

淘宝（中国）软件有限公司

接口测试白皮书

V0.1



淘宝网平台测试组 (qa.taobao.com)

目录

1	接口测试的背景	3
1.1	什么是接口测试.....	3
1.2	为什么做接口测试.....	3
1.3	接口测试的适用范围.....	4
2	接口测试的目的	5
2.1	战略方针.....	5
2.2	发展各阶段和目标.....	6
3	接口测试的定位	7
3.1	人员能力定位.....	7
3.2	职责定义.....	7
3.3	工作内容定位.....	7
4	接口测试的流程	9
4.1	项目工作流程.....	9
4.2	日常工作流程.....	9
4.3	流程步骤详解.....	10
4.3.1	需求分析和设计评审.....	10
4.3.2	测试框架和技术选型.....	10
4.3.3	测试计划制定.....	10
4.3.4	测试环境搭建.....	10
4.3.5	测试用例设计和评审.....	11
4.3.6	测试实现和执行.....	11
4.3.7	持续集成.....	11
4.4	质量评估标准.....	11
5	接口测试的技术简介	13
5.1	JUnit.....	13
5.2	DbUnit.....	13
5.3	Spring TestContext Framework.....	13
5.4	Unitils.....	14
5.5	TestNG.....	15
5.6	CruiseControl.....	15
5.7	Clover.....	16
5.8	Mock.....	17
6	接口测试的方向	18
7	参考资料	20
8	作者介绍	21

1 接口测试的背景

1.1 什么是接口测试

接口测试是测试系统组件间接口的一种测试。接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。测试的重点是要检查数据的交换，传递和控制管理过程，以及系统间的相互逻辑依赖关系等。

1.2 为什么做接口测试

在淘宝网系统的历史上，首先出现的是功能测试和性能测试，然后是自动化测试，但发展到今天，淘宝网的架构已经不再是传统的 MVC 结构，系统不断向着分布式、业务中心化和高可用性的方向发展，如今的系统架构纷繁复杂，系统间的接口庞杂繁多，传统的功能测试、性能测试和自动化测试已经难以满足系统发展的需求，迫切需要一种更加有效实用且可以持续进行的测试方式来保证系统的质量。

接口测试在这种需求下应运而生。

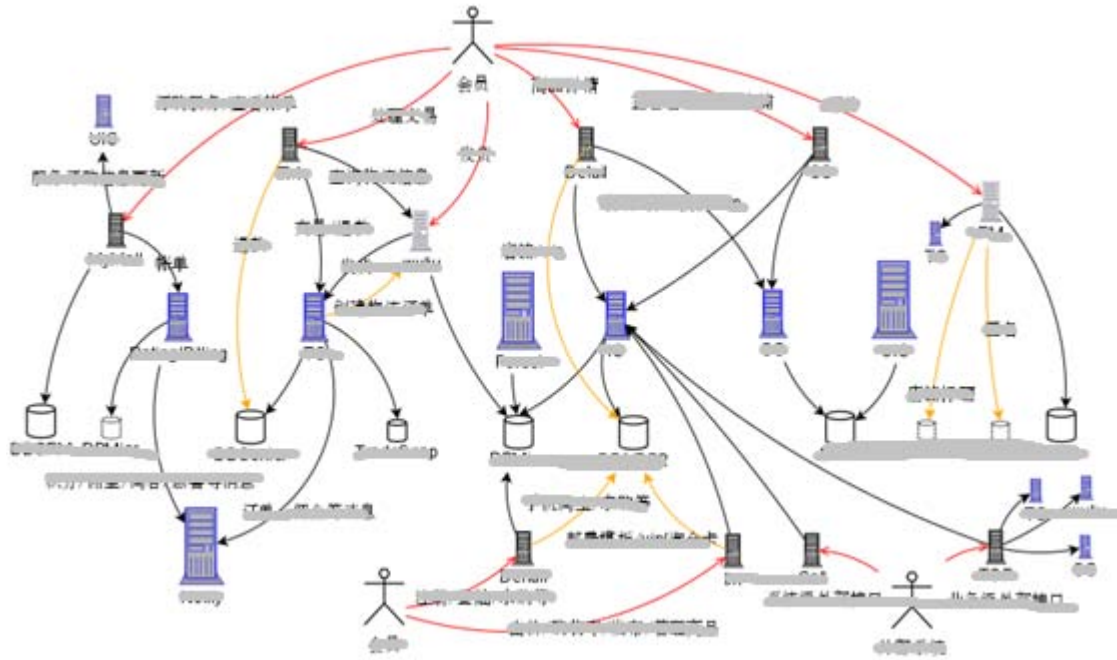
首先，随着系统复杂程度的上升，传统的测试方法测试成本急剧增加，测试效率大幅下降（数据模型推算，底层的一个bug能够引发上层的 8 个左右bug，而且底层的bug很容易引起全网的宕机，详见淘宝测试博文：<http://rdc.taobao.com/blog/qa/?p=2070>）。相反接口测试能够提供系统复杂度上升情况下的低成本高效率的解决方案。

其次接口测试不同于传统开发的单元测试，接口测试是站在用户的角度对系统接口进行全面高效持续的检测。

最后接口测试是自动化并且持续集成的，这也是为什么接口测试能够低成本高收益的根源。

总之接口测试是保证高复杂性系统质量的内在要求和低成本的经济利益的驱动作用下的最佳解决方案，接口测试是一个完整的体系，也包括功能测试、性能测试。

下图充分说明了系统间接口的复杂程度



1.3 接口测试的适用范围

接口测试一般应用于多系统间交互开发，或者拥有多个子系统的系统开发的测试。接口测试适用于为其他系统提供服务的底层框架系统和中心服务系统，主要测试这些系统对外部提供的接口，验证其正确性和稳定性。接口测试同样适用于一个上层系统中的服务层接口，越往上层，其测试的难度越大。接口测试在淘宝的应用是一个自下而上的发展过程。

