Яндекс

Flutter: тестирование

Богдан Лукин, Разработчик мобильных приложений

Теоретическая база

Что?

> Тестирование — проверка того, что фактическое поведение соответствует ожидаемому

Зачем?

> Прикольно же, крутится что-то, зелененьким светится

Зачем?

~~Прикольно же, крутится что-то, зелененьким светится~~

Задача тестирования -найти некорректное поведение

Цели

- Обнаружение дефектов
- Повышение уверенности в уровне качества
- Предоставление информации для принятия решений
- Предотвращение дефектов

Зачем?

~~Прикольно же, крутится что-то, зелененьким светится~~

Задача тестирования -найти некорректное поведение

Цели

- Обнаружение дефектов
- Повышение уверенности в уровне качества
- Предоставление информации для принятия решений
- Предотвращение дефектов
- Выявление багов на раннем этапе
- Упрощение отладки
- Минимизация регресса

Зачем?

~~Прикольно же, крутится что-то, зелененьким светится~~

Задача тестирования -найти некорректное поведение

Цели

- Обнаружение дефектов
- Повышение уверенности в уровне качества
- Предоставление информации для принятия решений
- Предотвращение дефектов
- Выявление багов на раннем этапе
- Упрощение отладки
- Минимизация регресса

А еще бонусом можно получить

- Документацию с примерами использования
- Способ закрепления договоренностей без детальной проработки документации

AAA


AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAA

AAA: Arrange → Act → Assert

AAA: Arrange → Act → Assert

〉 Arrange (настройка) — в этом блоке кода мы настраиваем тестовое окружение тестируемого юнита;

〉 Act — выполнение или вызов тестируемого сценария;

〉 Assert — проверка того, что тестируемый вызов ведет себя определенным образом.

Тестируемый код

SOLID

Тестируемый код

SOLID

Да, этого хватает

Тестируемый код

SOLID

Да, этого хватает

〉 Стремитесь к слабой связности.

〉 Не пишите сложную логику в конструкторах.
А лучше вообще никакую

〉 Избегайте статических зависимостей.

〉 Должна быть возможность проставить зависимости снаружи
Обычно это решается тем, что зависимости приходят в конструктор, на это можно накручивать любые сервис локаторы и di фреймворки)

stub, mock, fake, dummy, double, NaN

> **Дублер** (**Double**) — любой объект притворяющийся реальным

stub, mock, fake, dummy, double, NaN

〉 **Дублер** (**Double**) — любой объект притворяющийся реальным

Dummy — передается по коду, но фактически не используемые

stub, mock, fake, dummy, double, NaN

- **Дублер** (**Double**) — любой объект притворяющийся реальным
- **Dummy** — передается по коду, но фактически не используемые
- **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

stub, mock, fake, dummy, double, NaN

- **Дублер** (**Double**) — любой объект притворяющийся реальным
- **Dummy** — передается по коду, но фактически не используемые
- **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

> **Стабы**, **заглушки** (**Stubs**) — предоставляет заготовленные ответы на вызовы сделанные во время теста, обычно вообще не отвечая ни на что, кроме того, что запрограммировано для теста.

stub, mock, fake, dummy, double, NaN

〉 **Дублер** (**Double**) — любой объект притворяющийся реальным

〉 **Dummy** — передается по коду, но фактически не используемые

〉 **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

〉 **Стабы**, **заглушки** (**Stubs**) — предоставляет заготовленные ответы на вызовы сделанные во время теста, обычно вообще не отвечая ни на что, кроме того, что запрограммировано для теста.

Spies — как стабы, но дополнительно записывают какую-то информацию о том как к ним обращались

stub, mock, fake, dummy, double, NaN

〉 **Дублер** (**Double**) — любой объект притворяющийся реальным

〉 **Dummy** — передается по коду, но фактически не используемые

〉 **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

〉 **Стабы**, **заглушки** (**Stubs**) — предоставляет заготовленные ответы на вызовы сделанные во время теста, обычно вообще не отвечая ни на что, кроме того, что запрограммировано для теста.

〉 **Spies** — как стабы, но дополнительно записывают какую-то информацию о том как к ним обращались

Моки (**Mocks**) — Содержат заранее запрограммированные ожидания вызовов.

stub, mock, fake, dummy, double, NaN

〉 **Дублер** (**Double**) — любой объект притворяющийся реальным

〉 **Dummy** — передается по коду, но фактически не используемые

〉 **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

〉 **Стабы**, **заглушки** (**Stubs**) — предоставляет заготовленные ответы на вызовы сделанные во время теста, обычно вообще не отвечая ни на что, кроме того, что запрограммировано для теста.

〉 **Spies** — как стабы, но дополнительно записывают какую-то информацию о том как к ним обращались

Моки (**Mocks**) — Содержат заранее запрограммированные ожидания вызовов.

Пример:

〉 У меня вызовут метод foo с любыми аргументами ровно один раз

〉 Это произойдет до вызова bar, но после baz

stub, mock, fake, dummy, double, NaN

〉 **Дублер** (**Double**) — любой объект притворяющийся реальным

〉 **Dummy** — передается по коду, но фактически не используемые

〉 **Fake** — реально выполняющий свою задачу объект, но непригодный для продакшена

〉 **Стабы**, **заглушки** (**Stubs**) — предоставляет заготовленные ответы на вызовы сделанные во время теста, обычно вообще не отвечая ни на что, кроме того, что запрограммировано для теста.

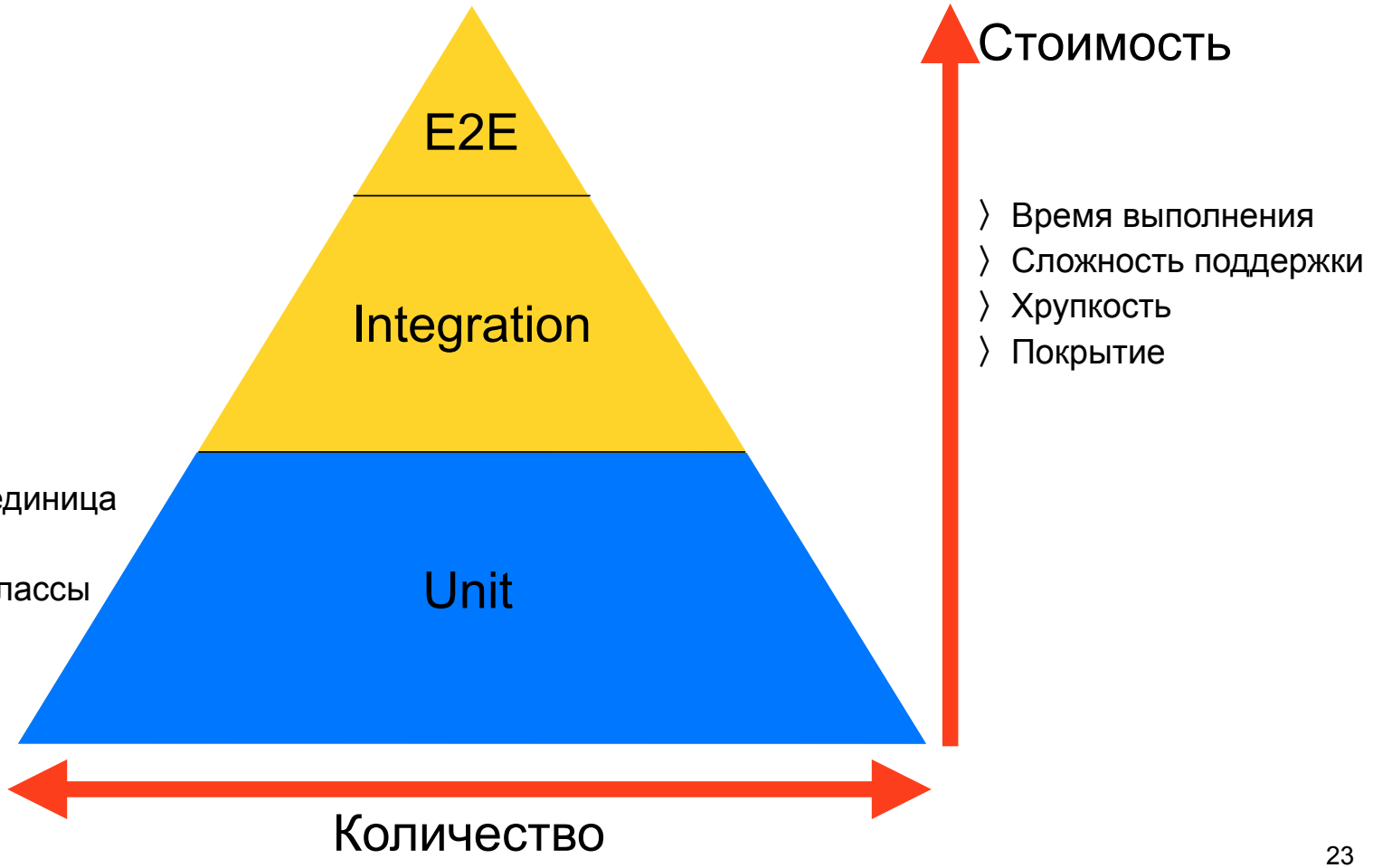
〉 **Spies** — как стабы, но дополнительно записывают какую-то информацию о том как к ним обращались

〉 **Моки** (**Mocks**) — Содержат заранее запрограммированные ожидания вызовов.

