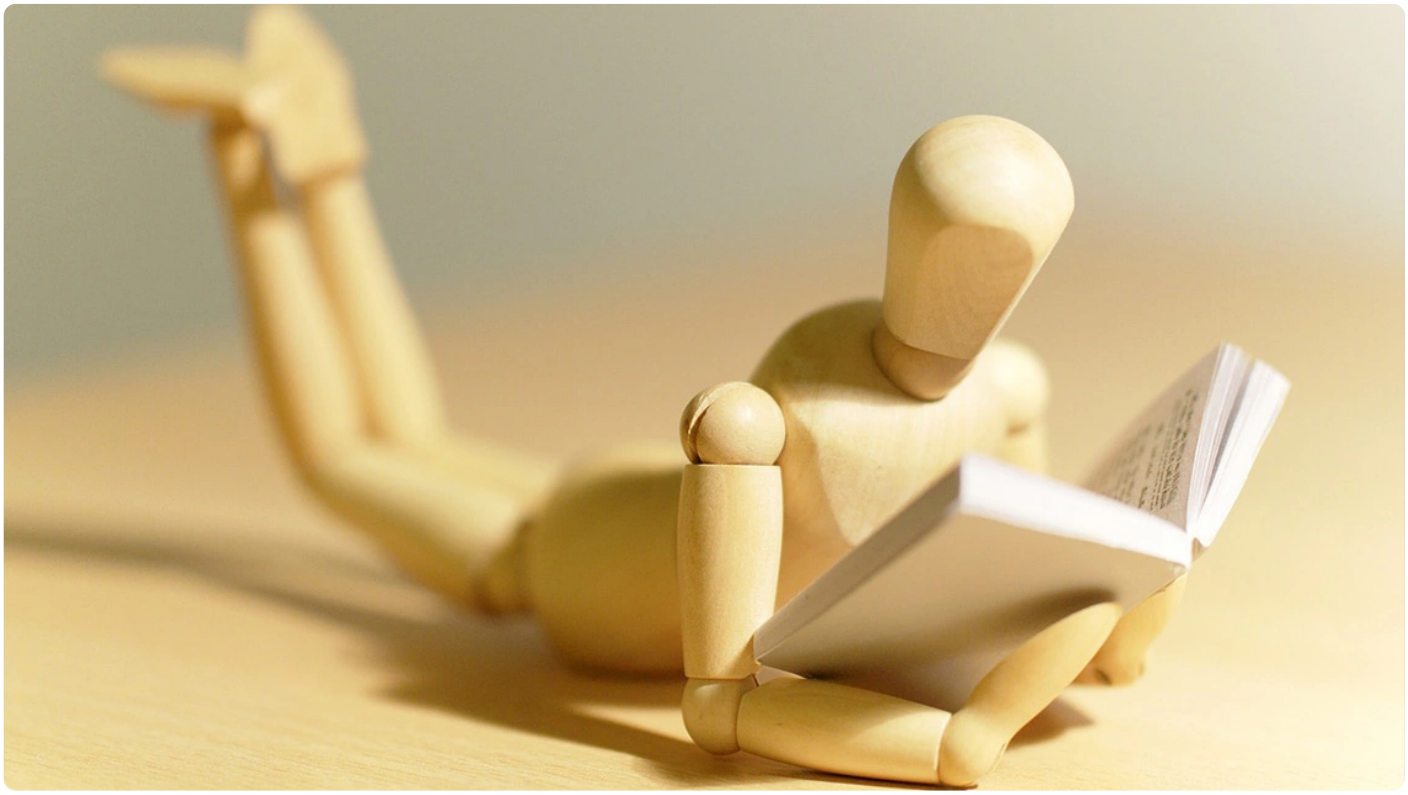
43 | 以VS Code为例，看大型开源项目是如何应用软件工程的?

2019-06-11 宝玉 来自北京

《软件工程之美》



你好，我是宝玉。如果你所在的团队在日常的软件项目开发中，能科学地应用软件工程的知识，让你的项目能持续取得进展，最终交付的产品也有很好的质量，那么是一件非常幸运的事情。

然而现实中，很多人并没有机会去参与或观察一个好的项目是什么样子的，也没机会去分析一个好的项目是如何科学应用软件工程的。

好在现在有很多优秀的开源项目，不仅代码是公开的，它们整个项目的开发过程都是公开的。通过研究这些开源项目的开发，你能从中学习到一个优秀项目对软件工程的应用，加深你对软件工程知识的理解，进而应用到你自己的项目实践中。

我想你对 VS Code 应该不陌生，它是一个非常优秀的编辑器，很多程序员包括我非常喜欢它。VS Code 也是一个大型的开源项目，整个开发过程非常透明，所以今天我将带你一起看一下 VS Code 是如何应用软件工程的，为什么它能构建出这么高质量的软件。

如何从 VS Code 的开发中学习软件工程?

