# 14 | 质量与速度的均衡：让“唯快不破”快得更持久

2019-09-23 葛俊



**讲述：葛俊**
时长 14:05　大小 12.90M

你好，我是葛俊。今天，我来和你聊聊团队可持续性的快速开发，怎样才能让“唯快不破”快得更持久。

最近几年，一提到开发，很多人想到的都是“天下武功，唯快不破”。也就是说，开发过程越快越好，越快越有竞争力。这的确是软件开发，尤其是互联网行业软件开发的不二法则。也正如我在前面文章中多次提到的，快速开发可以快速得到用户反馈，更快地验证用户价值假设。无疑，这是高效开发的重要原则。

因此，我们在实际工作中，往往会为了快而选择各种“捷径”。比如：

要开发已有功能的一个相似功能，因为时间很紧就先 copy & paste，保证功能按时上线。

> 需要在一个函数里增加功能，这个函数已经有 800 行了，加上新功能后会有 1000 行。重构这个函数是来不及了，先把功能加上去再说。

说是“捷径”，是因为这些都不是最优解，有点儿投机取巧。它们的确能让我们在短期内保证快速交付，满足业务发展需求。但如果没有任何补救措施的话，时间长了我们就再也快不起来了。

比如，“Copy & Paste”方式的编程，会导致后续添加功能时，需要在很多地方做类似修改，工作量大且容易出错。再比如，无视函数变大的操作，会导致后续的修改、调试异常困难。

这些问题都会成为开发工作中的**技术债**，也就是**在开发产品或者功能的过程中，没有使用最佳的实现方法而引入的技术问题**。无疑，这些技术问题会为将来的产品维护和开发带来额外开销。只有正确地处理技术债，才能让我们的研发持续地快下去。

接下来，我们就来看看技术债的成因、影响，以及对应的处理方法。

## 技术债的成因

从成因来看，技术债的引入包括主动和被动两种。

> 主动引入，即开发人员知道某一个实现会产生技术债，但仍采用这样的实现。最常见的情况是，由于业务压力，在时间和资源受限的情况下不得不牺牲质量。
>
> 被动引入，即不是开发人员主动引入的技术债。常见的情况有两种：一是，产品不断演化，技术不断发展，原来的设计、实现落伍了；二是，开发团队的能力和水平有限，没有采用好的开发方法、实践。

所以说，技术债是无法避免的，我们要做的就是明确它的影响、处理好它。

## 技术债有哪些影响？

提到技术债，我们想到的往往是它的坏处，比如难以维护、难以增加新功能等，但实际上它也有巨大好处。

关于技术债的好处，我们可以对应着金融领域的经济债务来理解。我们都知道，经济债务最明显的好处在于，可以帮助我们完成很多本来不可能完成的任务，比如贷款买房。相应的，

技术债可以在短期内帮我们快速完成业务开发，满足用户需求，就类似房贷的作用。

但跟经济债务一样，技术债也需要偿还，也会产生利息，而且是利滚利。也就是说，每一步累积的技术债都会叠加起来，为开发增加越来越大的难度。长期来看，如果一直借债不还，开发新功能会越来越慢，产品维护越来越难，甚至是无法维护必须推到重来，就像还不上房贷房子被银行收回一样。

那么，技术债务应该如何处理、如何偿还呢？

## 处理技术债的基本原则是什么？

在我看来，处理技术债的基本原则有以下两个方面。

**第一方面，要利用技术债的好处，必要时要大胆“举债前行”**。也就是说，在机会出现时，使用最快的方式完成业务服务用户，抢占市场先机，“不要在意那些细节”。

一个具体的例子是，RethinkDB 在与 MongoDB 的竞争中失利。在技术上，RethinDB 比 MongoDB 更追求完美，但比 MongoDB 发布稳定版本晚了三年，错过了 NoSQL 的黄金时机，最终在 2017 年 1 月份宣布破产。在这个过程中，他们没有充分利用技术债抢占市场，应该是竞争失败的一个重要原因。文章中我放了两个链接供你阅读参考，你可以了解一下 RethinkDB 公司的人以及外部用户对他们的失败进行的反思文章 1、文章 2。



