从瀑布到迭代
开发 —
项目经理面临艰巨的转换

Philippe Kruchten

Rational Software 白皮书

TP 173, 5/00

Rational®
the software development company

目录

介绍 1

迭代开发 1

优点：迭代开发的好处 2

降低风险 2
接受变化 3
边做边学 3
增加了复用机会 3
更好的总体质量 3

难点：意外的下降趋势和普通陷阱 4

预先注意到返工 4
将软件放在第一位 5
及早解决难题 5
由于不同生命周期模型而冲突 6
计算进度的方法不同 7
确定迭代的次数、持续时间和内容 7
好的项目经理和好的架构设计师 8

总结 9

关于作者 9

参考资料和更多读物 9

简介

Rational Unified Process（RUP）提倡用迭代或者螺旋形的方法来应对软件开发生命周期，因为该方法多次被证明比瀑布方法在很多方面要优秀。但是先不要认为迭代生命周期提供的许多好处是免费的。迭代开发不是一根魔棒，它不能挥一下就解决软件开发中的所有可能问题或者困难。项目不是因为是迭代的就变得更容易建立、计划或控制。当风险较高而且可能发生早期失败时，项目经理实际上会有更具挑战性的任务，特别是在他或她第一个迭代项目中，并极可能在那个项目的早期迭代过程中。本文中，我从项目经理的角度来描述迭代开发的某些挑战性。我也会描述一些 Rational 中的常见“陷阱”或者缺陷，在我们以往的咨询或从 Rational 同事的报告或“战斗”故事中，我们看到有项目经理碰到过这些问题。

迭代开发

经典的软件开发流程是依据瀑布生命周期的，这会在后面的图中作说明。在该方法中，如图 1 所示，开发按照线性进行，从需求分析到设计、编码和单元测试、子系统测试以及系统测试，其中很少有关于前一阶段结果的反馈。

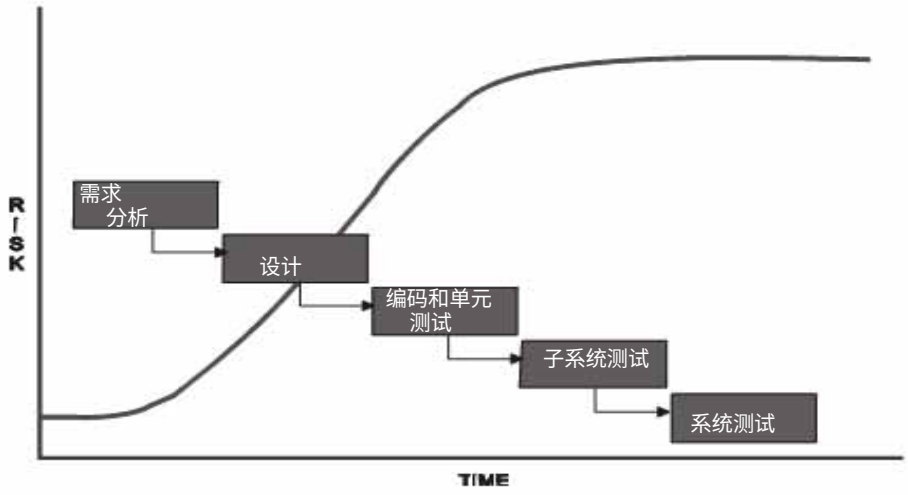

图 1：瀑布开发流程

该方法的根本问题是它及时将风险向前推进。要知道，改正上一阶段中的错误是要花昂贵代价的。初始设计可能因为关键需求而产生缺陷，并且，晚发现设计缺陷会导致成本无法控制和／或项目取消。瀑布模型会掩盖项目的真正风险直至无可救药时。

