



# Разработка через тестирование Why TDD?

Test Driven Development

Ivan Dyachenko <[IDyachenko@luxoft.com](mailto:IDyachenko@luxoft.com)>

# Содержание

1

Agile

2

Уровни качества

3

TDD – хороший дизайн

4

Почему автоматизированные тесты?

5

Почему Test First?

6

Почему TDD?

7

Вопросы

# Почему стоит применять TDD?

1

TDD придуман и одобрен коллективным разумом

2

Одобен выдающимися специалистами

3

Применяется уже далеко не первый год

# Что такое тестирование?



Тестирование – способ обеспечения  
качества продукта

# Что такое тестирование?

С технической точки зрения, тестирование заключается в:

- Выполнении приложения на некотором множестве исходных данных
- Сверке получаемых результатов с заранее известными (эталонными) с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам

# Что такое тестирование?



- Тестирование является одной из основных фаз разработки программного продукта (наряду с Дизайном приложения и Разработкой кода)
- Оно характеризуется достаточно большим вкладом в суммарную трудоемкость разработки продукта

Заинтересованными сторонами являются:

- Заказчик продукта
- Спонсор
- Конечный пользователь
- Разработчики
- Тестировщики продукта
- Инженеры поддержки
- Сотрудники отделов маркетинга, обучения и продаж

Таким образом, постановка задачи обеспечения качества продукта выливается в задачи:

- Определения заинтересованных лиц
- Их критериев качества
- Нахождения оптимального решения, удовлетворяющего этим критериям



# Уровни качества ПО



# Уровни тестирования



- Системное тестирование, в ходе которого тестируется система в целом
- Интеграционное тестирование, в ходе которого тестируются группы взаимодействующих модулей и компонент системы
- Модульное тестирование, в ходе которого тестируются отдельные компоненты

Основной задачей системного тестирования является проверка как функциональных, так и нефункциональных требований в системе в целом

В ходе системного тестирования выявляются следующие дефекты:

- Неверное использование ресурсов системы
- Непредусмотренные комбинации данных пользовательского уровня
- Несовместимость с окружением
- Непредусмотренные сценарии использования
- Отсутствующая или неверная функциональность
- Неудобство использования и т.д.

# Интеграционное тестирование



Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами)

# Интеграционное тестирование



Интеграционное тестирование так же может проводиться на различных уровнях:

- Компонентный: проверяется взаимодействие между компонентами системы после проведения компонентного (модульного) тестирования
- Системный: проверяется взаимодействие между разными системами после проведения системного тестирования

# Компонентное или Модульное тестирование



Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности (модули программ, объекты, классы, функции и т.д.)

# Почему TDD? Потому что:



## Во многих проектах:

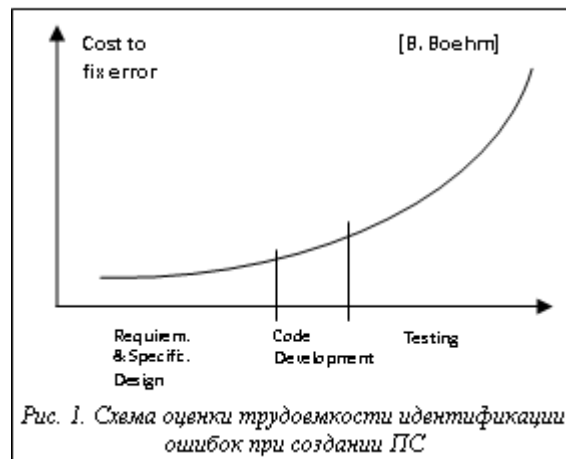
- Тестирование запланировано позже разработки

## Недостаточно времени / бюджета

- Короткий период тестирования
- Ручное тестирование исключается



# Эффективность автоматизации



Оценка распределения трудоемкости между фазами создания программного продукта: 40%-20%-40%

