



AGILE THINKING

敏捷思维

移动互联网和大数据时代
IT企业转型、升级与再造之道

谢明志◎著

IBM全球官方发言人首次著书，教你大象跳舞的秘密
教科书、工具书、改变命运的书
十年磨一剑，直插利润心脏



机械工业出版社
China Machine Press

Table of Contents

1. 敏捷思维：移动互联网和大数据时代IT企业转型、升级与再造之道 0
2. 序言一 1
3. 序言二 2
4. 序言三 3
5. 序言四 4
6. 序言五 5
7. 前言 6
8. 第1章 敏捷思潮 7
 1. 1.1 变革悄然而至 7.1
 2. 1.2 最简单的敏捷交付 7.2
 3. 1.3 百家争鸣 7.3
 4. 1.4 Scrum 7.4
 5. 1.5 敏捷旅行 7.5
 6. 1.6 敏捷实施报告 7.6
 7. 1.7 案例分析：IBM的敏捷转型之路 7.7
9. 第2章 敏捷落地艰难 8
 1. 2.1 敏捷实践误区 8.1
 2. 2.2 理论的局限性 8.2
 3. 2.3 传统团队践行 8.3
 4. 2.4 敏捷实施法则13条 8.4
10. 第3章 大企业的敏捷转型 9
 1. 3.1 重新诠释敏捷宣言 9.1
 2. 3.2 新敏捷扩展模型 9.2
 3. 3.3 持续改进转型模型 9.3
 4. 3.4 案例分析：IBM Rational中国的敏捷转型 9.4
 5. 3.5 故事分享：Rational中国敏捷开发者的一天 9.5
11. 第4章 初创企业的敏捷转型 10
 1. 4.1 初创企业的敏捷思维 10.1
 2. 4.2 以客户为中心 10.2
 3. 4.3 如何解决人事危机 10.3
 4. 4.4 人格与职业 10.4
12. 第5章 敏捷型人才的培养 11
 1. 5.1 如何培养自我管理的秉性 11.1
 2. 5.2 如何培养自知之明的秉性 11.2
 3. 5.3 如何培养自律的秉性 11.3
 4. 5.4 提供上升空间 11.4
 5. 5.5 团队的沟通 11.5
13. 第6章 非凡的测试 12
 1. 6.1 故事分享：从测试工程师成长为敏捷测试工程师 12.1
 2. 6.2 决策影响力 12.2
 3. 6.3 合理投入 12.3
 4. 6.4 “等待”和“浪费” 12.4
 5. 6.5 自动化敏捷测试 12.5
 6. 6.6 故事分享：Rational中国敏捷测试工程师的一天 12.6
 7. 6.7 选择自动化工具 12.7
 8. 6.8 案例分析：改进自动化工具 12.8
14. 附录A 有关敏捷思维、敏捷转型的常见问题和回答 13
15. 附录B 第二代敏捷猜想 14
16. 附录C 敏捷大赛评分规则 15

- 17. [结束语](#) 16
- 18. [参考文献](#) 17

敏捷思维：移动互联网和大数据时代IT企业转型、升级与再造之道

敏捷思维：移动互联网和大数据时代IT企业转型、升级与再造之道



AGILE THINKING

敏捷思维

移动互联网和大数据时代
IT企业转型、升级与再造之道

谢明志◎著

IBM全球官方发言人首次著书，教你大象跳舞的秘密
教科书、工具书、改变命运的书
十年磨一剑，直插利润心脏

机械工业出版社
China Machine Press



序言一

序言一

我作为最早加入摩托罗拉中国公司研发中心的人员之一，在IT通信行业工作了超过20年。和明志的经历相似，我早期主要从事测试方面的工作，而后又参与了开发、系统集成等工作，亲身经历了开发中心从最初的十几个人发展到上千名工程师的辉煌。我在10年前离开摩托罗拉加入NEC后，又开始侧重于市场、销售方面的工作，直至负责公司的全面运营管理。我看到一般大型企业尤其是外企的产品研发和市场运营都采用按部就班的模式，虽然有其严谨的一面，但有时不免效率不高，很可能由于一个环节的拖延造成整个项目的延迟。

明志书中的敏捷研发和敏捷管理恰恰另辟蹊径，开拓了另一片天空。在摩托罗拉等外企衰落的同时，华为等一批国内企业快速成长，成为市场上的领军企业。众所周知，华为聘请IBM为咨询顾问，任正非总裁曾经说IBM是最好的老师，明志也参与了其中的项目。华为把IBM的敏捷研发和敏捷管理灵活应用于软件研发、项目管理甚至客户服务流程中，取得了巨大的成功。

在书中，作者不仅对敏捷方法进行了系统和详尽的描述，还论述了如何在大型企业和创业型企业中应用敏捷，如何避免容易出现的问题和误区，结合大量的案例和切身经历深入浅出地讲解，以便读者理解和掌握。此外，书中也融入了许多关于管理和人才的讨论，例如人才培养和潜力发掘的相关内容，使方法、工具与人得以有机结合。

随着微信的普及和碎片化时间的利用，很多人放弃了系统的读书习惯。2012年，几位资深创业人秋游西山，有感于当前知识信息碎片化、社会功利浮躁、精神交流匮乏以及人际信任缺失，组织了学习互助型群体，现已形成以创业家为主体，同时包含金融投资家、企业高管、资深媒体人、专家学者在内超过一千人的西山读书会。明志和我皆是西山读书会的成员之一，每天读书也已成为习惯。本书的出版面世，从管理学的角度带来了新的思考，希望书中的内容能够给广大读者以启迪。

丁伟

NEC通信（中国）有限公司常务副总裁

序言二

序言二

武林至尊，宝刀屠龙，号令天下，莫敢不从，倚天不出，谁与争锋！21世纪初的科技界，一如武侠场，英才辈出，新式武功不断显现江湖，而在一场场移动互联网新技术的激烈逐鹿中，也诞生了一些名满江湖的高手，他们的传奇经历，一如倚天屠龙传记，不禁让人拍案叫绝。

2009年的一天，偶然在网络上搜索“敏捷测试”，排名第一的《敏捷测试的最佳实践》赫然映入眼帘，作者正是IBM资深敏捷专家谢明志。虽然素昧平生，但明志热情地接受了我们的咨询，并无私地利用个人时间多次接待了我们的登门造访和求教。明志身上IBM国际化的专业经验打动了我，我在中兴通讯所肩负的民族企业使命感和创新学习精神也打动了明志，这就是缘分。由于我个人在工作上向来讨厌官僚、追求高效，明志甚至帮我们在IBM与中兴通讯两个大公司复杂漫长的流程中找到捷径，灵活而“敏捷”地为我们打造了一次量身定制的敏捷测试咨询服务，让我们得以快速揭开敏捷的神秘面纱，见到倚天剑、屠龙刀的真容，这让我感动至今。

与明志认识的那一年，也正是中兴通讯手机测试团队试水敏捷的元年，经过与IBM的合作，我们收获了很多有价值的思想和方法，不仅有效促进了当时我们与北美顶级客户Verizon的一个项目进程，而且经过多年的学习、思考、交流与沉淀，中兴形成了一套独特的敏捷思路和方法，这对于后来整个中兴手机测试中心的组织转型和专业化发展都有非常大的帮助。但其实我更希望强调的是，一种思维、一种方法，无论以何种方式被应用，其结果如何都并不重要，重要的是领悟和知己。高手过招，有时求的也许不是胜负，金庸用“华山论剑”而不是“华山决战”、“华山比武”来形容南宋那一场武林“峰会”，高手自能领会其中真意。

得客户者得天下。敏捷是软件行业从传统模式走出来的一次伟大变革，除了传统软件业之外，敏捷天然地与互联网行业结合，造就了这个伟大的时代。敏捷模式的快速迭代和拥抱变化让产品、交付及其演进过程焕然一新，最好地契合了各类客户需求，在这个模式下工程师们的观念和思维开始不断进化，成为软件业具备互联网思维的先行者。我认为这是敏捷超越行业的一个重要贡献和意义，它让人们思考如何更好地融入这个时代，迎接变化，顺势而为。

开放思维、乐于分享、重视朋友，通过近年来明志领队的多次IBM workshop和交流，我的团队不仅收获了先进、专业、地道的敏捷咨询服务，造就了中兴通讯终端测试技术在业界的领先地位，更从明志身上看到了从为人到做事的诸多闪光点，并以为同业挚友相交多年。明志是有真才实学又乐于分享的人，其思想最能够不断成熟、完善，其实践最能够带来启发，也更接地气，相信此书能够为读者提供一份精美的敏捷大餐！

谢伟

中兴通讯终端事业部副总裁

中兴通讯终端事业部产品测试中心主任

序言三

序言三

本书作者谢明志以敏捷开发正规军的角度与我们分享他在敏捷开发管理中的实战经验。虽然讲述敏捷开发理论与案例的书籍绝对不缺，但是本书从更多维度来剖析敏捷开发的真谛。

我多年前刚回国的时候，就听说敏捷开发在国内已经流行了不短的时间，大部分IT企业也都说自己的研发流程是敏捷的。可惜当我有机会深入了解后却发现，大家除了能说出“快速迭代”、“响应变化”、“小步快跑”等关键词，绝大部分的研发团队其实并没有真正理解敏捷开发的核心思想，结果敏捷开发成为开发不写文档、需求无序进入开发环节、架构不断被推倒重来等恶果的最有力借口，用户也就不可能真正享受到敏捷开发所应该带来的用户体验不断递增的快感。其实，我们还能从实践误区中更好地理解为什么很多敏捷开发团队的产出并没有达到预期的高度。从Rational产品的发展历程中，我们也能体会到敏捷开发并非初创团队才能使用的专利，像IBM这样的大型企业也能完成从瀑布式到敏捷式的华丽转身，让敏捷开发成为基石，以应对万变的市场需求。

本书还用大量篇幅讲述了敏捷型人才的培养方法，详细地把优秀人员的素质罗列在我们的眼前，在我看来，这才是敏捷开发成败之间最大的分水岭。敏捷开发的确需要大量的前置培训来统一关键人员的思想，而这些培训并非单单提供理论上、流程上与技术上的知识，更重要的是提高大家的自我管理、自律与协作能力。更理想的状态是，团队不仅仅理解规划需求，而且对目标用户和市场有业务上的理解，使得呈现出来的产品能真正触动用户的内需，从而让用户真心实意地成为铁杆粉丝。

以前市场存在着各种经济红利，IT企业并不需要精益求精就能活下来，而且还能活得不错。可是温水煮青蛙还是终有煮熟的一天，本书不仅对国内的开发环境提出了有力的批判，同时也提供了可借鉴的转型之路。但愿我们能与作者共勉，为更多企业踏上康庄的敏捷开发之路做出贡献。

余康柱

资深IT/互联网企业管理人

前易宝支付运营副总裁

序言四

序言四

当明志请我为他的新书写序时，我毫不犹豫地答应了。因为这本书将为中国的IT行业带来非常正面的影响。

2006年，IBM软件部面临了很多市场和经济方面的挑战，那个时候的IBM软件部已经是独当一面的部门，每年的收益是200多亿美元，而每年并购进来的大小公司有几十家，在全球89个地区有27000多名开发人员，但也是由于这个原因，软件部内部产生了协作和创新的问题。要解决这些问题，并且实现收益的持续高速增长，IBM软件部肯定要转型。

IBM软件部确定将当时行业中非常流行且得到认可的敏捷方案作为转型的模式。当然，IBM Rational亦是这个领域的专家，它在软件和系统工程市场上累积了超过30年的经验，转型将建立在Rational Unified Process (RUP) 的坚实基础上，与之相辅相成。

2006~2013年期间，IBM软件部在敏捷开发模式下创造了有目共睹的成果。我们的Release Cycle从2006年的一年一次大大缩短至一季度一次，功能上的提交一点都没有减少，而且质量继续维持在一贯高的水平。

过去几年，IBM Rational致力于推动敏捷开发的方法论和管理，希望把我们的经验分享给客户和社区。很多大中型的企业对敏捷开发都非常感兴趣，特别是怎样将敏捷开发从理论上落地到大企业和大团队的开发实践中。过去几年，IT行业的不断变化和新技术的不断涌现，如云计算、移动互联网、大数据，都迫使我们的很多传统客户变得更敏捷。

但每次我们探讨敏捷话题的时候，结论都是：今天企业需要的不单是开发的敏捷，更是企业的敏捷。这就带出了另外一个课题——DevOps。

就如敏捷开发一样，DevOps过去几年也是IT行业里备受关注的方法论。DevOps本身当然是针对Development和Operations之间的协作矛盾。但是，如果引入敏捷开发，矛盾将会进一步加深，因为敏捷的一个重点是持续交付，即多版本多交付，这势必会减少大版本带来的风险和不灵活性。

Development和Operations中间是一个关卡，要达到企业敏捷，我们要过的关卡还有很多。所谓企业，是端到端的，由企业愿景开始，变成业务计划和需求，变成IT项目和需求，经过分析、设计、编码和不同环境的测试，如单元测试、性能测试、SIT、UAT，再交给Operations团队进行最后的生产环境上线测试，等等。每一个关卡都会影响整个流程的进度，当然，最终会影响我们满足业务的需求。所以，IBM Rational认为DevOps的定义应该是涵盖整个端到端的企业流程，只有端到端的DevOps才能做到企业敏捷。

本书通过深入浅出的讲解，把敏捷开发从传统的小规模敏捷伸展到跨地域、跨时空的DAD方法，是读者了解敏捷的必备参考书。

谢毓明

IBM大中华区Rational CTO

序言五

序言五

自从1946年第一台计算机诞生以来，人类一直在寻找开发软件的最佳方法，以提高软件质量，减少项目成本，并更好地控制项目进度。最早的软件开发模型是1970年W.Royce提出的瀑布模型（Waterfall Model），这种方法在20世纪80年代成为软件开发的主流。瀑布模型的概念是按照需求、设计、开发、测试的顺序，线性地开发出软件产品。在前一个阶段完成之前，不能进入下一个阶段。这种方法对需求明确、设计方法成熟且开发人员对编程工具非常熟悉的软件项目十分适用，但是对已经完成的工作进行修改却很麻烦。

为克服瀑布模型“死板”的缺点，在20世纪90年代出现了迭代和增量开发模型（Iterative and Incremental Development Model），其核心思想是每一次“需求-设计-开发-测试”的迭代只开发整个软件的一部分，这样做的目的是尽早发现并解决项目中存在的重大问题。在迭代和增量开发模型的基础上，也出现了一些更加“短小精悍”和更加“以人为本”的开发模型，比如RUP（1994）、Scrum（1995）、Extreme编程（1996）等。这些方法更加注重持续回顾和增量开发，并通过逐步完善的方法来开发和交付软件产品。