有一个办法可以用来代替瀑布方法，那就是迭代和递增流程，如图 2 所示。

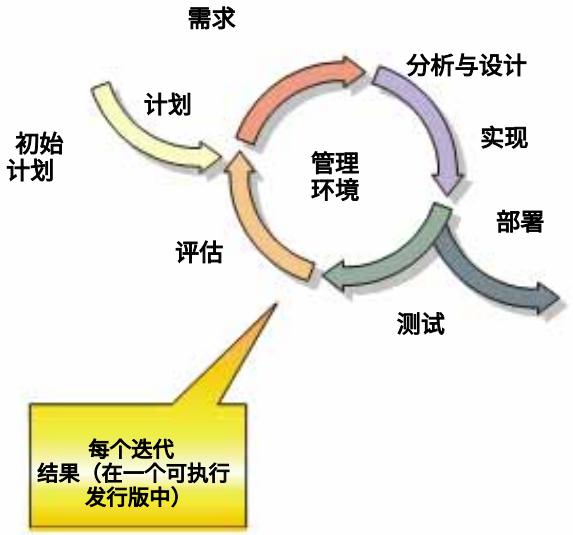

图 2：迭代开发方法

在该方法中，建立在 Barry Boehm 的螺旋模型工作上（请参见『更多读物』），在生命周期的早期强制确定项目中的风险，那时我们有可能攻击并使用有效的办法及时进行反应。该方法是使用每个迭代来进行不断发现、发明和实现的一种，而该迭代能促使开发团队以可预计和可重复的方式来关闭项目工件。

优点：迭代开发的好处

和传统的瀑布流程相比，迭代流程有很多优势。

1. 在生命周期的早期严重的误解会更容易发现，此时还有可能对它们做出反应。
2. 它允许并且鼓励用户反馈，从而使系统真正的需求更加明确。
3. 开发团队不得不重视那些对项目很关键的问题，那些问题往往会遮住团队成员的视线，使他们的注意力不能集中在项目真正的风险上。
4. 连续的迭代测试能对项目状态进行客观评估。
5. 能及早地发现需求、设计和实现之中的不一致。
6. 团队（尤其是测试团队）的工作负载会更平均地分布在整个生命周期。
7. 该方法能使团队平衡所学的课程，从而不断改进流程。
8. 项目干系人能得到整个生命周期中项目状态的真实反映。

降低风险

迭代流程让您能及早降低风险，因为一般只在发现或处理风险时才会进行集成。当您开始早期迭代时，您检查了所有的流程组件，练习项目的很多方面：工具、即用软件和人员技能。感觉到的风险经证明并不是风险，但会发现新的、未被怀疑过的风险。

如果一个项目因某个原因而失败，那么趁还未投入大量的时间、人力和金钱，让它尽早地失败。不要回避风险，而是应该面对风险。在其他风险（例如构建错误的产品）中，迭代开发流程有助于及早降低两类风险：

- 集成风险

- 结构体系风险

由于错误在若干个迭代过程中得到了纠正，因此迭代流程可以生成更强健的体系结构。当产品度过先启阶段时，在早期迭代中检测到缺陷。性能瓶颈在还能对它们进行处理时发现，而不是临近交付时才发现。

集成并不是一个在生命周期结束时的“大爆炸”，而是逐步集成元素。实际上，我们推荐的迭代方法包含大多数不断的集成。它过去是耗时、不确定而困难的（在项目结束时占用全部工作量的 40%），现在分裂成 6 到 9 个更小的集成，它们以少得多的元素集成开始。

适应变化

您可以看见几类变化：

- 需求变化

  迭代流程让您考虑不断变化的需求。事实是需求通常会变化。需求变化和“需求蔓延”一直是项目问题的主要源由，将导致延迟交付、错过时间安排、客户不满意和挫败开发人员。但是通过向用户（或用户代表）提交产品的早期版，您可以保证产品更好地满足任务。

- 战术变化

  迭代流程向管理层提供产品战术变化的方法，例如，与现有产品竞争。您可以决定在减少功能的情况下及早发布一个产品来应对竞争对手的步伐，或者您可以接受另一家供应商提供的技术。您还可以通过重新组织迭代的内容来减少需要由供应商修正的集成问题。

- 技术变化

  从更浅的程度来说，迭代方法使您适应技术变化。您可以在精化阶段使用它，但是您应该在构造和转换阶段避免这类变化，因为它本来就具有风险。

边做边学

迭代流程的优势是开发人员可以一直学习，在整个生命周期中更充分地利用各种技能和专门技术。例如，测试人员早些开始测试，技术写作人员早些开始写作等，而在非迭代开发中，同样的人将等待开始他们的工作和不停地制定计划。需要培训，或需要其他（可能是外部的）帮助，这些都在评估复审的早期发现。

