高质量的软件是否值得成本？

2019年5月29日

马丁福勒

通过查看这些标签查找**类似的文章**：清洁代码 · 生产力 · 项目计划 · 技术债务

内容

侧边栏

软件开发项目中的一个常见争论是花时间提高软件质量，而不是专注于发布更有价值的功能。通常，提供功能的压力主导着讨论，导致许多开发人员抱怨他们没有时间研究架构和代码质量。

Betteridge的头条新闻是一句谚语，说任何标题或标题以问号结尾的文章都可以用“否”来概括。那些了解我的人不会怀疑我想要颠覆这样的法律。但是这篇文章比这更进一步 - 它颠覆了问题本身。这个问题假定了质量和成本之间的共同权衡。通过本文，我将解释这种权衡并不适用于软件 - 高质量的软件实际上更便宜。

虽然我的大部分写作都是针对专业软件开发人员的，但对于本文，我不会假设任何有关软件开发机制的知识。我希望这篇文章对参与思考软件工作的任何人都有价值，特别是那些充当软件开发团队客户的商业领袖。

我们习惯于在质量和成本之间进行权衡

正如我在开幕式中提到的，我们都习惯于在质量和成本之间进行权衡。当我更换智能手机时，我可以选择更昂贵的型号，具有更快的处理器，更好的屏幕和更多的内存。或者我可以放弃一些这些品质来减少钱。这不是一个绝对的规则，有时我们可以在优质商品更便宜的地方买到便宜货。更常见的是，我们对质量有不同的价值 - 有些人并没有真正注意到一个屏幕比另一个更好。但大多数时候这种假设是正确的，更高的质量通常会花费更多。

软件质量意味着很多东西

如果我要谈谈软件质量，我需要解释一下它是什么。这就是第一个复杂问题 - 有很多东西可以算作软件的质量。我可以考虑用户界面：它是否容易引导我完成我需要完成的任务，使我更有效率并消除挫折感？我可以考虑它的可靠性：它是否包含导致错误和挫折的缺陷？另一个方面是它的架构：源代码是否分为明确的模块，以便程序员可以轻松找到并理解本周需要处理哪些代码？

这三个质量的例子并不是一个详尽的列表，但它们足以说明一个重要的观点。如果我是软件的客户或用户，我不会理解我们称之为质量的一些东西。用户可以判断用户界面是否良好。一位高管可以判断该软件是否使她的员工在工作中更有效率。用户和客户会注意到缺

陷，特别是它们会损坏数据或使系统暂时无法运行。但是客户和用户无法理解软件的架构。

因此，我将软件质量属性划分为**外部**（例如UI和缺陷）和**内部**（架构）。区别在于用户和客户可以看到什么使软件产品具有高外部质量，但无法区分更高或更低的内部质量。

乍一看，内部质量对客户无关紧要

由于内部质量不是客户或用户可以看到的 - 这有关系吗？让我们想象丽贝卡和我写一个跟踪和预测航班延误的应用程序。我们的应用程序都具有相同的基本功能，两者都具有同样优雅的用户界面，并且几乎没有任何缺陷。唯一的区别是她的内部源代码整齐有序，而我的内容却是混乱的。另外还有一个区别：我以6美元的价格出售我的产品，并以10美元的价格卖掉了她的产品。

由于客户从未看到此源代码，并且它不会影响应用程序的运行，为什么有人会为Rebecca的软件额外支付4美元？更一般地说，这应该意味着不值得为更高的内部质量支付更多的钱。

我说的另一种方式是，交换外部质量成本是有意义的，但内部质量的交易成本是没有意义的。用户可以判断他们是否想要支付更多费用以获得更好的用户界面，因为他们可以评估用户界面是否足够好以至于值得多花钱。但是用户无法看到软件的内部模块化结构，更不用说判断它更好了。为什么要为没有效果的东西付出更多？既然如此 - 为什么软件开发人员应该花时间和精力来提高工作的内部质量？