**第二个方面，要控制技术债，在适当的时候偿还适当部分的技术债。**

在我看来，国内大部分公司的业务驱动做得比较好，大都能够比较充分地利用技术债的好处，但在技术债的管控方面，通常做得不太够，具体来说就是常常有大量技术债堆积，给业务长期发展带来巨大阻碍。

所以在下面的内容中，我会与你详细讲述应该怎样控制技术债。

## 如何控制技术债？

从我的经验看，控制技术债主要有以下 4 步：

1. 让公司管理层意识到偿还技术债的重要性，从而愿意投入资源；
2. 采用低成本的方式去预防；

3. 识别技术债并找到可能的解决方案；
4. 持续重构，解决高优先级技术债。

接下来，我们分别看看这 4 步具体如何实施吧。

### 1. 让公司管理层意识到偿还技术债的重要性，从而愿意投入资源

通常来说，开发人员能直观感受到技术债的坏处，大都愿意去偿还技术债，所以技术债累积的主要原因是，管理层不理解，或者说是没有认识到技术债累积给业务发展带来的巨大坏处。

这也就意味着，解决技术债的第一步就是，让管理层意识到偿还技术债的重要性，从而愿意投入资源去解决。在我看来，让管理层理解技术债比较直观、有效的方式，就是上面提到的与经济债务的类比。

另外一个办法是，将偿还技术债与业务发展联系起来。如果能够说明某一项技术债已经阻碍了公司重要业务的发展，说服管理层投入资源解决技术债就会比较容易。

### 2. 采用低成本的方式预防

所谓具体问题具体分析，我们在预防技术债时，也需要根据技术债的成因采取不同的措施。

对主动引入的技术债，要尽量让管理层和产品团队了解技术上的捷径将会带来的长期危害，从而在引入技术债时客观地权衡其带来的短期收益和长期损害，避免引入不必要的技术债。

在被动引入的技术债中，由于产品演化导致设计落伍的问题不是很好预防。而由开发团队的能力问题引入的技术债，我们可以使用加强计划和代码审查等方法实现低成本的预防。

其中，加强计划，可以帮助开发人员更合理地安排时间，从而有相对充裕的时间去学习并选择更优秀的功能实现方案。而代码审查的作用就更好理解了，它可以帮助我们在早期发现一些不必要引入的技术债，以更低的成本去解决它。

关于技术债的预防，我还有一个小贴士，就是在接口部分多下功夫。因为接口涉及实现方和多个调用方，所以接口部分累积的技术债，影响范围通常比较大。而与之相对应的模块内部实现，技术债的影响范围就比较小。所以，在涉及主动引入的技术债时，我们需要区别对待接口部分和实现部分。

### 3. 识别技术债并找到可能的解决方案

对不能预防的技术债，我们需要高效地把它们识别出来，并了解常见的解决办法。其中，对于主动引入的技术债，可以在引入的时候就添加任务到 Backlog。而对于被动引入的技术债，则需要周期性的审视，这需要技术管理者主动地收集、整理技术债问题。

总结来说，技术债可以分为两大类：复杂度相关和重用性相关。我们可以关注这两个方面来识别技术债。

**第一是，复杂度相关。**

史蒂夫 · 迈克康奈尔（Steve McConnell）在其经典著作代码大全（Code Complete）中，提出的一个核心观点是：**如何处理复杂度是软件开发最核心的问题。**我非常认同这个观点，因为人类大脑容量有限，大概只能同时记住 7 项内容，而软件包含的元素非常复杂远超过 7 项。所以，要实现可维护的软件，我们必须想尽办法去降低其复杂度。



