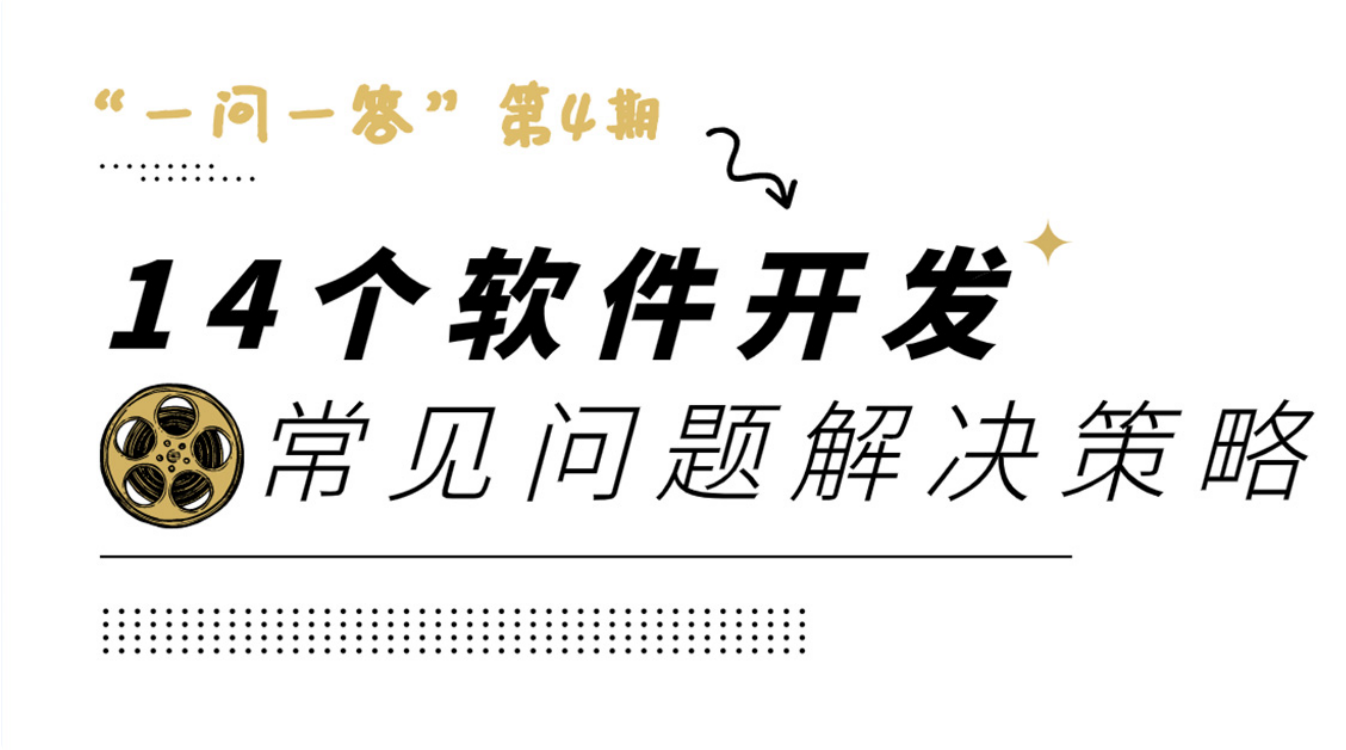
“一问一答”第4期 | 14个软件开发常见问题解决策略

2019-06-01 宝玉

软件工程之美 进入课程 >



讲述：宝玉
时长 03:22 大小 3.09M

你好，我是宝玉。恭喜你完成了软件测试和线上维护这两个模块的学习。

软件测试是保障软件质量的重要一环，但也不能过于依赖软件测试，毕竟软件质量体现在功能质量、代码质量和过程质量三个方面，而软件测试只能帮助保证功能质量，代码质量和过程质量还需要团队一起努力。

