采用XP方法使软件项目获得更大成功 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/149/2021_2022__E9_87_87_E7_94_A8XP_E6_96_c41_149682.htm 使用面向对象编程变得空前普及。它使软件开发发生了某种程度上的变革，但最近的研究表明，有半数的软件开发项目滞后，而三分之一的项目则超出预算。问题不在于技术，而是开发软件所使用的方法。所谓的“ 轻量型 ”或“ 灵活 ”方式，与面向对象语言的威力和灵活性结合起来，提供了一种很有意思的解决方案。最常见的灵活方式称为极端编程（Extreme Programming）或者 XP，但许多人并不真正了解它。对软件项目使用 XP 可以大大增加成功的机会。本文提供了 XP 的概述，并解释了它为什么很重要 -- 不是传言，也没有骗局。 在过去的十年中，CEO 们在产生稳步增加的收入方面面临巨大的压力。他们通过在许多方面采取一系列举措来解决这一问题，例如缩小公司规模、外包、再工程、企业资源规划 (ERP) 等等。这些对低效率的解决措施让 S&amp.P 500 中的许多企业在 90 年代末能够连续几年保持两位数的收入增长。但这种方式也带来了一些负面影响。 在 Gary Hamel 所著的 Leading the Revolution（请参阅参考资料）一书中，他声称已有一些迹象证明传统企业模式的优势已不那么明显。我们必须找到一些替代方法来为收入的持续增长提供动力。他建议唯一能让公司继续增长的办法是进行一次彻底的创新。我们认为在软件开发领域中尤其需要这样。 企业问题 如果使用标准软件开发方法，那么即使在常用的平台上进行开发，也要做好失望的准备。如图 1 所示，最近的研究表明，有一半项目将滞后，而三分之一的

项目将超过预算。这一推测比 1979 年由美国总审计局的研究结果好不了多少。 图 1. 软件项目成功和失败，过去和现在 如果我们希望这些数字有显著提高，则需要彻底创新的方法来开发软件。有两个主要因素影响现有的方法： 惧怕失败、对软件本质的误解。 没有人打算失败。具有讽刺意味的是为使失败最小化而创建的方法是失败的。对软件的误解是问题的根源。恐惧实际上只是一种症状。现有的方法是由那些有良好愿望但忘记了软件中的“ 软 ”的那些聪明人所创建的。他们假定开发软件就象造桥。因此他们从各种设计规范中借鉴了一些适用于“ 硬 ”物体（例如桥梁）的最优方法。结果是基于 Big Design Up-front (BDUF) 思想的反映迟钝的开发方法，软件不堪一击，人们无法使用它们。〈一〉一种解决方案：灵活方法 最近发生了一些转变，从所谓的“ 重量型 ”方法转向了“ 轻量型 ”或“ 灵活 ”方法，例如 Crystal 方法、适应性软件开发和（当前最流行的）XP。所有这些过程都有这样一个事实，即需要人们共同来开发软件。成功的软件过程必须将人们的长处最大化，将他们的缺点最小化，因为优点和缺点毋庸质疑都存在。在我们看来，XP 最出色的地方在于它能够解决所有影响参加人员的互补力量。 XP 提供了十年来最大的一次机会，给软件开发过程带来彻底变革。就象 Peopleware 作家 Tom DeMarco 所说，“ XP 是当今我们所处领域中最重要的一项运动。预计它对于目前一代的重要性就象 SEI 及其能力成熟度模型对上一代的重要性一样。 ”XP 规定了一组核心价值和方法，可以让软件开发人员发挥他们的专长：编写代码。XP 消除了大多数重量型过程的不必要产物，通过减慢开发速度、耗费开发人员的精力（例如干特图、状

