25 | 有哪些方法可以提高开发效率?

2019-04-25 宝玉 来自北京

《软件工程之美》

你好，我是宝玉，今天我想与你讨论一个每个开发人员和项目管理者都关心的话题：如何提高开发效率。

我其实也一直很关注这个话题，收集了很多方法让自己工作变得卓有成效。通过对这些方法的应用，我也可以算得上是一个高效的程序员：曾一个人在很短时间完成了飞信 Web 版客户端；在 DePaul 上学之余，帮学校完成了在线教学播放器系统的改造；三个月时间帮公司完成了主站从 jQuery 到 React 的迁移。

如果让我对学过的这些方法做个整理和总结，再进一步精选提炼，我觉得对我影响最大的是“积极主动”、“以终为始”和“要事第一”这几条看似简单的工作原则。

积极主动，行动起来改变自己

相信你也和我有过相同的经历。成为一个高效程序员，最大的阻力不是来自于不知道方法，而是自己的消极心态。遇到进度延迟、效率低下之类的问题，你就会下意识觉得：

时间进度太紧了；

我已经尽力了；

最近加班太多了没精神；

产品经理太不靠谱了，需求没想清楚，害的我瞎忙活。

是的，你也知道这些答案都很消极负面，可是怎么控制自己不这么想呢？首先你要知道，无论这些事情的本质责任在于环境还是个人，抱怨排斥的心态对于实际工作的改进是没有任何帮助的。

当然，很多人也知道抱怨没用，但具体怎样才能做到不抱怨，并且积极主动呢？史蒂芬·柯维写在他的《高效能人士的七个习惯》书中，对这个问题提到了两个行之有效的建议，我们可以结合着软件开发来分析一下。

想想再回应

我还记得第一次有人给我介绍单元测试很好用，能让你效率更高、代码质量更好时，我的第一反应是不可能，这样明明要多写很多代码，怎么可能会高效？

于是我有一段时间是很排斥的，直到后来参与一个已经有单元测试的项目，尤其是在重构代码的时候，我发现修改了大量代码后，程序还是很稳定，当时便觉得应了网友的那句话，“真香”！

每个人对于外界的刺激都会做出反应，本能的或者习惯性的，就像我前面举的例子，遇到事情会本能的觉得都是外部原因。如果一直这样，那就会进入恶性循环，变得更加消极麻木。

但如果在回应之前，给自己一点时间想想，站在积极的方面理性思考一下，就可以去控制你的本能反应。

所以很多次，就在我脱口而出“不可能”或者“不行”的时候，我提醒自己再想想。于是我会改口说：“我试试”、“我再想想”。这样很多次提醒自己以后，会一点点由“不可能”的本能变成“我想想”的习惯。

后来有人跟我说 CI（持续集成）很好，我思考过了之后进行了尝试，在 Github 上建了一个项目，把 CI 搭起来试了一下，觉得真的是很好。如果我还是秉持着以前的消极心态，不知道又要晚多久才能去尝试。

减少关注圈，扩大影响圈

我关注很多事情，比如编程语言、明星八卦、国家大事，这些都是“关注圈”。而这其中，要区分哪些事，是我可以影响和掌控的，这些事则是“影响圈”。

不要总盯着自己无法改变的部分，你需要要多花时间精力在影响圈上。

比如说，我不能改变 996，至少我可以利用这时间多学习一点，找机会换一个更好的环境；我不能要求每个人都写单元测试，但是我自己的代码写了单元测试，这样项目质量更好了我也更有价值；我不能决定跟什么样的人一起共事，但是我愿意跟他们分享我的经验，他们成长了我也受益。

工作一段时间后，你也可以尝试去扩大自己的影响圈。

比如说，很多程序员像我一样，有过不少因为产品经理需求没想清楚导致返工的经历，后来我就格外关注产品设计相关的知识，业余时间自己学习了不少，这就相当于扩大了我的影响圈。