接口测试实施在多系统多平台的构架下，有着极为高效的成本收益比。接口测试天生为高复杂性的平台带来高效的缺陷检测和质量监督能力。平台越复杂，系统越庞大，接口测试的效果越明显。

2 接口测试的目的

2.1 战略方针

接口测试的核心战略在于：以保证系统的正确和稳定为核心，以持续集成为手段，提高测试效率，提升用户体验，降低产品研发成本。

◆ 核心：保证系统的稳定

质量管理的目标是保证系统的正确和稳定，接口测试作为软件质量管理的一部分也是保证系统的正确和稳定的，更准确的是保证系统服务端正确和稳定，一个系统的服务端，越接近底层，对系统的影响就越大，甚至有可能牵一发而动全身，服务端的一个缺陷可能会引起客户端的几个甚至十几个缺陷，更可怕的是服务端的缺陷有可能引起系统的崩溃，这对整个系统来说，损失将是不可估量的，因此服务端接口的质量将直接影响到系统的正确和稳定。

◆ 手段：持续集成

什么是持续集成为手段，关键在于“持续构建”、“业务”、“集成化”以及“文档体系”，我们需要让被测代码进行持续构建集成，我们需要用业务化的思维去考虑接口定义的合理性，我们需要从性能、安全的角度去思考代码的正确性，我们还需要从集成化的角度去甄别接口间数据传递的正确性，我们更需要确定我们的测试范围，也就是我们测什么、不测什么。

◆ 目的：提高测试效率，提升用户体验，降低产品研发成本

接口测试要为代码的编写保驾护航，增强开发人员和测试人员的自信，让隐含的 BUG 提前暴露出来，要让开发人员在第一时间修复 BUG，要让功能测试人员和性能测试人员在测试的时候更加顺手，最大限度得减少底层 BUG 的出现数量，要让产品研发的流程更加敏捷，要缩短产品的研发周期，最后在产品上线以后，要让用户用得更加顺畅，要让用户感觉产品服务零缺陷。

另外在这个战略过程中，我们需要几类资源作为支撑，下面做简单描述。

首先在这个战略中最重要的一点是要强调团队的重要性，特别是团队中需要有合理的人力资源配置，在这个团队中，需要全才，也需要专才，需要技术专家，也需要业务专家，既需要高效的执行者，也需要有效的管理者，任何人在这个团队中都可以发挥重要作用。

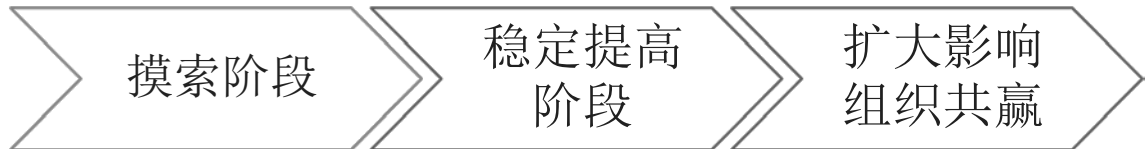
其次我们需要强大的测试技术以及测试框架去支撑我们的日常工作，包括基于 JAVA 以及基于 C++ 的测试框架，甚至以后会扩展到其他各个语种的框架，计算机软件的架构发展到今天，特别是分布式软件的发展，导致软件体系结构日益复杂化，各个系统之间的依赖逐渐加强，JAVA、C++ 以及多种技术的综合使用，使传统的单元测试已经无法满足于针对接口编程的架构方式，我们需要以一种更加干净的层面也就是从业务的层面对接口进行隔离测试，同时为了模拟真实场景，也需要在真实的环境中系统内根据业务流程对各个接口进行串联测试，最后以持续集成系统保证被测代码的稳定性。

再次要充分重视文档的重要性，包括需求文档，开发技术方案，测试技术方案，接口定义 JAVADOC，测试用例文档等等，完善这些文档可以大大减少软件工程周期中各个团队配合障碍，也可以降低后期软件维护成本。

因此贯彻和落实接口测试的战略可以最大程度地提高软件质量的稳定性。

2.2 发展各阶段和目标

本节将简要讲述一个接口测试团队从建立初期到发展起来经历了哪些阶段，以及我们期望将来做成什么样子。



◆ 摸索阶段

一个全新的团队成立之初一般都会经历一个比较长期的摸索阶段，在这个阶段我们会尝试不同的技术、框架和流程规范。直到在这些方面都找到了比较适合团队自身特点的方案，那么这个阶段的目标就算是达到了。

◆ 稳定提高阶段

摸索阶段过后就应该会进入一个稳定提高期。经历了摸索阶段过后，团队的技术、框架和流程规范都应该有了一个基本的定型。这个时候团队的目标就是通过不同的项目实践来不断优化这些定型后的东西，最终总结出一套最佳实践出来。它应该能够成为其它项目测试活动的参照，甚至是标准。这个时候，我们会发现所有的项目都在有序、统一、高效、可靠的进行。

