

Паттерны и практики написания кода



Бритва Оккама, KISS, YAGNI, BDUF

бритва Оккама

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

принцип **«Не следует множить сущее
без необходимости»**

Уильям из Оккама (XIII век)

ключевые задачи
принципа

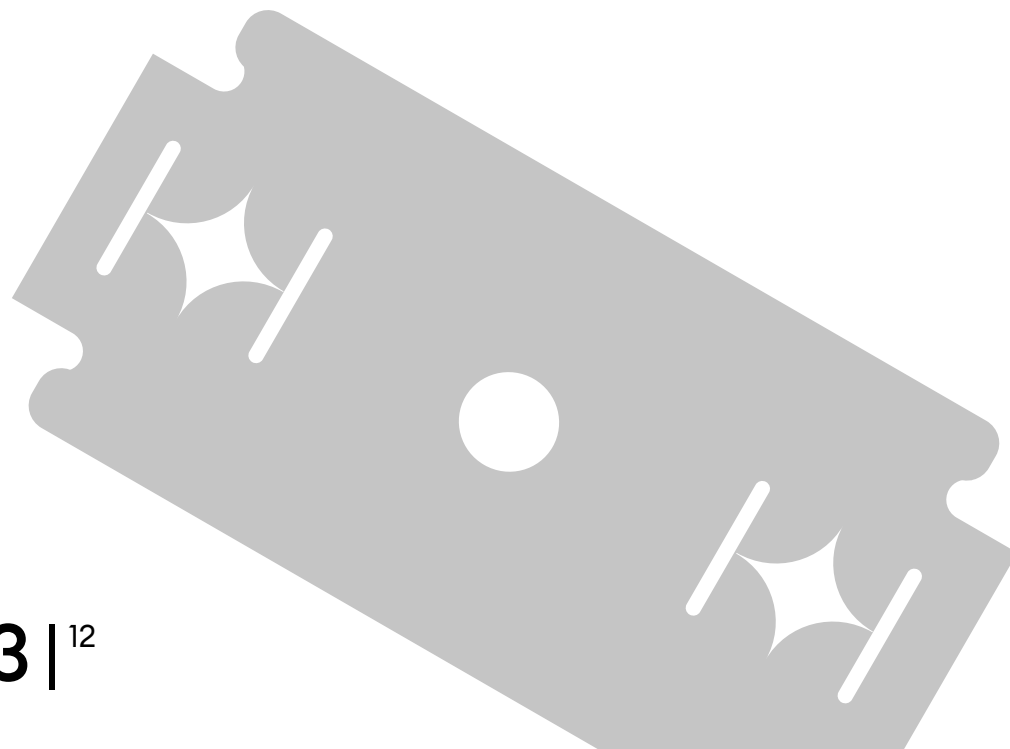
- **стремиться к оптимальности в суждениях →**
искать наиболее простые пути нахождения истины
- **выбирать наиболее краткое объяснение среди**
суждений с одинаковыми результатами → «бритва»,
которая отбрасывает лишнее

заметка *подход не запрещает сложные объясне-
ния, а лишь рекомендует порядок рассмо-
трения гипотез*

