Test-Driven Development Seminar
测试驱动开发研讨会

Author: Guan ZhiShi
Email: jayden.guan@gmail.com

Version 0.95
Last Update: Aug.08.2009
ShangHai

我们将了解的内容

- 了解TDD的基本内容。
- 可以判断TDD是否适用于特定项目。

Summary

内　　容

- 前言
- 开发现状
- TDD能带来什么
- TDD的实施
- 实际案例
- TDD适用范围
- 常见Q&A
- 参考资料

Module 1:
前言

目标参加人员

- 如果具备一些经验，可以使讨论达到更好的效果
  - 有实际项目开发经验
  - 对XUNIT自动测试框架有所了解更佳

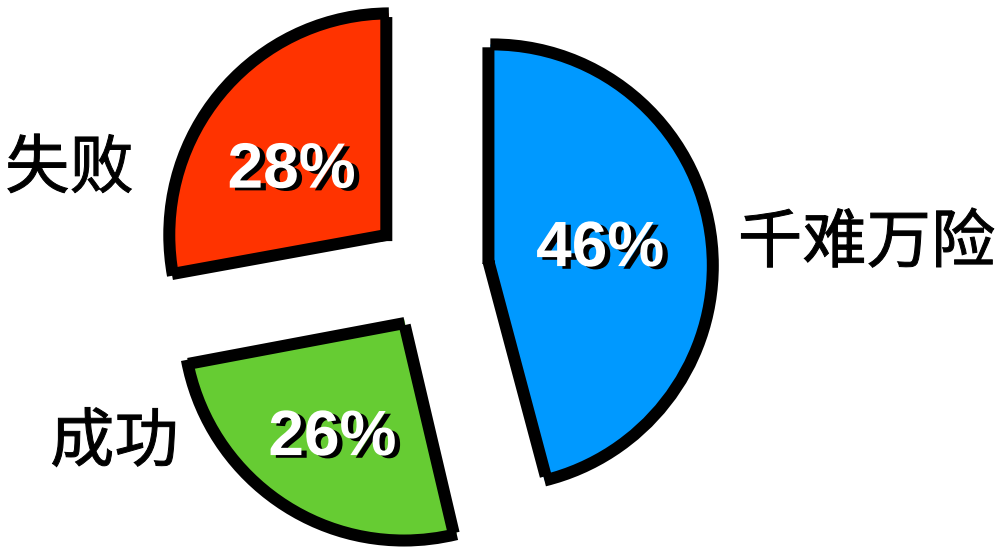
工具箱

- 一个优秀的工匠知道选择合适的工具。

- 适当的时候使用适当的工具
- TDD可能是你工具箱中的新工具

Module 2:
开发现状

项目结果统计



应用开发项目

人狼

- 人狼完全可以从熟悉的面孔变成可怕的怪物

  软件项目有着人狼的特性，常常看似简单明了的东西，却可能变成一个落后进度，超出预算，存在大量缺陷的怪物。

  P.Brooks

  《人月神话》

  1974

似曾相识之一

- 甲： 你那个“业务逻辑控制”模块完成了怎么样?
- 乙： 代码已经完成了，但还没测试，我现在很忙，等我把剩下的模块全部完成了再来测试。那当是完成了 90%。
- 甲心想：“但愿如此……”

人狼之一

- 工期紧张，忽略做测试。
- 没有测试的进度估算，失真的可能性很大。

似曾相识之二

- 甲： 我需要使用你的那个“业务逻辑控制”的模块。你的模块测试了吗？
- 乙：测试过了。

- 甲：都测试了哪些情况？
- 乙：除了没测试的，都测试过了。你先用着，出了问题我再改。

- 甲心想： 别又像上次那样，没测就扔给我。

人狼之二

- 对于程序的功能，往往缺乏合适的描述方法。
- 不良的沟通，可能导致彼此不信任。

似曾相识之三

- 乙： 那个“业务逻辑控制”的模块，效率太低，写得又乱以后很难维护，我想整理一下。
- 甲： 现在起码还能用，快交货了，没时间做完整测试了，别又搞出点BUG来。
- 乙心想：但愿以后不是我来维护。

人狼之三

- 没有完整测试保护的代码修改，有着很高风险。
- 混乱的代码，将增加维护的难度。

似曾相识之四

- 甲：这个项目现在移交给你了。你很幸运，这个项目文档很全。
- 乙：那太好了，上次我接到个项目没有文档。
- 甲：但不幸的是文档和当前程序不太一致。
- 乙：我怎么知道哪些地方不一致呢？
- 甲：这是个好问题...

