

划重点讲一次关于“沟通反馈”主题内容的复盘

# 划重点

一次关于“沟通反馈”主题内容的复盘



你好，我是郑晔，恭喜你，又完成了一个模块的学习。

在“沟通反馈”这个模块中，我与你探讨了与人打交道的一些方法，只不过，这并非传统意义上的谈话技巧。而是希望你能克服自己的心理障碍，主动与真实世界进行沟通，获取反馈，让自己对信息的编解码能力不断得到提升。

## 重点复习

在这个模块中，我们学习到了一些最佳实践。

### • 看板

- 一种来自精益生产的可视化实践。
- 按阶段将任务放置其中。
- 可以帮助我们发现问题。

### • 持续集成

- 做好持续集成的关键是，快速反馈。
- 本地检查通过之后再提交。
- 找到有效的反馈方式，比如：CI 监视器。
- 持续集成的纪律。
  - 只有 CI 服务器处于绿色的状态才能提交代码。
  - CI 服务器一旦检查出错，要立即修复。

### • 回顾会议

- 软件团队复盘的一种实践。
- 枚举关注点，选出重点，深入讨论，列出行动项，找到负责人。

- **5个为什么**

- 又一个来自丰田的实践。
- 沿着一条主线追问多个问题。

在这个模块中，我们还了解一些重要的思路，让我们把工作做得更好。

- **用信息论理解沟通反馈**

- **写代码的进阶路径**

- 编写可以运行的代码。
- 编写符合代码规范的代码。
- 编写人可以理解的代码。
- 用业务语言写代码。

- **会议是一种重量级的沟通方式**

- 减少参会人数。
- 找人面对面沟通。

- **聆听用户声音**

- 能做自己用户，做自己的用户。
- 能接近用户，接近用户。
- 没有用户，创造用户。

- **Fail Fast**

- 一种编写代码的原则。
- 出现问题尽早报错。

- **金字塔原理**

- 从中心论点，到分论点，再到论据。

## 实战指南

在“沟通反馈”的模块，我也将每篇内容浓缩为一句实战指南，现在一起回顾一下。

- **通过沟通反馈，不断升级自己的编解码能力。**

——《[20 | 为什么世界和你的理解不一样](#)》

- **用业务的语言写代码。**

——《[21 | 你的代码为谁而写？](#)》

- **多面对面沟通，少开会。**

——《[22 | 轻量级沟通：你总是在开会吗？](#)》

- **多尝试用可视化的方式进行沟通。**

——《[23 | 可视化：一种更为直观的沟通方式](#)》

- **做好持续集成的关键在于，快速反馈。**

——《[24 | 快速反馈：为什么你们公司总是做不好持续集成？](#)》

- 定期复盘，找准问题根因，不断改善。

——《[25 | 开发中的问题一再出现，应该怎么办？](#)》

- 多走近用户。

——《[26 | 作为程序员，你也应该聆听用户声音](#)》

- 事情往前做，有问题尽早暴露。

——《[27 | 尽早暴露问题：为什么被指责的总是你？](#)》

- 多输出，让知识更有结构。

——《[28 | 结构化：写文档也是一种学习方式](#)》

## 额外收获

在这个模块的最后，针对大家在学习过程中的一些问题，我也进行了回答，帮你梳理出一个思路，更好地理解学到的内容：

- 持续集成是一条主线，可以将诸多实践贯穿起来。

- 从持续集成到稳定的开发分支，到频繁提交，足够小的任务，到任务分解。

- 从持续集成到可检查，到测试防护网，到测试覆盖率，到单元测试，到可测试代码，到软件设计。

- 安全性检查，是回顾会议的前提条件。

- 在信息获取上，国内外程序员差别不大，开拓视野，改善工作习惯，是国内程序员亟需提高的。

——《[答疑解惑 | 持续集成，一条贯穿诸多实践的主线](#)》

## 留言精选

在讲到定期复盘，找准问题根因时，[西西弗与卡夫卡](#) 同学提到：

关于复盘，孙陶然曾经说过，如果他有所成就，一半要归功于复盘。他提出了几个步骤供大家参考。首先，先对比实际结果和起初所定目标之间有什么差距。其次，情景再现，回顾项目的几个阶段。然后，对每个阶段进行得失分析，找出问题原因。最后，总结规律，化作自己的技能沉淀，再次遇到时可以规避。

我再补充一点，复盘资料应该记录到知识库，无论新来的或是接手的人，都能从中获益，从而提升组织的能力。另外，好的复盘需要有坦诚的文化氛围，不然有可能变成互相指责甩锅，就失去了意义。

另外，[西西弗与卡夫卡](#) 同学还分享了提升开会效率的方法：

其他一些提升开会效率的方法，比如会前每个人要先做准备，把观点写下来，然后发给主持人。再比如六顶思考帽，大家按相近的思考角度讨论，而不是我说一趴，你说另一趴。还有，主持人控制这轮谁能发言，控制每个人的时长。方法很多，但实际上总有人破坏规则，特别是当这个人是老板...