态报告，以及多卷需求文档）从目标偏离。我们认识到一个称为“ 极端编程 ”的东西可能很难作为正式的开发过程推荐给管理层，但如果您的公司从事软件行业，您应该帮助管理层绕过其名称认识到 XP 可以提供的竞争优势。 Kent Beck 在他所著的 Extreme Programming Explained: Embrace Change 一书中概括了 XP 的核心价值（请参阅参考资料）。我们对它们进行了总结：1）交流 项目的问题往往可以追溯到某人在某个时刻没有和其他人一起商量某些重要问题上。使用 XP，不交流是不可能的事。2）简单 XP 建议您总是尽可能围绕过程和编写代码做最简单的事情。按照 Beck 的说法，“ XP 就是打赌。它打赌今天最好做些简单的事...而不是做更复杂但可能永远也不会用到的事。 ”3）反馈 更早和经常来自客户、团队和实际最终用户的具体反馈意见为您提供更多的机会来调整您的力量。反馈可以让您把握住正确的方向，少走弯路。4）勇气 勇气存在于其它三个价值的环境中。它们相互支持。需要勇气来相信一路上具体反馈比预先知道每样事物来得更好。需要勇气来在可能暴露您的无知时与团队中其他人交流。需要勇气来使系统尽可能简单，将明天的决定推到明天做。而如果没有简单的系统、没有不断的交流来扩展知识、没有掌握方向所依赖的反馈，勇气也就失去了依靠。 XP 的方法将这些价值转换成开发人员每天应做的事情。这里没什么新鲜内容。多年以来，行业认识到 XP 方法是“ 最优方法 ”。实际上，XP 中的“ 极端 ”来自两方面： XP 采取经过证明的业界最优方法并将其发挥到极致。 XP 将这些方法以某种方式进行结合，使它们产生的结果大于各部分的总和。 这是怎样的情景呢？代码复查是个好的做法，因此始终通过成对地编写

代码来做到。测试也是个好的做法，因此总是通过在编写代码之前编写测试来做到。文档很少与代码保持一致，因此只做那些最需要的事，余下的部分则取决于明确编写的代码和测试。XP 不保证人们总做正确的事，但它允许人们这样做。它将这些“ 极端 ”方法以一种相互支持的方式结合起来，显著提高了速度和有效性。〈二〉XP 的十二种方法 XP 的十二种方法（如图 2 所示）将其定义为规则。让我们仔细研究每一个方法来对“ 执行 XP ”表示什么有个更全面的理解。 图 2. XP 的十二种方法1）规划策略 有些人会指责 XP 是一种美其名的剽窃，只是一群牛仔在没有任何规则的情况下将一个系统拼凑在一起。错。XP 是为数不多的几种承认您在开始时不可能事事通晓的方法之一。无论是用户还是开发人员都是随着项目的进展过程才逐步了解事物的。只有鼓励和信奉这种更改的方法才是有效方法。状态限定方法忽视更改。而 XP 则留心更改。它倾听所用的方法就是“ 规划策略 ”，一个由 Kent Beck 创造的概念。 这一方法背后的主要思想是迅速地制定粗略计划，然后随着事物的不断清晰来逐步完善。规划策略的产物包括：一堆索引卡，每一张都包含一个客户素材，这些素材驱动项目的迭代；以及对下一两个发行版的粗略计划，如 Kent Beck 和 Martin Fowler 在他们的 Planning Extreme Programming 中描述的那样（请参阅参考资料）。让这种形式的计划得以发挥作用的关键因素是让用户做企业决策，让开发小组做技术决策。如果没有这一前提，整个过程就会土崩瓦解。 开发小组要决定：估计开发一个素材要花多长时间、使用各种技术选项所花费的成本、团队组织、每个素材的“ 风险 ”、迭代中素材开发的顺序（先开发风险最大的那一个

可以减轻风险）。客户需要决定：范围（一个发行版的素材和每一次迭代的素材）、发行日期、优先级（根据企业价值先开发哪些特性）规划经常发生。这样，在客户或开发人员学习新事物的同时，就为他们调整计划提供了频繁机会。