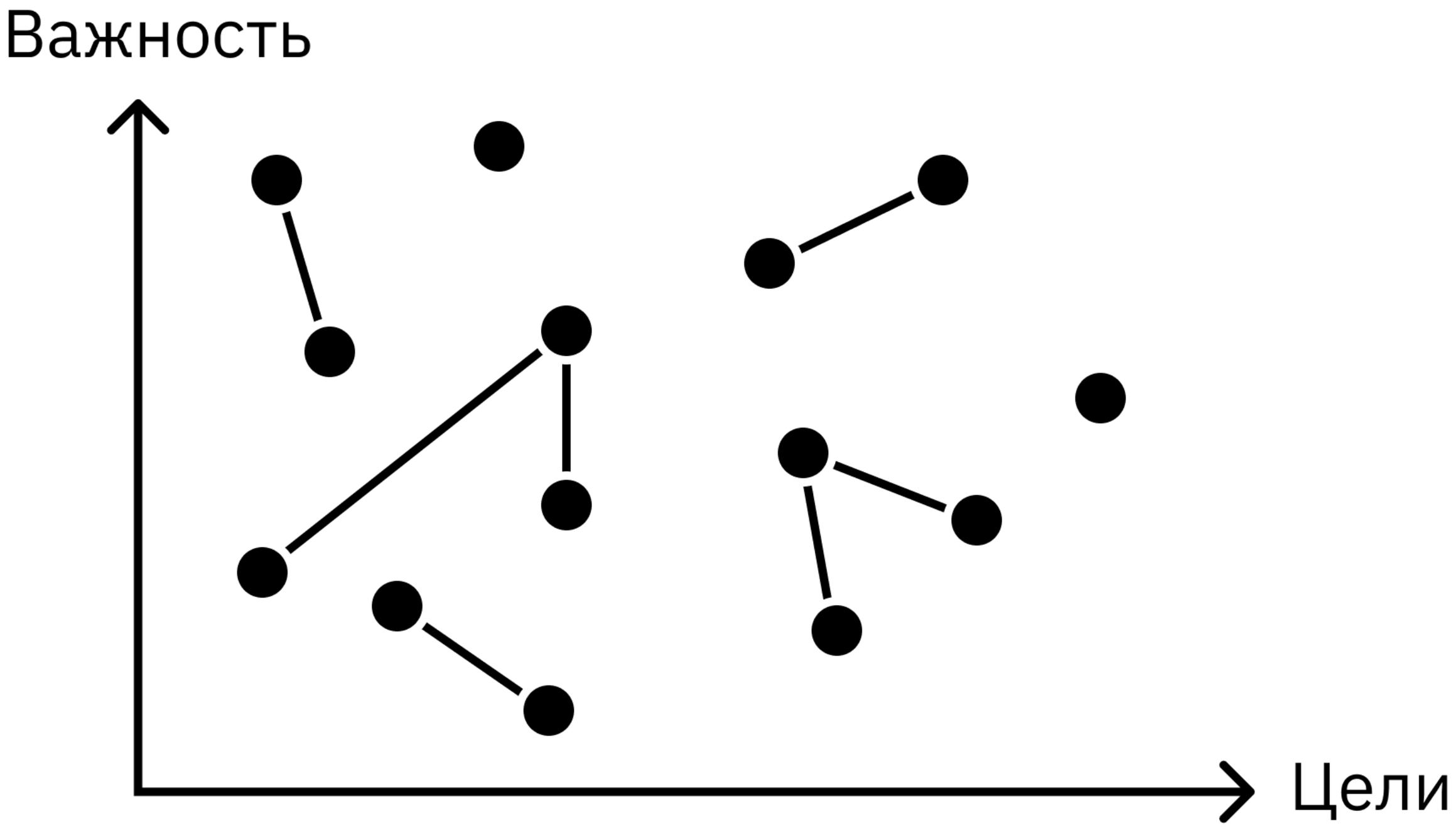
- Немножко опыта
- Фрагменты объединяются в аспекты
- Правила → рекомендации
- У всех аспектов равная важность
- Ищет предел использ. рекомендаций




8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H

3. Компетентный

- Самостоятельная боевая единица
- Делает дело, двигается к цели
- Приоритизирует аспекты от цели
- Стандартизированные процедуры
- Анализирует ситуацию, вырабатывает план
- Большинство людей здесь

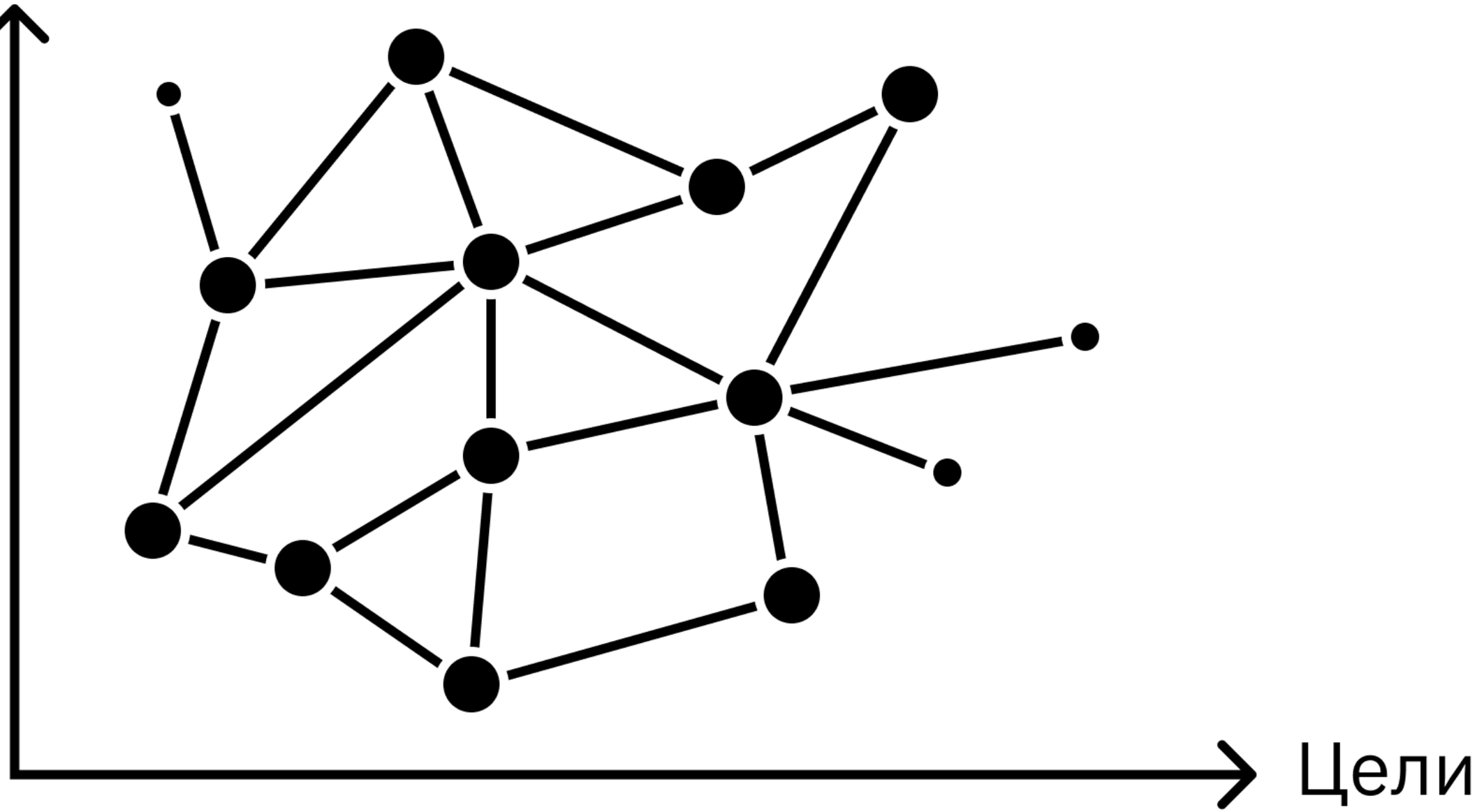

Важность
Цели


8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H

4. Специалист

- Погрузился глубже
- Большой практический опыт
- Полагается на интуицию
- Видит «ситуацию целиком», холистически
- Норма + отклонения
- Руководствуется максимами

