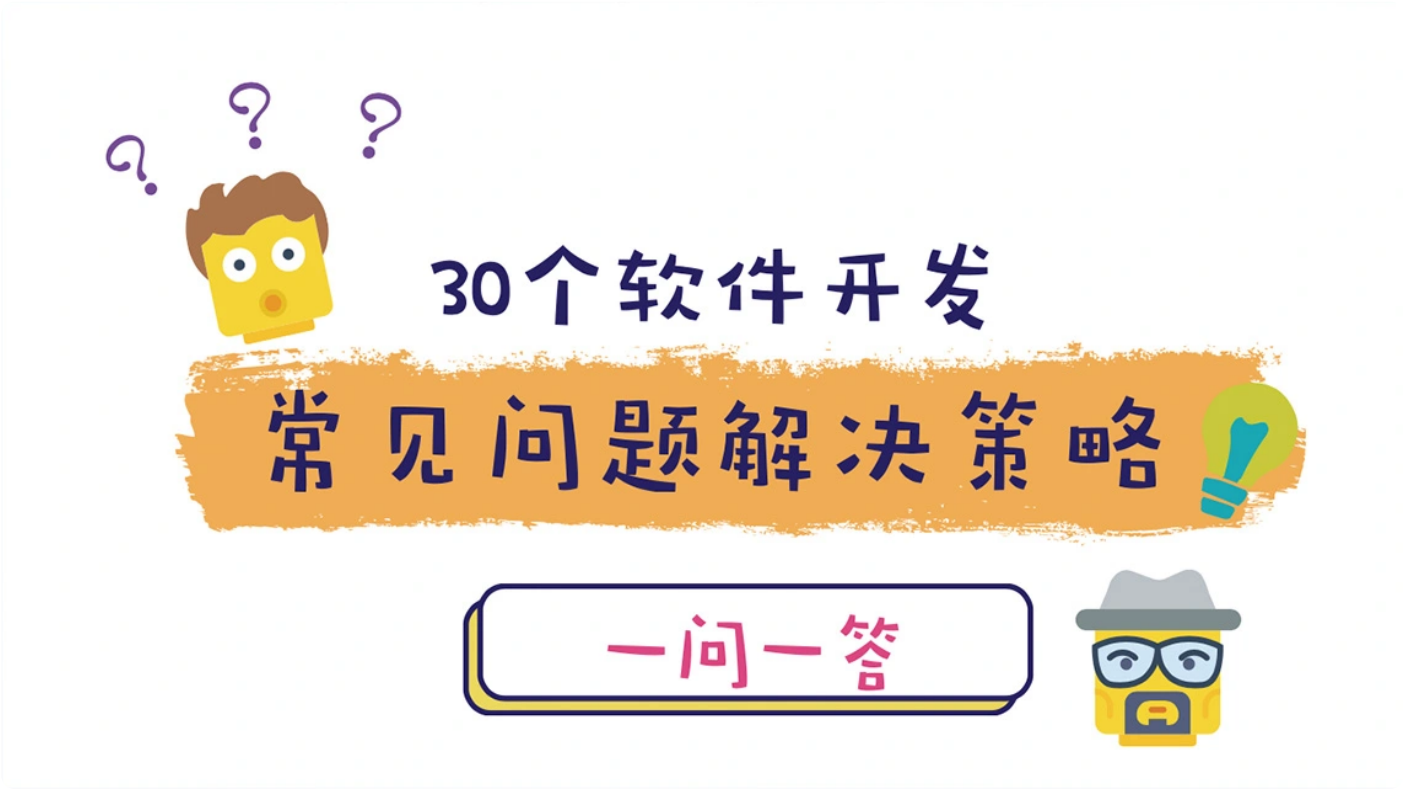
“一问一答”第1期 | 30个软件开发常见问题解决策略

2019-03-16 宝玉/专栏用户 来自北京

《软件工程之美》



你好，我是宝玉。专栏已经上线更新一段时间，看到同学们对软件工程有了更深刻的理解和思考，我很受感触。

有些人说走了很多弯路，日常搬砖，增删改查；也有的同学跟我一样“野路子”程序员出身，非常困惑希望建立自信；还有的同学发表上千字的学习心得，可以说非常用心。

编码的最终目的还是为了实现一整个软件的开发，在程序员的晋升之路上，总有单独挑大梁负责项目的时候。你会发现软件开发中的很多问题，都是可以通过软件工程的知识来解决的。

在已经更新的文章中，同学们经过思考，结合自己的工作实践场景，提出了非常好的问题。我们专栏的留言内容成为了专栏最好的补充。于是我就将留言板中的答疑和精彩留言进行汇总，方便你更好的查阅和理解专栏内容。

一问一答

No.1

hua168：学这个专栏需要哪些基础知识为前提的？开发都要学哪些基础东西？

宝玉：学习这个专栏，不需要你有特别的基础，当然有一些项目经验可以帮助你更好的理解。至于要学什么基础的东西，其实你可以从另一个角度思考一下：**开发的价值是体现在哪的？**

开发的价值是通过在项目中创造价值体现的，所以你要考虑学什么能帮助到你更好的在项目中创造价值。比如说除了具体的编程技能外，还可以从这些方面思考：

1. 提升对需求分析和理解的能力，这样你就知道要做的是什么，减少返工；
2. 提升架构和抽象的能力，能把需求抽象成架构设计，能把复杂的问题通过架构分解成简单的问题；
3. 高效率的编码，完成需求，等等。

No.2

hua168：软件工程在游戏项目上，是不是也一样呢？

zhxilin℃：对游戏行业的过程模型有什么理解？

宝玉：万变不离其宗。游戏项目一样离不开软件工程，游戏开发本身也是软件开发，只是有些名字换了，比如产品经理变成了游戏策划，产品设计变成了游戏策划案。游戏开发一样要有需求分析、架构设计、编码、测试等关键活动。只是游戏项目的需求变更频繁、节奏快，用增量或者迭代要好很多！另外也可以试试敏捷开发。

No.3

于欣磊：现在已经进入云计算时代，基本上大中小企业都在上云，复杂逻辑都在云端处理，真的还需要软件工程里讲的开发要搞这么多流程么？

宝玉：是的，云计算的兴起可以减少很多劳动，但不代表你就什么都不用做了，还是要做需求分析，再去做架构设计，做完架构设计你才能清楚哪些可以用云计算，那些需要自己去实现。最后编码完了，一样还要测试的。

