

Частые проблемы

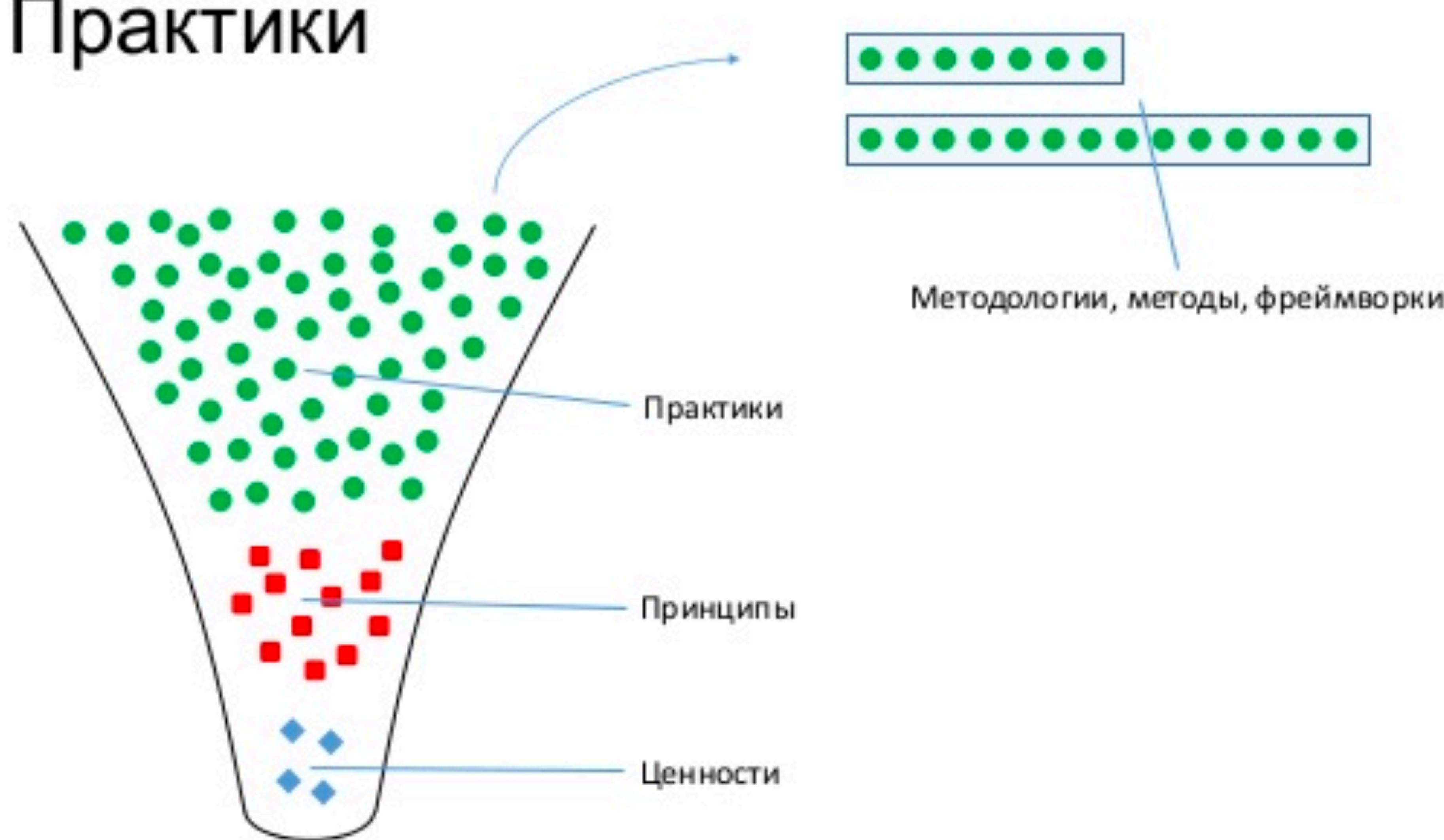
- тестирование отстает от разработки (очереди в тестировании)
- velocity команды измеряется по скорости тестирования
- автоматизация отстает еще сильнее
- тестировщики не начинают работать над фичей, пока не закончится разработка
- в конце спринта начинает пригорать
- ради выпуска релиза вы начинаете жертвовать качеством тестового покрытия
- фича не релизится, потому что не успели протестить
- не можете точно оценивать свое capacity из-за непрогнозируемого количества повторной работы и доработок (найденные баги инициируют возвраты на предыдущие этапы)

Профит от Agile Testing

Исследование Дэвида Рико

- 69 компаний участвовало в исследовании
- на 97% выросла производительность
- на 50% качество
- на 91% увеличилась точность планирования
- На 400% увеличилась удовлетворенность клиентов

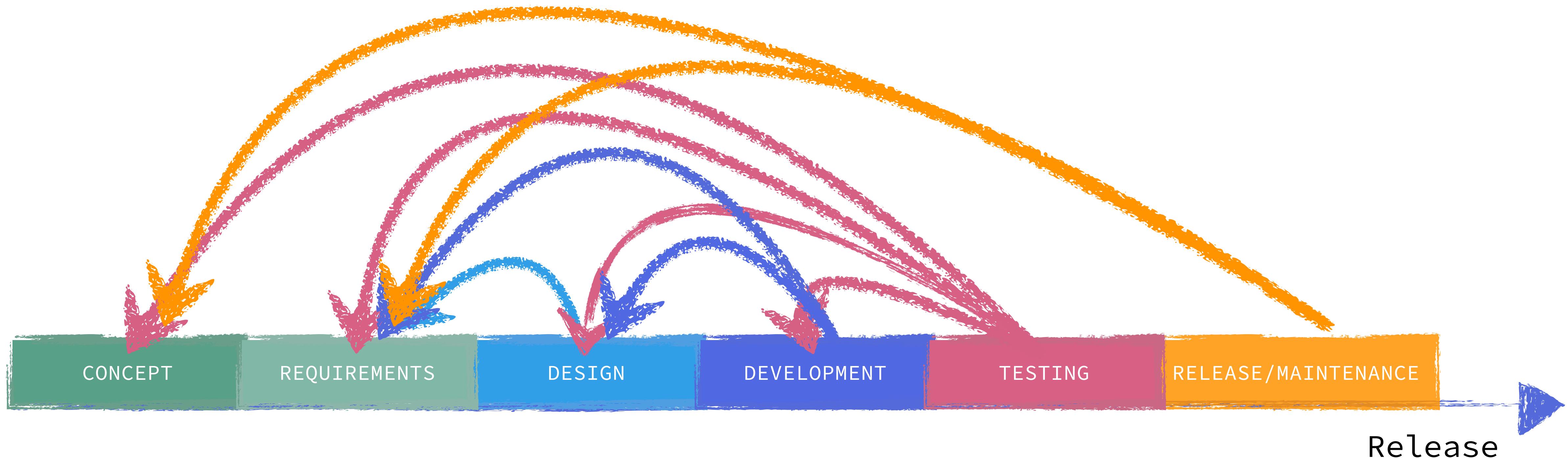
Практики



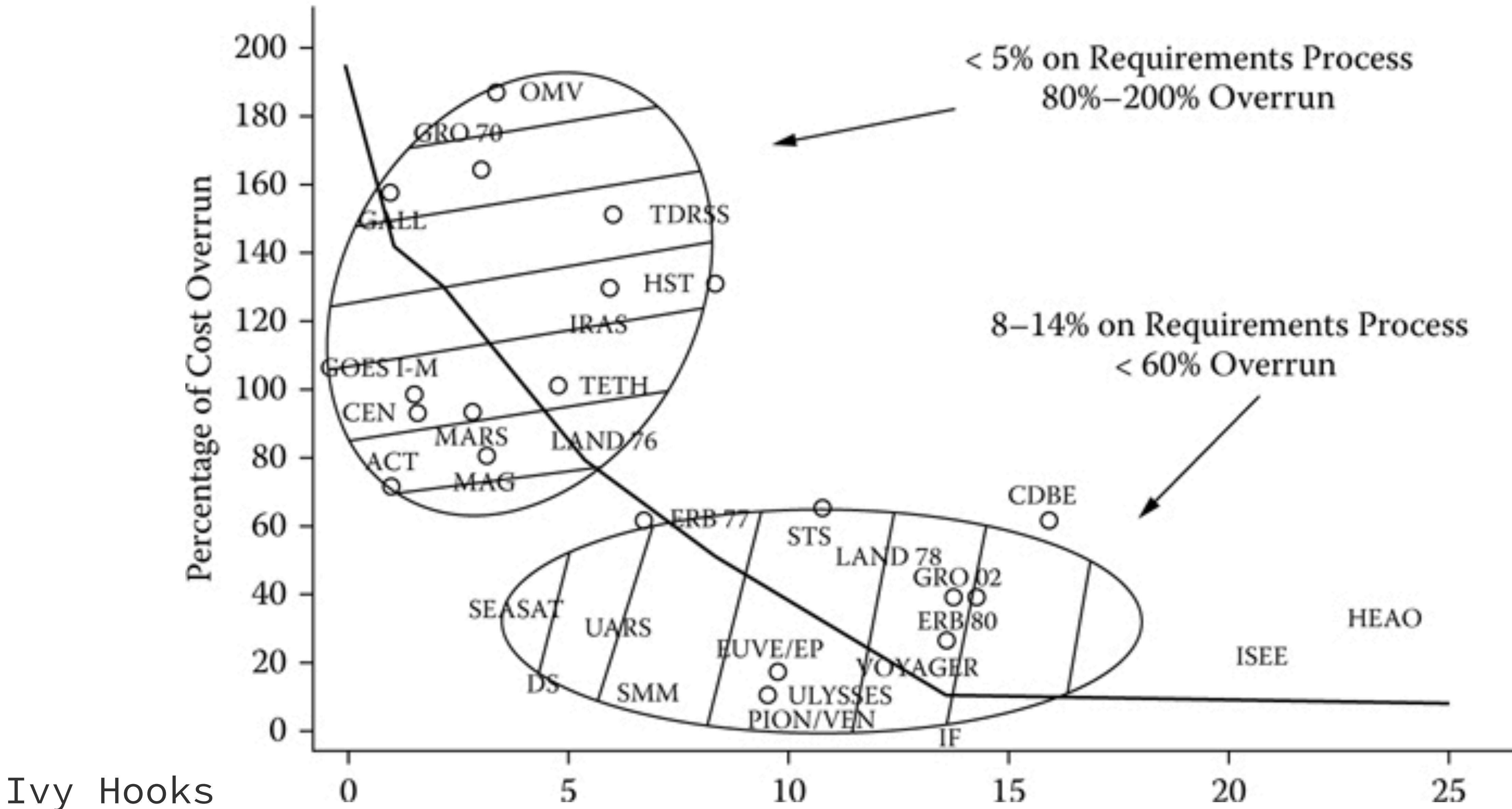
Методологии, методы, фреймворки

Возвраты

УТОЧНЕНИЕ ТРЕБОВАНИЙ, БАГИ



Value of Investment in Requirements Process



Причины уточнений требований

- ▶ Описан частичный набор функциональности
- ▶ Производительность не учитывается
- ▶ Неоднозначные требования
- ▶ Про безопасность забыли
- ▶ Интерфейсы не задокументированы
- ▶ Ошибочные и избыточные требования
- ▶ Слишком строгие требования
- ▶ Противоречивые требования

