19 | 怎么避免过度设计?

2019-02-15 范学雷 来自北京

《代码精进之路》

俗话说，“过犹不及”。“过度”这个词仿佛会给我们一些不好的暗示。不要紧张，我们先聊一个轻松的话题。

假设有一个小地方，要建一个火车站。这个地方有数十万人口，每列火车预计上下乘客数十人，高峰时段大概近百人。你会怎么设计这个火车站?

这个火车站可能是个富丽堂皇的建筑，有宽敞的售票厅和候车室。这种设计到处可见，你可以想一想你熟悉的火车站， 也可以观察一下旅途中的火车站。

也有些火车站可能只是一个一百平米左右的小房子，只有简单的售票窗口、进站口和出站口。比如说北京的清华园火车站，就是这样的。

也有的火车站只有标牌、售票机和遮阳棚的一小块地方，告诉人们火车在这儿停靠，就像我们常见的公交车站。

这三种火车站，都能实现旅客购票、候车、上车和下车的核心需求，帮助他们实现乘车旅行的目的。

既然乘坐火车的核心需求基本是一样的，为什么车站的差别这么大呢？

乘车旅行这个需求，衍生出了购票、候车、上车和下车的需求。

购票的需求衍生出了售票、购票、验票、检票以及各个环节排队的需求。

售票的需求衍生出了要有售票办公室和售票大厅、管理售票人员、购票人员和票贩子的需求。

售票办公室衍生出了科长办公室、科员办公室、会议室、售票窗口。售票窗口的需求也可以接着衍生出更多的需求。这个列表我们可以列很长很长，最后的结果就是火车站的建设耗资大，建设周期长，运营成本高。

哪一种火车站对旅客更方便呢？如果在一个小地方，那么第三种火车站旅客上车的环节最少，是最方便的。而且投资小，建设周期短，运营成本低。

软件开发和建火车站一样，都有设计、建设、运营和维护的环节。该怎么管理好需求和设计，是工程设计者需要重点考虑的问题。

避免需求膨胀

软件开发过程中，最让人痛苦的是什么？如果有这么一个调查的话，“频繁的需求变更”应该是一个高票选项。

频繁的需求变更确实让人抓狂。它变更的可不仅仅只是需求，还有不断重构的代码，不断延长的工期，不断增长的投入，以及越来越多的加班。

在一个五彩缤纷的世界里，拥有多种多样的观点，坚持不懈地改进，是一件好事情。但是，“多姿多彩”对于计算机程序而言，就是个巨大的挑战。现实世界需要丰富，而抽象程序则需要简化。这对不可避免的矛盾，就成了所有程序员的心头刺。

软件是为现实服务的，而现实总是变化的。作为程序员，我们是没有办法抵制住所有的需求变更的。为了限制无节制的需求变更，适应合理的需求进化，我们要使用两个工具，一个工具是识别最核心需求，另一个工具是迭代演进。

识别最核心需求

一个经济的系统，需要从小做起，而不是一上来就胡子眉毛一把抓，什么都要做。什么都要做的结果是什么都做不好。

要从小做起，最重要的就是选择。什么是必须做的？什么是现在就要做的？这是我们做选择时，要时刻准备提出和回答的两个问题。

回答这两个问题，有时候并不容易。我们知道的越多，见识越广，这两个问题越难回答。比如说开头中提到的火车站的建设。既然建造公交车站一样的火车站又方便、又省钱，为什么还要建造富丽堂皇的火车站呢？岂不是又费事又费钱？

但是，专家有他们的考量。逃票问题、安全问题、舒适问题、管理问题、就业问题等，都在他们的考虑范围内。

作为程序员，或者项目经理，我们懂得一大把的原理，学了一大把的技术，手里有一大把工具。这些技术运用起来，就是一个丰富的大世界。我们的很多需求，来源于心里的推断，而不是眼前的事实。推断产生需求，催生的系统就会形成新的事实，强化推断的演进。为了解决了不存在的问题，我们制造出真实存在的问题。