人狼之四

- 文档往往和现实情况不同步。
- 文档差异很难察觉。

似曾相识之五

- 甲：都连续加班一个月了，昨晚做梦都在改BUG，一进办公室就胃疼。
- 乙：我胃到是不疼，就是最近脾气不好，老发火。

人狼之五

- 长期被BUG所困扰。
- 士气低落，身心俱疲。

Module 3:
TDD能带来什么?

什么是TDD

- 只需要用自动运行的测试来推动开发，这种开发方式被称为测试驱动开发(Test-Driven Development，TDD)

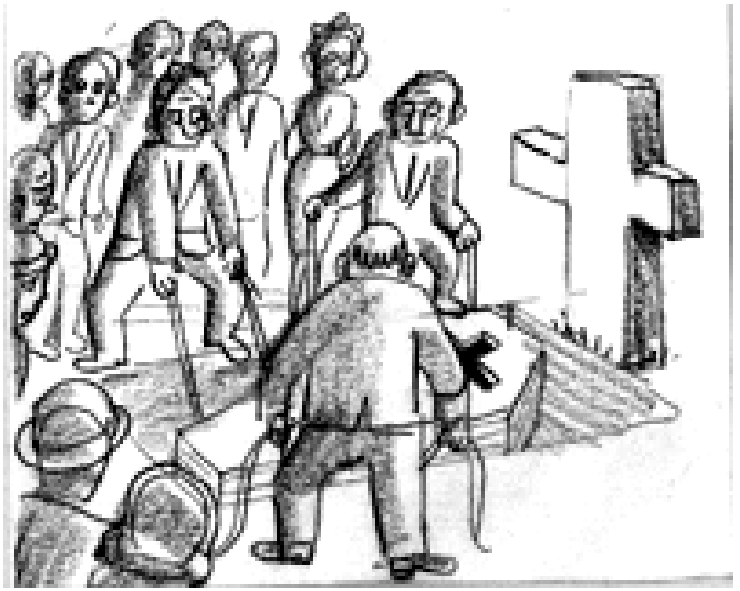
TDD的目标

- 代码整洁可用(clean code that works)是TDD所追求的目标。

压力绞架

压力越大

测试越少



错误越多

解开死锁

- 压力越大 ->测试越多->错误越少->成果越多->压力越少

TDD能带来的好处

- 它是个可预测的开发方法。
- 它给你一个全面正确地认识和利用代码的机会。
- 它让软件开发小组成员之间相互信赖。
- 这样的代码写起来感觉很好。

TDD对人的影响

- TDD是一种可以在开发过程中控制忧虑提供勇气的开发方式。

忧虑	勇气
让你一直处于试验性的阶段。	尽快开始具体的学习。
让你不愿意与他人交流。	更多的参与交流和沟通。
让你不愿意面对反馈。	寻找那些有益的、建设性的反馈。