NaN — NaNNaNNaNNaN NaNNaNNaNNaN BATMAN

Классическая пирамида тестирования

Классическая пирамида тестирования



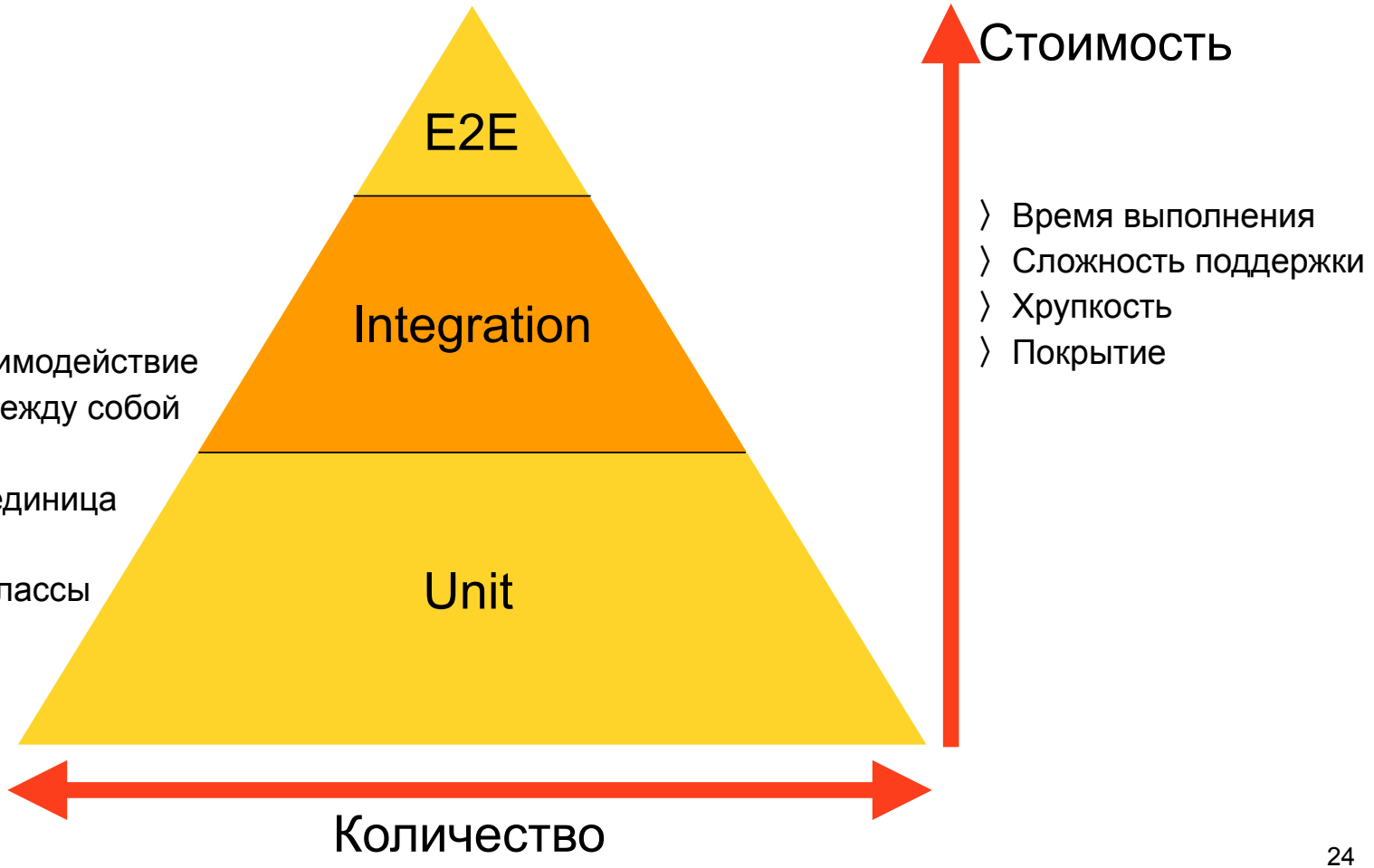
- Время выполнения
- Сложность поддержки
- Хрупкость
- Покрытие

Unit — минимальная единица тестирования.
Отдельные функции/классы изолированные от остальной системы

Классическая пирамида тестирования

Integration tests — взаимодействие нескольких модулей между собой

Unit — минимальная единица тестирования.
Отдельные функции/классы изолированные от остальной системы

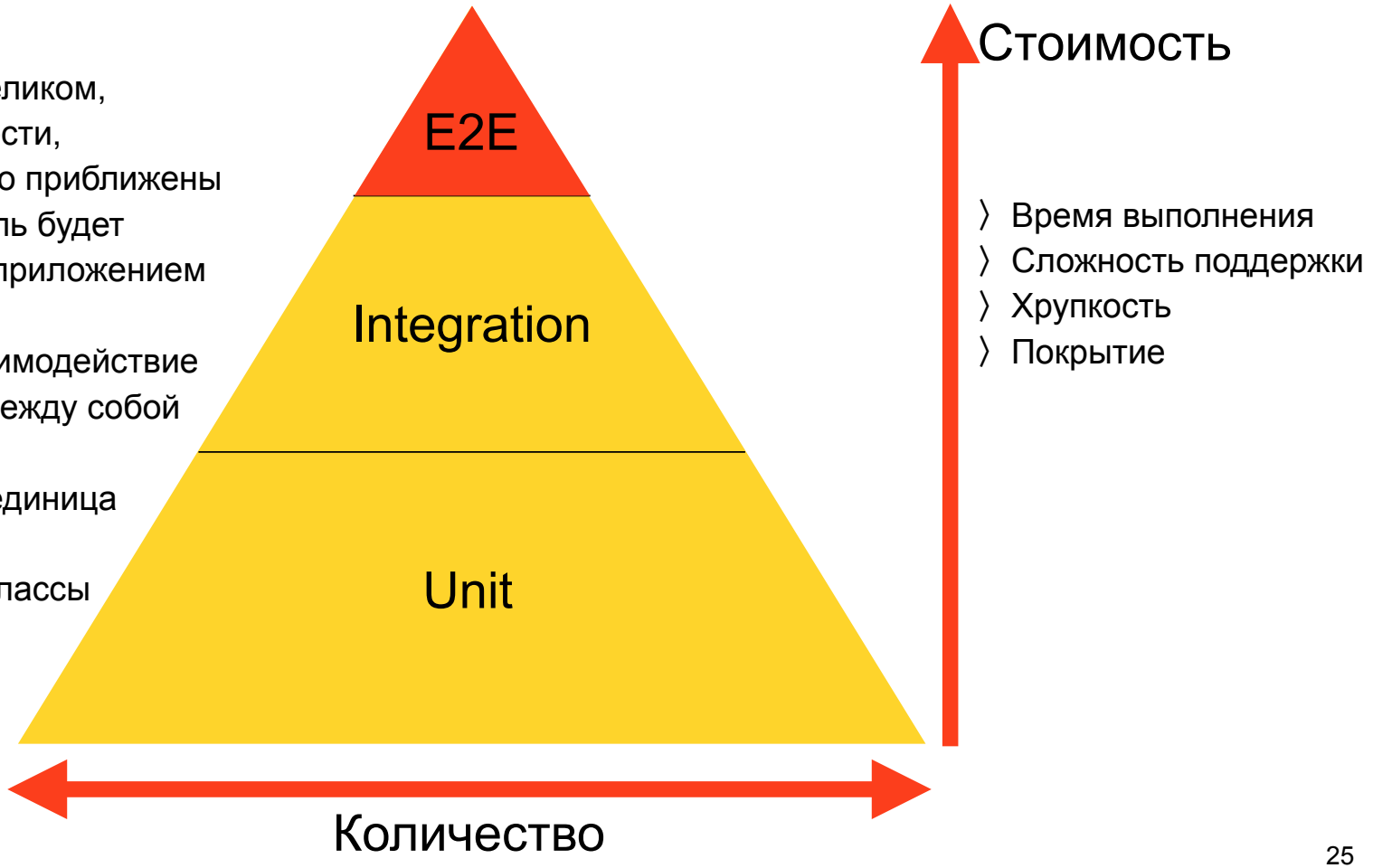


Классическая пирамида тестирования

E2E — вся система целиком, включая все зависимости, сценарии максимально приближены к тому как пользователь будет взаимодействовать с приложением