具体来说，我们在开发时，要时刻注意会增加代码复杂度的“坏味道”，比如：

- 组件间依赖混乱，职责不清晰；
- 组件、文件、函数太大，包含的内容太多；
- 使用不必要的、复杂的设计范式；
- 函数、接口参数太多等。

**解决复杂度问题的基本原则是，把一个系统拆解为多个子系统，用抽象和分层的方法，让我们同时只面对有限的信息，并且能够有条理地深入到每一个子系统中查看细节。**具体的解决方法有：

- 对系统进行二进制组件或者代码层面的解耦；
- 使用简单化的设计编码原则，避免不成熟的优化；
- 对常见的代码“坏味道”做出一些规范，比如限制代码行的长度、禁止循环依赖、限制圈复杂度（Cyclomatic complexity）；
- 对复杂的设计添加注释。

**第二是，重用性相关。**

软件开发的另一个重要原则是 DRY，即 Don’t repeat yourself。代码重复是一个很常见的技术债，在软件抽象的各个层次（比如应用、架构、组件、代码）都会出现。避免重复的具体方法有：

应用层面，复用业务单元，典型案例就是业务中台；

架构层面，复用基础设施后台；

组件层面，避免出现责任重叠的组件、数据存储等；

代码层面，避免出现重复函数、代码块。

接下来最后一步，就是要持续性地重构，去解决高优先级的技术债任务。

### 4. 持续重构，解决高优先级的技术债

作为技术管理者，除了业务目标外，还要制定团队的技术目标，来解决最重要、最紧急的技术债任务。

技术债任务的具体处理方法有两种：一种是，把技术债的任务和业务相关的任务放到一起，在每一个迭代中持续完成；另一种方法是，采用突击的方式，在某个特定的时间段集中解决技术债问题。

比如，我在 Facebook 和微软的时候，我们团队就都使用过 Bug Bash 的工作方式，也就是在每几个迭代以后，专门花几天时间来解决前面遗留下来的 Bug，而不开发新功能。这样做的好处有两个：

第一，集中精力修复 Bug 可以减少上下文切换，能够更聚焦在提高产品质量上，因为提高质量和写新功能的思路是有区别的。

第二，能够让团队成员短暂地从紧张的业务气氛中脱离出来，从而精力充沛地投入到下一个业务开发迭代中去。

## 小结

在今天这篇文章中，我与你介绍了要想让开发工作能够持续地快下去，正确的做法是在恰当的时间“举债前行”，而在平时的开发工作中要持续定位技术债任务，并解决高优先级的部分。