Пользовательские
требования
(user story + ac)

Приемочное
тестирование

Логический дизайн

Системное
тестирование

Физический дизайн

Интеграционное
тестирование

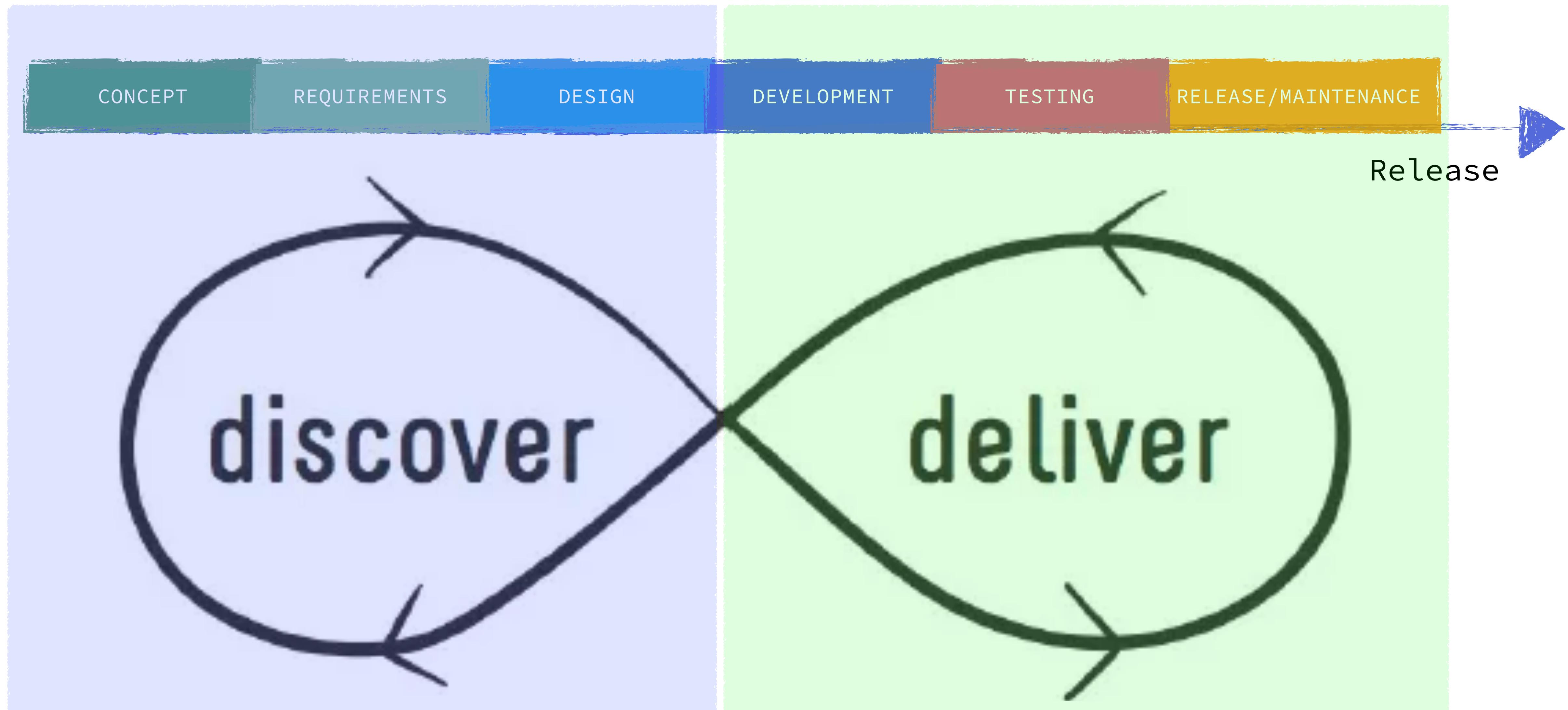
Дизайн кода

Юнит-тестирование

Разработка



ЖИЗНЕННЫЙ ЦИКЛ





THE TESTING Manifesto

we value:

Testing throughout
OVER
testing at the end

Preventing bugs
OVER
finding bugs

Testing understanding
OVER
checking functionality

Building the best system
OVER
breaking the system

Team responsibility for quality
OVER
tester responsibility

12



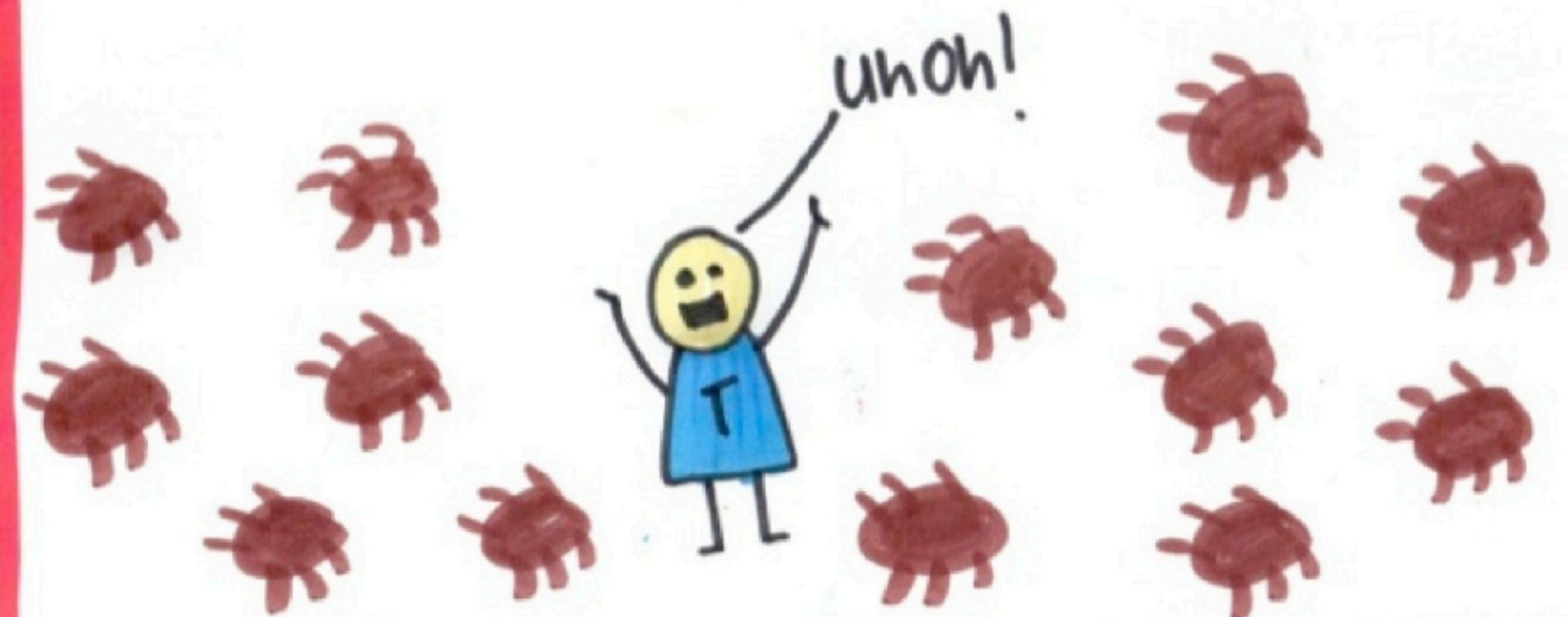
МЫ ЦЕНИМ:

ПРЕДОТВРАЩЕНИЕ
БАГОВ

БОЛЬШЕ ЧЕМ

ПОИСК БАГОВ

Finding bugs...



МЫ ЦЕНИМ:

ТЕСТИРОВАНИЕ В
ПРОЦЕССЕ

БОЛЬШЕ ЧЕМ

ТЕСТИРОВАНИЕ В
КОНЦЕ



МЫ ЦЕНИМ:

