Познание нового. Тестирование и развёртывание приложения.

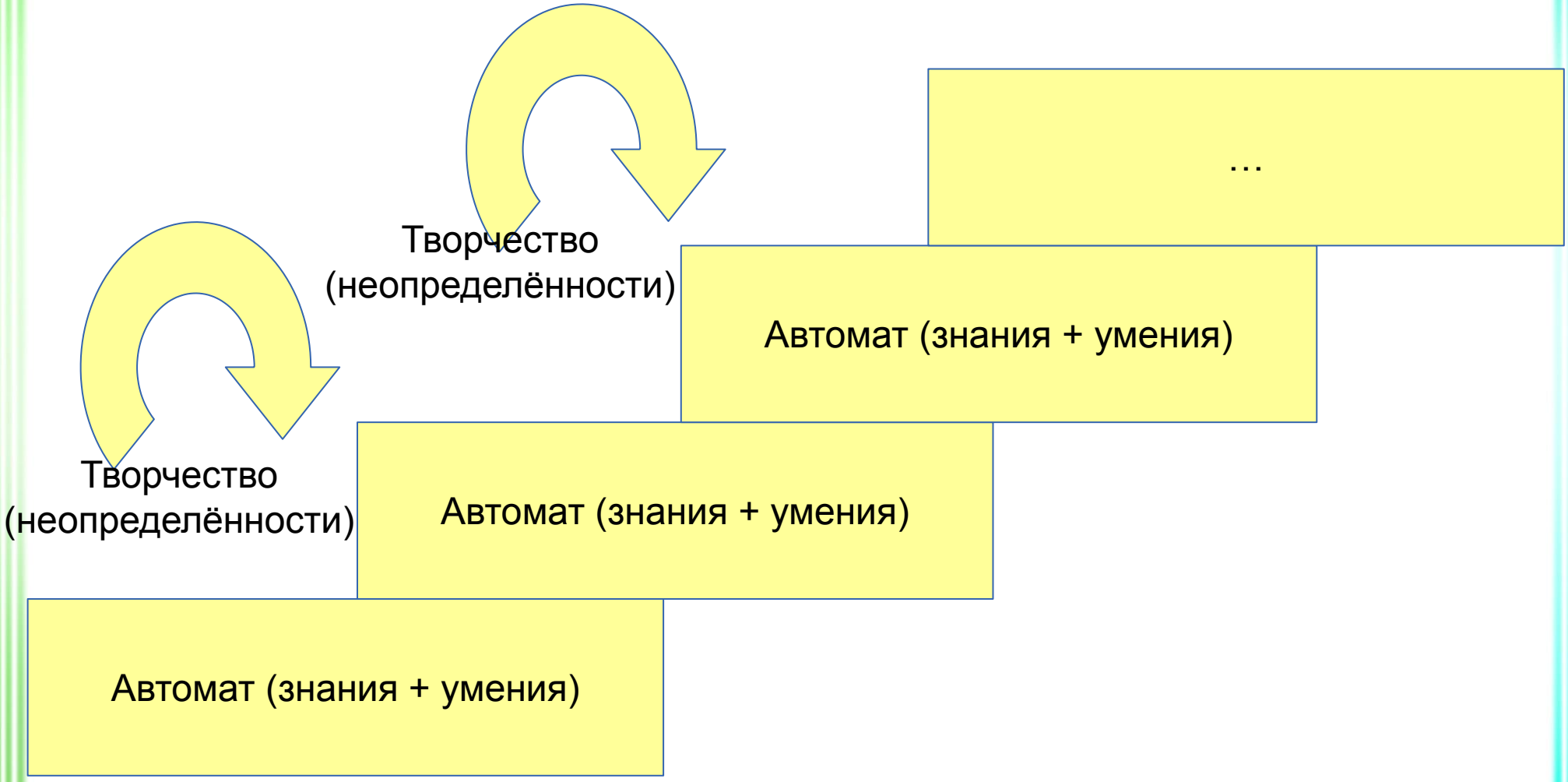
Личностное развитие

Как познавать новое?

Принцип

Саморазвитие = автомат + **творчество**

Ступени саморазвития



Неопределённость — отсутствие информации о чём-либо

Информационная энтропия

мера неопределённости или непредсказуемости информации, неопределённость появления какого-либо символа первичного алфавита

$$H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

Сколько информации должно прийти, чтобы неопределённость снизилась до 0.

Разрешение неопределённости

$H(x) \rightarrow 0$

Какие бывают неопределённости

Неясность взаимосвязи между непосредственно управляемыми и контрольными параметрами нашего процесса развития

Сравнение понятий между собой, чем одно отличается от другого

Что-то непонятно в условиях задачи, чаще всего термин либо набор терминов

Что-то не получается в процессе решения, возникают различные нештатные ситуации

…


ПОМОГИТЕ!!!
As Crônicas de Wesley
ЧТО СЛУЧИЛОСЬ?
У МЕНЯ ОГНЕТУШИТЕЛЬ ЗАГОРЕЛСЯ!!!
НУ ТАК БЕРИ И ПОГАСИ ЕГО С ПОМОЩЬЮ ОГНЕТУ....
...
ВОТ БЛИН!

Уровни развития компетенции

Репродуктивный → Прикладной → Творческий

Этапы развития компетентности

Творчество

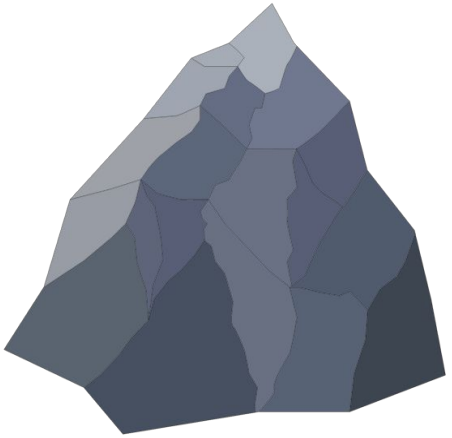
Автомат

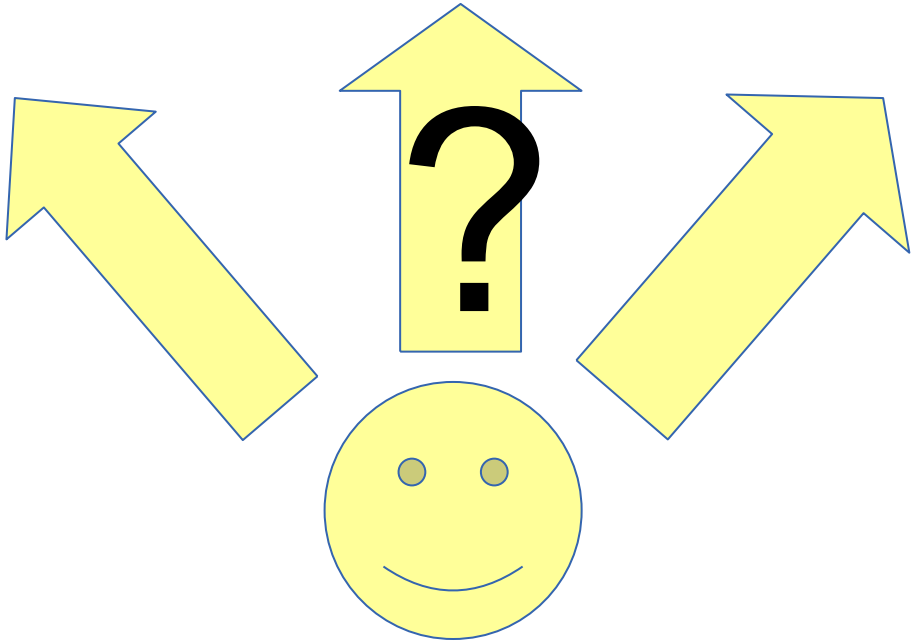
Что такое творчество?

