

## 40 | 最佳实践：小团队如何应用软件工程？

2019-06-04 宝玉 来自北京

《软件工程之美》



你好，我是宝玉。经过前期理论知识的学习，你可以开始尝试在实践中去应用所学到的软件工程知识了。

想象一下，现在你要加入一家创业公司，从头组建一个开发团队，一开始只有三五个人，你会怎么去应用软件工程的知识，让你的团队能高效率、高质量地开发软件产品？

或者说，你现在就是在一个小团队，各种流程不规范，开发效率低，软件产品质量不高，你打算怎么应用学到的知识去改善现状呢？

在这一篇里，我将带你一起运用学过的软件工程知识，看如何在小团队中应用软件工程？（在这里我补充说明一下：本文讨论的小团队，不是指大厂的一个小组，而是小公司或者三五个人的小开发团队）

### 小团队在软件开发中存在的常见问题

不知道你有没有在小团队工作的经历，如果有的话，建议你可以自己先总结一下：小团队在软件开发中存在的一些常见问题是什么？

为什么说你需要先自己去发现问题呢？因为在学习完软件工程的理论知识后，并不是说你把所有知识点一股脑儿全应用上就解决问题了，而是要先去发现问题在哪，然后针对这些问题，再去应用软件工程的知识去寻找问题的解决方案。

就像小团队如何应用软件工程这个问题，你首先要先找出来小团队的问题在什么地方，然后去分析这些问题可以应用软件工程的哪些知识，从而找到适合你的解决方案。

我个人有过一段时间的小团队工作经历，也见过很多类似的小团队的开发，就我的经验来说，小团队在软件项目开发上，主要问题体现在以下几个方面。

## **1. 小团队成本敏感**

首先，小团队对成本都很敏感，成本是小团队很多问题的根源，对成本的控制也衍生出一系列大公司可能感受不到的问题。

因为控制成本，所以开不出好的薪水，难招到优秀的程序员；因为控制成本，所以进度都催的紧，毕竟多干一天就要多发一天工资；因为控制成本，舍不得在工具上的投入，都得要尽量用免费的、开源的；因为控制成本，通常几个项目并行，一个人可能要同时在几个项目中切换。

## **2. 小团队人少活多**

小团队人一般不会多，但是活不一定少。

从分工上来说，通常在大厂前端后端几个人合作完成的事，在小团队就得一个人从前端写到后端了，可能甚至都不会有专业的产品设计和功能测试人员，都是开发兼任。

从人员构成来说，大厂在组建技术团队时会注意梯队的搭配，整个团队像金字塔的结构，顶部有几个特别资深的开发人员，中间有一些经验丰富的，底部的是有潜力但经验比较少的。而小团队就算是运气好，也可能只有一两个技术大牛，更多的是水平一般、经验比较少的。

这样的分工协作和人员构成，导致的问题就是大家每天都很忙，但是感觉技术上积累有限。对个别技术大牛的依赖性强，他们一旦离职，影响非常大。

### 3. 小团队缺少流程规范

在流程规范方面，恐怕是大家对小团队吐槽最多的地方，也是很多从大厂跳槽到小公司的程序员特别不适应的地方。

项目开发比较随意，拿到需求可能就开始直接写代码了，没有严格的需求分析、架构设计，写完了后简单测试一下就上线了，上线后再修修补补；需求变更是家常便饭；多个项目并行的时候，每个项目的负责人都觉得自己项目是最重要的，希望你能把他的项目进度往前赶一赶；老板权力很大、想法多变，经常会直接干预项目。

这样不规范的开发流程，导致的结果通常就是开发效率低下，软件产品质量不高，项目计划难以遵守甚至没有计划。

### 小团队如何应用软件工程？

成本敏感、人少活多、缺少流程规范，这几个是小团队在项目开发中存在的主要问题。那么在小团队应用软件工程的时候，我们就需要去解决好这些问题。

成本敏感的问题，如果这个是客观存在的，就没有太好的办法去解决，只能说我们在做一些决策、制定流程的时候，需要充分考虑好成本因素，减少浪费。

人少活多，那么我们就相应地提升个人和团队的整体水平和效率。缺少流程规范，那么我们就建立适合小团队特色的流程规范，让开发流程规范起来。

所以接下来，我就从团队建设、流程建设这两个维度来谈谈如何应用软件工程。

#### 1. 团队建设

也许你会觉得好奇，软件工程的各个知识点，都是在讲过程、方法、工具，似乎都没有讲人的，但为什么在实践的时候，反而最先考虑的却是团队建设？

但你要换个角度想就很容易理解了：软件工程上讲的所有的过程、方法和工具，最终还是落在人身上，需要人去基于开发过程去制定流程遵守流程；需要人去应用软件工程中总结好的方法；需要人去使用工具。如果团队对软件工程缺少认识，那再好的方法和工具也无法落地。