**ТЕСТИРОВАНИЕ
КАЧЕСТВА
ПРОДУКТА**

**БОЛЬШЕ ЧЕМ
ПРОВЕРКУ
НА СООТВЕТСТВИЕ
ТРЕБОВАНИЯМ**



МЫ ЦЕНИМ:
КОМАНДНУЮ
ОТВЕТСТВЕННОСТЬ
ЗА КАЧЕСТВО

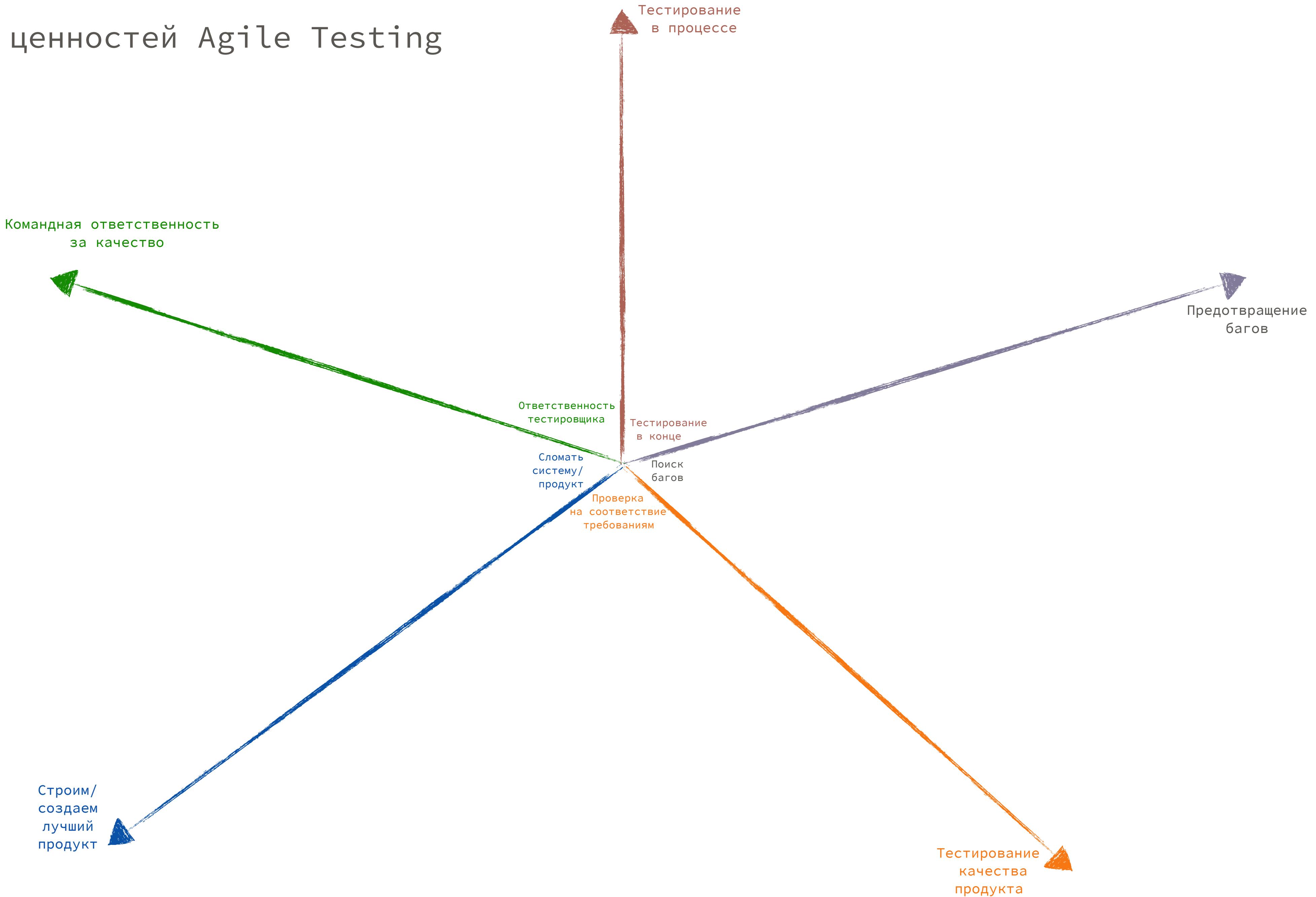
БОЛЬШЕ ЧЕМ
ОТВЕТСТВЕННОСТЬ
ТЕСТИРОВЩИКА

**Whole team takes
responsibility for quality**

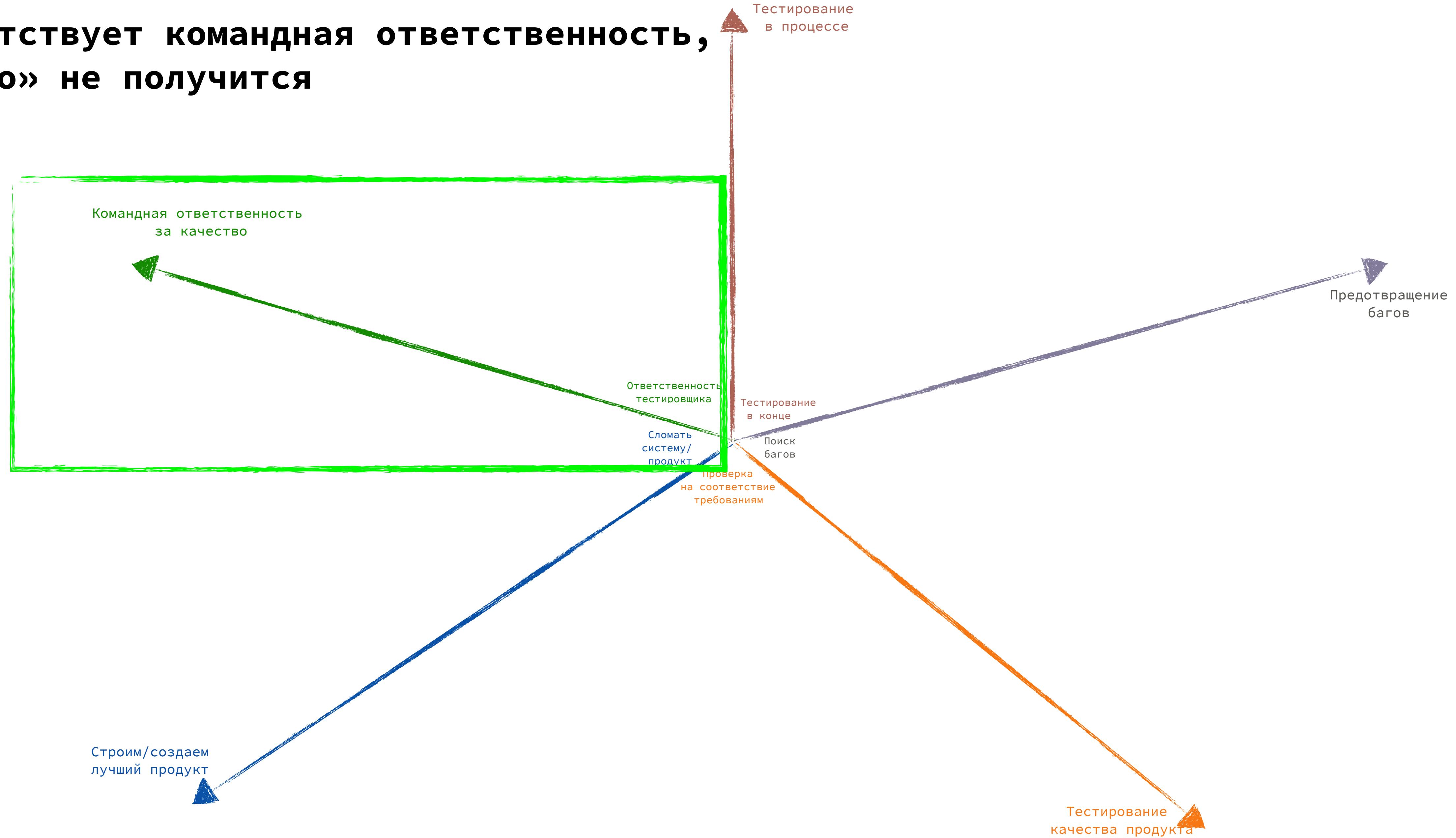


**МЫ ХОТИМ
ПОСТРОИТЬ ЛУЧШУЮ СИСТЕМУ
А НЕ СЛОМАТЬ СИСТЕМУ**

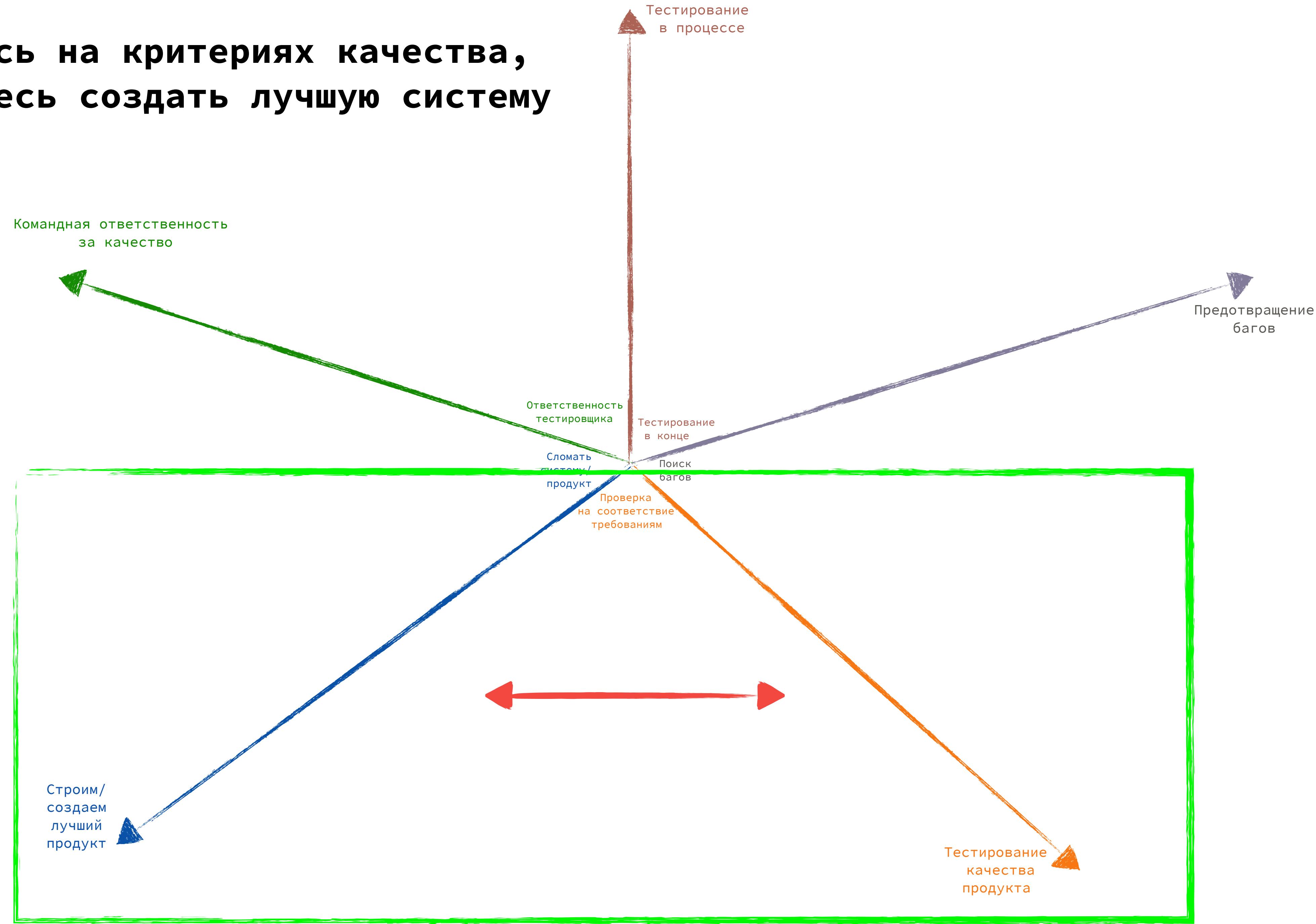
Манифест ценностей Agile Testing

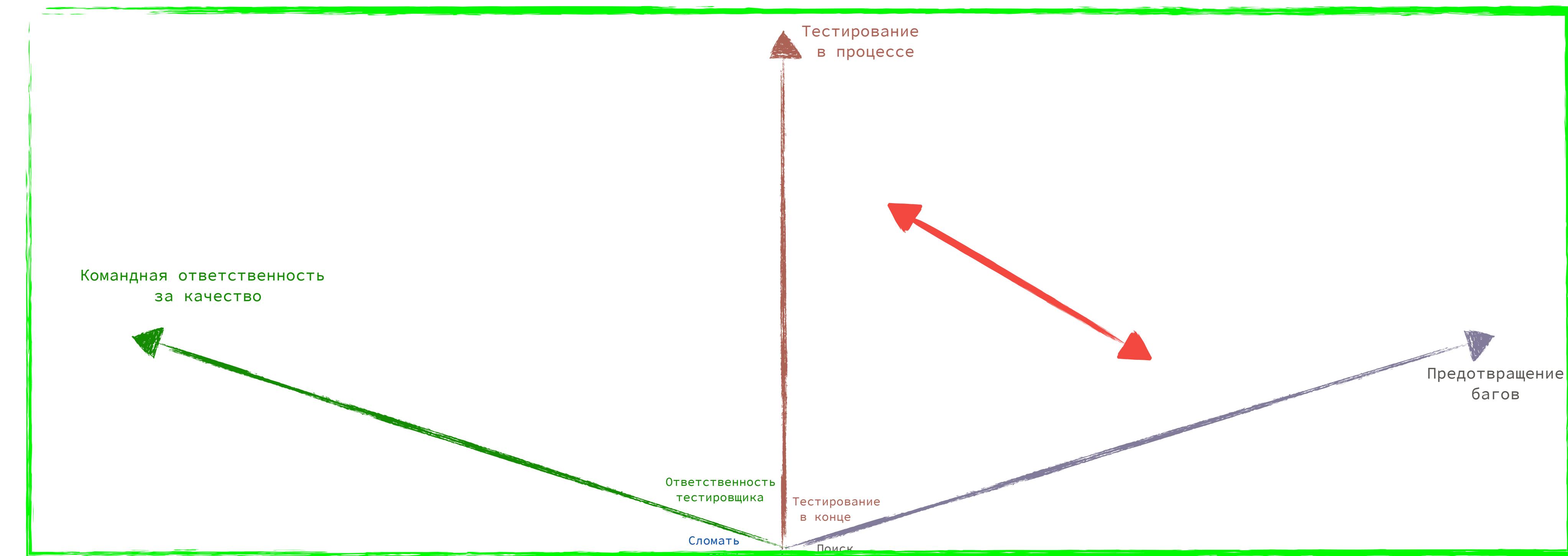


Если отсутствует командная ответственность, то «ничего» не получится