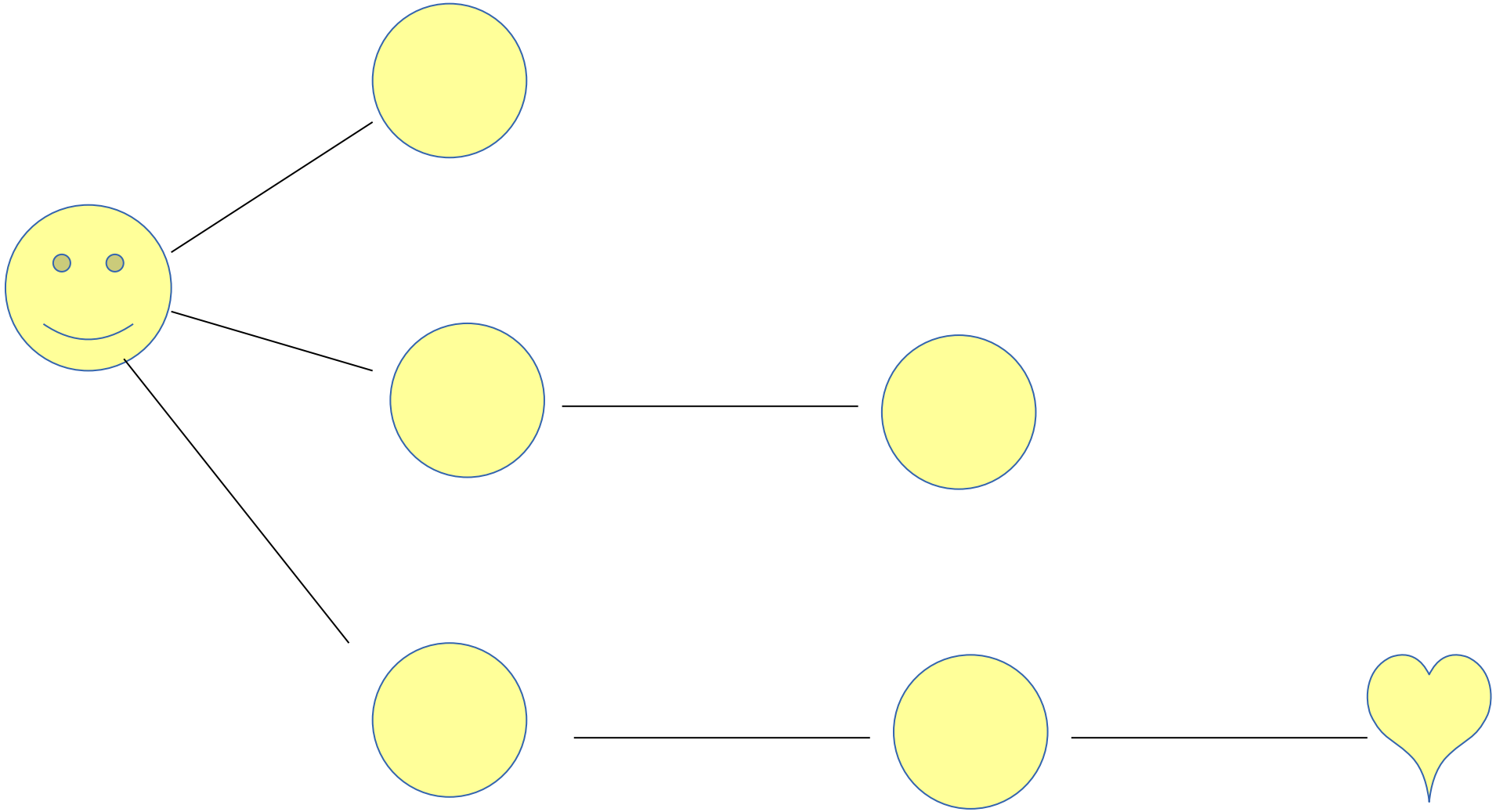
последовательность действий по разрешению неопределённостей в русле достижения поставленной цели

Раскрепощение психики

Есть только два способа прожить свою жизнь.
Первый – так, будто никаких чудес не бывает.
Второй – так, будто все на свете является чудом.







