# 结对编程的研究与扩展

## Study and Expanding for "Pairing Programming" Practice

张 猛 (北京石油化工学院经管学院信息管理系 北京 102617)

**摘 要:** 讨论了极限编程理论重要的"结对编程"实践在国内推广中遇到的问题,根据对实际软件开发过程的观察和调查,从企业管理的角度说明"结对编程"可以节省时间、提高效率和质量。对"结对编程"实践的形式进行了扩展,提出了"轮转结对"、"多人极对"、"极限结对"等新的实践形式,用来自中国的实践对"结对编程"的实践进行了丰富。

**关键词:** 结对编程 极限编程 软件工程 IT 团队管理

## 1 概述

### 1.1 什么是结对编程

"结对编程"是极限编程(XP)的最重要的实践之一。国际上对极限编程的研究已经持续了近十几年(Kent Beck:1996,1999)1,在国际软件巨头的开发实践中也大量采纳了 XP 的一些理论与实践。微软公司在微软软件框架 MSF3.0 版本中,正式加入对敏捷方法的支持2。

### 1.2 极限编程在国内推广的现状

XP 理论虽然在上世纪 90 年代末就引进中国,但在国内并未得到广泛推广和重视,大致原因是:

1)许多软件企业对于软件工程主流的RUP、CMM 框架尚不清楚,对于新型的极限编程(XP)就更无暇顾及。

2)国内学者对极限编程研究不足,对于迭代、极限编程(XP)等轻量级软件工程方法论的认识不足。国内核心期刊在 2004 年才见到介绍极限编程的文献(王辉、余雪丽:2004)3。

3)XP 的许多理念,需要先进的管理理念相支持,与国内软件企业的管理水平不匹配。

### 1.3 结对编程推广的主要困难

结对编程在国内主要遇到有两方面的阻力:

1)经理人员的反对

多数经理人员在听到"结对编程"时的第一反应是:"不能让两个人干一个人的活"。

2)技术人员的反对

向技术人员推广"结对编程"理念时,技术人员通常会表示出对这一"新"理论的兴趣。但对于制度化、长期化保持结对,技术人员表示出对失去个人自由和代码控制的担心。

## 2 本研究的目的和方法

### 2.1 目的

本研究期望对国内软件开发团队应用结对编程的现状和影响因素进行分析,消除应用结对编程的障碍,提出更加灵活更具操作性的结对编程形式。

### 2.2 方法

本研究采用访谈、追踪调查、理论分析和演绎的方法对"结对编程"实践的现状进行分析,并提出新的实践形式。

## 3 对国内软件公司的调查

在软件开发实践中,许多公司和团队经常在不知不觉中采用了"结对编程"实践。

通过对中软、中科软、大唐、神州泰越、航天四创、合力时等国内知名软件开发企业的访谈发现,这些企业的开发团队在无意识中或多或少都采取了结对工作方式,主要有:(1)程序经理修改了程序员的代码后,除了让程序员签出代码,还要向他解释为什么修改。(2)在完成非常规、研究性、紧急地,程序经理、系统分析员、程序员,经常挤在一台计算机前协同工作。这些非正式的结对编程,在项目时间要求紧、团队成员新老搭配的时候经常发生。只是这些结对活动没有被记录下来,没有正式化,而不为人所知。

## 4 “结对编程”实践的观察

在北京航天四创公司开发国家某部委的项目的时候，在没有规定结对编程、没有培训情况下，我们对某一开发团队的行为进行了持续的观察和记录，发现以下自发结对情况：

### 4.1 程序员结对

程序经理和程序员坐在一台机器前面共同编写代码。（如图1所示），这个临时结对一直持续到他们负责的模块开发结束。在结对过程中，程序经理边写代码边向程序员解释，体现了“结对编程”在知识传承、促进交流方面的优势；当程序员偶然独自编程后，程序经理看不懂程序员自行编写的代码，表现出不使用“结对编程”的情况下，程序员各自编写代码时沟通成本。