**Фокусируясь на критериях качества,
вы пытаетесь создать лучшую систему**



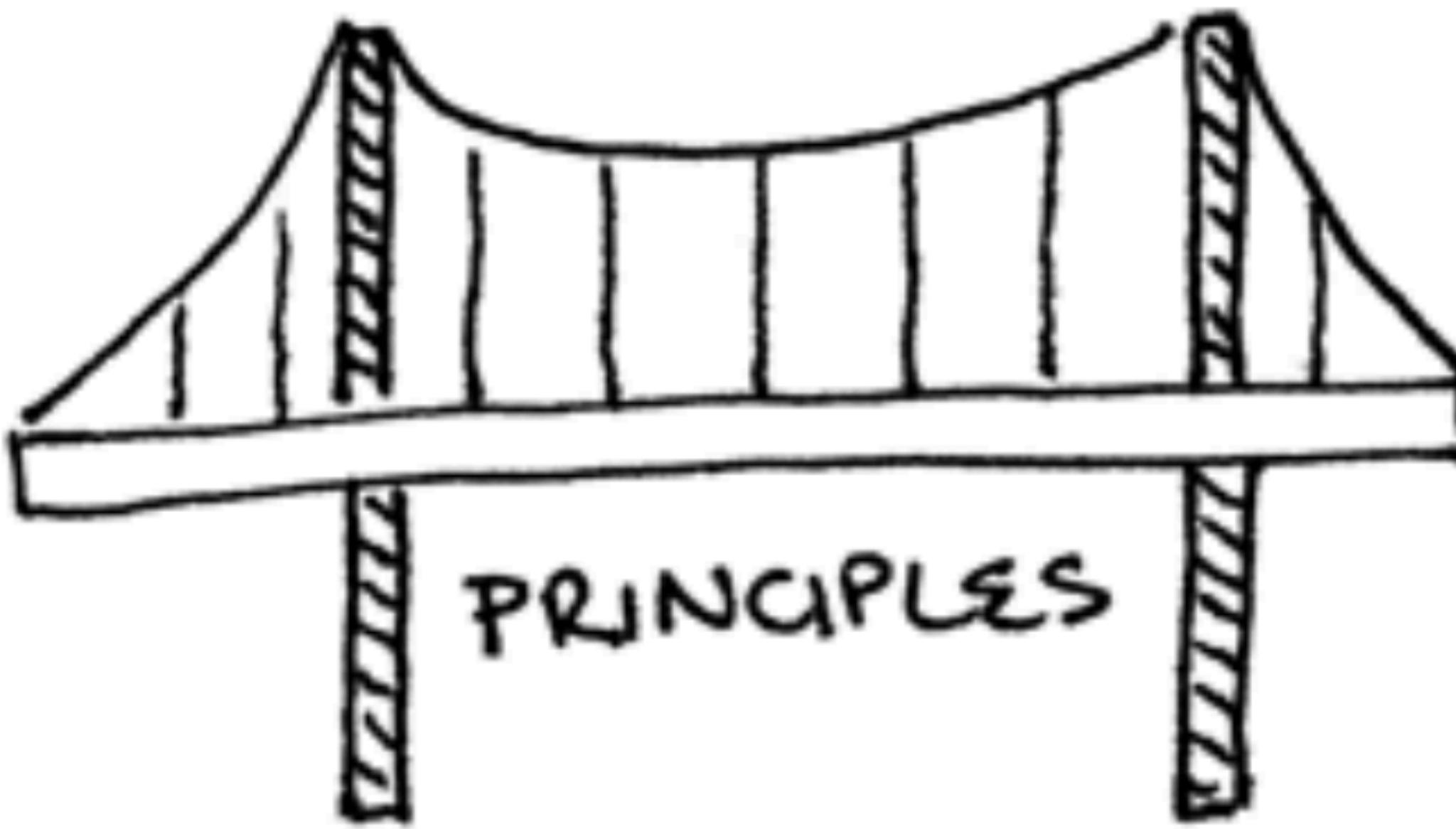


**Пока не начнете
пытаться предотвращать
баги, мотивации
тестировать в процессе
не прибавится**

Строим/
создаем
лучший
продукт

Тестирование
качества
продукта

VALUES



PRACTICES

Групповое задание

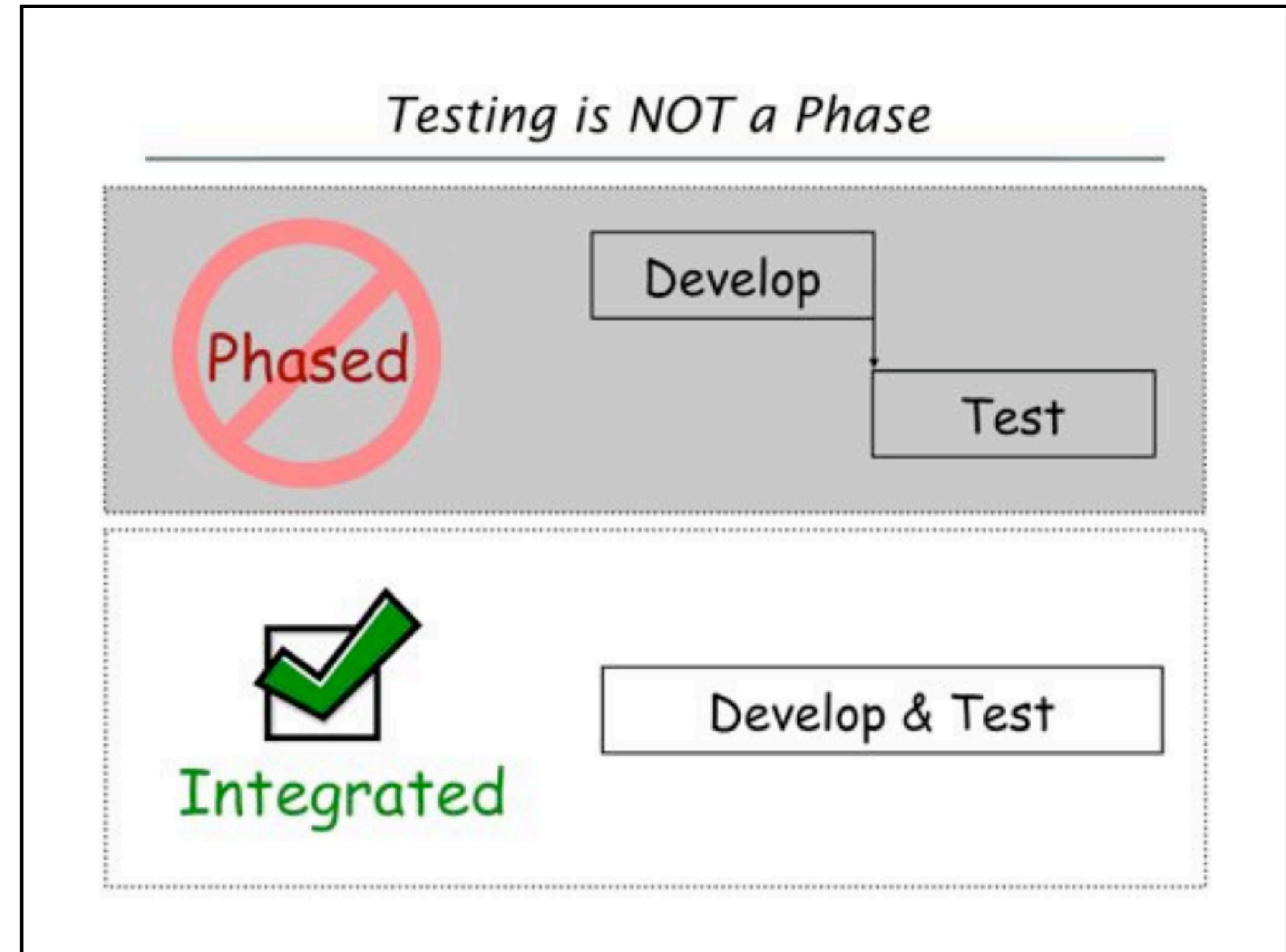


1. У группы – по 4 принципа
2. На каждый принцип надо вспомнить или придумать практики, соответствующие этому принципу
3. Записать на флипчарт практики