2001年2月17日，17位“草根”程序员在美国犹他州发布了敏捷宣言（Agile Manifesto），其中最重要的4项原则是：“个人和交流重于过程和工具，能运行的软件重于详尽的文档，与客户合作重于合同谈判，响应变化重于执行计划”。敏捷宣言的发布，在软件开发方法领域开启了一场轰轰烈烈的敏捷运动。所谓敏捷方法，其实只是一个概念，它是所有带有敏捷特征的软件开发方法的统称。上面提到的Scrum和Extreme编程都属于敏捷方法。

在过去的十几年里，随着网络的普及，以及近年来越来越多移动应用的出现，信息的传播速度越来越快，软件开发的周期也越来越短。很多项目启动时需求并不是很明确，传统的瀑布模型不再适用，敏捷方法的优点得以发挥。因此，敏捷方法越来越流行，特别是针对小型项目。在敏捷方法里，使用最多的是Scrum方法。据敏捷联盟（Agile Alliance）统计，在全球使用敏捷方法的企业里，66%使用过Scrum方法或Scrum方法的变种。

提到软件开发，我们不能不提到CMMI模型。有人认为CMMI和敏捷方法是相冲突的，二者不能共存，其实这是一种误解。CMMI规定了必须做什么，但是没有规定怎么做。在选择项目的生命周期模型时，CMMI项目可以选择瀑布模型，也可以选择敏捷模型。敏捷方法的出现，实际上是丰富了软件开发模型的可选项，是对CMMI的补充。敏捷方法合理地借鉴了CMMI的400多条最佳实践，可以最大限度地保证敏捷项目的成功。所以CMMI和敏捷方法是相辅相成的关系，不是对立的关系。写这篇序的时候，我正在美国华盛顿参加CMMI研究所举办的北美SEPG年会，今年年会的一个特点就是出现了大量CMMI与敏捷结合的论文和报告。可以预见，在未来的几年，CMMI与敏捷方法的结合将会是一个重要的趋势。

在敏捷方法不断发展的过程中，IBM扮演了一个非常关键的角色。从最早的RUP方法到DAD，再到最新的DevOps方法，IBM都走在时代的最前沿。本书作者非常幸运地亲身参与到这场轰轰烈烈的敏捷运动中，更为可贵的是，作者把自己在IBM十年的亲身体验写下来，与广大读者分享，为大家更好地理解敏捷方法，避免已经走过的弯路，提供了第一手的资料，非常有意义。

最后祝愿敏捷方法如作者所期望的那样，能在中国的企业里开花结果！也祝愿中国的软件产业能早一天走向世界！

高山

CMMI评估师，CSM

前言

前言

这是一部讲述管理思维变革的书，一部用失败教训、成功体验和行业前沿的系统管理思维打造的创新之书。它将管理学与心理学融为一体，告诉你如何将敏捷思维用于商业、工作和生活，破译初创团队成长为千人企业的平稳转型密码。

什么是敏捷方法？作为IBM敏捷官方发言人，无数人问过我这个问题。敏捷方法是一种让企业最大程度降低成本、增加利润的方法，一种通过增量互动对客户的需求和反馈保持开放的产品或服务。它不是出于恐惧项目超期、成本超支、团队离散而制定出的一些监测和控制项目的方法，而是赋予团队信任和自治、全力支持团队以满足客户需要、最大化交付价值的方法。迄今为止，我已经以敏捷方法为主题做了近8年的讲座，工作成绩得以飞速提升的学员不计其数。他们不仅跨越了知识和经验的门槛，运用“新式武器”打击顽固的症结，而且窥破了敏捷方法的奥秘，在工作中变得“放下”和轻松许多，仿佛不知不觉中，封存起来的内在力量冲破禁锢释放出来。他们能够全神贯注于自己感兴趣的工作，将精力和时间用在对人生和企业有价值的事情上。他们明白在职场中要做到不复制、不模仿，在团队中要证明自己的差异价值，通过真实的自己来影响和带动团队，成为团队中独具一格的领袖。这就是敏捷思维的有趣之处。敏捷思维是一种非颠覆性的创新，以“实现最小代价换取最大回报”为价值度量，在“简化工作流程+合理运用工具+智慧管理人心”的平台上实现降低产品交付成本且全心成就客户。

从2006年开始接触敏捷方法至今，我目睹了很多实践者追求和学习敏捷方法的热情不断熄灭在荒废的半成品中，也不断听说某些敏捷讲师将客户敏捷部署不落地的原因归结于企业领导的不理性、不支持或团队人员的素质低下。于是，我决心写一本书来为敏捷思维正名。事实上，我的敏捷实践也历经了长达8年的探索，尽管我有着得天独厚的成长环境和开明的企业平台，但我所走过的路，包括部署敏捷方法、推行新工具、培养敏捷团队、改良和变革企业流程和模式，每一步都布满荆棘。不过，坚持到今天，相信很多有过相似经历的人一定像我一样，正在品尝努力付出所结出的硕果。但是，至今也没有一部深度探索敏捷思维变革和企业级敏捷转型的书籍出版，于是，2013年春节，我开始了本书的写作。

第1章引出“何为敏捷”的问题，从Scrum方法出发，展示敏捷的价值，介绍敏捷方法的基本构成。同时，该章还将以IBM公司为案例，解析IBM作为一家有先进管理理念的大型企业，在面临外来挑战和内部革新的内驱力时，为何选择敏捷作为自身变革的策略，展示企业敏捷转型中可能面临的挑战和光明前景。

第2章通过一些典型案例，分析敏捷转型困境，总结敏捷实践误区，指出Scrum等核心敏捷方法的局限性，特别是与国内企业相结合时所暴露出来的水土不服问题。这一章还介绍了项目经理的重要作用，对比了传统项目管理模式和敏捷项目管理模式，向敏捷项目管理转型的企业提出了忠告和建议，并在最后总结了13条“敏捷实施法则”。

但是，仅仅照搬别人的成功方法还不能保证改革的成功，必须活血化瘀，疏通承载方法的实体，即“人”与“工具”的任督二脉，使得敏捷转型平滑、系统地进行。根据“人”的职业养成阶段、“工具和技术”的成熟度、“政治、架构、企业文化”的环境差异，“流程”也需要适当裁剪。必须让团队、团队负责人、产品负责人看到，风险可预期、转变非颠覆、目标可实现，他们才能有信心并自发地配合企业由下而上的变化。

第3章介绍了敏捷扩展模型（ASM）和规范性敏捷交付（DAD）方法。DAD是Scrum、Lean、Kanban等方法的复合体，本章主要讲解该方法的特色，例如重要角色及其不同于传统敏捷的典型实践。为了清楚地分析DAD方法，与第2章Scrum房地产公司的小故事呼应，本章

讲述了一家DAD房地产公司如何通过自身的优势规避来自政府规划的重大风险，建造楼盘并取得项目成功。本章以IBM Rational中国的转型为例，分享了以“工具+流程+人”为框架的敏捷转型实践。此外，还讲述了传统服装行业中DAD方法的应用，并在最后以“敏捷开发者的一天”的故事来完整描述一个大型组织中敏捷方法的成功落地。

第4章介绍了以敏捷思维开发商业模式的方法，并通过ASM因子的调整得到了适合10人以下初创团队的敏捷方法，同时指出初创团队成功的关键在于建设敏捷团队，以及团队领袖、带头人作为灵魂人物在敏捷转型中的重要性。此外，本章还讨论了敏捷团队建设的四个阶段和防止人才流失的方法，以及人格测评对招聘甄选和岗位匹配的影响。

第5章讲述人才培养中如何使团队自发地做到“自我管理、自我认知、自我控制”，特别是在团队建设、团队协作方面，通过几个小故事模拟“三自”培养方法，手把手地教你如何通过心理学方法精细地打造一个“轻管理”的敏捷团队，分享“沟通和倾听”的方法与技巧，降低团队建设中的“信任”成本。

第6章的主题是“非凡的测试”，这是因为敏捷测试实属不易，有太多经验需要分享。同时，这一章也讲述测试生涯规划，作为DAD敏捷方法的推崇者，详细讲述“独立测试”的必要性，测试的“决策影响力”、“投入产出效益”，“自动化测试”的策略和规划方法，以及“自动化工具”的选择。通过“敏捷测试工程师的一天”的故事分享，帮助测试者在工作中学习如何实现更高的自我价值。

目前，对于研发工作，我已经渐行渐远，而开始深入市场和客户需求、客户关系的梳理工作。但每当与客户们谈及其需求、趋势，我们都会延伸到流程、管理，以及关于持续改进理念的探讨。通过这些交流，我不仅收获了客户的信任，也听到了更多有价值的观点，遂在附录中分享了有关敏捷思维和敏捷转型的常见问题和回答，并大胆推断了第二代敏捷“DevOps”——持续交付软件驱动的创新。此外，还介绍了一个以完整的IBM敏捷方案和敏捷实施体验帮助团队深入了解IBM规范性敏捷方法的敏捷大赛活动，通过在线评分系统展示了一套简单的敏捷成熟度度量体系。

谢明志

2014年9月

第1章 敏捷思潮

第1章 敏捷思潮

- 1.1 变革悄然而至
- 1.2 最简单的敏捷交付
- 1.3 百家争鸣
- 1.4 Scrum
- 1.5 敏捷旅行
- 1.6 敏捷实施报告
- 1.7 案例分析：IBM的敏捷转型之路

1.1 变革悄然而至

1.1 变革悄然而至

新工业时代已经到来，为了迎接这场静悄悄的工业革命，产品交付团队中，从工程师到项目经理都在极力做出自我调整。医药、软件、汽车、集成电路等行业客户对新产品的不断需求以及行业研发成本的不断攀升，标志着从计划性开发方式到自适应式开发方式的重大转变，这个转变使得那些仍然坚守预见性的旧思维和旧规则的工程师、项目经理陷入了大混乱，因为旧思维方式正在迅速退出历史舞台。

2002年中期，加拿大多伦多市的Alias System公司开始开发Sketchbook Pro，这是一个与微软的Tablet PC操作系统几乎同时发布的软件包。起初，它的管理和开发团队并不确定开发计划需要包括哪些细节，但团队成员有一个明确的产品构想和商业计划，对于产品需要具备哪些功能有大体的想法。他们没有积极地参与市场营销，但有一个总体设计架构，并且每两周提交一次新测试完成的功能，然后根据产品测试的实际情况调整未完成的工作计划。最终，在微软发布日期（11月），该产品同步上市，并因其高质量标准在市场上取得了空前成功。

在汽车行业，日本丰田公司的副总裁大野耐一综合了单件生产和批量生产的特点，创造了一种在多品种小批量混合生产条件下高质量、低消耗的生产方式——准时生产（Just In Time, JIT）。JIT生产方式的基本思想是“只在需要的时候，按需要的量，生产所需的产品”，也就是追求一种无库存或库存最小的生产系统。丰田公司始终要求以用户的需求为生产起点，强调物流平衡，追求零库存。生产节拍由人工干预、控制，重在保证生产中的物流平衡，每一道工序均要保证对后道工序供应的准时化。

上述产品开发方法指出了一个非常关键的问题，当研发成本减少到一定程度时，产品开发的整个经济学就会发生变化——以计划性、合规性为基础的开发流程转变为以适应性为基础的流程（构想、探索、适应）。这种演变是一种进化过程，就像生物进化一样，但是它要比自然界的进化迅速得多。生物进化的流程是试验（突变和再结合）、探索（适者生存）和改进（产生更多的生存者），而产品开发流程越来越类似这个流程。

开发过程的不确定性增加、进度不断被期望压缩，研发任务将不仅仅局限于工作室内部完成。新的营销理念需要我们从客户参与管理和研发开始，还要以高质量为目标，对于服务精度和速度的要求也不断提高。因此需要创新，需要适应新环境的新研发过程，以更有效地增加产品价值并保持竞争优势。

事实上，大型公司对变革和创新的诉求比小型公司更加强烈，因为大型公司往往在新产品和交付新产品之间存在着更大的差距。新投放的产品失败率高达59%，这个数字从20世纪90年代早期以来几乎没有变化。而且，半途而废或者（在市场上）失败的产品消耗了46%的产品开发资源。虽然少数侥幸的公司还可以挽回败局，但真正成功的公司则越来越多地在实施敏捷交付方法。

2006年的IBM全球报告（见图1-1）指出，鉴于市场和竞争压力，面对多元化的变革，如果不能快速适应，那么企业赖以生存之基本，或者当下各种项目的最终价值可能会受到影响，甚至可能失去商业价值。因此，CEO们认为需要具备在项目进行的过程当中根据需要做出变革的能力。如果项目团队不能够在项目进行过程当中积极地对刺激进行反馈和变革，问题可能会接踵而来。

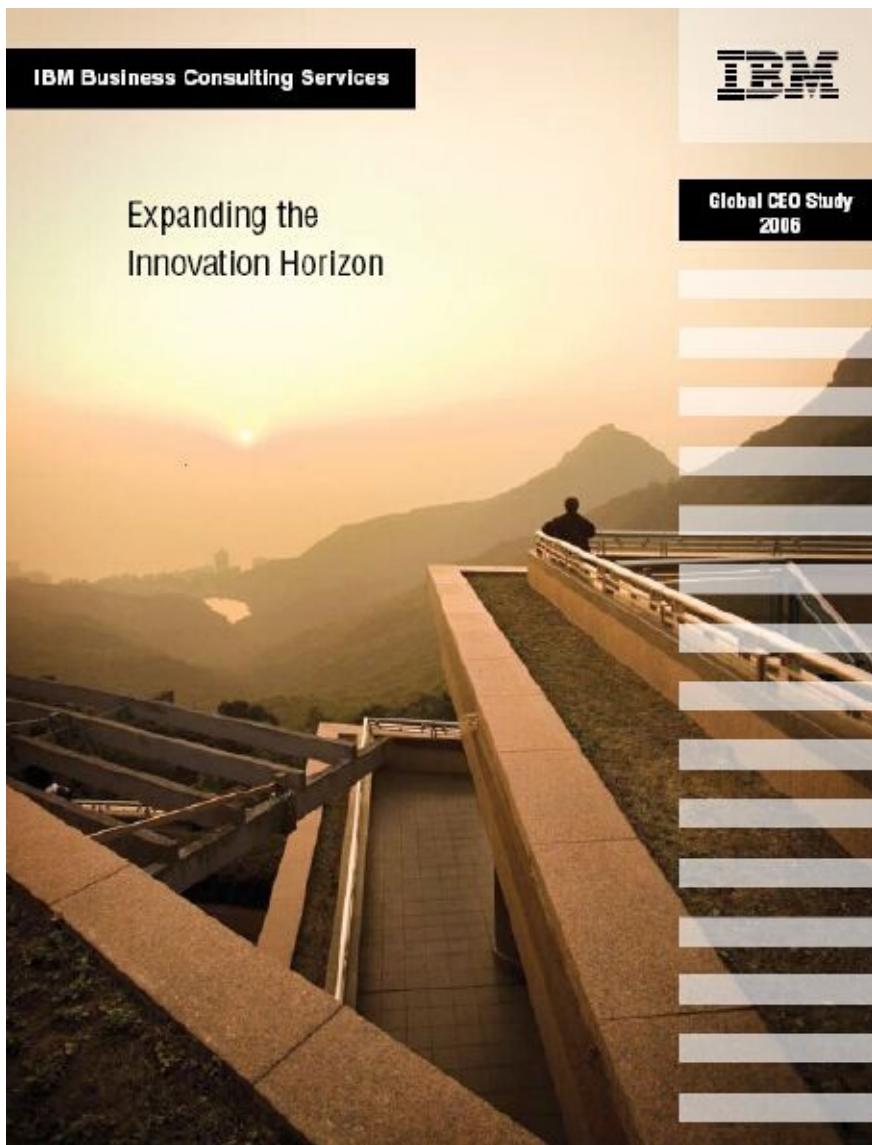


图1-1 2006年3月IBM全球CEO调查报告

CEO们皆希望对项目进行有效的变革管理。这个过程应该包括对变化的识别和判断，对变化的商业价值的判断，对变化会给项目带来的冲击和影响的判断，并将相应的信息提交项目投资人进行评估，最终由项目投资人决定是否将变化引入到项目当中。如果变化被引入，项目投资人还应该考虑变化对项目的影响程度，并且为之配备相应的额外资源，如延长时间和追加资金预算等。2006年，IBM决定在公司推动模块化设计以提升整合能力的设计思路，并且采用简单、迭代的敏捷开发过程作为项目组织和实施的重要决策。

