

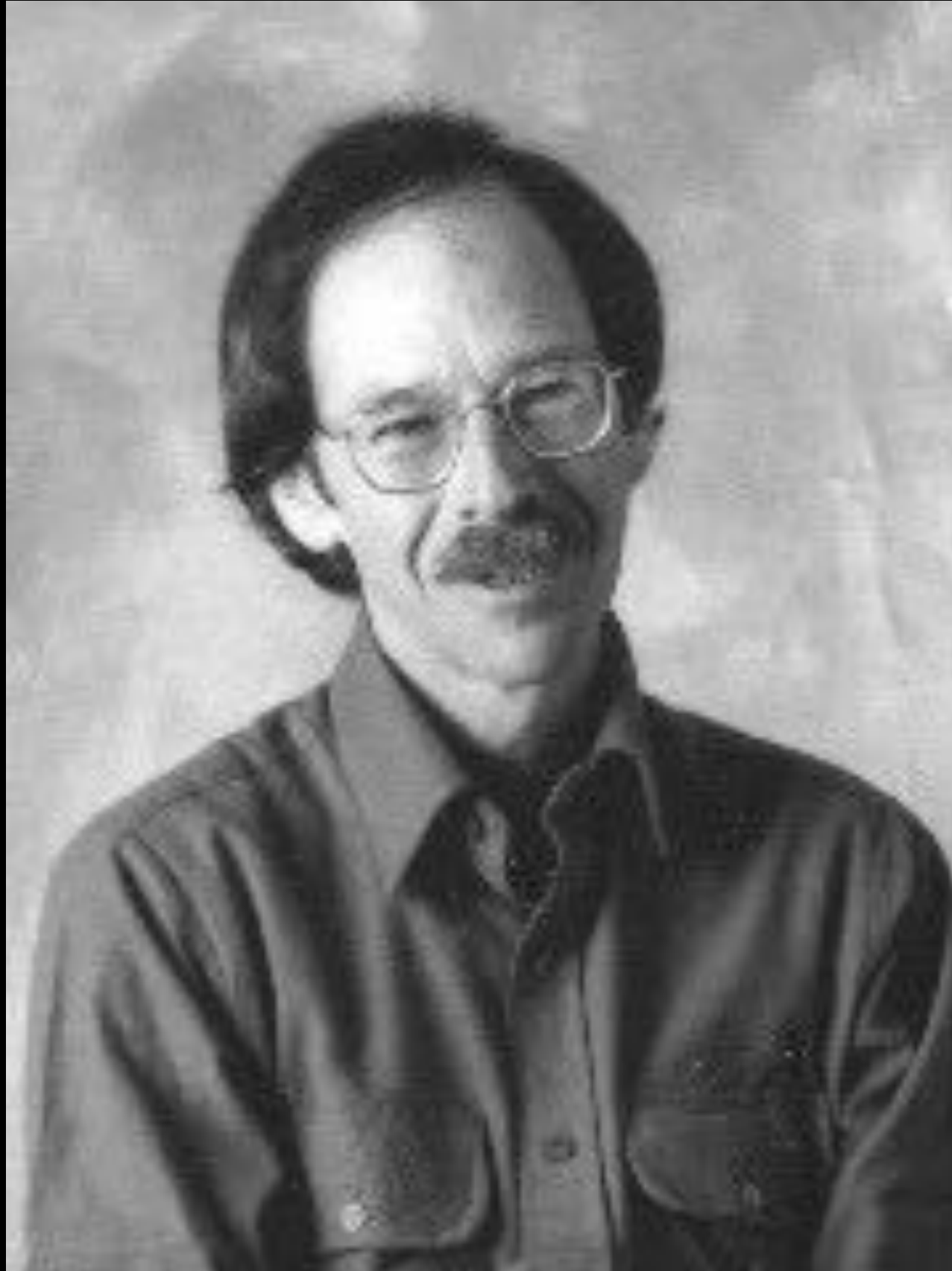
Обретение навыков

Воронеж

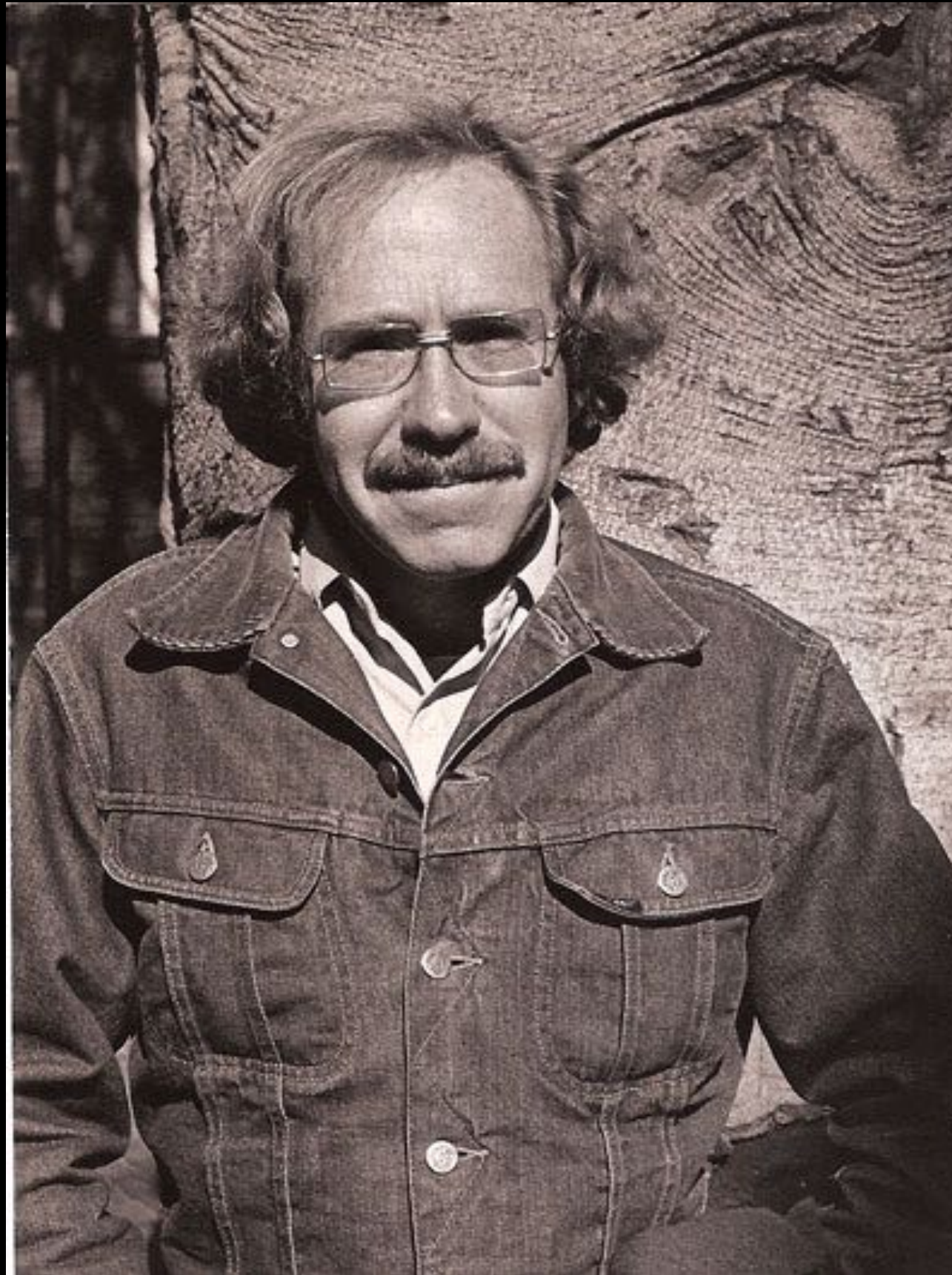
Март 2018

[@nikitonsky](#)