所以要实施好软件工程，也要同步做好团队建设，让你的团队有一点基础的软件工程知识积累，有几个技术骨干可以帮助一起推广和实施。如果对软件工程知识的推广能扩大到团队之外，比如你的老板和业务部门，那么在后续推进一些流程规范，会起到事半功倍的效果。

团队建设，绕不开几件事：招人、培养人、管理人和开除人。

### 小团队如何招人

小团队招人，难点在于成本有限，开不出很高的工资，品牌也不够吸引人，招人的时候相对选择有限，能否直接招到技术大牛就得看运气了。**但这不意味着就要大幅降低标准，比较现实的方法就是招有潜力的程序员培养。**

那么怎么知道候选人是不是有培养潜力呢？可以参考我们专栏《[@27 | 软件工程师的核心竞争力是什么？（上）](#)》这篇文章，考察候选人的学习能力、解决问题能力。

我以前在创业团队时，每年会招不少实习生，然后对实习生进行培训，参与实际项目，最后留下来一批优秀的有潜力的实习生，在一两年后，就能成长的不错，能独立完成小型的项目。

但我在这种方式上也犯过错误，就是新人的比例太高，中间断层，日常的技术指导和代码审查一度跟不上，导致代码质量低下。所以在招人时，也不能一味节约成本，还要注意梯队的建设，中间要有几个有经验的技术骨干帮助把控好代码质量。

### 小团队如何培养人

在培养人方面，相对来说，小团队不像大公司有完善的培训制度，资源也有限，难以请到外面的人来讲课，**所以培养人主要还是要靠内部形成好的学习分享的机制。**

在大厂，新人加入，通常会指定一个 Mentor，也就是导师或者师傅，可以帮助新人快速融入环境，新人有问题也可以随时请教。这种师傅带新人的机制其实对小团队一样适用，对新人来说可以快速融入，及时获得指导，对于师傅来说，通过带人，也能促进自身的成长。

除了有师傅带，新人的技术成长，更多还是来源于在工作过程中不断实践和总结，在这个过程中，及时准确的反馈很重要。软件工程中，像代码审查、自动化测试、持续集成都可以帮助对工作结果进行及时反馈。

代码审查，可以帮助团队及时发现代码问题，也能促进团队相互学习，代码风格统一；自动化测试，可以对代码结果马上有直观的反馈，有问题早发现修正；持续集成也是通过频繁地集成频繁地给出有效反馈，及早发现代码问题。

在小团队推行这样好的开发实践，让团队获得及时准确的反馈，有助于整个团队的成长。

另外，内部的技术分享也是很好的共同提升的方式，对于听的人来说可以学习到一些新鲜的知识，对于分享的人来说，准备一个技术分享，本身就是最好的学习总结方式。我以前在团队会定期组织这样的技术分享，不止我自己，每个团队成员都会去分享，整个团队分享讨论的技术氛围形成的很好。

还有在分工方面，不要因为一两个技术大牛能干，就把大部分工作都让他们做了，这其实对团队整体是不利的，“大牛”的发展也遇到瓶颈，而其他人缺少锻炼。所以最好是让“大牛”一半的精力负责一些重要的像架构设计、框架开发的工作任务，同时还要有一半的精力在代码审查、带新人等方面，帮助其他人一起成长，整个团队的发展才能更健康。

## 小团队如何管理人

因为小团队人数不多，对人的管理上，可以不需要像大公司一样用复杂的组织结构，用复杂的管理制度。**小团队的管理，核心在于营造好的氛围，鼓励成员自我驱动去做事。**

其实这个理念和敏捷开发的理念是吻合的。在专栏文章《[05 | 敏捷开发到底是想解决什么问题？](#)》中，我也提到了：敏捷开发的实施，离不开扁平化的组织结构，更少的控制，更多的发挥项目组成员的主动性。