Важность

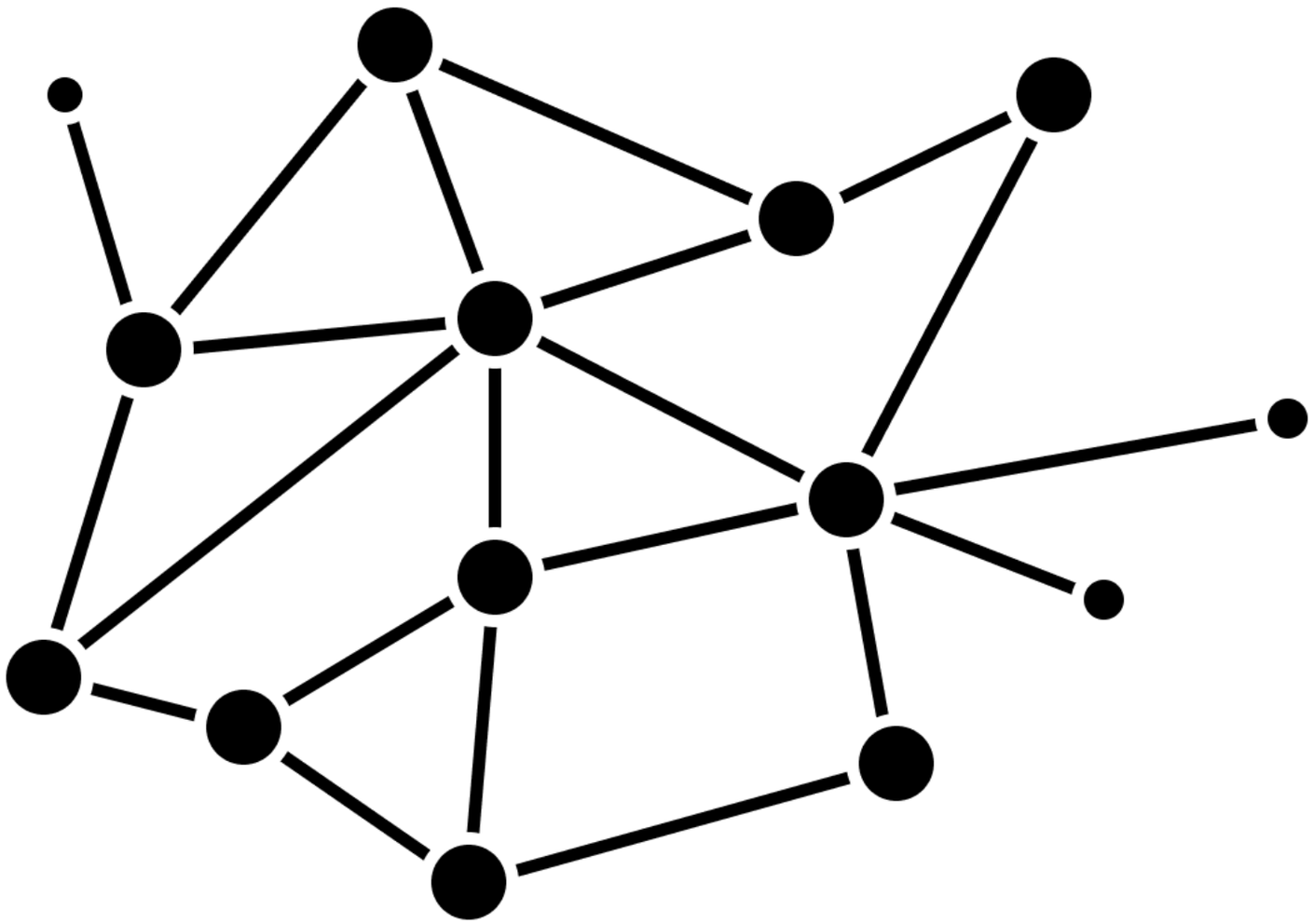


Цели


ДТВ

5. Эксперт

- Огромный практический опыт
- «Делается само»
- Не мыслит правилами
- Не может объяснить
- Известен в проф. кругах
- Видит «границы возможного»




8
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H


ОТВ

Повторим

1. Новичок — учит правила
2. Продолжающий — пробует силы
3. Компетентный — делает дело
4. Специалист — ситуация целиком
5. Эксперт — получается само

Карьера

github.com/basecamp/handbook/blob/master/titles-for-programmers.md

Junior Programmer

- Thoroughly reviewed with substantial back’n’forth before merging
- Basic language features
- Occasional issues following patterns and approaches within existing code bases
- Tightly scoped, routine problems
- < 2 years of experience

Programmer

- Reviewed with the occasional need for material direction/implementation changes
- Follows established patterns and approaches
- Clearly defined/scoped individual features/problems
- 2-5 years of experience

Senior Programmer

- Work doesn't need review, general approach maybe
- Substantial features from concept to shipping
- Material feedback on (junior) programmers
- Deep expertise in one programming env
  + basic proficiency in at least one more
- At lest 5-8 years

Lead Programmer

- Works autonomously with no need for review
- Owning and running subsystems
- Running small teams for substantial projects
- Projects across multiple domains
- Sets professional standards for organization
- Deep expertise in multiple programming envs
- 8-12 years

Principal Programmer

- Set and direct an entire department
- Designing, owning, running new & novel systems
- Running teams for large, long-running projects
- Recognized widely in the industry for material contributions to the state of the art
- Invents new concepts, pushes the whole organization forward regularly
- 12-15+ years of experience

Продвижение

Новичок: следуешь правилам, пытаешься делать хоть что-то

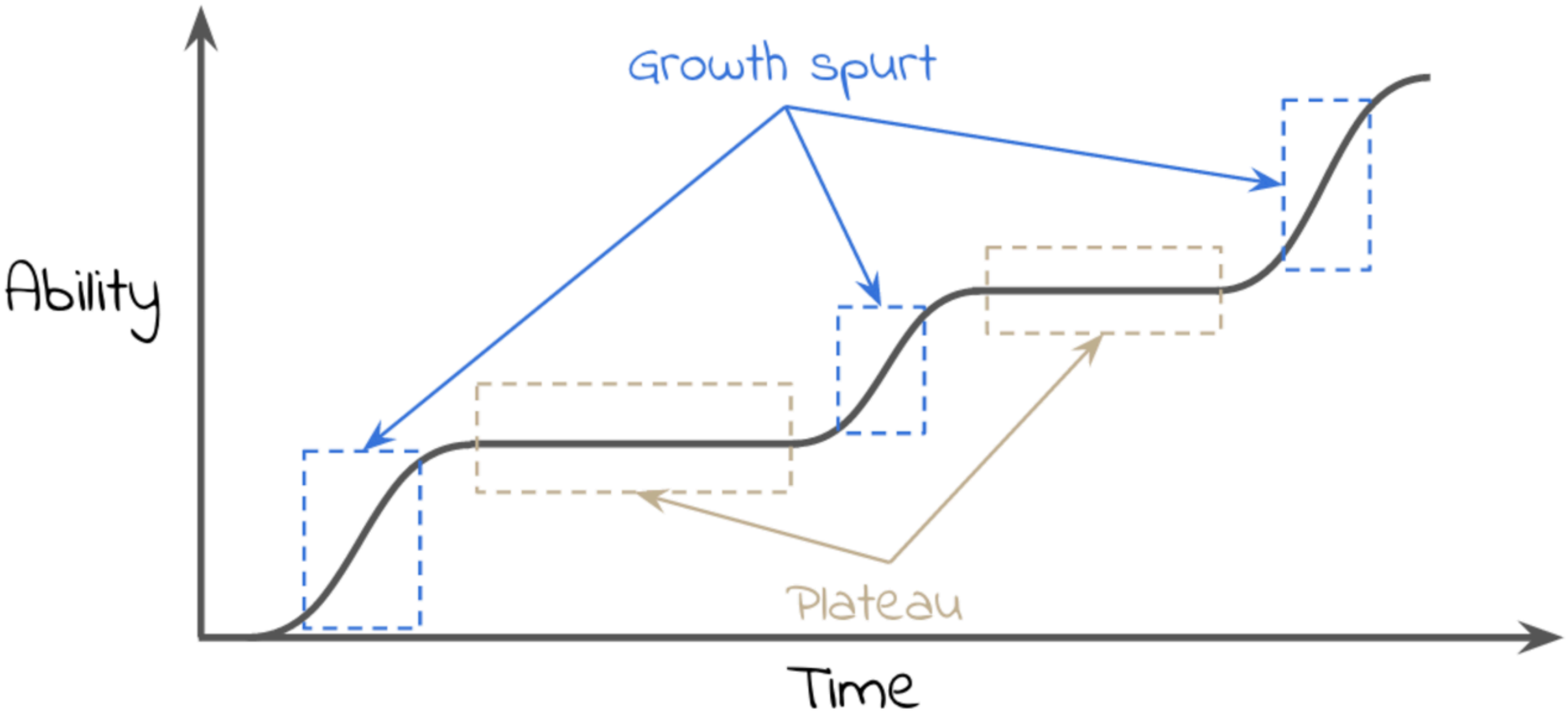
Продолжающий: решаешь задачи/ проблемы, тестируешь правила на прочность

Компетентный: участие в проектах (движение к цели, соответствие целей-усилий)

Специалист: управление проектом (контролируешь ситуацию целиком), погружение в основы

Эксперт: создание проектов

Процесс не ускоряется, стадии пропустить нельзя


Growth spurt
Ability
Plateau
Time

Правила

using global_variables, sometimes camelCase, sometimes_underscore, if it works - don't touch it, who needs comments.