敏捷绝不再局限于研发阶段，而是贯穿软件的生命周期，从产品的抽象概念，到客户渐渐体会到产品的真正使用价值，再到解决方案在组织内获得成功，敏捷的价值无不凸显其中。若用一言以蔽之，敏捷的核心价值是充分地将人的时间、精力、金钱始终集中在最有业务价值的部分，并以风险最小的方式将可用的产品推向市场。

1.2 最简单的敏捷交付

1.2 最简单的敏捷交付

如果用敏捷方法生产钢笔，敏捷规划了四个重要功能组成部分，即笔芯、笔杆、笔帽和雕花，相应功能若投入市场可以获得的价值分别为50、30、15和5。倘若每项功能开发所需的成本一致，时间都是1个月，那么敏捷开发就是要将每个月的开放力量投入到最有价值的部分，并依次类推。那么在没有生产规划的情况下，最开始的迭代应该将资源投入在钢笔笔芯部分，因为这个部分可以带来最高的价值。而第一个迭代完成时，客户体验到了用钢笔书写的刚劲有力、笔锋回转、酣畅淋漓，因此决定了第二次投资，即为钢笔打造第二项重要的功能——笔杆，一个可以方便把握的笔杆，一个可以保护笔芯、防止墨水侧漏的笔杆。当我们在第2个月全力完成此功能后，客户便拥有了第二个发布的功能。现在重新排列未完成的工作，此时客户给出重要的指示，决定放弃在笔帽上的雕花功能而选择一个既可以将钢笔别在衬衣口袋又能够保护鼻尖的笔帽作为第三项最重要的功能。因此，最终我们仅仅用3个月就开发出来价值为 $50+30+15=95$ 的新产品——钢笔。节省下的一个月时间将投入到第二代钢笔的研发中，很快，我们设计出第四项新功能——节省墨水50%。因为每个月都能够从市场上获得产品推出的反响，所以我们将新功能的价值评估为50。也就是说，我们用了4个月的时间，使用敏捷方法生产了145的价值，而且一共发布了4次产品，产品投产30天后，我们就收获了价值为50的回报。

1.3 百家争鸣

1.3 百家争鸣

如果用传统开发方式，也就是瀑布式模型来开发钢笔，将产品的架构划分为数据层、逻辑层、应用层和接口层，团队也划分为相应的4个平行开发团队。每一层的架构和设计都需在代码构造前完美地完成，因而为了实现产品的按期发布，四个团队都不能独立将自己的产出在4个月内的任何时间向外发布，也就无法评估笔杆上的雕花是否比新加入的“节约墨水”功能更吸引用户。所以，最后产出的产品虽然有 $50+30+15+5=100$ 的评估价值，但在这个过程中投资者只得一次性投入4个月的资金，直至4个月后才等到第一次面向市场的机会；而此时同类产品的竞争优势更加可观，产品实际带来的利润价值并没有预期的大。

图1-2给出传统开发过程与敏捷开发过程的对比。

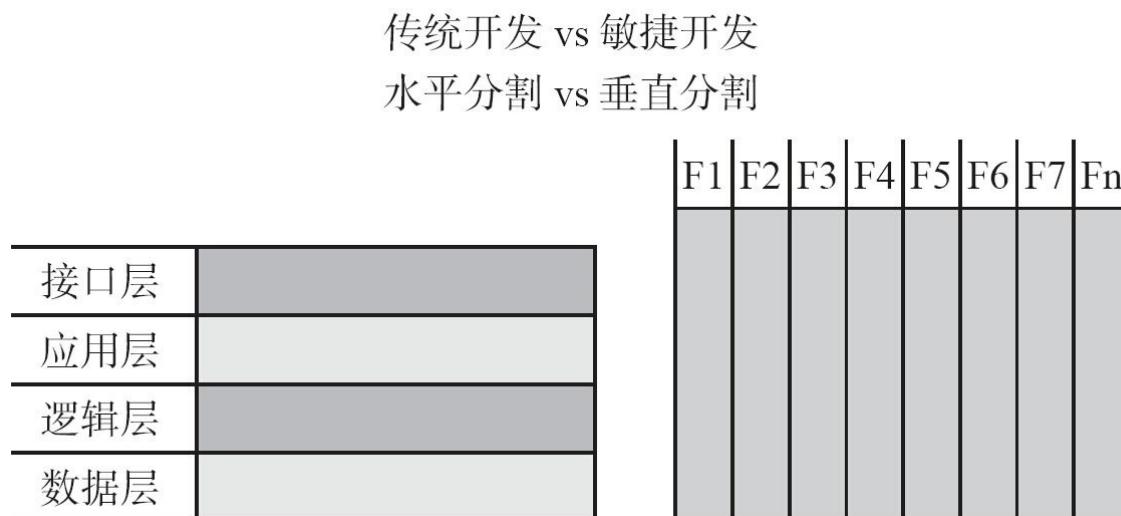


图1-2 传统开发过程与敏捷开发过程对比

在此不过多列举敏捷方法，但值得一提的是核心的敏捷方法Scrum，以及可以与Scrum媲美的敏捷方法论。XP (eXtreme Programming) 更加适合3~5人的小型项目团队，DSDM (Dynamic Systems Development Methodology) 与Scrum一样更加适合大型团队的项目开发，还有Lean等方法。这些方法其实都可以在IBM的某些项目中窥见一斑，后文将具体讨论。

1.3.1 极限编程

极限编程 (XP) 是由Kent Beck在1996年提出的。XP是一种注重协作、快速和早期软件创建的有技巧的开发实践方法，适合10人以下的团队。XP是一个轻量级的、灵巧的软件开发方法，同时也是一个非常严谨和周密的方法。它的基础和价值观是交流、朴素、反馈和勇气，即任何一个软件项目都可以从四个方面入手进行改善：加强交流，从简单做起，寻求反馈，勇于实事求是。XP是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期，通过积极的交流、反馈以及其他一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。

极限编程的基本原则：

- 通过客户、开发团队、项目经理三方共同参加的会议来确定开发计划。

- 小版本发布：尽快发布，尽早发布。
- 通过系统隐喻（metaphor）让每个人了解整个系统。
- 简单设计：为明确的功能进行最优的设计，不考虑未来可能需要的功能。
- 重构（refactoring）：不断优化系统设计，使之保持简单。
- 单元测试：先写测试，后写代码。
- 结对编程（pair programming）：系统的每一行代码都是两个人用一个键盘完成的。
- 代码集体拥有：开发队伍中任何人可以修改任何其他人的代码，代码不属于某个个人。
- 持续集成：至少每天将整个系统集成一次，保持一个能运转的系统。
- 40小时工作制：保证休息，保持体力。
- 现场客户：客户自己也是软件开发队伍的重要一份子。
- 编码标准：必须有统一的编码规范，确保代码的可读性。

关键词 测试驱动，结对编程，持续集成，重构，小版本发布，沟通。

1.3.2 水晶方法

水晶（Crystal）方法论由Alistair Cockburn在20世纪90年代末提出。他把开发看做是一系列的协作游戏，而写文档的目标是帮助团队在下一个游戏中取得胜利。水晶方法的工作产品包括用例、风险列表、迭代计划、核心领域模型，以及记录了一些选择结果的设计注释。水晶方法也为这些产品定义了相应的角色。然而，值得注意的是，这些文档没有模板，描述也可不拘小节，但目标一定要清晰，即满足下次游戏开始的条件。

对于水晶方法论，根据方法论的轻重可以分为透明水晶和橙色水晶等。透明水晶一般适用于轻量级的团队。不管是哪种水晶，都会对团队的角色、团队的工作项和产出、核心实践、支持过程等进行定义。

关键词 协作，角色，文档，迭代，风险管理。

1.3.3 动态系统开发方法

动态系统开发方法（DSDM）倡导以业务为核心，快速而有效地进行系统开发。可以把DSDM看成一种控制框架，其重点在于快速交付并补充如何应用这些控制的指导原则。

DSDM是一整套的方法论，不仅仅包括软件开发内容和实践，也包括了组织结构、项目管理、估算、工具环境、测试、配置管理、风险管理、重用等各个方面的内容。

DSDM的基本观点是，任何事情都不可能一次性圆满完成，应该用20%的时间完成80%的有用功能，以适合商业目的为准。实施的思路是，在时间进度和可用资源预先固定的情况下，力争最大化地满足业务需求（传统方法一般是需求固定，时间和资源可变），交付所需要的系统。对于交付的系统，必须达到足够的稳定程度以在实际环境中运行；对于业务方面的某些紧急需求，也必须能够在短时间内得到满足，并在后续迭代阶段中对功能进行完善。

DSDM的基本原则：

- 活动用户必须参与。

- 必须授权DSDM团队进行决策。
- 注重频繁交付产品。
- 判断产品是否可接受的一个基本标准是符合业务目的。
- 对准确的业务解决方案需要采用循环和增量开发。
- 开发期间的所有更改都是可逆的。
- 基本要求是高层次的并区分优先级（以在低优先级的项目上获得一定的灵活性）。
- 在整个生命周期集成测试。
- 在所有参与者之间采用协作和合作方法。

关键词 以业务为中心，用户参与，迭代，快速交付，团队协作和沟通。

1.3.4 精益生产

精益（Lean）管理的思想起源于丰田公司，旨在创造价值的目标下，通过改良流程不断地消除浪费。这种方法现已被广泛用于生产制造管理，对于IT系统建设，精益开发的常用工具模型是价值流模型，如图1-3所示。

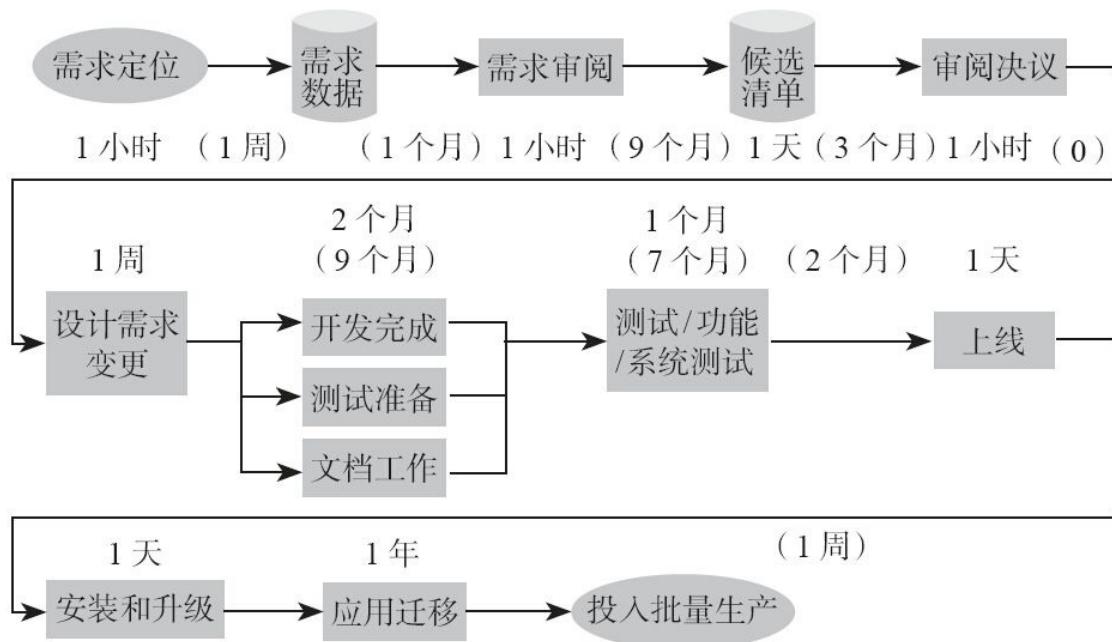


图1-3 精益方法的价值流模型（括号内代表浪费的时间）

精益开发的基本原则：

- 杜绝浪费：将所有的时间花在能够增加客户价值的事情上。
- 推迟决策：根据实际情况保持可选方案的开放性，但时间不能过长。
- 加强学习：使用科学的学习方法。
- 快速交付：当客户索取价值时应立即交付价值。