# Эффективность автоматизации



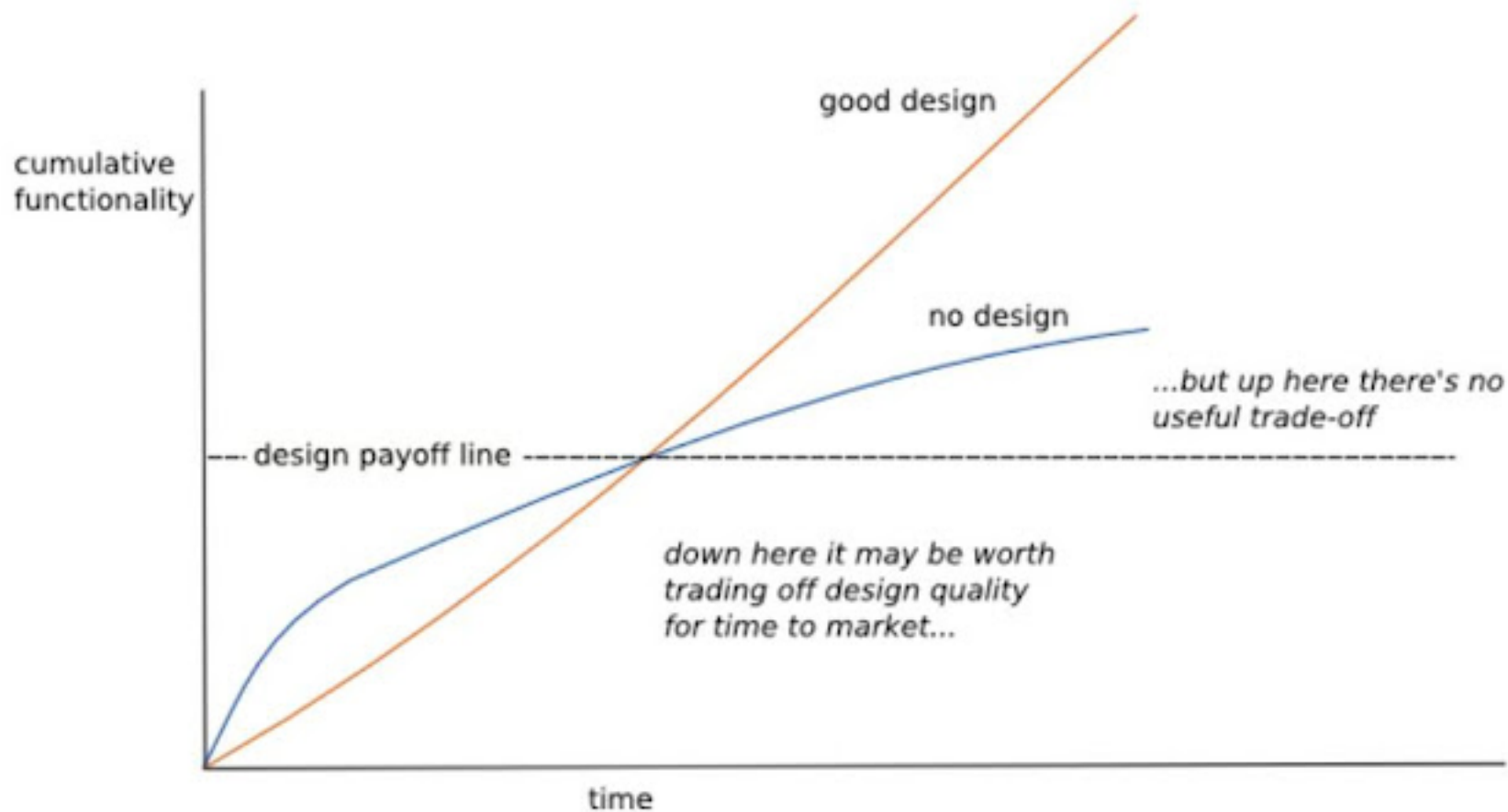
- Следовательно, наибольший эффект в снижении трудоемкости может быть получен прежде всего на фазах Design и Testing
- А значит и основные вложения в автоматизацию или генерацию кода следует осуществлять, прежде всего, на этих фазах