流程本身也可以在进行过程中得到改进和优化。迭代结束时的评估不仅从产品／进度表的角度查看项目状态，而且分析组织和流程中所需要的更改，以便在下一次迭代中更好地执行。

增加了复用机会

迭代流程简化了项目元素的复用，因为它可以更容易地确定通用部件是部分设计还是已实现的设计，而不是在开始时确定所有通用性。确定和开发可复用部件是困难的。架构设计师可以使用早期迭代中的设计复审确定未受怀疑的潜在复用，而在后续迭代中开发和完善该公用代码。正是在精化阶段的迭代中，发现了觉问题的常见解决方案，且确定了应用于整个系统的模式和体系结构机制。

更好的总体质量

迭代流程产生的产品质量总体上好于从传统的顺序流程中产生的产品。系统已进行了好几次测试，这提高了测试质量。需求已经过了优化，并更接近于用户真正的需求。到交付时，系统已经过了长时间运行。

难点：意外的下降趋势和普通陷阱

迭代开发不一定表示更少的工作和更短的时间安排。它主要的优势是给输出和时间安排提供更多的可预测性。它会产生更高质量的产品，这将会使最终用户的真正需求得到满足，因为您将有时间来演进需求、设计和实现。

迭代开发实际上包含更多的计划，因此它很有可能给项目经理带来更多的负担：需要开发总体计划，还要为每个迭代开发详细计划。它还包括在问题、解决方案和计划之间不断协商权衡。会更早地进行更多的体系结构计划。在每个修订版中，不得不重复修改、复审和批准工件（计划、文档、模型和代码）。战术变化和范围变化会导致不断地重新计划。因此，每次迭代中必须稍微修改团队结构。

陷阱：对于结果进行过度细化的计划

从头到底构造详细的计划通常是很浪费的，除了练习评估调度和资源的全球包装。该计划在到达第一次迭代的尾声之前是无用的。在具有正确的体系结构和牢固掌握需求之前（这大约发生在 LCA 里程碑阶段），您无法构造一个实际的计划。

所以，所计划的合并精度和您对计划的活动、工件或者迭代的了解是相关的。短期计划更详细和细致。长期计划以粗略的格式进行维护。

顶住无知或者不良的管理忍受尝试得来的“综合的整体计划”的压力。告诉并向管理者解释迭代计划的观念和在尝试预计未来细节的上所浪费的努力。一个类比非常有用：一辆车从纽约驶向洛杉矶，您计划了整条路线，但其实只需要详细的驾驶指导来让您离开这个城市，并踏上旅程。计划驾驶经过堪萨斯州的精确细节是不必要的，更不用说到达加州了，因为您发现通过堪萨斯州的道路正在修建而不得不寻找另一条路线等。

预先注意到返工

在瀑布方法中，在结束时会出现太多返工，比如在最后测试和集成期间寻找可恶的缺陷所带来的恼人且无法计划的结果。更糟的是，您发现大多数“故障”的原因来自于设计中错误，而您则想通过构建会引起更多故障的变通方法来减少实现中的故障。

在迭代方法中，您只需预先注意到将会有返工，且开始时会有很多返工：当您在体系结构原型的早期发现问题时，就需要修正它们。另外，为了构建可执行的原型，必须构建桩代码和大致结构，这些桩代码和结构在后来会被更多成熟和强健的实现所代替。在状况良好的迭代项目中，碎片或返工的百分比应该快速减少，随着体系结构的稳定和难题的解决，变化的影响应该迅速减小。

陷阱：没有聚合的产品

迭代开发不是说每次迭代都要废弃所有东西。随着一次次迭代，尤其在为体系结构设立 LCA 里程碑基线以后，碎片和返工必须减少。开发者常想利用迭代开发来完成管理镀金：来引入一项更好的技术，来执行返工等。项目经理必须保持警惕，不能允许返工一些没有被破坏，还很好的元素。还有，随着开发团队的壮大和人员的流动，新的人员加入了进来。他们可能有他们自己关于应该如何做事的想法。类似地，客户（或者他们的项目代表：市场营销、产品管理）可能会滥用迭代开发提供用来适应变化和／或无止境地更改或者添加需求的权力范围。这一影响有时称为“需求蔓延”。另外，项目经理在做权衡和协商优先级时，应保持冷静。关于 LCA 里程碑，需求是基线，除非重新协商调度和预算，否则任何变化都有有限的成本：即有得就有失。