为了帮助你理解技术债与公司业务发展的关系，我再和你分享一个案例。A、B、C 三个公司对待技术债的态度分别是：

A 公司：只关注业务，不偿还技术债；

B 公司：持续关注技术债，但对业务时机不敏感；

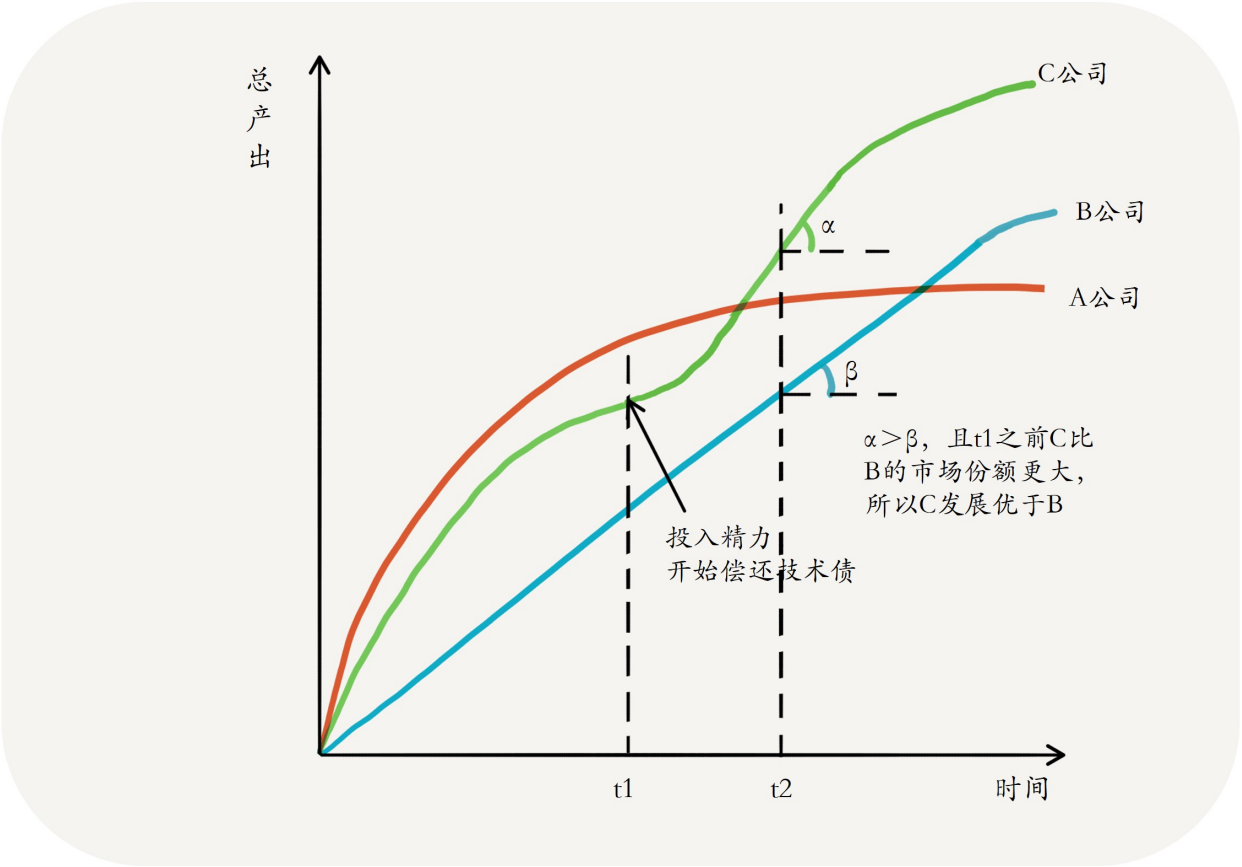
C 公司：持续关注业务和技术债。对业务机会很敏感，敢放手一搏大量借贷，也知道什么时候必须偿还技术债。

A 公司在开始的时候，业务产出会比较多，但由于技术债带来的影响，效率会逐渐降低。

B 公司在开始的时候，业务产出比较少，但由于对技术债的控制，所以能够保持一个比较稳定的产出，在某一时间点超过 A 公司。

C 公司在有市场机会的时候，大胆应用技术债，同时抽出一小部分时间精力做一些技术债预防工作。这样一来，在一开始的时候，C 的业务产出输出介于 A 和 B 之间，但和 A 的差距不大。

随后，在抢占到一定的市场份额之后，C 公司开始投入精力去处理技术债，于是逐步超过 A。另外，虽然 C 公司此时的生产效率低于 B 公司，但因为市场份额的优势，所以总业绩仍然超过 B。并在高优先级技术债任务处理好之后，生产效率也得到了提升，将 B 公司也甩在了身后。




这个例子很有代表性，你可以用它来说服管理层在偿还技术债上做投入。

## 思考题

经济债务可以申请破产保护，你觉得技术债可以有这样的福利吗？为什么呢？

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再见！




上一篇　13 | 代码审查：学习Facebook真正发挥代码审查的提效作用

## 精选留言 (1)

写留言

**李双**
2019-09-23

敢于留债，并定期偿还技术债！

展开

1