# 第3章 敏捷软件工程方法

【学习重点】

(1) 理解敏捷软件工程概念、过程及原则。

(2) 理解极限编程要点。

(3) 理解结对编程概念及特点。

在传统的软件开发方法中,开发人员努力构建客户想要的产品,他们花费大量的时间从客户那里获取需求,针对需求进行分析和建模,并且归纳成规格说明书,然后评审说明书,与客户开会讨论,最后签字确认。表面上看他们开发的软件是符合客户的要求的,但通常事与愿违。在项目快要结束的时候,需求和范围、软件的适用性常常成为争论的焦点。

敏捷软件工程方法(简称敏捷方法)告诉我们开发软件是一种学习的体验,没有谁能完全理解所有需求之后才开始软件开发,即使是客户也一样。客户一开始有一些想法,但是他们也在软件项目的进展过程中逐步了解他们对软件的需要。同样,开发人员在一开始有已知的知识,但是他们需要通过项目来继续学习。没有人完全清楚会构建出什么样的软件,直到项目结束。每个人都在通过项目学习,敏捷方法改变了过程以便识别持续学习,培养了每个人的学习能力。

## 3.1 敏捷软件工程过程

### 3.1.1 敏捷过程

敏捷是一类过程的统称,它们有一个共性,就是符合敏捷价值观,遵循敏捷的原则。敏捷就是“快”,要快就要发挥个人的个性思维多一些。虽然通过结对编程、代码共有、团队替补等方式可以减少个人对软件的影响力,但仍会造成软件开发继承性地下降,因此敏捷软件工程是一个新的思路,并不是软件开发的终极选择。对于长时间、人数众多的大型软件应用的开发,文档的管理与衔接作用还是不可替代的。如何把敏捷的开发思路与传统的流水线工厂式管理有机地结合,是软件开发组织者面临的新课题。

敏捷软件工程的两大主要特征是对“适应性”的强调与对“人”的关注。经典的软件工程方法借鉴了工程学领域的实践,强调前期的设计与规划,并尝试在很长的时间跨度内为一个软件开发项目制订严格且详尽的计划,然后交由具备普通技能的人员分阶段依次达成目标。而敏捷过程强调对变化的快速响应能力,它通过引入迭代式的开发手段,较好地解决了如何应对变化的问题。迭代并非是一个新概念,以迭代为特征的开发方法由来已久。例如,螺旋模型便是一种具备鲜明的迭代特征的软件开发模式。

敏捷过程将整个软件生存周期分解为若干个小的迭代周期，通过在每个迭代周期结束时交付阶段性成果来获取切实有效的客户反馈。其目的便是希望通过建立及时的反馈机制，以应对随时可能发生的需求变更，并做出相应的调整，从而增强对软件项目的控制能力。所以，敏捷过程对变化的环境具有更好的适应能力，相比于经典软件开发过程的计划特征，敏捷过程在适应性上具有更大的优势。例如，作为敏捷过程典型代表的极限编程，迭代开发是其核心思想之一。

经典的软件工程方法旨在定义一套完整的过程规范，使软件开发的运作就像设备的运转，人在其中像是可以更换的零件，不论是谁参与其中，该设备都能完好地运转，因此它是面向过程的。这种做法对于许多软件公司来说是有效的，因为开发进度是可预见的，流程的方法能固化与复用，还可节省人力成本，人员的流动不会对软件开发构成影响。敏捷过程也非常强调人的作用，没有任何过程方法能够代替开发团队中的成员，因为实施过程方法的主体便是人；而过程方法在其中所起的作用，则是对开发团队的工作提供辅助支持。