记住“不求最好，但求更好。”（或在法语中“Le mieux est l’ennemi du bien.”）

陷阱：让我们开始，我们会决定接下来干什么

迭代开发不是说一直进行模糊的开发。您不应该简单地开始设计和编码以保持团队忙碌或者带有突然能出现明确目标的希望。您仍需要定义明确的目标，将它们记录下来，并且得到所有方面的同意；然后优化，扩展它们，并再次获得同意。在迭代开发的优点是您无需在开始设计、编码、集成、测试和验证它们之前陈述所有需求。

陷阱：牺牲别人，成就自己

重要的风险在项目即将结束，交给消费者时发生。我们是说用户从认为什么都没交付变成了认为团队确实会实现它。好消息是对于项目外表的感觉已经转变了：尽管在周一如果有东西交付了，用户会感到很高兴，在周二，他们开始担心不是所有的东西会交付。这是坏消息。在第一和第二个 beta 中，您会发现您处于人们关心的问题之中，这包括将它加入到第一个发行版中。突然，这些变成主要问题了。项目经理从担心交付最小可接受功能到面临所有最终需求现在对于第一次交付都是“必要的”的局面。尽管当这发生时，所有出色的项目都提升到了“A”优先级状态。事实是仍有相同数量的事情要做，并且需要相同时间量才能完成它们。当外表感觉变化时，优先级仍然非常重要。

如果在这一关键时刻项目经理失去了勇气并且开始向需求妥协，他实际上将项目再次推进了调度危险！此刻，他或她必须继续保持冷静并且不向新的需求屈服。此时即使用一新需求替换已有的部分也会增加风险。没有警惕心，可能会在快要成功的时候，遭遇失败。

将软件放在第一位

在瀑布方法中，许多重点放在了“规范”（即，问题空间描述）和使它们正确、完整、精良和关闭上面。在迭代流程中，您开发的软件首先出现。软件体系结构（即，解决空间描述）需要驱动及早地决定生命周期。客户不会买规范，当并行演进规范和软件时，软件产品才是关注的主要焦点。对“软件第一”的关注对各团队有一些影响：例如，测试人员以前常在收到完整稳定的规范和大量预先注意的情况才开始测试，而在迭代开发中，他们必须在仍然演进规范和需求的情况就立即开始工作。

陷阱：过度关注于管理工件

一些经理说：“我是项目经理，所以我应该着重于尽可能获得最好的管理工件集，它们对于任何东西都很重要。”不完全正确！尽管好的管理很关键，但项目经理最终必须确保最终产品是所能生产出的最好产品。项目经理不能通过展示他可能拥有的最好管理（事实上他失败了）来掩盖他自己的过错。类似地，您可能注重于开发可能是最好的规范，因为您在过去受过糟糕的需求管理的危害。如果相应的产品缺陷多、速度慢、不稳定或是太脆弱，这都没有用。

及早解决难题

在瀑布方法中，很多难题、有风险的和一些实际未知的事情都放在计划流程中，作为一些糟糕系统集成活动的解决方案。这使得项目的上半部分相对轻松，问题通过书面方式解决了，而不用涉及到许多项目干系人（测试员等）、硬件平台、真正用户或者真实环境。然后项目的集成突然出问题了，所有东西都变得不可靠了。在迭代开发中，计划是建立在风险和未知事物之上的，所以万事开头难。困难、关键、通常是低等级的技术问题应该马上解决，而不是将其推出去或拖到以后的某个时间再解决。总而言之，就如某人曾和我说过的：在迭代开发中，您的谎言迟早会被揭穿（对您自己或是对世界来说）。注定失败的软件项目应该在迭代方法中尽早地失败。