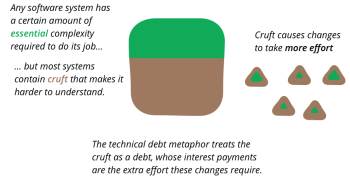
内部质量可以更轻松地增强软件

那么为什么软件开发人员会因内部质量问题而解决问题呢？程序员大部分时间都在修改代码。即使在新系统中，几乎所有编程都是在现有代码库的上下文中完成的。当我想为软件添加新功能时，我的第一个任务是弄清楚这个功能如何适应现有应用程序的流程。然后我需要更改该流程以使我的功能适应。我经常需要使用已经在应用程序中的数据，因此我需要了解数据代表什么，它与周围数据的关系以及我可能提供的数据需要为我的新功能添加。

所有这些都是关于我理解现有代码的。但是软件很难理解。逻辑可能变得纠结，数据可能难以理解，六个月前用来指代事物的名字可能对Tony有意义，但对我来说和离开公司的理由一样神秘。所有这些都是开发人员称之为**残余的形式** - 当前代码与理想情况之间的差异。

内部质量的一个主要特点是让我更容易弄清楚应用程序的工作原理，这样我就可以看到如何添加内容。如果将软件很好地划分为单独的模块，我不必阅读所有500,000行代码，我可以在几个模块中快速找到几百行。如果我们将精力放在明确的命名上，我可以快速了解代码的各个部分，而不必赘述细节。如果数据明智地遵循基础业务的语言和结构，我可以很容易地理解它与客户服务代表的请求之间的关系。Cruft增加了我理解如何做出改变所花费的时间，也增加了我犯错误的可能性。如果我发现我的错误，那么就会有更多的时间丢失，因为我必须了解故障是什么以及如何解决它。如果我没有发现它们，那么我们就会遇到生产缺陷，以及以后花更多的时间来修理。

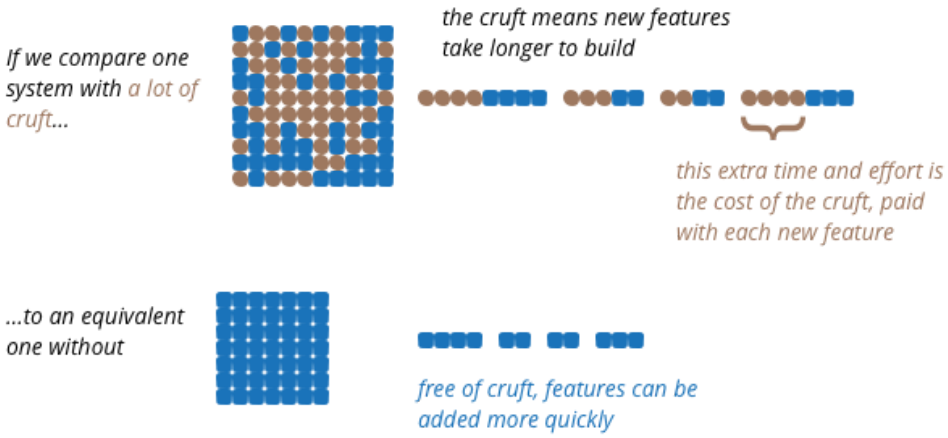
技术债务是一个常见的比喻。添加功能的额外费用就像支付利息一样。清理残骸就像支付本金一样。虽然它是一个有用的比喻，但它确实鼓励许多人相信cruft比实际更容易测量和控制。



我的变化也会影响未来。我可能会看到一种快速的方式来放入这个功能，但这是一条违背程序模块化结构的路线，增加了瑕疵。如果我采取这条道路，今天我会更快地为我做准备，但是在未来几周和几个月里，其他所有必须处理此代码的人都要放慢速度。一旦团队中的其他成员做出相同的决定，一个易于修改的应用程序可以迅速积累到每一个小小的变化需要花费数周努力的程度。