Модель



Stuart E. Dreyfus



Hubert L. Dreyfus



16

LEVEL[#]

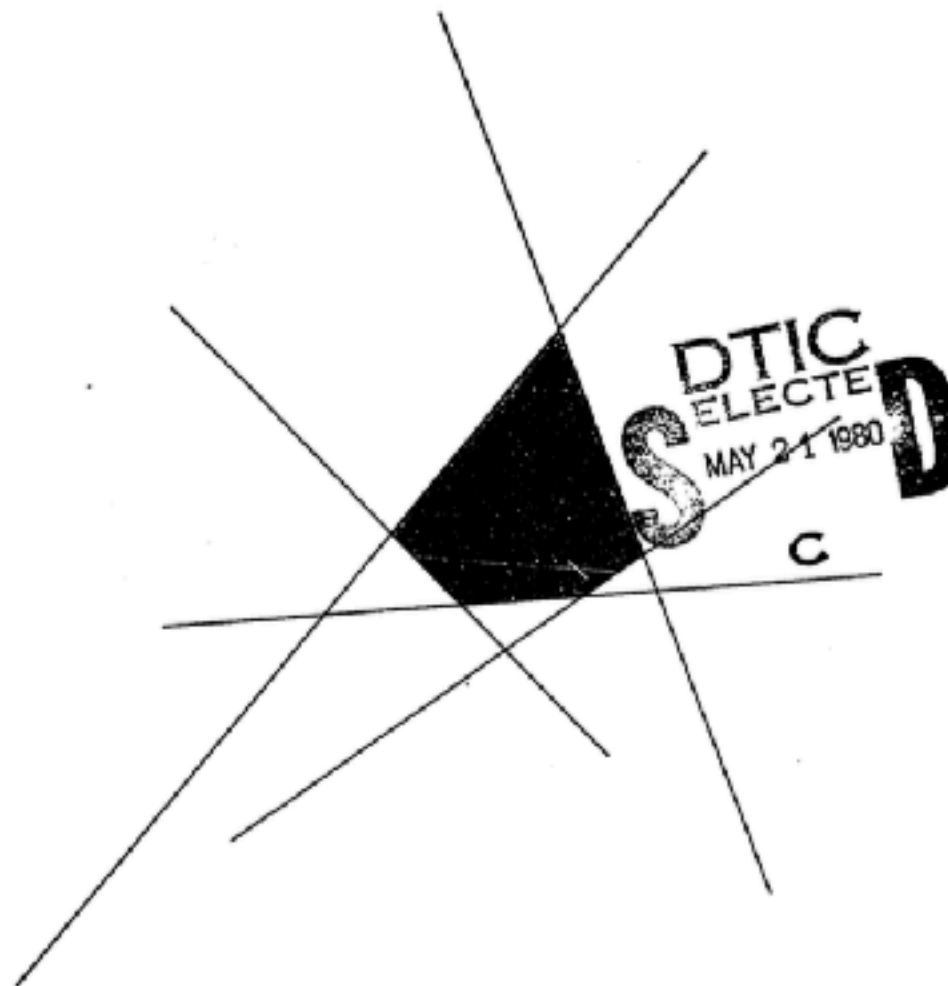
ORC 80-2
FEBRUARY 1980

(12)

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

by
STUART E. DREYFUS
and
HUBERT L. DREYFUS

ADA084551



OPERATIONS
RESEARCH
CENTER

This document has been approved
for public release and sales in
distribution is unlimited.