也许你会很好奇，平时也去看过 VS Code 的网站，但并没有提到软件工程的呀?

是的，VS Code 的网站并没有特别突出这些信息，但是如果你有心，可以找到很多有价值的信息，它的整个开发过程都是公开透明的。

比如通过它项目的WIKI和博客栏目，可以看到项目的计划、项目开发流程、测试流程、发布流程等信息。通过它的GitHub网站，你可以看到团队是如何基于分支开发，开发完成后提交 Pull Request，团队成员如何对代码进行审核，合并代码后如何通过持续集成运行自动化测试。

除此之外，团队成员在网上也有一些对于 VS Code 开发的分享，比如说 VS Code 主要负责人 Erich Gamma 2016 年在 GOTO 技术大会上有一个专门关于 VS Code 的主题演讲。

也许你还会想问：这些信息我也知道，也能从网上看到，但怎么通过这些信息去观察和学习它跟软件工程相关的部分呢?

不知道你是否还记得，在我们专栏的第一篇文章《01 | 到底应该怎样理解软件工程?》中提到了：**软件工程的核心，就是围绕软件项目开发，对开发过程的组织，对方法的运用，对工具的使用。**所以当我们去观察一个软件项目，我们就可以去看它的开发过程是怎么被组织的？运用了哪些软件工程的方法？使用了哪些工具?

接下来，我就带你一起从以下几个方面分析 VS Code 对软件工程的应用：

VS Code 的开发过程；

团队的分工角色；

各个阶段如何进行；

使用了哪些工具。

VS Code 的开发迭代过程

如果你是 VS Code 的用户，你会发现 VS Code 每个月都会有新版本的更新，每次更新都会有很多新酷的功能。这是因为 VS Code 每个版本的开发周期是 4 周，每四周都会发布一个新的版本。

从开发模式来说，VS Code 采用的是快速迭代的开发模式，每四周一个迭代。那么这四周迭代的工作都是如何进行的呢？

第一周

每个版本的第一周，通常是起着承上启下的作用，一方面要准备新版本，一方面还要对上一个版本的工作进行收尾。

在这一周里，开发团队要去做一些偿还技术债务的事情，比如说重构代码，优化性能。所以如果你的团队抱怨说没有时间做偿还技术债务的事情，不妨也去学习 VS Code 团队，定期留出专门的时间，做偿还技术债务的事情。

另一个主要工作就是一起讨论下一个迭代要做的功能。其实这有点类似于敏捷开发中，每个 Sprint 开始之前的项目计划会议。

如果上一个版本开发完成的功能，发现了严重 Bug，第一周还要去修复这些紧急 Bug。

第二周和第三周

第二周和第三周主要工作就是按照计划去开发，一部分是开发新功能，一部分是修复 Bug，所有的 Bug 都是通过 GitHub 的 Issue 来分配和跟踪的。

团队成员每天还要先检查一下分配给自己的 Issue，如果遇到线上版本紧急的 Bug，要优先修复。

第四周

VS Code 团队把最后一周叫 End game，你可以理解为测试周，因为这一周只做测试和修复 Bug。

这一周要测试所有新的 Feature 和验证已经修复的 Bug，确保被修复。同时还要更新文档和写 Release Notes。

测试完成后就发布预发布版本，这个预发布版本会先邀请一部分人使用，比如说微软内部员工、热心网友。

下一个迭代第一周

每个迭代开发测试完成的版本，会放在下一个迭代的第一周发布。如果在预发布版本中发现严重 Bug，需要在第一周中修复。

如果没有发现影响发布的 Bug，那么第一周的周三左右就会正式发布上一个迭代完成的版本。

前面我在专栏文章《40 | 最佳实践：小团队如何应用软件工程？》中，建议小团队可以缩短迭代周期到 2-4 周，有同学担心不可行，但你看 VS Code 这样稳定的 4 周迭代，不但可行，而且还是 VS Code 能保持每月发布一个新版本的关键所在。

VS Code 团队的角色和分工

VS Code 的开发团队现在大约 20 人左右，一半在苏黎世，一半在西雅图。整个团队基本上都是开发人员，结构很扁平。

从分工上来说，在开发新功能和修复 Bug 的时候，会有一些侧重，比如有人侧重做 Git 相关的功能，有人侧重做编辑器部分功能。这样有侧重的分工对于提升开发效率是有好处的。

从角色上来说，除了开发，还有主要有两种角色：Inbox Tracker和Endgame Master。这两种角色在每个迭代的时候是轮值的，每个人都有机会去担任这两个角色。

