如何爱上结对编程

第一次听说结对编程的时候，我觉得太反直觉了，两个人用一台电脑写代码，效率不就下降了一半吗？后来我在团队里去尝试引入结对编程，也没感觉有多好，而且小伙伴们的反馈也觉得不好，还是一个人写代码更自在，也就做了一周就放弃了。

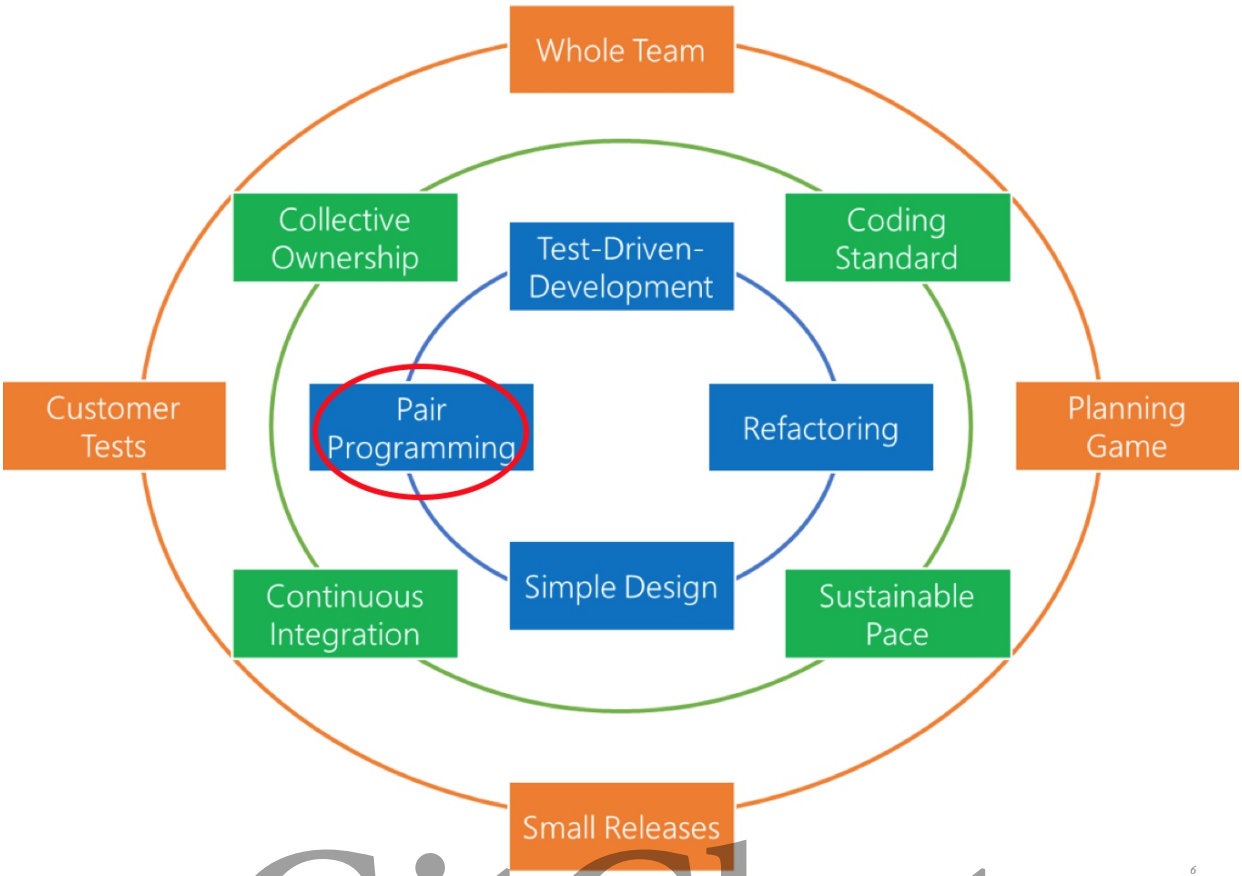
直到 2012 年，第一次参加 Code Retreat 活动的时候，一天之内和日本人，德国人，英国人，美国人结对编程，不停地发出类似「哇哦，这道题目还能有这样的思路啊；哇哦，代码还能这么写啊；哇哦，原来 Vim 是这么用的啊」之类的感叹，让我对结对编程有了一点新的认识。

后来加入 ThoughtWorks 以后，结对编程变成了日常的工作方式，入职第一天，我就看到对面有两位一看发际线就知道很有经验的人在结对编程，充满了欢声笑语，一起去领取故事卡，一起去上厕所，一起去挪动物理墙上的故事卡。让我看得好生羡慕。后来在工作中，通过结对编程，我终于学会了如何做测试驱动开发，如何用 Vim，如何写 Rails 等等等等。到后来，我通过结对编程让别人学会测试驱动开发，学会 Vim，学会 Rails。