ORC FILE COPY

UNIVERSITY OF CALIFORNIA • BERKELEY

80 5 19 102

Эксперт

Специалист

Компетентный

Продолжающий

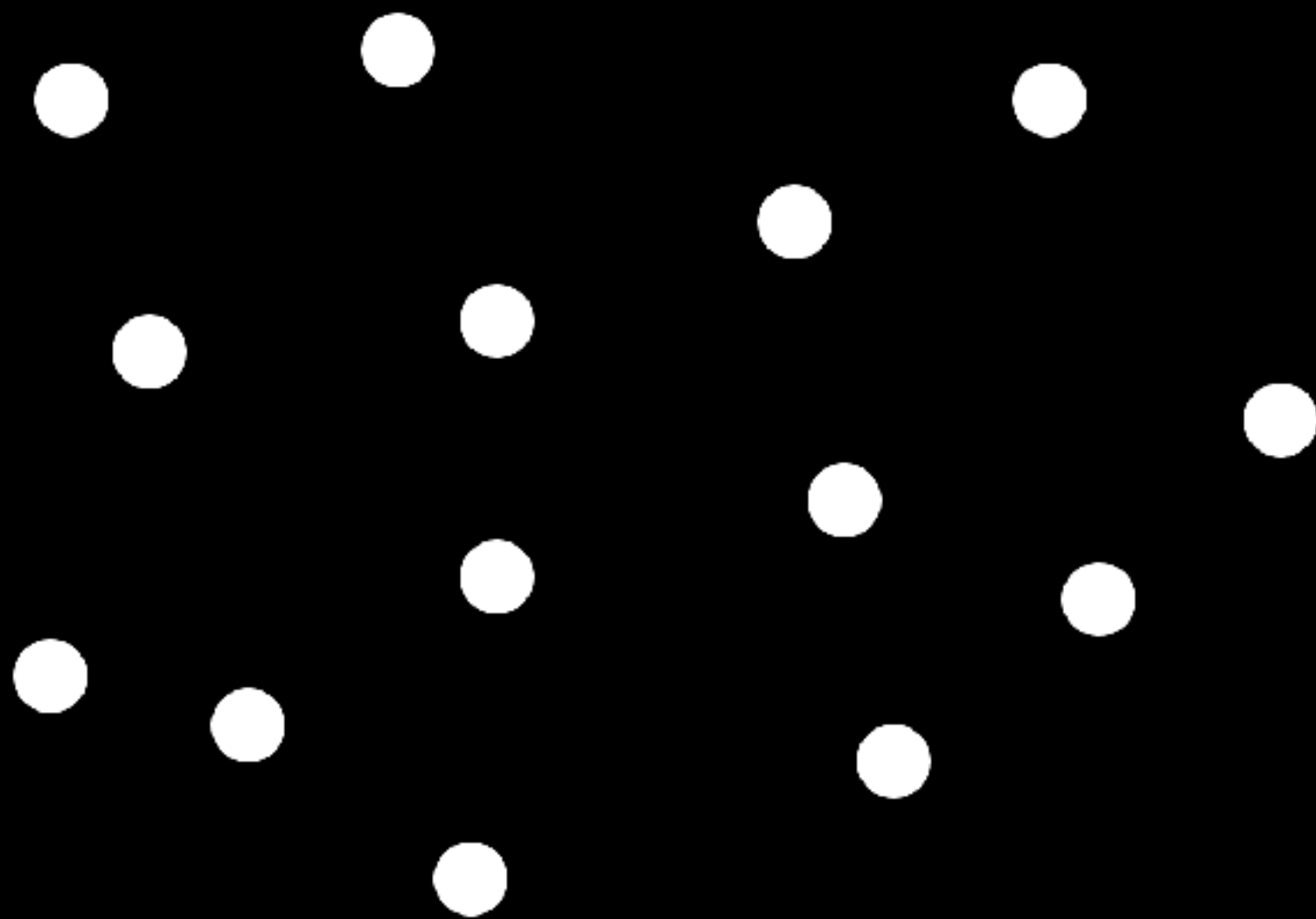
Новичок

1. Новичок



1. Новичок

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





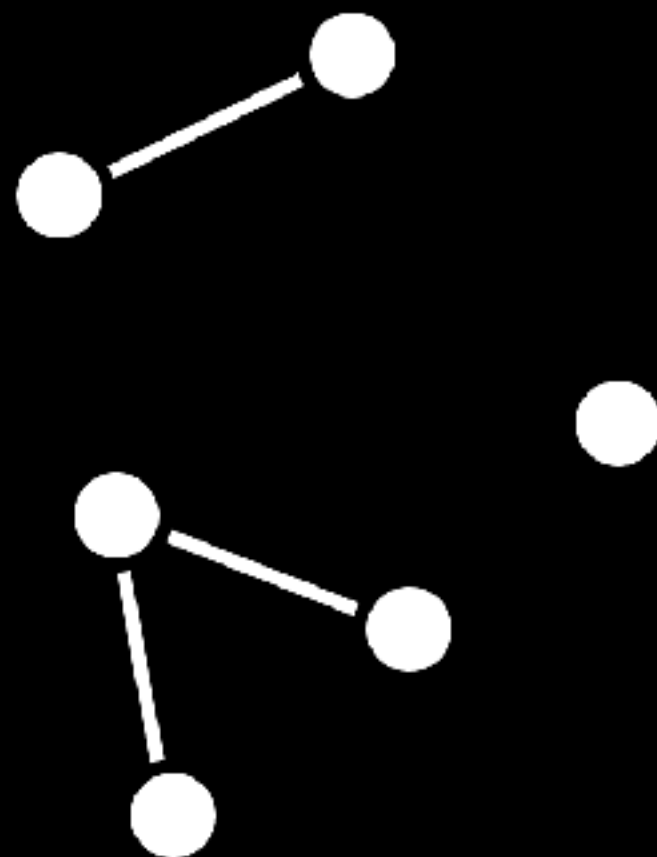
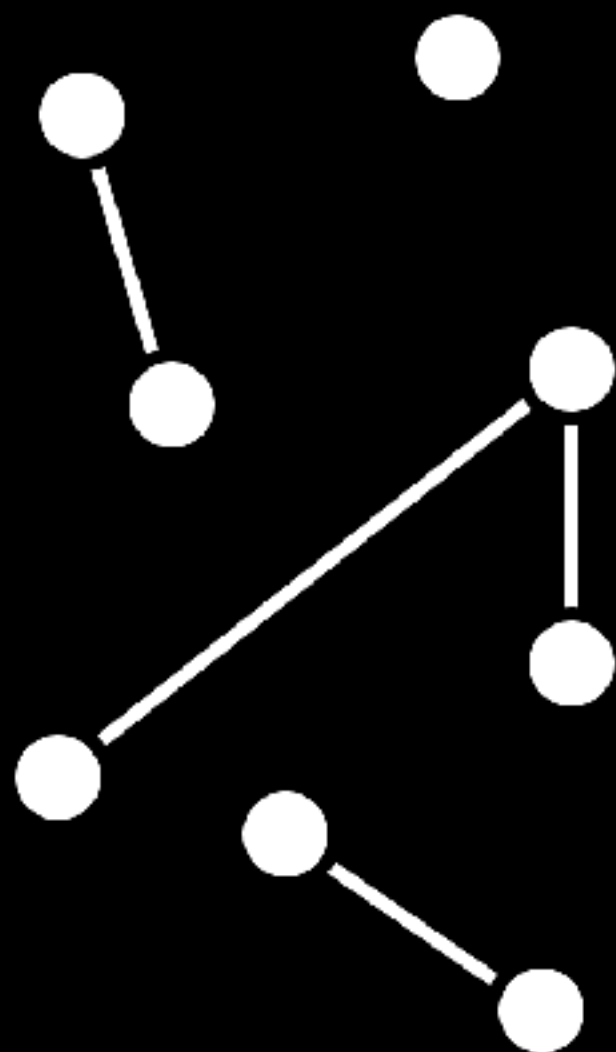
2. Продвину́тый новичок