所以后来产品经理给我一个需求，我不需要在开发完成后才抱怨他不靠谱，而是在给我需求的时候就去跟他讨论，是不是有可能没想清楚。

当你不仅仅局限于程序员的角色思维，扩大了影响圈之后，你就可以试着向产品经理提出很多有价值的建议，比如：

> 这个布局在文字很长的情况下会有什么变化？

如果网络很慢，加载数据的时候应该显示什么？加载失败显示什么？

如果数据为空的时候这个列表应该显示什么？

其实“减少关注圈，扩大影响圈”这个道理也很简单：**接受不能改变的，改变能改变的，尽量扩大可改变项的范围。**

以终为始，想清楚再开工

如果对比一下我在十几年前作为一个新手程序员，和现在作为一个老手写代码有什么不同的话，我认为在新手阶段，我是拿到需求就开始写，写到一半发现好像不对，马上修改，好像还不太对，就这样反反复复，效率很低下。

而现在拿到一个需求，我会先仔细的分析需求文档，反复和产品经理确认各种细节，然后做个简单的设计，考虑清楚模块怎么设计，他们之间是什么关系，然后再写，写完还要加上测试代码。

你知道最大的区别是什么吗？我现在做一件事之前，会先想清楚“终”，然后才知道怎么“始”。所以我先搞清楚需求这个“终”，然后再设计规划出这个从“始”通向“终”的路线，最后从“始”出发写代码，一气呵成，不仅快，而且质量好。这就是“以终为始”。

要做到“以终为始”，就是在做事情的时候注意三点：**目标、原则和计划。**

经常停下来想想目标

我刚毕业参加工作的时候，要开发一个内容管理系统，其中涉及有数据库访问，这就需要把数据表的字段和类对应起来，觉得太体力活了，于是我开始写数据库生成代码工具。而要想写代码生成工具，我还得学习 Winform 知识……就这样几个月过去了，关于这个系统的代码还是最开始的几行！

我的目标是写一个内容管理系统，结果却跑去写代码生成工具，这样怎么能做到高效呢？正确的做法应该是手动完成这几个类的生成，其实用不了几分钟，或者用一个现成的工具。如果觉得代码生成工具是个有意义的项目，应该另外立项，而不应该影响当前的项目。

这样的事情在我身上还发生过几次，所以我后来就逼着自己隔一段时间要停下来想想：我的原始目标是什么？我正在做的事是我的目标吗？如果不是，那么马上回到自己的原始目标去。

制定原则

其实大部分很好的编程方法都是需要坚持做才有效果的，比如说自动化测试代码，有时候时间进度一紧，就会来不及写，时间一长，就会欠下技术债务。

所以我给自己定了一个原则：增加一个功能，就要写自动化测试，如果来不及写，就给自己写一条 Ticket。

这条原则我坚持得很好，所以我的自动化测试代码得以坚持，从而真正帮助我做到高效开发。

你也可以给自己定一些原则，比如：

> “先运行再优化 (Make it Work Make It Right Make It Fast)”——也就是在优化代码之前，先用简单的方法实现，再考虑怎么优化，这样可以保证设计的简单，也可以避免你陷入技术细节中而忽视了原始目标。
>
> “不复制粘贴代码 (Don’ t repeat yourself)”——复制粘贴会导致代码臃肿，不便于维护，提取抽象可以保持简洁。
>
> “每个 Pull Request 要尽可能小”——这有助于把复杂的任务分解成几个简单的任务，简单的任务更容易高效完成。

有原则了，你才能不忘初心，有始有终。

公开自己的计划

那么有了原则就够了吗？显然不是，有了原则，你还要坚定不移地去执行。如何执行呢？做计划。

刚开始工作时，我是害怕做计划的，怕计划了完不成，问到我工作的时间安排时，我会给一个模凌两可的答复，这其实导致了我在实际开发时，缺少进度压力，从而迷失在细节中导致延误进度。