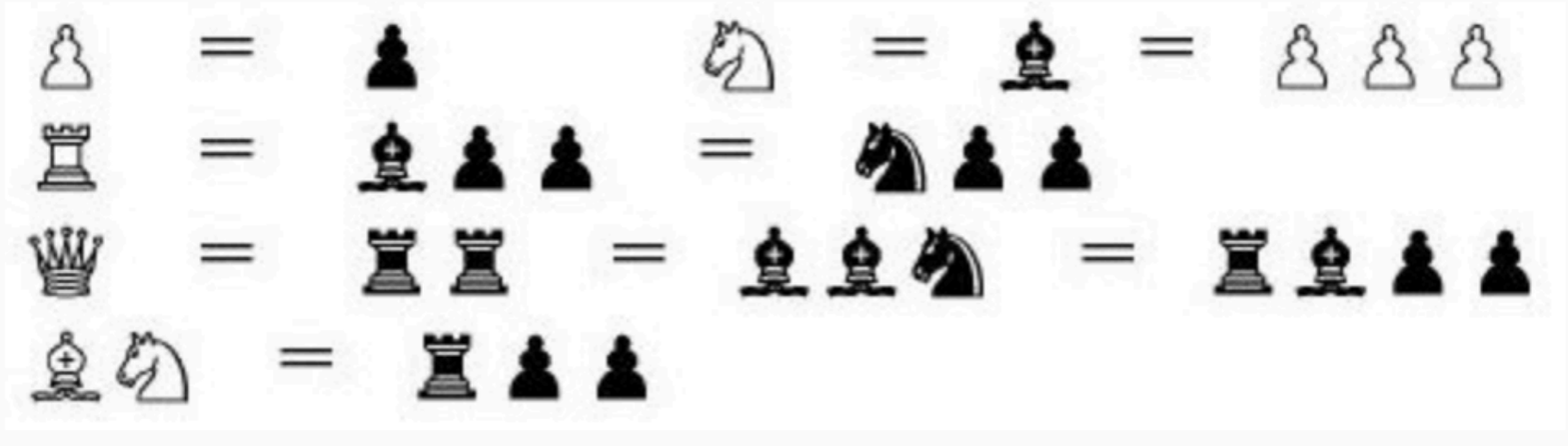
follow consistent naming conventions, overuse inheritance, comment obvious things, being aware of technical debt, get into fights in language A vs B threads

unit testing, source control, refactor often, doc comments, language agnostic, composition over inheritance, everything is about states and data flow, solving problems on a paper, dreaming with Bret Victor

using global_variables, sometimes camelCase, sometimes_underscore, if it works - don't touch it, who needs comments.

- Делай это, не делай того.
- Четко, строго, без контекста (его нет)
- Помогают сориентироваться и начать хоть с чего-то

Чтобы быстрее войти в шахматную жизнь нужно знать следующие равенства.

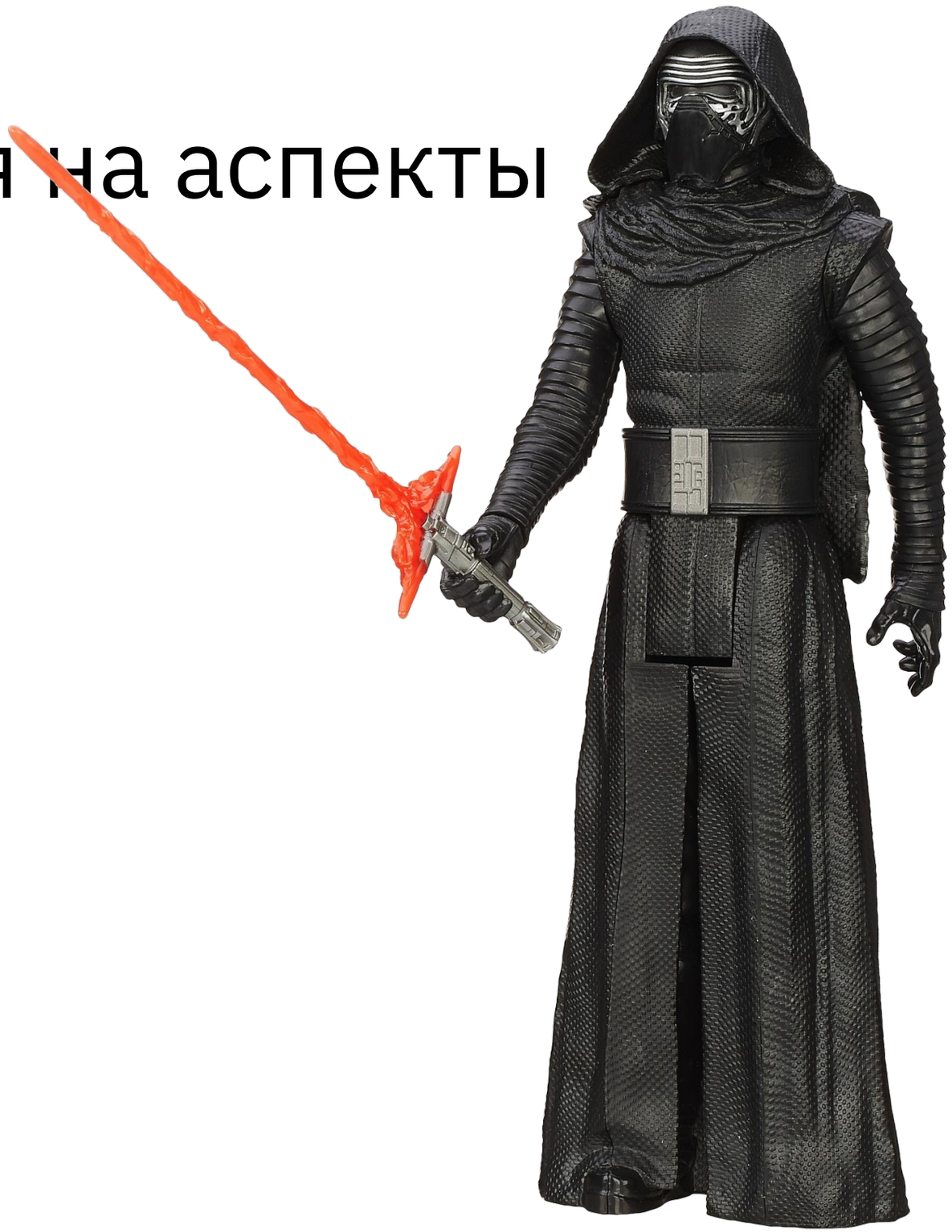


Если в процессе борьбы один из соперников обзавелся лишней фигурой или пешкой, то говорят, что у него **материальное преимущество**.

Если ты после борьбы поменял коня или слона на ладью, то у тебя **лишнее качество**.

Если ты поменял равноценные фигуры, например коня на слона, то считается, что произошел **шахматный размен**. А если после произошедших обменов у одной стороны не хватает до равенства материала, то произошёл **неравноценный размен**.

- Более размытые
- Можно ссылаться на аспекты



- Планы, процедуры, методики
- Как двигаться к цели

- Максимы
- За всё хорошее и против всего плохого
- Меняют смысл по ситуации
- Изучение основ

— Нужно ли программисту знать алгоритмы?

Ben Lesh 🛋️👑🔥

@BenLesh

If you're a complete beginner starting off in JavaScript, there is nothing wrong with just learning a framework up front. Don't listen to people that say you have to master fundamentals first. You have to master them *eventually* but you need to have fun and be productive first.

4:17 AM - Apr 22, 2018

 3,349  1,024 people are talking about this 

- Разбираются сами

using global_variables, sometimes camelCase, sometimes_underscore, if it works - don't touch it, who needs comments.

follow consistent naming conventions, overuse inheritance, comment obvious things, being aware of technical debt, get into fights in language A vs B threads

unit testing, source control, refactor often, doc comments, language agnostic, composition over inheritance, everything is about states and data flow, solving problems on a paper, dreaming with Bret Victor

using global_variables, sometimes camelCase, sometimes_underscore, if it works - don't touch it, who needs comments.