◆ 扩大影响，组织共赢阶段

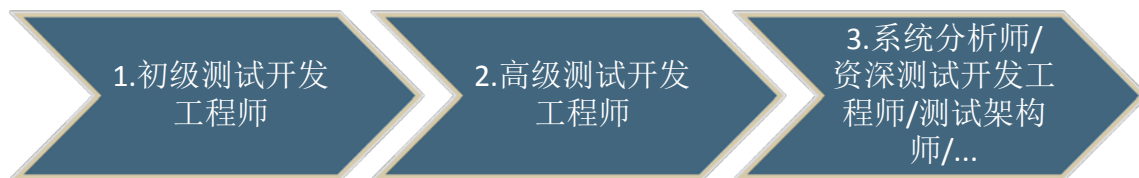
那么到达上面这个目标之后是不是就是接口测试团队的终点呢？显然不是，不要忘了，到目前为止，无论你在接口测试的工作上做得再好，那也仅仅只局限在接口测试本身上。我们不应该满足于此。通常来说接口测试团队在整个质量保证团队中占据了众多的核心技术人员。他们擅长使用各种技术来解决问题，甚至比开发团队做得还好。拥有如此多的技术资源，如果我们不懂得合理利用，那真的是很大的浪费。在做好接口测试本身的基础上，我们还应该积极了解其它测试团队面临哪些问题，这些问题是不是可以利用技术手段来解决，如果可以，我们是否可以为他们实现一些实用的工具来帮助他们解决问题或者提高工作效率；我们自己的技术是否需要分享给其它测试团队，甚至是整个软件团队，以帮助他们更好地完成工作。总之，我们应该思考如何更有效、更合理地利用接口测试团队的资源，来提高整个组织的业绩，这不仅会扩大接口测试团队本身的影响力，让接口测试团队成为整个组织的核心竞争力，同时它还创造了一个共赢的局面。

另一方面，在工作的流程上，各个测试角色是可以互补的，接口测试的设计、用例可以跟功能和性能测试共享，接口测试的报告可以作为功能测试的重要参考，让其了解底层都经历了哪些测试，哪里是 bug 的密集区，哪里相对安全一些。在功能测试工程师找到 bug 之后，接口测试工程师可以用代码直接覆盖这个 bug 产生的代码，使这个 bug 永远不会出现第二次。接口测试人员还可以直接绕过页面，对底层系统进行性能和压力的测试，在测试中各个角色的密切配合，也减少了测试的成本，提供系统全方位的质量保障。

3 接口测试的定位

3.1 人员能力定位

- ◆ 熟悉软件测试流程,测试理论和测试方法。能够根据测试需求,制定测试计划和设计测试用例。
- ◆ 了解软件工程理论知识和开发流程,有一定的编程能力。能够根据测试用例,准备测试数据以及编写和执行测试脚本,并对软件 bug 进行跟踪分析和报告。
- ◆ 掌握 JUnit, Spring, Maven, CruiseControl, DBUnit, Unitils, Ibatis 等技能,并且能够运用这些技能搭建接口测试框架。
- ◆ 思维活跃,善于发现问题,有较强的逻辑分析能力和学习能力。
- ◆ 具备良好的表达和沟通能力。
- ◆ 工作认真细致,踏实肯干,责任心强。



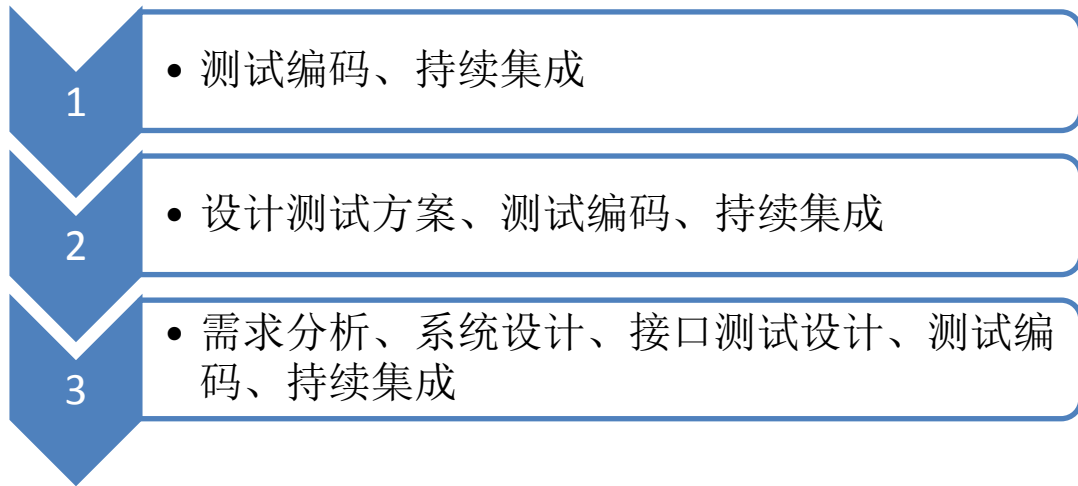
在测试开发工程师发展的每个阶段,对以上几点的要求是不同的。

3.2 职责定义

我们的客户是调用接口的人,不是开发接口的人
对业务的理解要达到开发人员的水平
掌握软件测试的理论知识
要能够独立设计和开发测试,有定位问题的能力
要能搭建系统的测试框架
有权利在质量不达到要求的情况下阻止产品(项目)的发布