要鼓励团队自驱动，具体做法上也可以参考敏捷开发的一些做法，比如说通过任务管理系统和看板，让团队成员自己领取开发任务；在制定一个迭代的计划的时候，让团队成员一起参与对任务的打分，参与计划的制定。

除了这些鼓励成员自驱动，发挥主动性的做法，在营造好的团队氛围上，还要注意的就是遇到线上故障、进度延迟这些不太顺利的情况，更多的是提供帮助，一起总结复盘，而不是甩锅问责。

### 有关开除人

在应用软件工程的时候，团队中可能有些人会成为障碍，要么是能力不足无法落实，要么是态度有问题抵触软件工程的实施。

在这种情况下，首先对于有问题的成员肯定是要努力挽救，如果是能力不足，就给予帮助，给时间成长，对于态度有问题的，明确指出其问题，限期改正。但如果最终结果还是达不到预期的话，那就必须要果断地将这些成员淘汰。

## 2. 流程建设

小团队被人诟病较多的地方就是在于流程规范的缺失，但像大公司，流程规范繁多，也容易造成效率低下，人浮于事的情况，这也就是为什么现在大公司的开发团队也在分拆，从大团队拆分成小组，精简流程规范。

对于小团队，一开始也不宜有太多的流程规范，不然，如果流程不合适反而会成为一种束缚，最好只是先设置最基本的流程规范，然后在实践过程中针对团队特点和业务特点去逐步完善。

那么哪些流程是软件开发中最基本的流程规范呢？

### 选择适合你的软件开发模型

现在的软件开发，已经不再像以前那样采用原始边修边改的开发模型，而是应该采用科学的开发模型。我们专栏一开始就有大量的篇幅介绍各种开发模型，大的方面有瀑布模型和敏捷开发，基于瀑布模型还有很多衍生模型。



那么小团队应该采用哪种开发模型比较合适呢？

也许你会认为应该采用敏捷开发。敏捷开发确实是一种非常适合小团队的开发模型，整个开发过程非常有效率。如果能采用敏捷开发是最好的。

但需要注意的是，如果你的团队是以瀑布模型为主，大家都有丰富的瀑布模型开发经验，但是对敏捷开发都没有实践过，对于敏捷开发的各项活动还不熟悉，还没能充分理解敏捷的价值观和原则，那么最好不好贸然直接换成敏捷开发。

因为这样做的话，团队在一段时间内，都需要去摸索如何用敏捷开发，可能反而会降低开发效率。

对于团队只熟悉瀑布模型这种情况，有条件的话，聘请外部的敏捷顾问帮助实施敏捷开发是个不错的选择。如果条件有限，可以先尝试逐步借鉴敏捷开发中好的实践。

敏捷开发中哪些实践是适合小团队借鉴的呢？

首先在开发周期上，应该缩短交付的时间，使用快速迭代的开发模型。因为小团队的一个特点是需求变化快，要求交付的速度快，那么快速迭代或敏捷开发就是一个合适的开发方式。即使团队习惯了瀑布模型开发，切换到快速迭代也会比较容易，只需要把大瀑布拆分变成小瀑布。

具体在实施上，可以缩短并固定开发周期，比如说每 2~4 周可以发布一个版本。在做迭代的规划时，优先选择当前最核心最重要的功能；在一个版本内，不轻易接受新的需求变更，有需求变更放到下一个迭代中；在迭代时间结束了，无论新功能是否开发完成，都按时发布新版本，没完成的放入下一个迭代。

通过这样的变化，可以保证在一个迭代中整个团队的开发状态是稳定的，不需要受到需求变更的干扰，也可以慢慢形成适合团队的迭代节奏。

另外在会议上，敏捷 Scrum 的几个会议也可以借鉴，像每日站立会议，可以帮助团队及时了解项目进展，解决进度上的障碍；每个迭代的计划会议，可以让大家一起参与到计划的制定