Спаривание

VS

VS

Use your fucking brain.

Handprinted on a
Korrex proof press
from original wood type
by Erik Spiekermann at
ppda Berlin

Introducing Developer Driven Development (DDD)

Development Driven Development (DDD) is a revolutionary new approach to development that focuses on … you guessed it, development! DDD takes a radical approach in an industry filled with tests, metrics, and processes by allowing smart developers to write code. Some managers and even developers have a hard time wrapping their heads around it conceptually because it does require a shift from existing thinking.

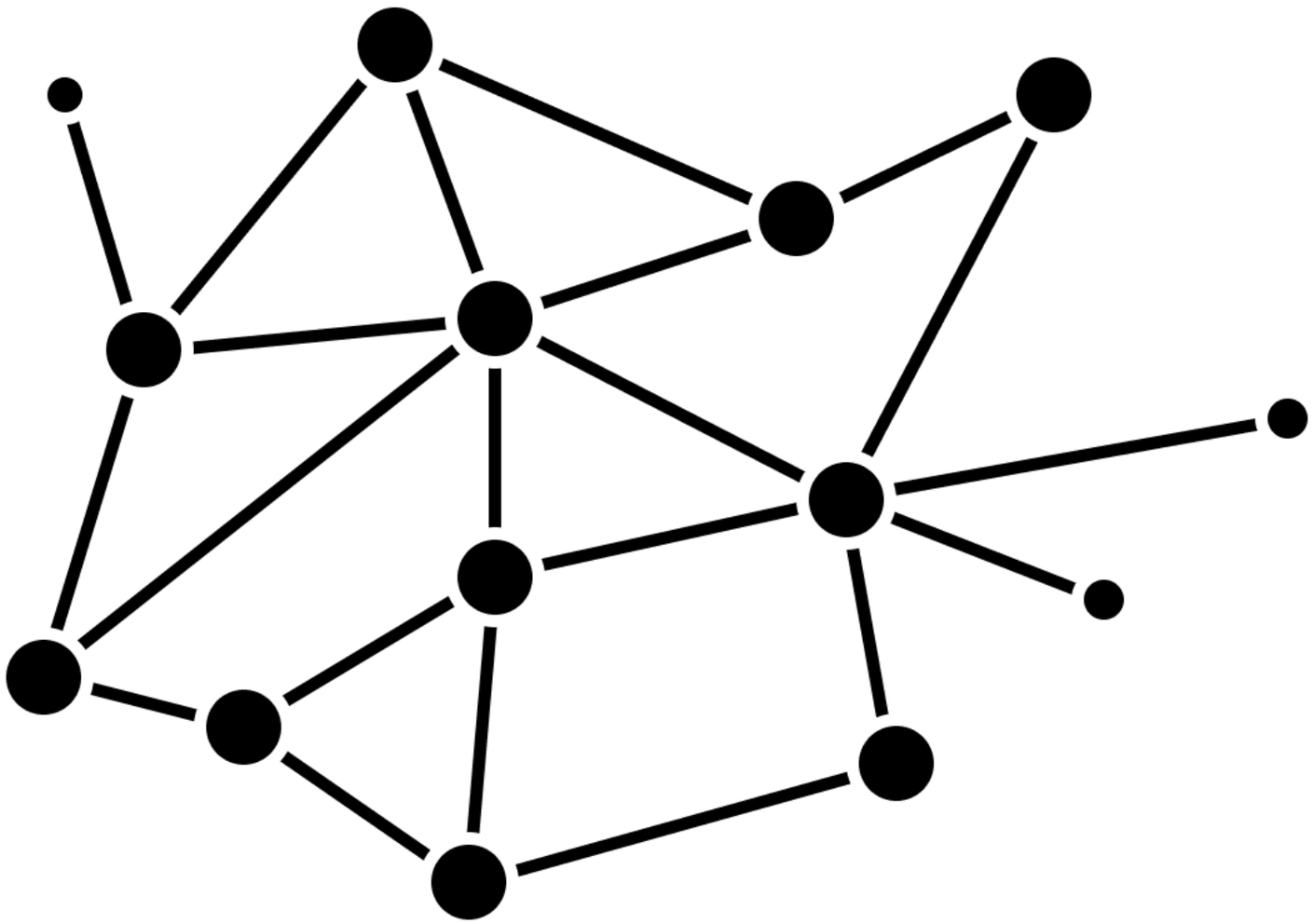
How does it work?

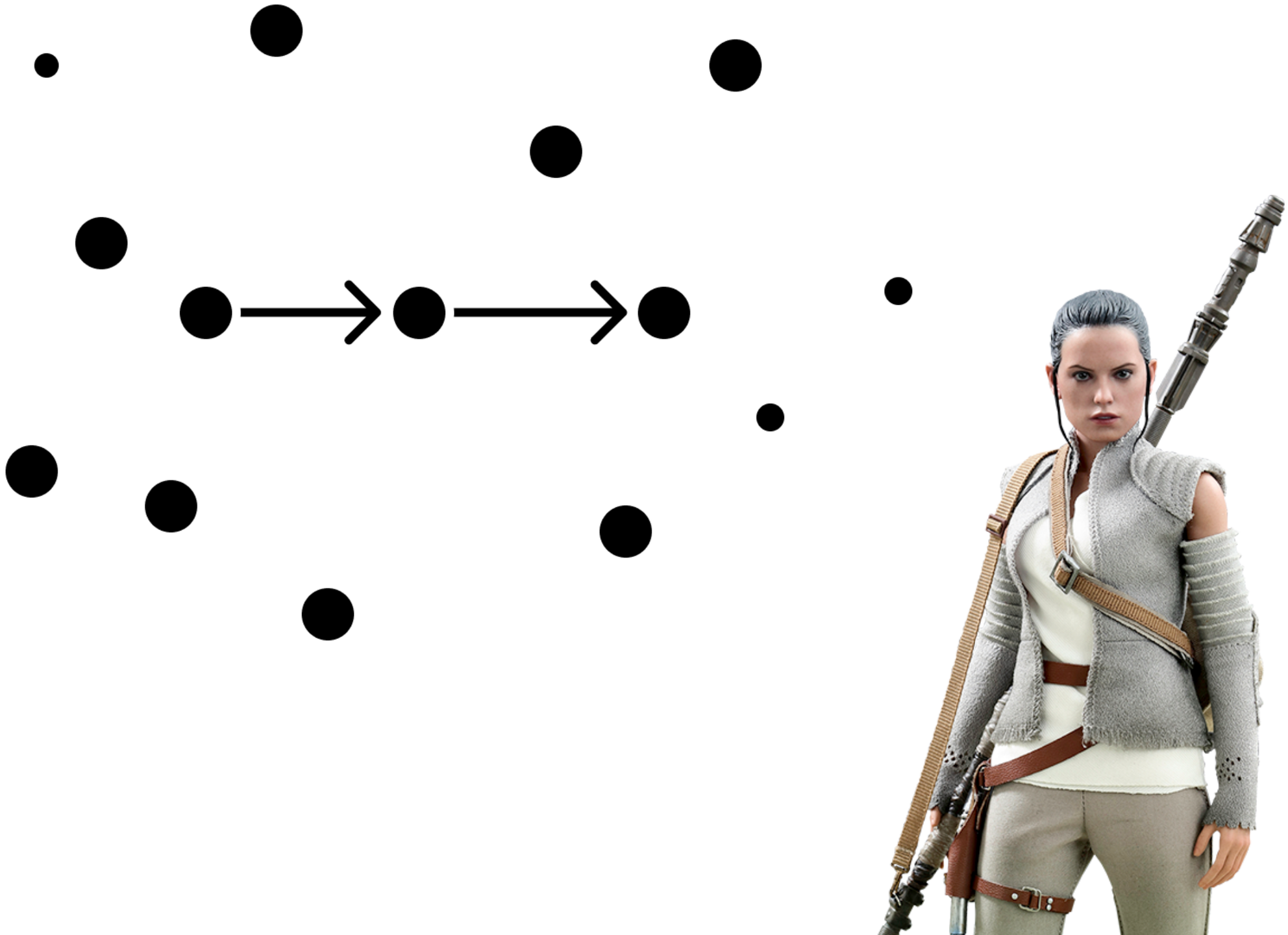
1. First, the developer thinks about the problem they need to solve
2. Then, they solve it (usually by writing code, not always)
3. Repeat until done

DDD vs TDD vs BDD vs FDD vs ADD vs …

There are people out there writing code who really shouldn't be. They produce bad code and crappy software.