No.4

hua168：现在运维的前景怎么样？感觉竞争很激烈，很多小公司都不招，开发兼职了运维，各大云出了一些维护监控工具，对他们来说够用了，感觉发展空间变小了，运维开发招也少了……也有人提到运维职业会消失，难道要转开发？那运维开发能力也争不过真正的开发啊。

宝玉：你这个问题很有代表性，现在云服务兴起后，传统运维的职位在减少，所以 DevOps 在兴起。DevOps 和运维的主要差别就是 DevOps 不仅有运维能力，还有开发能力，可以站在运维和开发更高的角度去看问题，帮助自动化的，稳定的交付部署产品。你不用完全转开发，但是应该要学习一些开发知识，尤其是自动化脚本相关的。

No.5

行者无疆：传统瀑布模型前期进行了完整的需求评估，在技术选型，系统架构，实施路径上可以做好全面的规划，虽然周期长，不必要的反复工作会少很多，目标也更容易控制。

那敏捷模型的迭代方式并不会把需求都考虑全面，未来的迭代可能会造成前面的技术架构或者实施细节等都不能满足新需求的要求。所有工作都要重来的问题，会存在大量的重复工作和资源浪费。 敏捷模型是如何有效地规避这些问题的呢？

宝玉：你说的问题确实存在，导致常说的技术债务问题，所以需要定期去重构，改进这些问题。迭代过程中的重复工作确实存在，但是软件开发中的浪费其实主要不是在于迭代过程中的重复工作，而是在于需求不明确和需求变更导致的返工或失败。

敏捷开发持续发布稳定版本的理念还是利大于弊。有一些项目其实是瀑布模型和敏捷开发的结合，需求分析和系统设计的时候用瀑布模型，开发和测试阶段用敏捷，也是个不错的选择。

No.6

Geek_85f782：如果说软件工程 = 过程 + 方法 + 工具，其中过程是否就是具体指软件的生命周期？方法是指选用的生命周期模型，比如瀑布、螺旋、迭代、敏捷？

宝玉：我认为过程应该包含过程模型采用的方法。而方法是指基于过程模型之下的方法。因为过程模型决定了软件开发过程是什么样的，进而决定了采用什么开发方法。

比如你选择了瀑布模型，整个软件开发过程就是按照瀑布模型的分阶段来进行，对应的方法就是瀑布模型中的方法，例如需求分析、架构设计；如果你选择了敏捷开发，则整个开发过程就是一种敏捷迭代方式，后面的方法对应的就是敏捷开发的一套方法体系，例如 Scrum、用户故事、持续集成等。

No.7

hua168：中小公司，开发人员流失严重，如果像工厂流水线那样，即使核心开发人员全部走了，新招的开发在没有人带的情况下，能继续接上开发……除了制定规范，开发文档之外还有哪些措施？

宝玉：软件开发，核心就是人，如果没有人，规范和文档都没意义的。要留住人，一个是得舍得给钱，另一个得有个好的环境，还有就是要有梯队，能把新人培养上去。饭店里只有一个大厨，大厨当然敢乱提要求，如果大厨多几个，就不担心了。还是得要舍得下本钱招优秀的人。

No.8

白发青年：如果是外包项目，作为项目的乙方，如果采用敏捷开发，最初的工作量就很难完整估计，不利于双方的合同签订。不知老师是否有好的建议？

宝玉：这个问题通常有两种解决方案供参考：

1. 你按照瀑布模型的方式去估算工作量，然后签订合同。开发的时候你需求分析和架构设计还是用瀑布模型的方式，但是编码和测试用敏捷开发。这是一种不错的折中方案；
2. 你把所有需求拆分成用户故事，对用户故事进行打分（了解下计划扑克之类的打分方案），然后可以算出来一个总分数。另外按照你以前敏捷开发的经验，可以知道每个 Sprint 大概

能完成多少分，这样你就能大致推算出来工期。

No.9

Charles：瀑布模型非常考验人的能力，会造成互相扯皮推卸责任，上线以后有什么问题，还会互相推锅背，这种情况下管理者有啥好的方式去解决？

宝玉：虽然我觉得甩锅不是什么好事，但是如果你真要甩锅，最简单有效就是设置流程去划分责任。上线后有问题其实很正常的，重要的是要有合理的机制：

1. 及时发现问题，监控报警、用户投诉反馈等；
2. 马上解决问题，对线上版本有专门的代码分支，可以随时打补丁修复，测试上线；
3. 避免后续再犯同样的错误。要分析原因，看什么导致问题，然后改进流程。

No.10

Linuxer：小公司有很多项目就是一两个人，没有那么多角色，怎么做到按这种流程去开发项目呢？比如常常在代码编写过程中发现很多问题都没考虑全面又感觉在交流需求的时候根本就没想到，要怎么在之后的项目中不再犯这种问题呢？

宝玉：即使只有一个人，建议也要做简单的需求分析和设计，做完后，形成简单的文档，找人评审一下，提一些意见。因为你写文档的过程，给别人讲的过程，其实是在帮助你思考，帮助你梳理清楚逻辑，避免在实现的时候发现好多问题没想清楚。

还有一个思路就是快一点迭代，每一个迭代解决优先级最高的问题，然后下一个迭代中改进上一个迭代的问题。项目中犯错误其实很正常的，重要的时候要总结，看看通过什么方式能改进，避免犯类似的错误。

No.11

bearlu：是不是每种模型，有其应用场景，不能只追求最新，要适用才行？

宝玉：你说的太对了！举个例子来说，敏捷开发肯定又新又好，但是如果成员没一个懂敏捷开发，强行照葫芦画瓢，可能结果还不如用瀑布模型。

No.12

clever_P：瀑布模型难以快速响应需求变化，那有没有可能通过对业务领域的深入研究，对业务的发展和变化做出一些前瞻性预测，在软件设计的时候将这些预测考虑进去，以此来减小后期需求变化对整个项目的影响呢？