М

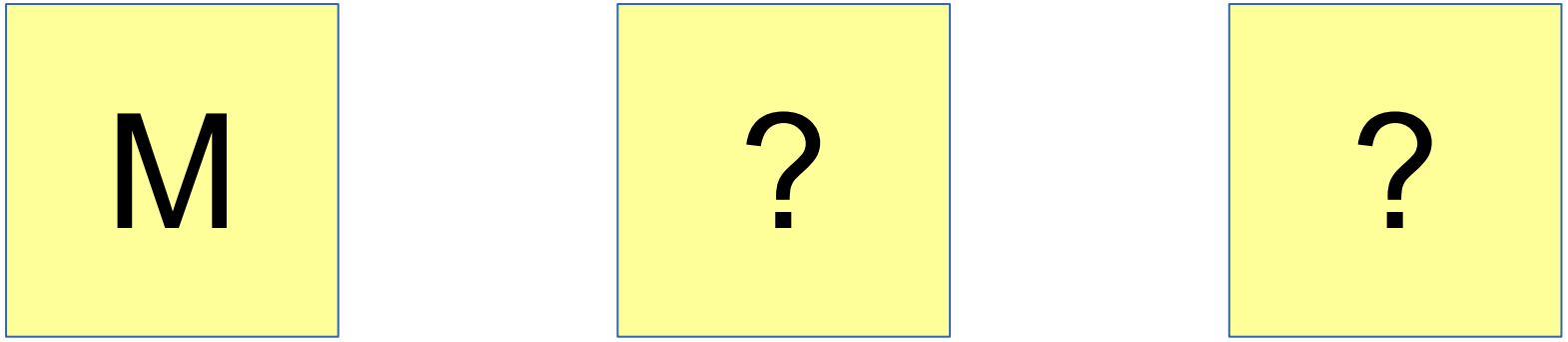
И

Р

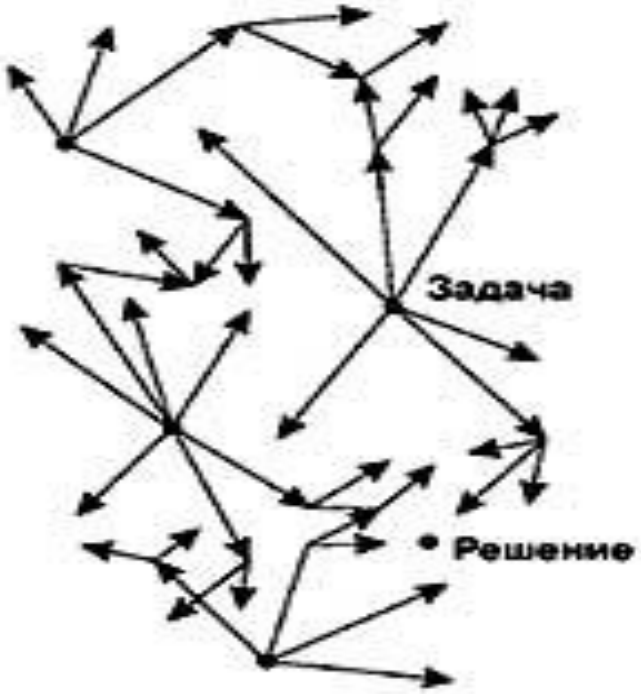
М

И

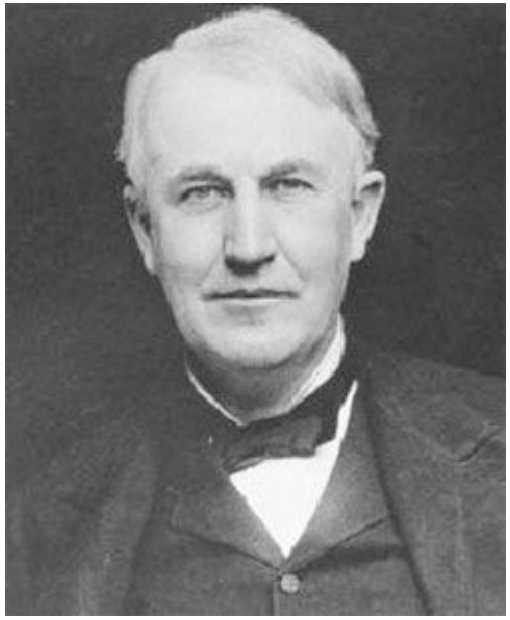
?


M
?
?