Integration tests — взаимодействие нескольких модулей между собой

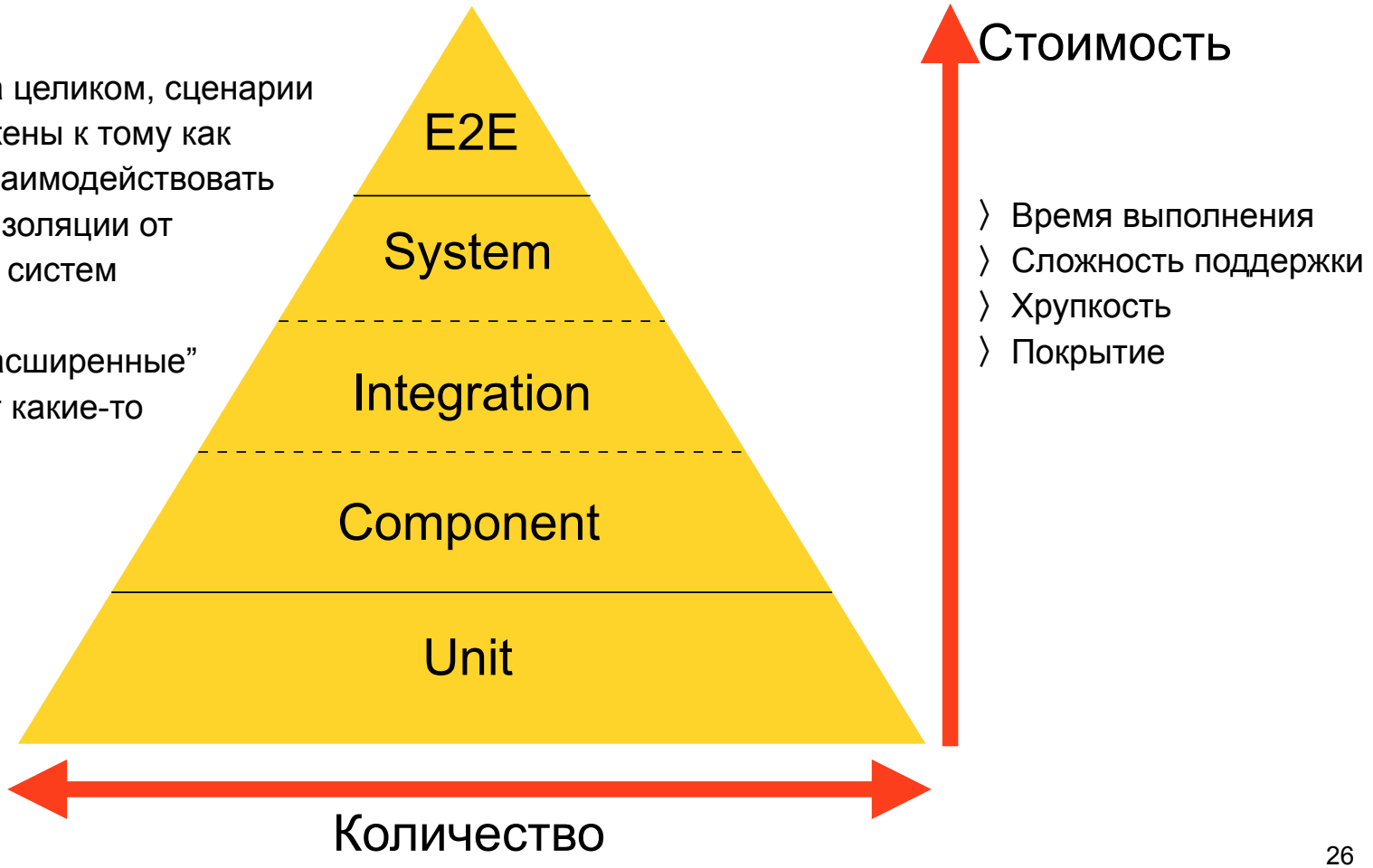
Unit — минимальная единица тестирования. Отдельные функции/классы изолированные от остальной системы



“Идеальная” пирамида тестирования

System — вся система целиком, сценарии максимально приближены к тому как пользователь будет взаимодействовать с приложением, но в изоляции от смежных и зависимых систем

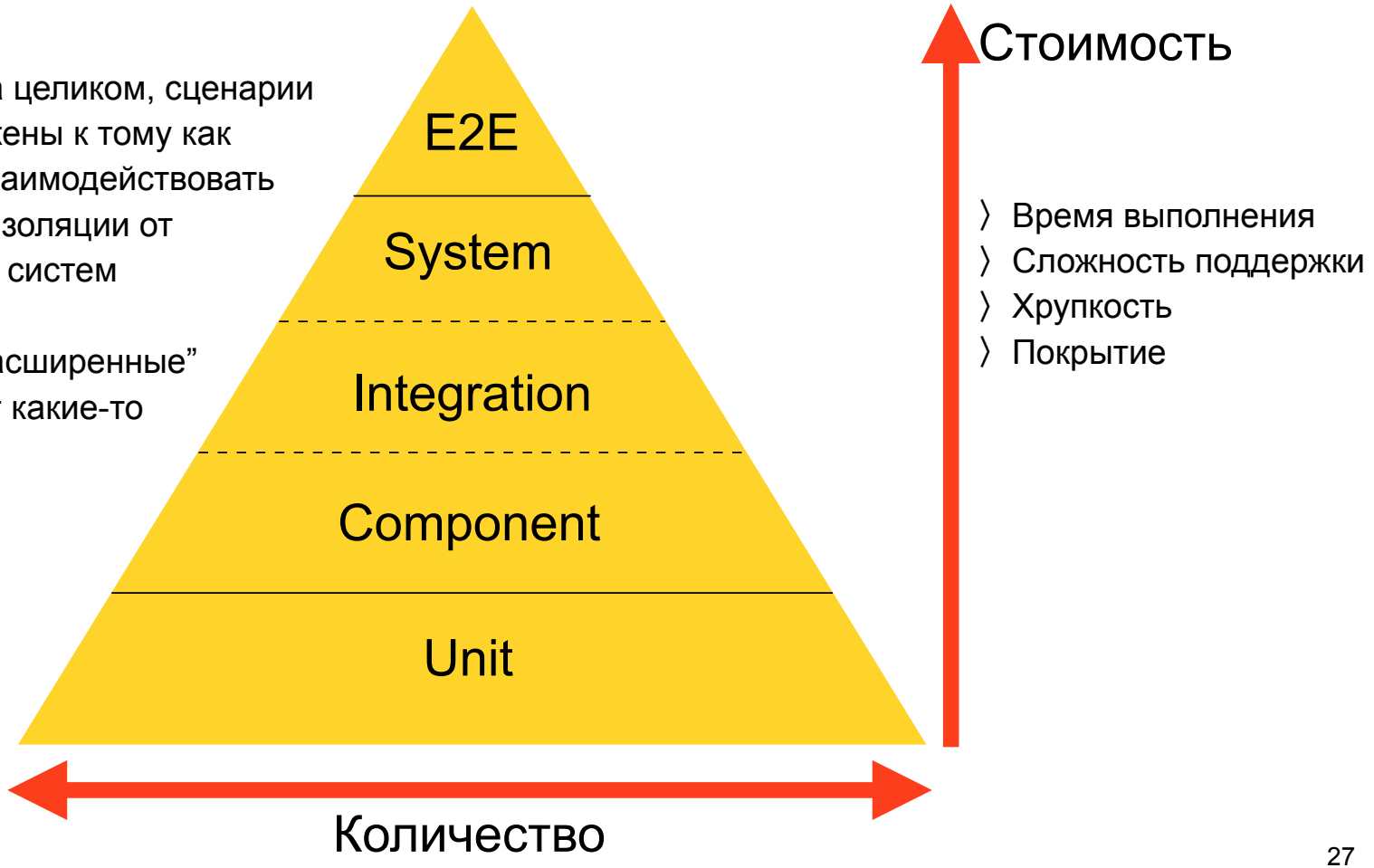
Component tests — “расширенные” Unit тесты. Проверяют какие-то



“Идеальная” пирамида тестирования

System — вся система целиком, сценарии максимально приближены к тому как пользователь будет взаимодействовать с приложением, но в изоляции от смежных и зависимых систем