中；每个迭代的验收会议，可以让业务部门、老板及时的验收工作成果，看到大家的工作进展；每个迭代的回顾会议，可以帮助阶段性复盘总结，不断优化开发流程。

还有基于看板的任务可视化，也可以帮助团队直观的看到当前迭代中的任务进度，可以主动选取任务，而不需要去问项目经理下一步该做什么。

以上这些内容，也可以参阅专栏文章《[🔗06 | 大厂都在用哪些敏捷方法？（上）](#)》，里面有更详细的解释。

## 构建基于源代码管理工具的开发流程

很多小团队开发质量低，开发混乱的一个原因就是没有使用源代码管理，也没有一套基于源代码管理的开发流程。在专栏文章《[🔗26 | 持续交付：如何做到随时发布新版本到生产环境？](#)》和《[🔗30 | 用好源代码管理工具，让你的协作更高效](#)》中，对于如何基于源代码管理工具构建和开发已经有了非常详细的介绍，这些开发流程一样适用于小型团队。

有一点要注意的是，小型团队完全没有必要自己去从头搭建自己的源代码管理工具、持续集成工具，应该尽可能采用在线托管的服务，这样可以节约大量搭建、维护工具的人力和时间成本。

类似的策略也应体现在技术选型上，小团队应该尽可能使用现成的工具、框架，而避免自己造轮子，把主要精力放在业务功能的开发上面。

## 建立外部提交需求和任务的流程

小团队在流程规范上混乱的一个体现是，业务部门包括老板对于提交开发任务非常随意，可能直接找某个开发人员私下让改一个需求，增加一个功能，导致开发人员不能专注于任务开发，经常被打断。还有多个项目并行而资源又紧缺的情况下，每个项目负责人都觉得自己的业务是最重要的，希望能尽快完成。

如果你有过在火车站售票口排队买火车票的经历，你会发现，无论人有多少，只要大家有序排队，售票窗口就能按照先后顺序为大家服务，如果大家一窝蜂挤上去买，就会乱成一团，如果



有人插队，那么其他人的进度就会受影响。

其实软件项目开发也是类似的，对于开发团队来说就像是售票窗口，买票的人就相当于一个个的开发任务，无论开发任务有多少，只要你将这些开发任务排成队列，就可以有序地解决。如果一个业务团队的开发任务特别紧急要插队，那么意味着其他业务团队的任务就必须要受影响，那么就需要大家一起去协调。如果你不去通过流程规范任务，那么任务一多，必然就会乱成一团，无论是开发团队内部还是外部，都不会满意。

建立外部提交需求和任务的流程，可以参考专栏《[🔗 14 | 项目管理工具：一切管理问题，都应思考能否通过工具解决](#)》的内容，让所有人都基于任务跟踪系统去提交需求和开发任务，所有任务都先进入 Backlog（任务清单），然后在每个开发迭代中，去按照优先级选择当前迭代的任务，如果有优先级的冲突，应该需要事先沟通解决。对于提交需求和任务的人，也能通过任务跟踪系统，及时的了解到任务的进展。

在团队之外推行这样的流程是会有一定阻力的，最好是能事先去找几个关键的业务负责人私下沟通，取得理解和支持，让他们知道这样做对他们的好处，比如说可以更好地跟踪任务的进展，让开发效率更高，更好地为他们完成任务。

以上这几个流程，就是在小团队的软件开发中应用软件工程，需要建立几个最主要的流程，把这几个基础流程建立起来后，就可以帮助小团队的开发，从无序逐步进入有序。

## 总结

今天，我带你一起分析了小团队在软件项目开发上的主要问题是：对成本敏感、人少活多和缺少流程规范。相应的，我们就需要从团队建设和流程建设两个地方入手，去解决这些问题。

在团队建设方面，需要从四个方面入手：招人、培养人、管理人和开人。

招人的时候，找一些有潜力的培养，也要注意梯队建设，中间有技术骨干补充；

对团队的人才要悉心培养，通过给新人安排师傅的方式培养新人，日常注意代码审查，内部技术分享是个不错的共同提高的方式，技术高手要注意不只是闷头干活，也要承担一定的带人的工作；

管理人核心在于营造好的氛围，鼓励成员自我驱动去做事；

对于不适合团队的人也不要手软，及时的淘汰。

在流程建设方面，要着重建设好三个方面的流程：

选择合适的软件开发模型，建立项目开发流程；

构建基于源代码管理工具的开发流程；

建立外部提交需求和任务的流程。

团队建设和流程建设是在小团队中应用软件工程的关键，通过团队建设让团队成员有共同的软件工程意识，有实施软件工程的基础，通过流程建设让软件工程好的实践流程化、工具化。