*в случае программирования суть принципа
в стремлении уменьшить код и сущности*

паттерны и практики написания кода

avito.tech



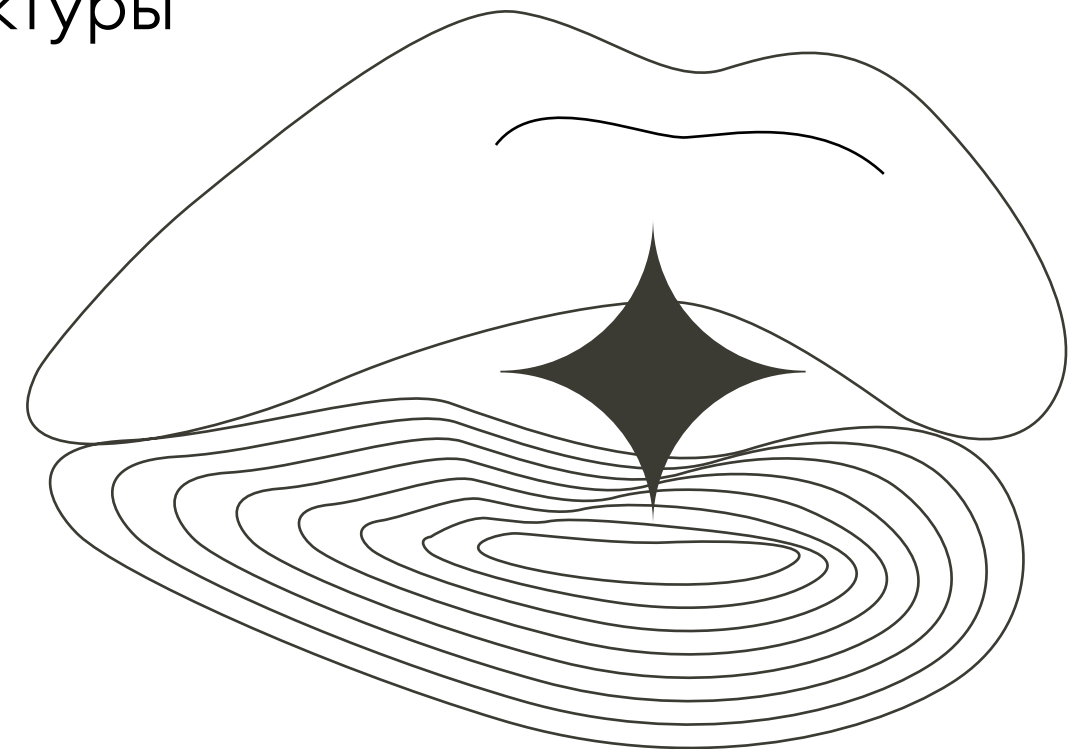
KISS. Keep It Simple Stupid

(«делай это проще, дурачок» → упрощай)

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

ключевые задачи
принципа

- ⚡ **программы без усложнений работают лучше.**
В них меньше багов, они легко читаются – коллегам будет несложно погружаться в новый код
- + **простота системы декларируется в качестве основной цели и ценности.** Поиск уязвимостей идёт легко, нет запутанной архитектуры и непонятных конструкций



когда нарушается принцип KISS

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

☼ **если учитываются все возможные варианты поведения системы, пользователя и среды.** Это реализация излишней бизнес-логики вместо выполнения поставленной задачи, из-за чего затрудняется тестирование и работа с кодом

○ **при неуместном применении паттернов проектирования.** Пока просто запоминаем эту мысль, к ней мы еще вернемся

◉ **при применении сложных и ненужных технологий.** Всегда ищем более простой путь: для этого уточняем детали поставленной задачи

какие плюсы у принципа KISS

- + простое объяснение является самым верным решением. Помним о «бритве Оккама», где простота – самоцель
- ■ простая система позволяет быстрее разобраться с ней. Именно поэтому выясняем, насколько простое предлагается решение и есть ли вариант попроще
- + не нужно делать решение сложнее, чем исходная проблема. Просто не усложняем