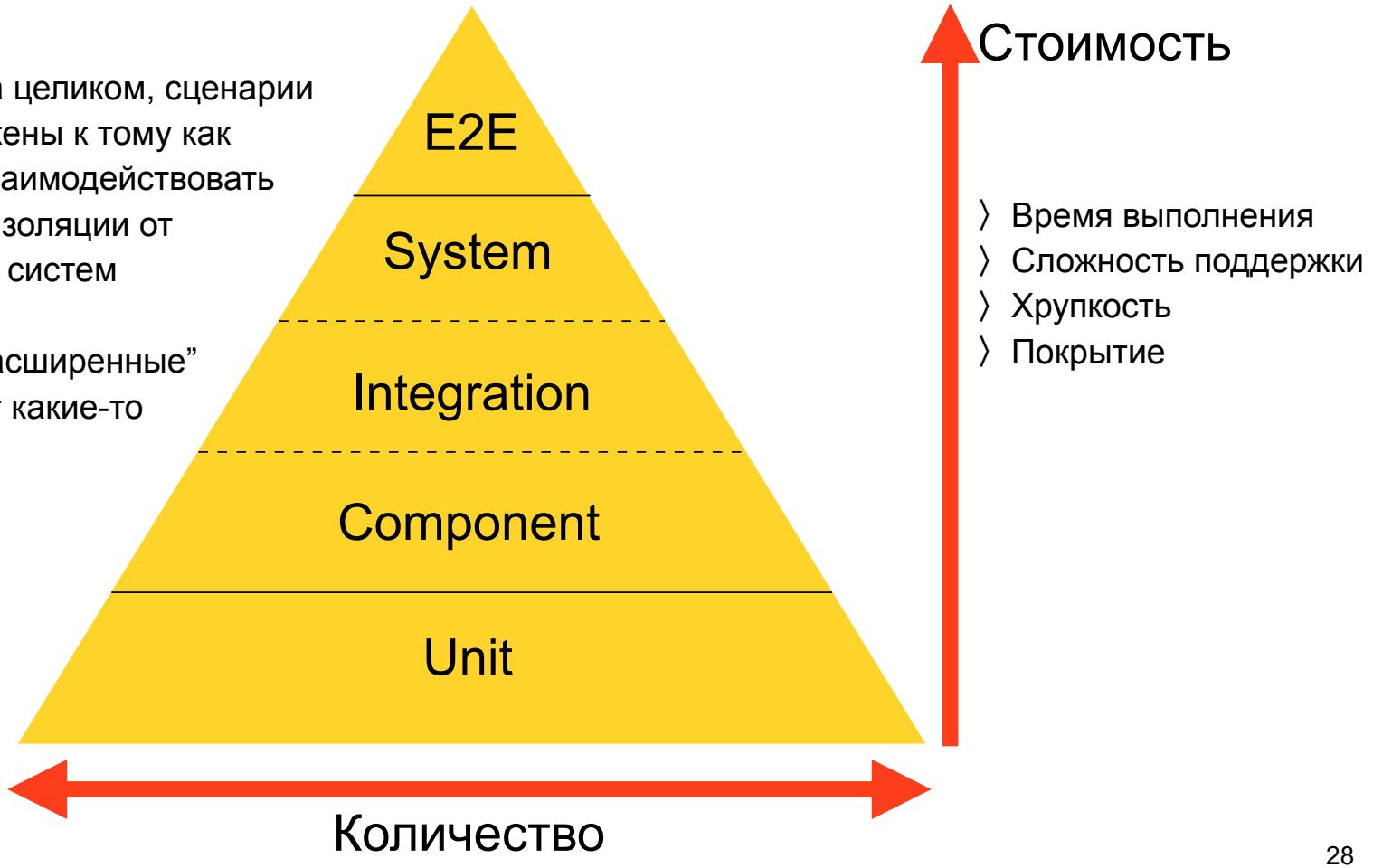
Component tests — “расширенные” Unit тесты. Проверяют какие-то



“Идеальная” пирамида тестирования

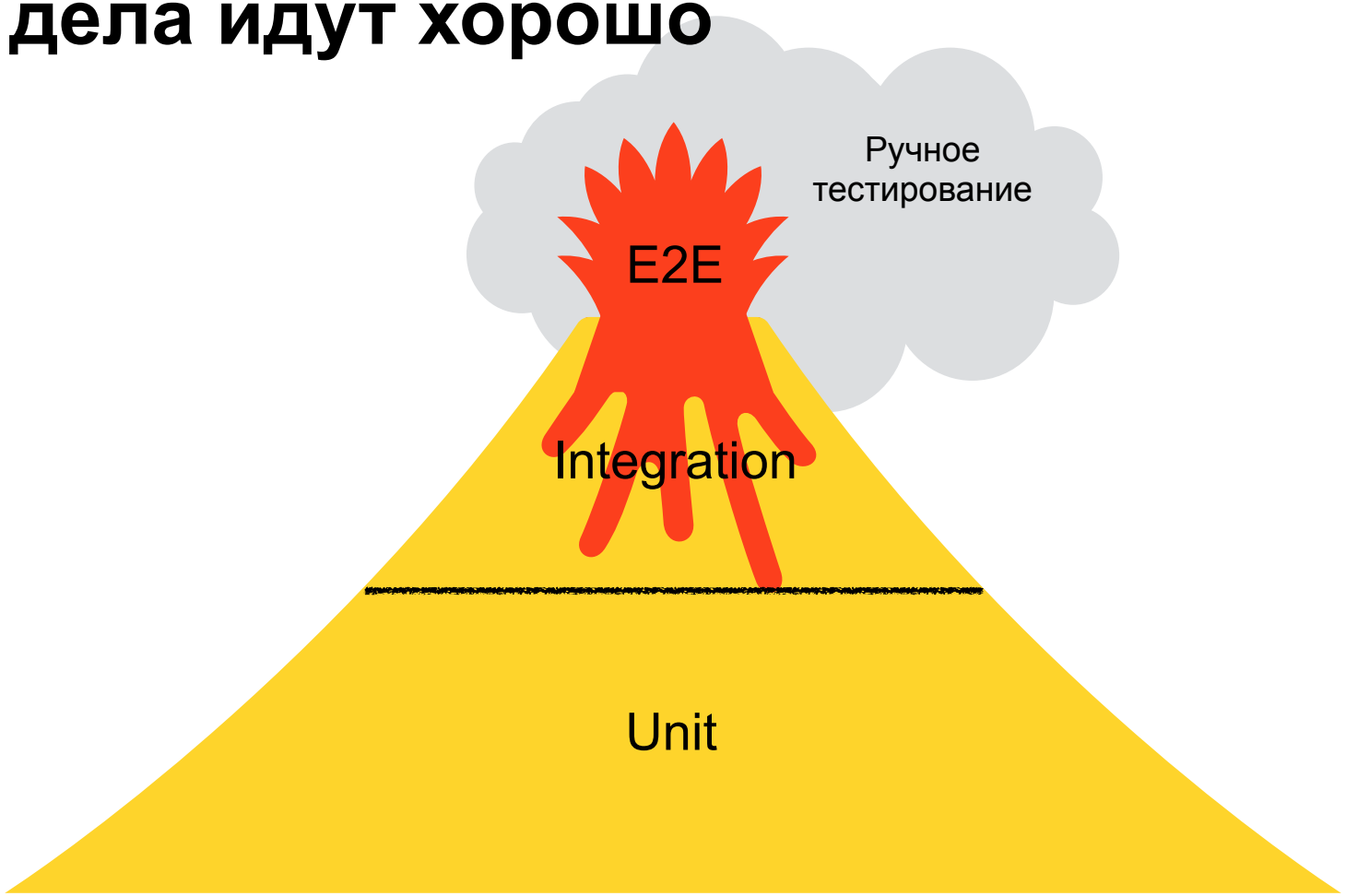
System — вся система целиком, сценарии максимально приближены к тому как пользователь будет взаимодействовать с приложением, но в изоляции от смежных и зависимых систем

Component tests — “расширенные” Unit тесты. Проверяют какие-то

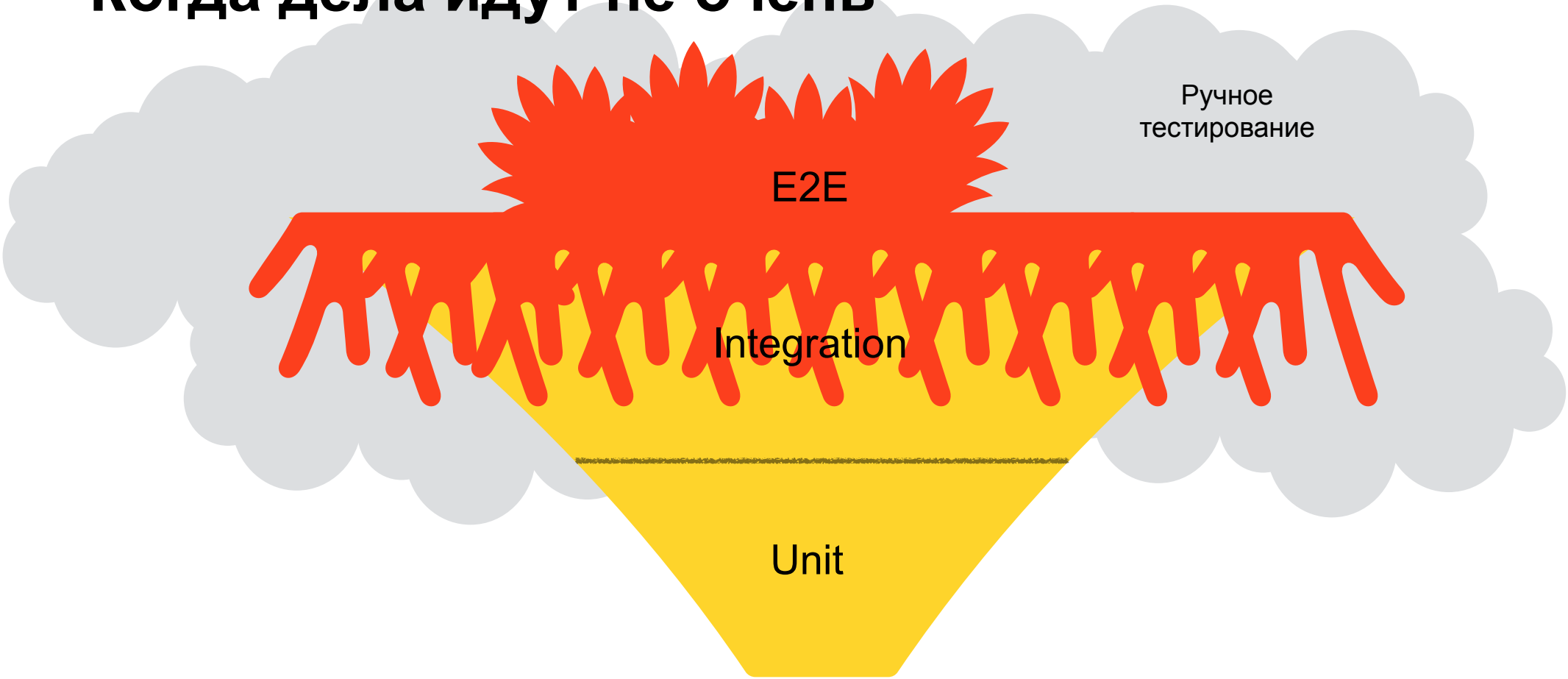


А как это выглядит на практике?