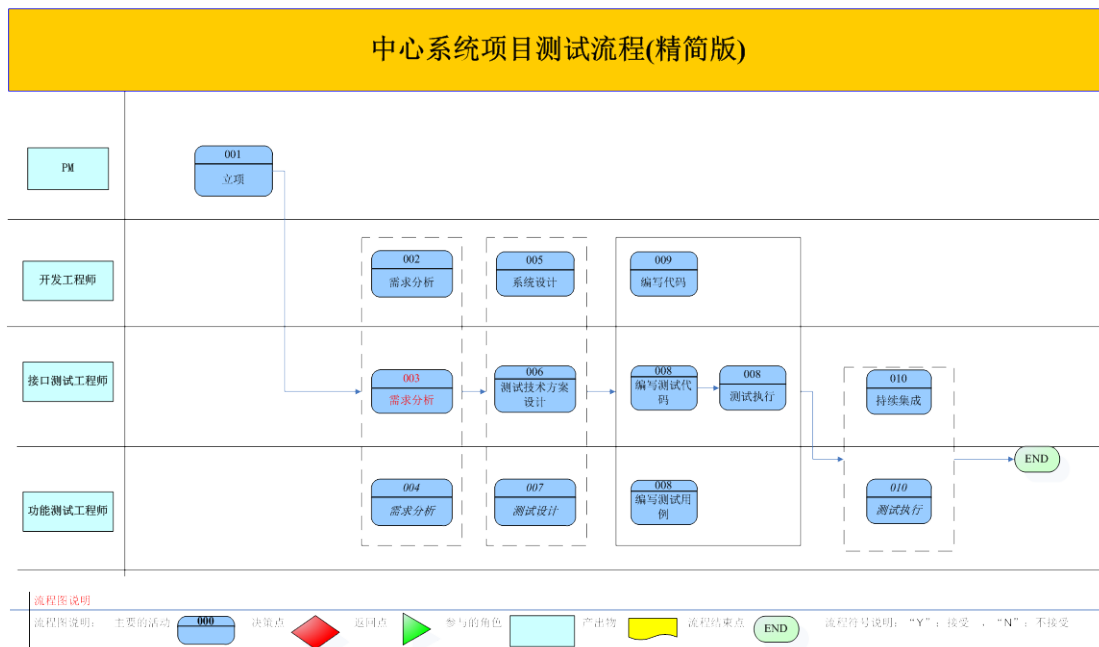
3.3 工作内容定位

下图是接口测试在各个发展阶段的工作内容,在初期阶段以编码和持续集成为主,以后逐渐增加测试方案的设计和系统本身的分析、设计内容,最终为系统的整体质量负责。

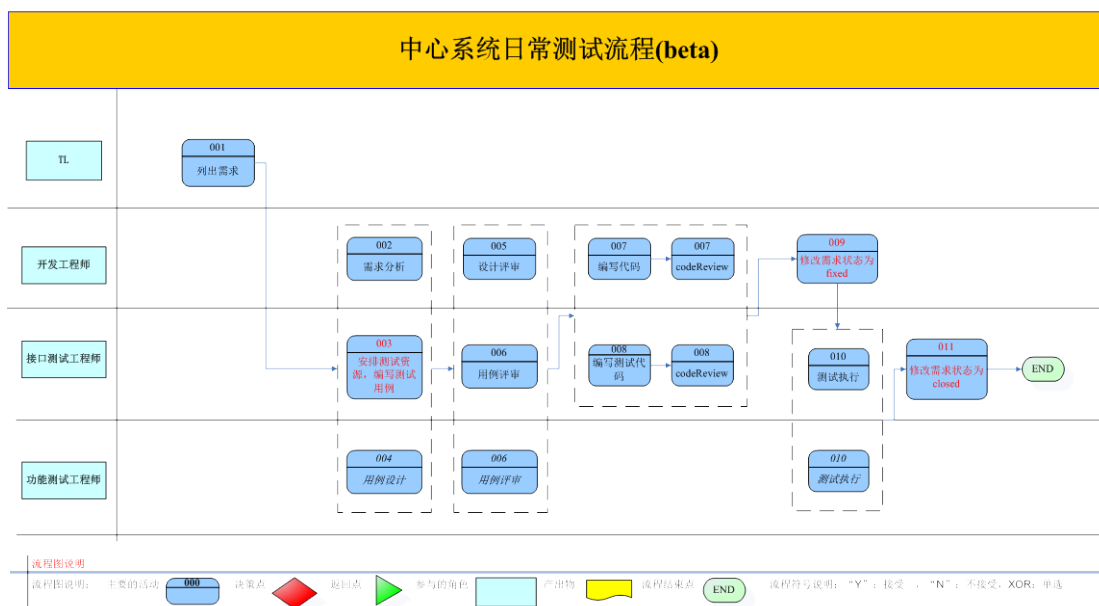


4 接口测试的流程

4.1 项目工作流程



4.2 日常工作流程



规范而统一的工作流程是大家分工合作的基础，只有每一个人都遵循统一的流程，项目才可能统一有序地进行。因而，规范的工作流程对提高团队的合作能力以及工作效率都有至关重要的作用。

4.3 流程步骤详解

根据以往的实践经验总结，接口测试可以分为以下几个步骤：需求分析和设计评审，测试框架和技术选型，测试计划制定，测试环境搭建，测试用例设计和评审，测试实现和执行，持续集成。下面，将对每一个步骤作详细说明。

4.3.1 需求分析和设计评审

几乎所有软件活动都是以需求分析开始的，接口测试也是如此。在本阶段我们有两个任务：一是充分理解需求，并保证所有人对需求的理解一致；二是尽可能找出需求本身所存在的问题。

需求分析之后，紧接着就应该进入系统设计阶段了。系统设计不应该仅仅只是系统设计师或者开发人员的事情，作为接口测试人员，应该可以从测试的角度为系统的设计提供一些方案或者建议，优化设计的同时提高系统的可测性。

4.3.2 测试框架和技术选型

系统设计评审之后，系统实现所需要使用到的技术都应该已经选定。在这个阶段，接口测试人员就需要根据系统设计来选定自己的测试框架和要使用到的技术。当然，这并不是必须的，如果你所测试的项目和之前已经测试过的项目技术架构都差不多的话，那么你可以沿用之前的测试框架和技术，或者在这个基础上稍加调整。如果所测试的项目采用的是一种不同的技术架构，那么你就需要仔细考虑如何选定与之相适应的测试框架和技术。