# Unit Testing

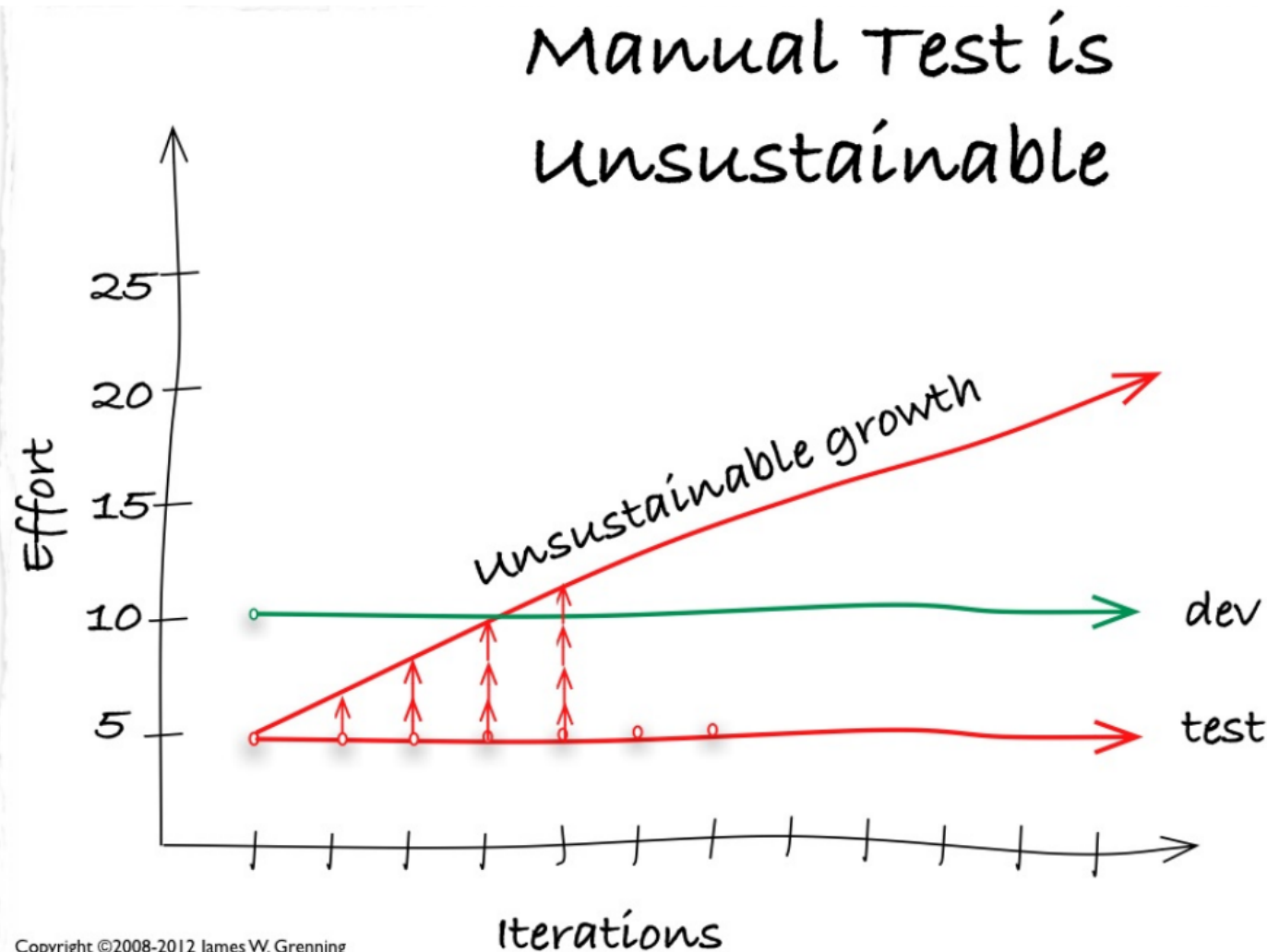
- Облегчает работу программиста, позволяя маленькими шажками реализовывать большой функционал. И гарантирует работоспособность на каждом шаге
- Вынуждает проектировать красивый дизайн проекта, который в дальнейшем легче использовать
- Есть гарантия, что покрытый тестами код - работает!
- Рефакторинг становится почти безболезненным.

TDD – это о дизайне!