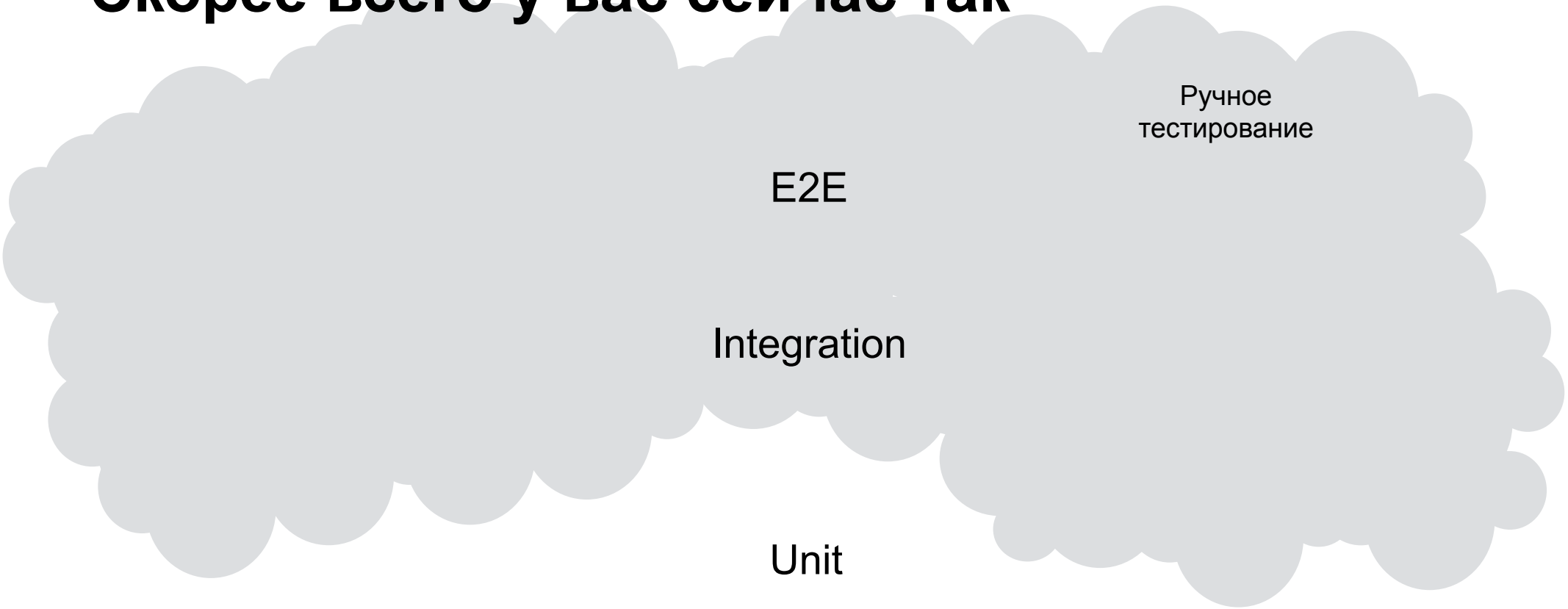
Когда дела идут хорошо



Когда дела идут не очень



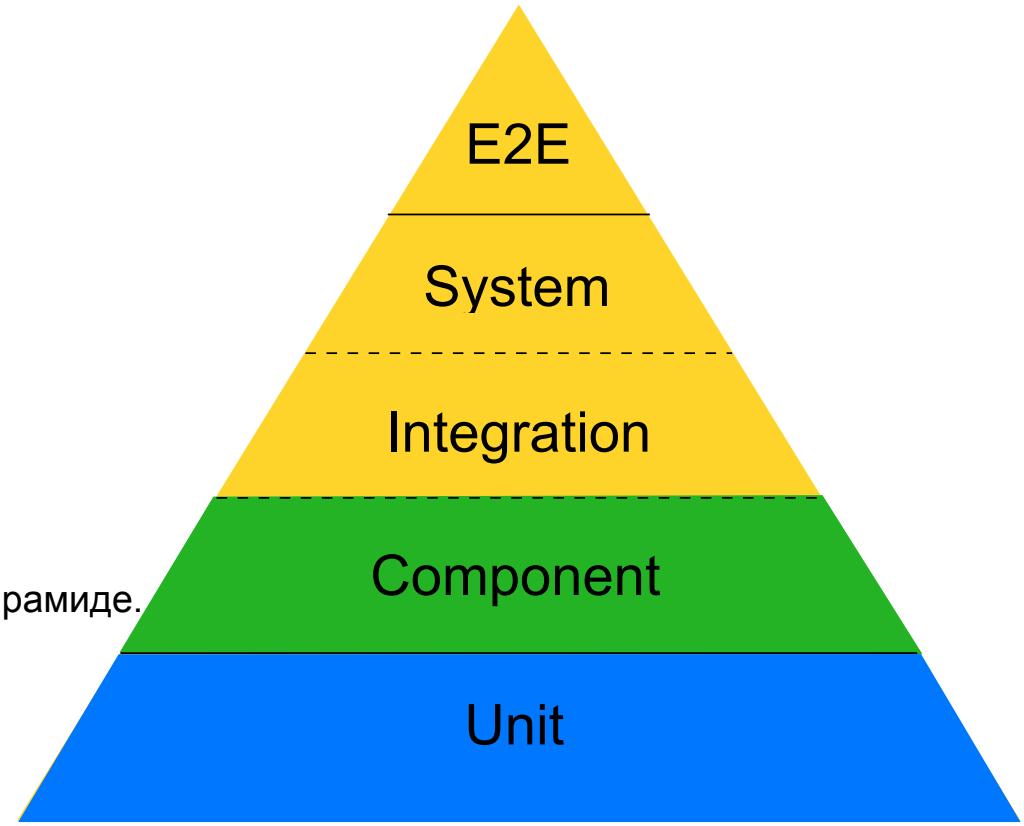
Скорее всего у вас сейчас так



А что в мобильной разработке?

А что в мобильной разработке?

Unit&Component tests — так же как и в обычной пирамиде.
Нет практически никакой специфики

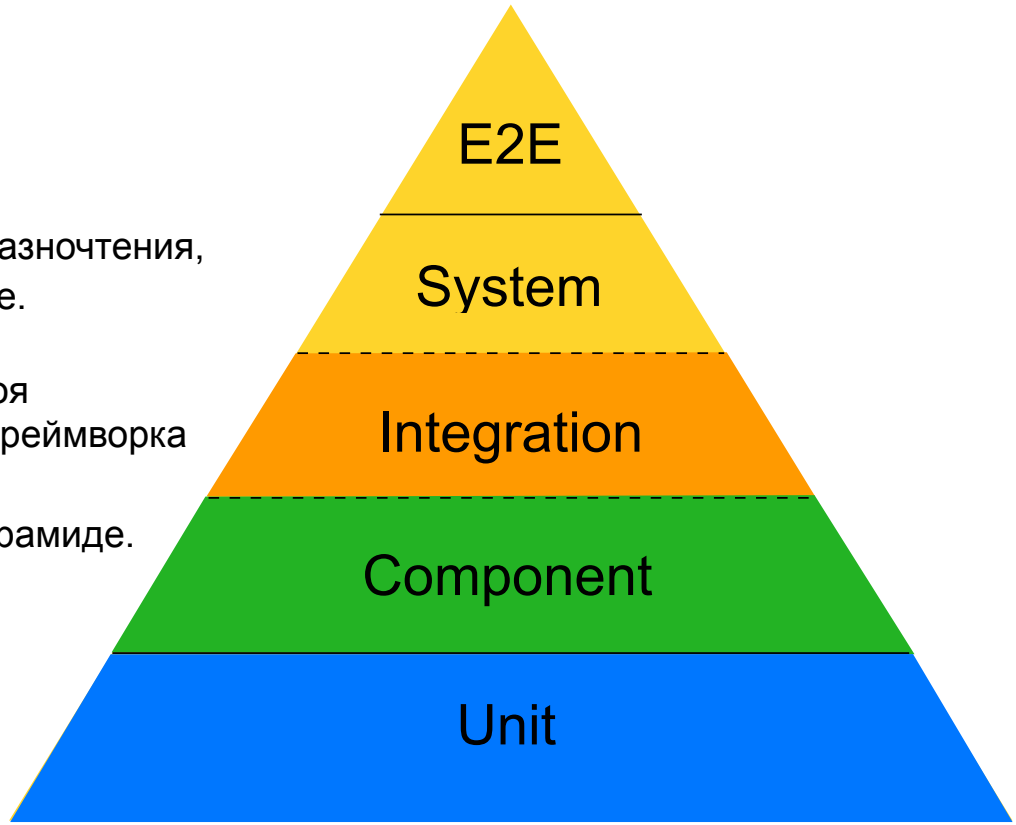


А что в мобильной разработке?