这些年，和形形色色的人进行过结对编程，有内向不说话的，也有外向霸着键盘的，有几十年编程经验的老手，甚至也有从未写过一行代码的新手。遇到过很多问题，也慢慢摸索出了一些门道，彻彻底底地爱上了结对编程。
我想分享一下我对结对编程的理解，消除对结对编程的一些普遍误解，影响更多的程序员朋友爱上结对编程，享受它带来的好处。我将从「是什么」，「为什么」和「怎么做」三个方面来展开。

是什么

结对编程是「极限编程（eXtreme Programming）」里的一个实践。



> 结对编程技术是指两位程序员坐在同一工作台前开发软件。

对于企业老板而言，结对编程太过反直觉。两个人用一台电脑，不是浪费了一个人吗？效率不是降低了一半？

为什么

接下来就说说为什么要结对编程，我认为它主要有五个方面的好处。

更专注

对于大多数 996（早九晚九，每周六天）程序员而言，一天 12 小时的工作时间中，有多少时间在看新闻，看技术博客，查邮件，刷朋友圈？有没有统计过一天真正高效编程的时间？我敢打赌， 90% 的人每天真正高效编程的时间，不超过 3 小时。

好的结对编程，注意，我说的是好的，可以让两个人更专注，会让人觉得时间过得特别快，而到了 8 个小时后，一定会觉得没有精力再继续工作了。这就是为什么极限编程里还有一个实践强调每周只工作 40 小时，也就是每天 8 小时，每周 5 天。

提高解决问题的效率

编程中，也符合「二八原则」，80 % 的时间花费在 20% 的工作上。一帆风顺时，不会消耗太多的精力，但一旦被一个问题卡住，就要花大量的时间和精力去解决。随着对精力的消耗加剧，自控力急剧下降，于是掏出手机刷起朋友圈。等反应过来的时候，又到了下班时间了，工作没干完，只能无奈继续加班了，加班时萎靡的精神状态和不甘愿的心态，导致效率和质量都不会太高。

结对编程可以集合两个人的聪明才智，让编程变得更顺畅，减少被问题卡住而浪费的时间。

减少错误

> 错误发现的越早，修复成本越低。

我们在解决程序错误的过程中，80% 的时间是用在定位问题上，一旦定位到问题，多数时候可能只需要修改一行代码就能解决。
举个例子，用户报了一个线上错误，我们需要去定位是哪个子系统的问题，是前端还是后端的问题，是表现层还是逻辑层或者数据层的问题。
如果能在编写代码过程中就及时发现错误，就能节省大量的定位问题的时间。

极限编程的信条是：如果一个实践有价值，我们就把它做到极致。
思考一下，结对编程是将哪个实践做到了极致？对了，就是 Code Review（代码评审）。

知识传递

在软件项目中，有很多的知识，包括业务知识和技术知识。很多团队习惯用文档来沟通，来了新人，先扔一堆文档自己看一个星期，这样的带人方式通常上手很慢，新人也因为没有产出而感到焦虑。文档也可能疏于维护，导致与现实不符，看了也白看。
用结对编程的方式来传递知识，可以让最新的知识在团队中流动起来，所有人都掌握整个团队的业务和技术知识。用结对编程带新人的时候尤其方便，因为大部分的需求都是增删改查，就算不理解更深的原理也可以照猫画虎地解决问题，新人第一天就有产出。一个需求做下来，就基本摸清了项目里的业务，代码结构，框架使用，部署等方面的知识。
《解析极限编程》一书里对结对编程的描述很极致，它不是说一对 Pair 做一个需求，而是定时轮换。早上 A 和 B 结对认领了需求一，C 和 D 认领了需求二，下午 B 和 C 交换了一下， A 和 C 继续做需求一，B 和 D 继续做需求二；第二天早上，A 和 D 又交换了一下，C 和 D 继续做需求一，A 和 B 继续做需求二。用这种方式，再配合 Code Review，就可以让每个人都了解大部分的需求，尽可能消除信息瓶颈。

增进友谊

最后来说说增进友谊，也就是标题所说的：结对编程是程序员最好的社交。如果你的团队实践过结对编程，可能打死都不信。我的很多朋友告诉我尝试结对编程后，大家觉得很不习惯，还是一个人写代码更好，甚至增加了内部矛盾。
这些现象是正常的，我也曾经历过这个阶段。因为结对编程是一门技术，就好比一把倚天剑，如果掌握的不好，不仅无法增加威力，甚至会伤害到自己。所以要重点关注下后面如何做的部分。
现在，我最好的程序员朋友，都是长时间结对工作过的同事。每次久别重逢，最重要的事就是打开电脑来结对，看看对方又研究了什么好玩的。