我第一次见到像公交车站一样的火车站时，心里想，这也算火车站吗？多多少少有点震惊。我真的没有见过这么简单的火车站。有一段时间，我每天都要经过这个车站，也没发现什么不妥的地方。只要提前 30 秒到达火车站，就能赶上准时出发的火车，像坐公交车一样很方便。我之所以觉得它方便，因为我是乘客。

如果从最终用户的眼里看软件，类似于从乘客的眼里看火车站。很多软件，承载了太多中间客户的期望和推断，最终用户的真实需求和关键需求反而被膨胀的无效需求弱化了。

所以，我们要回归到最终用户。只有从最终用户的眼里看需求，才能够识别什么是最核心的需求，什么是衍生的需求，什么是无效的需求。这样，我们才能找到一个最小的子集，那就是现在就必须满足的需求。

首先就必须满足的需求，是优先级最高的、最重要的事情，这些事情要小而精致，是我们的时间、金钱、智力投入效率最高的地方，也是回报最丰厚的地方。我们要把这些事情做到让竞争对手望尘莫及的地步。

不要一步到位

有一些需求很重要，但不是现在就必须做的。这就需要另外一个方法——迭代演进。第一次我们没有办法完成的事情，就放在第二次考虑。

迭代演进不仅仅需要考虑上一次没有完成的事情，还要考虑变化促生的新需求。所以，在这一步，还要像第一次一样，先找到最小的子集，也就是现在就必须满足的需求。然后，全力以赴地做好它。

这样迭代了几轮之后，一定有一些第一次看起来很重要的需求，再看反而不重要了，根本就不需要解决。

在 OpenJDK 社区中，每年都会关闭一些有些年头的需求请求。这些需求，要么没有真实用户，要么已经有了替代的解决方案，要么就是已经被抛弃的技术。所以一些曾经看起来值得考虑的需求，时间为我们过滤掉了它们。

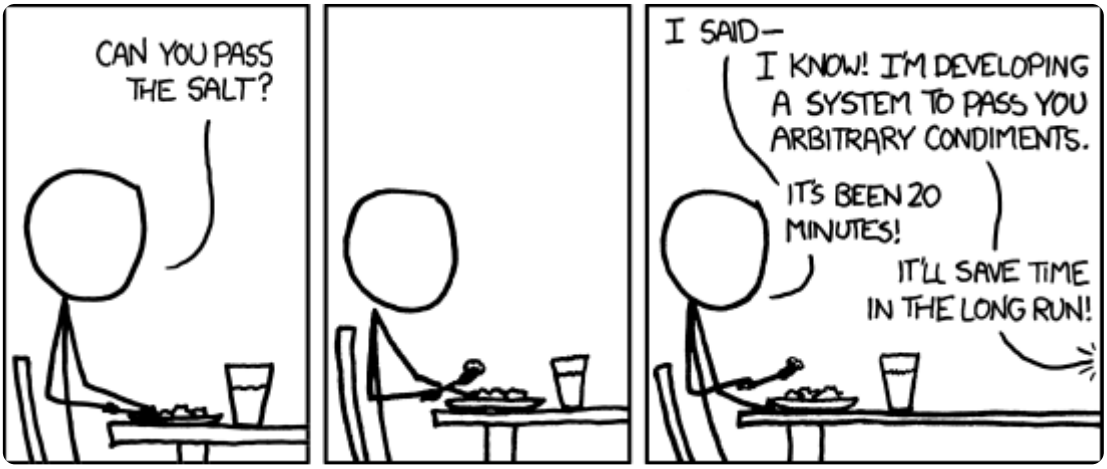
是不是迭代的时候，就可以考虑一些不重要的需求了呢？不，永远不要考虑不重要的需求。有时候，遏制住添加新功能、新接口的渴望，是一个困难的事情。我们需要学会放手，学会休假，以及拥有空闲时间。

管理好需求，是提高我们的工作效率以及软件效率最有效路径。但遗憾的是，我们不是总有机会决定软件需求的范围，以及优先顺序。

幸运的是，我们是产品的设计者和生产者，代码该怎么写，我们有很多话语权。

避免过度设计

其实和需求一样，设计也是一个容易膨胀的环节。 看看下面的漫画，是不是有些好笑又熟悉？我们只是需要一点盐，设计师会设计一个能给我们任何调味品的接口。设计接口系统会耗费很多时间，但设计师认为这会节省我们未来的时间。