就好比有一大学课程，老师在上半学期讲相对基础的概念，这就让我们感到这课很容易，学生花最小的努力就能在期中获得好分数。

突然，当学期临近结束时，课程开始加速。老师在期末考试前，一笔带过了所有的难题。这时，大多数学生通常会为考试压力所困，在期末考试中表现得不尽如人意。令人吃惊的是其他聪明的老师对此每年每班都发生的灾难感到困惑。较聪明的做法是将课程重点挪到上半学期，在期中之前解决 60% 的内容，包括那些难以理解的内容。管理迭代项目与该例子的联系就在于不要在开始的时候浪费宝贵的时间来解决不是问题的问题或是完成一些琐碎的任务。技术在一开始就失败的常见原因是：“他们把所有时间都花在了解决简单的事情上。”

陷阱：回避问题

我们常会说：“这是一个细致的问题，我们要花许多时间来考虑。我们等一段时间再作出解决方案，这会给我们更多的考虑时间。”随后项目开始处理所有简单的任务，而从不花精力在难题上。当它到了一定需要解决方案的时候，我们草率地做出了决定和解决方案，使得项目偏离了正题。您想要做的恰巧相反：立刻处理难题。我有时候说：“如果一个项目注定要出错，那么在我们花去所有时间和金钱之前尽早地让它出错。”

陷阱：忘记新风险

您在先启阶段进行了风险分析然后用它来进行计划，但是在项目开发中却忘记了风险。它们在以后又回过头来危害您。风险必须每周（如果不是每月的话）进行不断的重新评估。您原来所制定的风险列表只是尝试性的。只有当团队开始做实质性的开发时（首先是软件），他们才会发现很多其他风险。

由于不同生命周期模型而冲突

迭代项目的经理常会看到他的环境和其他组（比如高层管理、客户和承包商）之间的矛盾，这些人还没有接受或者明白迭代开发的本质。他们期望在关键的里程碑处有完成且冻结的工件；他们不想在小型安装中复审需求，他们对于返工会感到震惊，且他们不明白一些丑陋的体系结构原型的用途和价值。他们一边靠毫无目的的摸索来理解迭代，一边在玩弄科技，在规范确定之前开发代码，并且测试无用的代码。

至少，使目的和计划清晰可见。如果迭代方法只在您的脑子里或在团队共用的几个白板上，那么您以后会遇到麻烦。

项目经理必须保护团队不受外部攻击和政策影响，以防止团队受外部世界的干扰和阻碍。他或她必须担当起缓冲的角色。为了成为掌舵人，项目经理必须和外部团体建立信任和诚信。因此，可见性和“跟踪计划”仍然很重要，尤其对于在许多人眼里不太平常的计划来说。事实上，它更为重要。

陷阱：不同组进行他们各自的调度

让所有的组（团队或次承包商）在同一阶段和迭代计划中进行操作会比较好，也更为简单。项目经理常会看到每个独立团队花时间在优化调度，这些优化以他们各自的迭代调度结束。当此事发生时，所有能看到的优势会在以后丢失，且团队不得不和较慢的一组保持同步。尽可能保持可行，把所有人的力量集中在一起。

陷阱：先启阶段用修正过的价格进行竞标

许多项目用来在很早的时候竞标合同开发，这是在先启阶段的中期。在迭代开发中，对于各方来说，做这种竞标的及时最佳时间是在 LCA 里程碑（精化结束时）。此处没有什么神奇的解决办法：它需要项目干系人之间进行协商和培训来展示迭代开发的好处和最终的两步竞标流程。

计算进度的方法不同

传统的求值系统计算进度的方法是不同的，这是自工件不再完成和冻结，但若干次增量返工以来开始的。如果在一个工件在求值系统中有某个值，且您因为您所创建的第一次迭代而受到好评，那么您进度的评估会相当乐观。如果您仅在两到三个迭代以后它稳定了而得到好评，那么您的进度评估会变得相当悲观。因此，当使用此方法来监视进度时，必须将工件分解成许多块；例如，初始文档（40%），第一修订版（25%），第二修订版（20%），最终文档（15%）。每块都必须分配一个值。然后，您无需完成每个元素就能使用求值系统。