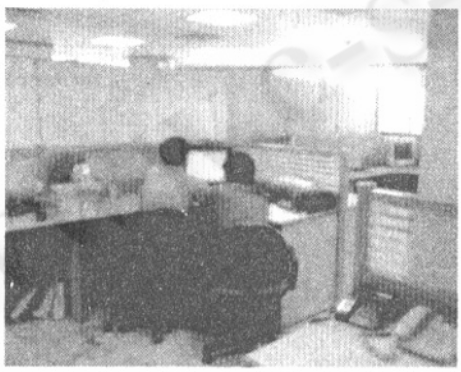

图1 自发结成的程序经理和程序员

### 4.2 系统分析员结对

在进行数据库设计时，两个分析员也自发结对进行设计，在一台计算机前研究（如图2所示），这个临时结对一直持续到数据库设计、实施结束。

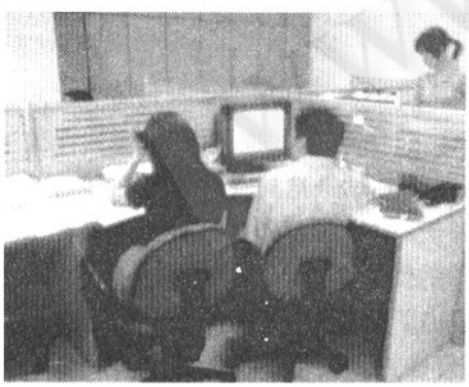

图2 自发结对的系统分析员

### 4.3 现象解读

基于现实观察，可以看出，技术人员在不清楚什么是“结对编程”的情况下，在实际工作中会自然地自发采用“结对编程”实践。

## 5 结对编程的理论解读

### 5.1 结对编程的时间管理理论基础

根据国外的研究，结对编程可以有效地提高生产率。

Brook（1963）4记载，大型软件开发团队的开发效率约为1000行/人/年。根据非正规调查，国内软件开发团队每年的平均有效编码量不足1000行。

根据“二八原则”，一个人每天的有效时间不会超过2小时。软件开发工作的特点决定了更多的时间是在构思，所以不论是否一人一台机器，两人一起工作的有效时间至少不会比两人分开工作时有效时间少。

### 5.2 结对编程有助于提高代码质量

传统的代码走查在完成编码之后，由另一个人审查，这种线性的工作安排在任务紧张的情况下经常被取消，造成代码质量低下。而返工、调试的成本远远大于最初就通过结对编程节约的时间和人力成本。结对编程在代码设计和编写阶段就保证了代码的质量（Martin：2000）5。在组织学生进行数据录入实习工作时，采用结对录入的录入质量远高于不结对的录入质量，从另一个角度提供了证据。

### 5.3 结对编程的知识管理理论基础

软件开发最重要的是团队协作和开发人员交流，如何将团队沟通和知识交流，系统化、制度化，正是“极限编程”（XP）的“结对编程”实践所要解决的问题。

当前软件开发工作中一人一机、彼此隔断封闭的形式，实际阻断了团队协作和沟通。团队似乎在同一办公区内，但最终的代码都是单打独斗的结果。甚至可以说，最终的软件只是一群没有组织的乌合之众产出的大杂烩。

“结对编程”用制度化的方式，在形式上强迫人们共同工作，强迫人们共同产出，强迫人们共享代码和知识，将人们隐性、无意识的结对工作状态制度化、规范化，从而促进沟通与协作。

### 5.4 结对编程符合软件开发学习规律

软件开发有许多必须通过群体实践操作才能掌握

的内容，例如命名规范、编码习惯、重构、迭代等等。这些最佳实践无法通过集中教学、开会传达，然后由程序员单独实践就能化作程序员的正规动作。必须通过不断地传帮带训练，程序员才会养成统一正规的动作。

### 5.5 结对编程解决管理问题

许多企业最头痛的就是上网聊天、看新闻、炒股等做与工作无关的工作。两个人工作可以有效杜绝这些问题。

## 6 “结对编程”形式归纳及扩展

受文献所限，人们往往把结对编程的形式僵化成两个程序员使用一台机器，这种机械性的认识不仅限制了结对编程的推广，而且受到了许多攻击(Matt Stephens,2004)6。如果认识到前面所述的结对编程的目的和优势，那么可以采用更加灵活的结对编程形式：

