12 | 测试也是程序员的事吗?

郑晔 2019-01-25

你好，我是郑晔。

在“任务分解”这个模块，我准备从一个让我真正深刻理解了任务分解的主题开始，这个主题就是“测试”。

这是一个让程序员又爱有恨的主题，爱测试，因为它能让项目的质量有保证；恨测试，因为测试不好写。而实际上，很多人之所以写不好测试，主要是因为他不懂任务分解。

在上一个模块，我们提到了一些最佳实践，但都是从“以终为始”这个角度进行讲解的。这

次，我准备换个讲法，用五讲的篇幅，完整地讲一下“开发者测试”，让你和我一起，重新认识这个你可能忽视的主题。

准备好了吗？我们先从让很多人疑惑的话题开始：程序员该写测试吗？

谁要做测试？

你是一个程序员，你当然知道为什么要测试，因为是我们开发的软件，我们得尽可能地保证它是对的，毕竟最基本的职业素养是要有的。

但测试工作应该谁来做，这是一个很有趣的话题。很多人凭直觉想到的答案是，测试不就该是测试人员的事吗，这还用问？

测试人员应该做测试，这是没错的，但是测试只是测试人员的事吗？

事实上，作为程序员，你多半已经做了很多测试工作。比如，在提交代码之前，你肯定会把代码跑一遍，保证提交的基本功能是正确的，这就是最基本的测试。但通常，你并不把它当成测试，所以，你的直觉里面，测试是测试人员的事。

但我依然要强调，测试应该是程序员工作的一部分，为什么这么说呢？

我们不妨想想，测试人员能测的是什么？没错，他们只能站在系统外部做功能特性的测试。而一个软件是由它内部诸多模块组成的，测试人员只从外部保障正确性，所能达到的效果是有限的。

打个比方，你做一台机器，每个零部件都不保证正确性，却要让最后的结果正确，这实在是一个可笑的要求，但这却真实地发生在软件开发的过程中。

在软件开发中有一个重要的概念：软件变更成本，它会随着时间和开发阶段逐步增加。也就是说我们要尽可能早地发现问题，修正问题，这样所消耗掉的成本才是最低的。

上一个模块讲“以终为始”，就是在强调尽早发现问题。能从需求上解决的问题，就不要到开发阶段。同样，在开发阶段能解决的问题，就不要留到测试阶段。

你可以想一下，是你在代码中发现错误改代码容易，还是测试了报了 bug，你再定位找问题方便。

更理想的情况是，质量保证是贯穿在软件开发全过程中，从需求开始的每一个环节，都将“测试”纳入考量，每个角色交付自己的工作成果时，都多问一句，你怎么保证交付物的质量。

需求人员要确定验收标准，开发人员则要交出自己的开发者测试。这是一个来自于精益原则的重要思想：内建质量（Build Quality In）。

所以，对于每个程序员来说，只有在开发阶段把代码和测试都写好，才有资格说，自己交付的是高质量的代码。

自动化测试

不同于传统测试人员只通过手工的方式进行验证，程序员这个群体做测试有个天然的优势：会写代码，这个优势可以让我们把测试自动化。

早期测试代码，最简单的方式是另外写一个程序入口，我初入职场的时候，也曾经这么做过，毕竟这是一种符合直觉的做法。不过，既然程序员有写测试的需求，如此反复出现的东西，就会有更好的自动化方案。于是开始测试框架出现了。

最早的测试框架起源是 Smalltalk。这是一门早期的面向对象程序设计语言，它有很多拥趸，很多今天流行的编程概念就来自于 Smalltalk，测试框架便是其中之一。

真正让测试框架广泛流行起来，要归功于 Kent Beck 和 Erich Gamma。Kent Beck 是极限编程的创始人，在软件工程领域大名鼎鼎，而 Erich Gamma 则是著名的《设计模式》一书的作者，很多人熟悉的 Visual Studio Code 也有他的重大贡献。

有一次，二人一起从苏黎世飞往亚特兰大参加 OOPLSA（Object-Oriented Programming, Systems, Languages & Applications）大会，在航班上两个人结对编程

写出了 JUnit。从这个名字你便不难看出，它的目标是打造一个单元测试框架。

顺便说一下，如果你知道 Kent Beck 是个狂热的 Smalltalk 粉丝，写过 SUnit 测试框架，就不难理解这两个人为什么能在一次航班上就完成这样的力作。