The obvious (and correct) solution to this problem is to only use bright, thoughtful programmers.





Bob Marshall

@flowchainsensei

Mostly, I feel like the chap at the right-hand table.

11:29 AM - Sep 14, 2017

80 63 people are talking about this

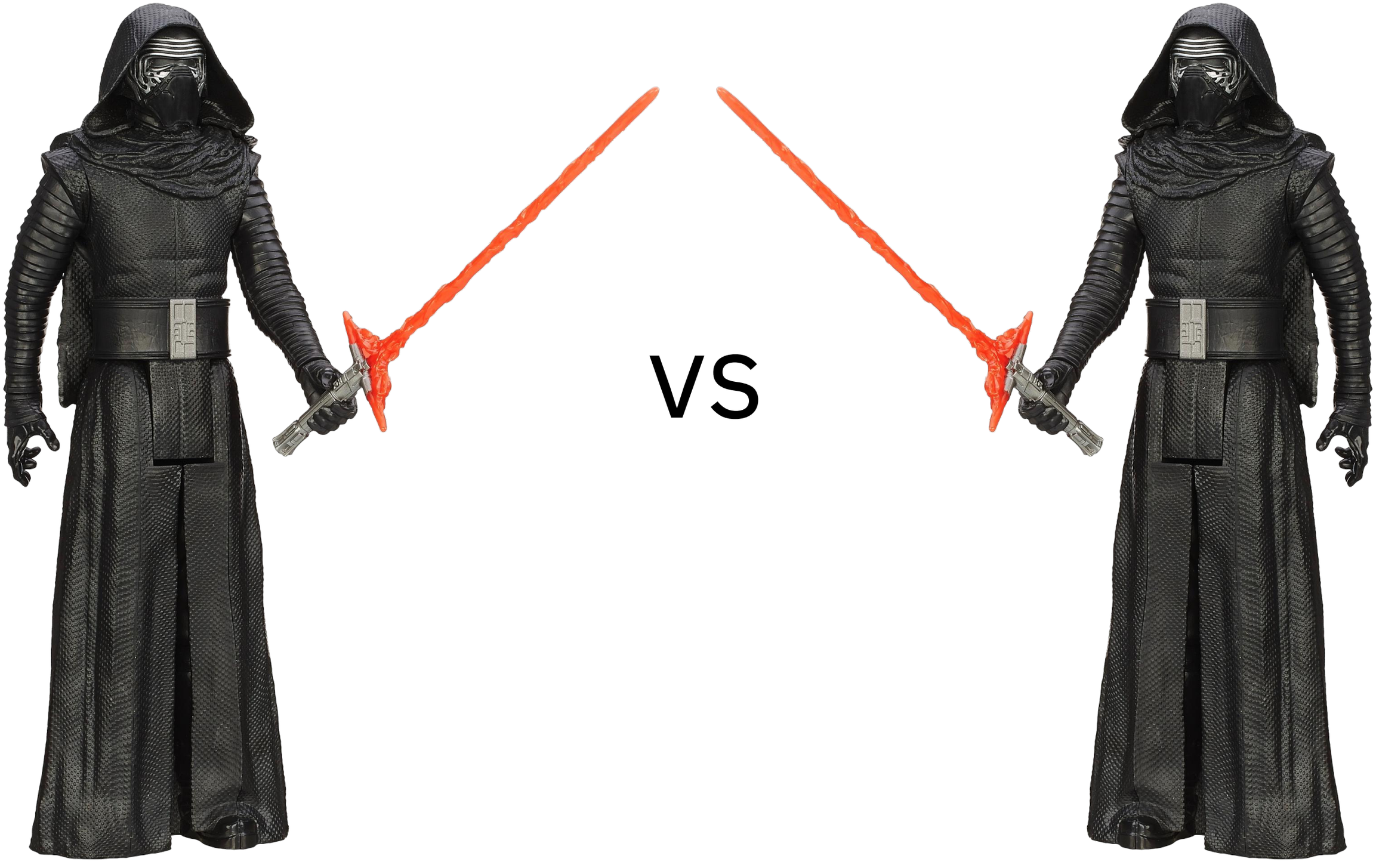
Nickolas Teixeira
@NTTL_LTTN

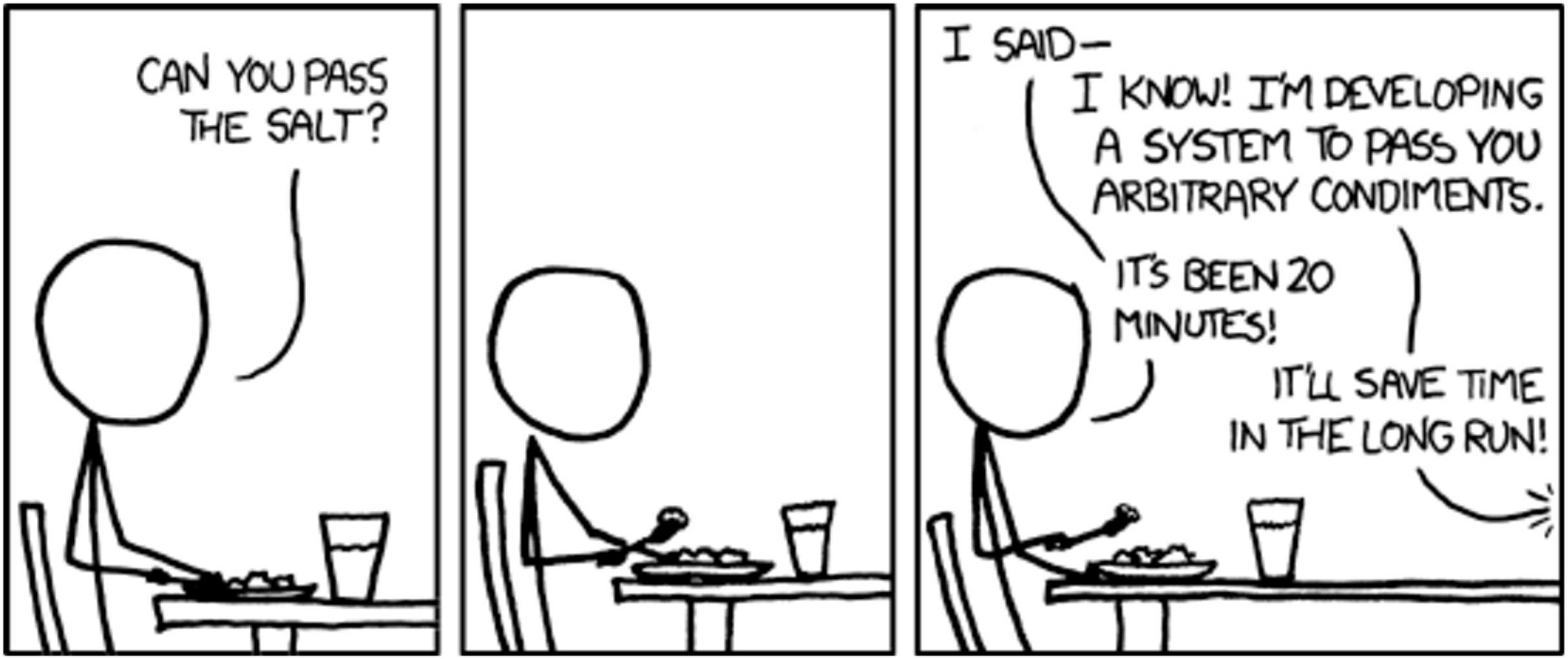
"The best technical tutorials are written by engineers who don't know the technology before writing them."

Thanks 🙏 @sehurlburt for the boost of confidence.