Метод проб и ошибок

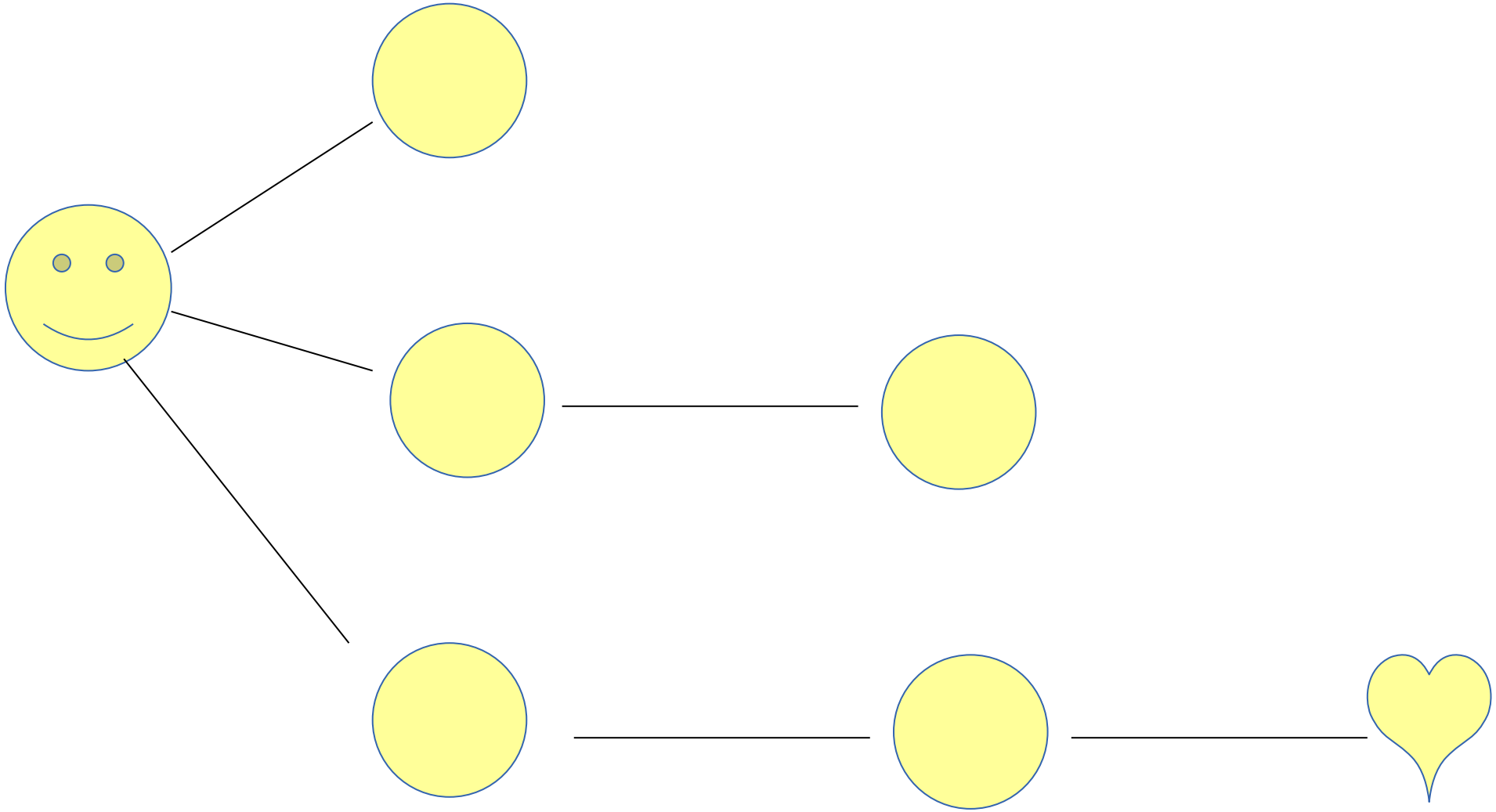


- Метод проб и ошибок



- Томас Эдисон:
  10 тыс. попыток

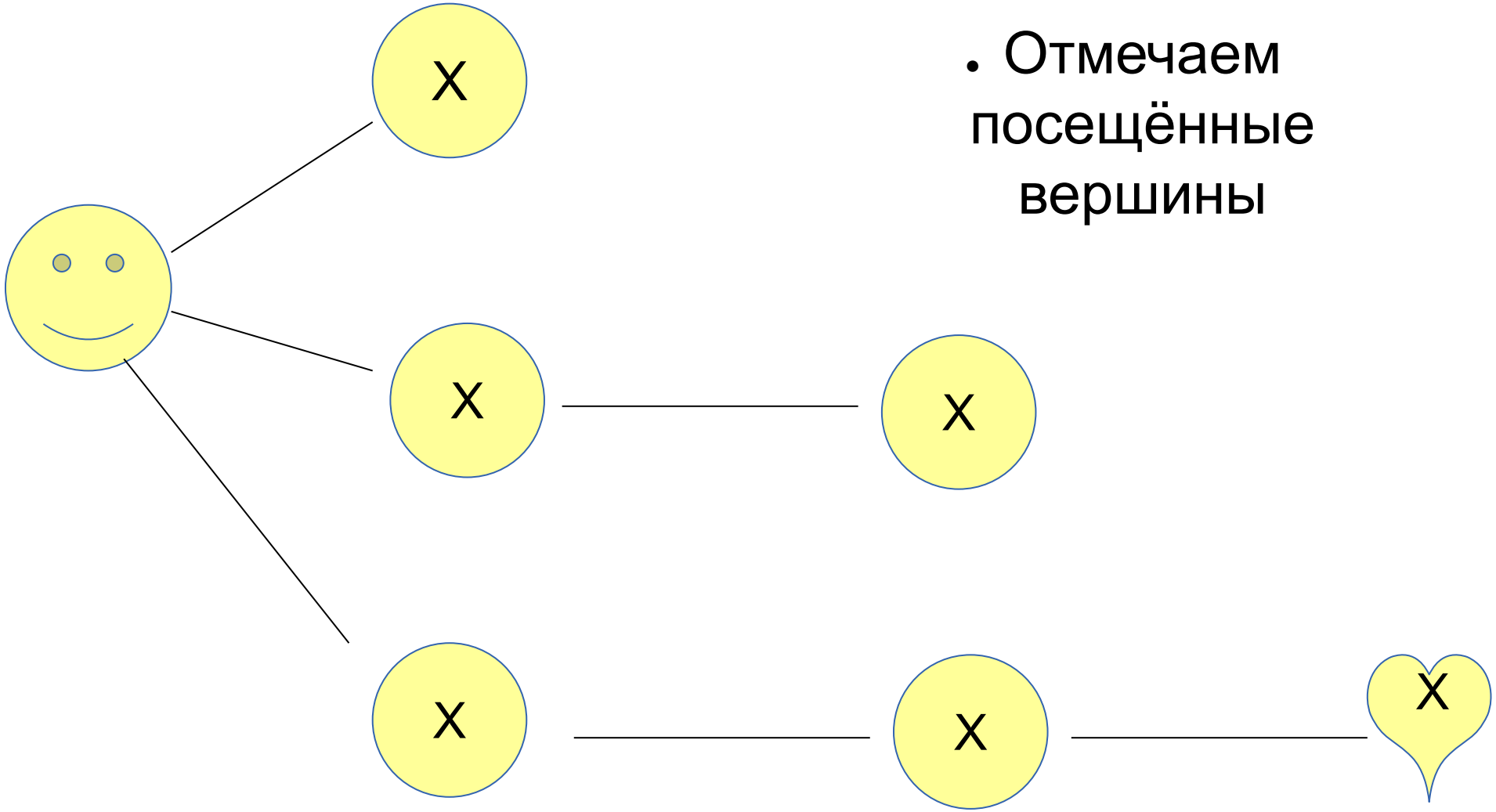
Алгоритм полного перебора



- Теорема Гриса

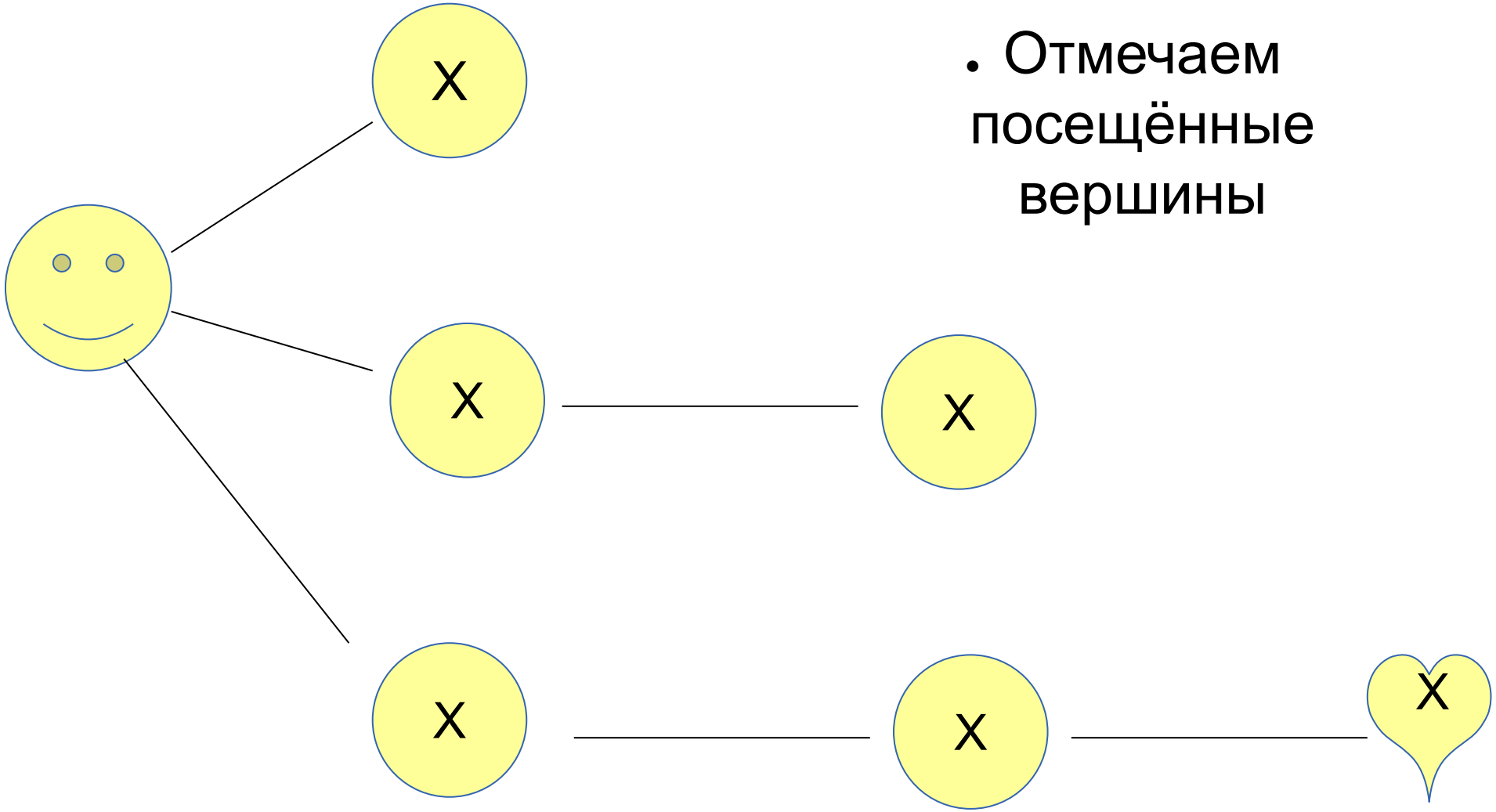
Если существует кратчайший путь к целевой вершине, он будет найден методом полного перебора