宝玉：你说的是一个方向，但是要预测其实是很难的，结果很可能是过度设计，设计了很多可能最终完全用不上的架构，反倒不如快速迭代快速响应来得实际。

No.13

一年：开发一款测试市场反应的产品，使用快速原型模型是不是好点呢？

宝玉：恐怕不行，因为快速原型模型是牺牲质量的。质量差的软件，去测试市场，你不知道是因为质量问题不行还是需求没抓住不行。这种情况，可以考虑用迭代模型，先开发核心需求，然后再逐步迭代。

No.14

凯纳软件：感觉自己之前做任何事情都没有章法，觉得只要做了就可以。通篇学完之后，知道自己哪里欠缺，应该怎样去学习及工作。

宝玉：谋定而后动。还有一点经验就是：如果你想更有章法，更有大局观，做一件事情前先做个计划，可以帮助你更好的思考，也更容易执行。

No.15

Joey：我们公司是比较大的国企（多个业务部门对一个开发部门），对质量要求较高，现在业务条线也比较多，业务部门基本都嫌我们开发部门效率低，对于我们研发部门，组织架构还

是按照瀑布模型设计的，开发模型基本是迭代 + 增量，如果想推行敏捷，肯定需要调整组织架构，一旦调整，就会触发一些利益关系，在这种背景下，有没有什么好的招数，既可以提高研发效率，又可以保证质量？

宝玉：如果你想推行敏捷，可以先找个小项目，组个小团队试点，成了可以作为一个参考，领导可以去邀功，以后可以更大规模尝试；失败了也损失不大，领导也不用担责任。

不管用不用敏捷开发，你都可以学习其中好的实践，例如持续集成用起来，帮助你高效的集成部署；自动化测试代码写起来，帮助你提高项目质量；迭代快起来，以前 3 个月变成 1 个月，以前 1 个月的变 2 周。有些事情即使只是程序员都是可控范围内的，做着做着其实你就“敏捷”起来了。

No.16

一步：最小可行性产品 MVP 应该就是迭代开发了？

宝玉：MVP 更多的是需求定义上的概念，和开发模型并没有关系。但是你使用迭代开发或者敏捷开发，必然要优先选择最核心最重要的功能需求先开发。所以通常 MVP 的方式选择核心需求，用迭代模型或敏捷开发开发需求。

No.17

龙哥：有依赖交叉的用户故事应该怎么做，比如用户系统的数据库该由谁搭建。毕竟注册、登录、修改这些都可能基于一个数据表。表字段这些需要统一，不能一个程序员改一次字段名吧

宝玉：敏捷开发中有一个迭代 0，也就是第一个迭代，就是做这些准备工作、基础架构搭建的。敏捷团队小，有个好处就在于遇到你说的这种情况，在做之前，大家都在一起开个小会一商量就可以定下来了。

No.18

阿神：敏捷开发里开发也要写集成测试用例吗，那么测试人员主要做手工测试？

宝玉：对，开发不仅要写单元测试，还要写集成测试。但开发都是用模拟数据，假的 API。而测试的自动化测试会用真实的数据，调用真实的 API，而且也要做一部分手动测试。至于比例多少，还得看项目特点。

No.19

holylin：如果合同金额一开始就是根据商务阶段了解的情况评估的工作量而确定的，那么在合同执行过程中，如果按敏捷开发的思路，客户不断改需求我们不断地响应，然后工作量甚至已经超过了原先合同的金额，这个时候要如何处理？

宝玉：这是个好问题，我对这个问题上没有什么经验，但我可以试着帮你分析一下。

你的合同是按照当时的需求签订的，如果后期客户变更需求或者增加新需求，那相当于需要重新签订变更这部分的补充合同。

应用敏捷开发的时候，你也可以让产品经理或者项目经理充当客户的角色，这样他们会更偏重产品需求的解读，而不是重新提出新的需求。还有一点，合同执行的时候，这时候你不需要太过于纠结是不是用敏捷还是迭代还是瀑布，而是哪一种开发模式，可以让你高质量高效率的完成，那就是最好的最适合你的开发模式。

No.20

长眉 _ 张永：作为一个电商 ERP 服务商，既要关注产品的研发进度，又要对产品做维护。人员一旦离职，发现没有较为详细的文档，就需要去猜测，之前的业务了。敏捷后上线，留下的技术债务应该归谁负责呢？

宝玉：敏捷还是要写必要的文档，只是会简化。尤其是这种涉及交接的、维护的，文档不能省。技术债务应该团队成员集体负责，大家在迭代计划会上应该将技术重构列入后续的 Sprint。

No.21

刘晓林：敏捷开发这么强调扁平化，这么重视人，这么强调开放而弱化约束，那和最初没有软件工程时期的开发主要区别是啥呀？

宝玉：好问题，你难倒我了。前面介绍过，没有软件工程的时候呢，开发就是边写边改模式，没有需求分析、没有架构设计、没有测试，就导致很多问题。

No.22

邢爱明：对于企业管理的软件，核心需求涉及多个部门，需要反复沟通确认周期很长，这种情况下是否还适合使用用户故事的方式做需求分析呢？

另外，我按照瀑布开发模式的习惯分析，开发人员和 po 沟通需求后，如果没有文档作为输出物，在开发和测试的时候就没有标准，反而会造成工作返工。这是否意味着，团队成员需要高度的协同和配合？以完成任务为导向，而不是强调各自的分工？

宝玉：好问题！敏捷开发这种方式，需要客户紧密配合，也就是可以方便确认需求，否则还是少不了要写需求文档。另外我在文章中描述用户故事，有些描写不清楚或者歧义的地方，其实用户故事还应该包括验收标准，这样可以解决你说的开发和测试没有标准的问题。

团队成员需要高度的协同和配合那是一定的，尤其是架构和需求两部分。需求简化后，就意味着开发过程中需要反复沟通确认；没有专门的设计阶段，也就意味着每个 Sprint 开始前，团队要商量有没有要设计或者修改架构的，有就需要有个简单可行的方案对架构进行修改。如果各自分工，这样的目标就很难达到。

No.23

D：在敏捷开发过程中如何保证业务的传承？当有新同事加进来，如何让他快速的熟悉整个业务。

宝玉：这个是个好问题，也是个大问题！通常我的经验是：

1. 团队要有自己的知识库或 WIKI，常用的知识要花时间整理上去，这样新人来了可以自己查；
2. 先给他简单的任务，再慢慢稍微复杂一点，给予必要的指导，做中学是最快速有效的；
3. 遇到一些典型的问题可以通过结对编程的方式带着一起做。
   仅供参考。