2. Продолжающийся



2. Продолжающий

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



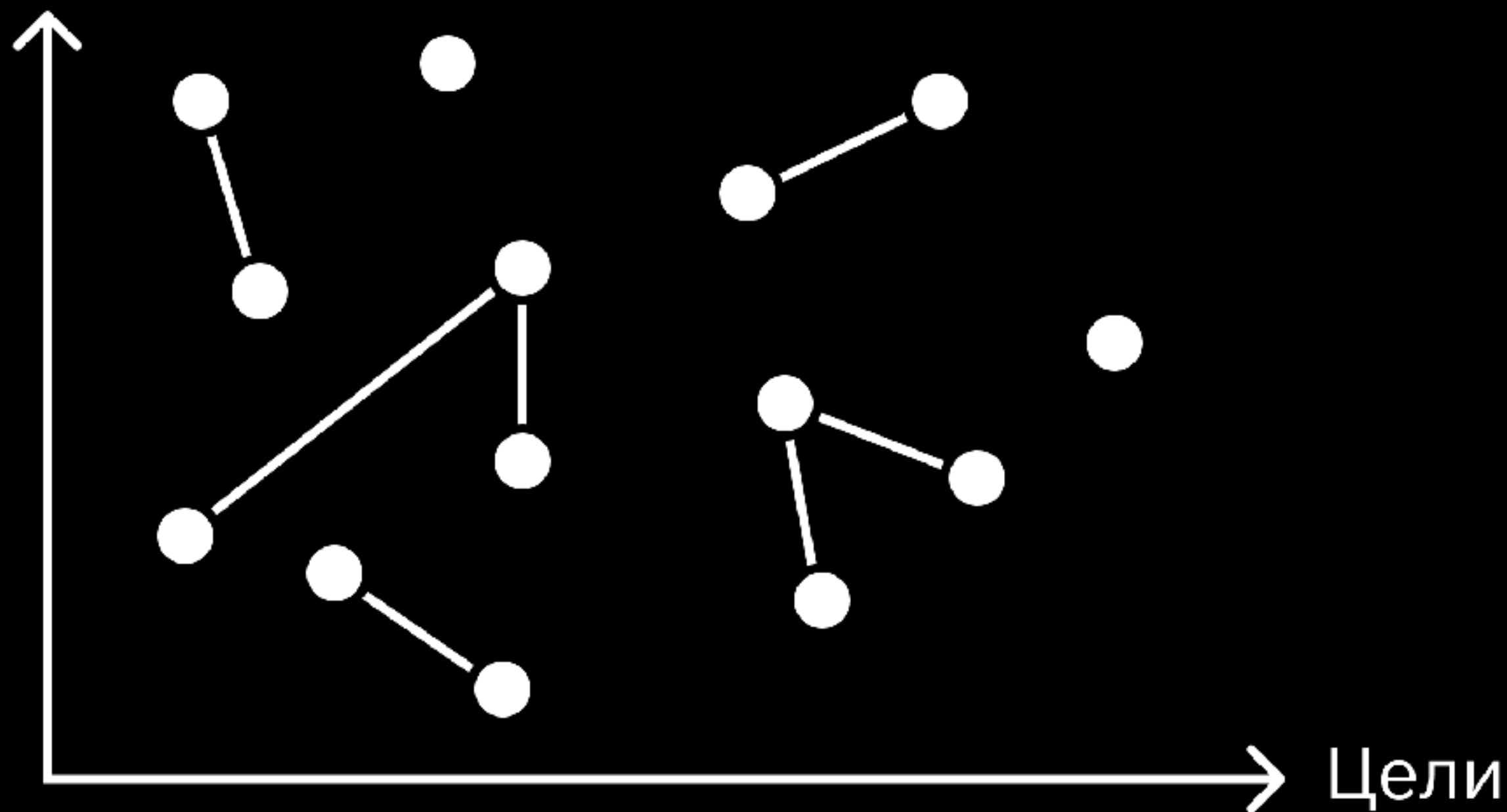




3. Компетентный

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

Важность



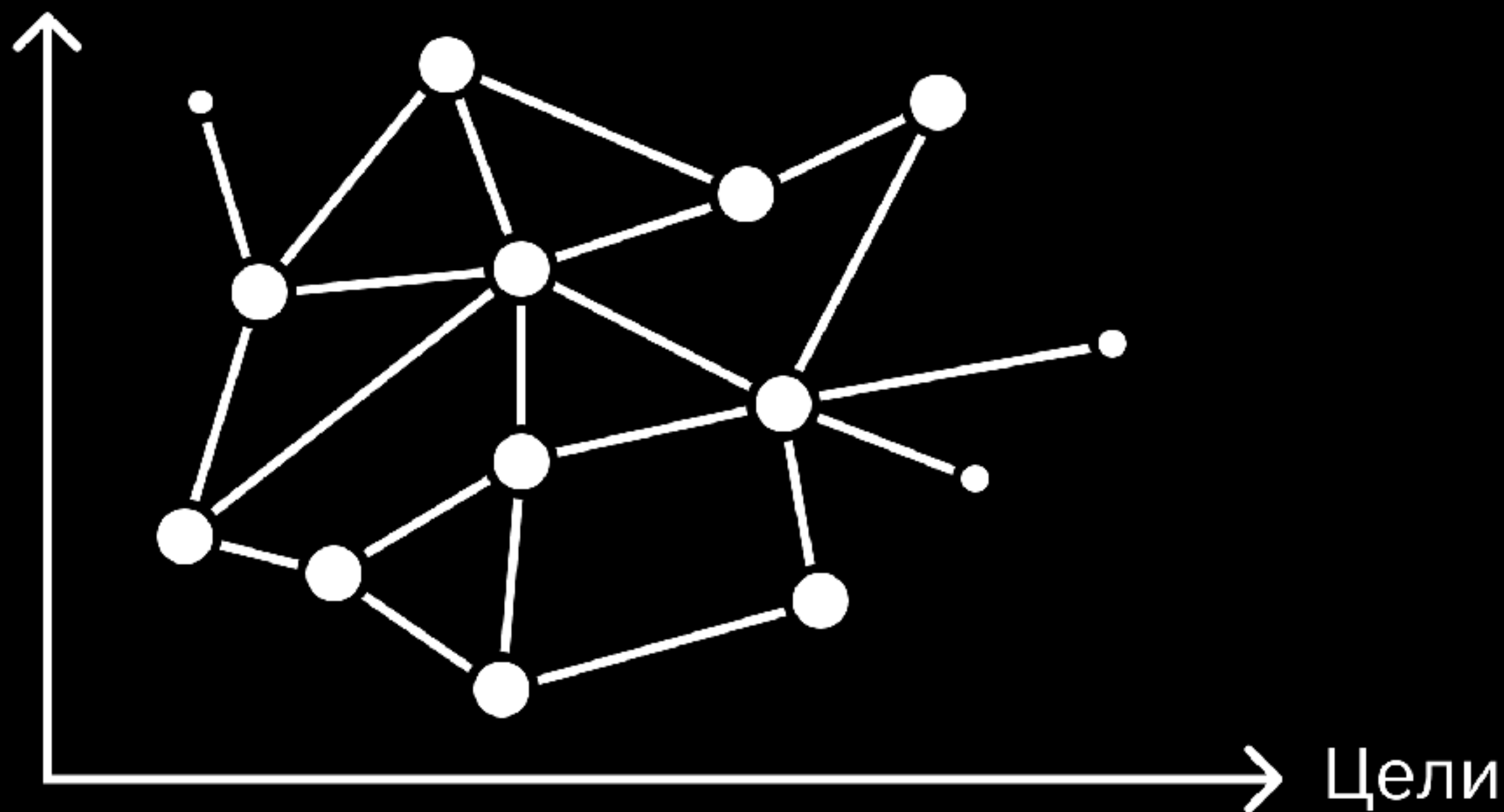




4. Специалист

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

Важность



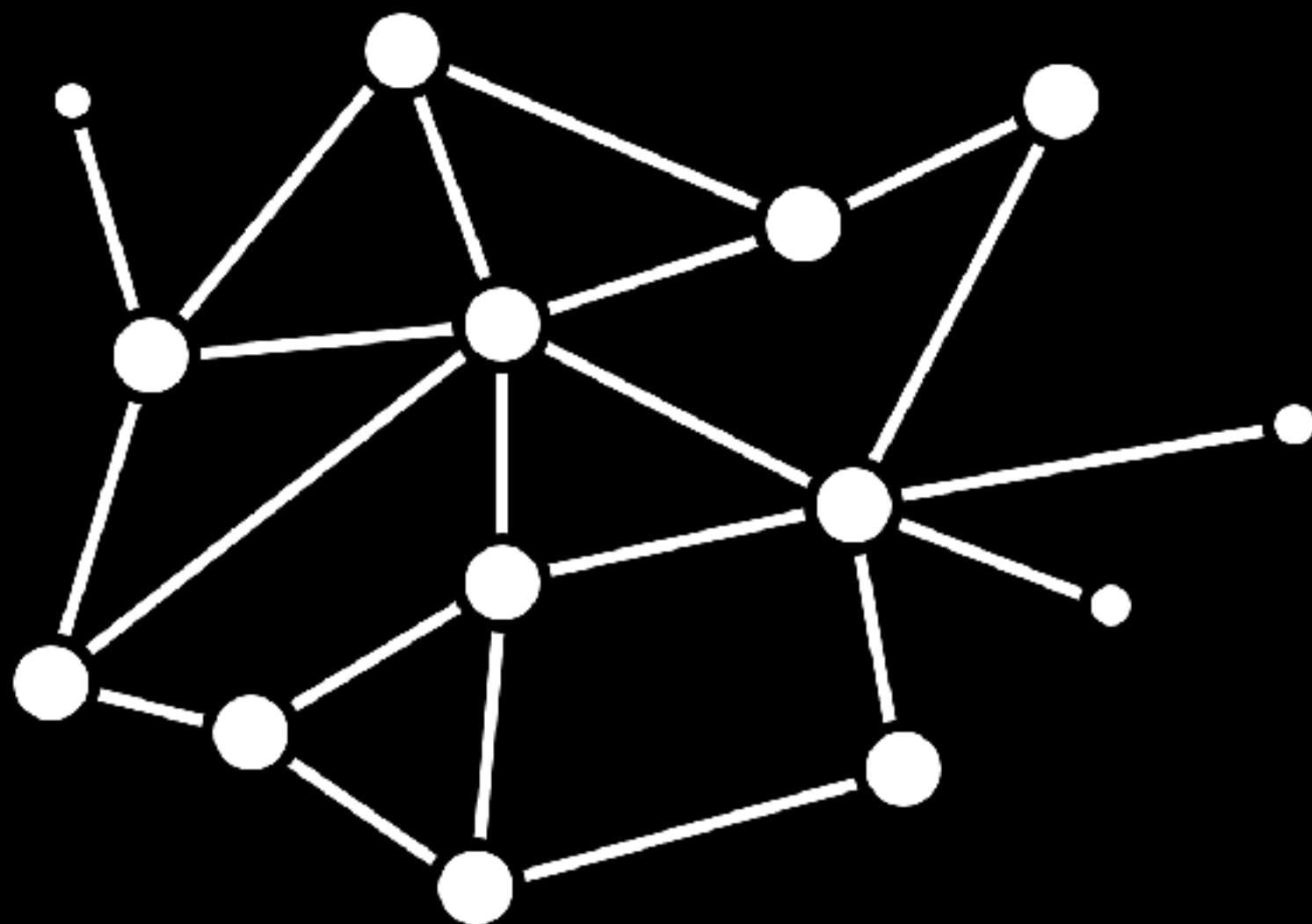
ДТВ





5. Эксперт

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







