Atitit各种驱动的xdd tdd bdd设计 ATDD ddd v3 u66.docx

Atitit各种驱动的xdd tdd bdd设计 ATDD ddd v2 s66

开发方法论与效率提升

Atitit各种驱动的xdd tdd bdd设计 ATDD ddd

Tdd bdd 行为驱动

目录

[1. 2. 软件开发过程中最常见的两个问题 1](#_Toc30954)

[1.1. 需求和开发脱节： 1](#_Toc16398)

[1.2. 开发和测试脱节： 2](#_Toc12955)

[2. 高效率的开发范式 开发方法论 2](#_Toc7348)

[2.1. Xp 2](#_Toc3098)

[2.2. 完美的组合是TDD，DDD和BDD 2](#_Toc7176)

[2.3.  FDD (Feature-driven Development)  2](#_Toc23630)

[2.4. BDD 指的是Behavior Drive Development 2](#_Toc27642)

[3. Xdd "X" Driven-Development Methodologies 2](#_Toc10602)

[3.1.  DDD (Defect-Driven Development) – 3](#_Toc20976)

[3.2. RDD (Responsibility-Driven Design) – 3](#_Toc848)

[3.3. UGDD (User Guide-Driven Development) 3](#_Toc17928)

[3.4.  MDD (Model-driven Development)  3](#_Toc12868)

[3.5. DDD (Documentation-Driven Development) 3](#_Toc26005)

[3.6. Edd 事件驱动 3](#_Toc25602)

[3.7. TFD -- Test First Development) 3](#_Toc24139)

[3.8. 2. BDD 指的是Behavior Drive Development，也就是行为驱动开发 3](#_Toc4785)

[3.9.  DDD DDD指的是Domain Drive Design，也就是领域驱动开发 3](#_Toc16368)

[3.10. CBD（核心Core+行为Behavior+驱动Driver）架构模式 4](#_Toc31821)

[3.11. TDD（测试驱动开发(Test-Driven Development)） 4](#_Toc20634)

[3.12. ATDD：验收测试驱动开发（Acceptance Test Driven Development） 4](#_Toc30330)

[4. 区别 5](#_Toc8117)

[4.1. 简而言之，完美的组合是TDD，DDD和BDD 5](#_Toc2531)

[5. ref 7](#_Toc13931)

# 2. 软件开发过程中最常见的两个问题

## 需求和开发脱节：

* 用户想要的功能没有开发
* 开发的功能并非用户想要
* 用户和开发人员所说语言不同

## 开发和测试脱节：

* 开发和测试被认为割裂
* 从开发到测试周期过长
* 测试自动化程度低

3. 如何解决上面说的两个问题

使用BDD可以解决需求和开发脱节的问题，首先他们都是从用户的需求出发，保证程序实现效果与用户需求一致。

# 高效率的开发范式 开发方法论

## Xp

## 完美的组合是TDD，DDD和BDD

## FDD (Feature-driven Development)

## BDD 指的是Behavior Drive Development

# Xdd "X" Driven-Development Methodologies

## DDD (Defect-Driven Development) –

## RDD (Responsibility-Driven Design) –

## UGDD (User Guide-Driven Development)

## MDD (Model-driven Development)

## DDD (Documentation-Driven Development)

## Edd 事件驱动

## TFD -- Test First Development)

## 2. BDD 指的是Behavior Drive Development，也就是行为驱动开发

BDD指的是Behavior Drive Development，也就是行为驱动开发。这里的B并非指的是Business，实际上BDD可以看作是对TDD的一种补充，当然你也可以把它看作TDD的一个分支。因为在TDD中，我们并不能完全保证根据设计所编写的测试就是用户所期望的功能。BDD将这一部分简单和自然化，用自然语言来描述，让开发、测试、BA以及客户都能在这个基础上达成一致。因为测试优先的概念并不是每个人都能接受的，可能有人觉得系统太复杂而难以测试，有人认为不存在的东西无法测试。所以，我们在这里试图转换一种观念，那便是考虑它的行为，也就是说它应该如何运行，然后抽象出能达成共识的规范。如果你用过JBehave之类的BDD框架，你将会更好的理解其中具体的流程。这里我推荐一

## DDD DDD指的是Domain Drive Design，也就是领域驱动开发

DDD指的是Domain Drive Design，也就是领域驱动开发。这是一种非常好的思想，在我们刚开始学习程序，甚至刚开始学习三层架构的时候，我们曾经面临过很多疑惑，比如如何来实现我们的数据层？后来我们开始学习MVC，MVP等架构，如何设计Model层又成了我们的新问题。我们见过太多这种情况，Model变成了单纯的数据容器，也就是我们经常说的贫血模式。DDD实际上也是建立在这个基础之上，因为它关注的是Service层的设计，着重于业务的实现，因此不可避免的以贫血模式为基础而存在。但是它最大的特别是将分析和设计结合起来，不再使他们处于分裂的状态，这对于我们正确完整的实现客户的需求，以及建立一个具有业务伸缩性的模型，是有很大帮助的。

## CBD（核心Core+行为Behavior+驱动Driver）架构模式

## TDD（测试驱动开发(Test-Driven Development)）

TDD的好处自然不用多说，它能让你减少程序逻辑方面的错误，尽可能的减少项目中的bug，开始接触编程的时候我们大都有过这样的体验，可能你觉得完成得很完美，自我感觉良好，但是实际测试或者应用的时候才发现里面可能存在一堆bug，或者存在设计问题，或者更严重的逻辑问题，而TDD正好可以帮助我们尽量减少类似事件的发生。而且现在大行其道的一些模式对TDD的支持都非常不错，比如MVC和MVP等。

但是并不是所有的项目都适合TDD这种模式的，我觉得必须具备以下几个条件。

首先，项目的需求必须足够清晰，而且程序员对整个需求有足够的了解，如果这个条件不满足，那么执行的过程中难免失控。当然，要达到这个目标也是需要做一定功课的，这要求我们前期的需求分析以及HLD和LLD都要做得足够的细致和完善。

其次，取决于项目的复杂度和依赖性，对于一个业务模型及其复杂、内部模块之间的相互依赖性非常强的项目

## ATDD：验收测试驱动开发（Acceptance Test Driven Development）

通过单元测试用例来驱动功能代码的实现，团队需要定义出期望的质量标准和验收细则，以明确而且达成共识的验收测试计划（包含一系列测试场景）来驱动开发人员的TDD实践和测试人员的测试脚本开发。面向开发人员，强调如何实现系统以及如何检验

# 区别

## 简而言之，完美的组合是**TDD**，**DDD**和BDD



# ref

TDD（测试驱动开发(Test-Driven Development)）\_百度百科.html

TDD、BDD、ATDD、DDD 软件开发模式 - CSDN博客.html

大话TDD，BDD，ATDD的本质 - CSDN博客.html

关于TDD、BDD和DDD的一些看法 - ustbwuyi - 博客园.html