No.24

dancer：对比瀑布模型来说，敏捷开发在需求分析和软件设计上要薄弱一些，这会导致越向后迭代，软件越难以变更和维护，请问老师有什么好的方法和建议吗？

宝玉：需求分析是在 Sprint 进行中同步进行，也就是开发具体的用户故事之前要和客户或产品经理充分沟通了解需求。如果用户故事不是特别大，这并不是很大的问题。另外并非只能用用户故事，也可以用传统的产品设计文档代替用户故事，也一样是很不错的实践。

对于架构设计，架构只设计当前迭代的，所以迭代到一定阶段，是要考虑重构的。通常重构代码也是 Sprint 的工作任务的一部分。

No.25

Dora：瀑布对人员要求不高 (各自负责各自的工作，比如需求只管需求)，而敏捷流程，一个人什么都要过一遍。这样理解，对吗？

宝玉：瀑布对人员也不说要求不高，但分工确实更细一点，比如像你说的，需求只管需求；开发一般就不操心怎么测试，写完等着测试报 bug；敏捷开发里面，分工没那么细，需求不仅要写需求文档或者用户故事，还要和团队成员紧密合作，及时讲解需求；开发也要自己写很多自动化测试代码；敏捷团队也不是没有测试，但是会用自动化测试分担一部分测试任务。

No.26

Tiger：在敏捷里面，开发写自动化脚本测试，那是不是就不需要测试这个角色了啊？感觉在敏捷里面，只需要开发这一个角色就可以了啊？

宝玉：在《07|大厂都在用哪些敏捷方法？（下）》我有谈到这个问题。自动化测试是辅助的，还是离不开人工的测试。而且开发写的集成测试和测试写的自动化测试还是有一点差别的，一个是用程序模拟操作的固定数据，而测试用的是真实的数据环境。举个例子来说，网页的自动化测试，开发只会用 Chrome Headless，数据都是事先写好的模拟数据；测试的话会用主流的 Chrome、Safari、Firefox、Edge 分别测试（自动化或手动），数据都是测试环境的真实数据。

No.27

一路向北：对于小公司小团队的项目，因为项目经理，产品经理都是身兼数职，是否有更好的实施方式呢？

宝玉：项目经理、产品经理兼多个项目是正常的，也没大问题。但是让程序员同时兼做开发和项目经理工作就很不好，因为项目经理需要更多全局掌控，而一旦要花精力在开发上，很难跳出具体的开发工作，会极大影响项目管理工作；项目管理工作也会频繁打断开发，造成进度延迟。

所以我建议应该有专职的项目经理，不应该让程序员兼职项目管理。新旧项目交织并不是问题，可以放在一个项目一个 Sprint 里面一起管理，也就是同一个 Sprint 里面有维护的 Ticket，也有新需求的 Ticket，只要保证开发人员同一时间只是做一件事，而不要几件事并行，就可以最大化发挥敏捷优势。

No.28

天之大舒：怎样培养团队成员？

宝玉：有一些建议仅供参考：

1. 招人和开人都很重要，招优秀的，开掉没有责任心，没能力的。这两点都不容易做到，不过得坚持做；
2. 设置合理的流程，配合一定的奖惩制度；你奖励什么，团队就会往哪方面发展；

3. 团队要有梯队，不能都是资历浅的也不能都是资深的，保持一个合适的比例是比较健康的；
4. 实战中锻炼，实战中磨合；给他们有挑战的任务，给予合适的指导（这就是有梯队的原因，需要高一级别的待低一级别的）。

No.29

星星童鞋：请问老师，对于需求更新极快，基本上每周都需要迭代更新上线的项目，在架构设计和项目部署上会不会有什么特殊的要求？

宝玉：架构设计上，一定要定期需要重构，优化设计，不然后续新需求效率会降低，包括代码上也会越来越臃肿。比如我现在所在项目组，每 1-2 年会有一次大的架构升级调整，日常每隔几周会有小的架构优化，这样基本上可以保证快速迭代不会受太大影响。

部署的话，一个是要自动化，可以快速方便的部署，另外一个部署后，需要有配套的数据监控和高于阈值报警的机制，因为上线后可能会有严重问题，需要及时发现，及时处理。

No.30

alva_xu：如果一个迭代里没有评审会，怎么知道我上线的系统是符合要求的？

宝玉：没有评审会，但是有专职测试针对最初提的需求进行测试，另外产品经理也会验收，如果验收不合格会提交 Ticket。也就是说是有验收，只是没有专门的会议。

精选留言

阿杜：

软件过程不是搞科研，不是搞艺术，而是解决多人合作将一个想法落地的学科，其中包括严谨的过程步骤、规范，用于提高效率或防范风险的工具。软件工程的主体是工程，这就要求我们具备基本的工程思维：模块化思维、抽象思维；具备一些关键的意识：质量意识、风险意识、交付意识。

相关阅读：01 | 到底应该怎样理解软件工程？

alva_xu：

对于大型系统的建设，可否用敏捷方法来实现，一直是个问题。

敏捷方法，适合于小团队（比如两个披萨团队）、小架构。对于大型单体应用的开发，至少在架构设计上是不适合用敏捷迭代方式的。

为了解决大型系统建设的迭代开发、快速交付问题，业内不断在探索。随着微服务架构的提出，以及容器技术的成熟，和 cicd 的实现，单体巨石应用被拆解成分布式的微服务应用，此时，敏捷方法也就开始真正大行其到了。

所以，微服务、容器、devops 这三剑客和敏捷方法一起，互为依存、互相促进，成为了软件工程中最有生命力的技术工具和流程，使软件开发在质量和效率上得到极大提升。

相关阅读：01 | 到底应该怎样理解软件工程？

老张：

在今天没有不可替代的硬件，却有无数不可替代的软件。硬件早已不是共享的壁垒，而曾经被认为有很强可塑性的却已经是最硬的壁垒。一台服务器、一块磁盘、一根内存以及交换机、防火墙等网络设备，更遑论鼠标、键盘、显示器，在冗余、复用、虚拟化等等技术之下，更换、替代、扩容如此之方便，经过简单培训的工人就可以轻松完成。