### 3.1.2 敏捷开发原则

敏捷开发提出了以下 12 条原则。

(1) 我们最优先要做的是通过尽早地、持续地交付有价值的软件来使客户满意。

(2) 即使到了开发的后期，也欢迎改变需求。敏捷过程利用变化来为客户创造竞争优势。

(3) 经常性地交付可以工作的软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。

(4) 在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。

(5) 围绕被激励的个体来构建项目，给他们提供所需的环境和支持，并且信任他们能够完成工作。

(6) 在团队内部，最具有效果并富有效率的传递信息的方法，就是面对面地交谈。

(7) 工作的软件是首要的进度度量标准。

(8) 敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一个长期的、恒定的开发速度。

(9) 不断地关注优秀的技能和好的设计会增强敏捷能力。

(10) 简单是最根本的。

(11) 最好的构架、需求和设计出于自组织团队。

(12) 每隔一定时间，团队会在如何才能更有效地工作方面进行反省，然后相应地对自己的行为进行调整。

敏捷开发是针对强调过程中未能解决的问题而提出来的。针对一些重型软件过程中重过程、轻人员的缺点，敏捷开发提出了把软件开发的模式从以“过程”为重心转到以“人”为重心的方向上来。软件是人开发出来的，而开发人员执行过程不可能像计算机执行软件一样严格，开发过程也不可能被非常详细地计划出来。试图把开发过程进行详细的分解、计划和跟踪，无论在技术上还是成本上都有难度。敏捷开发正是基于此提出了一套轻量级的方法。该方法一经提出，就得到软件界的欢迎。但是，敏捷开发对于开发人员的技能、职业素养、开发团队的文化氛围都有较高的要求。

# 3.2 Scrum 软件开发过程



## 3.2.1 Scrum 思想


视频讲解

Scrum (英式橄榄球争球队)软件开发模型是敏捷开发的模型之一。Scrum 的基本假设是开发软件就像开发新产品,无法一开始就定义软件最终的方案,过程中需要研发、创意、尝试错误,所以没有一种固定的流程可以保证方案成功。Scrum 将软件开发团队比拟成橄榄球队,团队有明确的最高目标,熟悉开发流程中所需具备的最佳典范与技术,具有高度自主权,通过紧密地沟通合作,以高度弹性解决各种问题,确保每天、每个阶段都朝向明确目标推进。

Scrum 开发流程通常以 30 天或者更短的一段时间为一个阶段,由客户提供新产品的需求规格开始,开发团队与客户在每一个阶段开始时挑选要完成的软件需求,开发团队必须尽力于 30 天后交付结果,团队每天用 15 分钟开会检查每个成员的进度与计划,了解其所遭遇的困难并设法排除。

与传统的软件开发模型如瀑布模型、螺旋模型或迭代模型相比,Scrum 模型的一个显著特点就是能够尽快地响应变化。当然,随着系统内外部因素的复杂度增加,采用 Scrum 模型的软件项目成功的可能性会迅速降低。

## 3.2.2 Scrum 术语与过程

Scrum 是一种迭代的增量软件开发过程,通常用于敏捷软件开发,包括了一系列实践和预定义角色的过程框架。Scrum 主管负责维护过程和任务,产品负责人代表利益所有者,开发团队则包括了所有开发人员。虽然 Scrum 是为管理软件开发项目而开发的,但它同样可以用于运行软件维护团队,或者作为计划管理的方法。下面是 Scrum 模型常用的一些术语与过程。

(1) Backlog(订单):可以预知的所有任务,包括功能性的和非功能性的。

(2) Sprint(冲刺):一次迭代开发的时间周期,一般最多以 30 天为一个周期。在这段时间内,开发团队需要完成一个制订的 Backlog,并且最终成果应是一个增量的、可以交付的产品。

(3) Sprint Backlog(冲刺订单):一个 Sprint 周期内所需要完成的任务。

(4) ScrumMaster(Scrum 负责人):负责监督整个 Scrum 进程,是修订计划的一个团队成员。

(5) Time-box(时间盒):一个用于开会的时间段,比如一个每日 Scrum 站立会议的 Time-box 为 15 分钟。

(6) Sprint 计划会议:在启动每个 Sprint 前召开,一般为一天时间(默认为 8 小时)。该会议需要制订的任务是产品客户和团队成员将 Backlog 分解成小的功能模块,决定在即将进行的 Sprint 里需要完成多少个小功能模块,并确定这个产品 Backlog 的任务优先级。另外,该会议还需详细地讨论如何才能按照需求完成这些小功能模块,完成这些模块的工作量以小时计算。

(7) 每日 Scrum 站立会议：开发团队成员召开每日 Scrum 站立会议，一般为 15 分钟。每个开发成员需要向 ScrumMaster 汇报 3 方面的内容：今天完成了什么？是否遇到了障碍？即将要做什么？通过该会议，团队成员可以相互了解项目进度。