现在大厂已经很少有手工测试的岗位了，大部分测试工作已经转移到开发上，同时自动化测试的比例越来越高，但这相应的对开发的要求更高了，不仅要写功能代码，还要写自动化测试代码。

软件测试也离不开对工具的使用，通过 Bug 跟踪工具报 Bug 和跟踪 Bug，使用测试管理工具管理测试用例，基于自动化测试框架写自动化测试代码，借助性能测试工具对软件性能

进行压力测试。测试工具还可以和持续集成一起整合，最大化发挥测试工具的效应。

软件测试做好了不代表你的软件就是安全的，不会导致账号密码泄漏，要想构建安全的软件，需要在整个软件生命周期中都重视安全问题，各个阶段考虑到安全方面的问题，防患于未然，构建出安全的软件。

当你的软件已经测试通过，准备上线发布了，不要忘记在软件发布前做好版本规划，尽可能的让软件的功能质量，满足好用户的预期。一方面要尽可能提供应有的功能和保证质量，另一方面也可以通过合理的发布策略来降低用户的预期。

传统软件发布上线后，就是运维负责保障线上运行了，但这种分工也导致了开发和运维之间在沟通协作上不够紧密，甚至有冲突，于是 DevOps 出现了，它帮助解决开发和运维之间的沟通协作问题，提升运维开发和自动化能力。通过自动化、信息透明可测量、共同协作的原则，达到更快更可靠的构建、测试和发布软件的目的。

软件发布后不代表项目就结束了，还需要处理线上故障，保障运行。遇到线上故障不用慌，要先恢复生产减少损失，然后再去找原因，最后再去总结复盘。日常还要对故障多演习，这样真的有故障发生了才能做到从容应对。日志管理工具是线上故障排查的好帮手，能帮助你快速定位线上故障，还可以对日志的数据进行监控，有问题提前预警。

最后，项目结束后不要忘记对故障进行复盘，总结成功经验，吸取失败教训。把好的实践继续发扬光大，对于不好的实践马上停止做出积极有效的改变。

以上就是两个模块内容的总结，希望你在学习这些知识后，能应用到实际的项目中去，帮助你项目的质量再上一个台阶。

今天加餐继续分享我们专栏的精彩问答和留言，这些问答和同学们的分享都是对专栏内容的最好补充，希望可以帮助你更好地学习和理解软件工程知识。

一问一答

No.1

hua168：质量是怎么打分的？算进 KPI 考核吧？直接用代码质量管理软件（如 sonar）实现自动检查可以吧？

宝玉：很遗憾，都不好量化，软件检查只是辅助，可以作为一个参考。代码质量要看满足需求，是否设计良好，代码简洁逻辑清晰，可维护、可测试、安全高性能；过程质量要看开发过程对软件工程和项目管理知识的应用；功能质量要看客户满意度。

No.2

砍你一刀：能分享一个比较好的测试用例模板吗？

宝玉：我建议你试试 testrail，它的测试用例模板非常专业。

对于测试用例：

几个关键的字段是：标题、描述、优先级、分类。

测试类型：功能测试、性能测试、回归测试、冒烟测试...

自动化状态：没有自动化、只能手动测试、只能自动化、集成 CI...

先决条件：这个用例需要满足好什么条件。

测试步骤：写清楚一步步的执行步骤。

期望结果：操作完成后结果应该是什么样的。

No.3

kirogiyi：最近发现一种现象，开发人员面对测试人员的时候，会展现出一种职业选手遇到业余选手的姿态，傲慢、理所当然，我觉得这是一种不正常的心理状态，应该怎么去管理？

宝玉：这确实是常见的现象，核心还是多一起合作多相互了解吧，让开发人员看到测试的核心价值，就是对测试方案的设计。

我对测试人员敬佩的地方不在于他们会写自动化测试，毕竟这个我写起来还更好，而是他们总能从我没想到的角度测试出来 Bug，从而帮助我提升程序质量。

可以安排一些开发人员和测试人员一起合作的事情，比如测试人员提供测试方案测试用例，开发人员按照测试用例去实现自动化测试，让开发人员明白，做好测试其实不是他们想的那么容易。

No.4

和：我的单位不是软件公司，但是企业内部也会开发一些业务类软件。可是从需求分析到设

计，开发，测试，部署，运维，都是我一个人的工作。这样的情况，如何工作效果会更好一些?