可是即便是美国国会图书馆，依然认为纸质是保存资料最好的方式，因为大量资料电子化后存放在不同介质，需要当时定制的软件才能读取这些格式。今天的软件就是这么硬。也许有一天，有人会写写如何开发真正的软件。

相关阅读：01 | 到底应该怎样理解软件工程？

阿银：

软件工程的本质在于工程。利用工程理论来保证高质量软件产品的产出。工程讲究效率，成本，质量，除此之外，容易忽略的是工作量与效益的权衡，这一点尤为关键。

相关阅读：01 | 到底应该怎样理解软件工程？

hyeebeen：

软件工程的产生源自于对高效产出可靠稳定的软件产品的需求，在各种“+”的现实生活中，不掌握合格的软件工程管理技巧，日常的项目工作会很容易有瓶颈。

学习对应的工程技巧，内化为自身素质，在项目过程中既能预防工程风险，也能建设面对风险的反应机制。这种人，各个企业都喜欢。

相关阅读：01 | 到底应该怎样理解软件工程？

阿杜：

1. 做任何事情都要按照一定的理论指导来，例如，依靠系统化、结构化的“工程思维”，将生活和工作中的每个事情都看做一个项目，可以提高做事的成功率和效率，虽然不用这些理论指导也能做成事情，但是相对来说是偶然性的，不是常规性的。这就是常说的认知（意识）先行，持有高级的认知去跟低认知的人竞争，是一种降维打击。

2. 工程思维的核心有两点：系统化，也就是全局观，要从站在整个项目的高度去看问题，不能做井底之蛙；结构化，也就是有步骤、有节奏得做事情的意识。

3.《软件工程之美》这个专栏，我给自己定了一个小目标：全部跟完，并且坚持留言跟老师交流想法。我制定了简单的阅读步骤：（1）文章至少阅读两次，第一次通读，第二次做笔记摘抄、整理文章的思维导图、提出自己的想法；（2）整理自己的学习心得，形成阅读笔记发到自己的博客（公众号）上。

4. 课后思考：我今年年初开始运营自己的公众号，我把它当做一个项目，就从下面几个方面进行了思考：我要提供的内容和定位是什么样的、我的用户是谁、我应该如何去运营；这些东西

想好后，我就将要做的事情拆分为：公众号设置、文章内容输出、运营推广三块，然后按照一定的步骤去执行，现在公众号的设置已经基本完成，整个项目进入内容输出和运营推广的循环中了。站在项目的角度去看这个问题，可以让我在动手执行的时候更有方向感和节奏感，也会对自己清楚自己某个小的点做的改动会对全局产生什么影响；在没有使用这个角度去看问题之前，我只是简单得主张内容才是核心，但是不懂运营和推广，没什么章法。

相关阅读：02 | 工程思维：把每件事都当作一个项目来推进

起而行：

项目思维的两个关键在于：

注意局部任务与总体时间的关系；

用熟悉的办法解决问题。

以留学为例。

1. 在距离留学申请还有两年的时间，可以先做不确定性强，见效慢的事情，不是不重要，而且短时间内会来不及。比如未来职业规划，兴趣的培养，长期的科研与项目。

等到了申请还有半年的时候 要注意局部的任务与总体时间的关系。那么确定性不强的任务效果将不会特别好，而刷语言考试的成绩，这种确定性强，时间可控，反馈见效相对快的事情就要提上日程。

2. 用熟悉的办法解决问题。在留学，这种高度信息不对称的领域，可以自己试着了解前人经验，但我认为，专业的事情给专业的人去做，那么找中介辅助申请就是个好的主意。

相关阅读：02 | 工程思维：把每件事都当作一个项目来推进

alva_xu：

工程方法就是有目的、有计划、有步骤地解决问题的方法，而工程思维就是用工程方法解决问题的思维模式。这种思维模式，首先要求有全局观。

而事实上，由于工程中的不同职责分工，导致各个角色有可能只从自己的分工角度去考虑问题，这实际上是软件工程中最大的障碍，也是传统的 CMMI（过程域的规范化）、现在的 DevOps 和敏捷方法想要去解决的问题。

正如《凤凰项目》中的观点，既要有自左向右的工作流，又要有自右向左的反馈流。通过人员和组织（自组织）的调整、工具和技术（CICD 工具）的使用以及流程（Scrum 等敏捷工作流程）的推广，确保整个工程项目不会被不同角色割裂开来，从而确保工程的实现。

相关阅读：02 | 工程思维：把每件事都当作一个项目来推进

纯洁的憎恶：

工程方法不仅给团队提供了一系列成熟的理论范式与实践工具，提高效率与成功率。更重要的是把团队的视角“强行”抬到全局高度，避免我们紧盯着自己关心局部问题，更好的统一思想、形成合力。

相关阅读：02 | 工程思维：把每件事都当作一个项目来推进

hyeebeen：

非常赞同“一切即项目”的思考模式，作者经历的培训活动我也经历过，确实在时间意识上会比没受过工程训练的人强一些，更注意过程控制。

ps：学习自动化测试其实不等于一定能缩短测试周期，“测试周期”的定义如果是测试独占的项目时间段的话，可以通过测试前移，加强自测，契约优先的接口自动化测试等来缩短独占时间。没有系统或者不够工程化的自动化测试脚本，反而会增加测试时间。

相关阅读：02 | 工程思维：把每件事都当作一个项目来推进

纯洁的憎恶：

我对瀑布模型感触颇多啊！

> 瀑布模型把复杂的软件生产过程，按照时间线索，切分为若干较为独立和专业的部分，条理清晰。
>
> 在每个阶段内只需要集中精力于阶段任务即可，不用胡子眉毛一把抓。
>
> 每个节点有交付件，过程可控、权责清楚明白。

瀑布模型特别符合我所在的大型央企的性格。但是我经手好几个项目，也被瀑布模型折腾的死去活来。比如我现在正在处理的项目。

首先，从可行性分析、立项、批预算、采购建设单位就花了一年多的时间。可行性分析做了 1 个月，立项流程走了不到 1 个月，批预算的时候，主管业务的领导变卦了，要求重新做可行性分析，于是我们又花了 1 个月。

二次立项的时候，主管信息的大领导突然决定要把区块链和人工智能等热门技术加进去，于是又要求重新立项。但是我们要做的事情实在和区块链八竿子打不着，死去活来的找了个理由扯上关系了，来来回回又花了 2 个月，这就半年了。