Inbox Tracker

Inbox Tracker 的主要任务就是收集、验证、跟踪 Bug。但这个工作对于 VS Code 团队来说可不轻松，现在 Issue 的总量已经超过了 5000，每天提交的新的 Issue 的量大概有 100 左右。所以 VS Code 团队写了一个机器人叫 VSCodeBot，可以帮助对 Issue 先自动处理，打标签或回复，然后 Inbox Tracker 再对剩下的 Issue 进行人工处理。

Inbox Tracker 要检查新提交的 Issue 是不是一个真正的 Bug，如果是提问，建议到 StackOverflow 去问，如果是 Bug，打上 Bug 的标签，并指派给相应模块的负责人。

> Endgame Master

VS Code 团队是没有专职的测试人员的，所有的测试工作都是开发人员自己完成。在每一个迭代中。Endgame Master 在这里就很重要，要组织管理整个迭代的测试和发布工作。

Endgame Master 在每个迭代测试之前，根据迭代的开发计划制定相应的测试计划，生成 Check List，确保每一个新的功能都有在 Check List 中列出来。

因为 VS Code 团队没有专职测试，为了避免开发人员自己测试自己的代码会存在盲区，所以自己写的功能都是让其他人帮忙测试。Endgame Master 一个主要工作就是要将这些测试项分配给团队成员。

最后整个测试计划会作为一条 GitHub Issue 发出来给大家审查。比如说这是某一个月的 Endgame 计划。

团队的日常沟通是通过 Slack，在测试期间，Endgame Master 需要每天把当前测试进展同步给所有人，比如说总共有多少需要测试的项，哪些已经验证通过，哪些还没验证。

VS Code 的各个阶段

接下来，我们来按照整个开发生命周期，从需求收集和版本计划、设计开发、测试到发布，来观察 VS Code 各个阶段是如何运作的。

1. VS Code 的需求收集和版本计划

VS Code 每次版本发布，都能为我们带来很多新酷的功能体验，那么这些功能需求是怎么产生的呢？又是怎么加入到一个个版本中的呢？

VS Code 的需求，一部分是团队内部产生的；一部分是从社区收集的，比如 GitHub、Twitter、StackOverflow 的反馈。最终这些收集上的需求，都会通过 GitHub 的 Issue 管理起来。如果你在它的 GitHub Issue 中按照 feature-request 的标签去搜索，可以看到所有请求的需求列表。

VS Code 每半年或一年会对下一个阶段做一个 Roadmap，规划下一个半年或一年的计划，并公布在 GitHub 的 WIKI 上，这样用户可以及时了解 VS Code 的发展，还可以根据 Roadmap 上的内容提出自己的意见。

大的 RoadMap 确定后，就是基于大的 RoadMap 制定每个迭代具体的开发计划了。前面已经提到了，在每个迭代的第一周，团队会有专门的会议讨论下一个迭代的开发计划。在 VS Code 的 WIKI 上，也同样会公布所有确定了的迭代计划。