Поиск в глубину



- Отмечаем посещённые вершины

Поиск в ширину



- Отмечаем посещённые вершины

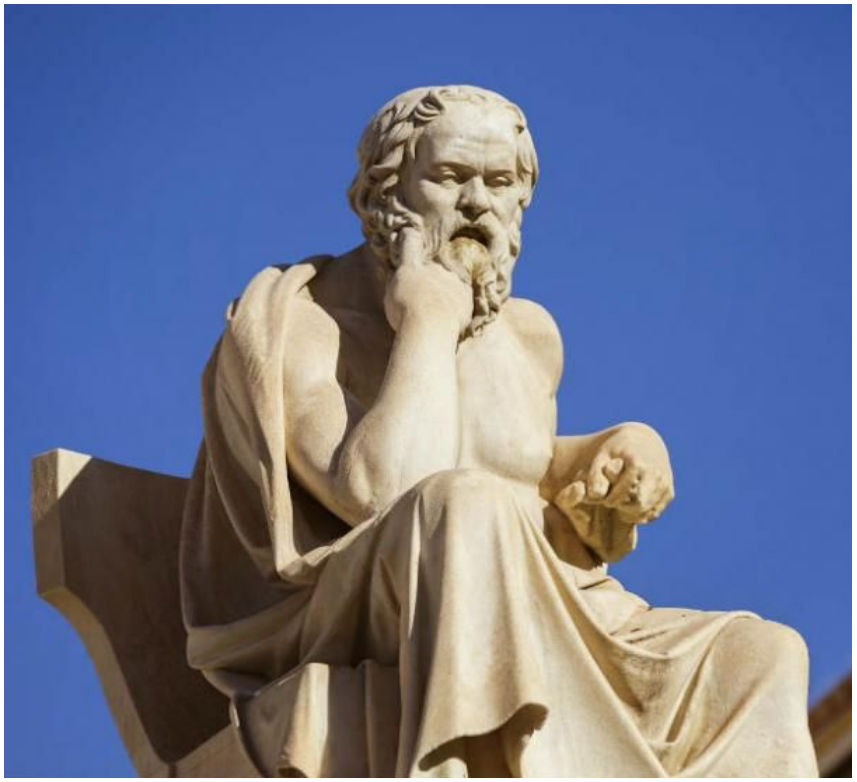
Усовершенствование способов разрешения неопределённостей

- Снижать число неизвестных
- Применять методологию познания
- Задавать правильные вопросы
- Сводить задачу к предыдущей

- Усовершенствование способов разрешения неопределённостей

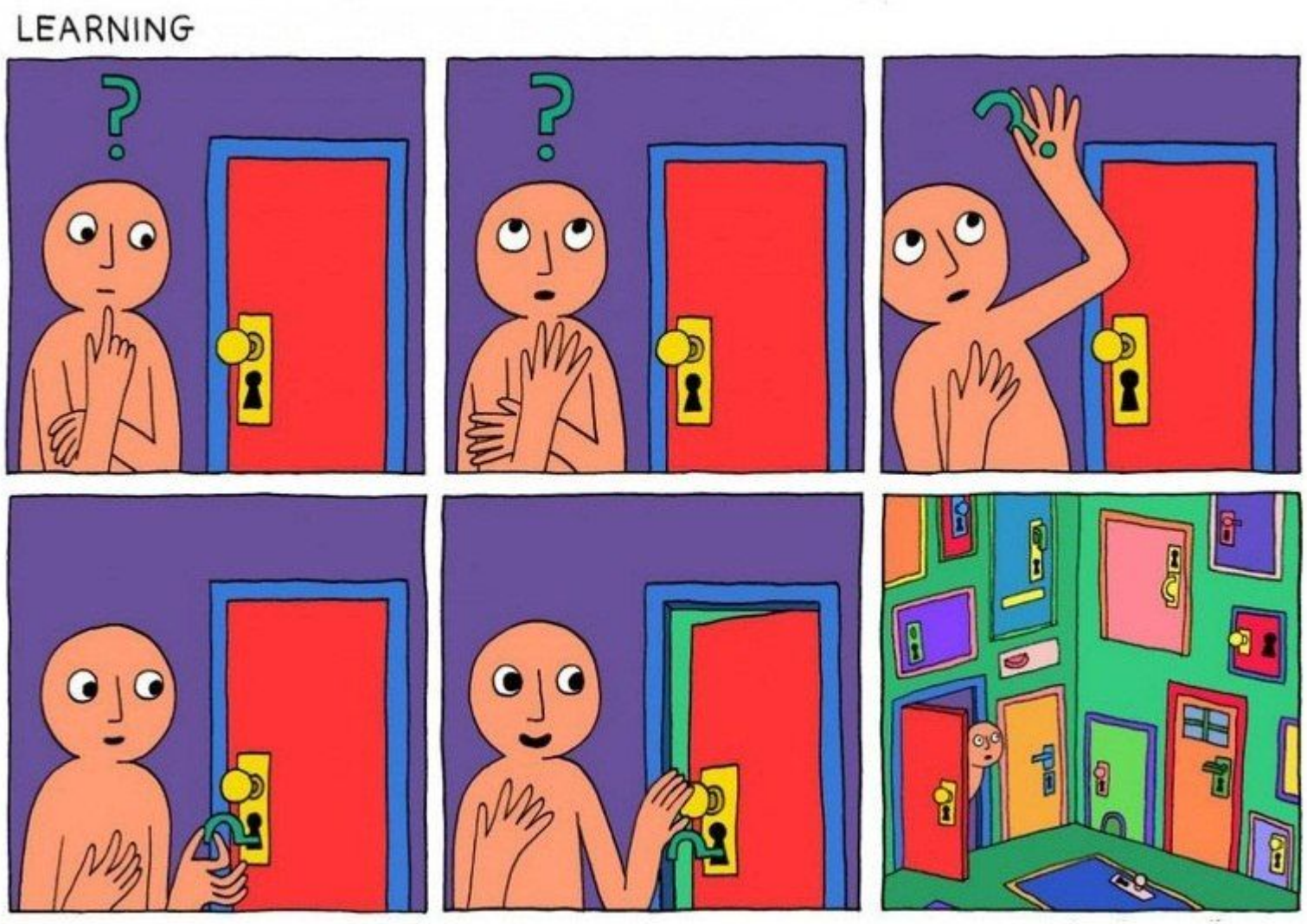
- Павел Яблочков: 4 попытки

Диалектика



- Искусство постижения истины путём задания правильных вопросов

- Выстраиваем цепочку правильных вопросов



Задавание правильного вопроса = измерение, разрешающее неопределённость

Ищите ответ самостоятельно

Спутник

Я

Google

Если возник вопрос — значит, ответ существует

Вопрос → X → Y → Z → Ответ

Концентрация на вопросе = Получение ответа

Уважающая себя мысль 2 раза не приходит

Список источников

- Основное
- Апология Сократа
- Альтшуллер. Как стать гением
- Альтов. И тут появился изобретатель

Профессиональное развитие


Оранжевое настроение
BEST-DEM.RU

Настроение

- Цель: научиться тестировать и развёртывать веб-приложение
- Внимание: тесты, ошибки в приложении, среды разработки (окружения)
- Намерение: чтобы приложение содержало меньше ошибок и было доступно в веб
- Вербальный настрой:
  - «я создаю веб-приложения без ошибок»


bomz.org
ЗНАНИЕ - СИЛА
незнание - рабочая сила
bomz.org

Знания

- Тест, степень покрытия
- Окружения разработки
- Модульное и функциональное тестирование с помощью Minitest
- Интеграционное тестирование
- Подход «разработка через тестирование» (TDD)
- Развёртывание приложения на heroku
- Подход BDD с Cucumber

