RUP®/XP 指南：成对编程

Robert C. Martin
Object Mentor, Inc.

Rational Software 白皮书

TP 158, 3/01

Rational®
the software development company

目录

概述

配对的简短描述

成对编程是由各对程序员编写关于一个项目的软件的技术。每对人员在一个工作站上一起工作。其中一个成员驱动工作站，而另一个成员仔细查看产生的代码。驱动者从战术上考虑，关注当前编写的代码行。观察者验证语法并从战略上考虑整个程序。他们频繁交换这些角色，并且结果代码的编写速度比单个人员更快且缺陷更少。而且，至少有两个开发人员密切了解此代码。

配对案例

请考虑典型的代码复审会议。需要一个人用八小时开发的模块将由八个人用一小时复审。最后的结果是 16 人时花费在模块上。但是，复审人员不能花费所需的时间熟悉代码，因此他们的复审是相当肤浅的。单个开发人员十分熟悉代码，但可能太熟悉了而不能找到大量缺陷。

将这种做法与成对编程的做法进行对比。如果模块需要两个人用八个小时来开发，则将花费总共 16 人时。但是，在这种情况下，将有*两个*开发人员密切了解代码。对一个开发人员隐藏的缺陷将对另一个人是可见的。

成对编程的案例很简单，但影响是微妙和深远的。成对编程只是编写和复审代码的一种有效得多的方法。有两个人密切熟悉模块，写入代码的缺陷将少得多。代码将具有更好的结构并且将有两倍的人员密切了解代码。如果只是这些益处，那么也已经足够了，但是配对操作还提供了更多的益处。

成对人员更有勇气：单个程序员可能害怕尝试的东西，两个人将有勇气去尝试并有技能去评估。

配对促进了团队协作：因为模块不是由一个人编写，因此代码成为团队（而不是特定开发人员）的财产。

配对促进了知识的传播：彼此配成对的开发人员越多，在整个团队散播的系统知识就越多。结果是团队成员熟悉了系统的所有部分，而不是每个成员只了解一个特定部分。

配对促进了生产力：单独编程的人经历过精力旺盛的时期后，跟着就是相对不活跃的时期。成对人员则可以彼此调整步伐。当一个人感到疲劳时，就交换角色。他们设法使工作强度保持得比单个人通常可容忍的更久一些。

配对很有趣：与另一个开发人员一起工作是受教育的、有刺激性的并且就是有趣。配对增强了工作满意度和总体士气。

做法

配对

当负责任务的开发人员请求其他人的帮助时，配对就开始了。规则是：当询问您时，必须说是。这不意味着您必须立即停下正在做的工作。而是意味着您必须协商可以提供帮助的一个时间和可以获得帮助作为回报的另一个时间。

搭档伙伴不承担任务的职责。该职责仍然由任务所有者承担。搭档伙伴也不承诺与所有者一起呆到任务完成时。搭档伙伴只承诺给予帮助。

搭档中的一个成员变成驱动者，而另一个人旁观。驱动者输入代码、运行编译器、运行单元测试等。观察者检查每次击键、每个命令、每个测试结果并提供帮助和建议。双方一直在忙碌。

有时，驱动者将最了解要做什么，而观察者将只是跟着做。而其它时候，观察者将指示驱动者做什么。有时，驱动者感到受挫时，就将键盘递给观察者，从而交换了角色。而其它时候，观察者将要求键盘并交换角色。在配对会议中这将发生多次。

不断变化的搭档

搭档伙伴不是长期的。典型的配对会议将持续大约半天。任一成员可由于任何原因而选择脱离所在的对。当这种情况发生时，任务所有者必须寻找另一个搭档伙伴。这可能意味着该是任务所有者回报上周与他或她配对的人的关照的时候了。另一方面，或许他或她应该请求具有恰当经验的某个人来帮助解决特别棘手的问题。

这种不断变化的配对导致系统知识散播到整个开发团队中。在很短的时间内，团队的每个成员都将花时间处理系统的几乎每个部分。这大大减少了项目流动的敏感性并使每个程序员在处理整个系统时更加自信。

集体所有权

由于每个人都处理系统中的所有不同模块，因此没有人拥有任何特定的模块。这意味着系统的职责不是以模块为基础划分的。而是整个团队集体负责整个系统。团队中的任何成员可以出于任何原因检出和更改系统中的任何模块。当某一对成员对模块 X 作出更改，而更改导致模块 Y 的单元测试失败时，该对成员修复模块 Y。

调整步伐和协作

成对编程的交流形式非常激烈。口头对话经常是争吵，并且局外旁观者可能很难弄清它的意思。作为观察者，您会听到搭档蹦出一个词，比如：“分号”或“右括号”。或者在程序员赞同或不赞同屏幕上显示的内容时，只听到不清晰的咕哝声。两人是如此密切专注于代码，以致于看来许多沟通都是非口头的。身体语言担负着重要的作用。搭档可以分辨出对方何时对代码不满意（即使什么话也不说）。一个鬼脸、叹气、不安烦躁的情绪 — 所有表现都加大了合作伙伴之间的沟通范围。

有时，一个伙伴将握住鼠标，而另一个人操作键盘。握住鼠标的人控制着模块中将进行工作的位置。键盘操控者控制在该位置更改或添加的内容。在其它时候，一个伙伴输入，而另一个伙伴将预见即将进行的函数调用并恰在编码员需要说明时在适当的页面打开 API 文档。