宝玉：对于这个问题，我觉得你可以自己先分析一下，你觉得目前哪些地方做的好，哪些地方可以有改进？可以从几个方面分析，比如：

工具：

有没有用源代码管理工具?
有没有用持续集成跑单元测试?
有没有用 Bug/ 任务跟踪系统?

开发流程：

多长时间一个项目周期，一个功能从开发到上线要多长时间？质量如何?
有没有做需求分析和确认？会不会做出来东西不是业务部门想要的?
有没有开发前先做简单的技术设计?
有没有写一些基础的测试用例?
有没有开发后自己测试和找业务部门帮忙测试?
线上的故障有没有一个合适的流程去处理?
有没有写一些基本的文档，如果来一个新人了能否接手?

技术实践：

有没有写自动化测试?
有没有用好的技术框架或开源组件?
有没有自动化部署?

分析出问题，知道哪些地方做的不够好以后，就可以有一个改进的计划 ，看如何将改进方案落实下来。

还有就是一个人开发，缺少向其他人合作和学习的机会，可以有意识地创造更多这样的机会，比如内部多和其他部门合作。外面可以参与一些开源项目。

No.5

hua168：现在不是流行测试驱动开发吗？先写测试代码再写实现代码，那写完再让专门的测试去测？

宝玉：测试驱动是一种很好的开发实践，但普及率也不算很高。可以看到自动化测试那一篇，测试驱动写的是单元测试，并不能保证不出 Bug，只是说能有效提升代码质量。还有就是开发人员测试自己写代码，很容易遗漏编码时就没考虑到的逻辑。

No.6

果然如此：对于自动化测试一直有些疑问，如果每次发布都对所有方法自动化测试，那么：

1. 一定会耗费很多时间；
2. 数据库产生很多测试历史数据；
3. 写测试用例能达到覆盖率高的写代码技巧，如边界测试代码、幂等测试代码如何实现。

宝玉：

1. 自动化测试确实会耗费很多时间。自动化测试代码通常是金字塔结构：

   单元测试（小型测试）代码最多，执行也最快，占总比例的 70% 左右，通常 1 分钟内；

   集成测试（中型测试）代码其次，执行比较快，占比 20% 左右，控制在 10 分钟以内；

   端对端测试（大型测试）最少，执行慢，占比 10% 左右。

一般 CI 里面跑单元测试和集成测试，耗时 10-15 分钟左右，其实还可以接收。

2. 跑自动化测试，数据库有不同策略。单元测试不访问数据库，完全模拟。集成测试只访问本机数据库，或者模拟的内存数据库，每次创建新数据库，或者使用完清空数据库。端对端测试，每次创建唯一数据（例如增加固定数据 + 时间戳），连接真实的测试环境，可以不清理数据。

3. 高覆盖率的关键在于，在写代码时就注意让代码方便地被测试。也不必过于追求 100% 覆盖，70% 以上我觉得就不错了。

No.7

宝宝太喜欢极客时间了：对测试这块一直很疑惑，测试脚本、测试用例、测试数据这三者如何配合一起通过 CI 进行自动化测试？

宝玉：是这样的，CI 本质上只是一个像流水线传送带，你的代码提交了，流水线传送带开始工作，你可以在传送带上面添加任务。

简单来说，你可以想象成 CI 的一个任务启动后，给你一个干净的虚机（实际是运行 Docker Container），然后帮你把当前代码下载下来，帮助你配置好运行环境，然后你就可以在里面安装任何软件、服务和运行任何脚本。