Module 4:
TDD的实施

TDD该怎么做

- 只有自动测试失败时，我们才重写代码。
- 消除重复设计，优化设计结构。

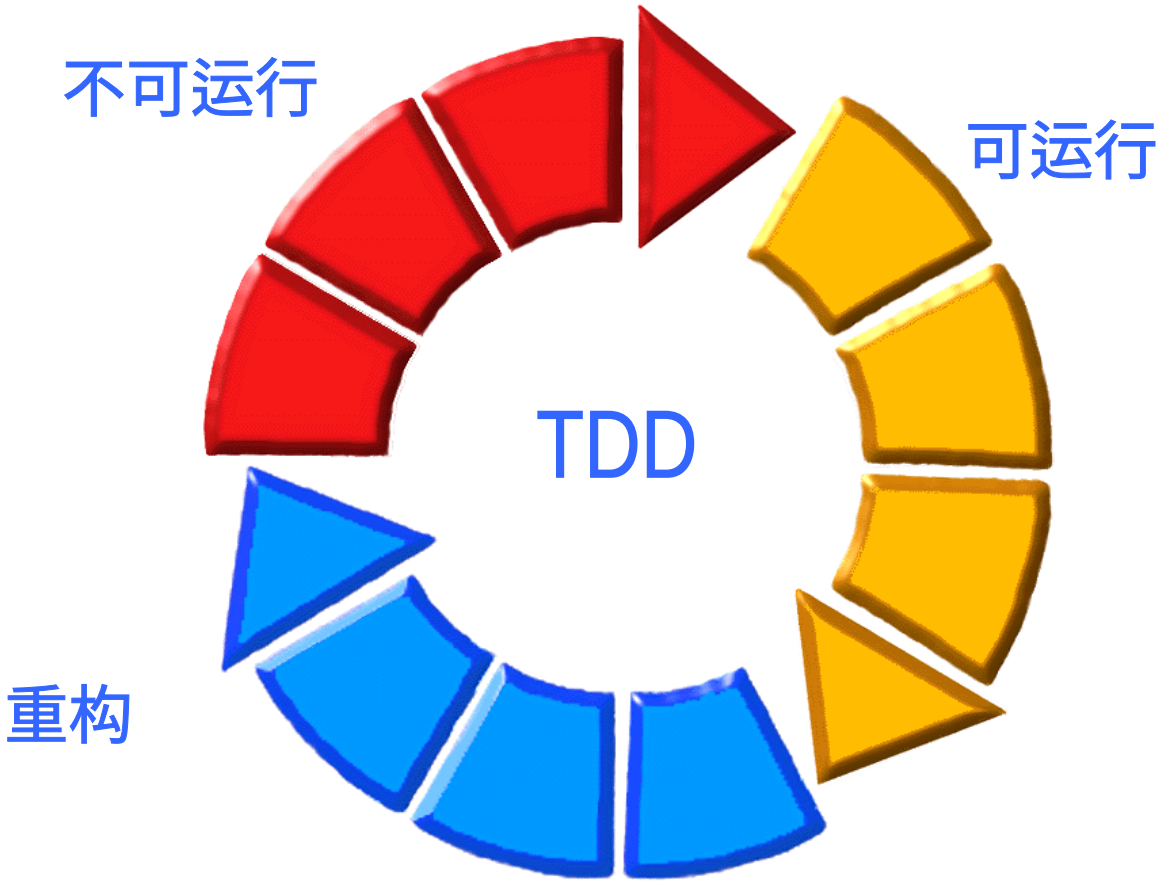
技术含义

- 我们必须通过运行代码所提供的反馈来做决定，并以此达到有机设计目的。
- 我们必须自己写测试程序，因为测试很多很频繁，我们不能每天等待别人写测试程序。
- 我们的开发环境必须能够响应哪怕是很小的变化。
- 为使测试简单，我们的整个设计必须是由很多高聚能、低耦合的部分组成。

写程序的时间里都干了些什么

- 用来决定下一步做什么。
- 花在设计上。
- 编写代码。
- DEBUG 40%-50%的时间

TDD 的口号



TDD都干了些什么

- 不可运行：写一个不能工作的测试程序，一开始甚至不能编译。
- 可运行：尽快让这个程序工作，为此可以在程序中，使用一些不合情理的方法。
- 重构： 消除在让测试程序工作的过程中，产生的重复设计，优化设计结构。

什么时候写测试

- 如果某个方法的接口还不清楚，那么在编写该方法前要编写测试。
- 如果接口很清楚，但是你认为实现会稍微有一点复杂，那么在编写该方法前要编写测试。
- 如果你想到了代码运行的一种异常情况，那么应该编写一个测试以针对这种情况进行沟通。
- 如果你打算重构某些代码，但是不确定它应该有什么行为，那么应该先编写一个测试。

运行自动测试

- 单元测试必须**100%**运行。
- 如果单元测试中有一个出错了，那么对于团队中的所有人来说，**最重要**的工作就是修复该测试。

火警测试

- 文档
- 源程序
- 发布的产品包

- 只能带一件东西，该拿哪个跑呢?