### 6.1 高级程序员与初级程序员结对

这种形式有利于知识传承，也可以促进高级程序员进步来说：新程序员没有定见，更贴近普通用户的需求，会提出有创意的意见，冲击高级程序员的思维定势。

### 6.2 程序员与程序员结对，分析员与分析员结对

这种形式适合于专注于集体的编程和分析任务。有助于同级别的人之间共享知识、刺激灵感。

### 6.3 程序员与分析员结对

XP 主张代码建模与编程结合，主张设计映射到代码。现代的 CASE 工具，例如 together、Rose，都提供了设计——代码互相映射的功能，甚至可以直接与 IDE 结合将设计与编码工作过程融合。设计人员要了解一些实现细节，以便更好地设计，编程人员要了解些设计，才能提高代码质量和重用性。可以采用程序员与分析员结对的方式，提高设计、编程、集成、测试迭代速度。

以上三种形式基本属于传统的“结对”方式，在经典的 XP 著作中都有涉及(Ken Auer:2002)7，本文对结对的伙伴进行了扩展。下面介绍三种新的对结对形式。

### 6.4 轮转结对

两个人结对久了，容易形成“搭档”，形成小团体固有的行为模式和方言，与其他结对之间又失去了知识共享和沟通，因此定期轮转(以某一模块，某一阶段的完成为单位)，有助于整个团队的沟通和共享。

### 6.5 三人或 N 人结对

结对不仅限于 2 人结对。可以 3 人、4 人结对。这适合解决重要的技术问题或项目时间要求非常紧张的情况。这时，可以由系统分析员、程序员和用户根据需要组合在一起。如图 3 所示。

图 3 三人结对编程

### 6.6 极限结对：一对 n，n 对 n 研讨结对

这种方式是最极端的结对编程方式，来源于我们在专业外语教学实践、企业实践项目培训、以及带领技术团队进行技术攻坚的多方面经验。

在软件开发时，随时可由某人或某个结对发起，对关键的问题、最佳实践展开研讨，参与人员头脑风暴进行研讨，主设计师负责记录和实现，直接在出设计和实现。对于解决重大问题，推广最佳实践，这种方式非常有效。

后面的三种结对形式，在 IT 行业，尤其是国内的软件开发企业中比较少见。但在日本企业和 IT 行业之外的行业(例如医疗行业)中，师徒制、岗位轮转、专题研讨则是常态，因此有充分的管理理论和企业实践基础。

## 7 结束语

软件项目的压力越来越大，对于软件工程方法的需求非常迫切。XP 这一优秀理论在国内迟迟得不到推广，源于对它的误解和了解不足。本研究探讨了“结对编程”的优势，提供了管理科学和知识管理等理论的解释，提出了适合具体操作的更加灵活的结对形式，希望能够促进更多软件开发企业打消疑虑采用这一实践，并吸引更多学者参与工程实践的推广和研究。

(下转第 68 页)

(上接第 64 页)

## 参考文献


1 Kent Beck, Cynthia Andres. Extreme Programming ExplAied: Embrace Change, 2nd Ed, Addison Wesley Professional, 2004. 79 - 83.

2 Geoffrey Lory 等. Microsoft 解决方案框架版本 3.0 概述, http://www. microsoft. com/china/technet/itsolutions/ techguide/msf/msfovrvw. mspx#EMAA

3 王辉,余雪丽. 极限编程. 计算机系统应用,2004,13(1):39 - 41.

4 弗雷德里克. 布鲁克斯(Frederick P. Brooks, Jr.)著,汪颖译. 人月神话清,华大学出版社 2002.

5 Robert C. Martin 著,邓辉译. 敏捷软件开发:原则、模式与实践,清华大学出版社 2003.

6 Matt Stephens, Doug Rosenberg 著,汪丰,赵浩译. 重构极限编程——XP 的实践与反思,清华大学出版社,2005. 105 - 125.

7 Ken Auer, Roy Miller 著 唐东铭译,应用极限编程——积极求胜,人民邮电出版社,2003. 88, 174, 188.