后来我尝试做出了一些改变，把任务细化，做个简单计划，主动给出一个明确的时间点。有了计划指引和时间点的压力，会倒逼着自己时刻专注于目标是什么，“终”在哪里，还有多少没有完成，这样下来工作效率自然而然就会高起来。

通过在做事时，围绕着目标、原则和计划这三个点，反复地刻意地练习，也可以让你慢慢养成“以终为始”的好习惯。

要事第一，把时间用在刀刃上

作为程序员，其实大部分时间并不能专注写程序，总有各种各样的事情打断我们，比如，一会产品经理找你过去开个会确认个需求；一会测试过来找你重现一个 Bug；一会还有同事过来请教你一个问题；微信上老朋友找你叙叙旧；突然生产环境出故障了，需要马上去解决。

就这样一天下去，感觉一直在忙忙碌碌，其实并没有多少时间在写程序。这时候怎么办呢，对手头的事情进行优先级管理。

时间四象限也许你不陌生，就是把事情分成重要紧急、重要不紧急、紧急不重要、不紧急不重要四个象限，不同的事情有不同的应对策略。

重要紧急的事情马上处理

比如说，生产环境出故障了，测试环境部署失败了，这些都是重要并且紧急的事情，只能是马上处理。

重要不紧急的要事，要花最多的时间在上面

对代码重构、写自动化测试代码、确认清楚需求文档，这些事情都属于重要不紧急的事情，但是如果不及时处理，就有可能变成重要紧急的事情，比如不偿还技术债务，就可能会变成生产环境故障。

所以这部分事情我会多花时间，重点做。通常我会每段时间只专注做一两件重要的事，其他事情尽可能忽略，比如前一个阶段我主要的工作就是重构前端代码，这个阶段我就在忙排查性能隐患，至于其他事情，就先放一放。

紧急不重要的事凑一起集中做

像微信的消息通知，无关紧要的会议，请教一个不算很急的技术问题，这些都是紧急不重要的问题，然而却会占用我们大量时间。如果时间都用在这些事情上面，必然会占用在重要事情上所需的时间。

所以我有些小技巧也许你可以参考。比如我在专注干活时，会全屏幕、关掉所有消息通知，保证没有消息干扰，忙完一段后再去集中处理。

还有如果有人找我时我正在忙，如果他的事情不是重要紧急的，我会礼貌地告诉他我正好手头有点事情，大约多少时间后我主动去找他。相应的我也会尊重别人的时间，找别人的时候会先问一下：“你什么时候有 10 分钟左右时间，我想请教你一个问题？”

不重要不紧急的事情能不做就不做

不紧急不重要的事也很多，比如说我的 Mac 电脑突然提示我要更新系统了。我有点强迫症，以前系统一有要升级，我就迫不及待要升级到最新，结果一升级系统，半天就不能干活了。所以后来这种事情，就放在不重要的时间去做，比如周末、睡觉之前让它慢慢升级。

其实我在做开发的时候，觉得很多很杂的事情也不算太多，真正到后来转型做管理的时候，才真正体会到什么叫事情多而杂。但正是源于开发时期就形成的时间四象限方法的运用，让我可以在繁忙的时候，保证时间用在有价值的事情上。

要事第一，就是要保证你有限的时间用在最有价值的事情上。

总结

积极主动、 以终为始和要事第一，这三个原则以及其衍生出来的方法，正是帮助我逐步变成一个高效程序员的关键所在，希望也能对你有所帮助。

如果你已经学习了很多类似的原则或者方法，而觉得没什么效果，那也许只是因为没有尝试把它们变成习惯。你可以像我一样，把认同的好的原则或方法，通过反复的刻意练习，反复地提醒自己，训练成习惯，然后用习惯指导你的日常开发。

当然，这样的改变不会是一天两天就能完成，但也不用着急，因为习惯的养成需要时间的积累，才能变成条件反射。当你把好的原则或方法变成了直觉反应，自然就会成为一个高效的程序员。