10:25 PM - Mar 20, 2018

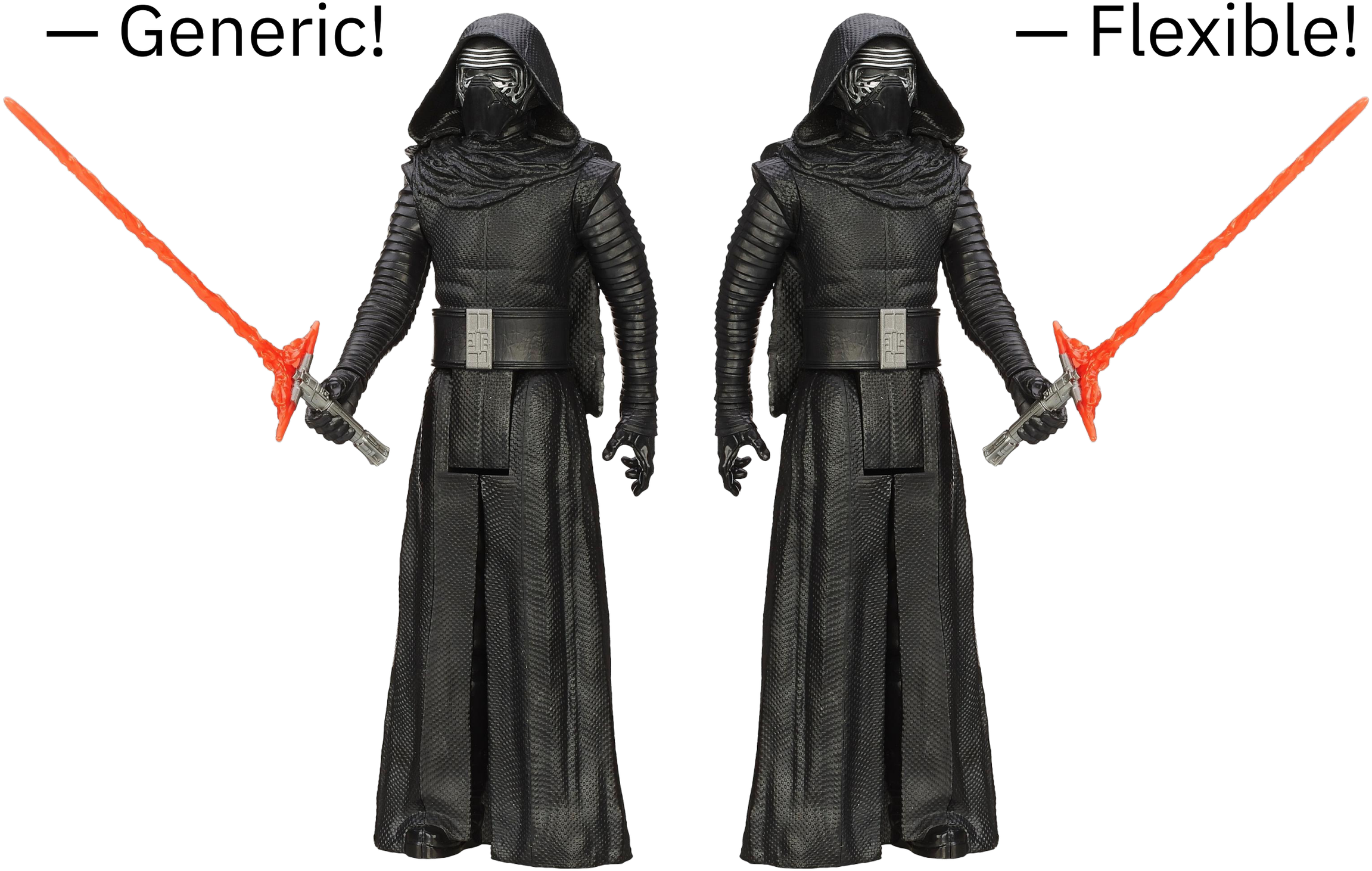
424 83 people are talking about t


VS


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

— Generic!

— Flexible!



— Abstraction layer!

— UI widgets!

— Parser!

— ORM!

Eric Brandes

@BrandesEric

If a developer tells you they are building something "Generic" and "Flexible", rest assured that they're wasting your time and money.

7:21 PM - Jul 13, 2017

213 196 people are talking about t

VS

VS

- Нужна разница в уровнях
- Лучший специалист не всегда лучший учитель
- Удачные и неудачные комбинации

Test-Driven Development

Dan Lebrero

@DanLebrero Follows you

Hands on software architect. All-time #Java developer. Blinded by #Clojure brightness.

danlebrero.com

Joined March 2016

labs.ig.com/code-coverage-100-percent-tragedy

```
Scenario Outline: Get returns correct DtoToAdditionalDataModelMapper

  Given a DefaultDtoToAdditionalDataModelMapperRegistry configured with DtoToAdditionalDataModelMappers:
    | HiLoMarketDTO      | HiloBinaryMarketDtoToModelMapper      |
    | TunnelMarketDTO    | TunnelBinaryMarketDtoToModelMapper    |
    | OneTouchMarketDTO  | OneTouchBinaryMarketDtoToModelMapper  |
    | UpDownMarketDTO    | UpDownBinaryMarketDtoToModelMapper    |

  And a mock marketDto which returns <dtoName> from marketDto.getMarketDTOType()

  When DefaultDtoToAdditionalDataModelMapperRegistry getMapper(MarketDTO marketDto) is called with marketDto

  Then <dtoMapper> is the DefaultDtoToAdditionalDataModelMapperRegistry returned

Examples:
  | dtoName            | dtoMapper                             |
  | HiLoMarketDTO      | HiloBinaryMarketDtoToModelMapper      |
  | TunnelMarketDTO    | TunnelBinaryMarketDtoToModelMapper    |
  | OneTouchMarketDTO  | OneTouchBinaryMarketDtoToModelMapper  |
  | UpDownMarketDTO    | UpDownBinaryMarketDtoToModelMapper    |
```

```java
public class DtoToAdditionalDataModelMapperRegistryStepDefs {
    private DefaultDtoToAdditionalDataModelMapperRegistry testClass;
    private DtoToAdditionalDataModelMapper returned;
    private DtoToAdditionalDataModelMappersStepDefs additionalMapperTest = new DtoToAdditionalDataModelMappersStepDefs()
    private MarketDTO marketDTO;

    @Given("^a DefaultDtoToAdditionalDataModelMapperRegistry configured with DtoToAdditionalDataModelMappers:$")
    public void a_DefaultDtoToAdditionalDataModelMapperRegistry_configured_with_DtoToAdditionalDataModelMappers(Map<Stri
        Map<String, DtoToAdditionalDataModelMapper> registry = new HashMap<>();

        for (Map.Entry<String, String> entry : mapOfMappers.entrySet()) {
            DtoToAdditionalDataModelMapper mapper = createInstanceOf(PACKAGE, entry.getValue());
            registry.put(entry.getKey(), mapper);
        }

        testClass = new DefaultDtoToAdditionalDataModelMapperRegistry(registry);
    }
```

```java
    @Given("^a mock marketDto which returns ([^\\s]+) from marketDto.getMarketDTOType\\(\\)$")
    public void a_mock_marketDto_which_returns_from_marketDto_getMarketDTOType_(String type) throws Throwable {
        marketDTO = Mockito.mock(MarketDTO.class);
        when(marketDTO.getMarketDTOType()).thenReturn(type);
    }


    @When("^DefaultDtoToAdditionalDataModelMapperRegistry getMapper\\(MarketDTO marketDto\\) is called with marketDto$")
    public void DefaultDtoToAdditionalDataModelMapperRegistry_getMapper_MarketDTO_marketDto_is_called_with_marketDto() t
        returned = testClass.getMapper(marketDTO);
    }

    @Then("^([^\\s]+) is the DefaultDtoToAdditionalDataModelMapperRegistry returned$")
    public void _dtoMapper_is_the_DefaultDtoToAdditionalDataModelMapperRegistry_returned(String expectedMapper) throws T
        assertEquals(expectedMapper, returned.getClass().getSimpleName());
    }

}
```

```java
public class TestClassLoader {

    public static <T> T createInstanceOf(String className, Object... initArgs) {
        try {
            Class clazz = Class.forName(className);
            Constructor constructor = clazz.getDeclaredConstructors()[0];
            return (T)  constructor.newInstance(initArgs);
        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
            throw new RuntimeException("Could not create and instance of class: " + classNam
                    "does it have a default constructor? is it public?", e);
        }

    }

    public static <T> T createInstanceOf(String packageName, String className, Object... i
        return createInstanceOf(packageName + className, initArgs);
    }
}
```

```java
public class DefaultDtoToAdditionalDataModelMapperRegistry implements DtoToAddi

    private final Map<String, DtoToAdditionalDataModelMapper> registry;

    public DefaultDtoToAdditionalDataModelMapperRegistry(Map<String, DtoToAdditi
        registry = new HashMap<>(pRegistry);
    }

    @Override
    public DtoToAdditionalDataModelMapper getMapper(MarketDTO marketDto) {
        return registry.get(marketDto.getMarketDTOType());
    }

}
```

Yes, a simple map lookup. I had enough trust with the developer to bluntly say, "That is a big waste of time."

"But my boss expects me to write test for all classes," he replied.

"At the expense of?"

"Expense?"

"Anyway, those tests have nothing to do with BDD."

"I know, but we decided to use Cucumber for all tests"

Me: "…"

I understand the mental satisfaction of bending the tools to your will, but nonetheless it made me sad.