YAGNI. You Aren't Gonna Need It

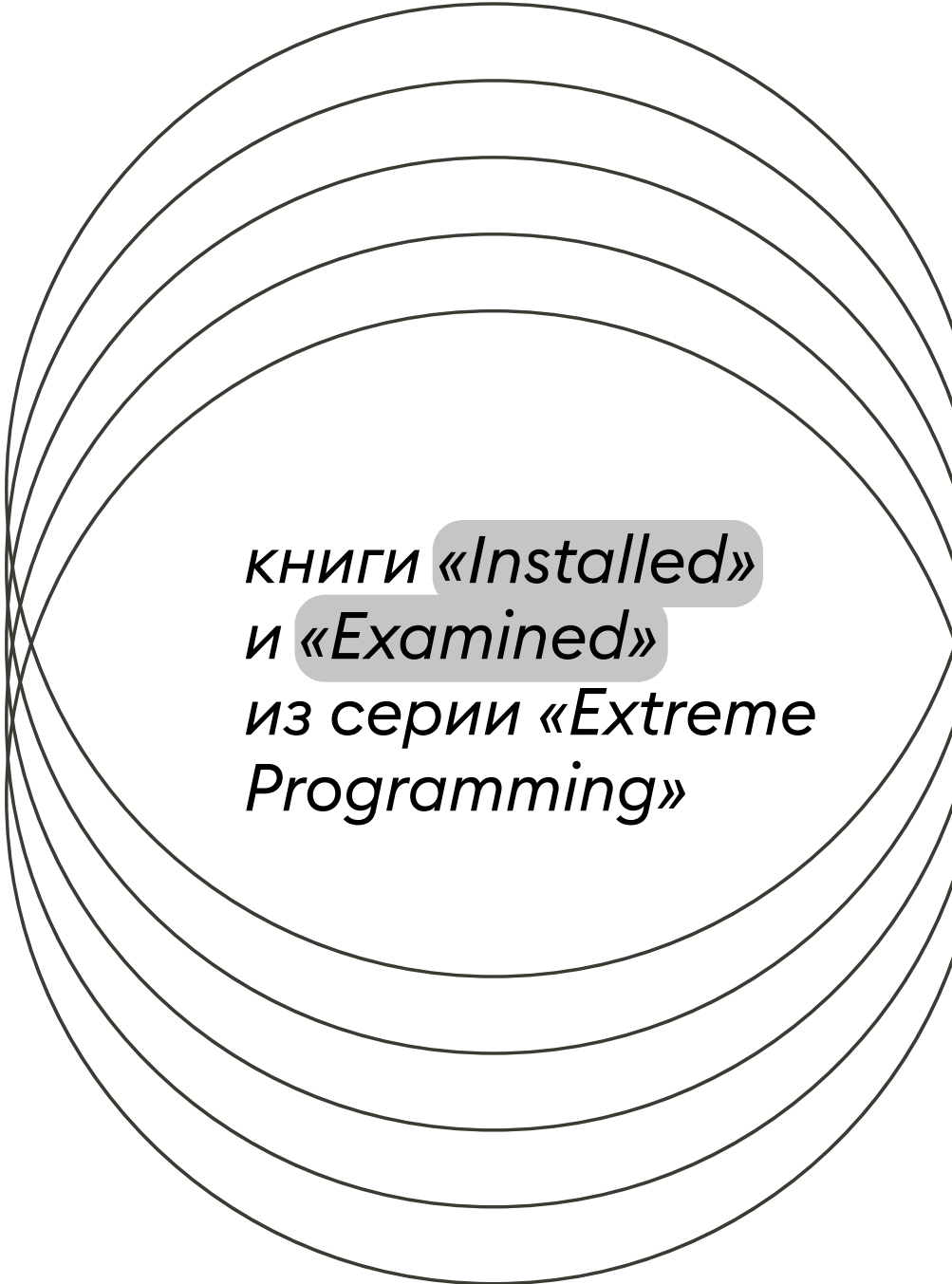
(вам это не понадобится)

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

ОПРЕДЕЛЕНИЕ

Всегда реализуйте только те вещи, которые действительно нужны и никогда то, что лишь кажется нужным → отказ от избыточной функциональности, в которой нет необходимости

YAGNI – это о методологии, в которой подходы разработки являются одной из её частей. Работая по SCRUM или XP, вы автоматически принимаете YAGNI. Принцип охватывает всю компанию: опять же, в отличие от DRY и KISS



книги «*Installed*»
и «*Examined*»
из серии «*Extreme Programming*»

какие проблемы могут случиться, если делать избыточные реализации «на будущее»?

- ✗ **вы потратите время на разработку чего-то одного, а может понадобится совсем другое.** Как и в случае с MVP, подойдёт любая реализация. Но будьте готовы, что его придётся сильно переписывать или начинать с нуля к моменту, когда будут сформулированы требования
- **частично подходящая реализация мешает создать целевую функциональность.** Если сделать реализацию без конкретных требований, в будущем придётся переписывать всё заново или встраиваться в изначально неправильно созданные интерфейсы
- **возникнут сложности в формулировке задачи без конкретной цели.** Пока не будет определено целевое видение, код будет много раз переписываться
- ✦ **качество тестирования упадет из-за абстрактных целей.** Непонятно, что и как тестировать, пока нет приёмочных критериев

сравним KISS, DRY и YAGNI

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

DRY и YAGNI уменьшают сложность системы, но **DRY разделяет систему на компоненты, а YAGNI уменьшает их количество.**

Между YAGNI и KISS есть незаметная, на первый взгляд, разница. **YAGNI делает вещи как можно проще, а KISS помогает искать простые решения.** Сверхрезультат для YAGNI – не делать решений вообще.