再次走到批预算的环节，主管领导发现原来的预算干不了这么多事情，又不同意增补预算，于是继续扯皮。经过多番协调，总算解决了，这时候叶子已经黄了。

我们立刻开展需求调研，但各个需求部门和最终用户都借口工作太忙不搭理我们，我们只好自己憋需求，简直是闭门造车。等需求憋出来了，大领导把需求部门都叫来议一议，结果被集体口诛笔伐。

大领导怒了，强令需求部门专门抽时间参与需求调研，各部门也是不情不愿啊，效果可想而知。需求总算审查通过了，就在我们准备采购实施单位的时候，国资委红头文件一直下来，公司的采购流程发生重大调整，项目被硬生生搁置下来。眼看年根了。

第一年就这么过去了。等来年采购流程也理顺的差不多了，预算又出问题了。去年批的预算只能去年用，不允许跨年。只好等到年中调整预算，又小半年过去了。采购流程走完，实施单位也很够意思，不等合同签订就投入工作。我们用极短的时间完成了软件设计，并且开始如火如荼的开发工作，此时又到了金秋。

就在这时，新的纪检书记上任了，他对我们的系统设计很不满意，要求相关部门限期整改，于是需求大调整，可是这会儿编码已经进行了 1/3 啦...之后就是上线日期一推再推，从 10 月初推到 10 月底，再推到 11 月、12 月，眼看又要跨年了。我们和实施单位连续半年 997，总算看到上线的曙光，这时候公司一把手退休了...

新领导上任后对整个流程极不满意，否定了纪检书记的指示，于是我们又开始第 2 伦大调整。现在已经是项目的第三年了，我们依旧没能上线。整个团队都要累趴下了，全公司一点成果也没看见。

相关阅读：03 | 瀑布模型：像工厂流水线一样把软件开发分层化

纯洁的憎恶：

稳定、可靠、一步到位的瀑布模型，不太适用于违约风险大、需求不明确、快速见效的场景。

快速原型模型：不见兔子不撒鹰。期初不考虑质量、架构，用最快的速度见效，并向用户确认需求。经过几轮直观、快速的反馈，把需求确定下来。接下来，既可以抛弃原型用瀑布精密重构，也可以在模型基础上完善。优点是快速有效地确认需求。不足难以有效应对后续的需求变更。

增量模型：分而治之。将大系统横向拆分成相对独立的若干小模块，每个模块采用瀑布模式分批次交付。优点是较快见到成果，且能够及时了解项目进展。不足是存在需求明确、系统可拆分、交付可分批等适用条件。

迭代模型：罗马不是一天建成。把软件项目纵向划分成若干阶段，从核心功能入手，逐渐深化、细化，直到满足用户的全部需求。每个阶段都是一个瀑布，都要在前一阶段成果基础上加

工、打磨。优点是快速满足基本需要，并体会软件演进的快感。不足是需求演化具有不确定性，会导致代码冗余、系统重构风险、项目周期不可控。

我做甲方管过不少外包项目，大 V 模型再熟悉不过了。整个过程冗长繁琐，走流程比建软件更累心。而且等项目结束的时候，需求早就变得面目全非了。乙方只能硬着头皮做，不然连业绩都没有，真是血本无归。在增量或迭代模型的每次交付后都做一次风险评估，演进为螺旋模型，可以及时止损。

项目做成这样，更深远的原因是业务都是在摸着石头过河，需求不变更才怪呢。但每年几个亿的信息化预算还是非常诱人的，投标单位络绎不绝。RUB 看起来不错，但需求快速演化会依然带来无法回避的系统重构压力，终归还要具体问题具体分析。

相关阅读：04 | 瀑布模型之外，还有哪些开发模型？

西西弗与卡夫卡：

当前不够明确、后期可能有较大变化的需求，准确说首先要考虑的不是用哪种开发方法，而是最好避免一开始就投入开发资源。开发的代价非常高，推倒重新开发的代价更高。最好是先想别的办法，验证需求是否真实存在之后再动手写代码。

相关阅读：04 | 瀑布模型之外，还有哪些开发模型？

alva_xu：

对于增量或迭代开发，大型企业需要考虑这些不适应点：

> 大型官僚机构的办事程序和敏捷过程不匹配。比如开发想敏捷，但财务采购等都不敏捷。代码敏捷了，基础环境不敏捷等。
>
> 伴随增量的添加，系统结构会逐渐退化。特别是对于大型系统，其系统结构的建设，就需要提前制定计划，而不是增量开发。

与开发方的合同问题，需要新形式的合同。旧形式的合同是固定合同，做多少事拿多少钱都在合同时谈好了，不适应工作量的变更。

相关阅读：04 | 瀑布模型之外，还有哪些开发模型？

纯洁的憎恶：

流程、工具、文档、合同、计划都是工业化的标志。它们带来了稳定的质量、惊人的效率、超大规模的协作，对于软件工业也是如此。

然而软件工业具备轻资产、知识密集型、从业人员素质高等特点，充分发挥人的创造力和价值，是其相较传统工业更高阶的要求。加之软件工程面对的不确定性与复杂度更显著。于是“个体和互动高于流程和工具，工作的软件高于详尽的文档，客户合作高于合同谈判，响应变化高于遵循计划”的敏捷思想应运而生。

通过用户故事，理解用户需求；在迭代中采用渐进的架构设计；定期重构解决技术债务；功能开发的同时编写自动测试代码；自动化持续构建。

由于淡化了部分工业思维中兼顾稳定、质量、效率、成本的传统手段，敏捷思想的最终落地，需要素质极高的从业人员参与其中，且数量不宜过多，以此来弥补流程上的缺失。同时要团队与客户紧密协作，上级的充分信任，才能够有效发挥其灵活应变，又万变不离其宗的优势。

这是大胆的返璞归真，好似回到了瀑布模型前的蛮荒时代，实则是更高级的打法，就像独孤九剑一般。所以，敏捷开发“道”的属性更浓。

敏捷开发具有快速迭代、持续集成、拥抱变化等诱人的特点，但也有苛刻的条件要求。不过，即使无法推行完整的敏捷开发，依旧可以在传统模式下，有针对性的应用敏捷开发的实践方法。