Методика: тесты помогают повысить стабильность системы к изменениям

Методика: тесты помогают повысить стабильность системы к изменениям

Методика: тесты помогают повысить стабильность системы к изменениям

Правило: Тесты — хорошо, больше тестов — еще лучше.
Пишем 100% тестов, и пишем их до кода

OOP Patterns

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

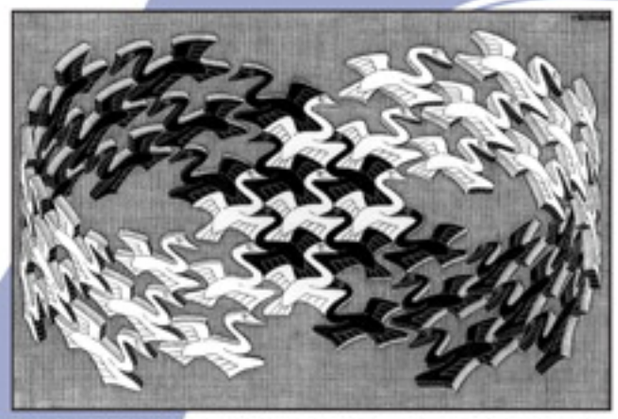
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



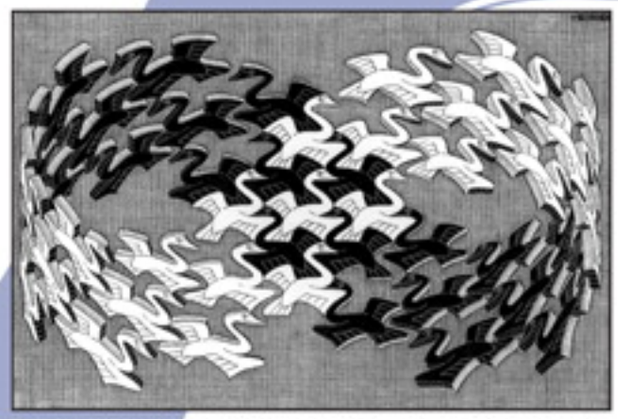
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Don’t repeat yourself

(объединяет два похожих класса в «абстракцию»)

— You’re building an abstraction that doesn’t exist in your domain

— DRY решает локальные проблемы но создает глобальные

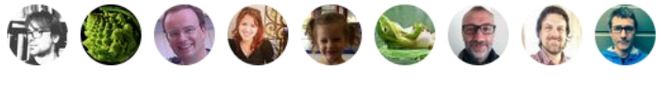
Nicolò Pignatelli
@nicolopigna

Follow

The *original* DRY principle. And it's about knowledge, not code. @kandddinsky @ziobrando

2:43 PM - 20 Oct 2017

231 Retweets **288** Likes

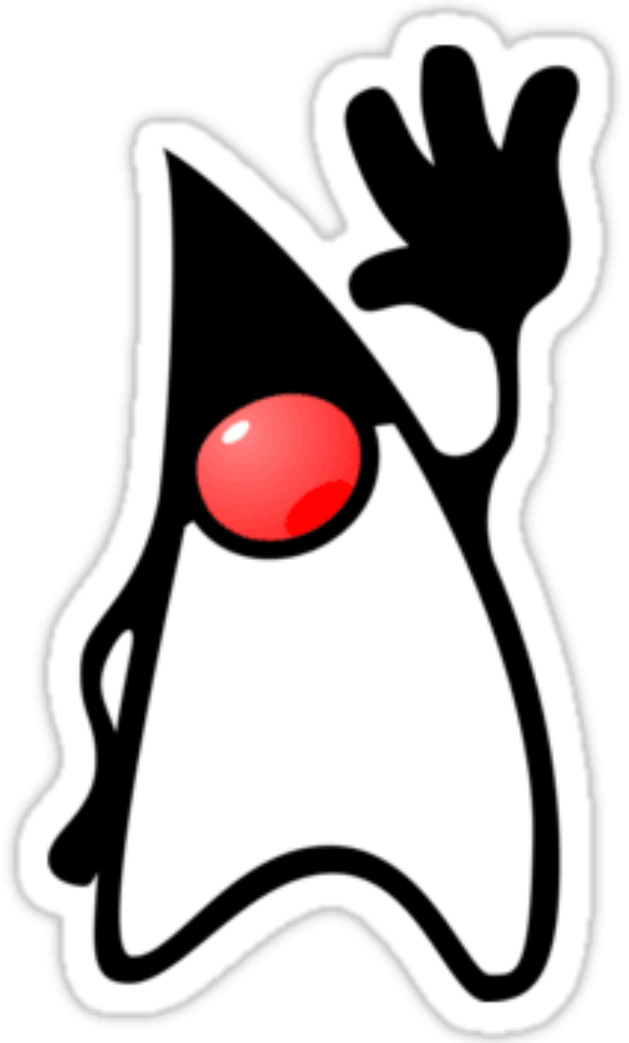


4 231 288

— We took a very good insight about knowledge management and consistency, and turned it into code nonsense

— LOL что?

Языки



Классы, интерфейсы, иерархии, правила наследования, правила видимости, вложенные классы, исключения, статические методы

- Тоже ООП
- Динамический
- Соглашения вместо ограничений
- Monkey patching
- Максимы!

Beautiful is better than ugly

Explicit is better than implicit

Simple is better than complex

Complex is better than complicated

Flat 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!

Do it yourself

Классов нет. Типов нет. ООП нет.
Ничего нет.

Все доступно и открыто всем.

Все можно переопределить

Даже синтаксис

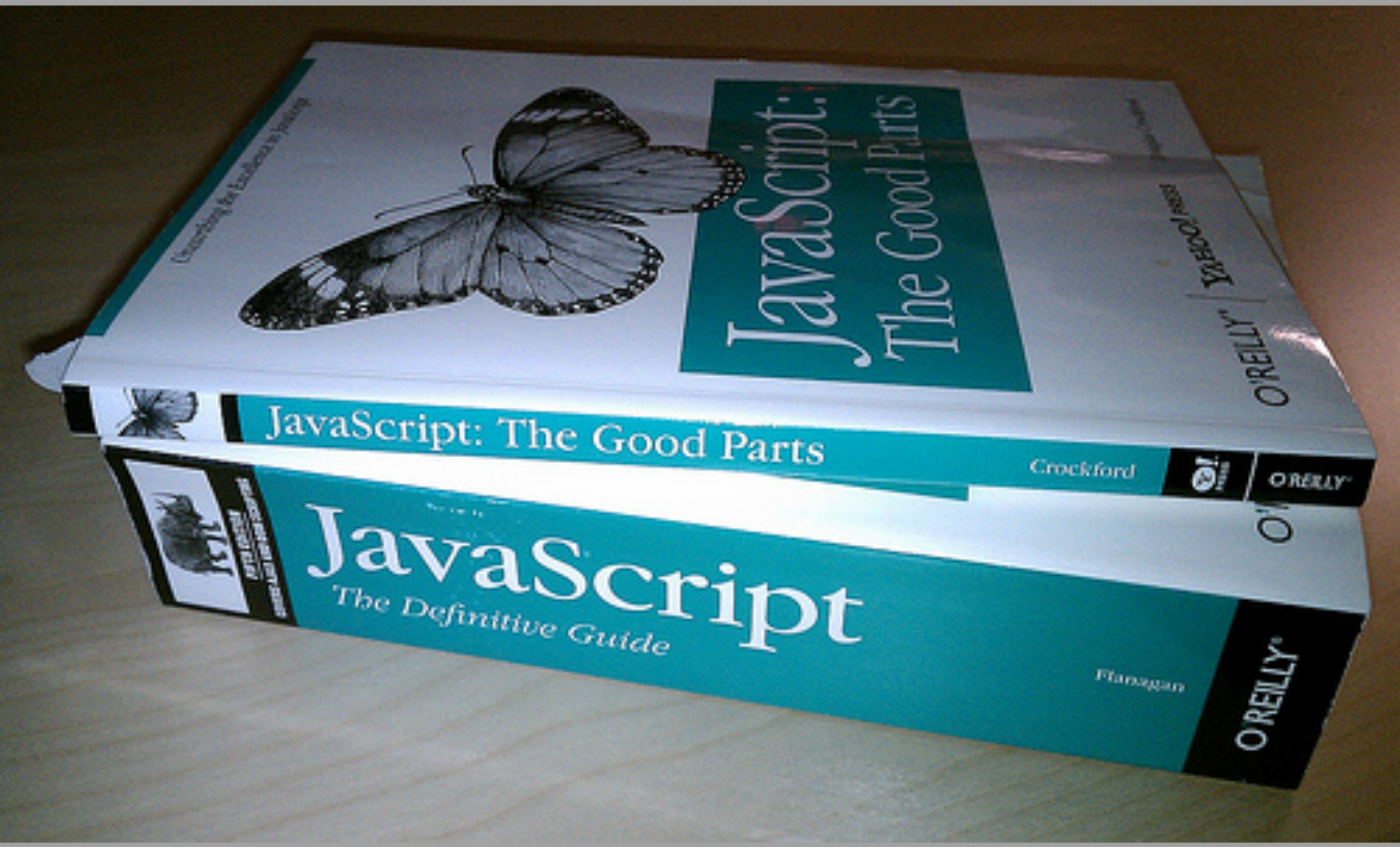
Пишите как хотите —

— **Can we make programs out of simpler stuff**?
That's the problem after 18 years of using, like, C++ and Java, **you're exhausted**.