另一个方法是为迭代本身组织求得的值，并且从评估标准中衡量价值。然后会根据迭代计划建立“状态评估”中报告的中间跟踪点（一般每月）。这需要比传统需求规范、设计规范等更详细地跟踪工件，因为您在跟踪各种用例和测试用例等的完成情况。

就如 Walker Royce 所说：“项目经理应该更注重测量和监视变化：如需求、设计、代码的变化，而不是计算文本的页数和代码的行数。”（请参阅下文的『更多读物』）Joe Marasco 补充说道：“不仅要小心变化，还要小心不稳定因素。变化多次的事物又回到同一起点，是问题更加严重的症状。”

从积极的角度来说，明智的项目经理使具体的软件早点运行以更早地获得可信度积点。它能以更能理解的方式来展示进度，而不是用充满几百个复选框，经过复审然后完成的书面形式来展示。同样，工程师倾向于“展示它是如何工作的”来“记录它该如何工作”。先展示，后记录。

确定迭代的次数、持续时间和内容

我们先要做什么？刚接触迭代开发的经理常会在确定迭代内容地犯难。开始时，该计划是由风险、技术和程序，以及在建系统的功能和特征的关键因素所驱动。（RUP 提供了确定迭代次数和持续时间的指导。）在整个生命周期都包含这一条件在构造中，计划用于完成某个功能部件或某个子系统，在转换中，它用于修复问题并且提高健壮度和性能。

陷阱：在第一次迭代中处理太多事情

我们讨论了不要先处理难题。另一方面，在相反方向走得太远也会导致失败。现在有一种趋势，就是在第一次或者前几次迭代中处理太多的问题或者覆盖太多的方面。这没能看到其他因素：需要组成（培训）一个团队、需要学习新技术并且需要获得新的工具。通常，问题领域对许多开发者来说都是新的。这常导致第一次迭代速度太快，从而使整个迭代方法没什么作用。或者，迭代异常终止（在什么都没运行的时候宣布已经完成了），这基本上等于在什么经验教训都没获得时就宣布“胜利”，这会失去迭代开发的大部分优势。

当您疑惑或者面临危机时，想办法减少它们的影响（这可适用于问题、解决方案和团队）。记住完整是生命周期的后期所需考虑的问题。“适当的不完整”应该是项目经理在生命周期早期所考虑的。如果第一次迭代包含很多目标，那么将它分为两次迭代，然后客观冷静地设定优先级，确定先尝试完成哪个目标。

在项目早期，先瞄准较为简单和保守的目标更好。注意我们没说容易。在进度早期获得一个坚实的结果有助于提升士气。许多错过第一次里程碑的项目永远无法恢复。大多数错过里程碑很多的项目注定失败，不管您随后做出多大努力。通过计划确保您不会错过早期的里程碑很多。

陷阱：太多迭代

首先，一个项目不应该混淆每日或每周构建迭代。因为在计划、监视和评估迭代上有固定的开销，一个不熟悉此方法的组织不应该在其第一个项目上尝试太多迭代。迭代的持续时间也应该考虑组织的大小、地理分布程度和所包含不同组织的数量。请再次访问我们的“6 加或减 3”主要规则。

陷阱：重叠迭代

另一个常见的陷阱是使迭代重叠太多。在当前迭代的大约最后五分之一处开始计划下一次迭代，同时尝试重要的活动重叠（即，在完成当前迭代并吸取经验之前，开始详细分析、设计下一次迭代和为其编码）会比较有吸引力，此时盯着GANTT 图表看会导致一些问题。一些人不会答应跟随并完成他们自己对当前迭代的贡献；他们不会负责修复事情；或者他们只会在下次迭代时才决定考虑一些或者所有反馈。一些软件部件还没有准备好来支持已经推进的工作等。尽管可以转移一些人力来执行与当前迭代不相关的工作，但应该尽量少用，或者偶尔为之。一些组织成员由于缺乏一定技术或是严格的组织常会触发这一问题：Joe 是一个分析家，这是他唯一能做或是想做的事情，他不想参与设计、实现和测试。另外一个负面例子：一个大型的命令和控制项目使用了过多重叠的迭代以至于它们基本上全部在同一时间点上平行运行，这需要在迭代期间将全部人员分成几部分，而且不希望从前一个迭代的反馈中吸取经验来用于后一个迭代。