- 打造精品：使用恰当的方法确保质量。
- 授权团队：让创造增值的员工充分发挥自己的潜力。
- 优化整体：防止以损害整体为代价而优化部分的倾向。

上述方法都是流行的敏捷方法。图1-4是VersionOne在2013年发布的敏捷方法的使用情况。最主流的敏捷方法仍然是Scrum，下面来看看Scrum是什么。

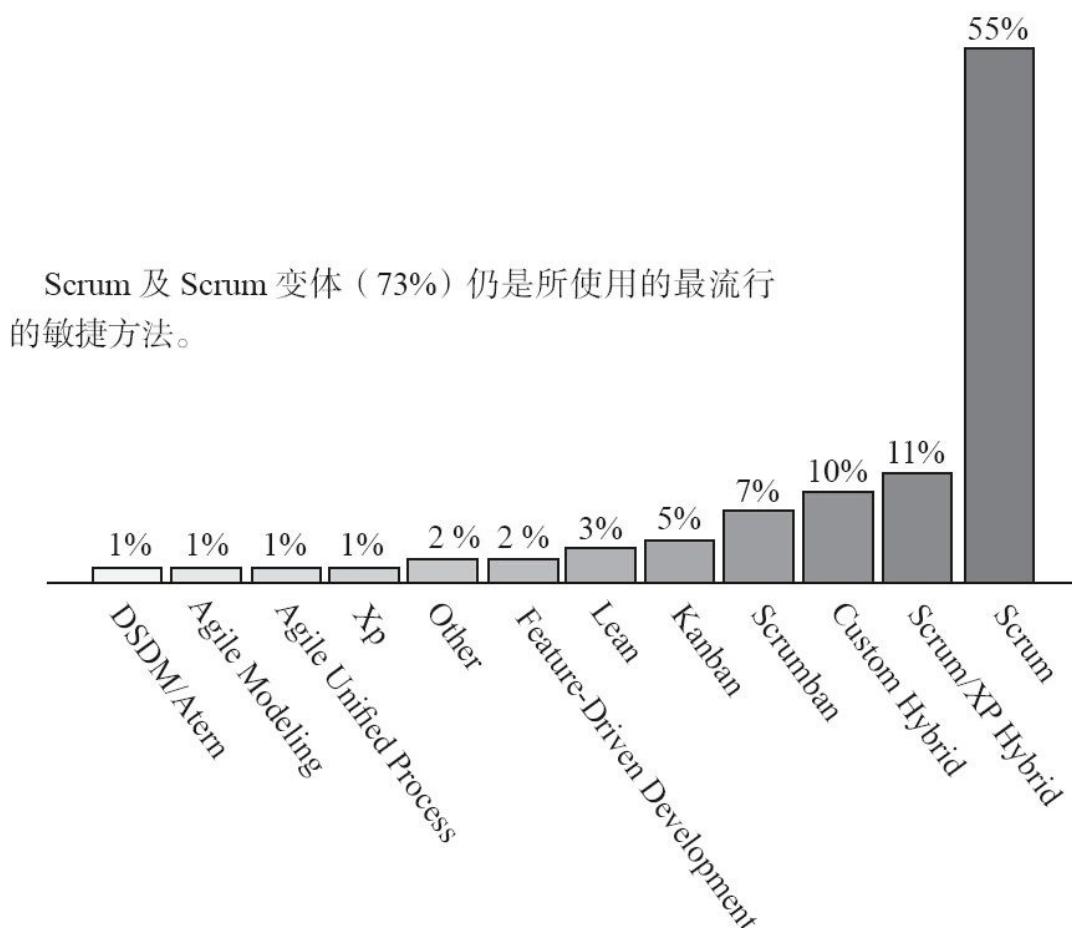


图1-4 敏捷方法的使用情况

1.4 Scrum

1.4 Scrum

1986年1月的《Harvard Business Review》期刊登载了一篇题为《The New New Product Development Game》的文章，其要义说的是：“原有的‘接力跑’式的产品开发模式一定程度上违背了以人为本、最大化生产力、灵活生产方式的原则。相反，另一种独特的团队，采用‘橄榄球’式的团队合作方式——整个团队合作无间，灵活机动地接球、传球，并作为一个整体迅速突破防线——这可能更加适应今天更具挑战的市场需求。”

这就是Scrum源起的故事，从产品研发的角度来理解这个恰到好处的比喻，可以更好地理解决Scrum的精髓：

- Scrum使我们能够专注于如何在最短的时间内实现最有价值的部分。
- Scrum使我们能够快速、经常地监督实际产品的发展状况（每两周或一个月）。
- 按照商业价值的高低先完成高优先级的产品功能，采用自主管理模式，凝结团队智慧创造出最好的方法，因而效率提高。
- 每隔一两周或者一个月，就可以看到实实在在的可以上线的产品。此时，可以进一步决定是继续完善功能实现更多需求或者直接发布。

Ken Schwaber在他的书中指出，Scrum不是一种构建更好产品的方法，也不是快速构建高质量软件的方法。Scrum是一种工具，一种可以帮助我们找到用来快速构建高质量软件的要素的工具。

1.4.1 发源与发展

Scrum之父Jeff Sutherland于1993年在Easel Corp提出了Scrum的方法，而大师Ken Schwaber创作并出版了三本有关Scrum的书：《Agile Project Management with Scrum》、《Agile Software Development with Scrum》、《Enterprise Scrum》，Scrum自此开始坚实地扎根于软件行业。

2002年，Ken Schwaber和Mike Cohn联合创办了Scrum Alliance（Scrum联盟）。在发布后的10年内，Scrum相继被微软、雅虎、谷歌、飞利浦、西门子、IBM、第一美国不动产经纪公司等知名企业采纳，所涉及的领域已经超出了软件研发的范畴，涵盖嵌入式系统、游戏、网站、手机、网络交换设备等众多领域。

1.4.2 自我管理

Scrum方法并没有特定的工程实践惯例，但是定义了几个角色，例如，团队（Team）、团队负责人（Scrum Master）、产品负责人（Product Owner），以及角色相应的职责。团队这个角色被注入了“自我管理”的使命，团队成员会主动朝着产品研发的进展方向规划自己的迭代计划、日计划，并主动完成工作。产品围绕着产品清单（Backlog）展开，这个清单作为一个容器容纳所有待完成和已完成的工作项，而每个工作项都可以继续拆分和重新排列优先级，当Scrum开发开始时，团队将从中选择重要且能够完成的产品清单，在2~4周内完成这些产品清单的发布。产品正是通过一次次的迭代（Sprint），由增量构建的产品清单所组成的。

Scrum同XP、水晶等方法一样均属于敏捷方法范畴，敏捷宣言强调，在项目实施过程中：

个人与交互	重于	开发过程和工具
可用的软件	重于	复杂的文档
寻求客户的合作	重于	对合同的谈判
对变化的响应	重于	遵循固定的计划

1. 拥抱变化 (embrace the change)

无论多么明智、多么正确的决定，也有可能在日后发生改变。因此，团队要充分理解利益关系人（Stakeholder）和客户代表为什么经常提出新的需求，一句话，“唯一不变的是变化”。团队要相信利益关系人做出的每次决定和需求调整都是将产品开发推向更正确的方向，新变化将进一步降低风险，实现团队最大化利益，这是适应市场变化的必然行为。

而在接受变化的同时，我们应该积极地向利益关系人和客户代表反映开发过程中暴露出来的可能的设计缺陷和错误。在实际工作中，团队成员应该用优先级制度来划分事情和目标的先后顺序，在迭代周期内，对于还没有最终决定的设计方案，可以置后实现、测试，不用急于投入资源展开全面的开发、测试活动。这样一来，开发、测试团队也将更加适应，真正拥抱变化。

2. 客户的参与 (with customer representative on site)

首先，谁是客户（Customer），谁又是客户代表（Customer Representative）？利益关系人可以理解为客户、产品的最终使用者（End User）、内部使用者（Insider User）和商业伙伴（Business Partner）。利益关系人作为团队中最了解业务的人，将帮助开发团队快速达到目标和做出适时决策。开发团队拥有很好的技术，但在业务方面需要利益关系人的帮助。通常在敏捷的开发项目中，当团队中的任何一个人需要帮助时，只要简单地邀请大家参加一个15分钟会议，或一封邮件、一个电话便可以解决。

但是，如果利益关系人各执一词怎么办？为解决这个问题，将产品负责人引入到讨论中来，作为利益关系人的代表，有权在分歧中做最后抉择。因此，通过客户代表的参与，团队更好地了解了所做事情的价值和意义，其工作效率也因而得到很大提高。

利益关系人能够帮助团队中的每一个人更好、更快地完成工作，他们的直接参与成为了敏捷开发、敏捷测试的重要前提。

3. 较少的文档 (with less documents)

敏捷开发更重视生产出可用的产品而不是详细文档，但文档又是敏捷与传统开发、测试中不可或缺的一部分。传统开发的文档在敏捷开发里仍有大用，只是原来十来页的内容精炼到现在一页半页。

敏捷主义者相信文档不是最佳的沟通方式，他们鼓励通畅的交流和沟通，要求避免和减少陈词滥调和空头支票。复杂的文档说明只能增加沟通成本，因而敏捷开发、测试的文档不需要长篇累牍，而是简洁清晰。任何一段清楚的文字，甚至一张图片、照片、一封会议记录邮件都是我们认可的敏捷文档。因为任何沟通方式和载体都是为了帮助团队进行更高效的交流和沟通，只有团队保持着沟通上、理解上的一致，才能够充分发挥出团队的最佳战斗力。凡是利于有效沟通的方式，敏捷团队都不会舍弃。

4. 最大的生产力 (maximize productivity)

敏捷开发模式要最大化地提高团队的工作效率。无论是依靠剪除冗余的文档，还是提供民主的、通畅的沟通平台，都是为了帮助团队能够集中有限的精力处理有意义的问题。据调查，通常人会在两个或多个任务并行的情况下产生最高工作效率，而敏捷正是通过各种方法得到团队的最大生产力。

敏捷开发的Scrum模式，要求团队成员在计划阶段主动制定迭代周期的所有工作任务，因此，从开始迭代活动的那一刻起，团队已经在多重工作的压力下紧张工作了。而在日常的迭代生产活动里，各个成员需要当众简单汇报当天的工作进度并制定下一个24小时的工作计划。因此，通过增加敏捷人员的工作透明度，无形之中，团队成员的生产力进一步得到提高。

5. 自动化冗余工作 (automate the redundant work)

将团队成员从冗余的劳动中解放出来，无论是自动化测试还是自动化工具的开发，只要能够节约成本，都是敏捷开发、敏捷测试的目标。

6. 民主的团队 (democracy in team)

敏捷团队是一支民主的团队，团队关系是平行的，每个团队成员能够平等地参与讨论、决策。传统开发的垂直式机构在敏捷开发中已经过时。

7. 尊重团队 (respect to team)

敏捷团队的决定权在团队自己手中。无论是产品设计方案还是产品的功能实现，都是团队决定的最佳结果。团队脱离了任何一个成员的工作都是不完整的，所以我们应当足够尊重其他成员的劳动果实并表达对其他成员的充分信任。尊重团队、尊重团队中的每一个成员都是敏捷开发的原则之一。

1.4.3 七经八络

如图1-5所示，一个完整的Scrum的项目周期可以由一组迭代周期组成，这个周期也可以与极限开发（XP）的迭代周期类比，一般典型的迭代周期为2~4周，最多一个自然月，太长就失去了敏捷的意义。但这只是一般而言，真正的迭代长短由产品负责人决定，必须保障需求变化不影响产品研发。一个固定的周期能够创造出更优美的节奏感，我们用时间盒子（Time Box）来形象描述这个节奏，而所有产品的设计、开发、测试全部需在一个迭代周期内完成，时间盒子必须严格执行，每个迭代都是如此。

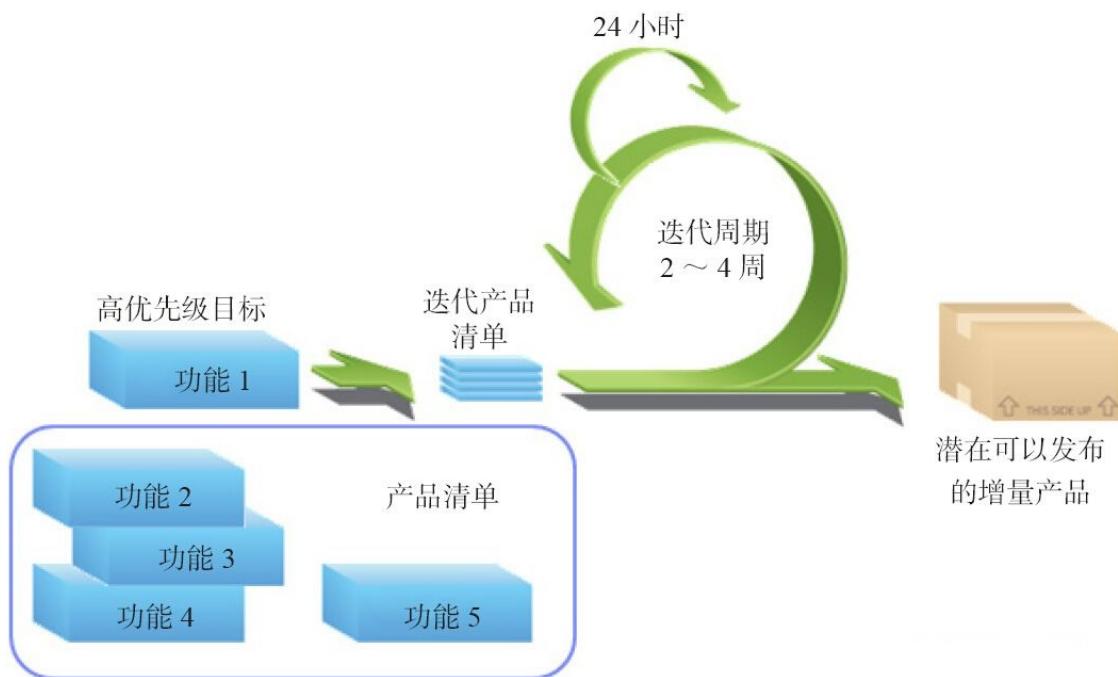


图1-5 Scrum的标准流程

Scrum的重要参与角色是产品负责人、团队负责人和团队。

产品负责人的主要职责是定义产品的功能，并决定产品发布的日期和内容，在和投资人的谈判中，产品负责人负责规划产品的投入并对产出负责。产品负责人需要始终保持着对市场和来自客户的阶段性需求和反馈的敏感度，并根据这些变化对产品所需开发的功能模块进行优先顺序排列，在拥有多位业务人员的团队中，产品负责人尤其需要统一团队对需求的定义和优先级排列。在迭代开发的流程定义和产品的计划发布时，产品负责人需要合理地调整迭代主题和迭代周期长短；在产品阶段性验收时，产品负责人还需要参与并决定迭代成果是通过验收还是拒绝交付。

团队负责人的主要职责是对项目进行直接管理，领导团队掌握Scrum的方法和实践，并体现其价值。当团队在产品研发过程中遇到问题时，团队负责人的首要责任是利用资源直接或间接协助其解决问题，以确保团队胜任其工作并保持高效的生产率。团队负责人需要保护团队不受外来的无端影响，让团队能够在良好的工作流程和环境中紧密合作，发挥出团队中各成员的职能价值。

团队的主要职责是完成迭代中的设计、构建、测试、维护等工作。所有成员均分享团队的一致目标，懂得如何配合，且无需外界过多监管而独立完成工作，也愿意在需要的时候帮助团队其他成员完成艰巨的工作。如图1-6所示，最佳的Scrum团队成员由5~9名成员组成，虽各司其职，如开发、测试、需求分析、图形界面设计等，但在工作中均表现为能力综合的多面手。良好的团队能够自我管理并有较强的自知力、自制力。而在一个迭代、甚至多个迭代中团队成员均应全职工作，即全心全意地从事一个项目，个人职能在当下迭代中最好是固定的，而在新的迭代开始后可以转换角色。