举例来说，你可以在传送带上增加以下任务：

1. 安装所有的依赖包；
2. 运行模拟服务（比如一个内存数据库）；
3. 运行单元测试；
4. 运行集成测试。

如果上面所有任务都成功了，那么这一次的 CI 任务就成功了，其中一个失败，这一次的 CI 任务就失败了，然后你就要检查什么原因导致的，然后修复再重新执行，保障 CI 任务成功执行位置。

No.8

一步：问一个有关 code review 的问题，code review 是指把代码放到大屏幕上大家一起看呢？还是类似 github 上的合并代码的时候发个 pr，然后另一个人对需要合并代码进行检查，检查通过后才同意合并请求？

宝玉：我觉得两种都算 Code Review，只是形式不一样。

通常 PR 这种 Code Review 应该是贯穿到日常开发过程中的，每个人都可以去 Review，有 1-2 个人 Review 通过就可以。合并的话不止是 Code Review 通过，还需要自动测试通过。

而大屏幕这种参与人多，成本高，属于偶尔针对特殊故障分析、学习研讨等目的才做的。

No.9

yellowcloud：我们目前有一个项目是做实时数据采集的，对方将实时数据推送给我们，基本上每天每个时刻都可能有数据推送过来。这样就导致一个问题，我们部署新的版本时，

他们的数据还在推送，这样就不可避免地丢失了部署过程中的数据，对方也没有重新推送的机制。请问，这种问题有没有比较好的解决方案，以解决更新版本时数据丢失的问题？

宝玉：这个问题其实不复杂，你可以将服务分拆，独立出来一个专门接受数据的服务，这个服务极其简单，只做一件事：接收数据，并存储到数据库或消息队列。

你原有的服务，改从数据库或者消息队列读取即可。更新部署的时候，接受数据的服务就不要轻易更新了，这样就不担心会丢数据了。真要更新，只要和对方协商一下，暂停推送就好了。

No.10

Charles：如果是瀑布流带点敏捷部分实践的开发方式，总觉得 UI 这个岗位工作量不怎么饱和，一个版本过来特别是小版本迭代，周期可能是两周，UI 可能 1 天就搞定了，其他岗位都差不多要全程跟下来，这个问题出现在哪里？

宝玉：这个问题有点不好回答，毕竟对你项目情况不够了解。

我觉得，如果 UI 这个岗位对你的团队来说是必须的，并且 UI 设计师很好地完成了他 / 她的工作，那么就很好，没有任何问题。毕竟有的人效率就是比较高，好过故意磨洋工看起来很忙。

如果 UI 这个岗位是可有可无，那么就可以考虑不设置这岗位，将工作外包出去，或者尽可能用一些标准的 UI，或者让前端工程师兼职 UI 设计工作。

No.11

yellowcloud：宝玉老师能不能介绍一整套可以简易部署使用的 devOps 的工具，方便小公司快速部署、实现，在实践中感受 devOps 的魅力。

宝玉：如果你要部署持续集成环境，可以先试试 Jenkins 或者 Gitlab CI。如果你要部署日志和监控系统，可以试试 ELK，也就是 Elasticsearch、Logstash、Kibana 三者的结合。网上可以找到很多安装使用教程。

No.12

一步：搭建自动化测试，自动化部署，自动化监控系统，都自动化了，开发都做了，是不是就不需要运维和测试了？

宝玉：自动化只是把重复的体力活做了。自动化测试的话，还是需要测试人员写测试用例才能有更好的测试效果；自动化部署和监控，也离不开专业运维人员的设计和搭建。

但是可以预见的是，以后低端的手工测试和运维岗位会被挤压的很厉害。如果你看大厂的招聘岗位，这些低端手工岗位都极少或者根本就没有。

No.13

邢爱明：我在传统的制造业甲方公司，系统大部分是内部管理系统，对于线上故障处理，一直有几个疑问:

1. 谁来主导线上故障处理的过程?
2. 故障排查是不是应该有一个标准的分析过程，让运维、开发、安全各方能更好的协作?
3. 便利性和安全如何平衡?