客户确实关注新功能的快速发展

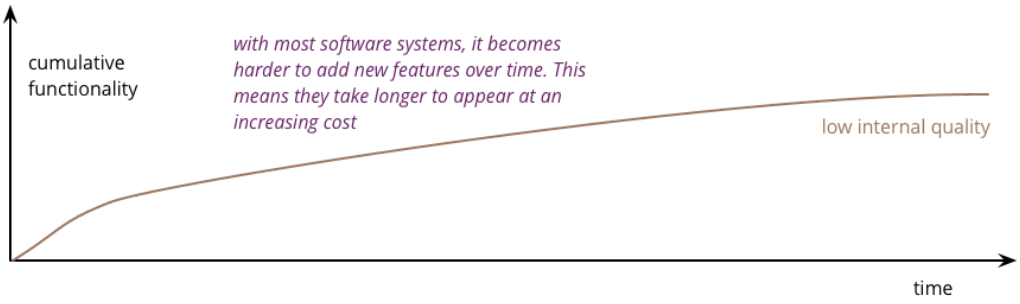
在这里，我们看到了内部质量对用户和客户至关重要的线索。更好的内部质量使得添加新功能更容易，因此更快，更便宜。丽贝卡和我现在可能有相同的应用程序，但在接下来的几个月里，丽贝卡的高内部质量让她每周都能添加新功能，而我却一直试图通过简单的方式来获得一个新功能。我无法与Rebecca的速度竞争，很快她的软件就比我的软件更具特色。然后我的所有客户都删除了我的应用程序，并获得了Rebecca，即使她能够提高她的价格。



可视化内部质量的影响

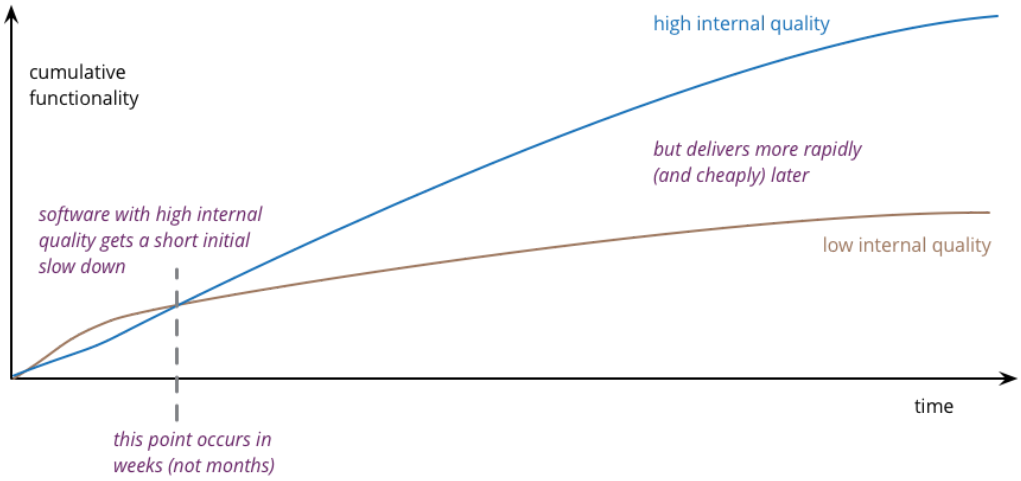
内部质量的基本作用是降低未来变革的成本。但是编写好的软件需要额外的努力，这在短期内会产生一些成本。

一种可视化的方法是使用以下伪图，其中我绘制软件的累积功能与产生它的时间（以及成本）。对于大多数软件工作，曲线看起来像这样。



这是内部质量差的情况。最初进展很快，但随着时间的推移，添加新功能变得更加困难。即使很小的变化也需要程序员理解大面积的代码，这些代码很难理解。当他们进行更改时，会发生意外断裂，导致测试时间过长，需要修复缺陷。

专注于高内部质量就是要减少生产力的下降。事实上，有些产品会产生相反的效果，开发人员可以通过利用先前的工作轻松构建新功能。这种快乐的情况是一种罕见的情况，因为它需要一支技术精湛，训练有素的团队来实现这一目标。但我们偶尔会看到它。



这里的微妙之处在于，有一段时期，低内部质量比高轨道更具生产力。在此期间，在质量和成本之间存在某种权衡。当然，问题是：线路交叉前的那段时间有多长？

在这一点上，我们遇到了为什么这是一个伪图。无法测量软件团队提供的功能。由于无法衡量产出，从而无法衡量生产率，因此无法对低内部质量的后果（这也很难衡量）提出可靠的数字。无法衡量产出在专业工作中非常普遍 - 我们如何衡量律师或医生的生产力？

我评估线路交叉的方式是通过拉长我所知道的熟练开发人员的意见。答案让很多人感到惊讶。开发人员发现质量差的代码会在几周内显着降低速度。因此，内部质量和成本之间的权衡取舍并不多。即使很小的软件工作也会受益于对良好软件实践的关注，当然我可以从我的经验中证明这一点。