图1-6 Scrum团队组成

在Scrum的游戏规则里，主要有迭代计划会议、例行会议、迭代评审会议和迭代回顾会议四项主要的活动。

迭代计划会议。计划会议标志着迭代的开始，所有成员都需要参加，目的是通过分析、评估已有的产品清单，选择一些成为当前迭代的目标；决定如何通过协作完成当前计划，从选出的产品清单中细分，可以以人时为单位，或者以故事点为单位，团队成员领取工作项并评估所需要的工作量。在计划会议上，产品负责人和团队负责人领导会议的进行。其中，产品负责人需要确认哪些工作项最有商业价值，参考团队在技术角度的复杂度和可行性分析，设定当前的迭代目标。团队负责人则主要负责记录和协调团队工作项、计划的认领，在已有产品之上构建新的迭代内容。当迭代认领的工作项产生相互依赖时，团队负责人必须非常清晰地引导团队做出合理的工作项评估和必要的风险评估，并及时同团队和产品负责人沟通。团队的主要目的是根据所需要的最有价值的目标，尽可能按照价值队列来完成工作项目。而当团队成员认领的工作相互之间有依赖关系和关联关系时，需要考虑可行的计划和方案，以使自身的工作能够在不受干扰的环境中开展，且使被依赖工作项提交时间点尽可能对其他人产生小的影响。再者，就是将工作量细分，最好不要超过32个工作小时（如果以4周为迭代周期），而细分越详细，对工作量的评估也越准确。图1-7给出了迭代计划中迭代工作项工作量细分的示例。

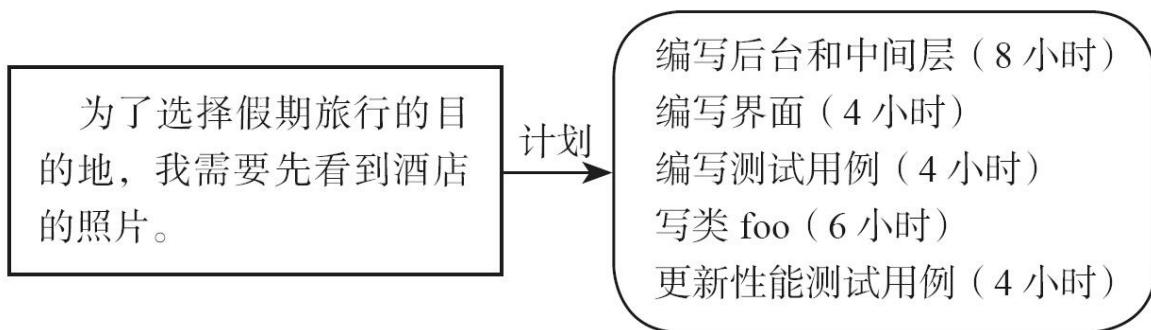


图1-7 迭代计划中迭代工作项的计划展开

例行会议。例会是一个禁止闲聊和废话的会议，所以要求大家每天都站着开，这个会不是为了解决问题，而是为了给团队一个完整的画面——我们在哪里，我们将去向哪里？在例会上，无关的人，即使是经理，只要不是团队成员、团队负责人和产品负责人，都不需要发言。会上大家会交换彼此的承诺，交换方式则是回答以下3个问题：

- 1) 过去的24小时，我做了什么？

- 2) 将来的24小时，我要做什么？
- 3) 现在我有什么工作被阻碍，或者我可能有什么工作会影响到其他人？

会议时间尽量控制在最短，说话的时间控制在1~3分钟，不需要阐述细节，也不值得展开讨论。在队员回答问题的时候，尽量不要打断，团队负责人如果没有很好的协同工具，则需要即刻记录并投影出来，让每个人都看得到。

迭代评审会议。迭代评审会议意味着迭代结束。评审会议需要团队演示新增部分的功能，还要和技术利益关系人演示底层架构的实现；如果是非正式的评审会议，即没有客户参与的情况下，仍然需要提前准备，但是时间最好不要超过2~3小时。所有的利益关系人、团队成员都需要参加会议，团队需要虚心接纳产品负责人和其他各位利益关系人的意见，在时间允许的情况下，尽可能做出充分的响应，即使没有能力回答或需要调查研究后给出反馈也需要在会上给出回复的时间期限。最后，通常产品负责人将会给出结论，即此次迭代是否完成了计划的工作，迭代是否成功。在累积了一定的基础功能后，客户将被邀请加入评审会议，此时的评审会议需要更正式。

迭代回顾会议。迭代回顾会议是为了总结工作中的经验和教训，是保证团队持续性进步和提高的重要工作，一般在1~2次迭代完成后，团队成员、产品负责人、团队负责人通过讨论和自我剖析，分析在当前环境下最迫切的问题和所需提高的能力，大家尽可能地针对自己的工作提出改进意见和方案。团队负责人需要做好调节会议气氛的准备，避免讨论陷入僵局或者争吵。

1.5 敏捷旅行

1.5 敏捷旅行

分解钢笔的再造过程让我们了解了什么是敏捷，而下面的故事将会说明，敏捷不仅仅在制造业、IT行业和软件研发中有价值，相对于谨遵计划、对抗性沟通、以文档和活动为驱动的传统交付方式，敏捷方式的灵活、以结果和事实驱动过程的交付方式让我们更加关注产品本身的质量，而不是时间和预算的控制。我们来看看这样的旅行故事。

你和你的家人计划出游，首先进行敏捷规划：去几天？去什么地方？几口人出行？坐什么交通工具？除了这些问题，还有一个终极目标：保证大家绝对愉悦，增进家人感情，并且安全、按时返回。需求并不多，就是简单的“快乐之旅”，不管去哪，和家人一起就好。或许会有一大堆待完成的任务，放在那里，不要把那些当成负担，只有当你做出承诺时，它才会成为你的既定目标，在这之前，无论任务放在那里多久，它都有可能被改变、被重新解释、甚至被放弃。所以，无需详细计划景点或时间，而只需要一个大体的构想。

你将这次出行定为四天，第一天主要完成“去”的工作，第二三天在目的地玩耍，第四天返回。日出而作，日落而息，这是生活经验，因此应该选择一个可控的频度，阶段性地开展旅行计划：早晨检查车况后驱车前往目的地，临近傍晚时分，找到下榻的酒店，和孩子们享受一顿丰盛的当地晚餐，接着爱人数着星星，然后储蓄能量迎接第二个天明。

第一天的行程中，原计划的抵达目的地时间不得不向后推延，因为可能遇到堵车、施工或者更难以预料的交通管制等突发事件。在必要情况下，要选择其他路线，或者干脆接受这样的风险：等待直到道路通畅。同时，你的“客户们”可能有许多新点子：大一点的孩子希望在河边停下来看看清风扶柳、小鸟归巢，妻子希望在经过一座城市的时候参观那里的自然博物馆，而小儿子可能因为空间太拥挤需要下车放松放松。总之，行程、计划、速度都可能被突发情况所影响，尽管如此，仍然要让大家开心、安全并可以按时返回。就这样，第二天依然耗费在路上，傍晚时分，当一家人终于满怀兴奋地步入目的地的大酒店，孩子们和妻子开了花的笑脸一定令你十分感动。然而，目标飘忽不定甚至无法预计，原本要去的博物馆因为下水管道破损早已闭馆，但你用卓越的领导力和沟通能力让妻子很快从不快中解脱出来，她放弃了不合时宜的目标，和你一起着手定制了下一个替代目标，那就是利用这段闲暇时间去当地的温泉泡个热水澡。

最后一天，依依不舍地告别目的地，而当你们异常顺利地回到家里时，才发现自己变得更能够适应一路风尘，知道苦中作乐，懂得现实中真情最令人感动，你们的心和身体都得到新的提升，感情也更加融洽。

在旅途中，最为重要的就是合理安排休息时间，保证旅行进展的可持续性。你无法想象如果因为第一天计划延期而放弃休息继续赶路将会是什么结果，或许会早些到达，但是，或许会……你亲见了道路边的一次车祸，想想还是可持续性地开展旅程更值得。况且，你的客户们似乎没有因为拖延而产生不快，因为你让他们得到了他们真正想要的，那就是一家人的天伦之乐。有时候，真的很难了解客户想要什么，或者你的客户表现得什么都想要。记住，让客户一开始对一些可以设定的目标清晰定义并诚恳认同非常重用，不要过早地、不切实际地给出承诺，因为一旦给出承诺，就要尽可能做到言必信、信必果。

旅行的故事让我们不由地联想到自己的状况，大多数埋头工作而很少抬头看路的人可能会不明觉厉；而对于那些不仅能做好本职工作，又有大局观的职场达人、创业精英来说，他们会对敏捷的价值更有认同感。事实上，这种价值正在于适应了变化且巩固了团队的信任。

然而，即使理解了前面讲述的所有敏捷方法，要在实际项目中实现敏捷仍然存在困难。原因是我们在讲述核心敏捷思想时，为了突出其要领，有意忽略了一些细节，例如，团队规

模、技术复杂度、企业架构和企业规范的限制对团队实施敏捷的影响等。

决策者决定采用敏捷交付往往都发生在“行至水穷处，坐看云起时”的那一刻，企业的敏捷演变过程周而复始，就好像历史和物种的进化，永远不会停滞。因此，无论任何时候，企业都将处在敏捷转型道路中的某个阶段，并展现出不同的状态。那么，这些企业是否遇到了一些共性问题，是否都尝到了普适的硕果呢？我们来看看2008年的敏捷实施报告。

1.6 敏捷实施报告

1.6 敏捷实施报告

从2008年Dr.Dobb的敏捷开发调查报告（见图1-8）可以看出，使用敏捷开发方法后，82%的团队的生产率得到提高。从生产钢笔的例子即可看出，敏捷开发团队单位时间生产出了比传统开发团队更大的价值，自然生产率更高。敏捷开发在激励团队、凝聚有效生产力、降低损耗、消除部门墙等方面都有很好的表现，也铸就了敏捷开发在生产率上的卓越性。质量源于客户体验，客户体验产品时满意度高则说明质量好，高质量的产品需要具备吸引眼球的界面、充分理解客户的简单化操作流程、可以定制的符合多种环境的配置需求，且处理速度能满足客户的最低容忍程度，而这些功能与非功能的需求在敏捷生产中通过和客户的不断交互逐步明确，目标也不断进行重新定位。市场就像一个游走的目标，敏捷方法就是跟踪目标的箭矢，即使目标在运动也能够击中靶心。

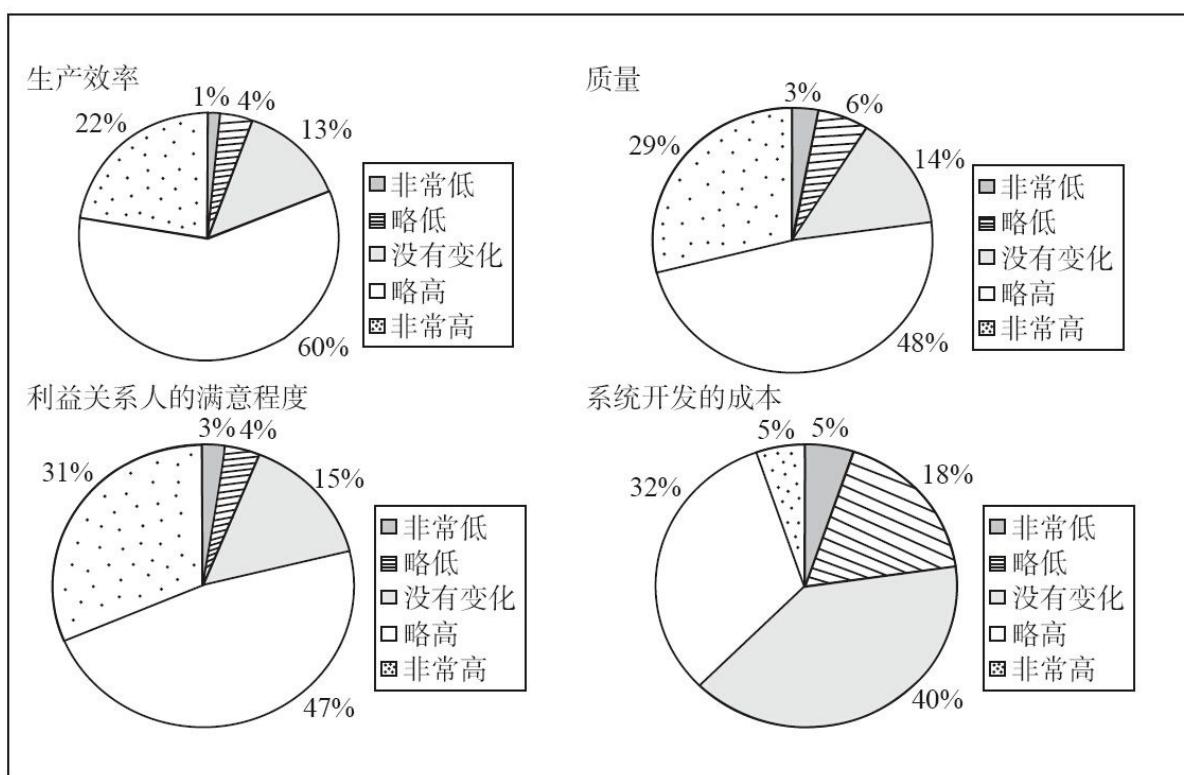


图1-8 2008年Dr.Dobb全球敏捷实施报告

2013年VersionOne的年度敏捷调研报告（见图1-9）中，在问到敏捷项目失败的主要原因时，认为敏捷项目并没有失败的占18%。而认为敏捷项目失败的人中，由于企业文化与核心敏捷价值冲突导致失败的占12%，外部压力使得公司的理念和文化遵循瀑布进程的占11%，组织或沟通问题导致失败的占11%。

如图1-10所示，在此次调查中，当问起究竟是什么组织问题影响了敏捷项目的成败时，大多数人选择了“团队缺少凝聚力”（34%），或者“未能实行以团队目标为主的思想”（28%）。