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 least 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

Правила





- 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.

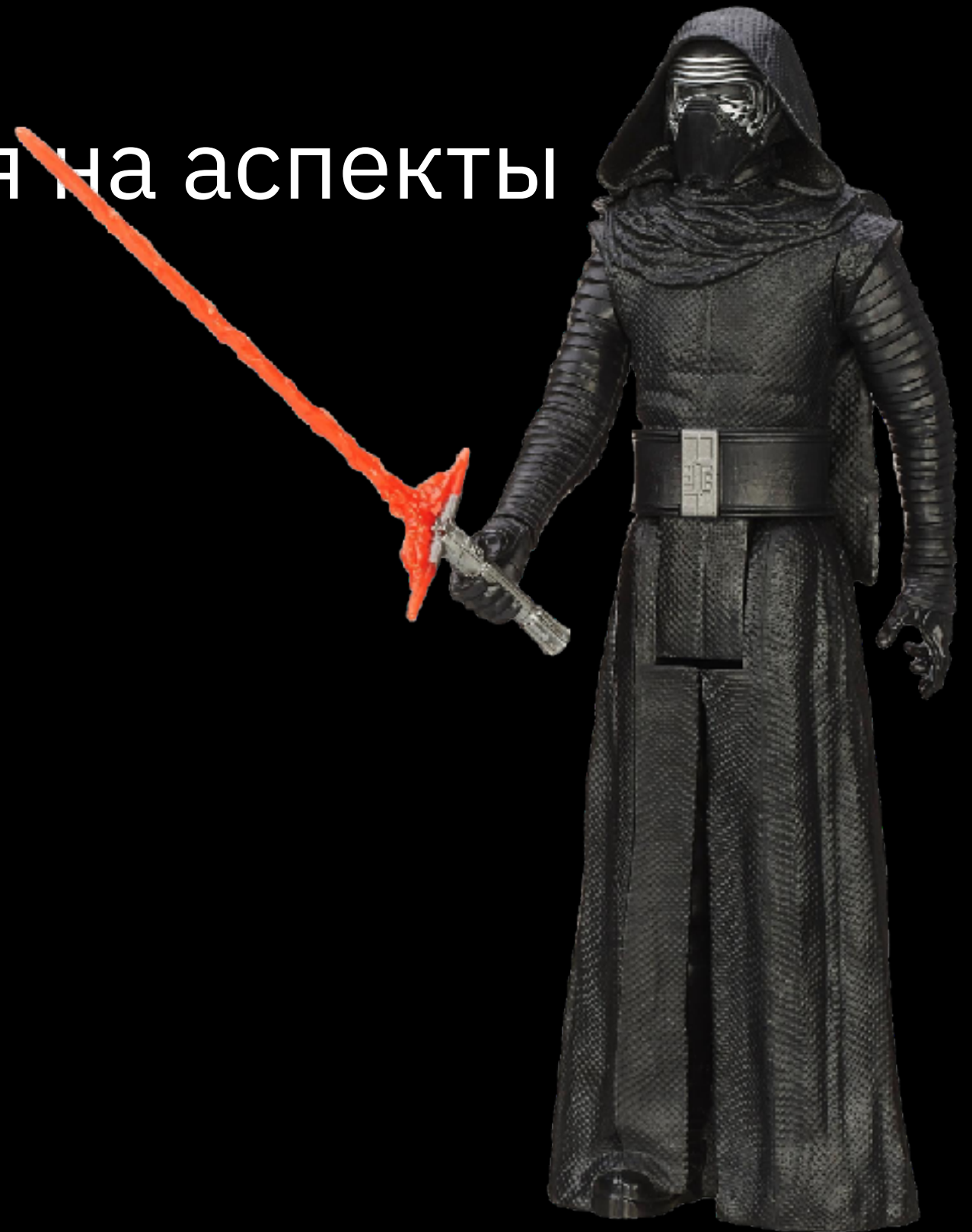


Если в процессе борьбы один из соперников обзавелся лишней фигурой или пешкой, то говорят, что у него **материальное преимущество**.

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 произошёл **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