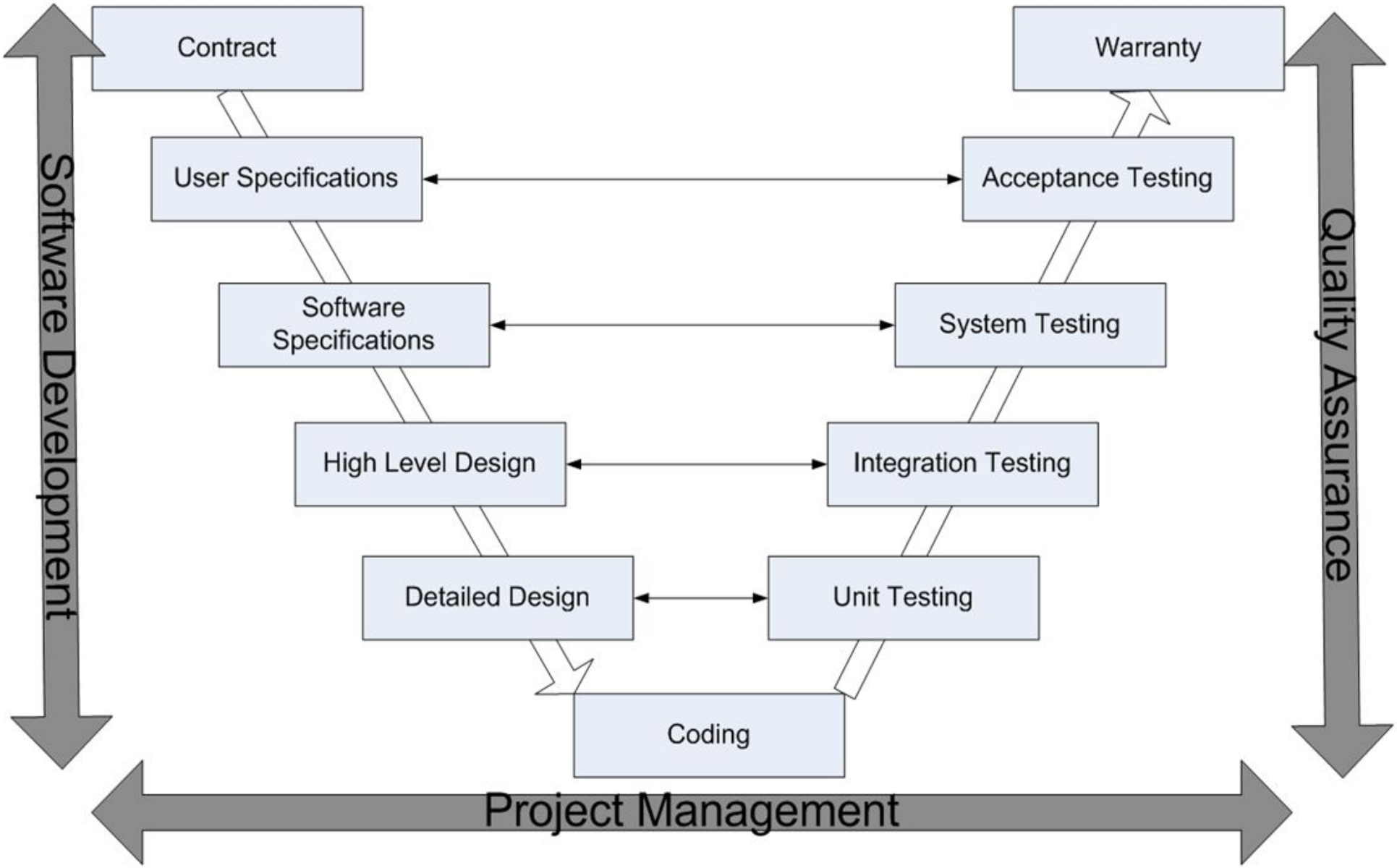
Тест и степень покрытия

- Тест — набор утверждений о разработанной функциональности
- Степень покрытия (%) = (код покрытый тестами) / (весь код)

Окружения разработки

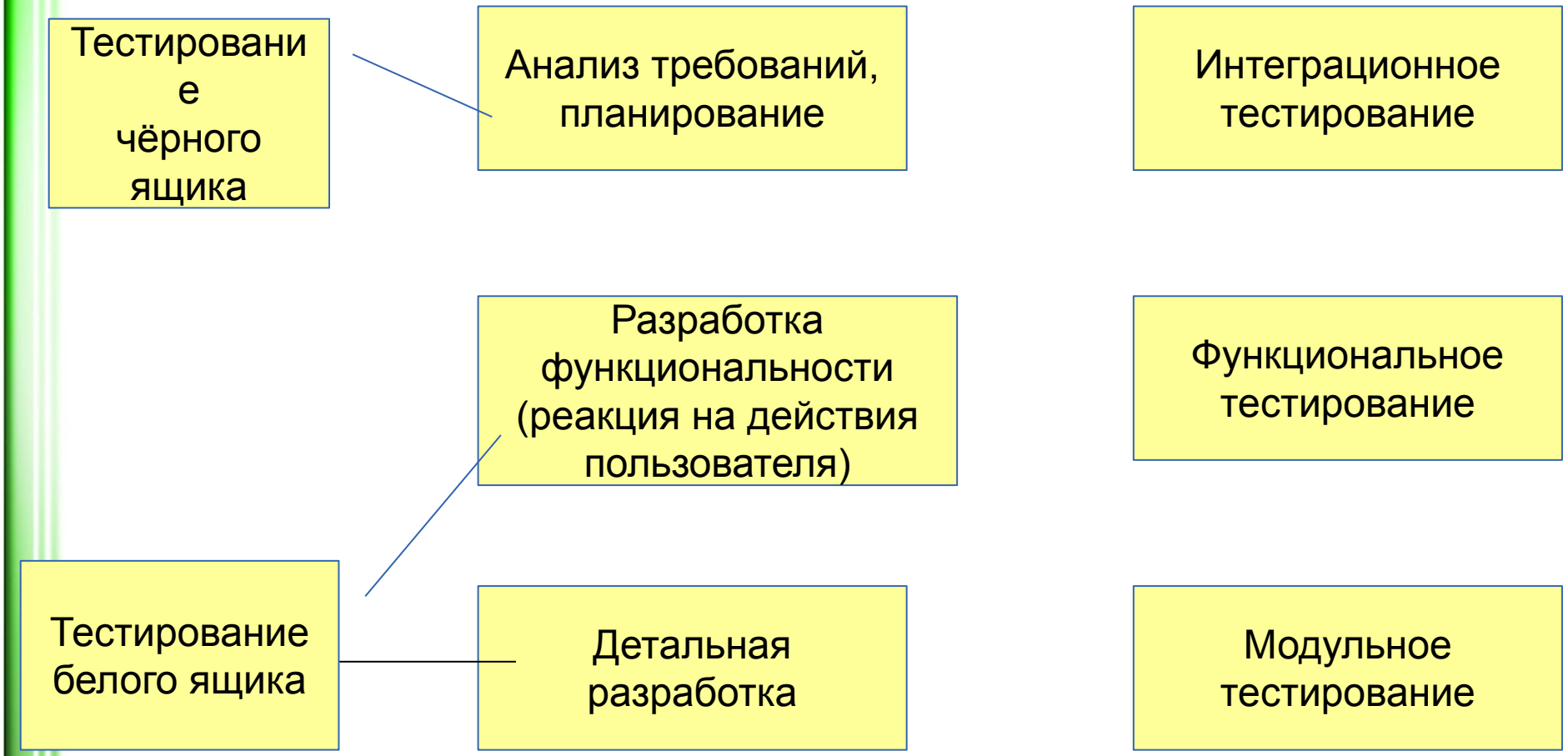
- Определяют фазу ЖЦ проекта, в т.ч. гемов, БД и т.д.
- Основные:
  - development — фаза разработки
  - test — фаза альфа-тестирования
  - production — фаза эксплуатации
  - staging — фаза бета-тестирования
- Настройки — в config/environments/*.rb
- Задание извне — ENV["RAILS_ENV"]