课后思考

你知道的有哪些提升开发效率的方法？你身边那些高效程序员都是如何做到高效开发的？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。

精选留言 (20)

Joey
2019-09-10

再次请教宝玉老师：对于一个研发部门，一次性入职300+的新员工（原来老员工近700人），可以通过哪些环节的哪些手段，来加强对新员工的研发质量管理，以免出现新员工频频挖坑的情况？

作者回复: 对于大量新员工加入的情况，我建议可以从多个方面入手：

首先是人员的培训方面，有相关的入职培训，新人入职后能快速了解整体的开发流程、规范，新人入职后有指定的Mentor（师傅），在遇到问题时知道该找谁。

然后是项目的开发流程方面，多花时间在流程建设上。具体体现在：
- 整体的项目流程是基于软件工程的标准开发流程的，并且适合你业务特点的，比如说敏捷、迭代或瀑布
- 对于关键环节要有审查，比如设计方案的评审，比如代码合并前的Review
- 要有规范的代码开发流程，比如我多次提到的Github Flow，基于分支开发，代码审查通过并且自动化测试通过才能合并主干
- 要应用好工具，有一定量的自动化脚本和工具辅助，比如说Git，CI/CD的应用，日志数据的收集和监控等。如果这些工具的应用还不够到位，作为一个近1000人的团队，应该成立专门的小组去实施，去做针对公司业务的定制化，去帮助其他团队实施和推广。

再有就是团队组成上，要形成梯队，一个健康的团队，应该有资深的工程师、架构师，有中间的丰富开发经验的工程师，有新手程序员。对于资深的工程师，应该鼓励他们把一部分精力用在评审和帮助新人上，多参与设计的Review和Code Review，帮助团队一起成长，从而实现个人的成长和整个团队的成长。

最后就是对于整个组织架构和技术架构要有针对性设计，一个公司的研发团队目标是满足公司业务需求，这些业务需求又是有各个大大小小的项目组成的，这些项目之间有共同点，有不同点，需要有团队有人能站在整体去思考去设计整体的业务架构，提取公共的架构和服务，将复杂的业务能化整为零，拆分成和组织架构匹配的技术架构，或者按照技术架构去调整组织架构。简单来说就是你的技术架构和组织架构要是匹配的。

比如说你可以把整体的技术架构设计成为微服务的架构，整体组织架构也可以拆分成一个个小的业务小组，一个小组专注于一两个微服务，各自独立维护和发布。这样新人加入，不需要了解整体繁复的架构，只要先了解所属小组的服务架构，就能快速的熟悉业务，快速上手，就算挖坑了，也不会影响太大。

17

胡鹏
2019-05-17

请问宝玉老师:
我现在遇到一些情况，需求出来了，估时的时候，通常有两种心理
第一种: 尽量压缩自己的时间，当然领导也会压缩时间，　这时心里想的是要好好表现，把时间压短一点
第二种: 尽量多一点充裕时间，当出现问题能有足够的时间来解决，不至于延期