## 课后思考

你有小团队的项目经历吗？你觉得小团队面临的主要问题是什么？你觉得可以从哪些方面在小团队中应用软件工程？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

## 精选留言 (10)



**Sam\_Deep\_Thinking**

2019-06-04

我觉得两周为一个发布周期，很有可能导致代码质量低下。例如：两周一迭代里，我们可能没有时间在上个迭代里就做好下个迭代需求的分析，只能遗留到当前迭代，这个时候，需求分析、代码设计、接口设计就要花好几天，好了，由于限制死了两周要发布一次，导致测试人员死盯着发布日期，进行倒推，让开发人员尽量在某个时间点提测，不然迭代上线就风险很大，这样就导致了开发这边压力很大很大，开发时间短，代码质量低，提测后，又各种bug，也进而阻碍测试进度，整条线都非常疲惫紧张。最惨的是，由于测试人员急着测试，也未能做到详细测试，就上线了。又是各种线上bug。因此这种两周一上线，会容易让人死盯着上线日期，给全部人员带来很大的压力，相当于是给自己挖矿和约束了。很不应该的

我觉得软件工程里，开发阶段是最关键的阶段，得给到合理的时间，不然这个阶段被动了，乱了之后，就会产生一系列的不好级联反应。因此，我觉得应该有开发人员来把控节奏，给出工作量，给出哪些可以优先测试。

作者回复: 🍌好问题！你说的担忧完全合理，也确实可能会出现这样的情况。

我来解释一下为什么2-4周是可行的。我们假设你现在的项目是三个月周期，一共是12周，然后你大约2-3周在需求，2-3周架构设计，4周左右在编码，2-3周测试。

那也就是说需求分析期间，其实开发、测试做不了啥事，架构设计的时候，主要是架构师在忙，编码的时候，主要是程序员在忙，测试的时候，开发和测试在忙。

再假设你大概要完成10个功能，也就是这10个功能从设计到开发预计花了10周时间，平均每周一个功能。

如果换成2周一个迭代，那么我们可以考虑每个迭代只选取2个功能，但是在这两周，整个团队的运作可能是这样的：

迭代v1.1 (2周)

- 产品设计，准备下一个迭代v1.2的产品设计
- 开发，设计和开发这个迭代v1.1的功能，同步修复发现的v1.0的Bug
- 测试，测试上一个迭代v1.0开发好的功能
- 开发完成后，部署开发完成的v1.1到测试环境
- 发布测试验收完的迭代v1.0

迭代v1.2 (2周)

- 产品设计，准备下一个迭代v1.3的产品设计
- 开发，设计和开发这个迭代v1.2的功能，同步修复发现的v1.1的Bug
- 测试，测试上一个迭代v1.0开发好的功能
- 开发完成后，部署开发完成的v1.2到测试环境
- 发布测试验收完的迭代v1.1

也就是你差不多还是有两周时间开发新功能，两周时间测试，但是每两周可以发布一个小版本，而且整体节奏比较平缓。

如果到时间内完不成所有功能，那么就发布完成的，没完成的放到下一个迭代，这样可以保证每周

都可以发布。

配合代码审查和自动化测试以及基于分支开发的流程，可以保证合并后代码质量相对是可靠的。

如果这样操作有难度的，那么采用4周一个迭代，但是每个迭代功能减少，还是一样可行的。还有每个迭代结束后的上线发布，可以有两种类型，小迭代可以不发布生产环境，只是测试环境，几个小迭代后再发布生产环境。

也就是说，方法其实是有的，观念上可以先调整，因为这样的迭代周期肯定是可行的。

共 2 条评论 >

👍 17



**Joey**

2019-06-05

请教宝玉老师，研发过程文档，是否有必要进行统一模版，比如方案设计文档、功能测试报告等。

我们公司的现状，整个研发部门1000人左右，如果不设置模版，大家五花八门，别人不好检查；如果设置模版，研发人员又说限制他们的想象力等。

作者回复：我倒是觉得有模板的文档好写一点，填空就好了。

对于文档模板，我没有什么建议，毕竟每个公司情况不一样。

我经历过的公司没有强制规定要模板的，但会提供两种模板，一种是风格样式的，字体颜色什么的都采用公司品牌的风格；一种是基于内容的模板，把大标题小标题都列出来，写的时候填内容就好了。