(8) Sprint 评审会议：在每个 Sprint 结束后，项目团队将这个 Sprint 的工作成果向产品客户和其他相关的人员演示，并进行评审。一般会议不超过 4 小时。

(9) Sprint 回顾会议：对刚结束的 Sprint 进行总结。会议的参与人员为团队开发的内部人员。一般会议不超过 3 小时。

实施 Scrum 的过程如下。

(1) 将整个产品的 Backlog 分解成 Sprint Backlog，这个 Sprint Backlog 是按照目前的人力、物力条件可以完成的。

(2) 召开 Sprint 计划会议，划分和确定这个 Sprint 内需要完成的任务，标注任务的优先级并分配给每个成员。注意这里的任务是以小时计算的，并不是按人天计算。

(3) 进入 Sprint 开发周期，在这个周期内，每天需要召开每日 Scrum 站立会议。

(4) 整个 Sprint 周期结束，召开 Sprint 评审会议，将成果演示给产品的客户看。

(5) 团队成员最后召开 Sprint 回顾会议，总结问题和经验。

(6) 这样周而复始，按照同样的步骤进行下一次 Sprint。

Scrum 过程会产生以下的文档。

(1) 产品订单(product backlog)，是整个项目的概要文档。产品订单包括所有所需特性的粗略的描述，与将要创建的是什么产品有关。产品订单是开放的，每个人都可以编辑。产品订单包括粗略的估算，通常以天为单位。估算将帮助产品负责人制订时间表和衡量优先级。例如，如果“增加拼写检查”的需求估计需要花 3 天或更多的时间完成，那么这将影响产品负责人对该需求的期望。

(2) 冲刺订单(sprint backlog)，是大大细化了的文档，包含团队如何实现下一个冲刺的需求的信息。任务被分解为以小时为单位，没有任务可以超过 16 个小时。如果一个任务超过 16 个小时，那么它就应该被进一步分解。冲刺订单上的任务不会被分派，而是由团队成员签名认领他们喜爱的任务。

(3) 燃尽图(burn down chart)，是一个公开展示的图表，显示当前冲刺中未完成的任务数目，或在冲刺订单上未完成的订单项的数目。不要把燃尽图与挣值图相混淆。燃尽图可以使 Sprint 平稳地覆盖大部分的迭代周期，且使项目仍然在计划周期内。

和所有其他形式的敏捷软件过程一样，Scrum 需要频繁地交付可以工作的中间成果。这使得客户可以更早地得到可以工作的软件，同时使得项目可以变更项目需求以适应不断变化的客户需求。风险缓解计划由开发团队自己制订，开发团队在每一个阶段根据承诺进行风险缓解、监测和管理。

计划和模块开发的透明，让每一个人知道谁负责什么，以及什么时候完成。频繁召开所有相关人员会议可以跟踪项目进展。进行发布、客户、员工、过程等仪表板更新和所有相关人员的变更，必须要有预警机制，如提前了解可能的延迟或偏差。

Scrum 过程认为，认识到或说出任何没有预见的问题并不会受到惩罚，在工作场所和工作时间内必须全身心投入，完成更多的工作并不意味着需要工作更长时间。

# 3.3 极限编程



## 3.3.1 什么是极限编程


视频讲解

极限编程(Extreme Programming,XP)是一种软件工程方法学,是敏捷软件开发中最富有成效的几种方法学之一,是由著名软件工程学者 Kent Beck 于 1996 年提出的。极限编程具有强沟通、简化设计、迅速反馈等特点,一般只适合于规模小、进度紧、需求不稳定、开发小项目的小团队。

极限编程的支持者认为软件需求的不断变化是很自然的现象,是软件项目开发中不可避免的,也是应该被欣然接受的现象。他们相信,和传统的在项目起始阶段定义所有需求再费尽心思地控制变化的方法相比,有能力在项目周期的任何阶段适应变化将是更加现实和更加有效的方法。