即使是最好的球队也会造成残酷

许多非开发人员倾向于认为只有当开发团队粗心大意并且犯错时才会发生这种事情，但即使是最优秀的团队也会在工作时不可避免地产生一些残余。

我想用一个当我和我们最好的技术团队领导聊天的故事来说明这一点。他刚刚完成了一个被广泛认为是非常成功的项目。无论是在功能，施工时间和成本方面，客户都对交付的系统感到满意。我们的员工对该项目的工作经验持积极态度。技术领导者非常高兴，但承认系统的架构并不那么好。我的反应是“你怎么可能 - 你是我们最好的建筑师之一？”他的答复是任何一位经验丰富的软件架构师都熟悉的答案：“我们做出了很好的决定，但现在才明白我们应该如何构建它”。

许多人，包括软件行业的一些人，将构建软件比作建造大教堂或摩天大楼 - 毕竟为什么我们为高级程序员使用“架构师”？但构建软件存在于物理世界未知的不确定世界中。软件的客户只是粗略地了解他们在产品中需要哪些功能，并在构建软件时了解更多信息 - 特别是一旦早期版本发布给用户。软件开发的构建模块 - 语言，库和平台 - 每隔几年就会发生重大变化。物理世界中的等价物是客户通常在建造和占用建筑物的一半时添加新楼层并更改楼层平面图，

鉴于这种程度的变化，软件项目总是创造出新颖的东西。我们几乎没有发现自己正在研究一个之前已经解决的易于理解的问题。当我们构建解决方案时，我们自然会了解到这个问题，因此我常常听到团队只有在花了一年左右的时间构建它之后，才能真正理解软件的架构。即使是最好的团队也会在他们的软件中肆意妄为。

不同的是，最好的团队创造了更少的残余，但也消除了他们创造的足够的残余，他们可以继续快速添加功能。他们花时间创建自动化测试，以便他们能够快速解决问题并减少浪费时间。他们经常进行重构，以便在它们积聚足以阻挡之前将它们移除。由于团队成员在交叉目的下工作，持续集成可以最大限度地减少残留。一个常见的比喻就是清理厨房的工作台面和设备。你做饭时不能弄脏东西，但是如果你不快速清理东西，淤泥干涸，更难去除，所有肮脏的东西妨碍了烹饪下一道菜。

多拉研究精英团队

质量和速度之间的选择并不是软件开发中唯一具有直观意义的选择，但却是错误的。还有一个强烈的思路，即在快速开发，频繁更新系统和可靠的系统之间存在Bi Modal选择，这些系统不会中断生产。这是一个错误的选择，通过开发状态报告中的认真科学工作得到证实。

几年来，他们使用调查的统计分析来梳理高绩效软件团队的实践。他们的工作表明，精英软件团队每天多次更新生产代码，在不到一个小时的时间内将代码更改从开发变为生产。当他们这样做时，他们

高质量的软件生产成本更低

总结所有这些：

- 忽视内部质量会导致快速建立起来
- 这种错误减缓了功能开发
- 即使是一个伟大的团队也会产生残余，但通过保持内部质量，可以控制它
- 高内部质量使残留物降至最低，使团队能够以更少的工作量，时间和成本添加功能。

可悲的是，软件开发人员通常不会很好地解释这种情况。无数次我和开发团队谈过，他们说“他们（管理层）不会让我们写出质量好的代码，因为它需要太长时间”。开发人员通常需要适当的专业性来证明对质量的关注。但是，这种道德主义的论证意味着这种质量是有代价的 - 这使他们的论点失败了。令人讨厌的是，由此产生的苛刻的代码既使开发人员的生活更加困难，又花费了客户的钱。在考虑内部质量时，我强调我们只应将其视为经济论证。高内部质量降低了未来功能的成本，

这就是为什么以本文为首的问题忽略了这一点。高内部质量软件的“成本”是负面的。成本和质量之间的通常权衡，我们习惯于生活中的大多数决策，对软件的内部质量没有意义。（它用于外部质量，例如精心设计的用户体验。）因为成本和内部质量之间的关系是一种不同寻常和反直觉的关系，所以通常很难吸收。但了解它对于以最高效率开发软件至关重要。

的更改失败率明显低于慢速组织，因此他们可以更快地从错误中恢复。此外，这些精英软件交付组织与更高的组织绩效相关联。

重大修订

2019年5月29日： 出版