接口测试框架和技术的选型有很多因素，原则就是选择一个能满足你的测试需要的最好用的框架和技术，并且尽量是你的项目成员都比较熟悉的框架和技术。没有必要单纯为了提高测试的技术含量而选用功能奇多但却复杂难懂的工具来使用。

4.3.3 测试计划制定

接口测试的测试计划制定基本上和功能测试差不多。这个阶段主要要明确有哪些测试资源，测试资源如何分配，在整个测试过程中需要完成哪些事情，每个时间点应该完成哪些事情，还有最重要的也是很容易被忽略掉的一点就是风险评估。虽然我们不可能识别出所有的风险，但是我们可以根据经验值来识别出大部分的潜在风险并加以管理。良好的风险管理是一个软件团队成熟的体现。

4.3.4 测试环境搭建

在测试框架和技术选型完成以后就可以开始进行测试环境的搭建了，在接口测试中典型的环境搭建过程可能是这样的：首先你会为接口测试建立一个基本的工程，并为这个工程设计一个良好的结构，在这个工程中引入你所选定的测试框架和依赖，为这些框架和依赖编写好必要的配置文件，将该工程和待测系统的工程以某种形式联系在一起（通常是项目依赖），在该环境下能运行通过一个最基本的测试。

4.3.5 测试用例设计和评审

接口测试的测试用例设计是以接口为单位来设计测试,在设计的过程中我们重点关注的是接口有哪些可能的输入参数和预期的输出结果是什么。当然在需要的时候,也要考虑接口的性能和所期望承受的压力等。在这个过程中很重要的一点就是为不同的测试划分优先级,这在测试资源不足的情况下将会指导你哪些测试应该优先完成,哪些测试可以延迟完成。即便是在测试资源很充足的情况下,你也可以按照优先级来完成测试,这样一旦遇到某个风险爆发,那么基本上可以保证,优先级高的工作已经完成了,而不至于惊慌失措。

测试用例设计出来以后应该经过评审,并将评审结果以某种形式记录下来,作为测试实施的最终方案。评审最好由以下这些人员共同参与:需求方、设计人员、开发人员、功能测试人员、接口测试人员以及这些人员的直接主管。不同的角色会从不同角度对测试设计进行考虑,因此在这个过程中会使测试设计得到极大的完善。

4.3.6 测试实现和执行

测试设计一旦产出并通过评审,那么测试实现相对来说就是一件比较简单的事情了。无非是将一个个测试用例用编程语言实现出来并运行通过。

在测试实现的过程中可能会发现测试设计不够完善,或者是因为需求的变更而导致需要增加新的测试用例。不管是因为什么原因,在实现测试的过程中,一旦发现有可以完善的地方就应该立刻记录下来,这样可以更有效地保证测试的完备性。

在这个过程中我们还应该产出测试报告(包括日报和最终报告),让整个团队都及时掌握项目的质量情况,以便不同角色正确安排工作。

4.3.7 持续集成

持续集成是接口测试实现全面自动化回归测试的重要技术手段。简单来说,持续集成就是把写好的测试代码持续不断地运行起来,并且利用版本控制技术,让测试代码测试的始终是最新版本的系统接口。

当接口测试进行到这个阶段的时候,我们的目标就是要让测试代码持续不断的运行,并且保证在运行不通过的时候及时定位并解决问题。在开发人员维护系统的时候,我们同时也会根据持续集成的结果来维护我们的测试代码。

最后,需要注意的是,虽然以上提及的步骤是我们接口测试人员遵循的规范,但是不同于功能测试等其他测试,接口测试需要与开发同步进行。如下图所示,在项目启动的时候我们就要参与进去,在编码结束时测试也基本完成,中间的每个步骤也与开发紧密相关。因此,我们接口测试的工程师又叫测试开发工程师,我们既需要测试的知识,又必须具有一定的编码能力。

4.4 质量评估标准

- ◆ 接口覆盖率是否达到要求。

- 1) 所有供外部调用的接口必须有相对应的测试用例，覆盖率要达到 95%以上。
- 2) 所有供内部调用并涉及到产品主要功能的接口测试用例覆盖率要达到 90%以上。
- 3) 所有供内部调用并涉及到次要功能的接口，测试代码覆盖率可以随接口复杂度和重要程度增高而增加。

◆ 测试用例中对接口业务规则的验证是否完整。

- 1) 测试用例要覆盖接口的主要业务规则。

接口的主要业务规则，就是该接口的主要功能，它影响着接口的业务实现和调用状态。如发布一个宝贝，那么发布一个全新、二手、拍卖、闲置的宝贝等等就是主要功能。

- 2) 测试用例要覆盖接口的常用业务规则。

还是发布宝贝的例子，80%的卖家都会在“描述”中加入图片，旺旺链接等。这个业务规则不会影响接口的正常调用。但却影响着用户的使用习惯。所以测试用例中必须包含对“描述”字段中含有图片链接，旺旺链接的验证。

- 3) 参数验证要覆盖对边界值和参数特有业务规则的验证。

很多接口中都会对其参数有一定的限制，如一个字段长度限制为<5，那么就至少要存在对该字段的长度为 4，5 的测试用例。

◆ 测试用例中是否覆盖接口之间的关联性测试。

如：一个添加接口的关联性测试，就要以该添加接口的返回值为参数，来调用其他关联接口如修改和删除接口，验证其是否可调用并且调用成功。

◆ 遗留的 bug 对系统的影响程度。

- 1) 经常调用的接口，不可含有主要业务规则和常用业务规则相关的 bug，次要业务规则的 bug 遗留率为 0.2%以下。
- 2) 不常调用的接口，不可含有主要业务规则的 bug，常用业务规则的 bug 遗漏率为 2%以下，次要业务规则的 bug 遗漏率为 5%以下。