还需要重构

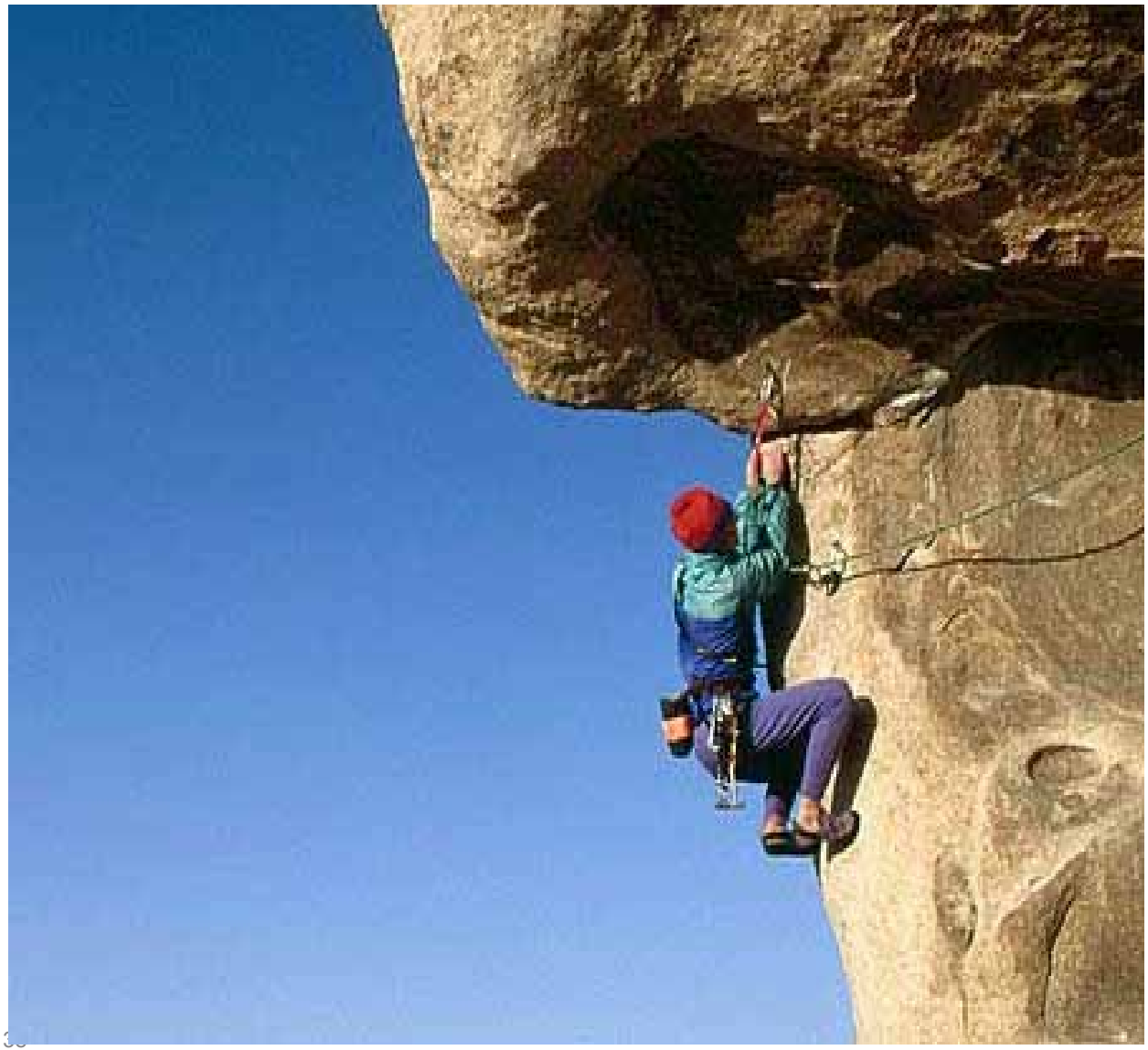
- 无重复代码。
  重复代码是万恶之源。
- 每个方法不超过15行。
  长方法是Bug的温床。
- 取个好名字。
  朗读你的代码。

源代码是写给人看的，它最重要的文档。

节奏是否过慢?