宝玉：

1. 谁主导线上故障，我觉得有两个指标要考虑：一个是这个人或者角色要懂技术懂业务，这样出现故障，能对故障进行评级；另一个是要能调动开发和运维去协调处理，这样出现故障能找到合适的人去处理，不然也只能干着急。
2. 故障排查上：如果是操作系统、数据库、网络等非应用程序故障，应该是运维负责；如果是应用服务故障，应该是开发去负责，即使开发最近没有去做发布也应该是开发去查。因为只有开发对于应用程序的结构最清楚，才能找出来问题。排查过程中，运维要给予配合。
3. 应该搭建起来像 ELK 这样的日志管理系统（可参考《38 | 日志管理：如何借助工具快速发现和定位产品问题？》），将应用程序日志也放上去，这样正常情况下就不需要去登录服务器了，直接就可以通过日志工具查看到异常信息。另外，一些特殊情况应该允许开发人员登录服务器排查定位。

No.14

Charles：目前只放了 nginx 日志到日志服务做一些简单的分析，还有其他什么日志是应该放到日志服务里的？有什么比较好的实践吗?

宝玉：我觉得应用程序的日志也应该考虑放进去，对排查问题很有帮助。应用程序的异常信息、错误堆栈非常有用，必须确保记录下来了。

举个例子来说，你的一个手机 App，一些特定场景下，某个 API 请求出错，而这个 API 可能背后会连接多个服务或者数据库，这样的场景下，光靠 nginx 日志是不够的，必须要有应用程序的日志配合才好定位。

你可以参考我在《37 | 遇到线上故障，你和高手的差距在哪里？》提到了一个 requestId 的实践。

精选留言

kirogiyi:
产品设计、软件开发、软件测试都应该对产品质量负责。

产品设计要重视产品需求的完整性，提升用户的操作舒适感，展现流畅的页面逻辑设计，这是产生良好软件质量的开端。在进行产品设计评审的时候，除了评审人员外，相应的软件开发团队和软件测试团队一定要派人员参加，不能坐等任务分配。有的开发团队和测试团队不去了解需求和产品设计情况，只是一味等待产品的 UI 设计，久而久之，就形成了少交流多看文档的习惯，于是大家就开始机械般各顾各的，做完了扔出去就好，就很难在产品质量上达成共识。

软件开发是核心，开发人员对产品的理解程度和自身的技术水平决定了产品的质量和迭代周期。如果开发团队不去与产品团队交流，不去与测试团队核对测试用例，那么在开发过程中大多只会去关注是否实现和能否实现，至于产品质量出现的问题，就不是他们关注的重点。然后就会自私地认为产品设计需要增删改是产品团队的原因，产品上线出现 Bug 是测试团队没有覆盖到，部门之间的战争就开始酝酿直至爆发。

软件测试更多的是发现问题、监督问题和约束行为。发现问题的重点在于通过完善的测试覆盖，去找到开发过程中的盲点，而不是去为别人的疏忽大意导致的错误埋单，比如：开发提交的产品有错别字什么的，这种错误就是开发负责人应该承担的。

在明确发现问题后，测试团队有权利去监督开发团队解决问题，直至问题得以彻底解决。除此之外，测试团队可以对开发团队行为进行约束，双方协作完成自动化测试体系和流程的构建，共同遵守规则：开发团队负责单元测试、集成测试和系统测试的代码编写，测试团队负责查缺补漏和必要的人工测试。

因此，个人认为，产品、开发、测试的紧密合作是保障产品质量的必备条件。

相关阅读：31 | 软件测试要为产品质量负责吗？

纯洁的憎恶：

解铃还须系铃人，要想提高软件质量，就要着眼于整个生产链条，每一个环节都要为提高质量出力，而绝不能仅仅依靠质量监控岗位或部门。相反，很多企业设置了类似的部门或岗位，并把质量、安全的重担压在他们肩上，但又没有赋予足够的权力去介入、影响整个链条，结果可想而知。不谋全局者不足以谋一域啊。