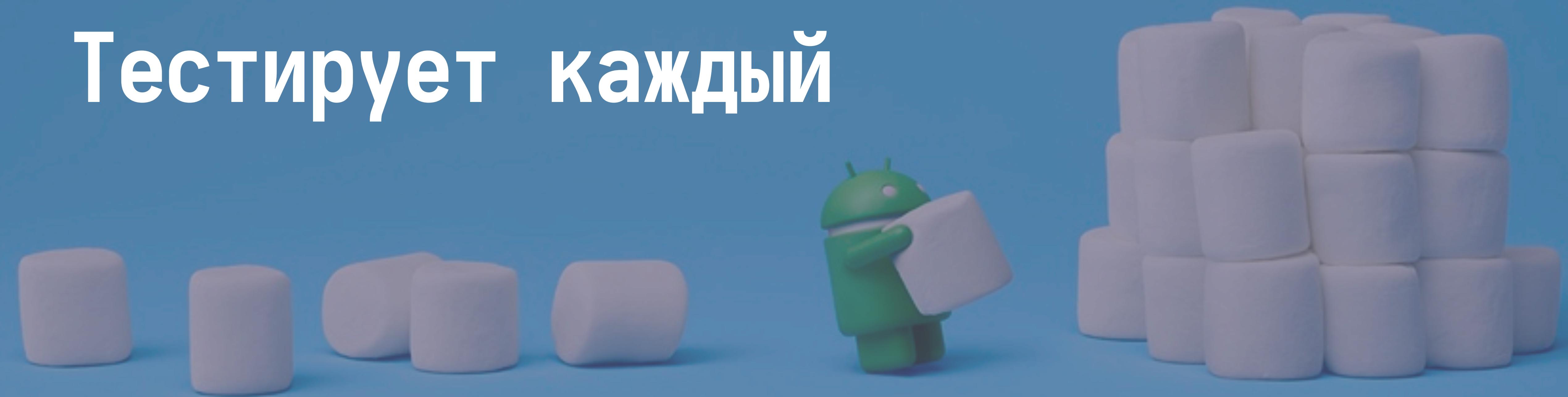
Тестирование двигает проект вперед



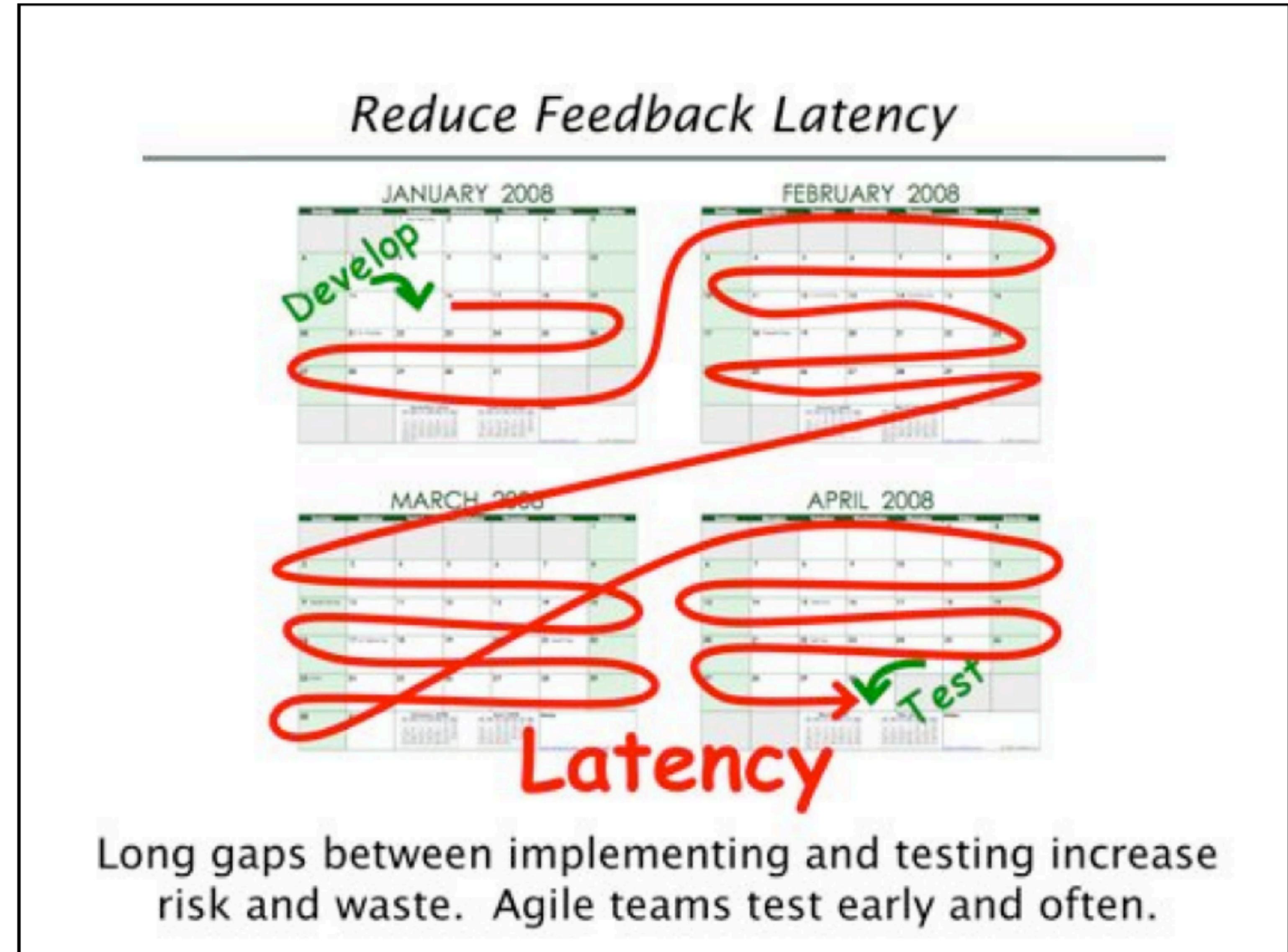
**Тестирование –
это не фаза/этап...**



Тестирует каждый



Укорачивание петли обратной связи

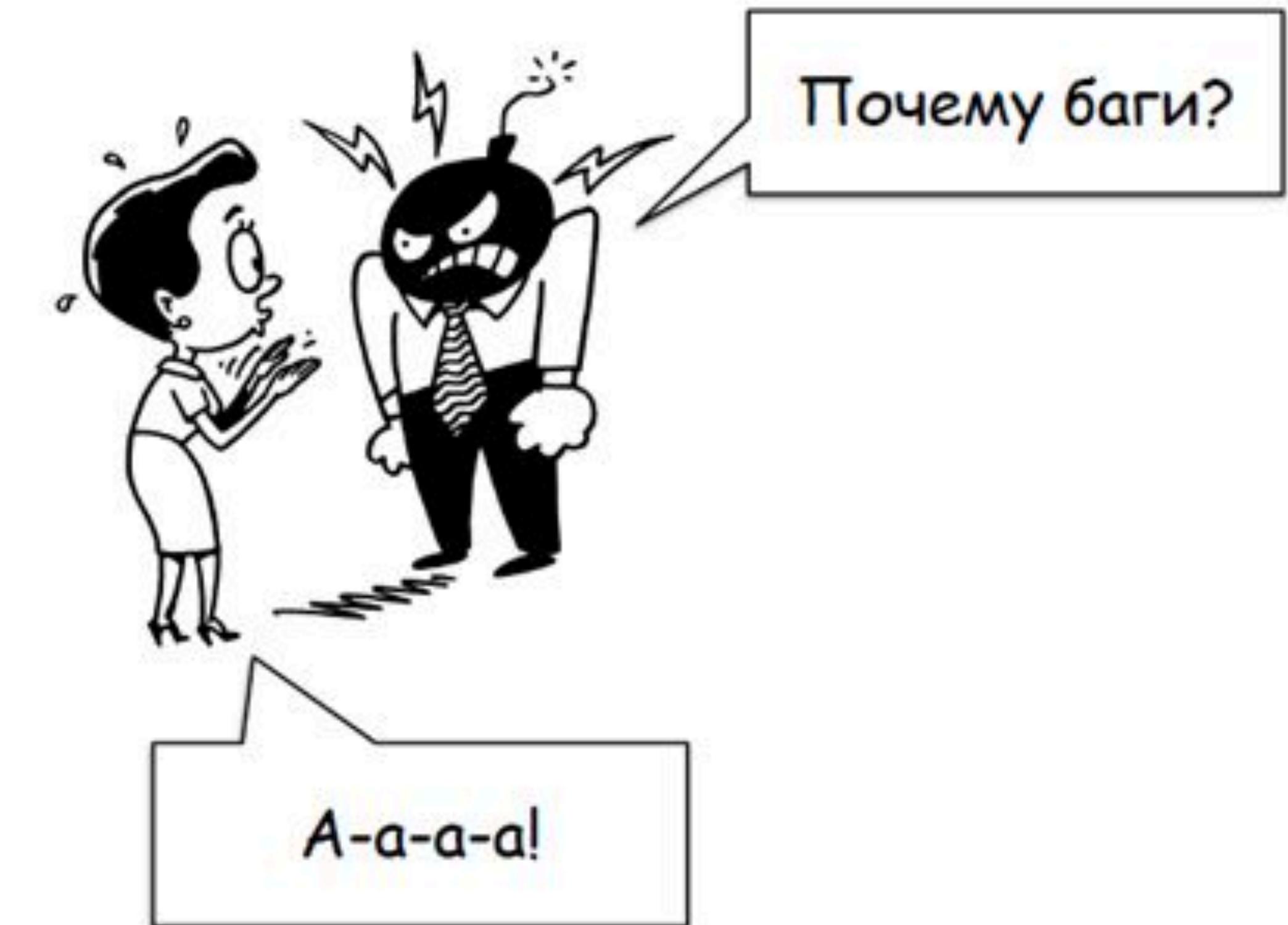


Тест формирует
ожидание



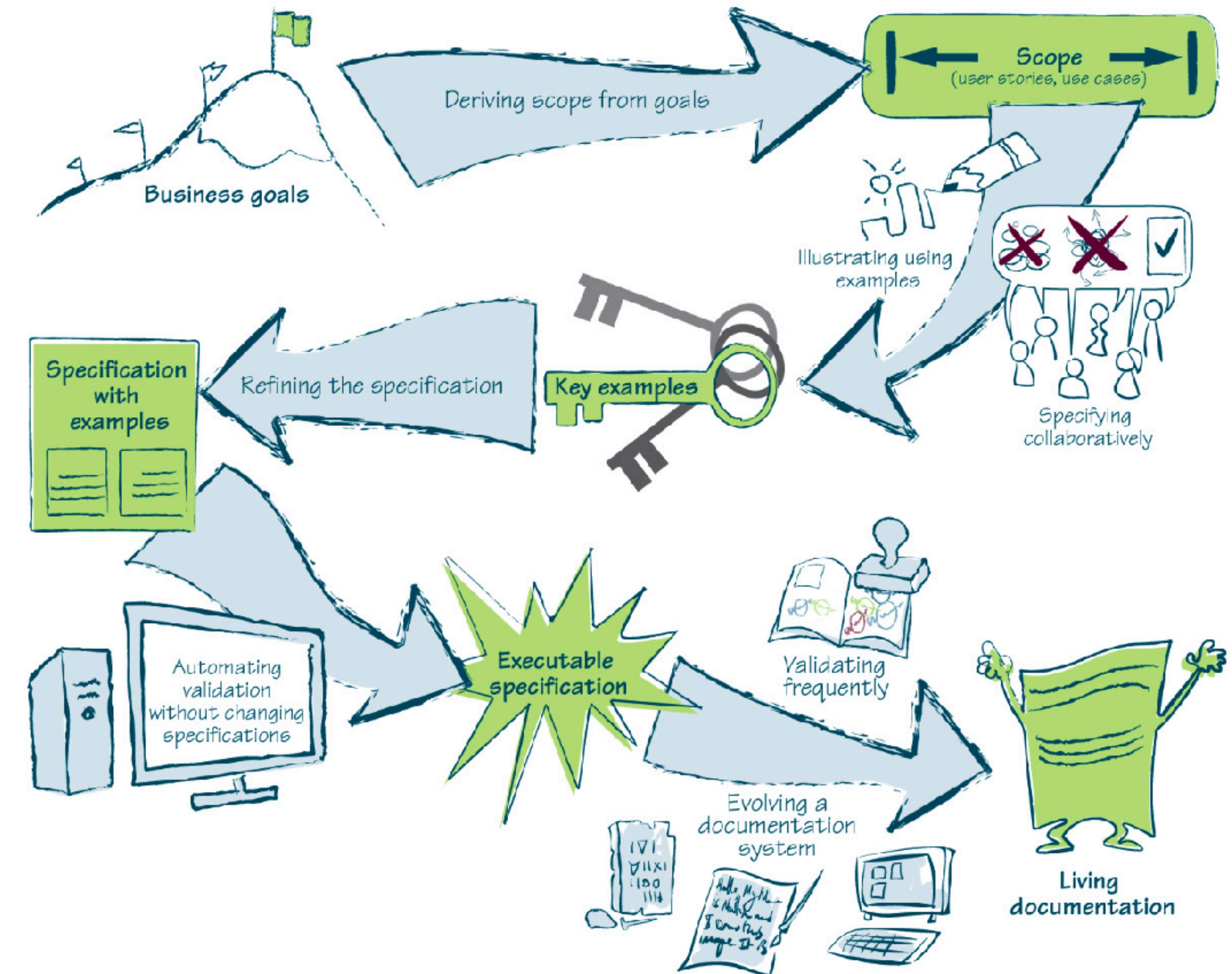
Некоторое время спустя

Нулевая терпимость к ошибкам

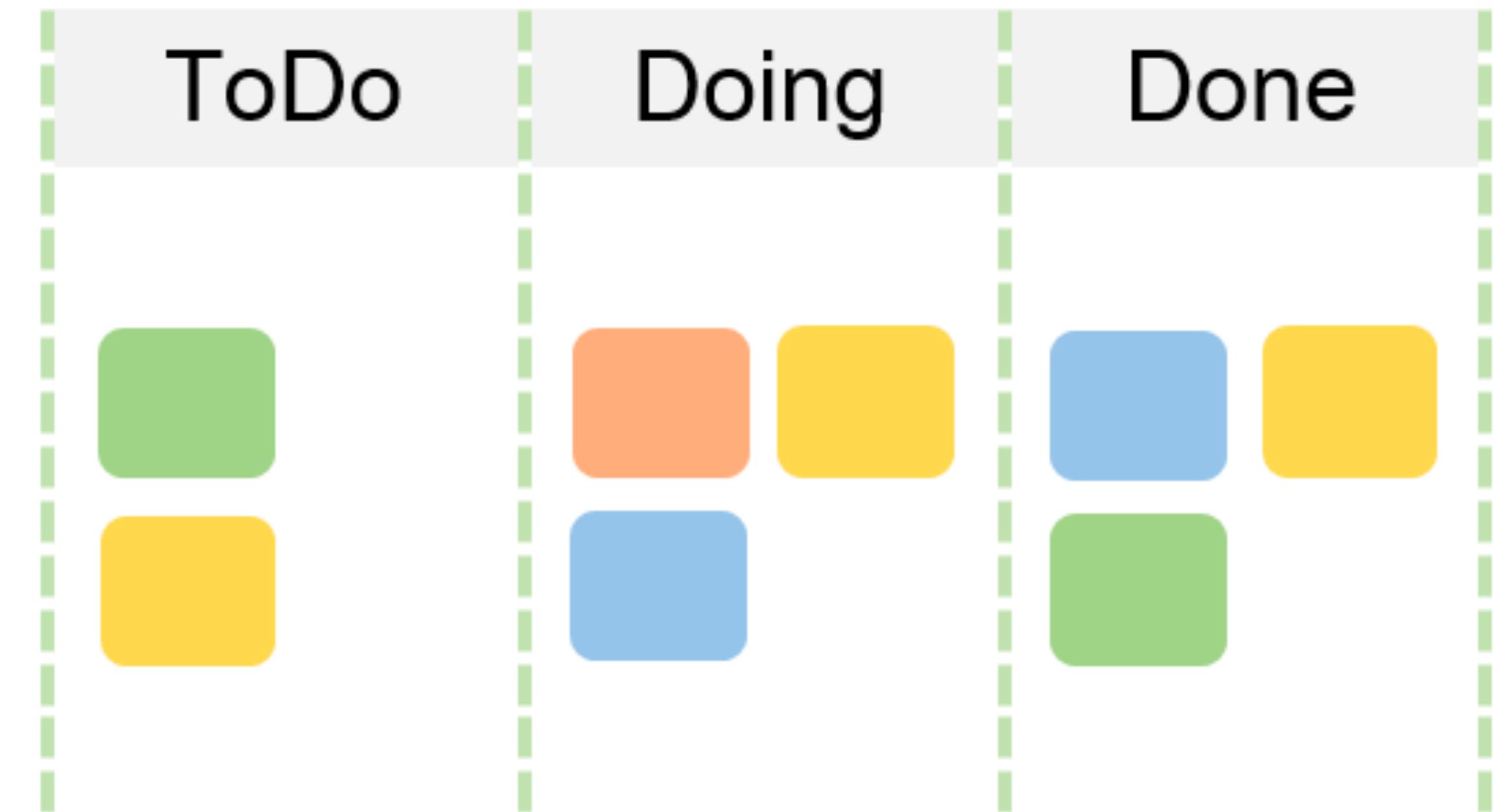


Легковесная документация

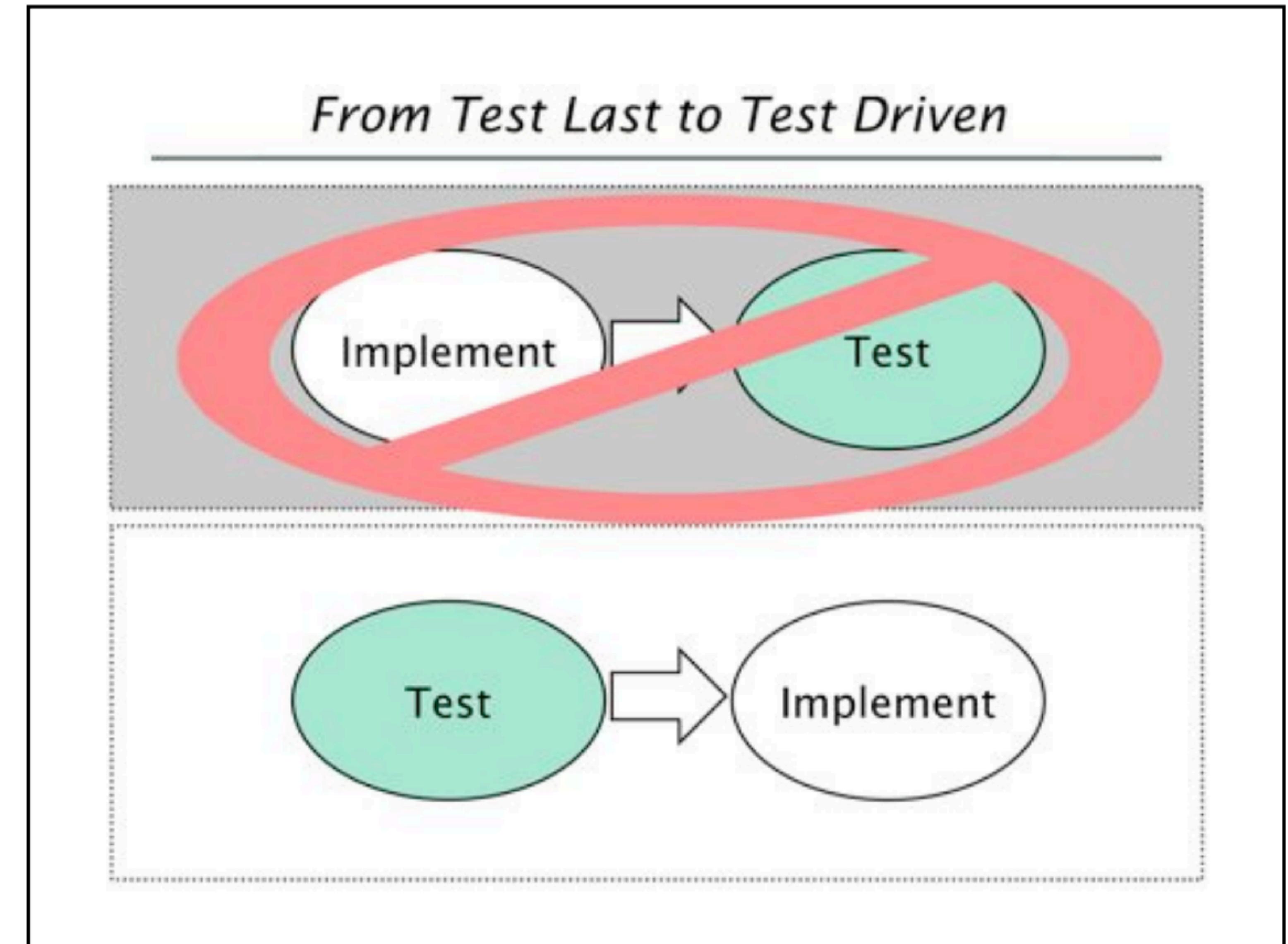
Key process patterns of Specification by Example



**Протестировано –
часть "Done"**



**Сначала тест –
потом код**





SHIFT LEFT TESTING

Сім — це підход до розвитку, який використовується в індустрії та в сучасному світі.

Це — це не тільки інноваційний підхід, а також методика, яка допомагає виявляти проблеми в програмах досить рано, що дозволяє зберегти багато часу та ресурсів.

Давайте дізнатимемося, що це за підхід, яким він відрізняється від інших, та чому він став таким популярним.

Що таке Shift Left Testing?

Shift Left Testing — це підхід до розвитку та тестування програмного забезпечення, який передбачає виконання тестів на ранніх стадіях розробки.

Це означає, що тестування починається з моменту створення коду, а не тільки після завершення всіх функціональних тести.

Цей підхід дозволяє виявити проблеми вже на ранніх стадіях розробки, що дозволяє зберегти багато часу та ресурсів.

Що відрізняє Shift Left Testing від інших підходів?

Shift Left Testing відрізняється від інших підходів тим, що він фокусується на виявленні проблем вже на ранніх стадіях розробки.

Це означає, що тестування починається з моменту створення коду, а не тільки після завершення всіх функціональних тести.

Цей підхід дозволяє виявити проблеми вже на ранніх стадіях розробки, що дозволяє зберегти багато часу та ресурсів.

Що відрізняє Shift Left Testing від інших підходів?

Shift Left Testing відрізняється від інших підходів тим, що він фокусується на виявленні проблем вже на ранніх стадіях розробки.

Це означає, що тестування починається з моменту створення коду, а не тільки після завершення всіх функціональних тести.

Цей підхід дозволяє виявити проблеми вже на ранніх стадіях розробки, що дозволяє зберегти багато часу та ресурсів.

Що відрізняє Shift Left Testing від інших підходів?

Shift Left Testing відрізняється від інших підходів тим, що він фокусується на виявленні проблем вже на ранніх стадіях розробки.