# Влияние хорошего дизайна

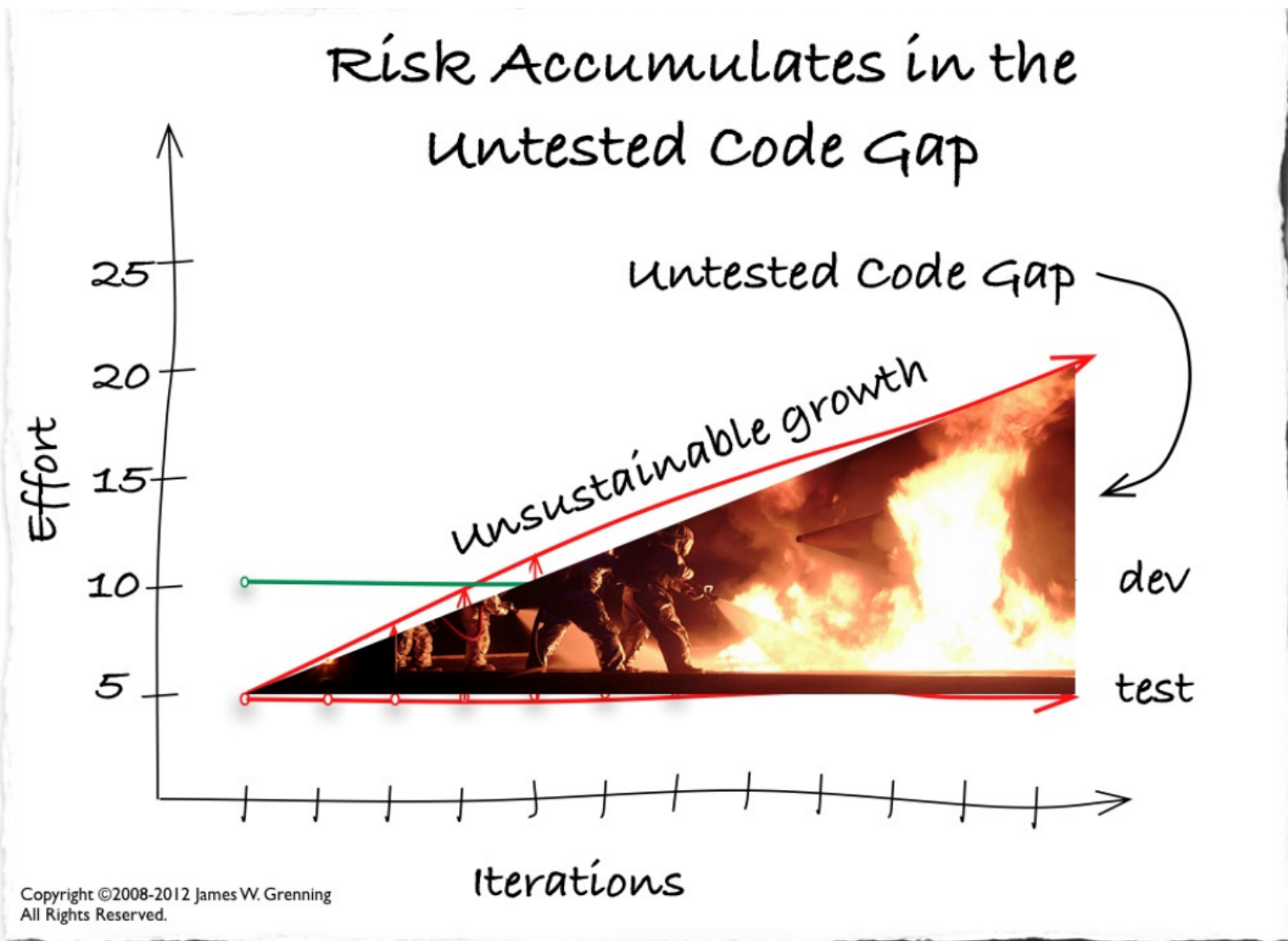


# Почему автоматизированные тесты?



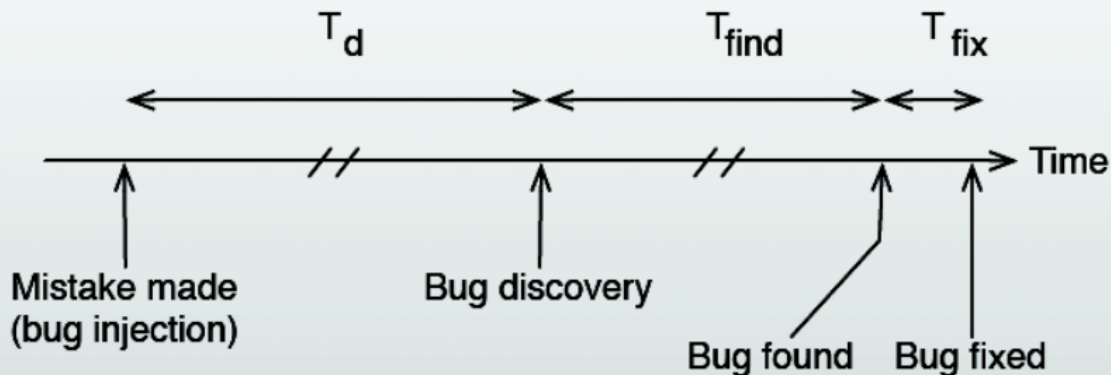
Copyright ©2008-2012 James W. Grenning  
All Rights Reserved.