对比传统的项目开发方式,极限编程强调把它列出的每个方法和思想都做到极限、做到最好,其他极限编程所不提倡的,则一概忽略。例如,开发前期的整体设计,不是特别重要则不需要花太多时间去做。一个严格实施极限编程的项目,其开发过程应该是平稳、高效和快速的,能够遵照一周 40 小时工作制且不拖延项目进度。

与一般流行的开发过程模型相比,极限编程具有如下的优点:

(1) 极限编程模型是“轻量型”或“灵活”的软件过程模型,并且可与面向对象语言结合起来,提供了一种很有特点的软件开发解决方案。

(2) 极限编程被用来解决大型软件开发过程所遇到的问题的方法,可以称为“专家协作”的开发方式。

极限编程为管理人员和开发人员开出了一剂指导日常实践的良方,这个实践意味着接受并鼓励某些特别有价值的方法。支持者相信,这些在传统的软件工程中看来是“极端的”实践,将会使开发过程比传统方法更好地响应用户需求,因此更加敏捷、更好地构建高质量软件。

极限编程的一个成功因素是重视客户的反馈,其目的就是满足客户的需要。极限编程强调团队合作,经理、客户和开发人员都是开发团队中的一员,团队通过相互之间的充分交流和合作,使用 XP 这种简单但有效的方式,努力开发高质量的软件。其中,程序员们通过测试获得客户反馈,并根据需求变化修改代码和设计,他们总是争取尽可能早地将软件交付给客户,能够勇于面对需求和技术上的变化。

## 3.3.2 极限编程的要素

极限编程有交流、简单、反馈和勇气 4 个要素。Kent Beck 和 Martin Fowler 把这 4 个要素统一起来,将其作为极限编程的精髓。

**1. 交流**

(1) 开发人员与客户的交流。开发人员与客户的有效交流是软件开发前期必不可少的,因为这些交流将直接决定一个项目是否能够符合客户的要求。在极限编程中,需要一个非常精通业务的现场客户,他们不仅随时提供业务上的信息,而且要编写业务验收测试的测试代码,这样就可以在很大程度上保证项目的方向不会出错。

(2) 开发人员之间的交流。在当前软件开发的过程中,项目经理们都会强调团队精神。因为在传统的教育中,人们形成的是一种独立解决问题的能力,所以,遇到问题时人们习惯自己解决,而不是和其他人合作。

(3) 开发人员与管理人员的交流。在一个项目组里面,开发人员和管理人员之间的关系是影响项目的一个非常重要的因素,如果处理不好,可能会直接导致一个项目的失败。其对管理人员所具备的素质要求很高,如果开发人员能够和管理人员进行良好的交流,他们的工作环境就会得到很大的改善,往往并不一定要非常豪华的房间和高级的家具,只需要一个非常舒服的工作环境,就可以让一个团队的战斗力得到很大的提升。而且,对于一个项目的计划和预算,如果开发人员能够提出自己的想法,就会避免其争取到了项目最终却得不到利润的情况的出现。

**2. 简单**

(1) 设计简单。极限编程提倡一种简单设计的观点,这样做的好处是不需要在设计文档上面花费太多的时间,因为文档没有不修改的,一般情况下在一个项目结束的时候,会发现当初的文档已经面目全非了。因此,在软件开发的前期,设计工作要做的就是确定需要实现的最重要的功能。简单的设计并不意味着这些设计是可有可无的,相反,简单的几页纸比厚厚的几十页甚至上百页更加重要,因为一个项目的核心内容都在上面,所以在编写的过程中一定要慎重。

(2) 编码简单。编码的简单表现在迭代的过程中,极限编程不需要一次完成所有需要的功能,相反,变化在极限编程中是被提倡的。我们可以先简单地实现一点功能,然后添加详细的内容,再对程序进行重构,最终的代码将是非常简单的,因为依照重构的原则进行修改了之后,所有的类和函数、过程都是非常简短而非冗长的,每一个模块完成的功能也是非常明确的。

(3) 注释简单。在某些项目中,有时对注释要求很严格。一般,程序员与其在程序中添加注释来解释程序,不如用大家都能够理解的变量、过程和函数名作为名称,使注释简单化。程序员要编写的是代码,如果带有太多无关紧要的注释,不仅会浪费开发时间,还可能引起歧义。