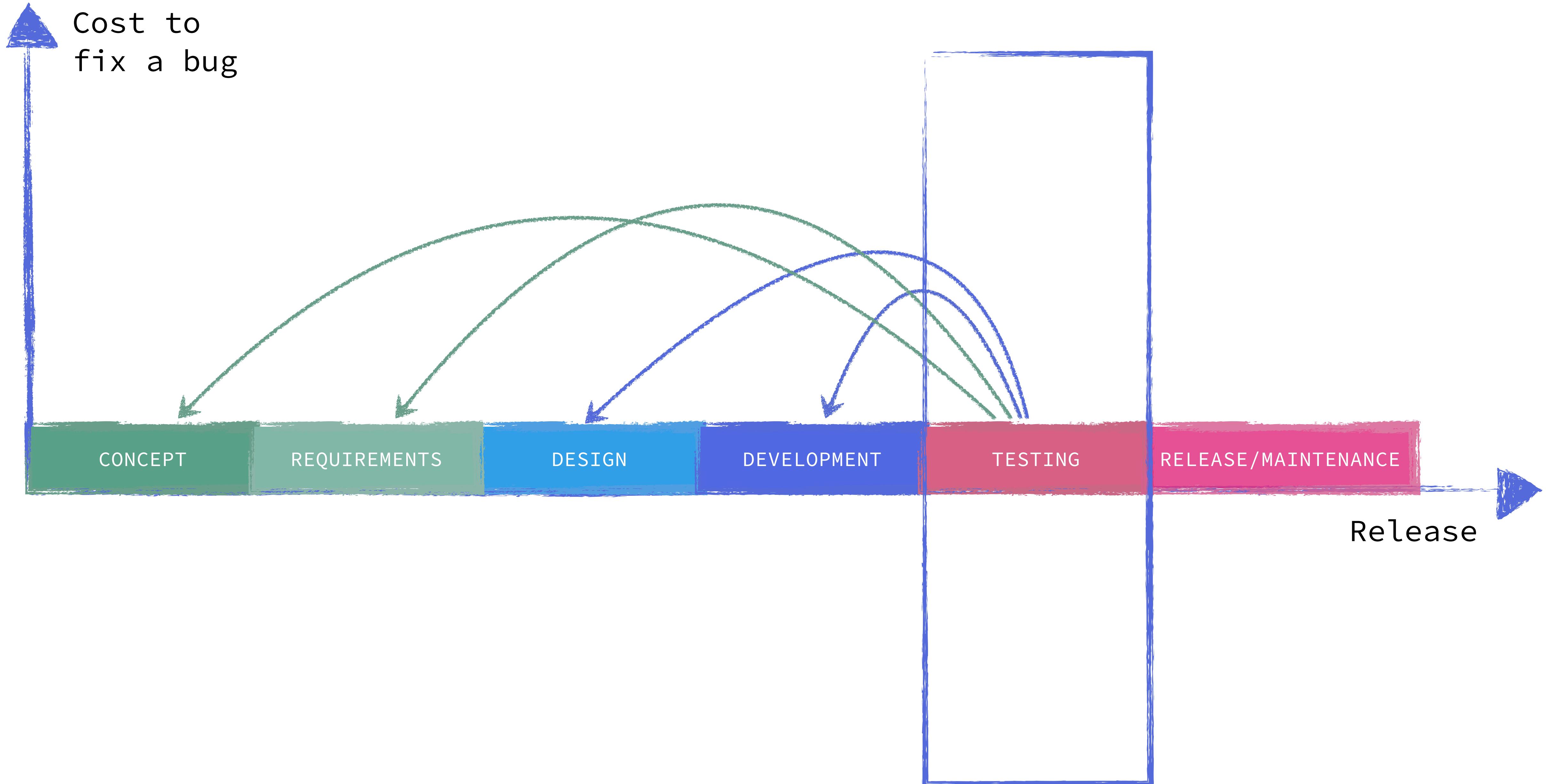
Це означає, що тестування починається з моменту створення коду, а не тільки після завершення всіх функціональних тести.

Цей підхід дозволяє виявити проблеми вже на ранніх стадіях розробки, що дозволяє зберегти багато часу та ресурсів.

Що відрізняє Shift Left Testing від інших підходів?

Shift Left Testing відрізняється від інших підходів тим, що він фокусується на виявленні проблем вже на ранніх стадіях розробки.

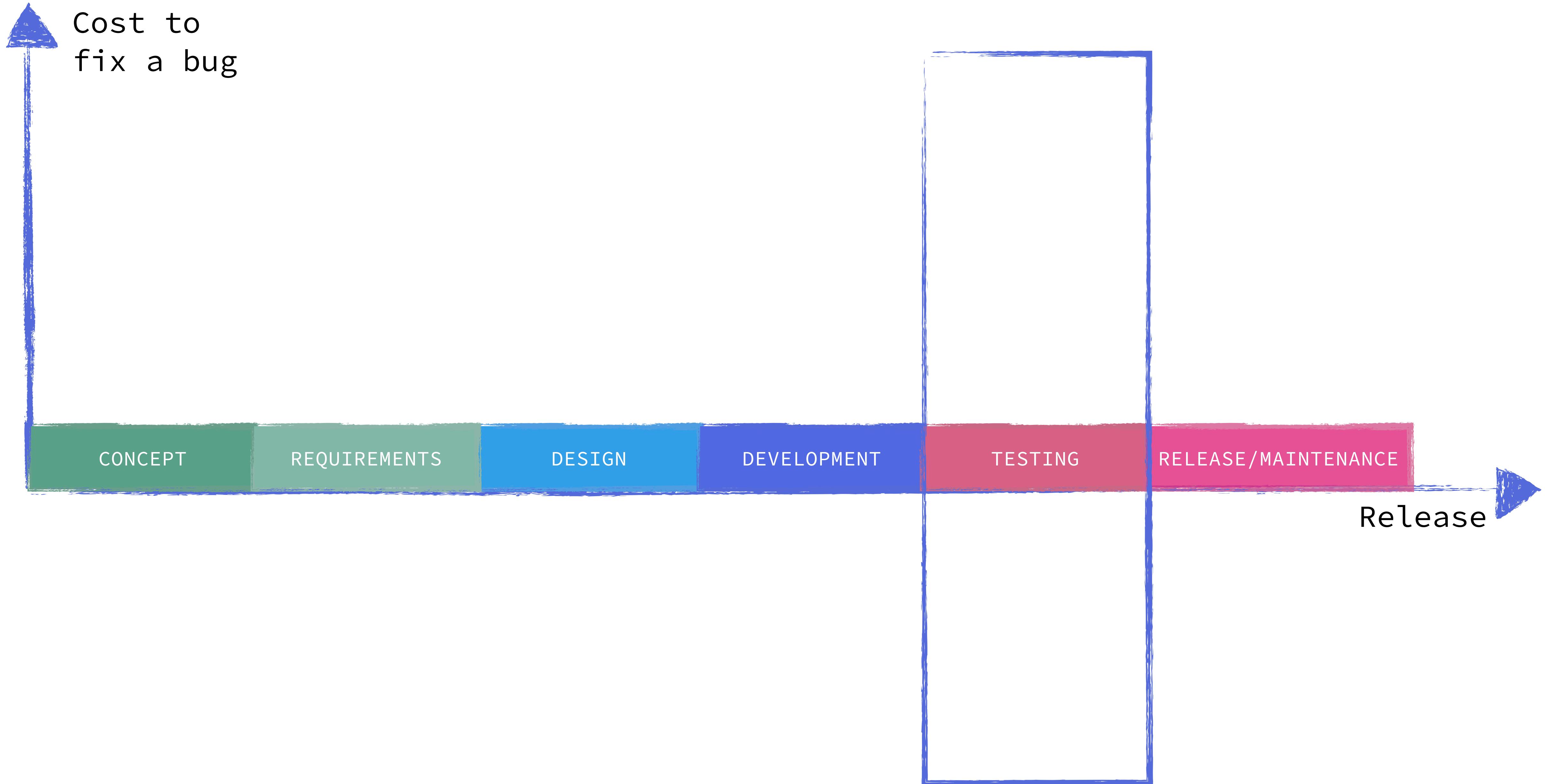
Це означає, що тестування починається з моменту створення коду, а не тільки після завершення всіх функціональних тести.



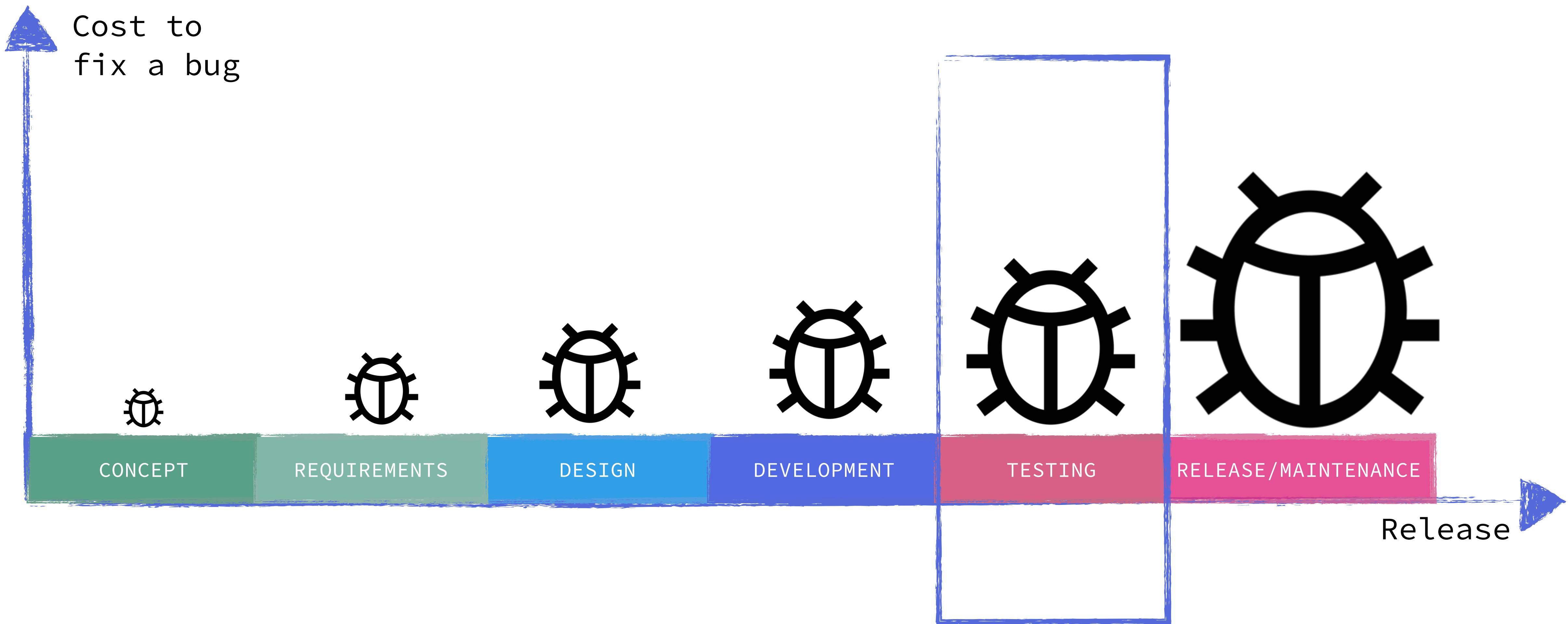
Каким образом
"сдвиг влево"
уменьшает затраты?