Уровни тестирования

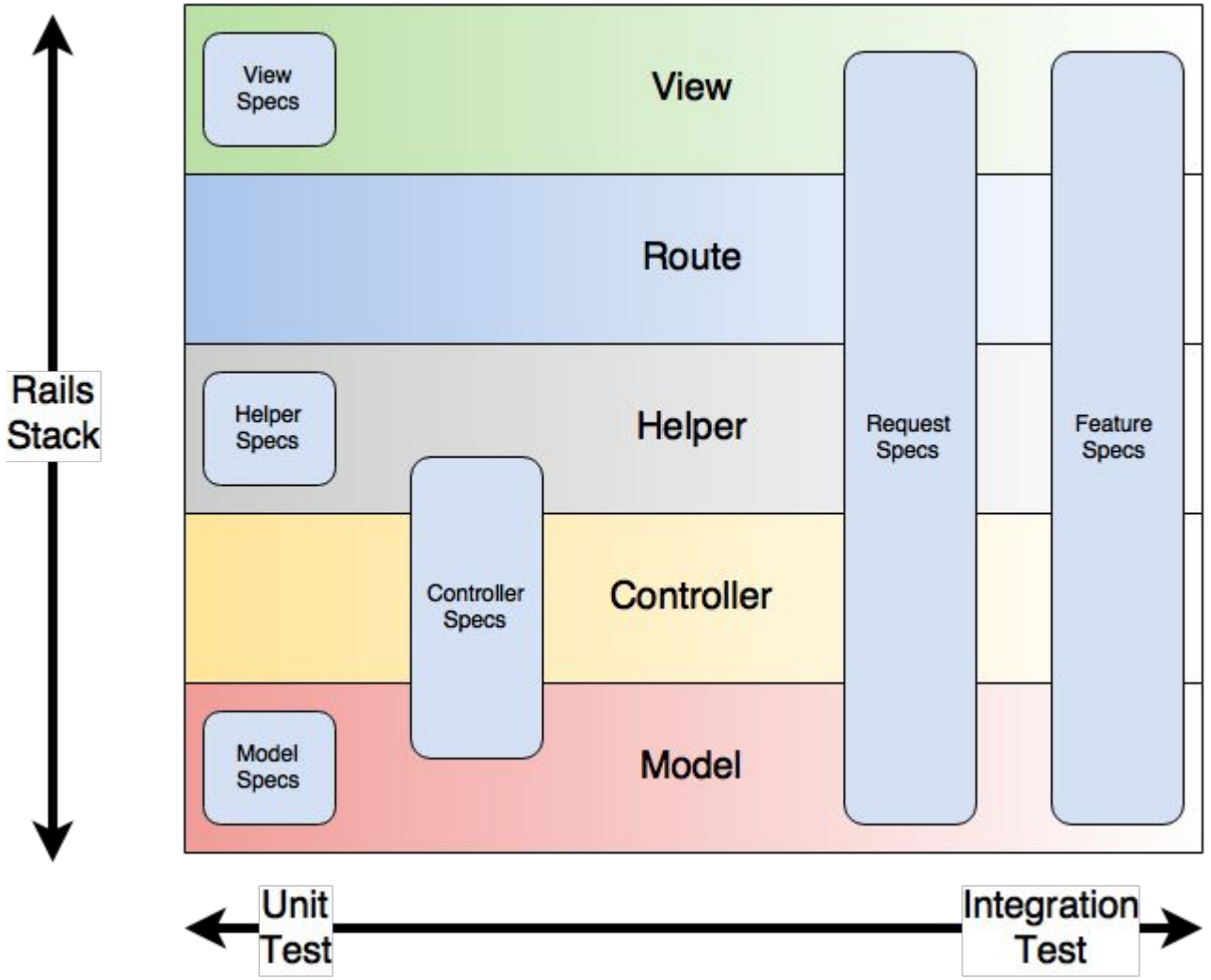


Уровни тестирования

Виды тестирования. Связь с фазами ЖЦ



Способы тестирования в Rails



Модульные тесты в Minitest

- Тестирование моделей
  - Валидация,
  - связи,
  - собственные методы

Модульные тесты в Minitest

```
require 'test_helper'
class CompetenceTest < ActiveSupport::TestCase
  test '.save' do
    compy = Competence.new(name: 'Minitest', user:
User.first)
    assert compy.save
  end
  test 'empty name do not allowed' do
    compy = Competence.new user: User.first
    assert_raise(StandardError) do
      compy.save
    end
  end
```

Утверждения

- assert(test) — test истинно
- assert_equal(expected, actual) — expected == actual
- …

Результат выполнения тестов

- Ошибка — E
- Провал — F
- Успех — S

Фикстуры

- специальные YAML-файлы, которые описывают объект, загружаемые в БД. Каждый файл — это отдельный класс (таблица).

Алгоритм загрузки

- Очищается БД от любых данных.
- Загружаются в таблицу
- Выгружаются в переменную, если это необходимо. Для прямого обращения.

Модульные тесты в Minitest

```
require 'test_helper'
class CompetenceTest < ActiveSupport::TestCase
  test '.save' do
    compy = Competence.new(name: 'Minitest', user:
User.first)
    assert compy.save
  end
  test 'empty name do not allowed' do
    compy = Competence.new user: users(:lomonosov)
    assert_raise(StandardError) do
      compy.save
    end
  end
```

Модульные тесты в Minitest

```
test 'user attribute is User' do
    compy = Competence.first
    assert_instance_of User, compy.user
  end
  test 'has many portfolios' do
    compy = competences(:one)
    assert_equal compy.portfolios.length, 2 #
  end
```

Использование ERb в фикстурах

```
<% 100.times do |num| %>
compy_<%= num %>:
  name: <%= "competence#{num}" %>
  user: lomonosov
<% end %>
```

Функциональные тесты: контроллеры

```
test "should get new" do
  get new_competence_url
  assert_response :success
end
```

Функциональные тесты: маршруты

```
test 'route show' do
  assert_generates '/competences/1', controller:
'competences', action: 'show', id: 1
end

test 'route create' do
  assert_recognizes({ controller: 'competences',
action: 'create' }, { path: 'competences',
method: :post })
end
```