# Риски ручного тестирования



# Почему Test First?

## The Physics of Debug Later Programming (DLP)

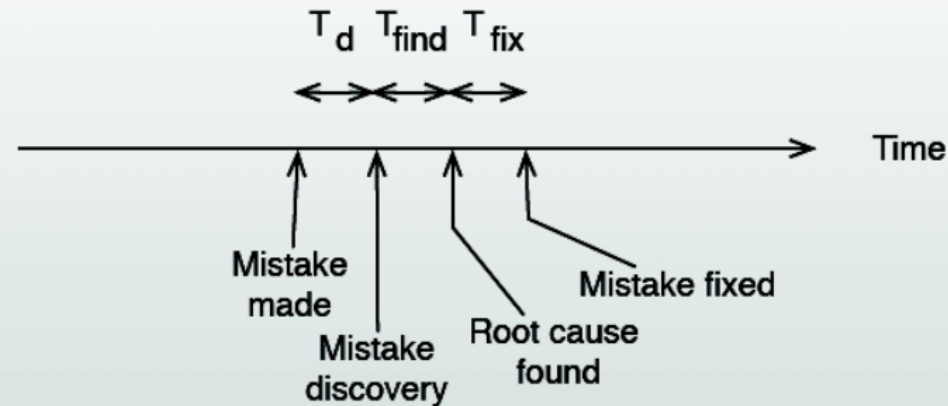



- As  $T_d$  increases,  $T_{find}$  increases dramatically
- $T_{fix}$  is usually short, but can increase with  $T_d$



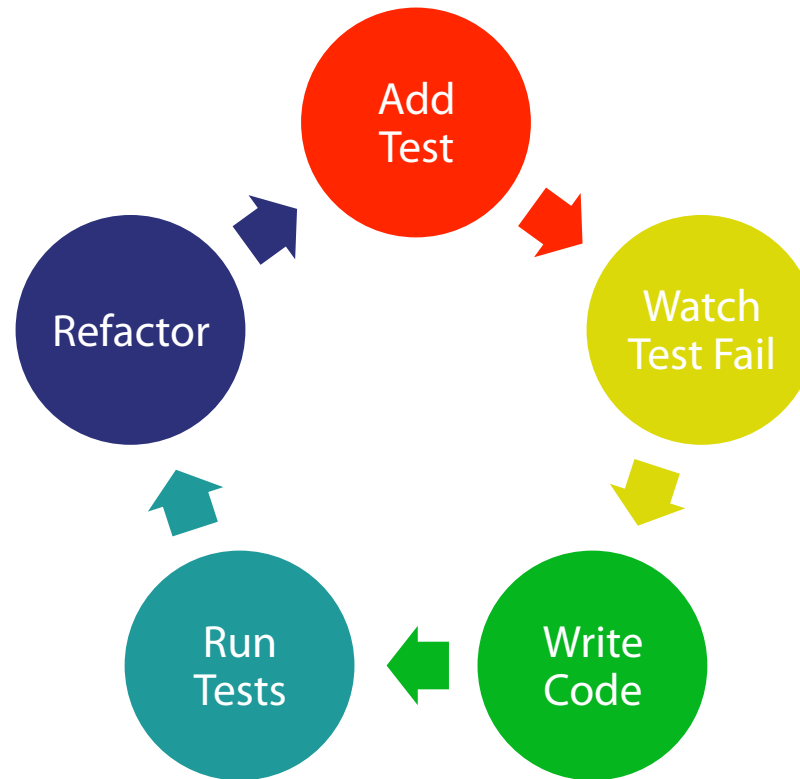
# Почему Test First?

## The Physics of Test Driven Development



- When  $T_d$  approaches zero,  $T_{find}$  approaches zero
- In many cases, bugs are not around long enough to be considered bugs. 
- See: <http://www.renaissancesoftware.net/blog/archives/16>

# Итеративный процесс



# Почему TDD?

- Testable code is cleaner
  - Lower Complexity
  - Loosely Coupled
  - Tighter Cohesion
- Keeps focus on Simple Design
  - YAGNI
  - DRY
- Continuous Refactoring



**Вопросы ?**



# Разработка через тестирование

[IDyachenko@luxoft.com](mailto:IDyachenko@luxoft.com)

```
git clone git://github.com/ivan-dyachenko/Trainings.git
```

```
https://github.com/ivan-dyachenko/Trainings
```