从企业的运行价值链说起我眼中的测试驱动开发(TDD) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/141/2021_2022__E4_BB_8E_E4_BC_81_E4_B8_9A_E7_c29_141540.htm 看了一期中央电视台的《对话》栏目，节目对三星CEO尹钟龙进行了访谈。其中，现场一位女士的一句话令我印象深刻。她提到一个企业的运行价值链，共分为三步：首先是发现价值，找到目标市场；第二步是生产价值，将高质量的产品生产出来；最后是保护价值或收获价值，做好品牌。 怎么理解呢？这位女士以Nike作比喻。第一步是设计Nike鞋，这就是发现价值，可能获得100美元的价值；然后再拿到中国来生产，大约是10美元；最后再将生产好的鞋子，贴上Nike的商标送回到美国去卖，又可以收获90美元。一双鞋售价200美元，而生产价值所能收获的却只有10美元。这一步获取利益最低，我们中国的公司却做得最好。而怎么去发现价值，然后又怎样去巩固自己的品牌和知名度，中国的公司就做得不那么好了。 据我的了解，国内的软件开发应用TDD相对较少，很多人认为：测试驱动开发是个好东东，但似乎不符合中国国情。说到原因，最多的一条就是项目时间紧，没有时间写测试代码。在项目中，到底该不该使用TDD，大多数人持怀疑或观望的态度。这种态度与观点，就让我想起了如上《对话》中的这一段话。 再仔细分析企业运行价值链的三步走，我觉得和软件开发的TDD价值链很相似。第一步，是发现价值。应用到TDD中，就是测试先行，通过测试来驱动我们编写代码。第二步，生产价值。毋庸置疑，这正是编写代码的一个阶段。而第三步，就是收获价值，在TDD中，我们收获的不仅有开发后

完整的产品，同时还收获了完整的测试套件。和Nike鞋的生产一样，我们在软件开发中，过度地重视了第二步生产价值阶段，而对于第一步和第三步，要么是忽略了，要么就是没有提高到相应的高度。 一、发现价值与生产价值 习惯了传统开发模式的程序员，非常不适应写代码之前，先写测试的方法，这其中也包括我。那么，我们一般是怎样去发现价值的呢？首先通过需求分析，然后进入设计阶段。在设计阶段期间，再围绕需求分析的结果，更多的是从实现的角度，而非从客户应用的角度出发。TDD颠覆了这种模式。因为需要测试先行，就驱动了程序员必须从功能出发、从应用出发。在写测试代码的过程中，我们需要考虑要实现那些功能，相应的类的名称、对象的创建方式，以及可能会应用到的模式和策略，如此种种，在这个过程中，如剥笋子一般逐渐地规定出来了。 在这个过程中，我们要审慎地选择测试的步子。昂首阔步虽然显得气势轩昂，行进快速，但往往会忽略沿途的风景。在测试驱动开发中，我建议你小心的规划测试样例，从测试样例的逐步完善中，渐进地驱动出你更加完善的代码。例如，我需要开发一个智能的个人助理，它目前能提供的功能是：能够让用户定制自己感兴趣的类别，然后个人助理根据用户的定制进行搜索，并将搜索得到的结果按不同的类别进行存储。 我们来尝试一下TDD的过程。根据对功能的分析，我们首先应该有一个智能助理对象，测试代码如下：[Test] public void TestSmartAssistor() { SmartAssistor assistor = new SmartAssistor(). Assert.IsNotNull(assistor). } 当然，这段代码是连编译都无法通过的，我们还需要创建SmartAssistor类型。然而，不要小瞧了这一步，它实际上促使你对项目进行初步

的理解，至少，你需要想好这个将要创建的类型，它的名字是什么？这就是一种驱动力。（为了简便起见，在本文只列出测试代码） 然后，这个类型能够做些什么呢？我们把个人智能助理的功能进行分类，应该包括三个功能：1、定制；2、搜索；3、存储。 仔细想想，实际上只有搜索和存储才是智能助理的职责所在，而定制不过是智能助理要运转的一个条件罢了。既然如此，从客户应用的顺序来考虑，我们应该先实现定制的功能。要定制类别，就应该具备类别类型，而定制类别这项功能，则应该由一个专门的控制器来承担责任。 [SetupUp] public void InitObject() { Category cg1. Category cg2. CategoryContainer cgContainer. SmartController control. } [Test] public void TestCategory() { cg1 = new Category(” SoftWare Engineering ” , ” TDD ”). cg2 = new Category(” SoftWare Engineering ” , ” Design Pattern ”). cgContainer = new CategoryContainer(). cgContainer.Add(cg1). cgContainer.Add(cg2). Assert.IsNotNull(cgContainer). Assert.AreEqual(cg1,cgContainer[0]). Assert.AreEqual(cg2,cgContainer[1]). } [Test] public void TestController() { control = new SmartController(). Assert.IsNotNull(control). Assert.IsTrue(control.CustomizeCategories(cgContainer)). }