在用信息论来讨论沟通反馈问题时，[毅](#) 同学将知识点融会贯通，提出了自己的心得：

不同角色间的沟通：克服上下文差异，分段解码，理解偏差早发现早反馈。相同角色间的沟通，信号相同，解码能力因人而异，要有一个主导的人，控制沟通广度与深度，抓主线适可而止，此时结合任务分解，反向沙盘推演。

关于如何做好复盘，[like\\_jun](#) 同学提到：

要让团队认识到复盘的重要性。

让每个人都深入思考项目运作过程中遇到了哪些问题。才能做好复盘。

在讲到通过金字塔原理进行知识输出时，Y024 同学丰富了金字塔原理的基本原则，具体如下：

金字塔原理的四个基本原则：“结论先行”（一次表达只支持一个思想，且出现在开头）、“以上统下”（任一层次上的思想都必须是其下一层思想的总结概括）、“归类分组”（每组中的思想都必须属于同一范畴）和“逻辑递进”（每组中的思想都必须按照逻辑顺序排列）。

前面两个特点是纵向结构之间的特点，后面两个特点则是横向结构之间的特点。以上内容收集整理自李忠秋老师的《结构思考力》，感兴趣的小伙伴可以看看。

另外，对于会议，Y024 同学也提出了他团队正在进行的摸索和尝试：

1.沟通的指导原则之一就是在同步沟通的时候（比如开会），人越少越好。而在异步沟通的时候（比如E-mail），涉及的听众越多越好。

2.关于开会分享下我们正在摸索的。

（a）每个会开始前，会议发起人在石墨文档上以“会议记录”模版（我们持续形成自己的模版）新建一个纪要：说明议程、及讨论内容等前提内容并提前告知与会人员。会议过程中在同一个石墨文档上做纪要，保证纪要可以收集全所有的笔记和行动计划。如果是关联会议，则使用上次相关的石墨文档进行追加内容（保持事件连贯性、完整性）。

（b）半小时的会议设置为 25 分钟，一小时的会议设置成 50 分钟，留有冗余量应付需要换地方等临时情况，保证所有的会议不会有成员迟到的现象。

对于领域驱动设计，小浩子 同学提到了要特别关注可变项和不变项的分离：

领域驱动设计确实是写出合适的代码结构的一项训练，程序员会不由自主地按照自己的习惯，也就是按照计算机运行逻辑去设计代码，这样的代码很容易陷入难以维护的坑。在开始动手写代码之前跟用户交流清楚，理解设计的概念、流程、使用场景、特殊情况，这些都很重要。另外我特别关注的一点是可变项和不变项的分离，因为我们的业务场景对可扩展性要求很高。

经验越丰富的程序员，越能体会到“走进客户”的重要性，关于这一点，David Mao 同学提到：

我做了好多年的软件测试，前几年和销售一起去谈客户，才深深地体会到客户声音的重要性。客户关注的才是真需求，产品经理和开发想出来的很多是伪需求，很多不是客户想要的功能。

感谢同学们的精彩留言。在下一个模块中，我将为大家分享“自动化”这个原则的具体应用。

# 自动化主题预告

“懒惰”应该是所有程序员的骄傲

——想懒惰先勤快

一个好的项目应该是什么样？

——构建脚本：让日常开发变得更简单

程序员怎么学习运维知识？

——一个思考DevOps的框架

有了持续集成就够了吗？

——持续交付：一种延伸的“持续集成”

如何做好验收测试？

——站在用户的角度看测试

你们的代码是怎么变混乱的？

——单一职责：划分界限

总是在说MVC分层架构，但你真的理解分层吗？

——分层思维，是计算机的核心理念

为什么总有人觉得5万块钱可以做一个淘宝

——不同量级的东西不是一回事

先做好DDD再谈微服务吧，那只是一种部署形式

——领域驱动设计：限界上下文

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

# 10x 程序员工作法

掌握主动权，忙到点子上

郑晔

火币网首席架构师  
前 ThoughtWorks 首席咨询师  
TGO 鲲鹏会会员



新版升级：点击「👤请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

## 精选留言



we

这些好方法，都只能熟能生巧。多用后，感觉它的价值了.自然做事就是顺水推舟了。

2019-03-14 08:16



桃子-夏勇杰

郑老师，我们的专栏讲了很多工作方法，以终为始、分解任务和沟通反馈，这些都很重要，而且需要时间磨炼。在这些都做的还不是很好的情况下，这些工作方法和程序员的基础技术能力应该各花多少精力去积累，按照您的直觉您可以和我分享一个大概么？

2019-03-13 11:55