◆ 测试用例与测试代码是否一致。

◆ 测试用例是否可持续回归。

◆ 经过测试的接口是否达到了调用方的标准，调用方能否使用该接口来开发出产品设计说明书所设计的应用。

5 接口测试的技术简介

接口测试用到的框架和技术很多,我们本着不重复发明轮子但让轮子协同工作的原则对这些框架技术进行整合、扩展,不断增强其功能和使用方便性。常用的技术简介如下。

5.1 Junit

JUnit 是 java 语言事实上的标准测试框架,是接口测试技术中最基本的利器,有以下主要特性:

- ◆ 批量运行
无论是利用 JUnit 命令行,还是 Eclipse,或者是当下很流行的集成测试框架,我们都可以使用一个简单的命令,按钮或者操作来启动我们成百上千或者更多的测试用例,想象一下,如果让我们用人力来做操作,那将是多么一件耗时耗力的事。
- ◆ 断言机制
通过断言机制,让我们的程序能够自动的判断测试结果是否正确,而无需我们人工的去分析和判断。通过批量运行和断言机制,使得我们的自动化测试变得可能。
- ◆ 测试报告
当测试运行完毕后,JUnit 可以产生很全面的测试结果,成功还是失败一目了然,对于失败的用例会报告明确的错误信息。这些都让我们对整个测试的运行情况有很好的掌控。
- ◆ 易于扩展
当下越来越多的工具都在基于 JUnit 的骨架上进行扩展,无论是 Unitils, Eclipse, 还是 Maven 都对 JUnit 提供了很好的支持。

5.2 DbUnit

DbUnit是一个基于JUnit扩展的数据库测试框架,目的是在测试运行前后使数据库处于可知状态。它提供了大量的类对与数据库相关的操作进行了抽象和封装,运用DbUnit的一般流程如下:

- 1) 根据业务准备测试用的数据,一般准备成 Excel 格式的数据集;
- 2) 在测试执行前,将数据集更新到数据库;
- 3) 在测试执行后,将数据库恢复到测试前的状态。

在实际测试中直接运用DbUnit的情况很少,通常是跟其他技术(如Unitils)一起配合使用。

5.3 Spring TestContext Framework

Spring TestContext Framework 是 Spring 2.5 版本推出的测试框架,方便使用 Spring 容器进行集成测试,而不依赖应用服务器或其它部署环境,是测试 Spring 应用程序的首选良器。主要提供以下功能:

- ◆ Spring 上下文管理和缓存

提供对 Spring 上下文的持久化载入和缓存机制，节省 Spring 容器启动时间，从而缩短大型项目集成测试执行时间，有效提高测试效率。

- ◆ 测试 fixtures 依赖注入
通过依赖注入选择配置测试类实例，方便使用已有的配置文件搭建测试 fixtures，实现 Spring 上下文在多个测试场景中的重用，从而能避免为单个的测试案例重复进行测试 fixture 搭建。
- ◆ 事务管理
适用于测试中需要调用事务代理对象的情形，对每次测试创建并默认回滚事务，避免因改变数据库的状态而影响后面的测试。
- ◆ 集成测试支持类
提供若干抽象类简化集成测试编码，方便测试类直接访问 ApplicationContext 来进行显式 bean 查找或整体测试上下文状态，访问 JdbcTemplate 或 SimpleJdbcTemplate 来验证数据库状态改变。
- ◆ 监听器机制
通过监听器实现依赖注入、事务管理等功能的灵活配置而无需继承任何 Base 类，提供了良好的扩展点方便实现自定义监听器实现特定功能。

5.4 Unitils

Unitils 是辅助单元测试的一个开源类库，其目的是使单元测试更为简单和便于维护。同时也对一些现有的类库（如 dbunit）进行更好的扩展，并且可以和 JUnit 以及 TestNG 测试框架进行集成。

Unitils 提供通用的断言工具，支持数据库测试，支持使用 Mock 进行测试，并提供对 Spring 以及 Hibernate 的集成。

Unitils 提供以下功能：

- ◆ 常用的测试工具。
通过反射来实现等值断言，并提供忽略 Java 默认值以及空值以及集合内顺序等选项。
- ◆ Mock 对象的支持
 - 1) 通过简单的语法实现动态定义句柄/存根（stub）的行为以及对 Mock 调用的验证；
 - 2) 良好的反馈，包括一个简单并且可扩展的调用场景报告以及建议式的断言声明；
 - 3) 通过使用参数匹配器来解耦方法参数间的约束，并可混合实际对象一起使用，当某些参数不对测试产生影响时可以使用空值；
 - 4) 当存根（stub）数据对象并不需要提供具体行为相应时，可使用虚假对象。
- ◆ 持久层测试的支持
 - 1) 通过一些增长的，可重复的，或后期处理脚本来自动管理数据库；
 - 2) 提供自动地使数据库约束失效，将序列（sequences）设为最小值等功能，支持 Oracle, Hsqldb, MySql, DB2, Postgresql, MsSql and Derby；
 - 3) 简化测试数据库连接的建立；
 - 4) 通过 DBUnit 简化测试数据的插入；
 - 5) 提供 Hibernate SessionFactory 的创建以及 session 的管理；
 - 6) 提供 Hibernate 之类的 ORM 类库的数据库映射的自动测试。
- ◆ 对集成 EasyMock 的支持
 - 1) 简化 EasyMock mock 对象的支持；

- 2) 简化 mock 对象的注入;
 - 3) 使用反射机制匹配 EasyMock 参数。
- ◆ Spring 集成
 - 1) 通过上下文配置, 使得 Spring 管理的 bean 可以简单的注入到单元测试中;
 - 2) 支持在单元测试中使用由 Spring 配置的 Hibernate SessionFactory。