(4) 测试简单。在极限编程中,测试主要是通过编写测试代码来自动化完成的,特别是在一些面向对象的编程环境中,可以使用 xUnit 工具来快速、有效地进行单元测试。每一次修改了程序之后,都要运行测试代码来检查程序是否有问题。而且对于程序的集成,极限编程提倡的是持续集成,也就是不断将编写好的通过了单元测试的代码模块集成到编写完毕的系统中,在那里可以直接进行 Test Suit 的集成测试,从而保证代码不会影响到整个系统。

**3. 反馈**

(1) 客户对软件的反馈。极限编程强调现场客户的重要性。因为一旦有了现场客户,就能够随时对软件做出反馈,能够保证在"反馈"的过程中不断调整,从而把控软件前进的方向。现场客户的选择也很重要,他们的选择将直接影响一个项目的开发,一个好的现场客户不仅可以准确地把握软件的方向、回答业务问题,而且可以编写验收测试,保证软件中的业务数据没有错误。这样就要求现场不仅有一个管理人员,而且计算机的水平也要有一定的高度。

(2) 测试代码对功能代码的反馈。极限编程强调的是先测试后编程的思想,也就是说

在编写功能代码之前就先要编写测试代码,测试代码可以用来保证功能代码运行正确。因此,开发人员要有一定的测试理论知识,明白需要采用什么样的数据作为测试用例,这样才能够做好测试,保证程序的质量。另外,测试代码的编写不是一次就能完成的,随着功能的不断添加,测试代码也同样需要改变,应在保证原有代码没有问题的前提下,继续编写新的代码。

**4. 勇气**

项目开始时,一般由管理人员为开发人员分配任务,但这种分配只不过是根据管理人员自己对每个人的大致估计来完成的,所以很难做到令每一个人都满意。事实上,凭管理人员主观的判断来给大家分配任务,一定会有一些人对自己的任务不够满意。在这个时候,管理人员应将所有的任务公布给大家,让开发人员自主选择想要做的任务。这样,由于任务是自己选定的,那么满意度会有很大程度的提高。

在这种情况下,开发人员要有接受任务的勇气,如果所有的人都选择自己觉得容易的任务而回避困难的任务,那么开发就肯定会失败了。在这个时候,管理人员应该采取适当的方式鼓励开发人员,让其能够选择一些具有挑战性的任务,如此对于其个人能力的提高也是很有好处的。


视频讲解

## 3.4 结对编程

极限编程的实践中有一个非常重要的原则就是结对编程,这里所谓的结对编程并非是一个人在编程,另一个在旁边看着的形式。在结对编程中,另外一个人同样起着非常重要的作用,他需要帮助正在编码的人找到低级的失误,防止其编码出现方向性的错误,特别是在出现一个正在编码的人不擅长解决的问题的时候,他会直接替换这个人来进行编程。这样做的好处也许只有在实践了之后才能够体会到,它不仅可以避免一些错误的发生,而且可以通过直接的讨论来解决一些容易产生歧义的问题,更加快速地完成开发。此外,在交流的过程中,大家的水平也会得到快速提高,因此结对编程的过程也是开发人员学习软件开发知识和提升能力的过程。

### 3.4.1 什么是结对编程

结对编程(Pair Programming)是一个非常直观的概念,简单地说是指两位程序员肩并肩地坐在同一台计算机前,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,一起编写文档等。基本上在所有的开发环节都面对面、平等、互补地进行开发工作,并且这两人的角色可以随时交换。

结对编程是一个合作式编程模式,是在必要的软件开发环节(如需求分析、设计、编码、测试、评审等)中,让两名程序员合作来完成同一个任务。Williams 等把结对编程定义为“在结对编程中,两名程序员合作开发同一产品模块”。这两名程序员就像是一个联合的智慧有机体,共同思考问题,负责产品模块的各个方面。一名结对者作为驾驶员(driver),控制鼠标或键盘并编写代码;另一名结对者作为导航员(navigator),主动持续地观察和辅助驾驶员的工作,如找出代码的缺陷、思考替换方案、寻找资源和考虑策略性的暗示等。结对