那么，有了功能需求和 Bug 的 Issue，也有了迭代的计划，怎么将 Issue 和迭代关联起来呢？

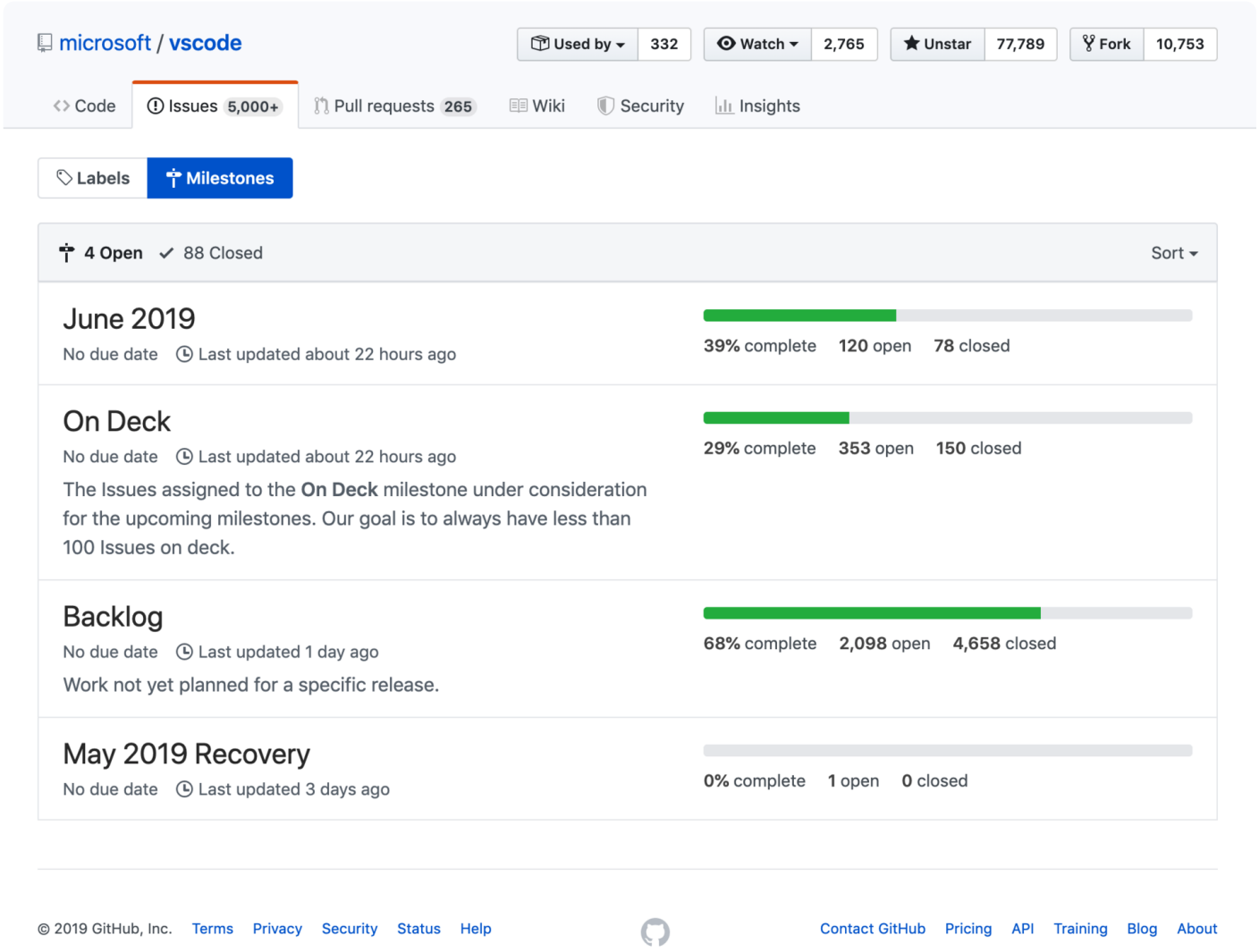
GitHub 的 Issue 管理有一个 Milestone 的功能，VS Code 有四个主要的 Milestone。

当前迭代：当前正在开发中的 Milestone；

On Deck：下一个迭代对应的 Milestone；

Backlog：还没开始，表示未来要做的；

Recovery：已经完成的迭代，但是可能要打一些补丁。



(图片来源：VSCode Milestones)

2. VS Code 的设计和开发

VS Code 的架构设计现在基本上已经定型，你在它的 WIKI 和博客上还能看到很多 VS Code 架构和技术实现的分享。

在每个迭代开发的时候，一般小的功能不需要做特别的架构设计，基于现有架构增加功能就好了。如果要做的是大的功能改造，也需要有设计，负责这个模块开发的成员会先写设计文档，然后邀请其他项目成员进行 Review，并给出反馈。