相关阅读：05 | 敏捷开发打底是想解决什么问题？

alva_xu：

我们现在着手的一个项目，是一个软件框架建设项目，外包给供应商做的。在签合同时，基本需求已经梳理得差不多了。所以按理是可以采用瀑布式开发来进行的。但由于以下原因，所以我们结合了增量开发和 Scrum 项目管理的模式进行系统建设。

1. 基本需求是可以分模块来实现的。
2. 我们这个项目所依赖的其他部门提供的基础平台也不是一次性可以交付我们使用的。
3. 我们的使用方（另外一个应用项目）对我们项目的时间要求很急，但可以接受我们分批次交付的模块。

基于以上原因，我们设立了几个大的增量阶段，每个增量阶段我们有分几个 sprint 来进行开发管理。到目前为止，进展还比较顺利。

但由于我们这个框架建设项目的外部干系人比较多，所以在协调上游平台和下游应用系统的时候，确实遇到了许多沟通方面的问题。由于其他项目没有进行看板管理，所以需要进行例会形式的沟通来确保关键节点的功能实现。

所以，我认为，开发模式和项目管理模式不可以拘泥于一种形式，关键还是要看是否真正达到了整体的敏捷和精益。对于文中老师提及的 scrum 管理和极限开发，确实是小团队内部协同作战的比较好的实践。但对于多团队协同作战，就要考虑综合运用各种方法了。

另外，对于文中提及的站会形式，从“道”的角度来说，当然是可以视实际需求来确定是否要开，但往往一种文化的培养，需要有仪式感，需要不断锻炼。所以对于我们来说，我们还是坚持开 Scrum 中要求的四个重要会议。

相关阅读：05 | 敏捷开发打底是想解决什么问题?

纯洁的憎恶：

分治策略是应对庞大复杂系统的惯用思路，但它的难点或精髓在于如何确保形散神聚。

详细计划（甘特图）VS 任务状态（Ticket）。

代码不稳定 & 环境部署麻烦 VS 代码审查 & 自动测试 & 自动部署（GIT、CI、DevOps）。

上传下达 VS 频繁沟通、提醒、分享。

大厂的敏捷开发实践，把枯燥的编码变得跟玩游戏一样。借助有效的流程与工具，能够有效节约团队成员的精力，聚焦于任务或角色，不会因频繁“统一思想”导致“技术动作变形”。

而另一面，在大厂里每个人通常都是螺丝钉，长此以往也许会养成不谋全局的习惯。如果能从自己的角色中跳出来，俯瞰整个组织协作的全过程，并站在这个视角上思考问题，一定会有更喜人的收获。

相关阅读：06 | 大厂都在用哪些敏捷方法？（上）

alva_xu：

在一个以 Scrum 为方法的敏捷团队里，首先，Scrum master 是呵护 develop team 的保护神，他的其中一个职责是保护每一次迭代的工作量是 dev team 能按时完成的，而且保护 dev team 能专注于现有 sprint back log 的实现，不会被其他干系人的新需求所打断。

其次，Dev team 是一个 T 型团队，技术比较全面，许多事情多能自助搞定，比如，开发人员同时又有测试技能，同时如果结合结对开发，测试驱动开发，那么，交付物的质量就更有保障。

再者，在一个敏捷团队里，人数比较少，dev team 的沟通能力都比较强，沟通可以比较充分，所以解决问题的能力就比较强，工作效率比较高

最后，敏捷模式的开展，也依赖于工具的使用，目前常用的 CICD 工具，与 jira/confluence 需求沟通管理工具的打通，部署次数的提高，无疑大大提高了开发发布效率，同时也提高了发布质量。

综上所述，只要在人员组织架构、工具产品、流程这三个方面都达到了敏捷的要求，那么发布质量就有了保证。

相关阅读：06 | 大厂都在用哪些敏捷方法？（上）

Felix：

Git 方面也要求团队 Master 中的代码必须通过 Merge Request(Pull Request) 来，也作为 Code Review 的最后一道关卡。持续集成方面大部分通过 Jenkins、几个微服务是通过 Gitlab CI，我们的终极目标是基于镜像部署发布，屏蔽环境影响。

相关阅读：06 | 大厂都在用哪些敏捷方法？（上）

alva_xu：

我觉得在计划会上，有几个事情必须要做好.

第一是需要定义 DOR 和 DOD，Define of Ready 和 Define of Done, 如果没有这两个定义，那么扑克牌可能会玩不起来。

第二 需求（用户故事分解成的 task) 一定要尽量明确。不管扑克估算还是其他估算方式，如果第一轮估算偏差过大，说明大家对需求不明确，需要产品经理进行更详细的说明。通过几轮估算，如果大家能达成比较一致的估算，那么工作量的估算就比较靠谱了，这也是 Scrum 这种工作方法带来的好处，能让需求得到合理的资源安排。

不管怎么说，在 Scrum 里，要重视估算，有了好的估算，速率才真正有意义，才能真正保证交付质量。

相关阅读：07 | 大厂都在用哪些敏捷方法？（下）

纯洁的憎恶：

“多、快、好、省”，软件工程的四难选择问题。由于质量是软件工程压倒一切的要素，因此“好”必须留在“盘子”里。剩下的要素都是可以根据具体情况权衡取舍的。四难选择变成了三难选择。于是，工程师在实践中面对不确定时，也能够有底气做到“不抵触，讲条件”了。

延长时间的另一面是提高效率。借助工具、优化流程、节约资源等方法，可以在一定程度上“冲销掉”延长的时间。

非常欣赏 MVP 模型，既可以快速见效，又降低了大量返工的可能。在瀑布模型中，通过会有过度设计的现象。一开始想了很多，结果发现恨不能 80% 都是瞎想。先拿出核心功能，再根据用户使用的情况，有指向性的完善，步步迭代演进，十分靠谱。

唯一令人担忧的是，在外包模式中，如果没有明确的需求，就难以估算出较为准确、合理的预算，进而无法立项、采购。如果先做一版需求申请下来预算再说，再用 MVP 模型步步试探。那么最后做出来的东西可能与需求文档严重不一致，存在较大的审计、内控风险。也许企业大了、规矩多了，做起事来确实别扭。