- Разбираются сами





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

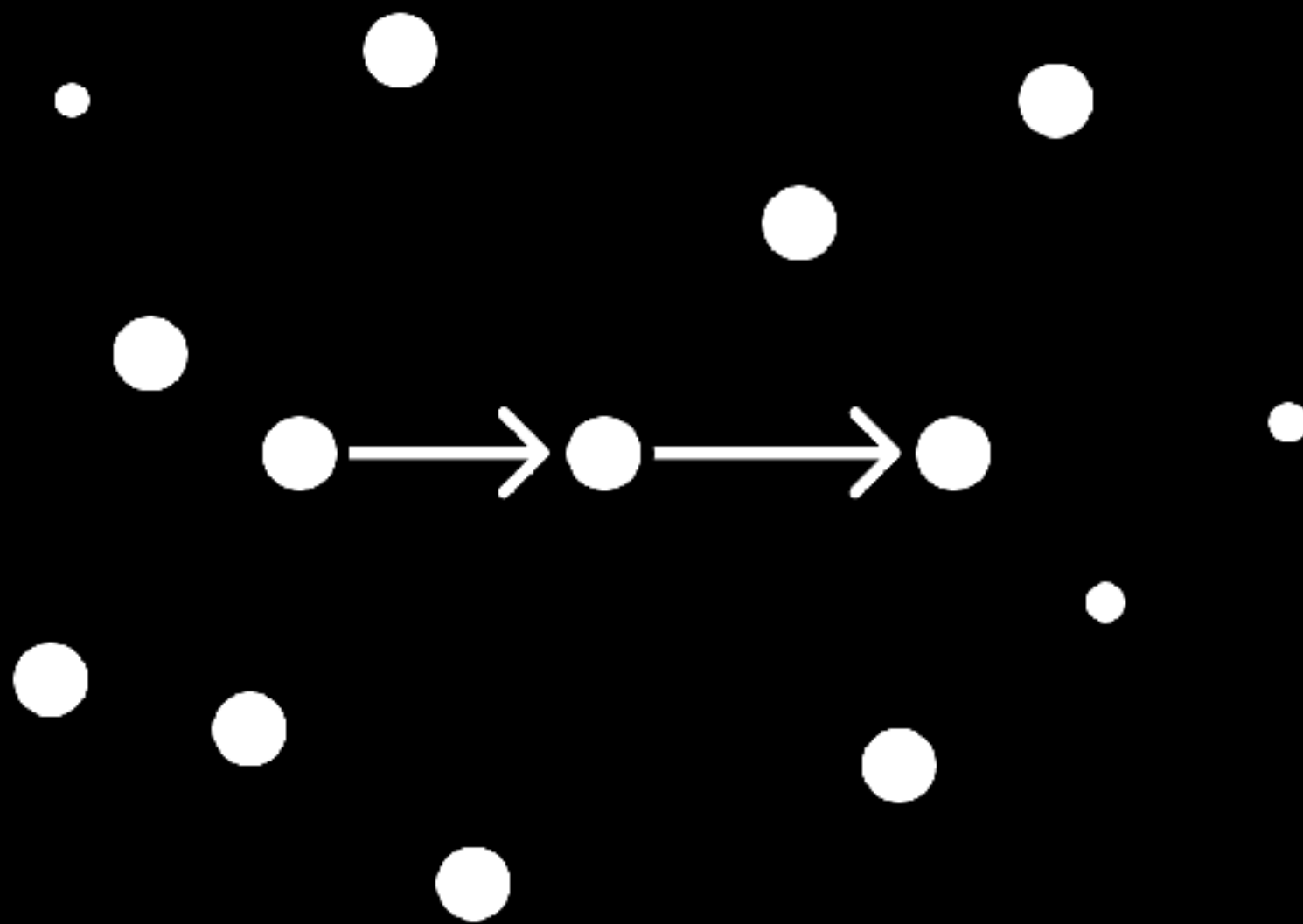
♡ 424 💬 83 people are talking about this