How many people have been programming for 18 years? How many for more than 20? More than 25? Less than 5? [...]

It may be an indictment of Clojure as a beginner's language or may be that **Clojure is the language for cranky, tired, old programmers**


ICT AWA
SWISS ICT
AWARD


JavaScript:
The Good Parts
O'REILLY
JavaScript: The Good Parts
Crockford
O'REILLY
JavaScript
The Definitive Guide
Flanagan
O'REILLY

- Ужесточение конструкций (prototypes → classes)
- Огромная библиотека тривиальных функций (npm)

Как выглядит эксперт

О книге «Александр Суриков: очерк графического стиля» – Владимир Кричевский

Академия Яндекса

Ортогональные структуры, рамки, ассюре

ЗА НОВЫЙ БЫТ
I. ВАРВАРСТВО В БЫТУ
II. БЮРОКРАТЫ и САМОДУРЫ
III. СОВЕТСКИЕ МЕЩАНЕ
IV. РОСТКИ НОВОГО.
КРАСНАЯ ГАЗЕТА

3 Печать обложки сборника.
Ленинград, «Красная газета», 1929.
Редкий для Сурикова случай (тема обязывала) применения палочного шрифта.

4 Печать обложки. Набор.
Нижегородское краевое издательство, 1932.

Последнее слово заголовка будет обязательно пояснено, а пока – о тривиальном прямоугольнике.

Для стиля, во всяком случае суриковского, самое существенное – композиция. Прямоугольник лежит в основе типографических построений, и именно он был знаменем материальной культуры двадцатых годов. Свои излюбленные нефигуративные «картины», почти всегда статичные, Суриков строил на угловых элементах (полурамках) и полных прямоугольниках. Они взаимодействуют – смыкаются между собой, наползают друг на друга, наконец, создают внешнее обрамление. Самое примечательное – рамка внутри рамки. Этот приём позволял усложнить графическое пространство, разбить его на планы, вычленить композиционное ядро, создать зрительно выгодный конфликт между нефигуративными и – да-да! – вполне фигуративными элементами, которые Суриков тоже успешно культивировал.

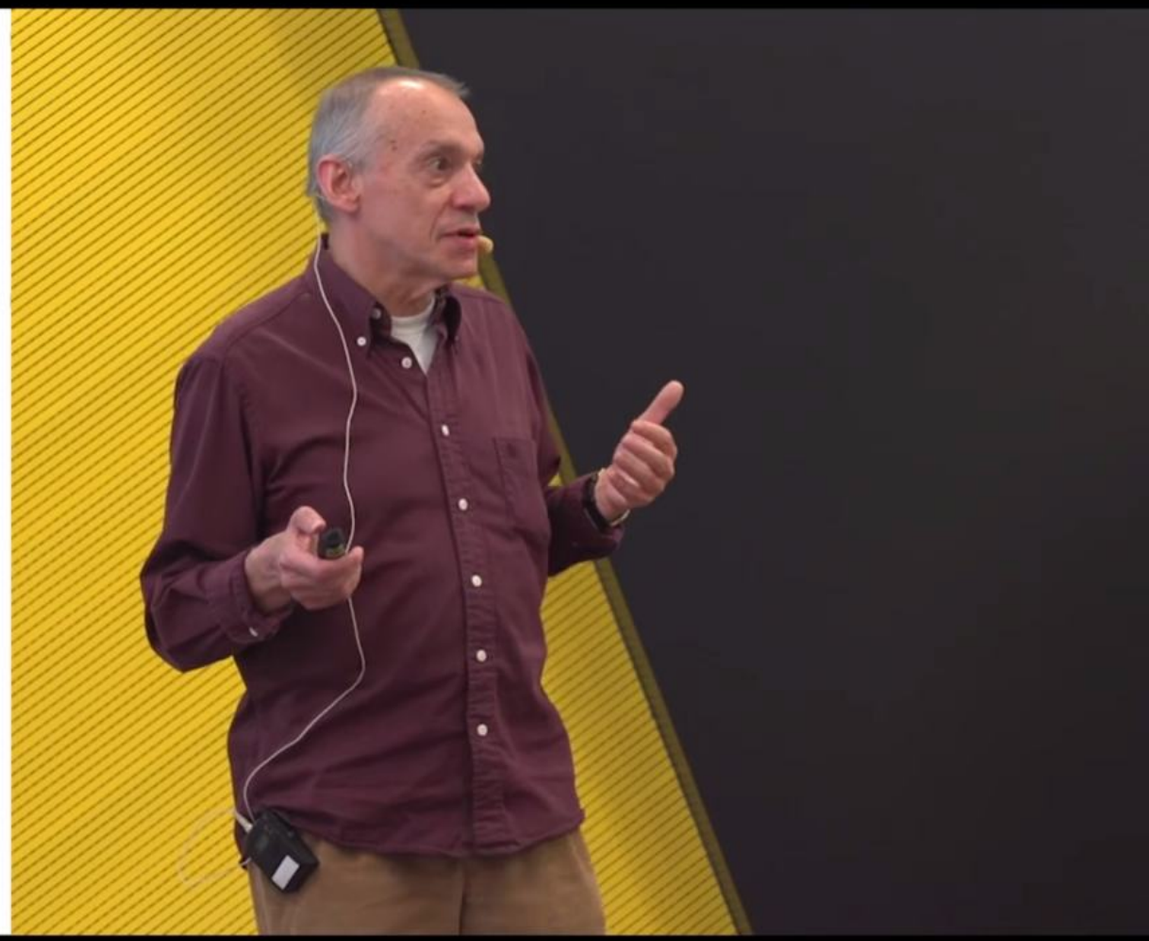
Левые оформители бодро отвергали рамку, но наш модернист – и это очень необычно – был к ней пристрастен и даже склонялся к жирной окантовке собственных иллюстраций. По мне рамка, как правило, избыточна, она нивелирует прелесть свободного контура. Но без неё не было бы и Александра Сурикова, хотя он не раз от оков рамки с успехом освобождался.

«Командарм 2» – весь в рамках. Работу, помеченную характерным вензелем, я впервые и не без приятного удивления встретил в каталоге выставки русского графического искусства XVIII–XX веков (Ньюкасл, 1978). На шмуцтитуле раздела «Конструктивисты и другие» дана именно обложка «Командарма 2», приписанная анониму. Военную тематику отражает только красная звезда. Впрочем, в присутствии интригующих полосок и многозначительной жирной точки – это всего лишь элемент утончённо острой геометрической абстракции. С первого взгляда стало ясно, что тогда ещё неведомый AS принадлежал явно к ДРУГИМ. Может быть, это и есть случай нашего недекларированного ар деко?

10

5 Печать лица (фрагмент) и спинки обложки. Москва, Гиз, 1930.
Держкий загиб заголовочной строки предиктивная структурой слова, его «срезмерной» протяжённостью и, возможно, самой формой буквы D, провоцирующей поворот своей симметрией относительно горизонтальной оси.
Тыльная сторонка строже лицевой.
На последней заметен некоторый перебор по акцентированным мелким деталям.
Лучше других знаков удалась двойка.

11



33:12 / 1:53:01

https://www.youtube.com/watch?v=g-q3YHUnSe0

youtube.com/watch?v=g-q3YHUnSe0

- Не может объяснить решения, только описать
- Нет плана, сетки, структуры. Каждая страница заново. «Не делая шагов назад»
- Материал «попросил» его сверстать себя определенным образом
- Противоречит сам себе и прекрасно это видит

— Почему вот тут так странно?
— А это произвол :)

— Лучше действовать ситуативно и спонтанно
— Это возникло случайно, и слава богу
— Это хорошая композиция, а это плохая. Но не всегда это объяснимо
— Ну я вас понял. Кажется

Заключение

- Более глубокое понимание причин
- Обучение, рост, общение, споры
- Книги, лекции, проекты, языки

Дискасс!

@nikitonsky