JUnit 之后，测试框架的概念逐渐开始流行起来。如今的“程序世界”，测试框架已经成为行业标配，每个程序设计语言都有自己的测试框架，甚至不止一种，一些语言甚至把它放到了标准库里，行业里也用 XUnit 统称这些测试框架。

这种测试框架最大的价值，是把自动化测试作为一种最佳实践引入到开发过程中，使得测试动作可以通过标准化的手段固定下来。

测试模型：蛋卷与金字塔

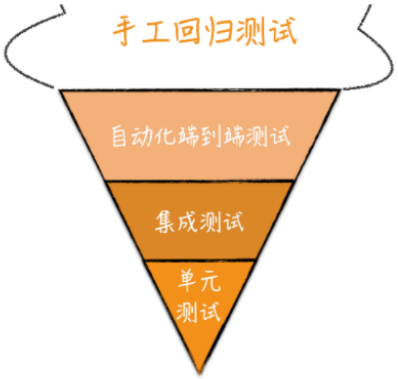
在前面的讨论里，我们把测试分为人工测试和自动化测试。即便我们只关注自动化测试，也可以按照不同的层次进行划分：将测试分成关注最小程序模块的单元测试、将多个模块组合在一起的集成测试，将整个系统组合在一起的系统测试。

有人喜欢把验收测试也放到这个分类里。为了简化讨论，我们暂时忽略验收测试。

随之而来的一个问题是，我们应该写多少不同层次的测试呢？理论上固然是越多越好了，但实际上，做任何事都是有成本的，所以，人们必须有所取舍。根据不同测试的配比，也就有了不同的测试模型。

有一种直觉的做法是，既然越高层的测试覆盖面越广，那就多写高层测试，比如系统测试。

当然，有些情景高层的测试不容易覆盖到的，所以，还要有一些底层的测试，比如单元测试。在这种情况下，底层的测试只是作为高层测试的补充，而主力就是高层测试。这样就会形成下面这样一种测试模型：冰淇淋蛋卷。



听说过冰淇淋蛋卷测试模型的人并不多，它是一种费时费力的模型，要准备高层测试实在是太麻烦了。

之所以要在这里提及它，是因为虽然这个概念很多人没听说过，但是有不少团队的测试实际采用的就是这样一种模型，这也是很多团队觉得测试很麻烦却不明就里的原因。

接下来，要说说另一种测试模型，也是行业里的最佳实践：测试金字塔。

Mike Cohn 在自己的著作《Succeeding with Agile》提出了测试金字塔，但大多数人都是通过 Martin Fowler 的文章知道的这个概念。

从图中我们不难看出，它几乎是冰淇淋蛋卷的反转，测试金字塔的重点就是越底层的测试应该写得越多。

想要理解测试金字塔成为行业最佳实践的缘由，我们需要理解不同层次测试的差异。**越是底层的测试，牵扯到相关内容越少，而高层测试则涉及面更广。**

比如单元测试，它的关注点只有一个单元，而没有其它任何东西。所以，只要一个单元写好了，测试就是可以通过的；而集成测试则要把好几个单元组装到一起才能测试，测试通过的前提条件是，所有这些单元都写好了，这个周期就明显比单元测试要长；系统测试则要把整个系统的各个模块都连在一起，各种数据都准备好，才可能通过。

这个模块的主题是“任务分解”，我必须强调一点：**小事反馈周期短，而大事反馈周期长。**小事容易做好，而大事难度则大得多。所以，以这个标准来看，底层的测试才更容易写好。

另外，因为涉及到的模块过多，任何一个模块做了调整，都有可能破坏高层测试，所以，高层测试通常是相对比较脆弱的。

此外，在实际的工作中，有些高层测试会牵扯到外部系统，这样一来，复杂度又在不断地提升。

人们会本能地都会倾向于少做复杂的东西，所以，人们肯定不会倾向于多写高层测试，其结果必然是，高层测试的测试量不会太多，测试覆盖率无论如何都上不来。而且，一旦测试失败，因为牵扯的内容太多，定位起来也是非常麻烦的。