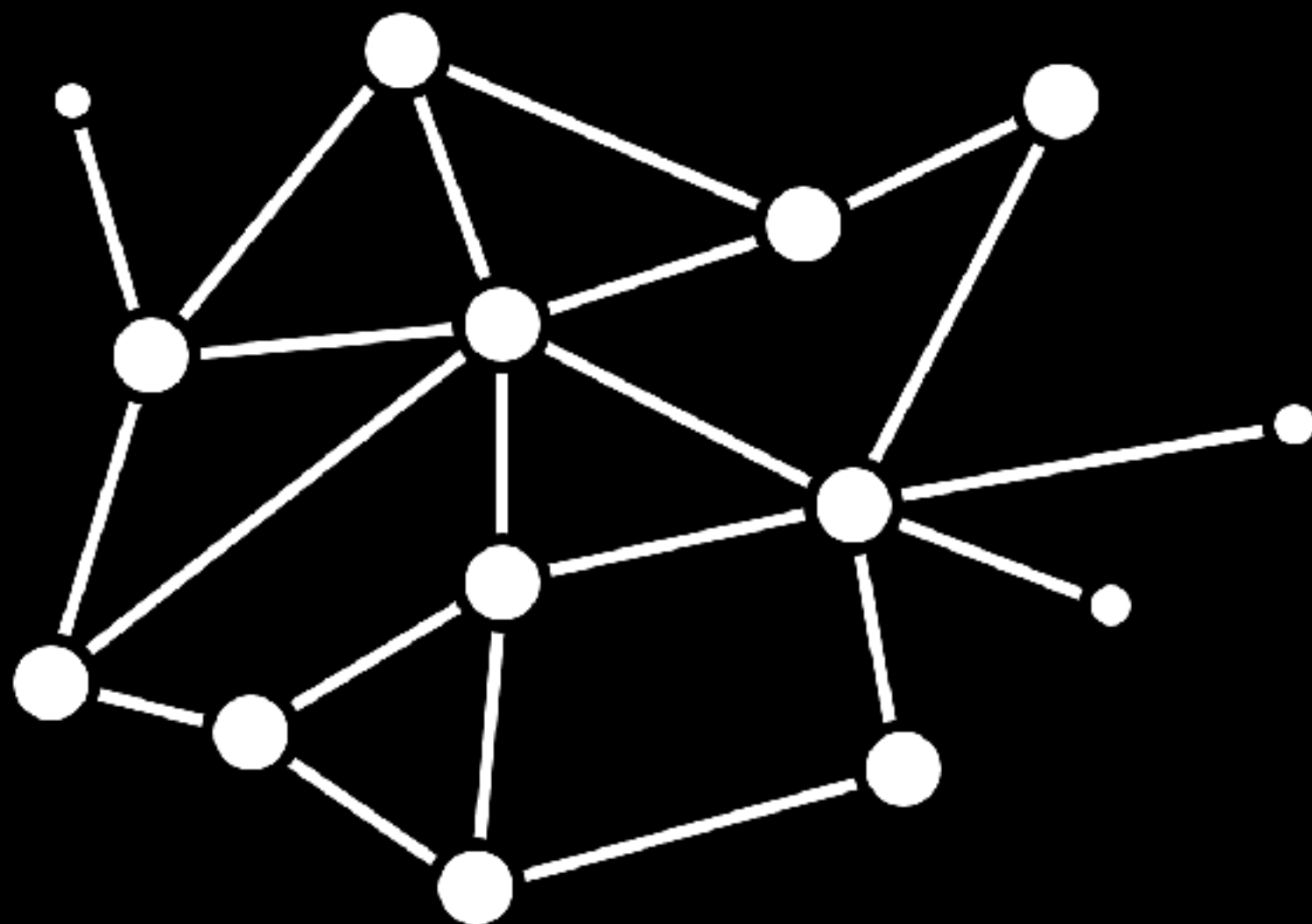
Спаривание



VS



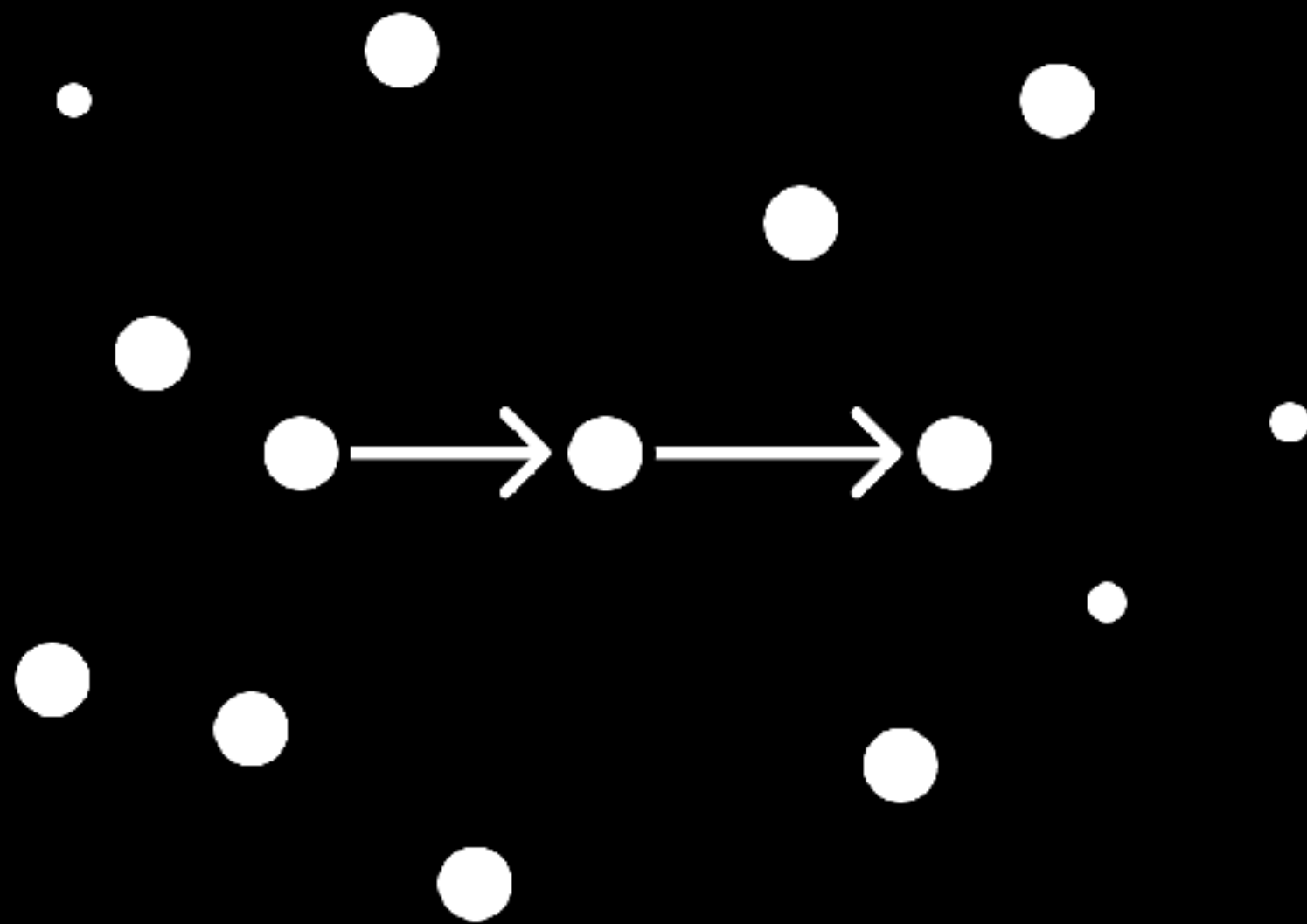






VS







Bob Marshall

@flowchainsensei



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

11:29 AM - Sep 14, 2017

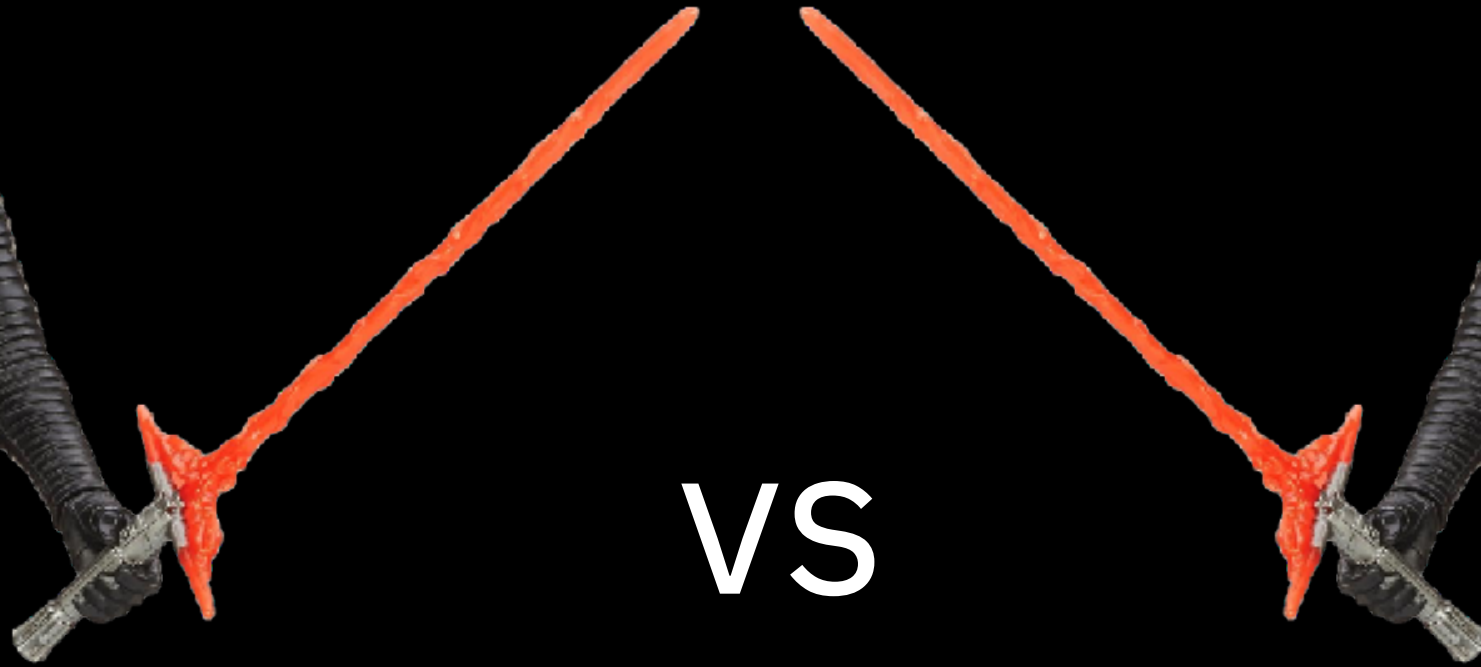


80



63 people are talking about this





VS





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 this

— Generic!



— Flexible!



— Abstraction
layer!

— UI widgets!



— Parser!



— ORM!