第一种优点是，估时的自己心里畅快一点
缺点是: 经常因为特殊原因延期，，印象分就越来越不好

第二种优点是基本上不会延期，有时候还能空一点时间来做项目优化，，但是这个时候要抗住领导压时间

现实场景肯定会更加复杂，有时候紧急需求都是高一两个层级的管理直接下发紧急需求，越快越好，，这个时候时间估短了容易做不完，，长了会被质疑能力问题。（个人感觉了）

我目前主要采用第一种，但是我发现自己做的事很容易延期，为了做好，我就有了更多的加班

和我行程鲜明对比的一个同事，他估时就是第二种，尽量多一点时间，估时的时候长，，，但

是整体有保证，，然后领导对他的信任程度比我高一些（我拿了年度最佳担当奖），

对于这种估时，取一还是取二还是如果在一和二之间平衡，
不知宝玉老师有什么好的建议和观点呢

作者回复: 太紧和太松的时间估算都不可取，应该是尽可能准确的接近实际情况的时间，并且留有一点富裕应对意外情况。

时间太紧了要加班加点还要被质疑能力；时间太松了会影响以后估算时间的真实性。你同事的估算时间应该不是偏松，而是接近实际情况的时间，你可以向他请教经验。

准确的估算时间是程序员能力的一种，做好不容易，一些建议供参考：
1. 充分理解清楚需求，知道要做什么，这是基本前提，不然做着做着发现需求没搞清楚，那一定是要多出很多额外时间
2. 非功能性的需求，比如说写自动化测试、搭环境、重构代码这些任务也应该作为计划的一部分，要把时间算进去
3. 拿到任务后，将任务要分解到尽可能细，越小的任务力度估算越准确，而且在跟领导说时间进度的时候也有理有据，底气足扛得住
4. 综合考虑任务并行的情况，给线上版本修bug、开会这些时间也要算进去，想想每天真正有效的工作时间是多少?
5. 计划保持及时更新，当出现延迟或者有延迟风险的时候，或者进度提前，需要及时和项目负责人沟通，作出调整，避免影响整体项目进度
6. 留一点余量，应对突发情况

反过来，如果你是领导，在下属估算时间的时候，也要参考上面的一些建议，让计划尽可能的接近真实情况，而不是下属给一个很紧的时间就按照这个时间执行，最后得加班加点，加班是为了应对突发情况的，而不是正常情况。

共 2 条评论 >　　6

Joey
2019-04-29

请教宝玉老师：软件工程领域，如果要“树精品意识，扬工匠精神”，有什么思路可以建议，感谢老师解答！（我们现在采用瀑布研发模式，部分条线实现了CI）

作者回复: 精品，其实核心是要小要精，所以你要考虑让产品需求需求小而精，让团队小而精。

工匠精神这已经超出软件工程范畴了：）

6

青石
2019-04-25

设置番茄钟，单次20分钟每天3-4个，关注微信、邮件通知、免打扰。
利用状态最好的时间解决重要紧急问题。
重要不紧急问题优先制定解决方案，定番茄钟解决。
紧急不重要的问题，看优先级集中解决。
不紧急不重要的问题，零散时间解决。

作者回复: 👍谢谢补充

5

易林林
2019-04-25

非常感谢宝玉老师的每一篇文章，逐渐完善了我在软件工程方面的知识和应用。

我觉得高效，意味着自律，自律的好坏是可以通过你散发出来的气息让周围的人感受到的，比如：说话有没有条理，做事拖不拖延等等。生活自律，你会发现每一分每一秒都充满了希望和力量，用积极乐观的心态去完成每一件事，知道自己上一步做好什么，下一步才能做好什么。工作也是一样，要想高效的完成任务，需要利用前辈们总结的思想和方法，去长期实际应用，在使用的过程中就会体现出你的高效，不能说我知道单元测试，我知道CI...，很少有人讲我一直在用。

如果我们注意观察，会看到身边的同事，有的很少加班（活蹦乱跳），有的经常加班（蓬头垢脑），做了一样的事情用了不一样的时间，此时就能真正的体会到高效做事的魅力了。我不提倡加班的原因就在于此，但那是针对高效人士的，低效人士不加班，老板是不会答应的。而一般对自己的时间把握比较好的人，在估算工作时间或工作量的时候，都比较果断，不会支支吾吾，还会主动给出具体时间点和阶段性成果，让人觉得这才是真正做事的人应该有的态度。