当一个伙伴感到疲劳时，另一个人可以带头，让他或她的伙伴通过扮演观察者的角色来得到休息。其它时候，两个伙伴将高度集中精力并频繁交换键盘和鼠标。

概括地说，规则很少并且过程更少。唯一真正的约束是双方必须一直处于忙碌中并保持密切的沟通。一个人输入，另一个人望向窗外，这样一对并不是真正的一对。

单独的开发人员可以做什么呢？

您不可能在所有时间都处于配对状态。某些项目 — 即采用 *eXtreme Programming* （XP）（请参阅参考资料 [1]）流程的那些项目 — 遵循“必须由成对人员产生所有生产代码”这一规则。在那种情况下，当未配对时，您可以检查电子邮件，阅读新的技术或 API，通读不熟悉的代码或与涉众谈论当前的迭代或未来的计划。实际上，在未找到配对伙伴的那几个小时中，开发人员总是能找到有收益的事情去做。

某些项目对于配对不太严格。一些项目将允许单个开发人员编写测试。其它项目允许单个开发人员编写抽象类或接口。还有另一些项目恰好允许开发人员决定何时最好实施配对。不过有一件事情是明确的 — 研究表明当实行配对时，缺陷率显著下降。

一些人不喜欢配对

有些人对于配对的概念感到不舒服。根据我们的经验，实际尝试配对一周左右的那些人会发现不适感消失，开始喜欢配对，并发现它很有用。很少有人坚持不喜欢此做法。因此，对于大多数人，只是尝试它并习惯它的问题。对于真正尝试过并发现仍然不喜欢此做法的那些人，团队必须找到适合的事情让他们去做。

设备、设施和后勤

显视器和键盘放置

设备放置对于配对成功是至关重要的。如果伙伴不能坐在彼此的旁边并快速交换键盘，那么这个人将无法很好配对。规则是：必须能够来回互递键盘和鼠标，而不必更换座位。

最佳的安排通常是一张好看的长形平面桌。将显示器放在中间并放置两把面对着显示器的椅子。坐下时显示器在两人中间。请确保能很容易地在你们之间来回滑动键盘和鼠标。也请确保使用键盘时感到很舒适并坐直。确保两个伙伴不必旋转显示器就都能看到它。

大房间

要促进配对伙伴的不断变更，安排在大房间工作通常是明智的。在单个房间中布置几对工作站。使用转椅以及漆布或瓷砖地面。将工作站布置成各对都彼此面对。这里的目的是加强沟通。有时，最重要的沟通是偶然发现的。我们希望增加此类沟通发生的机会。

内角

当前的许多小房间都将工作站放置在内角处。开发人员面对房间角落坐着，有一个显示器放在他或她的面前。虽然这对于单独工作很方便，但在此环境中很好配对几乎是不可能的。如果您的小房间的工作站是在内角的，则在别处设置一些成对工作站 — 或许在会议室里。在会议室桌上使用膝上型计算机配对通常是很有效的。

问题和疑虑

配对使生产力减半

它所持的理由是两个人一起完成一项任务将耗用的人时数是一个人完成同一任务耗用的两倍。看似合情合理，但看来并不是事实。独立研究（请参阅参考资料 [2]）已显示，成对工作损失（如果有损失的话）的生产力极少。那些同样的研究显示成对工作充分降低了缺陷率*和代码规模*，同时大大增强了工作乐趣。

成对伙伴之间的争论

任务所有者在所有设计争论中有最终发言权，但解决争论的最佳方式是尝试两种想法并选择工作得更好的那一个。

专家

传统观念建议专门研究特定领域（如数据库或 GUI）的开发人员应将他们的精力唯一地运用在那些领域。但是在成对编程环境中，这些专家成为不共享专业的其它人的指导者。开发人员可以签约所从事专业之外的任务并从专家那里获得帮助。这样，专家的知识倾向于散播在项目团队中，大大减少了项目流动的敏感性。

噪音

一对成员在编程时会产生噪音。充满各对成员的大房间将保持不断的、低低的嗡嗡声。有些人感觉这些噪音将令人心烦、分散注意力。但这未被证实是一个严重的问题。如果您发现噪音令您心烦意乱，可以到会议室呆一会儿。

莽撞的人

许多团队发现他们“骄傲地”拥有一个或两个莽撞的程序员。这些程序员的工作做得比其他任何人都快，无法与其它人一同工作，并且不允许（即使允许也读不懂）其他任何人读他们的代码。对付这些开发人员的最佳方法是将他们调出项目或换成不编写生产代码的角色。或许他们可以编写短期工具或做一些折磨人的测试。

实际障碍和障碍样式

有些人使用标准键盘。而其他人喜欢 DVORAK。有些人需要特殊键盘、鼠标、显示器、脚踏开关等等。有些人没有耳机和大声的音乐就无法编程。而其他人必须用空的 Twinkie 包围绕自己。有些人喜欢用 emacs。而其他人喜欢用 VI。还有些人想用 WordPad 工作。

实际上，可以说出的障碍是数不清的。但每个障碍都可经过一点思考和折衷的能力来解决。被此类事情阻碍的团队将只是在尝试的任何努力中都不会成功的团队。

团队可以如何计划配对？

我们不认为任务应该分配给各对。而是每个开发人员应当负责一组任务。我们希望各对是随意形成的。每个开发人员继续执行自己的职责，请求其他开发人员给予暂时的帮助。任务的所有者保留所有权和责任。搭档伙伴只是助手。

每个开发人员在提供任务估计量时，必须考虑他或她的配对时间量。

结论

成对编程是代码复审的经过良好测试和很好接受的备选方案。而且，它是从根本上不同的编写软件的方法。好处远远不止提高生产力和质量，还影响诸如团队的活力和士气之类的事情。

参考资料

[1] *eXtreme Programming eXplained*, Kent Beck, Addison Wesley, 2000.

[2] *Strengthening the Case for Pair Programming*, Laurie Williams, University of Utah, July/Aug 2000 IEEE Software.

Rational®
the software development company

Dual Headquarters:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
International Locations: www.rational.com/worldwide