# 结对编程

## 好句子

* 这种要求是对一个人的心智、道德修养的更高要求。结对编程中，编码不再是私人的工作，而是一种公开的“表演”。程序员的代码、工作方式、技术水平都变得公开和透明，这也许是有些同学不喜欢这一方式的原因。摘自《构建之法——现代软件工程》
* 有人要问，既然代码复审能发现这么多问题，有这么好的效果，如果我们每时每刻都处在代码复审的状态，那不是很好么？事实上，极限编程（Extreme Programming - XP）正是这一思想的体现——为什么不把一些卓有成效的开发方法用到极致（Extreme）? 结对编程就是一个例子。
* 在结对编程模式下，一对程序员肩并肩地、平等地、互补地进行开发工作。两个程序员并排坐在一台电脑前，面对同一个显示器，使用同一个键盘，同一个鼠标一起工作。他们一起分析，一起设计，一起写测试用例，一起编码，一起单元测试，一起集成测试，一起写文档等。
* 每人在各自独立设计、实现软件的过程中不免要犯这样那样的错误。在结对编程中，因为有随时的复审和交流，程序各方面的质量取决于一对程序员中各方面水平较高的那一位。这样，程序中的错误就会少得多，程序的初始质量会高很多，这样会省下很多以后修改、测试的时间。具体地说，结对编程有如下的好处：

（1）在开发层次，结对编程能提供更好的设计质量和代码质量，两人合作能有更强的解决问题的能力。

（2）对开发人员自身来说，结对工作能带来更多的信心，高质量的产出能带来更高的满足感。

（3）在心理上, 当有另一个人在你身边和你紧密配合, 做同样一件事情的时候, 你不好意思开小差, 也不好意思糊弄。

（4）在企业管理层次上，结对能更有效地交流，相互学习和传递经验，能更好地处理人员流动。因为一个人的知识已经被其他人共享。

总之，如果运用得当，结对编程能得到更高的投入产出比（Return of Investment）。

* 在结对编程中，任何一段代码都至少被两双眼睛看过，两个脑袋思考过。代码被不断地复审，这样可以避免牛仔式的编程。同时，结对编程避免了“我的代码”还是“他的代码”的问题，使得代码的责任不属于某个人，而是属于两个人，进而属于整个团队，这样能够帮助建立集体拥有代码的意识，在一定程度上避免了个人英雄主义。

结对编程的过程也是一个互相督促的过程，每个人的一举一动都在别人的视线之内，所有的想法都要受到对方的评价。由于这种督促的压力，使得程序员更认真地工作。结对编程“迫使”程序员必须频繁地交流，而且要提高自己的技术能力以免被别人小看。

结对编程技术是一个非常简单和直观的概念，能达到事半功倍的工作效果，是极限编程的一个重要原则[8]。

## 结对编程技术

在传统开发过程中，每个开发人员负责系统的一部分开发任务，各自分工、互不干扰，而在结对编程的情景下，是两个开发人员结为一对，来共同完成同一个开发任务。但是，人与人之间的合作不是一件简单的事情——尤其当人们都早已习惯了独自工作的时候。

实施结对编程技术将给软件项目的开发工作带来好处，只是这些好处必须经过缜密的思考和计划才能真正体现出来。而另一方面，两个有经验的人可能会发现配对编程里没有什么技能的转移，但是让他们在不同的抽象层次解决同一个问题会让他们更快地找到解决方案，而且错误更少。因为两个程序员具有相同的缺点和盲点的可能性很小，所以当我们采用结对编程的时候会获得一个强大的解决方案，而这个解决方案恰恰是其它软件工程方法学中所没有的。

在实际实施过程中，两人在同一台计算机面前进行编程活动，一个同伴使用鼠标和键盘来编码，另一个同伴观察代码并考虑设计问题。一方发现问题时，暂停编码工作，双方讨论解决。结对者的角色根据需要来不断调整和交换，包括与别的结对组交换成员。这种结对方式可以让每个成员对项目有一个整体的认识，并且有利于团队建立起良好的合作和学习氛围。极限编程的其他几项重要原则包括：频繁地小规模发布软件，简单设计，集体拥有代码和持续集成。

结对编程开发的成果保证了一部分代码至少有两个人熟悉，编码的同时也在被复查，既改善了复查的效果也在开发团队内部传播了知识，比传统的Desktop Review更有意义。原始的那种让程序员互相检查程序的方式是一种相当形式化的手段，其实效果不好，因为检查者无法像开发时那样完全融入程序的逻辑之中，很难从中发现业务逻辑错误，多数情况下根本都不会仔细去看。事实上在两个人同时编写程序的过程中，每个人的注意力都会比自己一个人编程时更加集中，考虑问题更加全面，互相提醒需要注意的事项。

当两个人共同开发时，充分的交流所带来的效率是传统纸张文档所不可比拟的，也更容易深入业务逻辑，发掘问题并拓展解决方案的思路，增进团队间的友谊。而且通过大量的交流和沟通，两个人在技术水平和业务能力上的提升都远比一个人闷着头干活儿快得多。不要吝音于和自己的partner交流，因为就算是经验老道的程序员也能从新手身上学到有价值的东西，开阔思路，对公司整体员工质素的提升也是很好的促进[4]。

结对编程保证了任何一行程序都有两个以上的人了解，也就是说对于项目每一个模块都有buckup人选，这对公司抵抗人员流动风险的能力有极大的帮助。由于参与结对的成员是经常在变化的(通常半天可以重新组队一次)，所以对于同一块程序，所有相关人员全部离职的可能性相当低。对于一个需要修改的程序，只要有程序员是熟悉它的，无论过了多久，也比一个“新手”做起来快，一个对此程序完全陌生的程序员相较于backup人选来说必然需要更长的时间去挖清楚其中的逻辑和实现风格，一定程度上增加了公司维护程序、升级程序的成本。

## 问题与演化

当然，凡事有利有弊，结对编程也不例外。有人垢病结对开发的效率问题，认为传统的两个人各自码代码的速度肯定比同一时刻只有一个人打字的速度要快得多，自然结对编程的效率就相对比较低下了。但从整体开发角度，开发过程中也许表面上会慢，其实由两个人同时开发所能避免的错误和返工将是非常客观的，对于经验并不老道的程序员来说尤其如此，再算上维护的时间和成本，应当说结对开发的效率是只高不低的[10]。

不过，结对编程并不是什么灵丹妙药，其中有一个相当关键的要素就是程序员们都是抱着期望提升代码质量和自身素质的目的参与到项目中的，如果所有的程序员都是以混日子、磨洋工的风格工作，以完成任务了事为最高使命的话，那无论是么开发方式都不会有任何的帮助，整个开发团队只会步步走向崩溃[11]。

由于现在开发环境的变更发展，出现了演化变形的结对编程形式——远程结对编程和乒乓结对编程。远程结对编程，又称虚拟结对编程或分布式结对编程，是指两个程序员不在同一地点，通过协同编辑器，共享桌面，或远程结对编程的IDE插件进行的结对编程。远程结对编程破除了传统结对编程对地域上的限制，更加适应大环境的发展，但同时还引入了一些在面对面的结对编程中不存在的困难，例如协作的额外时延，更多的依赖“重量级”的任务跟踪工具，而不是“轻量级”的索引卡片，以及没有口头交流导致的在类似谁“控制键盘”问题上的混乱和冲突[12]。乒乓结对编程是指观察者编写失败的测试用例，驾驶者修改代码以通过该用例，观察者编写新的单元测试用例，等等。这个循环持续到观察者不能写出失败的测试用例，但是实际实施过程花费时间会比估计的计划要超出很多。