把整体质量按照生产链条或链条上的不同角色，划分为若干子部分。通过有机的把控各个子部分质量，形成合力，达到提高整体质量的目的。

相关阅读：31 | 软件测试要为产品质量负责吗？

纯洁的憎恶：

看来是否需要专职测试人员，在一定程度上需要视具体业务情境而定。不同的情境会有不同的异常情况和极端情况，需要有针对性的设计出完备的测试用例。而且在 Bug 修复后，也要保证修复本身没有 Bug。

所以测试也是一个系统性的工作，如果取消专职测试人员，不仅对开发业务水平要求更高，还需要项目自身的不确定性低一些。感觉有测试思维的发开人员，更有可能写出健壮的代码。

相关阅读：32 | 软件测试：什么样的公司需要专职测试？

yasuoyuhao：

代码就像程序员的名片，要对写出来的代码负责，最好的负责方式就是写测试代码，让每次代码变动，都不会影响到其他代码的运行，避免所谓的改 A 坏 B，节省迂回的时间浪费。也为 CI/ CD 做好准备，无论目前有没有。

程序员不想写测试在我们公司的原因大多是，不知道怎么开始写，不知道重点应该测试什么。先写测试的开发模式让他们觉得不习惯，但这些都是过程，培养良好的撰写测试代码习惯后，开发品质更有保证，提升开发效率，提升个人能力，我想都是有帮助的。

程序难免有 Bug，透过追踪软件，良好的管控 Bug 数量与修复进度，并且补足测试。

相关阅读：33 | 测试工具：为什么不应该通过 QQ/ 微信 / 邮件报 Bug?

纯洁的憎恶：

新手用野路子解决问题，高手用模型解决问题：

1. 给问题评级。紧迫的调动优势资源先解决，一般的往后放放。
2. 尽快恢复生产。生产是企业的首要职责，遇到问题优先恢复生产，减少直接损失，然后再正式地解决问题。
3. 找到问题出现的位置。“搜集证据”，通过“粗调”在时空上缩小包围圈，再用“精调”明确问题点，运用排除法最终锁定问题。
4. 分析原因，修复问题。
5. 提出解决方案。钥匙不一定插在锁眼里，要沿着问题的线索不停“倒带”找到根源。再针对根源，站在系统和流程的高度制定解决方案，避免问题复现。

重点：

1. 通过故障报警 + 业务骨干轮值机制，让正确的人第一时间响应问题。
2. 通过实战演习，确保应急预案稳定可行。
3. 通过使用日志记录和分析工具，积累、整理日常生产信息，出现问题才有得分析，否则重现问题也无济于事。

相关阅读：35 | 版本发布：软件上线只是新的开始

alva_xu：

线上故障，这是 ITIL 要解决的问题。我觉得最主要还是从三个方面来看。一是从流程上，对于事件管理、问题管理、变更管理、服务等级管理等，要有明确的流程。二是要有合适的工具，比如 ticket 系统，CMDB，监控工具、日志平台等。三是从人员组织来看，要有一线、二线和三线团队的支持，根据所创建的 ticket 的严重性和紧急性，给予不同 level 的支持。当然这也是目前流行的 devops 要解决的问题。

相关阅读：37 | 遇到线上故障，你和高手的差距在哪里?

alva_xu：

我觉得 scrum 方法中提到的两个会，可以作为项目复盘会内容的参考。Sprint 评审会议（Sprint Review Meeting）和 Sprint 回顾会议（Sprint Retrospective Meeting）。Sprint 评审会议在 Sprint 快结束时举行，用以检视所交付的产品增量并按需调整产品待办列表，是对工作成果的评审。Sprint 回顾会议是 Scrum 团队检视自身并创建下一个 Sprint