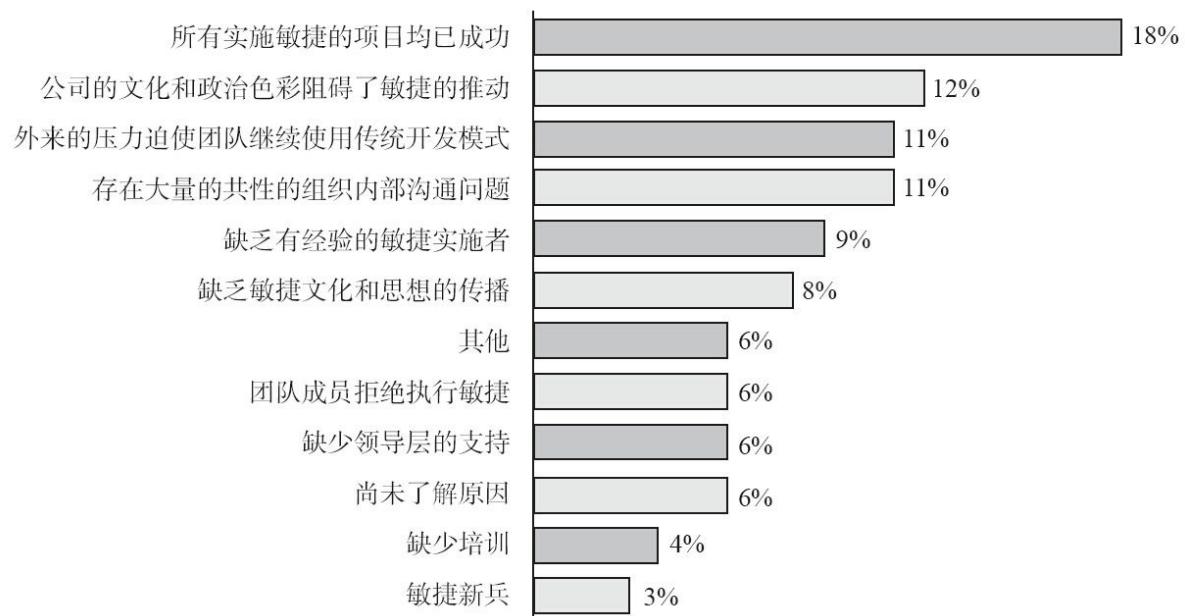


图1-9 2013 VersionOne 7th敏捷调查报告之敏捷项目失败的原因

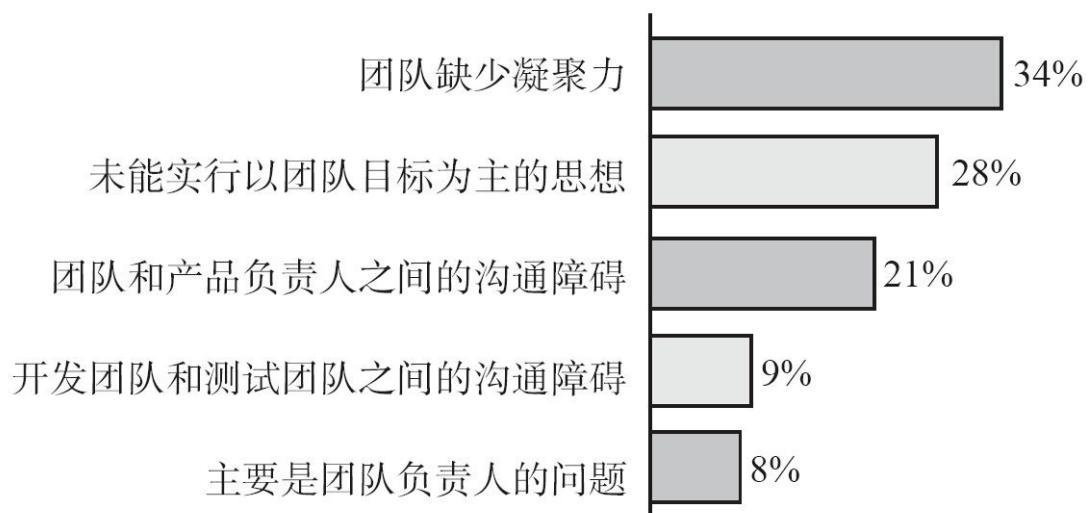


图1-10 影响敏捷项目成败的组织问题

1.7 案例分析：IBM的敏捷转型之路

1.7 案例分析：IBM的敏捷转型之路

IBM软件集团是一个庞大而复杂的组织，从2009年的软件部规模来看，全球26065名员工广泛分布在世界各地，拥有超过500个于各个时期收购的小公司以及相互独立的产品线，大量的独立工具和开发平台在公司内被采用。IBM的五大品牌行事作风迥异，很难用一种流程、一种组织结构来进行统一。

为了促进发展，各品牌组织均开发了自己的经营策略，以适合其业务所在的市场。然而，IBM为所有的品牌都定义和秉持了一套共同的技术策略：

·开放的计算：为了创新并确保客户体验的自由度，客户端必须保持灵活性，这种持续的发展和推演需要基于技术和产品自身的价值而非自身的局限性。

·SOA：为了实现产品的灵活性和资产的重用，SOA（Service Oriented Architecture）是一套可以在分布式系统的设计和实现中将系统的业务模式与IT实现紧密关联起来的方法。

·彻底简化：尽最大可能增强产品的实用性。

·产品整合力：从商业价值出发，确保中间件的产品能够易于集成，并提供适于客户环境的完整解决方案。

·产品研发的3C品质：消费性（Consumability）、模块化（Componentization）和社区（Community）三个原则，聚焦于改进开发流程和开发过程本身。

对于软件部尤其提倡3C原则，以有效推动业务成长并提高生产效率。消费性影响软件的销量、支持的成本和客户的满意度（这反过来又影响了销售）。模块化使得功能相同的组件被抽象出来，并根据需求进行聚合，从而将产品有效推向市场，且降低成本，快速交付，驱动大回报。而所有这一切都取决于我们作为一个单一的软件部社区的技术深度和广度。在扩大社区格局的同时，团队也收获了相互协作和勇于创新的硕果，这也正是我们的竞争对手所羡慕的。

3C原则带来的产品的质量提升，还仅仅是在以产品为中心的主流市场获得的成功体验，但是，当市场发生巨大变化时，战略又将产生变化。2006年的互联网高潮后，软件部受到全球IT经济低迷的冲击，股票开始下滑，面对开始失去信心的股东、董事会，为了生存，IBM不得不将战略导向从稳定业务转向了动态业务，提出“随需应变”的核心战略，软件部由此提出了敏捷研发方法，最大化地提高生产力，将产品更快、更好地推向市场。

1.7.1 转型历程

IBM的敏捷转型并没有残酷的清洗，反之，转型过程自然而和谐。IBM首先选择了一部分项目作为敏捷推广的先驱，通过招募以及内部发掘，培养了一批敏捷实践者和推广者，并把这中间最优秀的敏捷大师组成CoE（Center of Excellence），由IBM副总裁直接领导，对全球所有软件部门进行敏捷培训。2008年，当我被提名参加中国研发团队的首批敏捷培训时，我已经谙熟敏捷测试，而我们是直接接受CoE团队培训的第77批学员，在那里，我认识了我的导师Scott和IBM推广敏捷的执行副总裁Sue。从此，我热爱上了我的事业并知道我将向何处寻觅我的目标。

2006年或更早，IBM QSE社区作为IBM内部最权威和最大的技术社区，开辟了敏捷专区，分享各类相关的敏捷资料。2010年，敏捷社区注册人数已超7000人，阅读社区中来自不同国度的满怀热情的敏捷实践者提出的问题并给出回答，是我当时乐此不疲的事情。我从社区里

大量下载大师级敏捷领导者发表的关于敏捷开发、持续化集成、测试驱动开发等核心敏捷实践的经典原创文章，不久，我也开始发表自己的文章，虽然每每只是窥见一斑，但我的文章拥有大量真实的数据和作为实践者的谏言，因此颇受欢迎。2008年，我将部分有关敏捷测试的文章第一次发表在中文的DW网站上，这篇文章使我收获了更多的勇气。

2007年，有一篇发表于敏捷社区的文章，内容是转载了IBM跨软件部各个部门、产品线正式启动敏捷和精益方法的推广声明。

声明的大意是：“我们均已意识到持续增长的业务压力，这就要求开发团队更具灵活性并能将产品更快地投入市场。敏捷和Lean开发方法，旨在将最有价值的部分传递给客户，去除“浪费”，授权团队有效地面对和处理复杂的业务，这些均已成为我们在软件领域的目标和信念。一些先驱团队尝试在IBM内部推行敏捷方法已有一段时间，我们相信，软件团队有能力成为敏捷实践的最佳推广部门。”

2008年，软件部总裁在其个人年度计划（PBC）中包括了关于敏捷实践的两个重要条目：

- 加快敏捷实践的普及，并运用Lean的原则，提高开发者的工作效率和对客户需求的响应速度。
- 将IBM的“可扩展敏捷”业务模式展现给客户。

与此同时，总裁的个人规划被自上而下地推动到每个IBM员工的年度计划中，无论是管理层还是技术层，都承诺了与敏捷相关的目标，因此，软件部各个组织和团队都积极地希望做得更好，取得更好的年终绩效，成为其他团队借鉴的榜样并以此为殊荣。在这一年里，几乎所有人都在思考和学习如何在自身的工作中使用敏捷方法，以及如何配合团队实现敏捷目标。中国开发中心的员工也大量注册到IBM QSE的敏捷社区，主动成为敏捷的倡导者和实践者。主导的团队成员还自发组织成立了中国的敏捷社区，相互分享经验并帮助解决敏捷实践中遇到的各种问题。2010年，仅中国区注册的敏捷社区成员约为750人，全球有6000余人注册，领导团队成员达到12名。与此同时，敏捷专家组（CoE）成立，帮助IBM实现精益和敏捷转型。CoE的主要任务有两项：寻找能够适用于多团队的行为模式；开发出更优秀的适用于IBM团队的新行为模式。

1.7.2 品牌的力量

网易创始人丁磊经常跟他的员工讲《老鹰》的故事——一枚鹰蛋不幸落入了鸡窝，母鸡孵化了小鸡和小鹰，带着它们吃着农场主洒下的细碎的谷梗。当它们成群地在乡间踱步时，小鹰没有一次扇动翅膀飞起，也没有怀疑过自己的模样。小鹰慢慢长大，直到有一天，鹰妈妈发现了小鹰，把它叼到悬崖上的鹰巢，告诉它：“你是鹰，你属于蓝天，你能够飞翔。”然后把小鹰一把推下悬崖，于是小鹰挣扎着张开翅膀，飞了起来。这个故事蕴含的道理是，每个人、每个团队、每个企业都有潜在的能力，不要被习惯所遮盖，不要被成功经验蒙蔽，不要忘了远大的志向，要不断激励自己，让自己飞起来。这个故事对丁磊产生什么影响我们未必得知，但是从他带领的团队来看，这是梦想的开始：向往飞翔，成就未来。

愿景对品牌的建设尤为重要，缺少愿景，品牌建设只能在激烈的竞争中迷失方向，在“合理”的业务延伸中失去衡量的标准，致使发展受阻。良好的品牌愿景给员工、客户、股东、社会一个理想的世界，帮助企业找到前进的方向。

IBM的品牌历经起伏。1993年郭士纳接手IBM的时候，IBM正面临着前所未有的困境，IBM品牌等同于“古老、高傲、尾随新一代高科技公司之后”。郭士纳就任IBM舵手之位后，采取了紧急止血的业务运营措施，扭转了逆局。面对信息公路的高速增长，客户对整体解决方案逐渐倚重，对不同供应商的产品组装越来越没有耐心；芯片的计算速度、存储成本也越来越无足轻重。因此，IBM决定放弃文化传承的讨论，而是将公司从一家大型主机制造商转变为同时提供硬件、网络及软件整体解决方案的供应商。十年励精图治的变革使IBM品牌终于摆脱了困境，而在新一任CEO彭明盛上任时面临的是一种新的挑战：如何使员工不再回到自满和

安于现状的老路上，如何通过理想和抱负来鼓舞他们继续前进。因此，对于IBM来说，确立一套新的企业价值观成为公司任务的重中之重——这绝不是只做表面文章的空话，而是要让价值观成为团结和激励员工的有效手段，其实质就是要运用价值观进行管理。IBM公司凭借创新的服务理念和卓越的执行能力连获“2006年度中国最具价值IT服务商”、“金融行业专业服务满意奖”和“电信行业专业服务满意奖”。又一个十年过去，如今的IBM正在面临巨大的格局变革，从稳定业务转向动态业务，以新的软件系统结构满足丰富却变化的客户需求。

1.7.3 核心竞争力

为了做好全球资源整合，树立新的百年基业长青的成功企业，在如今的市场上取得更大的成功，IBM在战略上再次做出调整。IBM品牌战略团队认为，IBM的团队和个人应该具备新的能力，为了帮助员工与时俱进，培养新的能力和工作方法，也为公司未来的成功奠定基础，IBM的品牌核心竞争力有了新的定义。

1) 拥抱变化。IBM员工应当专注于已经发生以及将来可能发生的变化，这个变化会发生在我们或客户身边，面对挑战，我们要有能力找到问题的突破点并向前迈进。

2) 成就客户。IBM员工要以实现客户当前和未来的愿望为目标，并为成为客户的合作伙伴而感到骄傲。我们和客户协同创造的解决方案和收获的成功经验，都将会转化为企业和社会的宝贵财富。

3) 全球协作。IBM员工遍布全球，因此必须精于合作。我们需要跨越团队、领域、思想、学科、国家和文化的界限，通力合作以寻找正确的结果。作为全球一体化的企业，我们建立了自己的专家网络，鼓励员工使用集体智慧，以集体领导力带领IBM改造市场、社会和世界。

4) 用系统的眼光行事。IBM员工是系统思考者。我们帮助客户、同事和世界理解并设计系统——它是如何感知、关联和分析信息，发掘潜在模式，并将知识转化为信念和行动。无论系统中的问题属于技术、经济、社会、文化或自然范畴，这种系统性的观点使我们能够甄别出“真”的问题，并在适当的时间、以正确的方式采取明智的行动，同时敢于承担风险。

5) 建立互信。无论在IBM的内部组织或客户中，还是在发展、变化的外部世界中工作，IBM员工应善于建设“全方位的信任”：即使有不同的愿望、意图、约束和文化，我们都可以找到共同点。以诚信行事构建人与人的关系，即使未能达成一致，也应明确意图、确保开放和对信任的坚持。要知道做什么和怎么做，并有足够的激励来实现目标。一旦发现信任被侵蚀，我们必须立刻担当责任并迅速弥补。

6) 厚积而薄发。IBM的价值主张和商业模式植根于专业的知识。因此，需要不断加深专业性，成为IBM有价值的员工、合作者、领导者。员工可以通过反馈、辅导或者从有挑战性的任务中学习，从而发展技能和职业。

7) 持续改进。IBM员工都致力于建设一个更好的IBM公司和一个更好的世界，这是IBM公司在过去的一百年所致力完成的。我们的求知欲和不停息的进步不断为企业注入能量。今天的世界，未来难以预测，IBM员工正在积极寻求我们还不知道的和我们尚未想象到的。因此，我们培养开放的环境，尝试新的方法。我们重新思考假设，提出探索性的问题，掌握新情况，发掘机会，并创造新的市场。我们乐于结交拥有不同背景、文化、语言或工作作风的合作伙伴，这是我们适应不断变换的基因所决定的。

8) 沟通与影响。沟通的目的是相互了解，建立一个共享目标并分享成果。沟通始于倾听，要确保人们的想法和顾虑都能被我们听见。我们以专业的角度专注于简单、清晰的沟通，尤其是在复杂环境中，我们需要解释不同的概念、战略和意图，我们会利用对他人观点的理解来丰富自己，这种态度和方式让我们受益，即使在困难的话题上，或者不受欢迎的环境里都能够以最及时、最有效的方式进行沟通。