作者回复：迭代模型和 MVP 是非常好的组合，因为迭代的时候，会优先选取最重要的功能，慢慢的那些不重要的功能甚至永远不会被加入迭代中，就因为不需要浪费时间在上面了

相关阅读：08 | 怎样平衡软件质量与时间成本范围的关系？

alva_xu：

传统的大企业（不是指 BAT 这类大企业），比如我们企业，IT 项目牵涉到三个部门，一个是业务需求部门，一个是 IT 部门，一个是财务预算审批部门，采取的形式一般都是采用外包方式，而且往往是固定合同，也就是合同价格是确定的，需求范围也是确定的，这样的话，金三角的两条边就定下来了，剩下来的就是时间和质量的关系问题了。

按照金三角的理论，我们就可以知道前面所述的场景下项目组该重点抓什么了：作为甲方项目经理，重点抓的就是质量和时间了。如何通过提高效率，使单位时间的产出比原来的多（相当于增加了时间），来提高项目的交付质量，是我们甲方 IT 项目经理最关心的事。

所以这时候，我们的方法是建立统一软件框架、提供公共服务组件、制定代码和测试规范、培训乙方团队、搭建 CICD 平台和自动化测试平台、sonarqube 自动代码检测平台等，使原来几周一次测试变成一周几次测试，使原来低质量的代码快速变成高质量的代码...

反正是采用各种方法，提高工作效率，用于抵消业务部门不时提出的变更导致的项目进度的风险。当然在开发模式上，也会衡量敏捷的开发模式（特别是 scrum 的管理模式）和传统瀑布及衍生模式哪种模式更高效。

当然，理解了金三角，对于前期申请项目预算也是有帮助的，比如，可以和预算部门谈判，如果要砍预算，在时间一定的情况下，就只能减少项目范围，这是我们业务需求部门所不能接受的。这样，就可以使 IT 项目经理名正言顺地把预算部门和 IT 部门的矛盾转嫁到预算部门和业务需求部门去。

当然，最合理的做法应该是向 BAT 公司看齐，IT 部门转变为利润中心，自己管预算、自己有开发团队，那么金三角的三条边就都可以进行调优了。

相关阅读：08 | 怎样平衡软件质量与时间成本范围的关系？

最佳思辨

林云：

文中提出可以借鉴软件开发模型中的特点，这一点并不是普通软件开发成员可以使用的。任何一个软件开发模式都有对应的主要问题。就像你把飞机的引擎放在拖拉机上一样。需要对模型进行总体考虑。而且不同的软件开发模式都有对交付团队有能力的要求。

举个不恰当的例子，组合软件开发模式的特点就像让一个摩托车驾驶员开着安装了飞机引擎的拖拉机。这并不是软件工程想达到的结果。希望作者对组合研发模式的前提和应用过程进行描述，以减少软件工程方法使用的随意性。

宝玉：

谢谢指正，结合最近波音 747Max 的案例，确实不能乱用，不能说飞机的软件也用敏捷这种快速上线快速迭代的模式。我觉得组合研发模式的前提还是质量，软件工程的目标就是要构建和维护高质量的软件，无论怎么组合开发模式，都不能牺牲质量。

这里我也列一些我觉得好的实践：

1. 瀑布模型可以参考敏捷开发，引入持续集成、自动化测试，提升效率；
2. 敏捷开发可以参考瀑布模型，开发前多设计，开发后多测试，尤其是要辅助人工测试。

相关阅读：04 | 瀑布模型之外，还有哪些开发模型？

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。

精选留言 (11)

大茹
2019-03-16

不得不说，留言区的内容真是精彩，只是这些内容，这个专栏都买的值了！

> 作者回复: 是呀，单纯文章覆盖的范围有限，留言则有各种真实的案例、自己的学习感悟，都是非常有价值的👍

13

Felix
2019-03-16

向各位大神学习了，看到这么长的留言确实让人惊讶，日后一定仔细研读文后留言，结合文章定是另有一番收货
也给最后的最佳思辨的林云点个赞，这种讨论更能引发大家思考，不知道是不是我的前同事林云，哈哈

> 作者回复: 是呀，很多有价值的留言的👍

6

Amy

2020-09-03

以前通过看书学习软件工程，看完之后总感觉很朦胧，好像学习到了点东西，又好像没什么收获。通过宝玉老师的课程，既有理论基础又有实践分享，还有各位大神的提问，很多都是我在工作中也同样遇到的问题，收获特别大。学习软件工程，学习工程思维，把每一项工作都当成一个项目来实现！

3

KK_TTN

2019-03-24

一个小小的提议 以后可不可以更频繁地发Q&A以及留言精选(比如每周) 感觉这篇很精彩但是内容稍微有点多

编辑回复: 感谢你的建议，以后会酌情来安排内容排期哒。

3

Raymond吕

2020-03-12

这个专栏是我听过的专栏里留言互动最多，老师回复最精彩的。

1

小高

2019-08-09

值了，特别是纯洁的憎恶大佬的评论，真的打开了我的眼界！这真的是一场头脑风暴，值了！

作者回复: 专栏相比书籍有个优势就是可以和作者互动，可以看到其他读者的精彩留言👍

1

大王叫我来巡山

2019-07-29

这篇文章就值专栏的价格了，从几十万到上千万的项目都经历过并且主导过，干黄了不少，干成的也有，上述的场景基本都经历了，真正的设计考验的是应对变化的能力，守住稳定点，认真评估变化点，搞清楚主线任务和支线任务的关系，不被别人左右，守住底线，才能运筹帷幄，决胜千里。（然而我还做不到）

作者回复: 你这个总结的非常好👍

“真正的设计考验的是应对变化的能力，守住稳定点，认真评估变化点，搞清楚主线任务和支线任务的关系，不被别人左右，守住底线，才能运筹帷幄，决胜千里。”

1

一步

2019-03-16

哈哈，每次更新都是马上看的，下面都没留言😅，后面还要刷第二遍的

作者回复: 没关系，我们也会定期整理的：）

1

ifelse

2022-06-21

太长，看了2/3没往下看了

aoe

2022-01-12

这是我读的最长的一篇加餐内容！

XiangJiawei

2020-03-02

我就是属于那种买了课一直拖延不看的人，才看到第一期一问一答，感觉收获非常大。尤其是纯洁的憎恶的观点，很有启发，感谢~

作者回复: 👍有收获就好！