Меньше багов –
меньше стоимость
разработки



Стоимость бага



Затраты на исправление бага

1. Обработка звонков в колл-центр
2. Обработка звонков в отдел технической поддержки
3. Передача бага на исправление
4. Погружение новой команды в контекст для исправления бага
5. Исправление бага
6. Установка новой (исправленной) версии продукта
7. Исправление репутационных рисков
8. Коммуникации со СМИ
9. Судебные процессы

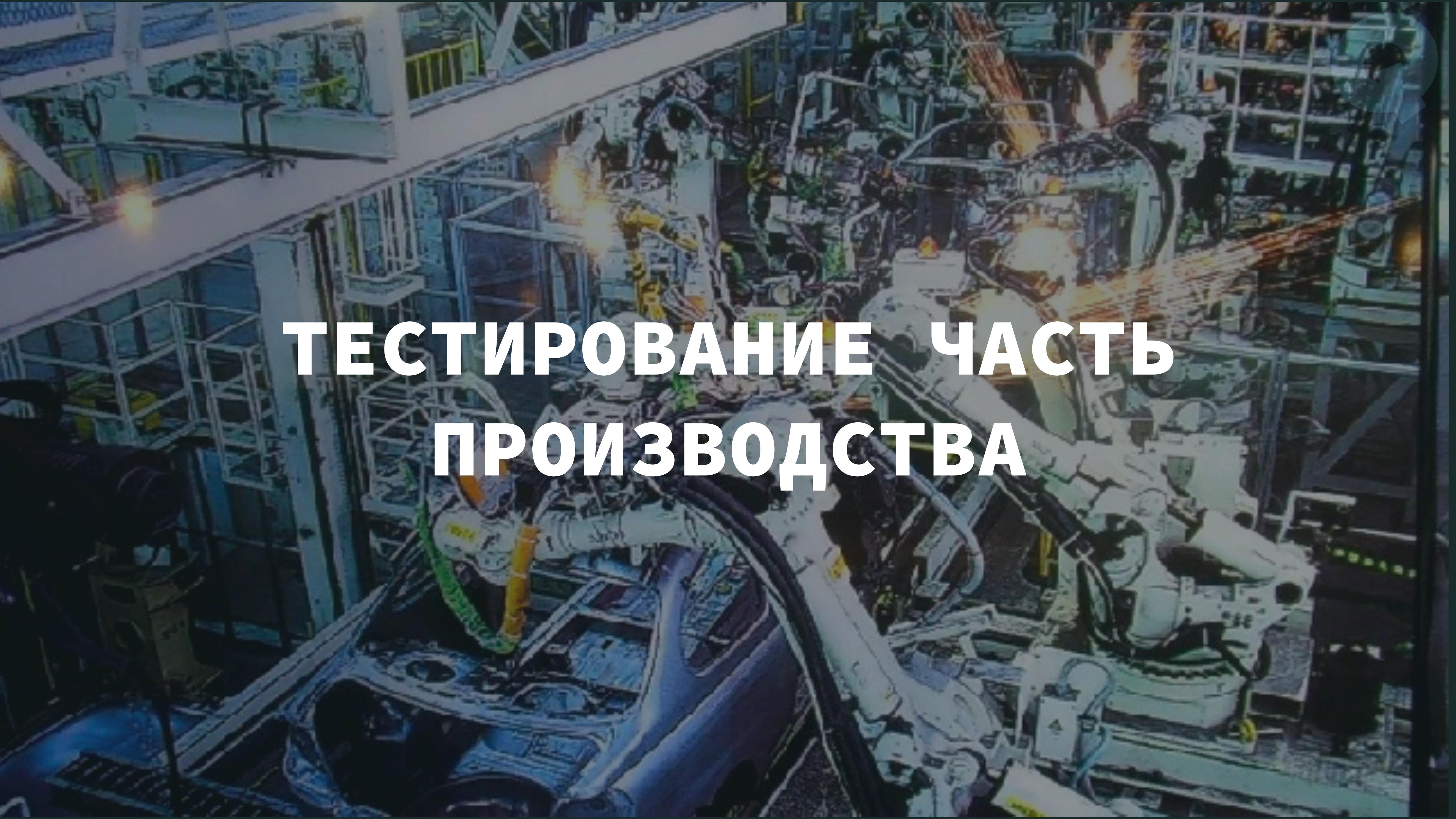
Затраты на исправление бага

О **Недополученная прибыль – ресурсы**, которые можно было потратить на реализацию фичи – **потратили** на исправление бага

Исправлять
баги дорого

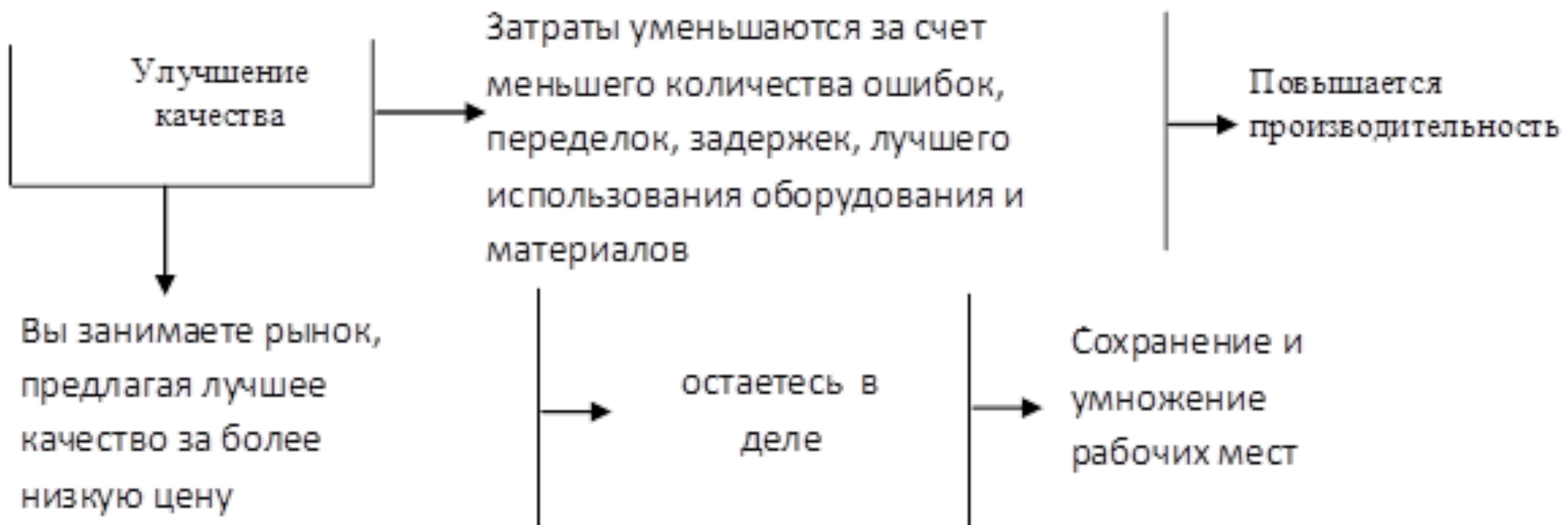


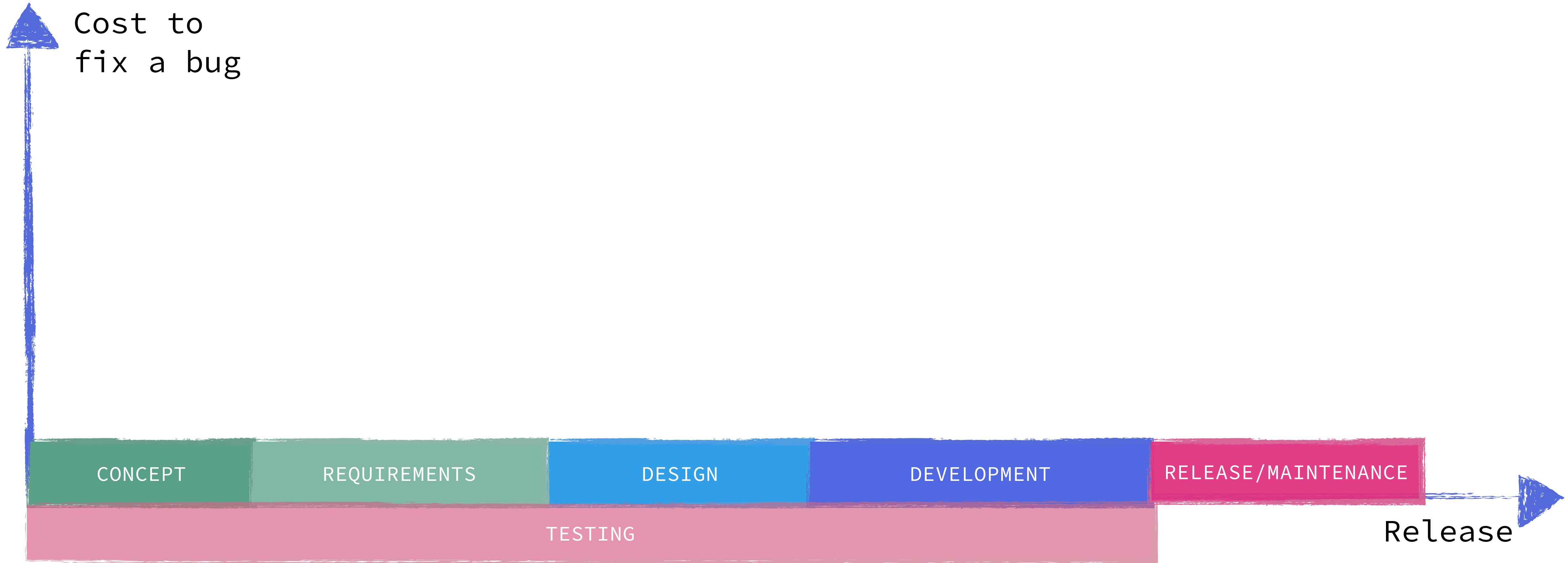
**Наиболее дешевый способ
разработки – если
жизненный цикл артефакта
длился
не более 1 итерации**



ТЕСТИРОВАНИЕ ЧАСТЬ
ПРОИЗВОДСТВА

Э. Дэминг ("Выход из кризиса"), 1984г







Не экономьте

на исправлении дефектов -

предотвращайте их



- ПРЕДОТВРАЩАТЬ ОШИБКИ
- ВЫПУСТИТЬ РЕЛИЗ В СРОК
- УСКОРИТЬ РАЗРАБОТКУ
- ВЫЯВИТЬ СКРЫТЫЕ ТРЕБОВАНИЯ