文档审查重点是检查内容，而不是格式。



👍 4



**Charles**

2019-06-04

一直在小团队，最多的时候也就20几个人，然后分成2-3组开发

碰到最多的问题：

1. 公司或产品目标不明确，团队凝聚力不强
2. 需求管理混乱，很多都是拍脑袋的需求，没有可行性分析
3. 老师文中也提到的人才专业度欠佳，招牛人难，如果像我们二三线城市的那就更困难了

另外问宝玉老师一个问题，在之前团队管理上的疑惑点，这篇文章中提到的团队内部分享，又让我想起来了，如下：

小团队可能就10来个人，每个岗位可能就1-2个人，这种情况下做内部分享，希望大家都能来参与，那么分享内容不好把控，如果太局限于本岗位知识，其它岗位参与度不高效果也不明显，浪费时间，如果只是本岗位的知识分享，那么就2，3个人讨论下就行了，很是纠结，有什么好办法解决这个问题？谢谢

作者回复：可以设定一些学习的课题分享，比如说最近有什么新技术很火，但是大家都不知道是什么，也很想了解，可以让一个人去学习研究，然后跟大家一起分享，分享的过程其实以讨论为主，分享的人也不需要太多压力，自己也能学到东西，其他听的人在讨论的过程中也能学到东西，共同学习提高。你可以从中做好主持的作用，最好提前也学习准备一些。



👍 4



**calvins**

2020-04-13

最难的是留人，新人有潜力的，工作1，2年就考虑跳槽了，所以大部分小公司，开始可能会规范，到后来都乱了，一个人顶多个用，时间紧，任务重，很多标准，流程化的东西就慢慢放弃了，举个例子，就那文档来说，很多项目开发完，文档是缺失的，可能有部分，但是规范需要的，大多没有，新人进来，一般都是请教老人，没有系统培训。

作者回复：小团队留人确实难，不同的人不同的阶段需求都不一样，比如说新手更注重自身的成长，如果能学到的东西就愿意呆着；水平高一些的会更关注自我实现，希望做的事情有意义和价值；年纪大一点的会希望稳定，以及加班少一点。了解员工的诉求，满足他们的诉求，相对好好留人一点。

流程化能顺利执行有两点非常重要：

1. 流程的设置本身要合理，要符合团队的现状，更不能反人性。

比如说你的流程要求PR的单元测试100%覆盖，但团队人少工期又紧，那就很难执行，不如改成主要流程要求覆盖就会更容易执行。

2. 流程要结合工具，自动化或者半自动化。

比如说你的流程要求PR自动化测试都通过才能合并，但是如果运行自动化测试成本很高，那么就很难执行，如果结合CI，让CI在PR提交的时候，自动运行自动化测试，直观的看到结果，那么这个流程就

比较容易执行了。

文档的问题可以多方面入手：

1. 给写文档留出专门的时间，写文档和写代码一样是重要的工作，需要留足时间。
2. 简化写文档的难度，最好内部有WIKI或者共享的在线文档库，可以方便的新建和编辑文档，文档内容可以多一些图少一些文字
3. 一些文档可以由代码注释和单元测试替代



**不靠谱的琴谱**

2019-08-31

我们公司两个开发，三个老板。项目涉及8种语言（不包含js）。现在基本上是老板提需求我估算个时间直接做，天天催进度，没时间搭建一套自动化的流程；项目上线直接替换指定代码重启就算部署完成；以前别人开发的项目不支持集群，只能大半夜没人的时候发布，重构的成本太高，只能维持现状。看完软件工程感觉不能被这样的公司束缚，我是一个很喜欢自动化的人，包括我自己开发一些代码生成工具，让我更高效的完成任务。