遗憾的是，对软件来说，过度设计的接口意味着更多的代码、更多的维护、更多的修修补补，未来也不会节省我们的时间。

费迪南德·保时捷曾经说过：“一辆完美的跑车，应该首先越过终点，然后立即陷入困境。”这多少有点苛刻，但这就是“少就是多”的极简主义追求。

过度设计导致过度复杂，过度复杂导致效率降低、危险加剧、性能降低。如果保持简单而不是复杂化，大多数系统都能发挥最佳作用。这就是“少就是多”的魅力。

避免过度设计，和避免需求膨胀一样，我们要时刻准备提问和回答的两个问题：什么是必须做的？什么是现在就必须做的？

这两个问题时常提问、经常回答，有助于我们始终在用户的需求范围内思考设计，有助于我们始终关注核心问题，并且保持设计方案的简介、优雅。

小结

影响代码效率的最重要的两件事情，就是需求的膨胀和过度的设计。为了这两个问题，我们需要回答两个问题：

1. 什么是必须做的？
2. 什么是现在就必须做的？

弄清楚这两个问题后，我们需要做的，就是做好现在就必须做的事情。

一起来动手

克制住过度设计的倾向，这需要非凡的自律和自信。有时候我就想，微信的团队到底是怎么克制住自己，让微信简洁的页面保持了这么多年。那么多的诱惑，那么多流量变现的办法，都能抵制住，得要有多强大的内心和清醒的认识！

微信的聊天页面是我们最关心的信息：谁发送了信息。一对一的聊天界面，永远只使用窄窄的一行，来完成丰富的功能，红包、语音、表情包、贴图，都可以在这一行完成。所有的其他功能，比如小程序，朋友圈、合作商家，都不能干扰核心功能的呈现。现在我们看着可能觉得很简单，其实这样的设计真的很难，真的很了不起。如果不相信的话，我们来做一做练手题。

这一次的练手题，我想请你思考一个银行账户管理 App，有哪些必须要做的事情。作为一个用户，你最关心的账户管理内容是什么？然后，你看下常用银行的 App，看一看你最关心的内容，需要多少步操作才可以获得，也想一想哪一些内容你会毫不犹豫地删掉。

欢迎你在留言区留言，分享你的看法。也欢迎点击“请朋友读”，把这篇文章分享给你的朋友或者同事，一起交流一下。

精选留言 (24)

Demon.Lee

2019-02-16

想了想，打开银行app。首先想看到总余额以及每张银行卡里的余额，第二，汇款功能，第三费用明细。

> 作者回复: 嗯，这些是核心的用户需求。不幸的是，很多App并没有第一时间、用最便捷的方式满足这些需求。

共 3 条评论 >　　11

Ricky
2019-03-17

现在的很多银行APP和电信运营商的APP做的真的是够了，一大堆花里胡哨的东西，真正想要的使用高频的功能都特别难找。

> 作者回复: 不知道给谁用的😂

共 2 条评论 >　　7

卞雪达
2019-05-28

哈哈哈，我经常有过度设计的冲动，有时候忍不住就实施了出来。冲动的原因可能是想试试新的技术，包括大小轮子、各种优化、编程思想等等。但是更多的原因是项目某些时候给与的时间较充裕，不折腾一下总觉得没发挥，我面向对象、接口、切面、组件等编程思想，都是在这个过程中学习的。有时候弄出来发现其实过度了，又往下砍。

> 作者回复: 工具一定要掌握一大把，还要不停的折腾它们，这就是学习啊。但是产品，要把它变成折腾后的东西，而不是折腾中的东西。折腾，需要充裕的时间。我们的很多设计和实现，都是反复折腾、反复砍减后的结果。