9) 帮助其他IBM员工成功。IBM品牌是关于IBM员工的，以及IBM员工如何出现在世人的面前。这不仅仅限于面向客户的IBM员工，每一个IBM员工都力争把最好的一面、最优秀的自我展现在工作中。为了帮助员工成功，要确保他们有充足的资源、持续的支持和清晰的里程碑。我们会彼此分享见解，讨论面前的挑战并排除障碍。我们承认他人的贡献，对他们的想法给予鼓励，并帮助每个IBM员工找到自己的内在驱动力。我们创造一个让员工感到目标一致且可融入的环境，以激发他们强烈的欲望并协同行事。

1.7.4 敏捷品牌

IBM的品牌愿景规划出公司明天的优势、使命和价值观。品牌愿景不仅使员工看到了高瞻远瞩、力拔头筹、高效协同的未来，也让客户对IBM更有信心。构建明确的品牌愿景需要企业对内外部环境进行非常透彻的分析，并真正从内外部环境需要出发，理解自己应该与客户建立一种怎样的关系，这是值得许多企业高度注意的问题。在很多情况下，企业容易遇到的问题是：单纯地从个人意志出发理想化地定义自己的品牌意义，如“为客户创造价值”、“努力奋斗”等，而敏捷方法论在软件部的试行和最终采纳则是源自内外环境的需求，针对可能存在的不确定性，敏捷实施的重点在于：

- 在发展进化、协同配合和自我管理规则下持续高质量的交付。
- 延展敏捷开发方法论，使其成为解决整个软件、系统工程全生命周期模式的方法论。
- 基于风险和价值驱动的方式来生产有价值的产品和解决方案。
- 从共性的规范性敏捷交付方法中，透析并实现独特的个性需求敏捷。

敏捷思想强调价值、沟通与合作、质量、增量型，这与品牌价值自然融合，也是许多公司同IBM一样在坚持使用敏捷方法推动软件项目发展的原因。敏捷方法并非一成不变，就目前行业统计数据来看，超过76%的团队正在使用多种敏捷方法，而且坚信敏捷方法可以使企业获得成功。与传统的开发项目相比，敏捷开发项目有更可观的成功率、更强的软件和系统生产能力、更高的客户满意度、更佳的投资回报率（ROI），并且能够更快地将产品投向市场。

1.7.5 成功转型

IBM敏捷转型的结果如何？不妨阅读一篇来自G.P同学的文章，《How IBM Saved \$300 Million by Going Agile》（如何通过敏捷赚取30亿美金），果然大象舞蹈得更加敏捷了！

“怎样才能使世界上最大的软件组织更加灵活、更加适应市场？您可能已经读过或听说过，任何我们的专家和思想领袖，如斯科特·安布勒、布鲁斯·道格拉斯、沃克·罗伊斯都会歌颂敏捷开发和IBM的敏捷交付方法。IBM Rational团队坚信，不管你有多大的软件开发团队，你都可以从敏捷转型和敏捷方法中受益。”

敏捷技术果真能够运用至像IBM软件部一样大小的“店”吗？这个巨大的组织，我们称为“SWG（软件部）”，已经成功尝试了敏捷转型。而这在一开始显得颇具挑战，毕竟，我们已经看到过去十年SWG业绩的稳步增长，我们还需要采用新的技术吗？而当时的SWG领导团队做出了最正确的决定，从而带来了今天的成功。

采用敏捷方法将帮助任何一个组织提高生产力和利润，而这远优于其之前采用的相关措施。几年前，软件部开始了敏捷软件开发实践及相关敏捷开发产品的研发。为了适应他们的需求，我们开发出适合敏捷环境的工具，包括用于协作的Rational Team Concert、用于协助组件复用的Rational Asset Manager，以及用于团队会议和消息功能的Lotus Connection，而这些工具也非常适用于跨国或者分布式团队开发环境。

如果需要通过数字说明敏捷转型的结果，那么，IBM软件部因为推行敏捷，节省了超过30

亿美元的软件研发成本。截至2012年，80%的IBM软件开发团队使用敏捷开发，员工人数超过62000人（2011年下半年为57000人），跨越346个产品领域均使用了Rational Team Concert，帮助所有人实现了项目实时可视性，保证了项目的健康并行。

而敏捷方法并非只在软件领域发挥出优势，IBM系统和科技集团软件开发人员克拉克·杜德克说：“我们以前使用配置管理版本控制，这需要一个漫长的代码检查过程。Rational Team Concert的引入有利于代码协作，团队能够更好地实现工作项跟踪。而与此同时，系统部的同事正在使用敏捷方法和工具进一步改进他们的工艺和生产率。”

敏捷为IBM带来的商业价值更多地是帮助企业、团队发现哪些要素需要用来构建质量更好的产品，或更快地将工作成功迅速地转变成商业价值。这既包括了“过程、人”在提升效率后产生的价值输出，也包括了其他帮助我们走上敏捷之路的工具（例如RTC）所带来的直接收益。30亿远远不及“敏捷的引入”带给IBM的全面商业价值。但对于“过程、人”等环节发生转型后所产生的价值，因为这个过程缺乏“可逆性”和“有效度量性”，我们没能给出这部分的价值度量。而对于RTC，它并非一个可以替代的没有灵魂的简单工具，RTC已经融合了敏捷流程模板、代码和敏捷主义的思想，成为IBM团队所认可的不可以替代的“智慧工具”，成为如今最好的“整合团队敏捷度和企业项目透明度”的工具，尽管它还有缺陷。

第2章 敏捷落地艰难

第2章 敏捷落地艰难

2.1 敏捷实践误区

2.2 理论的局限性

2.3 传统团队践行

2.4 敏捷实施法则13条

如果在IBM做个调查，几乎80%的敏捷研发人员，无论是做技术还是做管理，恐怕都会告诉你他们使用的是Scrum。而当你深入地了解他们的工作、流程、方法、交付结果时，却发现此Scrum非彼Scrum。

尽管大家对Scrum的理解不同，但我们必须承认，大多数IBM中国的研发项目敏捷转型确实是从Scrum开始普及的，从而衍生出IBM的核心敏捷（Core Agile）、规范性敏捷交付（Disciplined Agile Delivery, DAD）和可拓展敏捷模型（Agile Scale Model, ASM）等。

Scrum这种核心的敏捷方法对开发团队来说的确重要，就连RTC工具发布了大量的“开箱即用”的Scrum流程和模板。除了IBM，业内的同行也大量使用了各种核心的敏捷方法。

2.1 敏捷实践误区

2.1 敏捷实践误区

团队接触Scrum一两个月后，往往自认为对Scrum的方法和流程已经足够了解，此时容易犯下几个小错误。但如果不能及时加以纠正，小错误将可能导致更大的错误，以致阻止Scrum或者其他敏捷方法的正常实施和组织的顺利转型。

1.越频繁变更越“敏捷”

进入这个误区的人不在少数，大多数人认为“快”就是敏捷，Scrum中的迭代周期如能缩短，团队将会更加灵敏。有些团队成员一边“享受敏捷的礼遇”，一边“开着小差”，觉得自己隔三差五给团队个交代就可以了。于是，想到哪里设计哪里，不和客户保持经常的接洽，更不会管理客户需求，只要是客户说的，无论成熟与不成熟都扔给开发团队。这导致了产品三天一小改、一星期一大改，这样的“Scrum”开发模式把团队完全弄懵了，还没明白过来是怎么回事，就被告知“进入下一个迭代了”，一边重复工作，一边又被灌输越“快”就是越“敏捷”。这种做法显然可笑，因为敏捷的增量型递增模型完全不存在，团队变成了“无头苍蝇”。

不得不指出，这种现象经常发生在互联网行业，这个行业的客户其实就是普通大众，对于一项网络功能，一半客户稀罕、另一半不稀罕是经常发生的事情。例如，针对网页中的Flash动画，如果询问用户，喜欢和痛恨Flash的差不多一半一半。如果不仔细观察和分析这类普通大众的需求，很有可能今天加一个、明天再去掉一个Flash，并且开始争辩哪种方式最好，既浪费时间又耗费团队的精力。

实际上，如果仔细分析，就会发现痛恨Flash的用户其实是不喜欢使用不当的Flash，那些大而复杂的Flash下载时间长，用户没有兴趣完整地观看。所以，设计过程中变化不可避免，但绝不要用“缩小迭代周期”、“短频快”的做法来遮掩设计的缺陷。

所谓“武功唯快不破”的只是小说，而不是敏捷开发。

2.“自我管理”等于“随心所欲”

我们鼓励自我管理是因为一线的员工比以往更聪明、更训练有素，我们发现员工的高效率来自充分明确自己的工作，并以最好和最省力的方式来实现。而且，员工在自己处理的问题时，往往表现出高度的集中和快速的提高。自我管理是授权给员工自己设定目标，将工作当做自己的事情来做。在一个高信任、低敬畏的组织里，员工不需要太多的监管，但是他们需要理解和支持，而不是“老板”时时在左右。

自我管理制度和“层级组织制度”针锋相对，在新的制度里我们不希望见到“老板”、“总裁”、“管理人员”的字样，这些高级头衔并不代表有能力“Lead（领导）”低级头衔。一个领导的影响力来自他展现出来的做事能力，以及作为团队建设者的卓越性，这类人往往为团队成功做出了较多的贡献。

但自我管理制度仍然有界限，无政府主义和滥用权力是不允许的。一个自我管理的团队是高效的管理系统，自由度不是无限的。美国的Whole Foods连锁超市，其最基础的组织单元不是门店，而是团队。Whole Foods的团队是零售业史上空前的自治团队。每个门店约由8个团队组成，他们对门店的各个环节进行管理，从食品加工到收银结算。按规定，每位新来的同事都会被分配到某一团队，经过4周的试用，团队成员投票决定这位新同事的命运——新同事需要获得2/3以上的赞成票以获得全职岗位。Whole Foods的决策者相信，关键的决策问题应当由那些受该决策结果影响最大的人做出，例如，雇用谁的问题就应该由未来和他共事的人决定。Whole Foods的团队绩效考核是透明的，绩效超过一定额度的团队得到更多的奖金；如果一个团队将一个懒散的成员纳入团队，他们下个月的奖金就可能减少！事实上，如果这些团

队成员没能勇敢地站出来反对那些滥用权力的团队领导，或者没有否决雇佣低效率的新员工，就不算真正的自我管理。而这种高度的自治、自我管理方式向我们证实了一个道理：员工自我管理并决定自身的成败，而不由管理人员决定，自我管理绝没有变成“随心所欲”。

3. 年轻的团队最有“创造力”

这是肯定的，但是，没有经过“培养”的新人，在没能判断其是否可以独立工作之前，不要急于让他上岗。对新人的培养、新团队的建设是绝不能忽略的。事实证明，入职过程很少能在极短的时间内完成，一般需要2~3个月，而IBM会用6个月完成新员工入职的一系列培训，包括法律法规教育、公司文化熏陶等。

当然，并不是刚入职的员工不能成为Scrum团队的一员，有很多刚入职的员工是明日之星，只是不能将工作放心地交予没有基础的员工。年轻的团队需要教练、需要引导、需要有经验的人带着走入这个有声有色的圈子，还需要理解流程、通晓团队中的各项职能、理解如何定义工作的“完成”，然后才有能力上岗。一支团队从组建到完全释放每个人的能力需要经过团队初建期、过渡期、风暴期和成熟期四个阶段。在IBM的几乎所有工作项目中，都有比较近似的团队初建过程，也就是团队成员在一起共同分享和探讨下面这些问题，确保所有人都做好了准备。

- 互相介绍背景。
- 给予团队一个名字。
- 确定团队定期召开会议的时间和地点。
- 分享团队的一般工作流程，介绍团队工作中的主要递交物。
- 教会新人如何判断工作已经“完成”。
- 建立团队的独立规则，例如请假需要跟负责人提前沟通等。
- 如果需要，确定团队定期培训的时间和培训内容等。

任何团队都必须融入组织中。IBM的组织结构是矩阵式的，一般员工都有三个“领导”。第一个是行政经理，负责批复假期、做入职手续、离职手续。如果员工需要资源，一定要先找到行政经理，因为他离员工往往最近，没有人能够像他一样更好地支持员工，我们可以叫他“经理”。第二个是职能经理，负责员工的工作绩效考核，和员工一起制定工作季度规划、年度规划。员工要经常和职能经理保持畅通的交流，这样功劳、苦劳才会记载在功劳簿上，我们可以叫他“老板”。最后是员工所参与的项目中具备相同职能的一个“老人”，我们姑且叫他“Lead”或者“Technical Lead”，这个人身先士卒，极有威望。对于入职的新人，我一般都建议他们去学习Lead的工作方法：你看他怎么干，你就怎么干，一定没错的。

我们需要Lead参与团队的培养，Lead需要分出一部分时间来做团队的“领导”、“教练”、“老师”，学会时间管理和授权，让自己专注于更有价值的工作，而将执行和具体的操作交给团队的新人，同时做好以下几件事情：

· 在老板和利益关系人面前代表团队做出承诺（例如工作时间、交付期、工作范畴），任何时候都不需要告诉老板和利益关系人你还需要和团队商量后再确定，第一时间代表团队做决定，要果决。

- 监控团队的工作进度，最好是通过系统监控，不要在团队认真工作时打扰团队。
- 如果出现问题，必须立刻参与进来，判断和解决主要部分，并将一部分工作交由新人来做。

- 对团队新人需要经常表扬，必要的时候批评，棍子和糖都需要。
- 分配工作，让每个人都充实起来，多给予员工支持。和新人的沟通一定要到位，不要简单粗暴，一定要让他们认为工作是可以承受的，才能够放心授权。
- 经常帮团队申请资源，比如聚会、活动等，但是一定要让团队有机会展示自己。我的经验是，团队曝光率越高、占用资源越多对团队越好。

不过，对新人不能太“民主”，尚不能任由新人自己从Scrum的产品清单中选择工作项目。还有，和利益关系人的沟通也要一步步放开。这绝不是“反”敏捷，而恰恰是非常务实的经验之谈。让年轻人先将那些“天花乱坠的想法”收敛起来，虚心学习他人的智慧，懂得尊重前辈，不跳跃、不敷衍团队建设这个自然过程，才可以形成良好的团队氛围并奠定扎实的基础。对于年轻人来说，先学会“规则”，看到“边界”，虚心学习，这一切都将对他们的个人发展产生积极影响，他们会看到成长后将有更多的自由选择空间，包括喜欢的工作和喜欢的工作方式，也会期待将来的一天成为“Lead”，可以有“弟子们”的帮忙，做更有成就感的工作。同样，对于资格已老的团队员工来说，通过悉心帮助成长起来的新人让他们有更多的时间去做更有价值的事情，包括许多以前很难做到的事情，并带来成就感。而且，“师生关系”潜移默化地消除了“长江后浪推前浪”的不安，老员工心态会更平和，也能看到更远的人生。

总之，团队建设初期，或者新人刚加入团队的一段时间，建议经过培训和教育再参加Scrum工作模式。团队Lead扮演了重要的明星Scrum成员的角色，在团队中既要做导师、教练，又要在Scrum团队中承担工作。所以，一个比较好的组织方式是，Lead直接参与Scrum团队的各项工作，而新人在入职的一段时间里跟随Lead参与Scrum的所有活动，但是不发表意见，只做聆听者、学徒和主要的执行者，Lead会给出新人的一切工作安排并负责与其他Scrum成员和利益关系人沟通。