作者回复：你说的这种情况其实很普遍，要单独抽时间出来确实不容易，只能是日积月累，一点点改进，一方面要保证现有业务运行，一方面逐步增加自动化比例。

比如说新项目、新需求、修复Bug，都可以针对性的增加一些自动化测试代码，不用贪多求全，一点点来。

项目间隙或者其他时间，写一些自动化脚本替代一些重复的体力劳动。

另外尽可能使用现成的甚至收费的服务，通过简单的配置就可以帮助你实现自动化构建、部署等工作。

总的来说这些自动化工作都是磨刀不误砍柴工，长远看收益很大。



**hua168**

2019-06-07

像小团队比较乱的话，最好是规范哪些关键的地方？

像我们小团队开发，首先看这个功能也没开发过，如果是开发过，就直接基于以前开发过的代码直接改了。



然后就导致运维有问题，有些路径没有替换完，  
手工输入命令可以运行，用shell脚本监控，发现程序异常，就重启。  
结果就报错了，用脚本死活启动不起来。然后一发现没有路径及文件，叫开发改，要一拖再拖，都不愿意改。老是说赶其他项目😂😂

作者回复：流程规范的建立是一个逐步的过程，发现单个的问题，首先解决问题，解决完后就需要思考一下：是不是可以通过流程规范规避类似问题。

就拿你这个例子来说，可以先把CI持续集成环境搭起来，然后在发现这个问题后，就针对这个路径的问题，提一个Ticket，要求补上这部分的自动化测试代码。这样以后每次提交代码，CI都会自动运行这个测试，出问题了就能及时发现，不至于到了生产环境再发现。

开发人员任务多可以理解，但是你需要把这些任务通过任务跟踪系统统一管理起来，写一个Ticket给他，排上优先级。等其他任务忙完，就该把这个任务给做了。

所以小团队乱，任务跟踪管理、开发规范，这都是需要优先建立的流程规范。



👍 3



**hua168**

2019-06-07

宝哥，像我们之前公司，小团队一个项目就2-3个开发，2-3个项目共用美工、有时甚至前端，公司就一个运维...

有时候为了赶项目进度，有的开发，只是在很关键的，一般做了个简单的注解。

两三年基本，原来开发的项目人都走完了。

如果出现问题，人家都不知道怎么改。

是不是docker化比较好，但是如果有些客户要更改需求，那是不是没得玩了？

作者回复：这种情况跟Docker应该没关系。

这种情况应该从几个方面着手：

1. 首先还是代码要规范，日常通过lint工具和代码审查强制代码实现的时候必须满足代码规范
2. 要有必要的文档，例如设计的文档、架构的文档，可以建一个Wiki，日常有问题就记录在上面，尤其是一些操作手册什么的。



👍 3

**alva\_xu**



2019-06-05

就“选择适合你的软件开发模型”，这一点，我谈谈想法。软件开发模型是瀑布还是敏捷对于软件开发管理来说有很大的不同；但即使采用瀑布模型，对于团队管理来说，我们是可以借鉴敏捷模型的。比如，我们可以采用看板管理来提高任务管理的效率和透明度，可以通过站会来加快问题的沟通，对于“建立外部提交需求和任务的流程”我们也可以借助敏捷管理的思路，通过每天站会或者周例会的时候，一起做个新需求评估。对于技术交流，也可以像敏捷团队里说的培养T型技能的人员为目的来开展。而且，先通过团队管理方式的转变，培养大家的敏捷文化，然后再切到敏捷开发模式，就会更加顺畅。我觉得，小团队管理，一定要培养自主自治合作分享的文化和能力，通过用轮值scrum master 的办法，一点点提高这方面的文化和能力。

作者回复：赞同，敏捷开发很多实践可以借鉴。另外团队成员的自主自治合作分享的文化和能力也很重要👍

感谢分享



👍 3



小老鼠

2019-11-27

小公司赚到钱可能是首当其冲的，但需求方总在变需求，又不多给钱，阻碍了其他项目的接单，对于这个问题如何处理。

作者回复：如果说软件工程上那些控制需求的办法都用了还是没用，那还是得考虑其他方式了，比如寻求法律帮助，签订变更的合同，比如从人情关系想办法办法，找找关系什么的



👍 1



ifelse

2022-07-08

团队建设和流程建设是在小团队中应用软件工程的关键，通过团队建设让团队成员有共同的软件工程意识，有实施软件工程的基础，通过流程建设让软件工程好的实践流程化、工具化。--记下来