5.5 TestNG

TestNG 是为了解决 Junit 过于简单的问题而产生的, 它与 Junit 是同一类的框架, 两者不能同时使用, 这与对 junit 的扩展框架如 dbunit, unitils 等有非常大的区别。

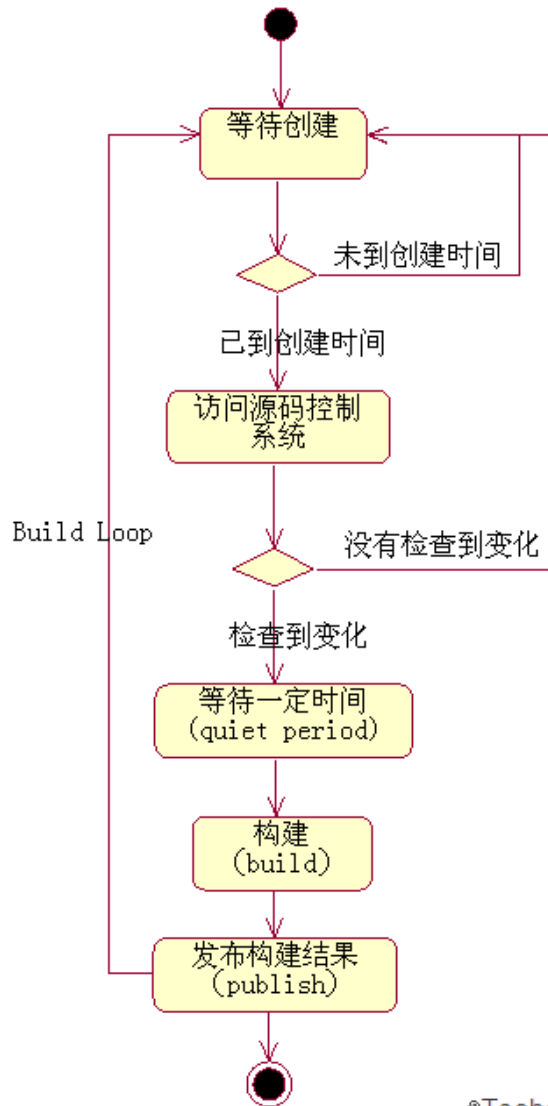
TestNG 相比 Junit 增强功能如下:

- ◆ 定义测试组的能力, 每个测试方法都可以与一个或多个组相关联, 但可以选择只运行某个测试组。
- ◆ 重新运行失败的测试, 对于每天都进行编译来说非常有帮助。
- ◆ 提供了依赖检查机制, 并可以严格控制执行顺序。
- ◆ 可以简单的直接进行多线程测试了。
- ◆ 提供 xml 方式的参数化测试。

5.6 CruiseControl

CruiseControl 是一种持续集成过程的框架, 包括了旺旺消息通知、邮件通知, Ant、Maven 和各种源码控制工具 (SVN、CVS) 的插件, 并提供了 web 接口, 用于查看当前和以前的构建的结果。

通过运用 CruiseControl 持续集成实现自动化构建脚本和测试完全自动化, 其自动构建机制工作流程流程如下图示。



@Taobao.com

CC 自动构建流程

5.7 Clover

Clover 是 Atlassian 组织的一款优秀统计测试覆盖率插件，可以免费用于非商业途径，可以为 Maven2 构建的项目生成报告。Clover 通过以下常用指标衡量测试覆盖率。

- ◆ Statement coverage， 也称作 Line coverage， 用于评价测试的代码语句覆盖率。
- ◆ Basic block coverage， 是 Statement coverage 的一个变种， 它把没有一个分支的代码区域作为一个计量单位， 而不是简单的代码行， 用于一个 if-else 分支代码行数远远大于另一个的情况， 在这种情况下， statement coverage 指标并不适用。
- ◆ Decision coverage（也称作 Branch coverage）， 用于评价代码分支地测试覆盖率。
- ◆ Path coverage， 和 Decision coverage 相似， 用于评价代码从开始到结束所有路径的测试覆盖率。
- ◆ Function coverage， 用于评价代码方法的测试覆盖率。

5.8 Mock

在测试当中，mock 是指使用各种技术手段模拟出各种需要的资源以供测试使用。被 mock 的资源通常有以下特征：

- ◆ 被测目标依赖该资源
- ◆ 该资源可能因为各种原因不稳定、返回结果不断变化或者并不总是能够获取到
- ◆ 该资源跟被测目标本身质量无关

这些资源可能是一个外部或底层接口、一个系统、一组数据对象或者是一整套目标软件的工作环境等。通过 mock 避免对外部真实资源的依赖实现对被测目标的孤立测试，从而大大降低测试的难度，节约测试成本。

需要注意的是利用 mock 通过的测试与使用真实环境通过的测试毕竟还是有一定差别的。有些时候我们就是需要所测试的系统能够处理依赖所产生的各种情况，包括正常情况和异常情况，我们同样不能保证我们的 mock 可以模拟到每种这样的情况。因此只在确实有必要的情况下才运用 mock。

6 接口测试的方向

在经历了接口测试的发起、稳定、成型以后，我们有必要探讨一下接口测试的未来。然而，这却是一个十分困难的命题，因为，未来永远是一个谜。任何形式的预测，最终结果常常成为笑料，IT 业界尤其如此。鉴于此，我们需要把更多的目光关注到接口测试的发展方向上，而不是接口测试发展的最终结果的预测上。

◆ 接口测试之框架改进

目前，我们已经形成了完整的接口测试框架。但在不同的项目中，差别仍然很大，没有形成统一的基础架构，从而导致构建接口测试框架的过程比较繁琐，因此，我们可以在以下几个方面进行改进与优化：