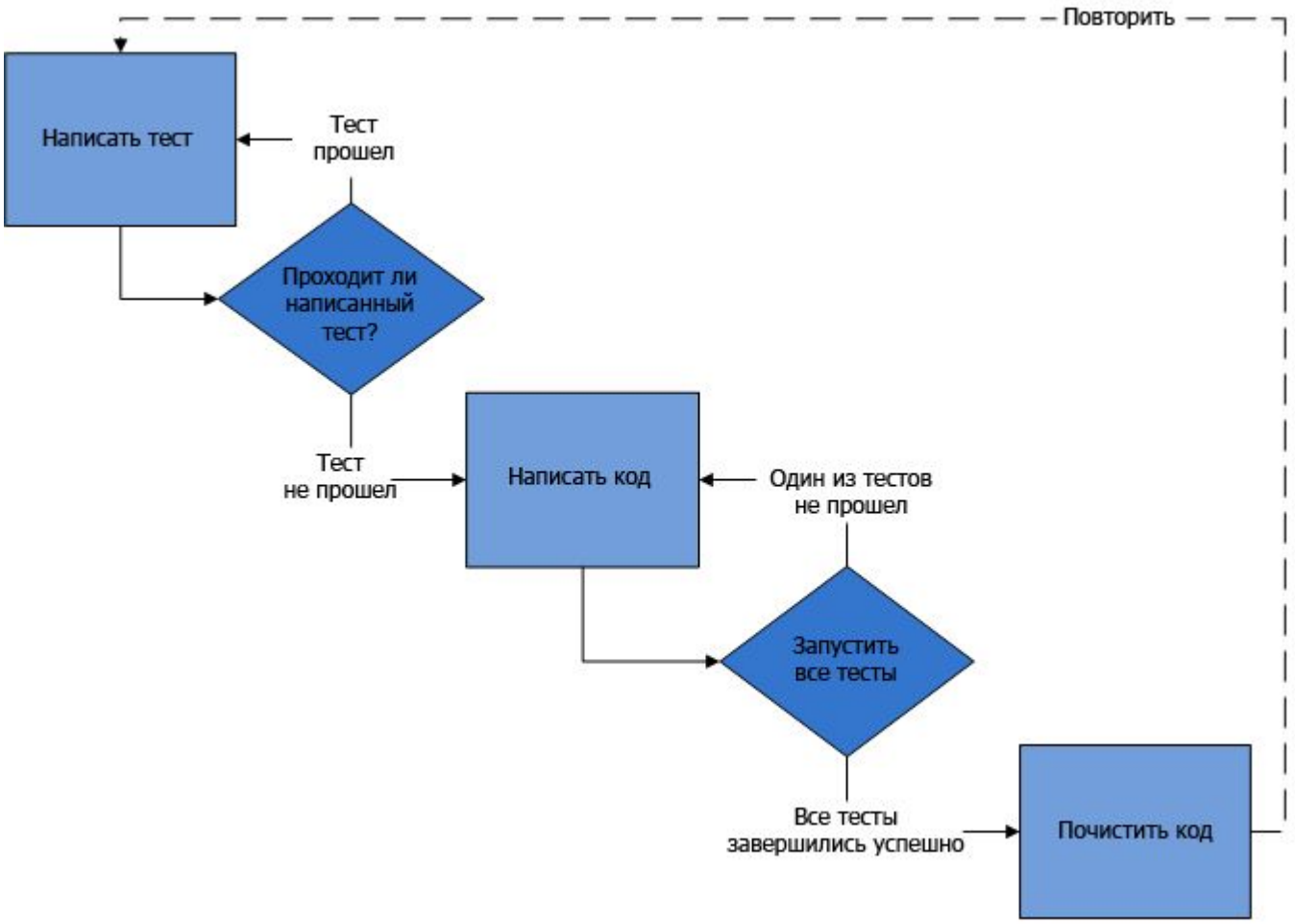
Функциональные тесты: представления

```
test 'index view' do
  get competences_url
  assert_select 'table.table' do
    assert_select 'thead' do
      assert_select 'tr' do
        assert_select 'th', 'Название'
        assert_select 'th', 'Автор'
      end
    end
  end
end
```

Интеграционные тесты

```
require 'test_helper'
class CompetenceFlowTest < ActionDispatch::IntegrationTest
  test 'create competence' do
    get new_competence_url
    assert_response :success
    post competences_url, params: { competence: { name:
'Сохранение вещества', user: users(:lomonosov) } }
    assert_response :redirect
    follow_redirect!
    assert_select 'div', 'Competence was successfully
created'
  end
end
```

Разработка через тестирование (TDD)



Ты просто не умеешь
Обращаться с
зазеркальными
пирогами.
Их сначала подают,
а уж потом разрезают.

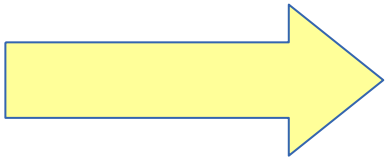
Принцип

Конечное состояние определяет текущее и последующие

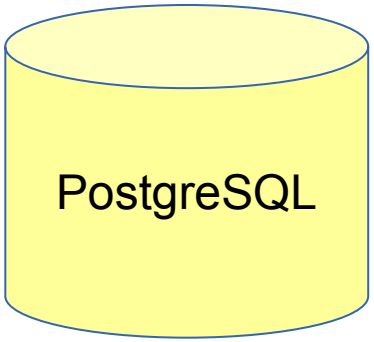
Развёртывание

production

Локальное веб-приложение (на Rails)



Веб-приложение на хостинге (heroku.com)



12 факторная модель

1. 1 код — много развёртываний (production, staging)
2. явно объявлять и изолировать зависимости (Gemfile)
3. хранить настройки в среде выполнения, разделять код и конфигурацию
4. Слабая зависимость от сторонних служб (СУБД и т. д.)
5. Разделять сборку, релиз (результат сборки + конфигурация) и выполнение
6. Любые данные сохранять в БД, не хранить в приложении

12 факторная модель

1. Экспортировать сервисы через привязку портов
2. Масштабировать приложение с помощью процессов (UNIX, не JVM)
3. Максимизировать надёжность, быстрый запуск и корректное завершение работы
4. Среды окружения разработки, промежуточного развёртывания и окончательного развёртывания д.б. максимально похожими
5. Журнал приложения (лог) — это поток событий
6. Выполнять задачи администрирования с помощью разовых процессов, например, для выполнения миграций

PostgreSQL

1. Открытый исходник
2. Большой размер таблиц (32 ТБ)
3. Особые типы данных (serial, xml, json)
4. Создание хранимых процедур на других языках (Java, Ruby и т. д.)
5. Наследование

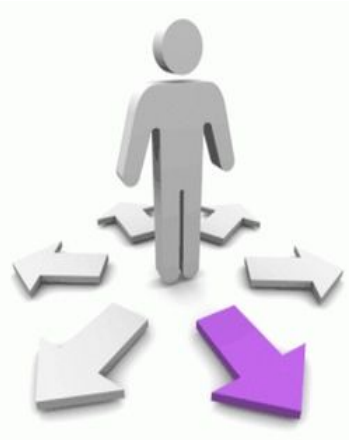

Ты молод, креативен, талантлив? Амбициозен, уверен в себе, полон свежих идей? А делать хоть что-нибудь умеешь?!
Atkritka.com

Умения

- Создавать модульные тесты на Minitest
- Создавать фикстуры
- Создавать функциональные тесты
- Создавать интеграционные тесты
- Создавать ветки в git, переключаться между ними
- Устанавливать PostgreSQL
- Развёртывать приложение на heroku


ТУДА
100м
СЮДА
200м
ВЕЧНАЯ НЕОПРЕДЕЛЕННОСТЬ!
КАК ЖИТЬ?
DEMOTIVATORS.RU

Неопределённости



- Зачем нужны фикстуры?
- Отличия MySQL от PostgreSQL?
- Почему не проходил тест создания компетенции?

Результат


Это же так просто!
Я это сделаю!
Я могу это сделать!
Я попытаюсь...
Как мне это сделать?
Я хочу это сделать!
Я не могу это сделать (не умею, не получится)
Я не буду этого делать.

Результат

- Изучены способы, как разрешать неопределённости и творить
- Научились тестировать приложение и развёртывать его на heroku

Список источников

- Основное
- Тестирование приложений на Minitest
- Развёртывание приложения на Heroku
- Дополнительное
- Установка PostgreSQL
- 12 факторная модель
- BDD и Cucumber
- Бек. Экстремальное программирование: разработка через тестирование