Integration tests — тут начинаются самые разные разночтения, чаще понимаются так же как и в обычной пирамиде.

〉 Используем mock бекенда, тестируем поведение API—контроллеров—моделей и далее вплоть до слоя презентации, но не доходя до уровня реального ui фреймворка

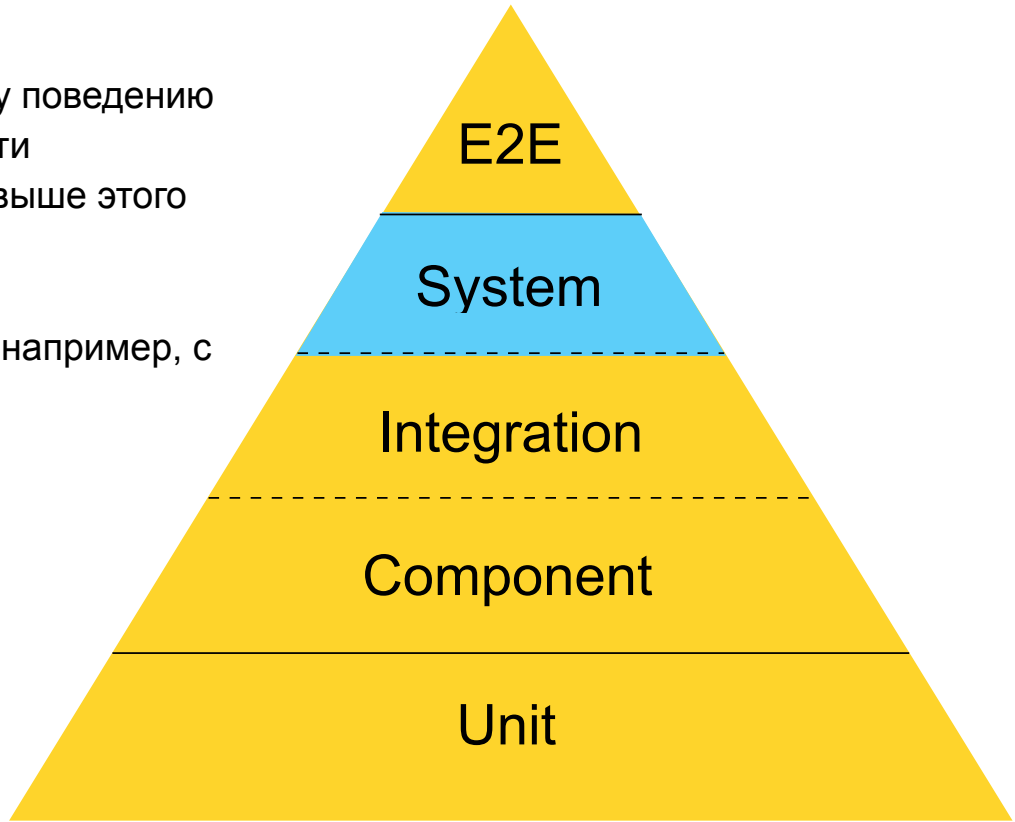
Unit&Component tests — так же как и в обычной пирамиде. Нет практически никакой специфики



А что в мобильной разработке?

> System tests - начинаем приближаться к реальному поведению пользователей, но допускаем некоторые условности (при тестировании отдельных модулей/библиотек выше этого уровня подняться не можем)

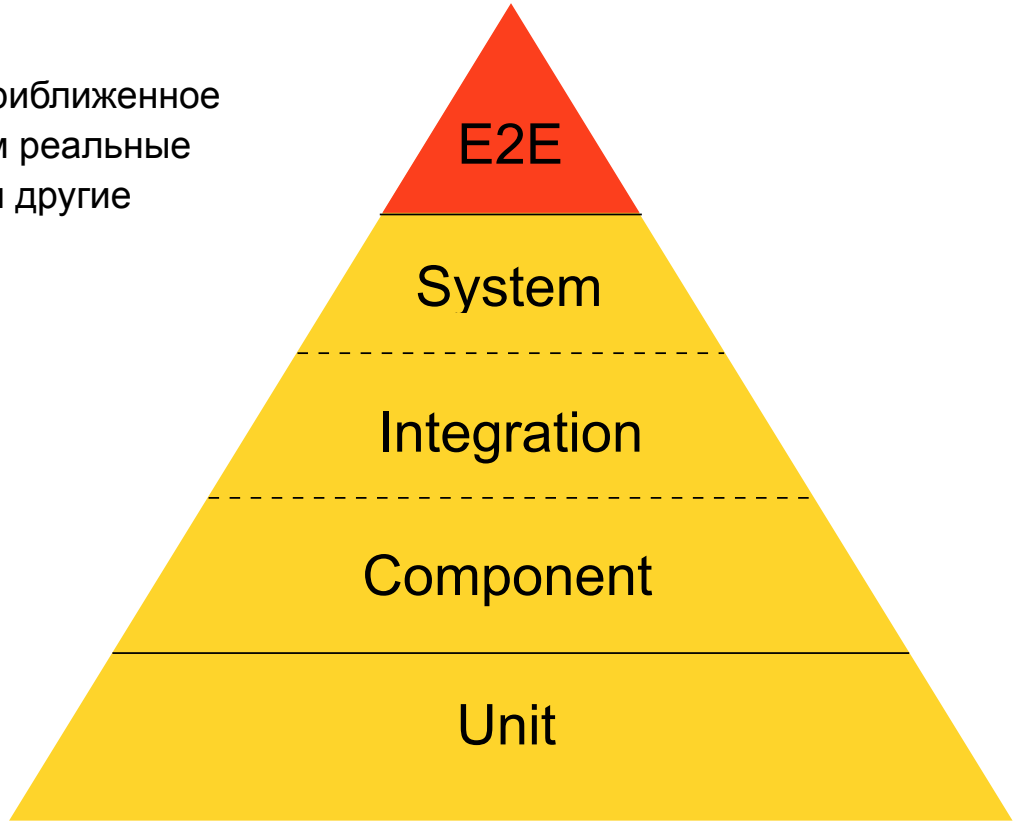
- UI тесты на реальных устройствах/эмуляторах, но, например, с замотанным API
- Тесты без UI на реальном API
- Другие тесты на реальных устройствах



А что в мобильной разработке?

E2E - UI тесты целого приложения максимально приближенное к реальному поведению пользователя. Используем реальные устройства или симуляторы, настоящий backend, и другие внешние сервисы

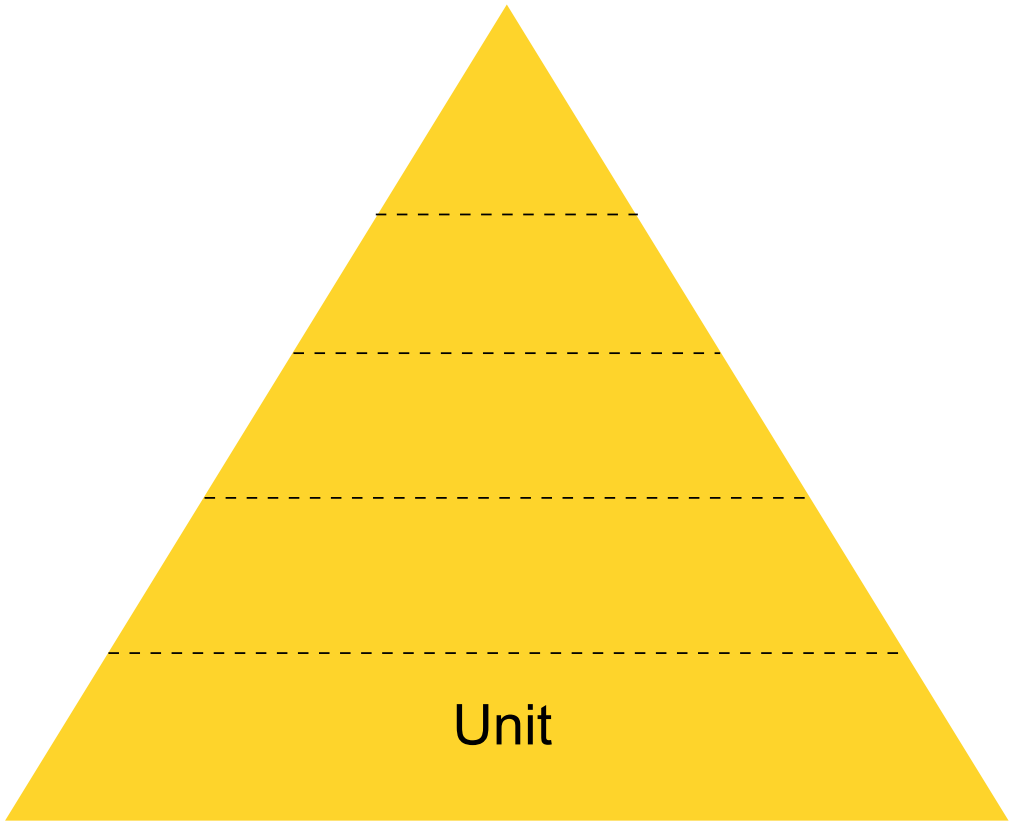
Очень часто эта часть остается целиком на ручном тестировании