怎么做

并不是说两个人坐到一起写代码就算结对编程，结对编程是一门技术，需要学习结对的形式和习惯，以及常见问题如何预防和解决。

三种形式

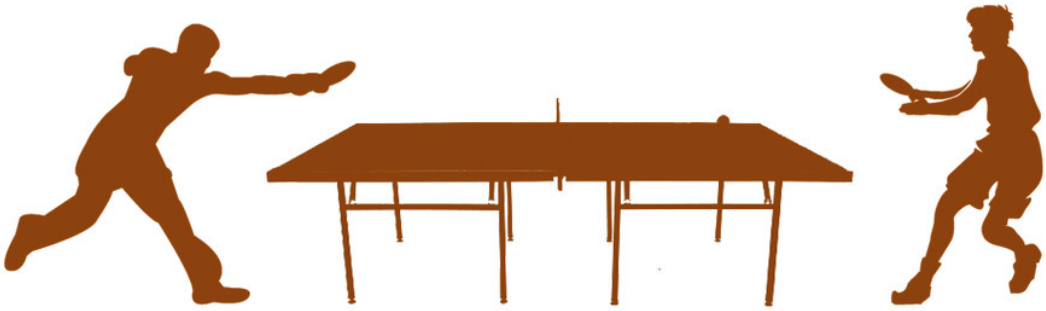
结对编程有三种形式：

- 键盘鼠标式
- Ping-Pong 式
- 领航员-驾驶员式

键盘鼠标式

顾名思义，就是一个人操作键盘，一个人操作鼠标。当然，这种方式越来越不常用，因为程序员们以使用命令行和快捷键为荣，能用到鼠标的地方越来越少了。

Ping-Pong 式



这种是采用 TDD（测试驱动开发）时常用的方式，A 写测试，B 实现和重构，然后 B 写下一个测试，A 来实现和重构。

Navigator-Driver式（领航员-驾驶员式）

最后一种是 Navigator-Driver 式，Navigator 的注意力放在如何实现宏观目标，以及 Review Driver 编写的代码。 Driver 编辑代码，关注的是短期目标，代码细节。需要强调的是，Navigator 之所以叫这个名字，说明他不只是在一旁观看，他因为不操作键盘，想的会比较快，他要引领 Driver 的思路。同时，他在 Review 代码的时候，不要立即指出 Driver 编码中的小错误，那样容易打断 Driver 的思路，因为有可能 Driver 已经发现了，但想先把整个写完再去修复，以保持思路连贯，Navigator 可以在确认对方没有发现的情况下提示对方。

我们最常用的是 Ping-Pong 和 Navigator-Driver 结合的方式，双方都熟悉结对编程并有默契的时候，是会随时切换角色的。原则呢就是要让双方都保持参与感，不然一个人跟不上另一个人的思路，就会开始玩手机，打瞌睡，就无法愉快地结对了。

好习惯

除了形式，我认为还有一些习惯决定了结对编程是否愉快。

身体清洁

试想一下，坐到同事身边，隐隐传来浓烈的脚气味道，一张口传来大蒜或咖喱的味道，还能专心地写代码吗？
所以我们要注意个人卫生，常备口香糖，结对前来一粒，他好我也好。

心态开放

尤其是在老手和新手结对的时候。不要觉得我的方案就是最好的。放低自己，倾听对方的想法，开放地探讨。

欣赏和赞美

当对方提出精彩的想法或写出漂亮的代码时，不要吝啬你的赞美之词。尤其要避免指责和嘲笑，特别是老手和新手结对时，你的目的不是为了证明比对方聪明，而是帮助对方成长。

感恩

结对结束后，真诚地感谢对方，告诉对方从他身上学到了哪些东西，对方会有满满的成就感。也会更愿意再次和你结对。

节奏一致

你观察一下结对比较默契的同事，他们会一起喝水，一起上厕所，一起抽烟。只有保持一致的节奏，才能最大化地提高效率。不然，一个去厕所了，另一个人在那写，回来可能就跟不上了。刚跟上，之前那位又要出去抽根烟…
你可能会问，两个人老是形影不离会不会很尴尬啊。在这里分享一个我的小技巧，我会看对方来决定自己的行为，比如对方去接水，我就去厕所，等我回来接水，对方也去厕所了，就比较不会尴尬。
另一个重要的技巧是，要合理地划分任务，每一个任务在半小时左右可以完成。任务完成，提交代码，休息几分钟，继续下一个任务。就算一个人工作时，这也是比较好的工作节奏。

切换角色

最后一个习惯，不要太过沉迷到自己的世界里，写代码写嗨了就独自霸着键盘，要有意识地察觉对方的状态，尤其是外向型和内向型结对、老手和新手结对时，要注意把控切换时机，给对方机会。

最后，推荐一本书《结对编程技术》，这本书里详细讲了结对编程的各种场景容易出现的问题以及应当如何解决。
比如：

- 新手和新手结对
- 老手和老手结对
- 新手和老手结对
- 同性结对
- 异性结对
- 内向型和内向型结对
- 外向型和外向型结对
- 内向型和外向型结对
- 和外国人结对
- …

以上，是我对结对编程的理解，是什么、为什么以及怎么做，希望对你有用。
结对编程，你试过了吗？感受怎么样呢？期待与你交流~