双方周期性地交换角色。在这个过程的任何时候双方都是平等活跃的参与者，并且不管是在一个上午还是整个项目的工作中，双方都完全分享所获得的工作成绩。

目前，有关结对编程的理论基本上来自国外。1995 年，澳大利亚悉尼科技大学计算机科学教授、国际公认的软件工程理论与实践权威人士 Larry Constantine 在专栏中第一次提到他所观察到的一个现象："两个程序员一起工作，可以比以往更快地完成经过测试的代码，而且这些代码几乎是没有错误的。"这便是结对编程概念的雏形。结对编程是极限编程的 12 个主要实践之一，它吸收了合作式编程(Collaborative Programming)的关键思想，强调合作和交流。随着敏捷开发思想和极限编程方法在 21 世纪初的快速普及，结对编程也迅速被大家熟知和尝试应用。

结对编程的结对角色分为驾驶员和导航员：

(1) 驾驶员控制鼠标和键盘的使用，负责编码工作。

(2) 导航员在驾驶员一旁观察和思考，负责检查错误和考虑解决方案。

结对角色是需要互换的。若驾驶员的编码活动停滞不前或者出现方向性的错误，结对双方可交换角色，让导航员转为驾驶员继续编码。这种角色互换应该经常发生，有时可能每隔几分钟(甚至更频繁地)就互换一次。一旦结对者习惯了这种方式，并且适应了另一个结对人员，结对者就会进入这种流程，很自然地互换角色。

大量的实验以及研究表明，结对编程具有如下的优点：

(1) 最大化地提高工作效率。软件开发并不只是程序员堆砌代码的过程，它更多的是一个创新的过程，是一个发现问题、分析问题、解决问题的过程。若一个人编程时，有了一丝零碎的想法就开始编写代码，写完代码之后却忽然发现这个方案行不通，于是只好废弃这些代码，重新开始新的想法。而结对编程则不同，一个人有了想法，首先要表达出来，让自己的同伴理解，经过深刻的讨论，一致认可之后才开始编写代码。一个人编写代码，另一个则在旁边思考，为下一步的工作提出建设性的意见，发现了问题可以及时指正，大大地提高了代码质量。在开发过程中，设计思考和编码实现不停地交换，保持了良好的开发节奏。结对双方互相督促，使彼此更加认真地工作。遇到问题和压力时，也可以一起面对，互相鼓励，同时一起分享解决问题的成就感和乐趣。

(2) 生成高质量的代码。两个人的智慧确实胜过一个人的，对于影响整个系统的设计决策更是如此。两个人编写的代码总比一个人写的代码好，无论一个程序员多么聪明，别人的意见仍有助于避免由于无知、自大或疏忽而产生错误决策。虽然许多程序员保持专心致志没有问题，但是让其他人帮助这些程序员不出闪失也是有好处的，特别是当程序员尝试解决困难的问题时，或当程序员想要放弃时，旁边有人鼓励和协助有利于继续前进。

(3) 降低风险。风险会使大多数团队停滞不前。在团队的软件开发项目中，管理者有时会想要做但却不敢冒险去做一些事，这是大多数管理者求稳的结果。降低风险的最佳方法是确保团队中的每个人都完全熟悉系统的所有部件以及对系统的所有更改。技术讲解和设计文档很有用，但对于大多数快节奏的项目，它们并不能很好且迅速地传播知识，而传播知识最有效的方法是让一个知道代码的人与另一个不知道代码的人一起解决问题。

(4) 是知识传播的最好途径。很多软件公司都建有自己的知识库，有的还建立了自己的培训部门，甚至高薪聘请一些专家做技术培训，但往往发现效果并不理想。而与有经验的同事一起结对则是在实际项目中学习，具有非常强的针对性。你学到的不仅是一些技术和

技巧,更多是他们思考问题的方式、解决问题的方法。和各种具有不同经验的同事一起结对,你的能力可以得到快速的提高。

(5) 打造出最佳的合作团队。团队是有组织有计划、合理有效地利用各种资源,进行最佳的组合。结对并不是一对固定的伙伴,结对编程鼓励在团队中经常交换结对伙伴。这时,项目不再是一个人的事情,也不是两个人的事情,而是整个团队的事情。通过结对,大家可以在最短的时间内完成磨合。结对能很好地促进团队的沟通交流,经常一起合作结对的伙伴彼此了解、熟悉,很多都是工作和生活上的好友。在这样的团队里,大家很乐意互相协助,一起分享知识,分享快乐。