改进计划的机会。是对方法论的回顾和提高。项目复盘会也应该从这两个角度去做总结提高。

相关阅读：39 | 项目总结：做好项目复盘，把经验变成能力

邢爱明：
回想一下，项目复盘想要效果好，需要做一些准备工作:

1. 复盘会前，要求项目组核心人员对项目的情况先自己进行总结，包括做得好和做的不好的方面，有书面文件输出。先要有思考，复盘会上大家才有可讨论的内容，否则会议上大家可能就是随便说说，复盘会成了走形势。
2. 复盘会的会议主持人，需要有比较强的会议主导能力，尤其是参加会议的人来自多个部门的时候。因为大家总结项目中做的不好的地方，难免会涉及到多个部门或团队配合的情况，且每个人的描述也不可能做到百分之百的客观和公正。

如果有人认为总结的内容有问责的含义或需要自己承担责任，复盘会就很容易变成了甩锅会。这时候就需要会议主持人正面介入和引导，让大家讨论解决方案和改进措施，确保按照预定的议程开复盘会议。

相关阅读：39 | 项目总结：做好项目复盘，把经验变成能力

思辨时刻

成：
我们公司团队小，每次 app 开发完成后，要求测试人员组织开发全体测试 2 次，用于保证质量。团队小测试人员技术有限，性能，安全等一般难以保证。

宝玉：
其实即使是小团队，也应该加大对自动化测试对投入，绝对是磨刀不误砍柴工，这样 App 开发完成后，很多测试就可以自动化完成，节约时间和人力。当然在没有自动化测试的覆盖的话，这也是很好的一种测试方式。

相关阅读：31 | 软件测试要为产品质量负责吗?

毅：
项目负责人为软件质量总责任人。功能，代码，过程都要关注，并不一定要亲力亲为，因为

除了质量他还要兼顾范围、时间和成本。提升质量意识最理想状态是组员有质量人人有责的意识与行动，但实际上这很难。如果自下而上做不到，就自上而下用制度强推，有奖有罚。

最后补充一点就是推行质量保障是需要公司层面作为支持的，否则在推行过程中会有不少阻力，也许在强人项目经理的推动下，个别项目能做的很好，但心会很累。

宝玉：
确实还要考虑金三角的因素。软件项目，也并非一定要有强人项目经理，其实只要按照软件工程，踏踏实实做好每一个环节，质量就不会差到哪去。

比如说在需求上多花点时间精力，把需求确认清楚，这就成本一半了，然后再基于确定的需求做好架构设计再开发，最后开发后做好测试，那么质量就有了基本保障了。

相关阅读：31 | 软件测试要为产品质量负责吗?

邢爱明：
谁来做测试工作，这也是我一直比较疑惑的地方。我现在是甲方，主要做的是企业管理软件，业务逻辑和流程控制都比较复杂，部分系统是需要一些领域的专业知识。

软件开发的时候，基本采用的是瀑布模式。首先由专门的人员做需求澄清，分析和设计，一般我们称为业务分析师，他们这些人会输出软件的需求规格说明书，包括软件原型、详细说明 word 文件，然后交给开发团队进行功能设计、开发和交付。

在这种模式下，是否需要专职的软件测试人员，有两种意见。

第一种，不需要测试人员，原因是业务逻辑复杂性，找一个普通的外部测试人员进来，还需要花较长的时间去了解需求，学习如何操作一些专业的应用软件，还需业务分析师花费宝贵的时间去做辅导学习。如果不让测试人员学习，就只能测试一些很简单的功能，对把控整个软件交付的质量作用非常小。解决方案就是让业务分析师兼职做测试工作，因为对需求本身非常清楚，学一点测试基础知识，在付出点加班时间，也是能完成功能测试工作的。

第二种，需要专职的测试人员，因为上一种方案中，需求分析师大部分做的还是正向测试，即按照自己设计的功能和流程，判断软件交付是否合格。但是对异常场景的测试还是比较少的，会导致软件上线后，在用户实际操作过程和设想的不同的时候，往往会出现一些功能异