而反过来，将底层测试定义为测试主体，因为牵扯的内容少，更容易写，才有可能让团队得到更多的测试，而且一旦出现问题，也会更容易发现。

所以，虽然冰淇淋蛋卷更符合直觉，但测试金字塔才是行业的最佳实践。

当测试金字塔遇到持续集成

测试金字塔是一个重要实践的基础，它就是持续集成。当测试数量达到一定规模，测试运行的时间就会很长，我们可能无法在本地环境一次性运行所有测试。一般我们会选择在本地运行所有单元测试和集成测试，而把系统测试放在持续集成服务器上执行。

这个时候，底层测试的数量就成了关键，按照测试金字塔模型，底层测试数量会很多，测试可以覆盖主要的场景；而按照冰淇淋蛋卷模型，底层测试的数量则有限。

作为提交代码的防护网，测试数量多寡决定着得到反馈的早晚。所以，金字塔模型与持续集成天然就有着很好的配合。

需要特别注意的是，**不是用单元测试框架写的测试就是单元测试。**很多人用单元测试框架写的是集成测试或是系统测试。单元测试框架只是一个自动化测试的工具而已，并不是用来定义测试类型的。

在实际工作中，区分不同测试有很多种做法，比如，将不同的测试放到不同的目录下，或是给不同类型的测试一个统一的命名规范。

区分不同类型测试主要目的，主要是在不同的场景下，运行不同类型的测试。就像前面提到的做法是，在本地运行单元测试和集成测试，在持续集成服务器上运行系统测试。

总结时刻

测试是软件开发重要的组成部分，测试应该是软件开发团队中所有人的事，而不仅仅是测试人员的事。因为软件变更成本会随着时间和开发阶段逐步增加，能在早期解决的问题，就不要将它延后至下一个阶段。

在测试问题上，程序员有着天生的优势，会写代码，于是，程序员拥有了一个突出的强项，自动化测试。写测试应该是程序员工作完成的重要组成部分。

随着人们对于测试理解的加深，各种各样的测试都出现了，也开始有了测试的分类：单元测试、集成测试、系统测试等等。越在底层测试，成本越低，执行越快；越在高层测试，成本越高，执行越慢。

人的时间和精力是有限的，所以，人们开始思考不同的测试如何组合。在这个方面的最佳实践称之为测试金字塔，它强调的重点是，越底层的测试应该写得越多。只有按照测试金字塔的方式写测试，持续集成才能更好地发挥作用。

如果今天的内容你只能记住一件事，那请记住：**多写单元测试。**

最后，我想请你分享一下，你的团队在写测试上遇到哪些困难呢？欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言(28)

Kǎfkā²⁰²⁰

我有个执念，不愿主动写测试代码的程序员，不太可能是优秀的程序员

作者回复: 我认同你的说法

2019-01-25 38

张飞洪

团队认知，开发周期，软件和生命财产关系不大，是单元测试的拦路虎

作者回复: 还有一点是，知识，很多人不愿意写测试的原因是不会写测试。

2019-01-25 16

Nemo

能不能手把手教一下如何写一个完整，优秀的单元测试?

作者回复: 好建议，我可以考虑加餐。

2019-02-13 15

Johnsen □

像前端项目主要以UI为主，版本迭代速度又很快的情况下怎么进行单元测试的编写

作者回复: 首先，迭代速度快慢与是否写测试没关系，取决于工作是否完成是前面提到的 DoD。其次，前端之所以能够在今天成为一个独立的项目，在于它有大量的逻辑需要写，如今 JavaScript 相关的测试框架已经发展得很完整了，就按照正常的方式去写测试就好了。

2019-01-25 8

helloworld

老师我有几个疑问:1. 单元测试是不是也要随着业务流程的变化而要持续维护？2. 对于变动非常频繁的业务流程是不是可以不写单元测试？因为考虑到时间的问题。3. 所有的大公司重要的项目（例如淘宝，京东等平台）是不是都有严格的单元测试编写或者执行规范？谢谢老师啦。