现在越来越多的项目都交给由不同地理位置的员工组成的虚拟团队来完成,很多开源软件也都是由分布在世界各地的开发者共同完成的,这使得结对编程很难应用于这样的虚拟团队,因为结对编程要求两个开发人员坐在一台计算机前去共同完成一个开发目标,以达到优势互补的目的。

另外,现有的即时通信软件也不能帮助两个开发者共同编辑一个源代码文件,地理位置的限制使得实施结对编程变得几乎不可能。为了有效地支持软件分布式开发,提高软件协同开发环境的易用性和有效性,国外软件工程方面的专家提出分布结对编程的概念,这是对结对编程的探索,从而发挥了结对编程在分布式软件开发中的作用。

### 3.4.2 结对编程方式

**1. 面对面结对编程**

面对面结对编程是指两个程序员肩并肩坐在同一台计算机前在同一个软件制品上一起工作的软件开发方式。面对面结对编程有许多可以得到证明的好处:直接快速的交流、高质量的代码和增强程序员工作的乐趣。

面对面结对编程最大的好处就是交流非常方便,因为两个人靠得很近,言语和手势的交流非常自然,效果也非常好。甚至遇到困难的问题时,两个人可以拿起一张纸,通过绘制各种图形和书写文字来表达问题,很容易达成一致,从而快速地解决问题。

**2. 分布结对编程**

在结对编程环境中,基础设施缺乏、地理位置分离和时间安排冲突这些障碍经常给结对编程带来困难。分布结对编程让程序员在不同的地方进行合作编程成为了可能。软件行业发展的一个大趋势就是软件的全球化。这个趋势背后的驱动因素包括软件公司雇佣不同国家或城市的高水平程序员,为客户就近成立研究小组,创造快速虚拟发展小组,持续做一些关键性的项目,即使他们不在一个时区也没关系。

近几年,灵活的软件方法在业界已经引起越来越多的关注,而极限编程被认为是这些灵活工具中最重要的一种。尽管已经有一些工具能够比较好地支持分布式灵活软件开发,我们仍然有必要对分布式极限编程的工具和处理进行更多的研究,特别是在提供共享编码方式的扩展解决上。鉴于全球化软件发展的趋势,要求两名开发者进行面对面的交流并不符合全球化软件发展的需求,其要求两名程序员虽然在不同的地点,但是他们还能一起合作使用结对编程方式编写代码,这种方法称为分布结对编程。

分布结对编程是一种编程风格,两个程序员在地理上是分布的,通过网络在同一个软件制品上同步工作。相对于面对面结构编程,分布结对编程可以支持结对者,结对者通过网络

可以随时随地结对工作，大大提高了结对的概率。为了进行分布结对编程，需要功能较为强大的结对工具支持结对者高效地工作。

## 3.5 小　　结

敏捷开发强调快速响应软件的变化，充分发挥人的能动作用。Scrum 是一种用于敏捷开发的迭代式增量软件开发过程，包括一系列实践和预定义角色的过程框架。极限编程和结对编程是敏捷过程的两个成功的重要实践。极限编程的思想是开发人员要做到极致。极限编程有交流、简单、反馈和勇气 4 个要点，它们统一构成了极限编程的精髓。结对编程要求两个程序员通过合作完成同一个任务，互相审查以减少编程错误，提高代码质量。结对编程方式分为面对面结对编程和分布结对编程两种。面对面结对编程要求两个程序员肩并肩坐在一起完成编程任务，而分布结对编程允许程序员在不同地方通过网络进行协同工作，提高了结对工作的效率。

## 习　　题

1. 什么是敏捷过程？简述敏捷开发的原则。

2. 阐述 Scrum 的特点和过程。举例说明相比传统的过程模型，Scrum 的优势是什么。

3. 简述极限编程的思想，举例阐述极限编程的要素。

4. 什么是结对编程？简述结对编程的优缺点。

5. 简述面对面结对编程相对于分布结对编程的优势与不足，并举例说明如何克服这些不足。

6. 结对编程有哪些角色？简述交换角色的目的。