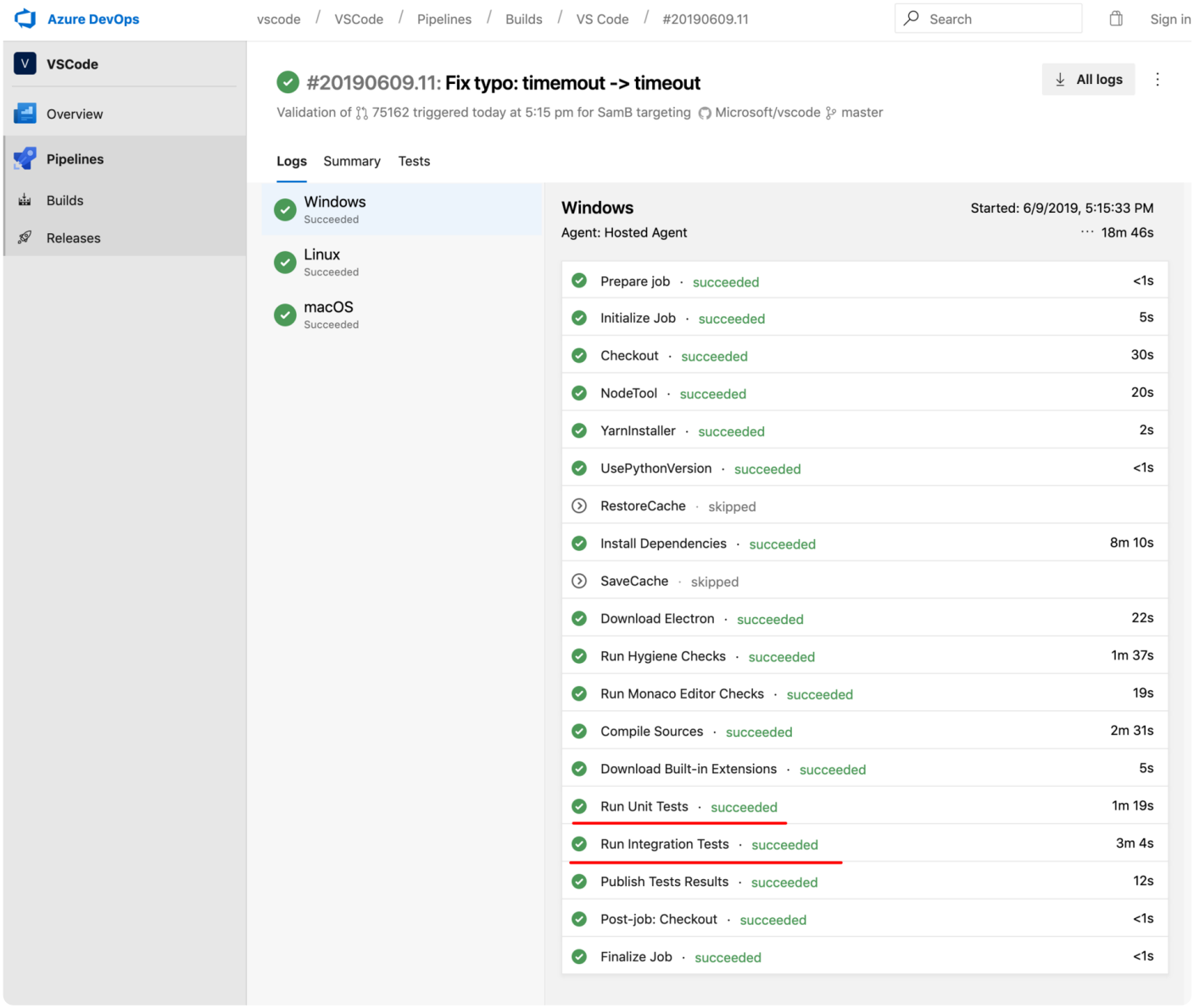
VS Code 的开发流程也是用的GitHub Flow，要开发一个新功能或者修复一个 Bug，都创建一个新的分支，开发完成之后提交 PR。PR 合并之前，必须要有核心成员的代码审查通过，

并且要确保所有的自动化测试通过。

对于 GitHub Flow 的开发流程，我在专栏文章《30 | 用好源代码管理工具，让你的协作更高效》中有详细的介绍。你也可以在 VSCode 的Pull requests中看到所有提交的 PR，去看看这些 PR 是怎么被 Review 的，每个 PR 的自动化测试的结果是什么样的。通过自己的观察，去印证专栏相关内容的介绍，同时思考是否有可以借鉴到你自己项目中的地方。

VS Code 对自动化测试代码也是非常重视，在实现功能代码的时候，还要加上自动化测试代码。如果你还记得专栏文章《29 | 自动化测试：如何把 Bug 杀死在摇篮里？》中的内容：自动化测试有小型测试、中型测试和大型测试。VS Code 的自动化测试也分为单元测试、集成测试和冒烟测试。

VS Code 的CI（持续集成）用的是微软自己的 Azure DevOps，每一次提交代码到 GitHub，CI 都会运行单元测试和集成测试代码，对 Windows/Linux/macOS 三个操作系统分别运行测试。在持续集成上可以直观地看到测试的结果，VS Code 现在有大约 4581 个单元测试用例，运行一次 1 分钟多；集成测试 466 个，运行一次大约 3 分钟。



（图片来源：VSCode 的持续集成工具 Azure DevOps）

如果你的团队还没有开始相应的开发流程，没有使用持续集成工具，不妨学习 VS Code，使用类似于 GitHub Flow 的开发流程，使用像 Azure DevOps 这样现成的持续集成工具。

3. VS Code 的测试

前面提到了，迭代的最后一周是 End game，这一周就是专门用来测试的，并且有轮值的 Endgame Master 负责整个测试过程的组织。

具体测试的时候，大家就是遵循 Endgame Master 制定好的测试计划，各自按照 Check List 逐一去检查验证，确保所有的新功能都通过了测试，标记为修复的 Bug 真的被修复了。对于验证通过的 Bug，在对应的 Issue 上打上 verified 的标签。

在人工测试结束后，Endgame Master 就需要跑冒烟测试，确保这个迭代的改动不会导致严重的 Bug 发生。

如果你的团队也没有专职测试，可以学习 VS Code 这样的做法：留出专门的测试阶段，事先制定出详细的测试计划，把所有要测试的项都通过测试跟踪工具跟踪起来，开发人员按照测试计划逐一测试。