- 分而治之，先可用后整洁。
- 小步快进的工作节奏。打破开发压力与测试的死结。
- 节奏可以调节。顺利时加大步伐，困难时减小步伐。

专业登山者 vs. TDD程序员



- 艰巨挑战
- 安全措施
- 小步前进

需要100%测试覆盖率吗？

- 测试应该是一种风险驱动(risk driven)行为，测试的目的是希望找出现在或未来可能出现的错误。
- 测试的要诀是:测试你最担心出错的地方。这样你就能从测试工作中取得最大的利益。
- 编写未完善的测试并实际运行，好过对完美测试的无尽等待。

短期收益与长期收益

- 短期：将开发过程中的设计、实现、测试所做过的工作都固化下来。
- 长期：整个软件项目生命周期，都能通过运行自动测试，取得反馈来获得收益。

Module 5:
实际案例

XXXSale1.2

- 既有项目改造，基于EJB、WEB、DB的项目。客户业务逻辑发生较大改变和扩充。采用TDD方式。
- Defect发生率 2.4/KLOC，开发效率 68 行/人日。
- 开发团队对TDD方式，反映良好。

XXX University System

- 8人的团队负责交付一个大型系统的子系统业务逻辑和数据库操作部分。系统架构与业务逻辑经常发生较大变动。新人与有经验开发人员各占一半。
- 开发效率 50 行/人日，Defect发生率 低于1/KLOC。
- 开发团队对TDD方式，反映良好。

Module 6:
TDD的适用范围

适用的技术类型

- 程序能够快速运行。

(有些程序编译一次可能需要几个小时，使自动测试无法频繁运行)

- 程序对其他环境的依赖性小。

(例如EJB程序对于容器强烈依赖，会增加自动测试难度)

开发环境的支持

- 自动测试框架的支持。
- 自动重构工具的支持。

价值观的支持

- 不担心阅读与编写代码。
- 不担心编写测试。
- 不担心更改运行中的系统的分析和设计。
- 能够接受纪律。

Module 7:
常见Q&A

常见问题1

- 1. Q: private 方法怎么测试?
- A: public 方法测到就行了。

2. Q: TDD 算是UT,还是算 IT? (我发现QA对于这个区分比较在意)

- A: 能驱动开发就行，只要这个test能让你对代码放心，不用管它UT还是IT。

- 3. Q:测试代码和工作代码比例是多少，覆盖率要达到多少？
- A:让你对代码觉得放心就行，这些不用管它。

常见问题2

- 4. Q:是否高水平的开发人员才能实行TDD方法？
- A: 从实践效果来看，从新手到高手都适合TDD。入门的方法是找个已经熟悉TDD的人和你结对编程。

5. Q:习惯了TDD后，不用TDD时的开发能力会不会降低？

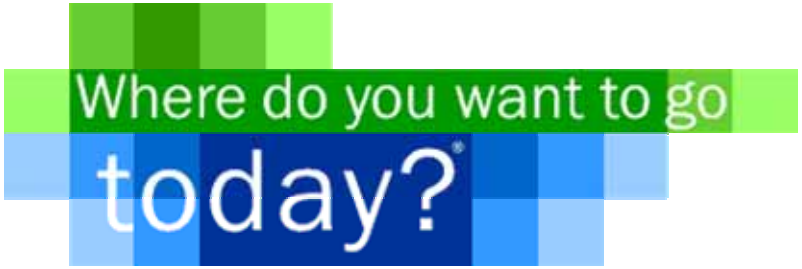
- A: -_-!，只能说TDD使人的开发能力提高了。

- 6. Q:项目时间紧任务重，能实施TDD?
- A:就没见过时间不紧，任务不重的实际项目。
- TDD就是为了省时省力。

Module 8:
参考资料

进一步了解

- 《测试驱动开发》 Kent Beck
- 《拥抱变化 XP解析》Kent Beck
- 《重构》 Martin Fowler



Just do it

- Email: jayden.guan@gmail.com