Flutter: виды тестов и инструменты

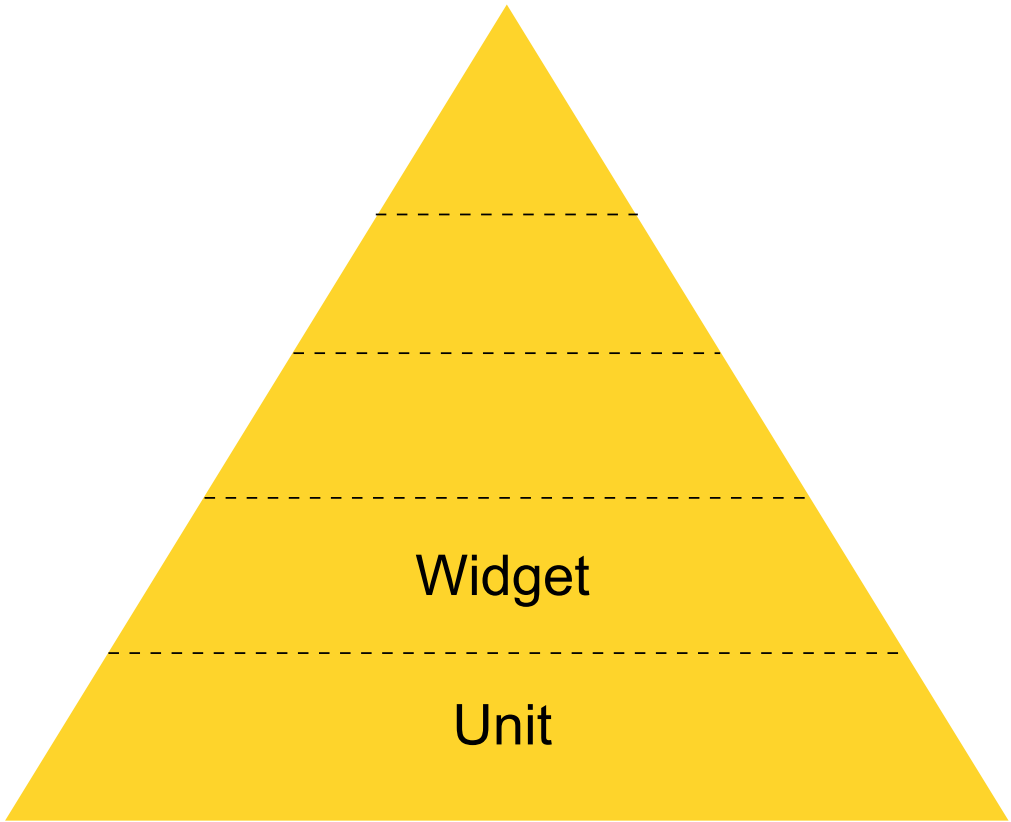
Flutter инструменты

```
〉package:test
```



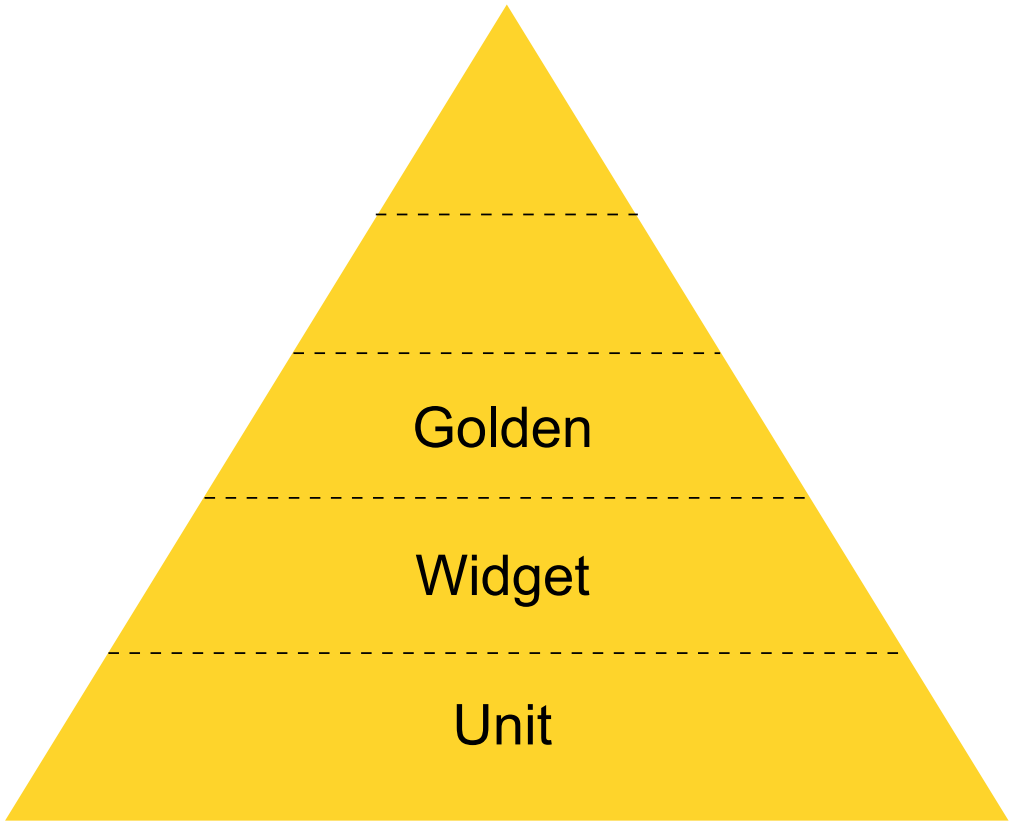
Flutter инструменты

```
〉package:test
〉package:flutter_test
```



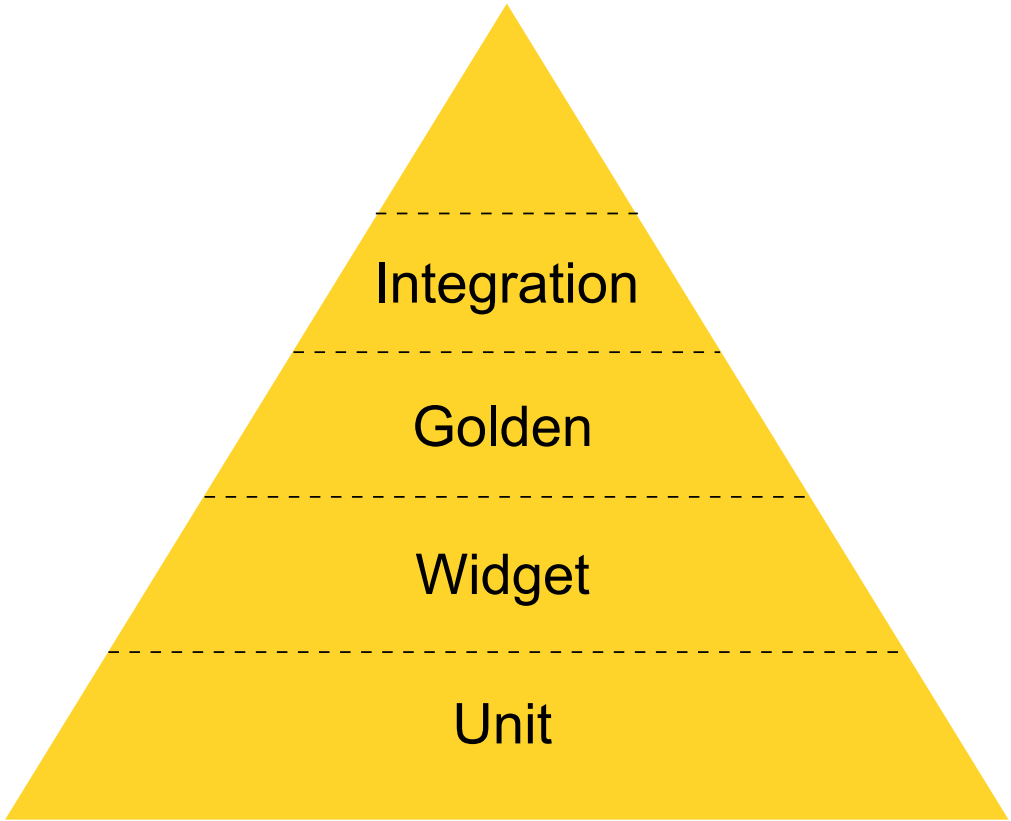
Flutter инструменты

```
› package:test
› package:flutter_test
› goldens
```