我的看法是，积极、主动、自律是高效人士的必备素质。

作者回复: 感谢分享！

👍自律的收益之一就是让你可以专注，比如不会经常刷个微博摸个鱼啥的：）还可以保证自己有更多时间去练习技能。

高效是个正循环，一旦开始高效起来，就不愿意低效率的做事情。

5

alan
2019-04-30

谢谢老师分享宝贵经验！积极主动、以终为始，要事第一。和吴军老师说的“不做伪工作者”很像！

作者回复: 👍吴军老师很多道理讲的特别好，比如把事情做好的三条边

2

alva_xu
2019-04-26

我记得有本极其经典的书叫《人件》，对如何提高效率，讲了很多方法。比如，办公室的布置。我自己的办公室就是按照《人件》的建议，在办公桌和门之间，放了一个隔断，这就可以让我在工作的时候，不会受外部的干扰。

作者回复: 没错，人件是很经典的软件工程书

2

bearlu
2019-04-25

老师，有没有事情管理的工具？因为如果不记录下来，一会儿就忘记了。

作者回复: 楼下的McCree同学推荐了滴答清单。

我个人的话，一般就用系统自带的记事本记一下，或者贴一个便签纸在显示器。如果时间跨度长，我就记到Calendars上，加上提醒。

工作中的任务，我则会创建成Ticket。

2

小老鼠

2019-09-21

要事第一，我也尝试过，往往我忙时别人闲，我闲时别人忙。另外作计划，往计划敢不上变化，中间总有意想不到的事情发生，如何处理？

作者回复: 可以参考文中“要事第一，把时间用在刀刃上”:
按照“紧急重要四象限”对事情进行分级，然后优先处理紧急重要的事，然后处理不紧急重要的事。紧急不重要的事情可集中处理。

1

Tomcat

2019-04-25

积极主动、 以终为始和要事第一重新确立了我的工作原则，我也从中收益较多。
作为程序员，的确会经常被日常的杂事所打断，没有整片时间来专注于写代码，我们能够奢求的，就是要做最重要的事情，其他的事情不要去做！

作者回复: 是的，仔细想想，一天的上班时间，其实做不了多少事情，能把最重要的事情做好就不错了

1

梁中华

2019-04-25

后面互赖圈的三点也很重要，比如双赢思维特别适合，努力工作，提升自我，与公与私都是双赢

作者回复: 《高效能人士的七个习惯》是一本非常好的书，值得学习。

1

miketan

2019-04-25

每次开发前先想清楚如何实现；核心接口做单元测试全覆盖；

作者回复: 👍是的，想清楚做什么特别重要。
除了单元测试，建议还可以配合一些集成测试来测试接口的调用。

1

ifelse
2022-06-30
积极主动、 以终为始和要事第一，这三个原则以及其衍生出来的方法，正是帮助我逐步变成一个高效程序员的关键所在，希望也能对你有所帮助。--记下来

aoe
2022-01-22
没有单元测试，机会一半时间是在找Bug，少许改Bug时间，更少的时间是在完成功能。最近看到Bob大叔的书中《代码整洁之道》提到TDD，这是一个很好的解决方法

LYy
2020-07-26
重要不紧急的事情 要保证每天都投入时间(自己设定一个底线)做

Raymond吕
2020-03-23
老师讲的方法是具有普适性的，其他岗位也可以参考。终归到底就是从入职那一刻起，就养成好的工作习惯，终身受益。

作者回复: 👍软件工程中很多方法在其他领域也适用

calvins
2019-12-31
我以前是给每天做的事排优先级，根据难易程度，完成时间长短，重要程度，综合后排序，同时把能快速不占用太多时间的事处理完毕，集中处理其他的事，用桌面便签做记录提醒

作者回复: 👍
另外把日历提醒用起来也是个不错的方式，不容易忘事

AI午间狂睡Steve
2019-10-01

谢谢老师，时间四象限实在太棒了👏🌟

作者回复: 🤝


Sudouble
2019-05-07

最近一直在用番茄时钟，中间还可以稍微休息一下。把现在的刻意练习逐渐变成习惯。

作者回复: 👍养成习惯了就容易多了。


McCree
2019-04-25

楼上的，滴答清单

作者回复: 🙏感谢推荐