4. VS Code 的发布流程

在 Endgame 测试后，就要从 master 创建一个 release 分支出去，比如说 release/1.10 ，后面的预发布版本和正式版本包括补丁版本都将从这个 release 分支发布。

如果在创建 release 分支后发现了新的 Bug，那么对 Bug 修复的代码，要同时合并到 master 和 release 分支。每一次对 Release 的代码有任何改动，都需要重新跑冒烟测试。

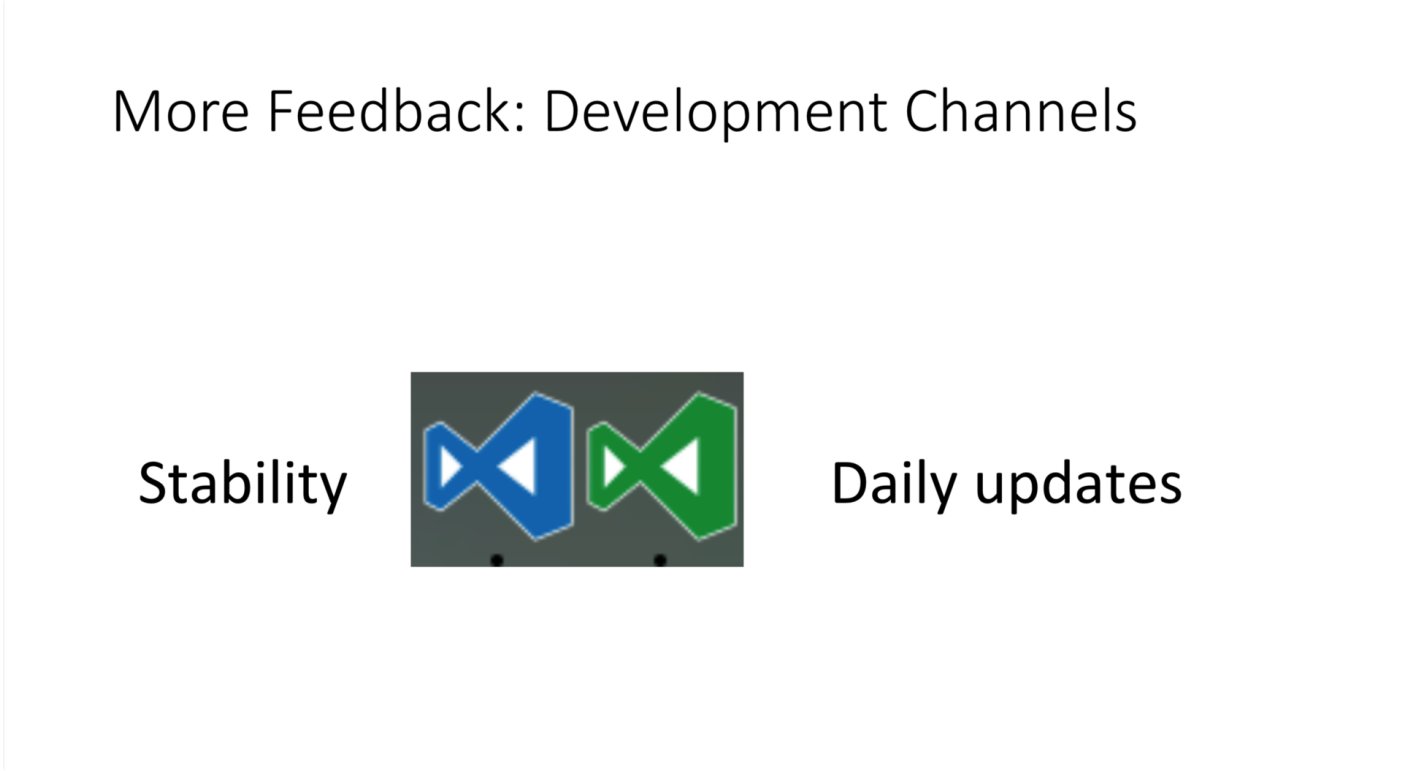
在 Release 分支的代码修改后的 24 小时之内，都不能发布正式版本。每次 Release 代码修改后，都会发布一个新的预发布版本，邀请大约两万的内部用户进行试用，然后看反馈，试用 24 小时后没有什么问题就可以准备发布正式版本。

发布正式版本之前，还要做的一件事，就是 Endgame master 要写 Release Notes，也就是你每次升级 VS Code 后看到的更新说明，详细说明这个版本新增了哪些功能，修复了哪些 Bug。