- 1、测试数据管理框架构建与统一；
- 2、接口测试项目构建基础框架；
- 3、mock 框架化；
- 4、高比例代码自动生成框架；
- 5、接口测试工具集与三方库的本地化应用。

◆ 接口测试之持续集成

对于接口测试的技术而言，核心内容是持续集成，即自动化。其实，这个内容和其他的测试也是一致的，从本质上来说，自动化代表了未来测试发展的主流方向。目前，我们已经实现了接口测试的持续集成，而更高度的自动化、更加广泛的自动化是我们未来的发展方向。

更高度的自动化包含：1、持续集成框架的柔性改进，包括更加丰富的测试结果、更加人性化的结果展现，测试结果旺旺/邮件推送，而不仅仅是项目构建失败的提醒；2、持续集成中接口测试项目自动构建与部署。

更加广泛的自动化包含：1、各开发语言接口测试持续集成，HUDSON 的研究与应用；2、将接口测试应用到更加广泛的项目当中；3、持续集成框架与测试框架（用例、缺陷）的整合。

◆ 接口测试之测试驱动

接口测试是白盒测试的一部分。作为接口测试而言，我们不仅需要保证代码的质量、系统的性能，我们还需要保证系统开发过程的正确性与高效性。测试驱动开发是接口测试的一个重要思想。

在接口测试的过程中，我们要力求做到覆盖到更多的代码行，覆盖到更多的逻辑过程，驱动开发更加准确、高效的实现自己的代码。同时，我们需要对代码进行分析与评审，驱动开发提高代码的质量。

接口定义在系统级架构的过程中，是非常重要的一个环节。接口定义是否合理，需要从更高层面，譬如系统的交互性、系统的易用性、系统的可扩展等方面来考虑。接口测试的职责，需要从这些方面来约束和规范系统的接口定义过程，驱动开发完善设计，避免接口定义的随意性以及接口改动的频繁性。接口的改动对系统的伤害是不言而喻的。因此，从一开始就驱动开发完善接口定义是接口测试的一个重要发展方向。

◆ 接口测试之未来遐思

我们生活在信息膨胀的时代，IT 行业的发展越来越快，也对测试提出了更高的要求。提出成本更低，效率更高的测试技术、测试框架、测试方法、测试理论是我们未来能适应高速发展的 IT 行业的基本要求。因此，我们提出如下的设想：

测试虚拟化：提供接口测试虚拟机，构建测试虚拟化层。将被测系统运行在虚拟机中，与外部系统剥离，进行内部代码检测、内存检测、数据校验与逻辑检测。

测试智能化：智能分析系统代码，智能生成测试代码，智能 mock 外部系统，智能执行测试代码，智能分析测试结果，智能定位缺陷，智能修复缺陷。

7 参考资料

郭芙《为什么要做接口测试》 <http://rdc.taobao.com/blog/qa/?p=307>

子柳《接口测试在淘宝的应用》 <http://rdc.taobao.com/blog/qa/?p=435>

子柳《测试复杂度模型》 <http://rdc.taobao.com/blog/qa/?p=2070>

宋缺《单元测试三叉戟—JUNIT, DBUNIT, UNITILS》 <http://rdc.taobao.com/blog/qa/?p=686>

梁建钊博客 <http://www.51testing.com/?uid-13997>

Aoxj <http://www.blogjava.net/aoxj/archive/2008/03/18/187105.html>

8 作者介绍

- ◆ 东坡
编写“接口测试的背景”章节
- ◆ 宋缺 (<http://rdc.taobao.com/blog/qa/?author=44>)
编写“接口测试的目的”章节
- ◆ 均量 (<http://rdc.taobao.com/blog/qa/?author=79>)
编写“接口测试的目的”、“接口测试的技术简介-Mock”章节
- ◆ 玉林 (<http://rdc.taobao.com/blog/qa/?author=225>)
编写“接口测试的定位”章节
- ◆ 子柳 (<http://rdc.taobao.com/blog/qa/?author=48>)
编写“接口测试的流程-项目工作流程”章节
- ◆ 守容
编写“接口测试的流程-日常工作流程”章节
- ◆ 婉佩 (<http://rdc.taobao.com/blog/qa/?author=41>)
编写“接口测试的流程-质量评估标准”章节
- ◆ 叶渡 (<http://rdc.taobao.com/blog/qa/?author=222>)
统编“接口测试的技术简介”，编写“Spring TestContext Framework”、“DbUnit”章节
- ◆ 一环 (<http://rdc.taobao.com/blog/qa/?author=220>)
编写“Spring TestContext Framework”章节
- ◆ 七修 (<http://rdc.taobao.com/blog/qa/?author=216>)
编写“接口测试的技术简介-JUnit”章节
- ◆ 元化 (<http://rdc.taobao.com/blog/qa/?author=83>)
编写“接口测试的技术简介-testNG”章节
- ◆ 光远 (<http://rdc.taobao.com/blog/qa/?author=63>)
编写“接口测试的技术简介-CruiseControl 和 clover”章节
- ◆ 见田 (<http://rdc.taobao.com/blog/qa/?author=219>)
编写“接口测试的技术简介-unitils”章节
- ◆ 文朗 (<http://rdc.taobao.com/blog/qa/?author=76>)
编写“接口测试的方向”章节

致谢：参与评审的人员有：郭芙、宝妮、瑶红、石棱、荣姣、长倩

编者注：第一个版本，疏漏之处在所难免。后续有更多精彩内容，将会有技术细节，开源软件修改方案和最佳架构实践。各位看官发现 bug 或有建议请不吝赐教。