> 作者回复: 1和2，本质上是一个问题。其实，很多人担心写测试会影响写代码，但实际上，写测试，尤其是单元会帮助写代码，否则，你用的手工测试或系统测试的方式来保证系统的正确性。如果你不花练习写测试，你永远学不会写测试，写测试在你看来就是浪费时间。
>
> 这背后其实还隐藏着另外一个问题，为什么会变化快？因为一方面，前期的工作做得少，这是我们前面讲的内容要解决的，另一方面，设计做的不好，变化都是大变化，设计不做好，那就到处是问题，这是我们后面要讲的内容。
>
> 3，国内公司在工程上做得都不够好，如果想看好榜样，可以看看国外的公司，比如，Google，它要求100%测试覆盖率。

2019-02-26 7

苦行僧

最近深有感触 单元测试写的越多 越能反思自己的代码 内建质量也能一步步建立起来 多写单元测试真是会产生蜕变的

> 作者回复: 小病不治，会生大病的。

2019-03-29 4

🌲树根🌲

我的想法可以在复杂度高，重要核心的模块先开始写单元测试。特别是公用、底层的，因为这些靠功能测试很难覆盖。

单元测试难以推行主要是没有碰到质量的痛点，通常都依靠测试工程师来保证质量。我们之前就在遇到过质量崩塌，倒逼着我们去做，以保证质量。

作者回复: 很多人不知道质量问题是一环套一环累积起来的。

2019-02-02 4

escray

我之所以回来重读《10X程序员工作法》，一方面是因为新专栏的更新，另一方面就是在写单元测试的时候，发现自己不会任务分解，所以写不出来。

在我看来，程序员去写单元测试，而且尽可能的保持高的代码覆盖率（同时还有代码风格检查）都是不需要讨论的事情。从敏捷编程被介绍到国内，一直到现在，似乎也没有多少开发团队能够做到。不知道国外的情况怎么样?

看了文中提供的关于软件变更成本的链接，那四张图片，应该可以用来说服开发团队和领导。但是，有自驱力，愿意提高自己编程技艺的程序员可能没有那么多。

我对于测试驱动开发和单元测试都是认可的，但是一来是没有太多机会实践，二来就是真的不不会写。现在可能好一点，但是仍旧需要多练习。

准备重读 Kent Beck 的《解析极限编程》和《测试驱动开发》。

2020-06-04 2

苦行僧

多写基于方法级别的单元测试

2019-03-08 2

One day

从一个以前基本不写j测试代码的我，不过基本都有自动化测试以及测试人员，现在也开始写自己的代码的那部分测试，最开始的不情愿，到现在一直写，并且我会一直写下去，觉得这个是一种验证，每次看到自己写的代码再用自己写的测试通过之后信心满满呐

作者回复: 坚持是蜕变的基础，加油!

2019-01-30 2

肉墩儿快跑

看绩效了，只要价钱足够高，就能保证了

作者回复: 经济学告诉我们，没有足够高的价钱。

2019-01-26 2

行与修

团队开发人员的编程功力不够，即使想写单元测试也是奢望。那种前后台代码不分，用着先进的设计模式但写着落后方式实现的代码的，一旦开始了单元测试，估计大部分时间不是在实现上而是在频繁修改单元测试代码上了。

作者回复: 说得有道理，不过，在接下来的几篇，我们先把关于测试的理念梳理顺了，知道问题出在哪，以后才好进行改进。

2019-01-25 2

wesleydeng

现在单元测试有很多涉及到资源（比如db）的测试，这种情况下往往有依赖spring，导致了两个问题:
1.spring启动慢
2.dao测试不能跨环境，导致竟然因为有dao用了junit，不方便批量运行
junit这两个问题怎么破?

作者回复: 首先，涉及到Spring启动，这是集成测试，不是单元测试。
其次，Spring提供了数据库的测试方案，测试之后，可以回滚，测试之间彼此不影响。用spring和test作为关键字可以搜到。
再次，我个人推荐使用Spring Data，自己少写数据库逻辑。

2019-06-17 1

Geek_fe0336

请教老师一个具体问题，mvc结构的工程，c层的单元测试要实际的去调用m层吗？需要和数据库里的数据做比对和验证吗？如果不用，c层的单元测试应该验证或测试哪些内容呢？