如果版本发布后，发现了严重的线上 Bug，那么就要在 Release 分支进行修复，重新生成补丁版本。

除此之外，VS Code 每天都会将最新的代码编译一个最新的版本供内部测试，这个版本跟我们使用的稳定版 Logo 颜色不一样，是绿色的 Logo。VS Code 内部有“吃自己狗粮”（eat

your own dog food）的传统，也就是团队成员自己会使用每天更新的测试版本 VS Code 进行开发，这样可以在使用过程中及时发现代码中的问题。



（图片来源：The Journey of Visual Studio Code）

像 VS Code 这样的发布流程，通过创建 Release 分支可以保障有一个稳定的、可以具备发布条件的代码分支；通过预发布内部试用的机制，有问题可以及时发现，避免造成严重的影响。

关于发布流程的内容，你也可以将 VS Code 的发布流程 对照我们专栏文章《35 | 版本发布：软件上线只是新的开始》中的介绍，加深理解。

VS Code 使用的工具

VS Code 的源代码管理工具就是基于 GitHub，整个开发流程也完全是基于 GitHub 来进行的。

它的任务跟踪系统是用的 GitHub 的 Issue 系统，用来收集需求、跟踪 Bug。通过标记不同的 Label 来区分 Issue 的类型和状态，比如 bug 表示 Bug，feature-request 表示功能请求，

debt 表示技术债务。通过 Issue 的 Milestone 来标注版本。

VS Code 的持续集成工具最早用的是Travis CI和AppVeyor，最近换成了微软的Azure Pipelines，在他们的 Blog 上有一篇文章《Visual Studio Code using Azure Pipelines》专门解释了为什么要迁移过去。

VS Code 的文档一部分是用的 GitHub 的 WIKI 系统，一部分是它网站的博客系统。WIKI 主要是日常项目开发、维护的操作说明，博客上更多的是一些技术分享。

另外 VS Code 团队还自己开发了一些小工具，比如说帮助对 Issue 进行自动处理回复的 GitHub 机器人 VSCodeBot。

通过这些工具的使用，基本上就可以满足像 VS Code 这样一个项目的日常运作。像这些源代码管理、任务跟踪系统、持续集成工具的使用，在我们专栏也都有相应的文章介绍，你也可以对照着文章的内容和 VS Code 的使用情况加以印证，从而加深对这些工具的理解，更好把这些工具应用在你的项目中。

总结

当你日常在看一个开源项目的时候，不仅可以去看它的代码，还可以去观察它是怎么应用软件工程的，不仅可以加深你对软件工程知识的理解，还能从中学习到好的实践。

比如观察一个软件项目的开发过程是怎么被组织的，团队如何分工协作的，运用了哪些软件工程的方法，以及使用了哪些工具。

VS Code 使用的是快速迭代的开发模式，每四周一个迭代：

第一周：偿还技术债务，修复上个版本的 Bug，制定下一个版本的计划；

第二、三周：按照计划开发和修复 Bug；

第四周：测试开发完成的版本；

下一迭代第一周：发布新版本。

在团队分工上，VS Code 的团队很扁平，没有专职测试，通过轮值的 Inbox Tracker 和 Endgame Master 来帮助团队处理日常 Issue 和推动测试和发布工作的进行。

在工具的使用方面，VS Code 使用的是 GitHub 托管代码，基于 GitHub Flow 的开发流程使用。还有使用 Azure DevOps 作为它的持续集成系统。

通过观察对 VS Code 对软件工程知识点的应用，再对照专栏中相关文章的介绍，可以帮助你更好的理解这些知识点，也可以借鉴它们好的实践到你的项目开发中。

课后思考

你可以尝试自己去观察一遍 VS Code 项目对软件工程知识的应用，得出自己的结论。你也可以应用这样的观察分析方法，去观察其他你熟悉的优秀开源项目，比如像 Vue、React，看它们是怎么应用软件工程的。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 (11)

胡浩🐸

2019-06-11

赞，有好多值的学习和借鉴的地方，打开 VS Code 经常看到更新提示，感觉被后就有一个非常高效的敏捷团队，今天终于学习到了。

VS Code 每 4 周发布一个迭代节奏非常棒，每周做的事情都很科学合理，而且把 Github Issue 的功能用的淋漓尽致，打的各种 Label 很值的学习。

作者回复: 谢谢蛙总指教🧡

11

一文字

2019-06-11

棒，感觉像打开了新世界的大门，赶紧把这种学习方法消化下

作者回复: 以后看到好的软件项目，尝试去观察它们的软件工程应用，学习借鉴：）

8

Y024
2019-06-11

怎么学习开源项目？除了眼前的代码，还有诗和远方……

作者回复: 有关如何学习开源项目的软件工程，我已经在这篇文章中介绍了。

如果你是想学习代码，我建议你可以把代码下载到本地，尝试着运行起来，然后去看看代码怎么运行的，再去尝试这修改代码，增加功能，去了解它的结构是什么样的，去思考它的设计有什么优缺点？哪些是可以学习借鉴的？