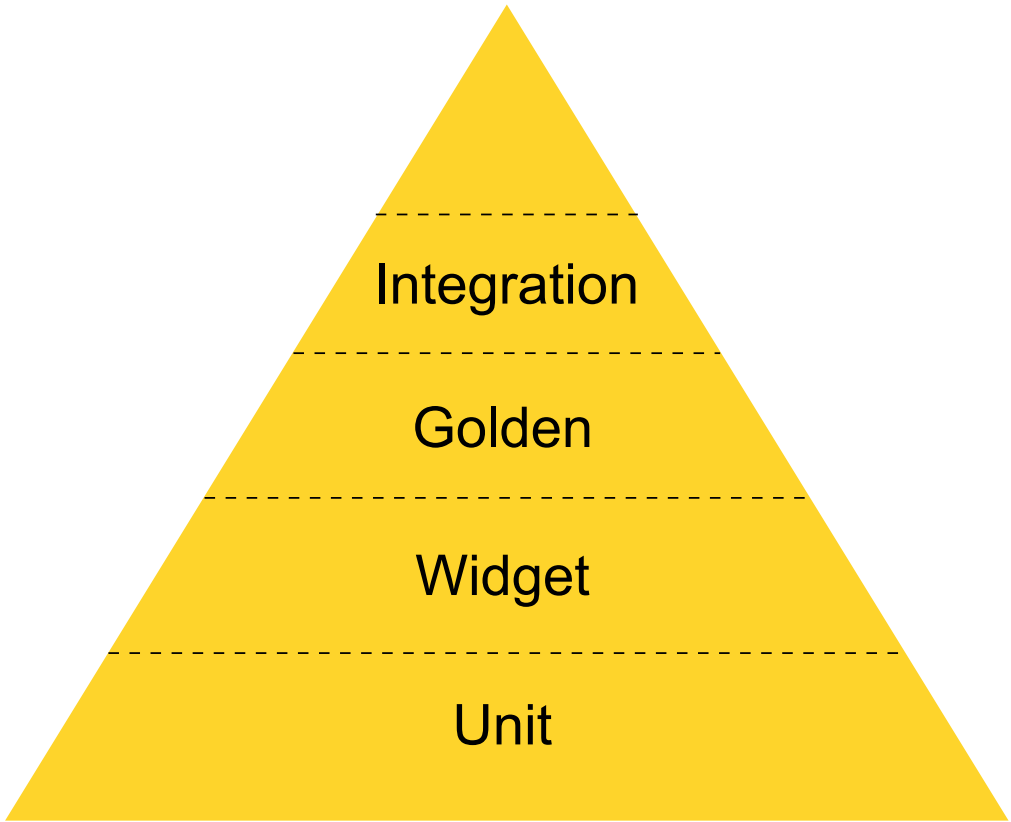
Flutter инструменты

```
〉package:test
〉package:flutter_test
 〉 goldens

〉flutter_driver
```



Flutter инструменты

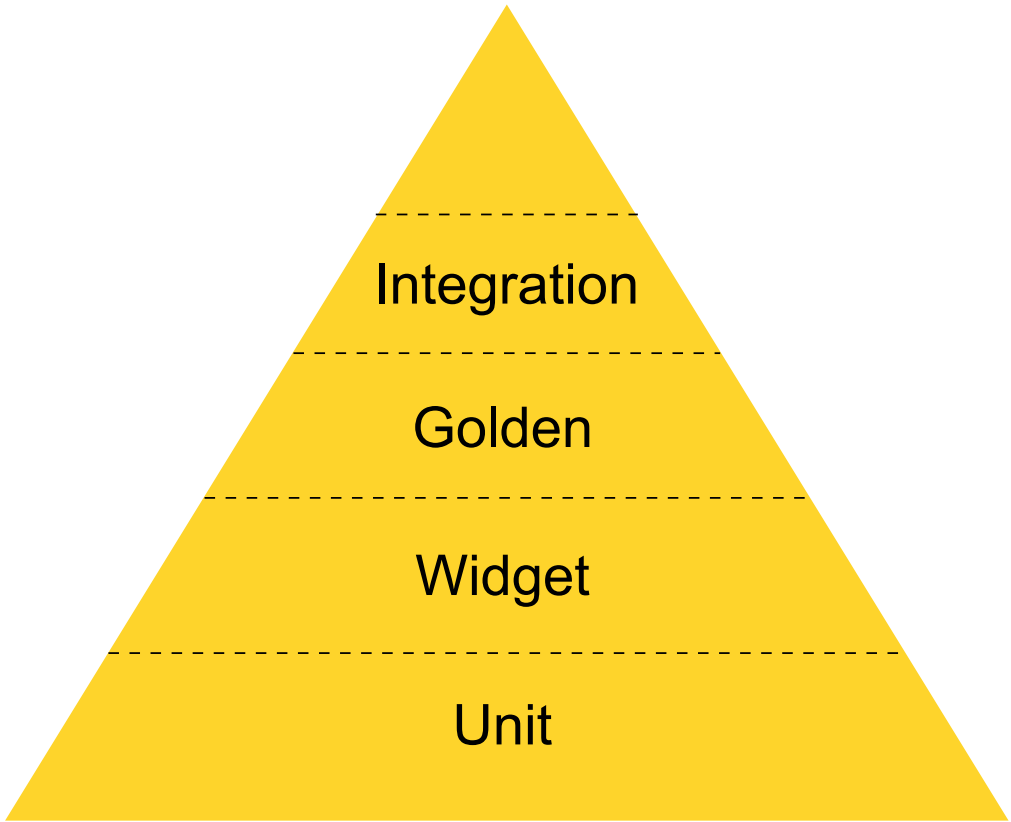
```
〉package:test
〉package:flutter_test
 › goldens

〉flutter_driver
 › package:e2e
```



Flutter инструменты

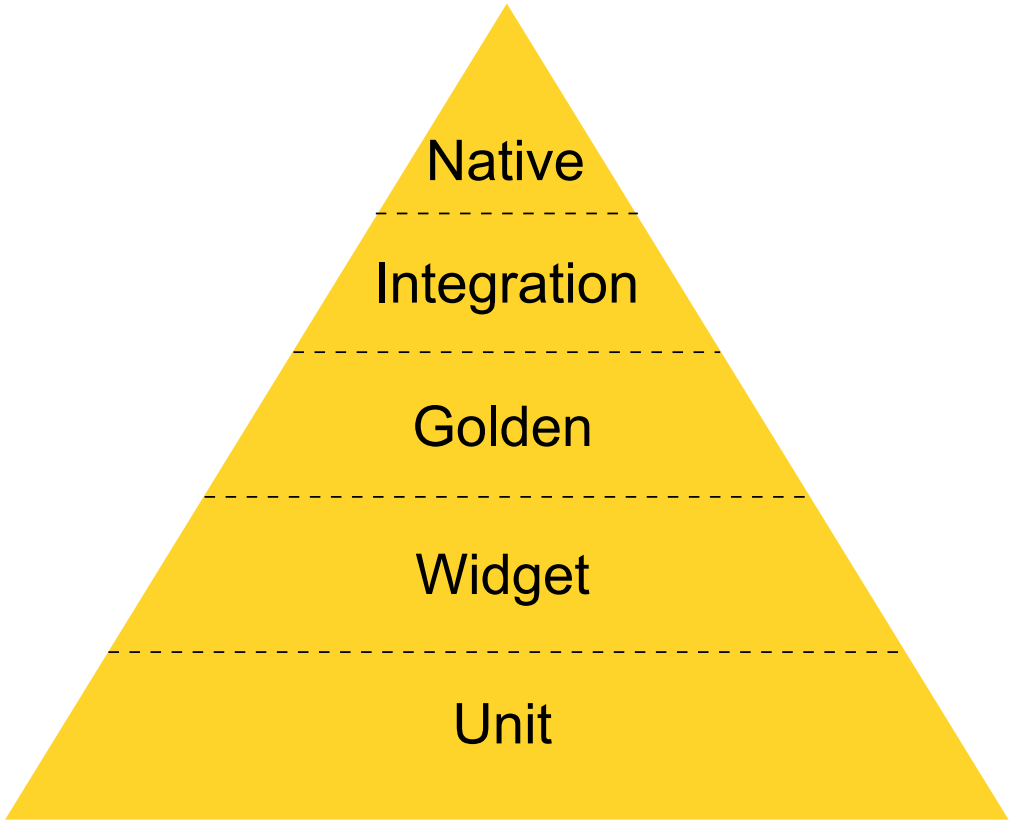
- package:test
- package:flutter_test
  - goldens

- flutter_driver
  - ~~package:e2e~~
  - package:**integration_test**



Flutter инструменты

- package:test
- package:flutter_test
  - goldens
- flutter_driver
  - ~~package:e2e~~
  - package:integration_test
- Native
  - unit tests XC
  - UI tests



Яндекс

Сириус
Образовательный центр

Спасибо

Богдан Лукин

Разработчик мобильных приложений

 2zerosix@yandex-team.ru

 @BogdanLukin

 2ZeroSix