> 作者回复: 在单元测试里，M 的接口是可以用 Mock 的，这才算是单元测试。涉及到数据库，就变成了集成测试。

2019-03-21 1

苦行僧

单元测试不好写 基本可以断定业务代码和通用代码纠结在一起了 需要重构

> 作者回复: 耦合是肯定的

2019-03-11 1

zapup

服务划分与编排做不好的、搞不清什么方法应该private/什么方法应该public的程序员，也会害怕写单元测试：不知道应该在哪切这一刀来看问题。

> 作者回复: 是的，良好的设计会让测试更好写。

2019-03-01 1

奕超

实际情况是：很多时候，项目时间很紧，经常会提测后，再补，或者直接code review，测试就不写

了；

> 作者回复: 项目永远很紧，时间永远不够，你打算永远这样吗？

2019-02-01 1

巫山老妖

测试没有银弹，主要看大家对测试这件事情的认知是否一致，现在都在推崇测试左移，尽可能玩发现问题，这就对程序员提出更高的要求。

> 作者回复: 优秀程序员总是少数的，会写测试就已经上了一级台阶，能在写代码之前思考测试，就会再上一级台阶。

2019-01-27 1

草原上的奔跑

作为一个程序员，怎么保证自己写出来的程序是好的，答案是写测试，只有自己通过了单元测试、集成测试、系统测试，那么提测的时候我们才会有底气，而不是时刻准备着测试出问题去改。但是，很不幸的是，团队内部成员没有写测试的意识，让他们写，以不会写、时间不够为借口，就是不写，不知道郑老师对这种情况有没有好的解决办法。

> 作者回复: 很多不喜欢写测试的真实原因是，不会写。他们先要知道写测试是怎么回事，可以把我这个系列学完之后，分享给他们。

2019-01-26 1

Better me

从高内聚低耦合的设计角度来理解测试金字塔：
单元测试更符合高内聚低耦合，一般是单个方法
服务测试一般是多个方法的组合
UI测试一般是多个服务功能的组合

系统的耦合强度越来越大，对于问题的定位也越来越难

2020-11-29

John Bull

好吧，我承认，没写单元测试，是我还不会写😄

2020-10-22

mgs2002

不是不想写单元测试，是确实不太会写。。。还有个问题写任务分解后预估工期的时候需要把写单元测试的时间也算上吗

> 作者回复: 任务分解和估算是两件事，是不是要算，自己来决定。

2020-07-17

蓝士钦

单元测试确实很有用，首先要保证在设计上方法粒度足够细，在造边界数据的时候也能让我们从更多的情况考虑问题。就算需求变动，变动后再运行一遍单元测试就能很快的排查问题。
遇到的困难是，很多场景下我们更多的是接手祖传的代码，一个方法里面写了太多逻辑，导致单元测试不好写。所以补单元测试就成了一种常态，而补的时候就需要重构代码，有的时候因为不理解某个逻辑的含义导致不好重构，所以僵持住了。我觉得一个好的项目应该有足够清晰的设计文档，让接手的人能够更加快速准确的分析出项目的模型和接口，如果没有，那么重构的人应该规范设计，让人能见代码知其意。

> 作者回复: 你说得很对！但是，程序员都太习惯于写代码了，而不习惯于写测试和文档。

2020-06-10

王维

记得以前为了赶进度，都是后面补写的单元测试，哈哈

作者回复: 聊胜于无

2020-03-17

刘冲

老师，集成测试是谁来写？测试人员么

作者回复: 谁来写都可以，无论是开发，还是测试。

2019-06-26

Amy

自从知道测试人员的测试只是基于界面的点点点，之后写的每个方法都必需写单元测试。写单元测试是对自己负责。

作者回复: 呃，这个角度好神奇，但你意识到要多写测试就好。

2019-04-27

enjoylearning

测试会提高我们的设计能力，写出clean code

作者回复: 想写好测试，要写好代码。

2019-03-28

Xunqf

目前好像大部分公司都不怎么要求写单元测试，即使有些公司要求自测最多也只是业务上的测试，就像前端和移动端很多时候都是UI还原度和适配上的问题，也只有视觉上的东西暴露的更直接一些，其他方面的问题一般都很难说清楚，一般公司业绩考核也更看重直观的东西，所以大部分人都不会去做一些底层的单元测试！

> 作者回复: 你说的是现状，我这里在讨论的是，应该是什么样。我反复提到，许多团队深陷泥潭不自知，就是现有的做事方式让他们陷了进去。

2019-01-28