请参阅图 3 来了解一些没有生产力的迭代模式。

图 3：一些危险的迭代模式

好的项目经理和好的架构设计师

要获得成功，一个软件项目需要有好的项目经理和好的架构设计师。如果没有一个好的体系结构，可能的最好管理和迭代开发无法生产出成功的产品。相反，如果项目没有管理好，那么一个好的体系结构也会失败。因此就有这样一个平衡，仅仅注重于项目管理不会带来成功。项目经理不能简单地忽略体系结构：它需要体系结构专家和领域专家来确定 20% 应该进入早期迭代的事宜。

陷阱：让同一个人来担当项目经理和架构设计师

让同一个人来担当项目经理和架构设计师只适合小项目（5-10 人）。对于更大的工作，让同一个人担当项目经理和架构设计师常会使项目要么无法正确管理，要么体系结构不理想。首先，角色需要不同的技能组合。其次，就角色本身来说，它远不止于一个全职工作。因此，项目经理和架构设计师必须每天进行协调，彼此交流，而且

协商。这两个角色有点类似于电影导演和电影制作人。大家都朝同一个目标努力，但是却负责具体工作中完全不同的方面。当一个人身兼两职时，项目很少会成功。

结论

在本页，您可能感到泄气：要面对太多问题，要陷入太多陷阱。如果计划和执行迭代开发如此之难，那又为什么要烦心呢？幸好，我们有方法和技术来系统地处理这些问题，回报会比完成质量更高的可靠软件产品所遇到的麻烦更大。一些关键主题："主动应战风险，否则风险会来攻击您。"（摘自 Gilb 的著作，在『参考资料和更多读物』中列出。软件首先出现。承认碎片和返工。选择一位项目经理和一位架构设计师来合作。探寻迭代开发的好处。

瀑布式开发对于项目经理而言很容易，但对于工程团队而言则很困难。迭代开发与软件工程的工作方式更一致，但在管理方面更为复杂一些。假设大多数团队中，工程师和经理的比例是 5:1（或者更高），这是一个很好的权衡。

尽管迭代开发比您第一次使用的传统方法更困难，但有真正的长期回报。一旦您了解了如何很好地进行迭代开发，您将发现自己成为了一个能力更强的项目经理，并发现管理更大、更复杂的项目将越来越容易。一旦您使整个团队都了解迭代开发并以迭代方式思考问题，那么这种方法将比传统的方法要好得多。

注意：John Smith、Dean Leffingwell、Joe Marasco 和 Walker Royce通过与我分享他们在迭代项目管理中的经验，帮助我完成了此文。本文中的部分内容来自我的同事 Gerhard Versteegen 有关软件开发的新书（请参阅下文的『参考资料和更多读物』）。

关于作者

Philippe Kruchten 在 1987 年加入 Rational Software，现在是 Rational（基地在 Vancouver，B.C.）的一员。担任业务单元流程董事和总经理时，他领导了 Rational Unified Process 的开发。除了专注于软件体系结构和设计，他还热衷于软件工程实践和开发流程。他从法国学院毕业，获得了机械工程学士和计算机科学的博士学位。

参考资料和更多读物

1. *Rational Unified Process 2000*，Rational Software，库珀蒂诺，加利福尼亚，2000 年。

2. Barry W. Boehm，"A Spiral Model of Software Development and Enhancement," *Computer*，1988 年 5 月，IEEE，第 61-72 页。

3. Tom Gilb，*Principles of Software Engineering Management*，Addison-Wesley，1988 年。

4. Philippe Kruchten，*The Rational Unified Process—An Introduction*，Addison Wesley Longman，1999 年。

5. Walker Royce，*Software Project Management—A Unified Approach*，Addison Wesley Longman，1999 年。

6. Gerhard Versteegen，*Projektmanagement mit dem Rational Unified Process*，Springer-Velag，柏林，2000 年。

Rational
the software development company

两家总部：

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

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

免费电话：(800) 728-1212
电子邮件：info@rational.com
Web：www.rational.com
全球网址：www.rational.com/worldwide