5

露娜
2020-05-08

老师：现在遇到一个特别尴尬的事情，每次发测试前，都会预留时间每个人自测自己的代码，但是总有些人根本就不测，直接扔给测试。测试一测一大堆问题。请问这种情况有么有好的方法可以避免？

作者回复: 这个问题我也遇到过，完全靠管理手段靠流程规范也难以完全杜绝。我觉得首先你要甄别一下，那些老是不自动测试的开发人员，他们只是因为习惯不好？还是就是混日子的？对于后者，还是早点清理出去更好，对于前者，可以想办法去帮助改进。

我现在用的开发流程（参考GitHub Flow https://help.github.com/cn/github/collaborating-with-issues-and-pull-requests/github-flow）还算比较有效，可以酌情参考：
1. 每个任务都要单独分支（branch）单独PR（Pull Request），这样以PR为单位，可以更好的跟踪评定一个开发任务所有相关的代码。否则你所有代码都直接提交到master就很难跟踪。
2. 引入代码评审（Code Review），PR要求有其他人批准才能合并到master，通过评审，让同组成员相互监督提醒。
3. 对于新功能或者Bug，可以要求在PR中提供功能的演示，比如说屏幕截图或者屏幕录屏，通过这样的方式来保证自己测试了，也有利于代码评审的人检查是否完成。
4. 要求有一定比例的自动化测试代码，自动化测试代码也是一种很好的自我测试方法，更可以有效保证质量。

5. 如果CI（持续集成）工具能帮助你自动运行自动化测试，这样测试结果就一目了然。如果新的PR导致了自动化测试失败，说明代码是有问题的，测试也是不充分的。那么就要先修复自动化测试的错误。

以上这个流程的执行，一方面要有开发工具的支持：Git、CI、自动化测试框架，这些网上都有现成解决方案，搭建起来应该不复杂。
另一方面团队里面要有认真负责技术好的成员帮助Review代码，如果没有合适的你就需要自己多费心了。

4

yellowcloud
2019-06-11

宝玉老师，我们目前项目使用的管理工具是TFS，它好像也自带CI和CD功能，我想请问一下，它和文中介绍的Azure DevOps，那个好用呢？

作者回复: Azure DevOps应该是TFS的升级版，如果在线托管的话，你应该考虑用Azure DevOps。

4

test
2020-01-01

看了vscode感觉项目管理也不复杂，关键是一群靠谱的人有动力有方法有共识，项目管理失败基本都是人的因素，说难就难说不难也不难。

作者回复: 是的，项目管理本身并不复杂，还是离不开人的实施。

3

小谢同学
2019-06-18

感觉vscode的日常开发管理工作非常饱和，这里有个问题想请教，以vscode为例，4周的一个迭代周期如何确保效率，特别实在第一周里，包含了历史遗留问题的处理，还要做本次迭代的规划安排，更何况开源项目如果不是全职铺在上面怎么办？如果在某一个迭代周期内因为不可抗力因素导致延期了怎么办？

作者回复: VSCode的开发团队是全职的，而且整体水平相当高。

一个迭代周期内，并非所有功能做完才能发布，而是功能开发完成才合并到主干，如果一个迭代内没完成，那么会放到下一个迭代。

举个简单例子，一个迭代周期在计划的时候，打算增加2个新功能，修复3个bug。结果在第三周结束，发现只完成了一个功能，修复了3个bug，那么另一个功能就放到下一个迭代继续开发，第四周对已经完成的1个功能和修复的3个bug验证就好了。

3

freda

2019-10-12

你好，我想请教下，我领导想用禅道软件做项目管理，可是我觉得禅道更适合做软件开发，想听听你的看法

作者回复: 像VSCode，基于Github的Issues，都可以做到项目管理，用禅道或者同类型的Ticket跟踪工具更没问题的。

我在《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》里面对项目管理工具也有一些介绍，可以作为参考。

项目管理工具，关键还是能对任务进行分配和进度跟踪，通过项目管理工具，项目成员能即时知道自己应该要做的任务，项目管理者能即时知道项目进展情况。

共 2 条评论>

1

远征

2019-06-12

师傅领进门：）不仅知道如何入手开源项目，而且在项目管理上也有新借鉴！谢谢老师

作者回复: 谢谢支持!
有具体问题也欢迎留言：）

1

ifelse

2022-07-10

当你日常在看一个开源项目的时候，不仅可以去看它的代码，还可以去观察它是怎么应用软件工程的，不仅可以加深你对软件工程知识的理解，还能从中学习到好的实践。--经记下来

清心草色

2021-11-18

window客户端使用WPF开发，是一个好的方向吗?