常，给用户的直接感受就是软件不稳定。所以说希望通过专业的测试人员，多采用一些探索式的测试方法，尽量多地发现软件中存在的缺陷，提升交付质量。

哪种方案更加合理一点？

宝玉：
我的观点是这种情况下需要专职测试的，业务分析师的重点应该是把需求文档写清楚。

业务复杂不能成为一个借口，想想看开发人员是怎么理解需求的，难道也是业务分析师代劳？肯定也是由业务分析师写成需求文档，然后开发人员基于文档开发，当然中间少不了很多确认环节。

测试也是类似，应该专业的人来做比较好，可以有更好的测试覆盖。一开始肯定是难一点，但是一段时间业务熟悉后，会极大提升整个团队的测试效率，而你也不需要再为这个问题纠结了。

相关阅读：32 | 软件测试：什么样的公司需要专职测试？

kirogiyi：
对于 Devops 我只是听说过，并没有具体的去了解过它的使用和应用场景。根据宝玉老师的讲述，Devops 的基础是自动化，那么自动化之外好像更多的是一种概念，可以因环境而产生各种不同的方式和方法，并没有比较明确的定论。感觉就像敏捷开发一样，满足敏捷宣言思想的操作都可以是敏捷开发，最终适合自己或团队的才是最好的。

宝玉：
自动化确实没有明确的定论，重要的是得要有应用自动化的意识，让自动化变成你项目开发流程的一部分，从而提升效率、改进质量。

应用自动化本质就是应用工具和基于工具二次开发，常用的自动化工具比如说自动测试框架、持续集成、日志监控报警。这些都是基础工具，还需要针对自己项目的环境，基于工具提供的 API，去定制化的写配置脚本，让它可以适合你的项目。

最重要的还是要把这些工具整合到你的开发流程中，比如说：

当你提交代码的时候，持续集成能帮你自动运行自动化测试脚本，可以直观看到测试结果，根据结果再决定是否合并或者继续修改；

当你合并代码后，持续集成能帮你自动化部署到测试环境，并且构建生成生产环境的部署包，甚至帮你自动部署生产环境。

当你要部署的时候，通过自动化的脚本直接部署到生产环境，而不需要手工去干预太多，避免人为因素的失误。

当你部署上线后，通过日志监控报警系统能实时看到部署后的数据变化，及时发现问题。

这样的流程其实是比较通用的、大部分项目都是适用的，只是前期需要投入一定的时间精力去研究和搭建，但是搭建好了这样的整套自动化环境和建设好了相应的开发流程，相应的从效率和质量上的回报也是很大的。

相关阅读：36 | DevOps 工程师到底要做什么事情？

好，今天的加餐就到这里，非常感谢同学们用心的留言，也希望我们专栏的同学都能每日精进，学有所成。

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。

上一篇　39 | 项目总结：做好项目复盘，把经验变成能力

下一篇　40 | 最佳实践：小团队如何应用软件工程?

精选留言 (3)

写留言

kirogiyi
2019-06-01

2

看到宝玉老师精心准备的每一期专栏，每一次认真而有效的回复，每一次“一问一答”的共享加餐，我都感觉无形中有了明显的进步，也对软件工程的深入理解充满了期待和信心，感谢宝玉老师。

展开

作者回复: 谢谢你的支持，也感谢你每一期精彩的留言分享🙏

yellowclo...

2019-06-03

从宝玉老师开设专栏，一直跟随着宝玉老师的脚步，使我对软件工程有了初步的了解，目前已经开始按照您的文章进行逐步实践，希望能够早日搭建一整套自动化的运维实施工具来解决项目的实际问题。

展开

作者回复: 💝能学以致用是最有收获的！

如果有具体问题也欢迎留言分享。

yasuoyuh...

2019-06-02

很开心留言也被精选了，感谢老师带来更广大的软件工程视野。

展开

作者回复: ㊗祝学有所成！