 　　4

悲劇の輪廻
2019-03-07

某些银行的客户端已经奔着150M+去了……我怀疑他们的开发人员是不是也经过层层外包，根本不考虑客户终端的运行环境(#笑哭

共 2 条评论 >　　3

Sisyphus235
2019-05-23

银行管理 APP 必要功能：
1.查看余额
2.转账
3.收支明细
4.消息通知
5.账户管理
6.方便的查询

2

ifelse
2022-07-22

小结
影响代码效率的最重要的两件事情，就是需求的膨胀和过度的设计。--记下来

刚毅坚卓
2022-04-28

存取钱、身份验证、余额

aoe
2021-12-14

银行App主要功能是广告，查余额只是吸引用户来使用，所以在启动页就能看到广告

作者回复: 使用它来干什么呢？看广告吗？

共 3 条评论 >

进化菌
2021-11-27

过度设计，让我想起过去领导总喜欢为后面的发展设计一些细节，比如数据库多留几个用不上的一段，比如代码预留未来会用到的功能……有未来思想是极好的，但是在工期紧的时候，让人很烦，甚至那些用不上的东西不仅增加不必要的风险，还真的可能几年都用不上。

作者回复: 未来思想，恐怕我的理解不是这样的。代码里现在用不到的东西，必须干干净净地删掉，减碳又环保。

集团军群

2021-08-11

主功能没有过度设计，只是加了很多广告，商业，运营设计，人家不是慈善，要恰饭的好嘛。

作者回复: 没明白什么意思。

共 2 条评论 >

williamcai

2021-04-09

关心账户有哪些卡，每张卡的余额，转账功能，信用卡待还金额，还有明细

慎独明强

2020-06-27

老师讲到这个深有感触，目前我负责系统经历过很多迭代，未来维护成本越来越大，回归到整个系统最核心需求，中途又衍生了很多需求，也有部分需求都无效。未来如果需要重构，首先就应该考虑对功能做减法，更进一步思考最核心的诉求，解决这个问题。 在一个项目，自己可能会考虑以后更好拓展，但同时也增加了代码量，效率不是很高。再比如举例说：怎么保证消息可靠性，如果100%保证，那么系统非常复杂复杂，设计过程中应该要考虑未来拓展，不然可能会是自己的瓶颈，但如何不过度，过度的可量化指标又是哪些呢

Michael ♡YZY

2020-03-24

如果看看CBA的app设计 堪称典范 国内的一些移动应用平台都在仿效支付宝或者淘宝的设计风格 大而全 用户经常在导航到想要的功能前迷了路 而且不断叠加的广告穿插吸引注意力 导致简单的一个操作最终变成和各种失焦的信息搏斗的过程

KEEP

2019-12-08

所有复杂的软件系统都应该是迭代演化而来的，设计需要遵从简单原则。在每一个迭代里面，只做重要而必须做的事，实现则是越简单越好。一切都要刚刚好。先解决生存问题。随着时间的推移，多个简单的实现渐渐的组合出复杂的系统，根据债务的严重性，再进行安排调整，架构重构。一口吃不成胖子，成了胖子后就该减肥了……

xavier
2019-04-18

避免过度设计的意识，我想大多数人都有。但是真正能实践起来，太难了。
有时候在支付宝APP里找个功能都要找半天， 最近用了招行APP，又开了眼界。

作者回复: 嗯，是一个非常难的事情！在设计的世界里，做复杂的事很简单，做简单的事情很复杂！

共 2 条评论 >

hhhh
2019-03-15

从火车票到银行啦

徐题
2019-03-05

很认同，然而大部分的需求就是不合理的怎么办

IamNigel
2019-03-05

对于银行app我最想看到钱，可以转账，可以管理我的银行卡信息，最近两年在用平安银行的手机app,看余额得输入密码，这个可应该是安全考虑，但其中的很多功能都让人从来不会去点，特别是任意门，一不小心就点上去了。好的地方也有，像转帐以后会记录我最近转过的信息，也是比较方便

风清扬笑
2019-03-03

话说第一眼看到钱这个需求貌似是很多人想要的，但是我觉得有部分原因也是基于安全考虑，一些app设计里把余额放到二级菜单里，而且想看的话还得输入密码

作者回复: 不登录有什么必须要看的吗？现代的App，登录很方便的，比如指纹，刷脸。

唐名之

2019-02-18

登录用户 绑卡 账户余额 收支明细 转账

作者回复: 绑卡是一个一次性的事情，非常重要但是使用频率不高。这个可以通过设计解决，你有什么好办法吗?