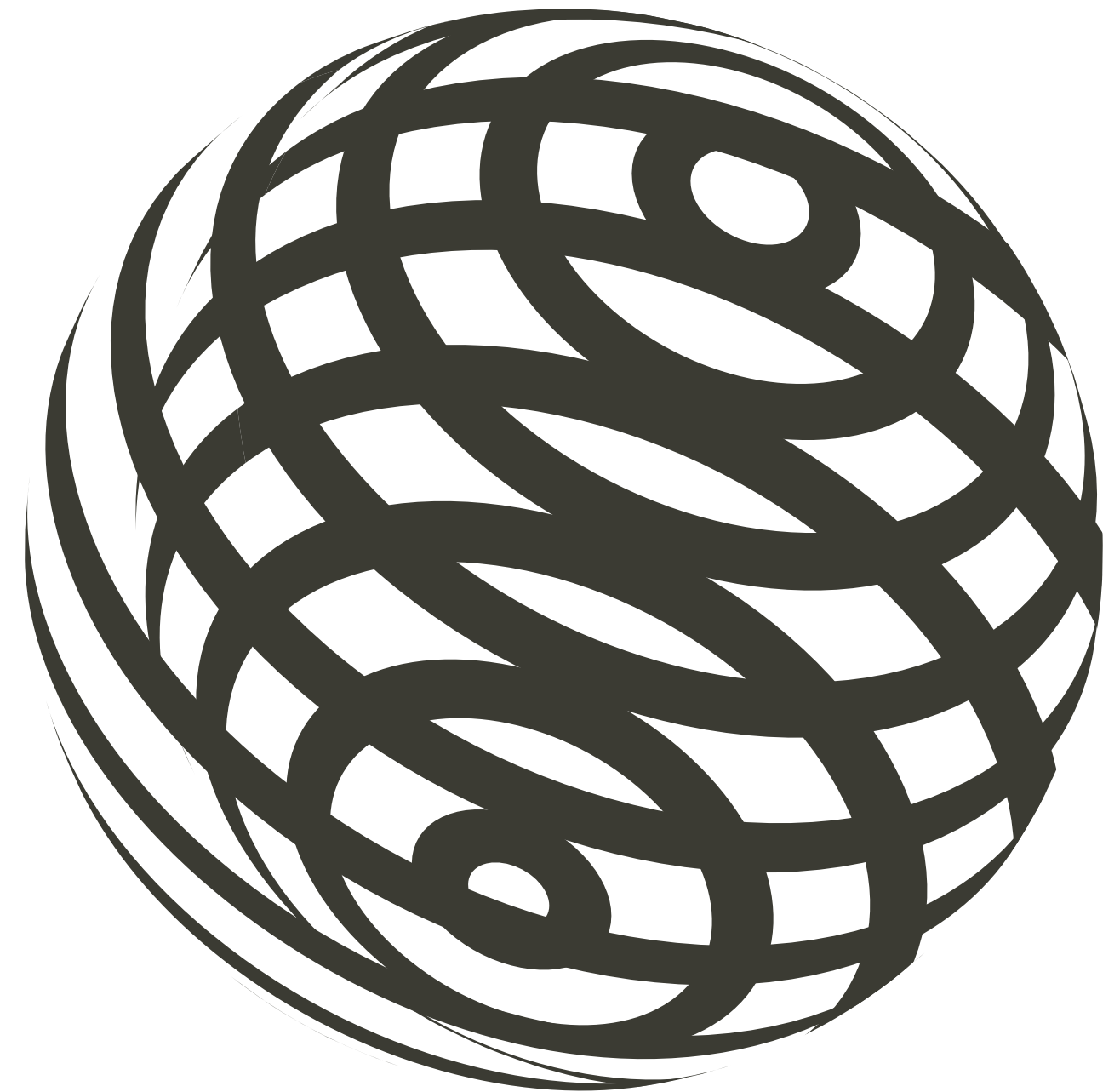
BDUF. Big Design Up Front

(масштабное проектирование прежде всего)

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

BDUF – полная противоположность YAGNI. **Его цель — создание архитектуры, которая покрывает все текущие и будущие случаи использования ПО**

Оба YAGNI и BDUF – методологические паттерны, но YAGNI пришёл из XP, а BDUF из Waterfall.



сравним BDUF и YAGNI

конспект 2
Бритва Оккама,
KISS, YAGNI, BDUF

Принцип YAGNI говорит об оптимальности написания минимально необходимого кода для работы приложения, а BDUF заставляет закладывать функциональность на будущее. Выбор подхода зависит от стратегии и методологии компании, в которой вы работаете.

 **avito.tech**