VS





VS



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

Test-Driven Development



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

Scenario Outline: Get returns correct DtoToAdditionalDataModelMapper

Given a DefaultDtoToAdditionalDataModelMapperRegistry configured with DtoToAdditionalDataModelMappers:

HiLoMarketDTO	<u>HiloBinaryMarketDtoToModelMapper</u>	
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	<u>HiloBinaryMarketDtoToModelMapper</u>	
TunnelMarketDTO	TunnelBinaryMarketDtoToModelMapper	
OneTouchMarketDTO	OneTouchBinaryMarketDtoToModelMapper	
UpDownMarketDTO	UpDownBinaryMarketDtoToModelMapper	

```

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<String,
        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);
    }

    @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() throws Throwable {
        returned = testClass.getMapper(marketDTO);
    }

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

```



```
public class TestClassLoader {
```

```
    public static <T> T createInstanceOf(String className, Object... initArgs) {
```

```
        try {
```

```
            Class class = Class.forName(className);
```

```
            Constructor constructor = class.getDeclaredConstructors()[0];
```

```
            return (T) constructor.newInstance(initArgs);
```

```
        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
```

```
            RuntimeException("Could not create and instance of class: " + className  
                + "does it have a default constructor? is it public?", e);
```

```
        }
```

```
    }
```

```
    public static <T> T createInstanceOf(String packageName, String className, Object... initArgs) {
```

```
        return createInstanceOf(packageName + className, initArgs);
```

```
    }
```

```
}
```

```
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



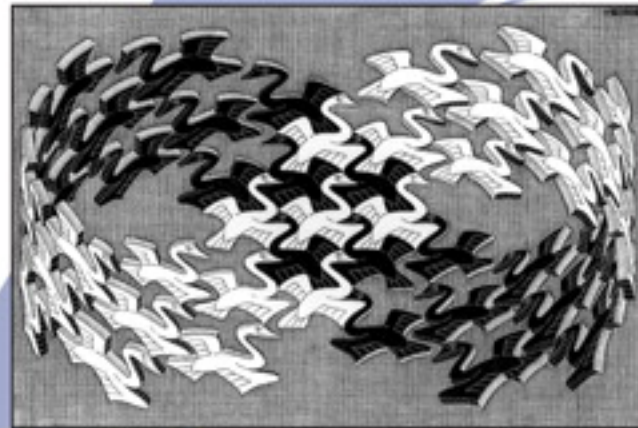
Правило: Тесты — хорошо, больше тестов — еще лучше.
Пишем 100% тестов, и пишем их до кода

OOP Patterns

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

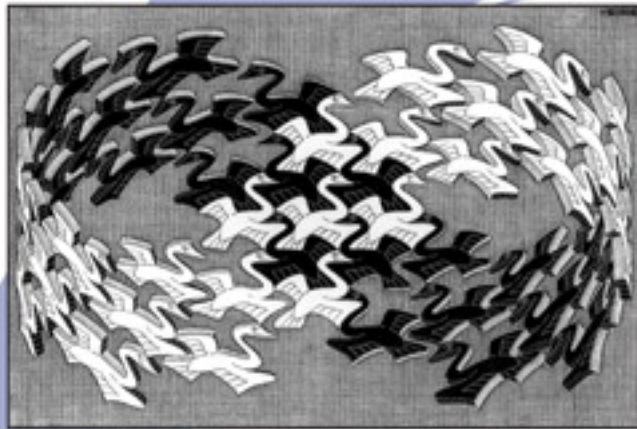


ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch



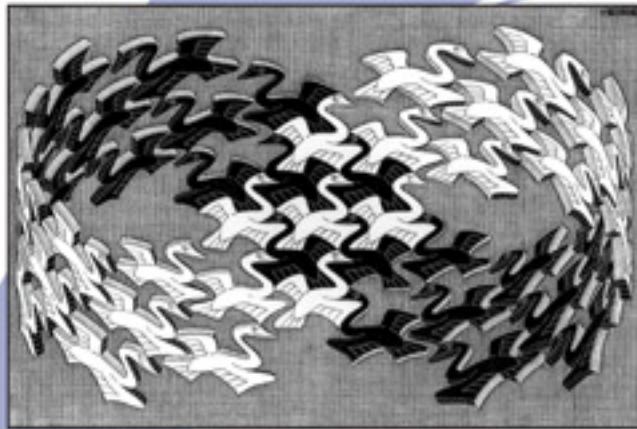
ADDISON-WESLEY PROFESSIONAL COURSE SERIES



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

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





— 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
- Максимумы!

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**





JavaScript: The Good Parts

JavaScript: The Good Parts

Crockford



O'REILLY

JavaScript

The Definitive Guide

Flanagan

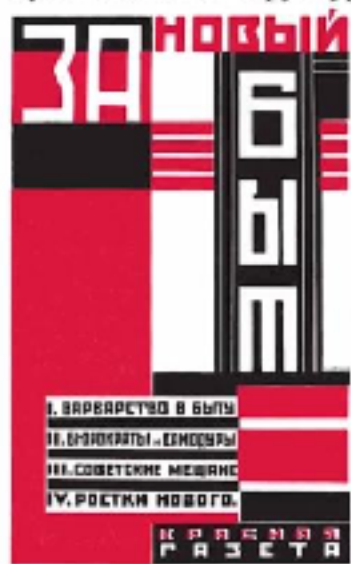
O'REILLY

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

Как выглядит эксперт



Ортогональные структуры, рамки, ассюре



3. Печать, обложка-обложка, Ленинград, «Красная газета», 1918. Подпись для Сурикова: «Судья (тем обложка)»



4. Печать, обложка-обложка, Ленинград, «Красная газета», 1918. Подпись для Сурикова: «Судья (тем обложка)»

Последнее слово изложения будет обществу известно, а пока – о творчестве Сурикова.

Для стиля, во всяком случае суриковского, было существенное – композиция. Приведу только один пример: в основе типичных построений, и конечно он был знаком материалу культуры, давших право. Сами изобретения не фигуративные «картинки», почти всегда статичные. Суриков строил на условных элементах (фигурах) и линиях, при этом они неслись, как бы взаимодействовали друг на друга, неслись, поднимались, обрамлялись, само приращивание – рамки, внутри рамки, этот край походил, условно, графическое пространство, разбитое на квадраты, выделенные линиями, создавая эффект ритмичности и – да-да – также фигуративными элементами, которые Суриков так же успешно культивировал.

Линии оформительские, более-менее ровные, но как выдерживает – и это очень важно – был в ней пристрастие и даже сбалансирован и дисбаланс, так как собственным выделением. Не выдерживало, как правило, выделено, так выделено, прелесть свободного каллиграфа. Но без неё не было бы и Александра Сурикова, хотя он не раз от этого рамки сущности освобождался.

«Командарм 2» – это, в рамках, Работу, поочередно характерная работа, в которой и не было никакого художественного интереса, и который выставлял русский графический искусство XVIII-XIX веков (Николаев, 1978). На эту тему раздала «Конструктивизм и другие» два человека обложки. «Командарм 2», представляла логичную тему, тематика отражает только реальность. Впрочем, в конструктивизме неприятие, а логика и многозначительной акцентной точки – это всего лишь элемент условности, строгий геометрический абстракционизм. С первого взгляда стало ясно, что тогда ещё не было 15-го приращивания, а в 1978 году. Может быть, это и есть случай настоящего неконструктивистского арт-дизайна.



5. Печать, обложка (Бриганти) и обложка обложки. Москва, 1920. Первый раз в издательской сфере использовался принцип «структурной» связи, что характерно для конструктивизма и, конечно, одной формой буквы 2, производимой линией, своей симметрией относительно горизонтальной оси. Типичная структура строки, которая в последующем является основой для работы по конструктивизму, которая, конечно, лучше других, только удалось добиться.



- 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](#)