4. 不了解项目的产品负责人

产品负责人（Product Owner）的主要职责是定义产品的功能，他的任务包括倾听用户需求、负责产品功能的规划等。后端的工作也需要产品负责人，如解决各种问题、处理组织关系、有效地和研发团队沟通产品的功能需求和设计等。产品负责人的问题是忽视燃尽图和整体产品研发进度的跟踪。“无论如何，那不是我的主要工作，团队负责人和项目经理、开发经理应该更关心燃尽图”，一个资深的产品经理甲曾经对我如是说。又有产品经理乙曾对我说：“产品负责人负责思考需要开发什么，而开发团队、开发经理负责如何开发，所以，燃尽图不是我的工作。”

不关心进展和状态的产品负责人很少有兴趣了解迭代的过程，也不了解团队当前遇到的问题、关心的话题、需要的帮助。所以，在迭代后期，会出现由产品负责人引发的不必要的需求变更。

不经常、甚至不愿意和团队保持通畅的沟通，因为那些对他们来说是浪费时间和效率的做法，与团队进行针对需求和设计的讨论对他来说也是对自己尊严和地位的挑战，因而，很难让团队能够“转身就可以和产品负责人讨论一个必要的细节”。相反，产品负责人更愿意一次性交付给团队全部需求和设计，同时给出详细的要求，甚至对自己的杰作有着另一种苛刻，细致到了“完美”的程度，而这样的需求和设计对于团队而言形同“一纸文书”。产品负责人愿意将和团队面对面的沟通回归到电子邮件或其他书面沟通方式，而这种做法深深地伤害了团队对自身工作的认可和参与新模式下研发工作的热情，团队会怀疑自己是不是真的找到了“为自己工作”的热情，面对产品经理，他们会觉得自己是“低人一等”、“受人支配”。产品负责人“独立”工作的态度和方式也将阻碍“有深度的思想”传达到位，同时将增加团队内部的沟通成本并损失一定程度的工作效率。

显然，这样的产品负责人仍然会给自己找不少“麻烦”，倒不如保持平常心，把自己和团队放到一个水平面上，更多地关心和尊重团队。比如，认同团队在设计细节上有着更为精准的定位，在产品质量和验收上是更为直接的受益者和关系人。而且要关心团队进度，如果团队

正处在峰值工作，应该避免给团队更多压力；如果团队正处在高效率、健康的工作状态中，可以将新的产品清单增加到迭代中。这些都需要产品负责人看懂各种状态的燃尽图，那么我们来了解一下燃尽图如何体现项目的状态和进展。

(1) 发布燃尽图

在图2-1中，实线部分描述出实际发生的每个迭代所增加的工作量（用故事点计算工作量），实线的曲率表示开发的速度，即单位迭代所完成的工作量，曲率上扬表示增量开发继续在先前的工作基础上增加了新功能，曲率下行则表示以前开发的故事已经被放弃（虽然不愿意看到，但是几乎不可能杜绝）。虚线部分代表发布计划，表示对这个发布燃尽速度的大概规划，这条曲线只作为参考。图中第二个迭代完成的工作量落后于计划，这意味着以后的工作强度要比计划的大，或者计划部分的工作量有可能需要从发布计划中剔除掉。

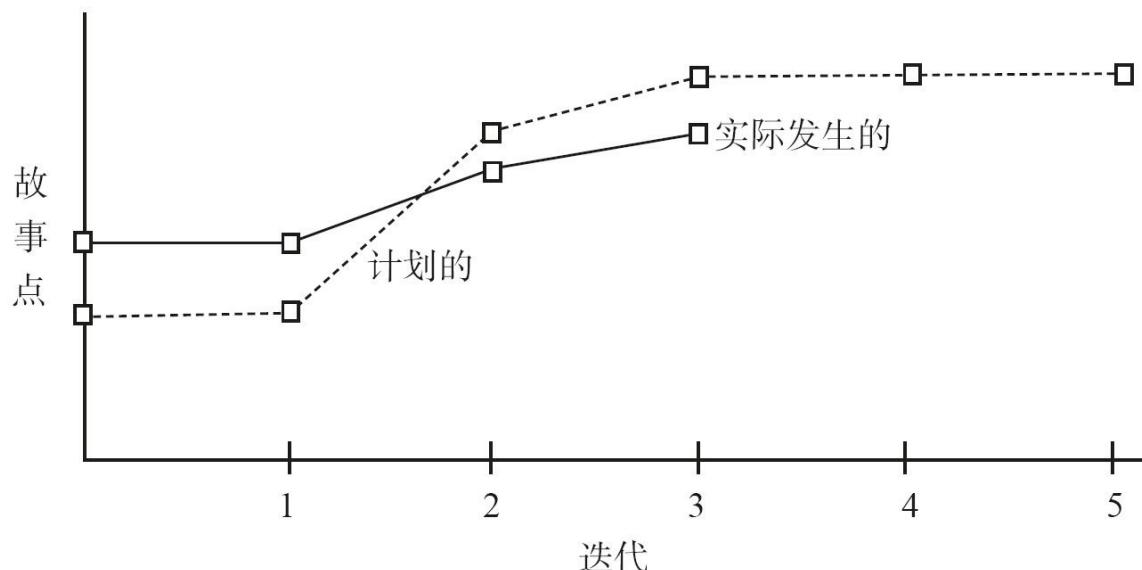


图2-1 发布燃尽图

(2) 迭代燃尽图

迭代燃尽图（见图2-2）即在一个迭代内标示工作量完成的进度，图中的虚线部分是每天实际发生的工作量，实线部分是计划燃尽曲线，这个图是用“减法”思考的，随着时间的推移（天/小时）曲线从最高点滑向“0”。

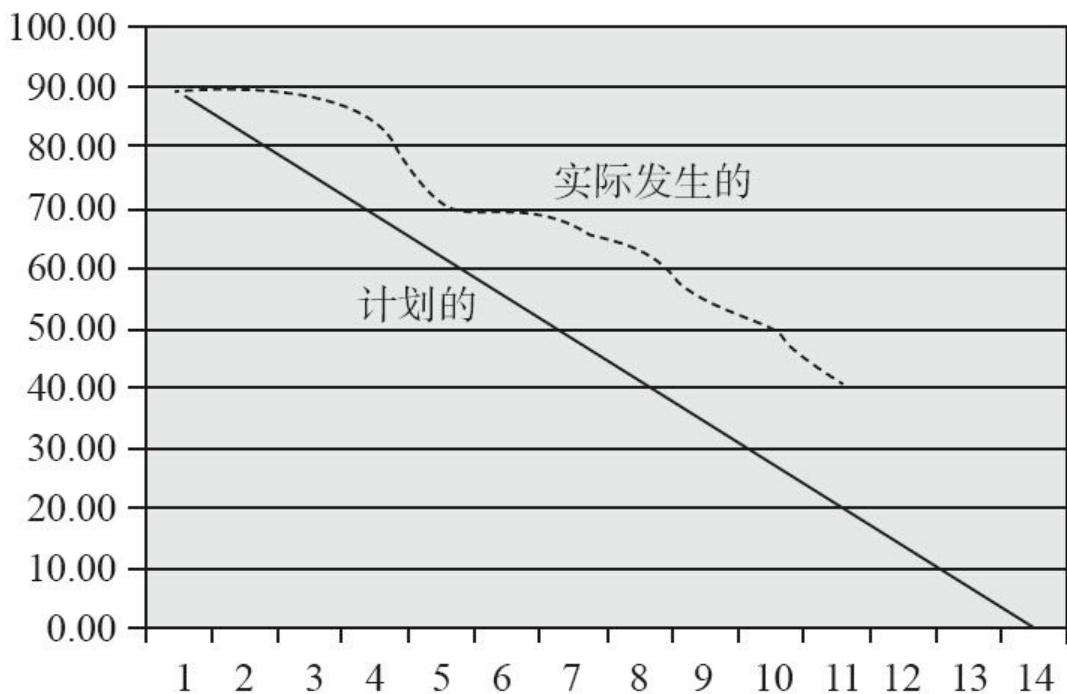


图2-2 迭代燃尽图

如果你是产品经理，燃尽图所表示出来的信息你能够都读懂吗？看看图2-3的分析，你就会明白了。

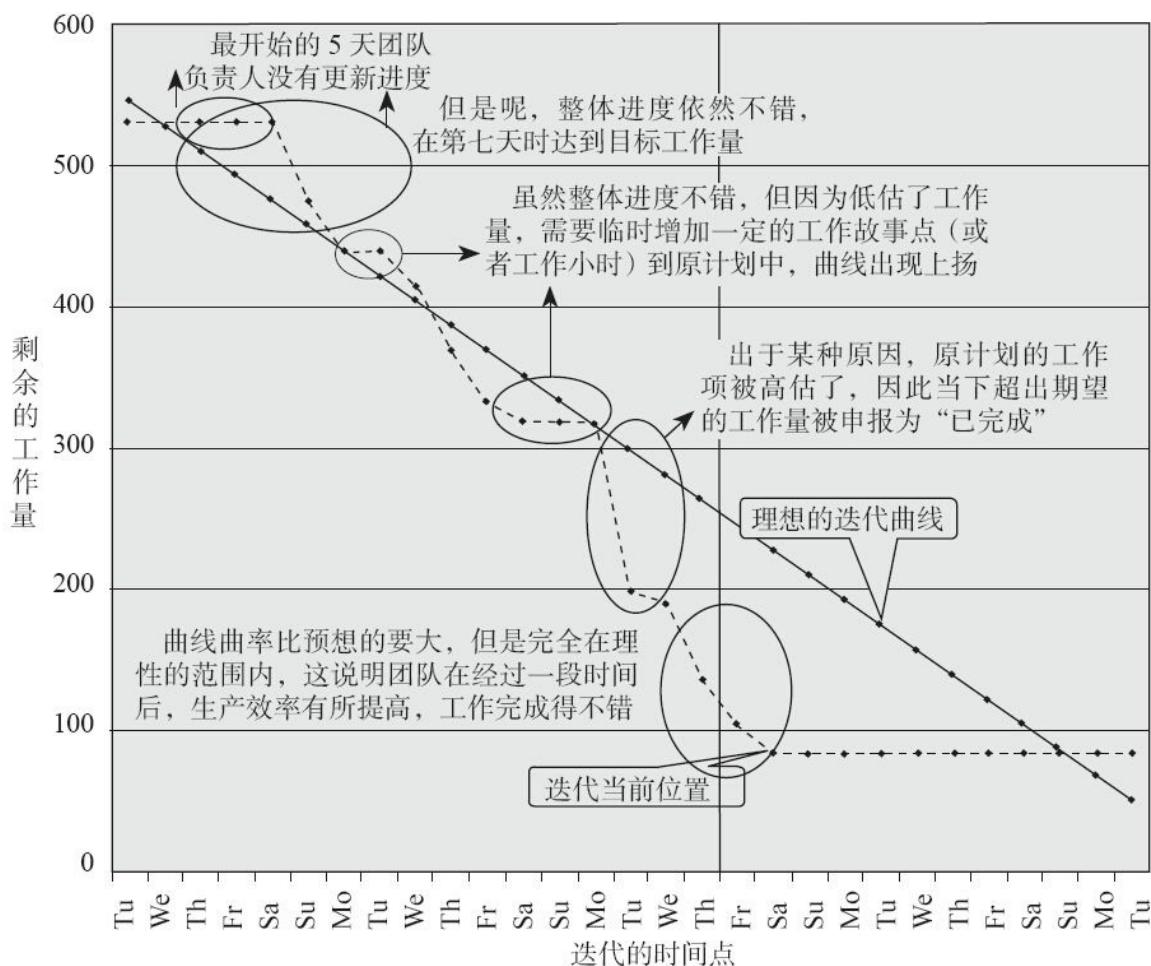


图2-3 如何读出燃尽图背后的故事

这张图是我在管理团队项目时的实际燃尽图，折线部分是实际发生的燃尽曲线，我们从上至下看看几个被圈出部分所代表的含义：

- 1) 因为没有更新工作项进度，或者没有统计每天的燃尽工作量，或者工作量阻滞了，这个区域燃尽曲线是水平的。
- 2) 当时间进展到周六、周日，大家通过加班赶上了原来被阻滞的进度，情况还算不坏，但看得出来大家很用功、很辛苦。
- 3) 第3个和第4个圈出的区域显示曲线的下行突然被打断，而实际情况是并没有工作被阻滞，也没有忽略统计，结果只有一个，就是团队低估了工作项（Over Commit），在开始迭代的第二周，团队增加了对该工作项的评估工作量。
- 4) 第5个圈出的区域有陡峭的曲线，这显然不合乎我们对团队工作能力和开发速度的认知，后来发现，这主要是由于团队对已完成的几个工作项有过高的估计工作量（Under Commit）导致。
- 5) 第6个圈出的区域也有较陡峭的曲线，原因是在迭代的后期，大家担心时间不够，怕拖了团队的后腿所以真的非常用心，也非常高效，而且周六也确实在加班。

可见，产品负责人学习读懂燃尽图，对于团队的工作速度、能力认知非常有帮助，能够明白团队的辛苦，了解当下团队的工作强度，帮助产品负责人做出判断，何时“见缝插针”，何时“减轻工作负担”。

5.好大喜功

中国式敏捷很像盲目的城镇化。有些人简单地理解城镇化就是钢筋混凝土的高楼大厦和复杂的交通设施，殊不知城镇化是一项历史过程，伴随着物质和精神文明的进步，产业功能的转变，和住房、教育、工作、就医、资源配置等一系列变化。敏捷也不是简单地叠加众所周知的最佳敏捷实践，如测试驱动开发、迭代、持续化集成，更不可解释为更好、更快地开发产品。敏捷首先将项目开发中的资源优先配置在高价值的需求上，通过增量型的产出和验证使得产品以最快的速度进入市场并成功盈利，进而带来了一系列的沟通、技术、过程、组织的变革。这是一个历史过程，盲目模仿可能会得到最快的结果，但一定不是有生命力的结果。忽视敏捷转型过程，仍然过份强调效率、时间、成本，鱼和熊掌皆想得，甚至从原来一个月一次的发布变成像网页开发一样一天多次发布才甘心，这些都是中国式敏捷好大喜功的表现。过速的敏捷转型、盲目的跟从、不依附客观条件而妄想的敏捷世界让我们又要深思，为什么敏捷，敏捷是否有效，或者敏捷转型如何做到稳健且波澜不惊？

2.2 理论的局限性

2.2 理论的局限性

在我们不断加深对Scrum的理解，并使用敏捷方法改进日常工作和生活的同时，却发现了Scrum在指导研发工作，特别是复杂的软件研发时存在很大的局限性：Scrum涵盖了软件开发生命周期中的构建阶段，却缺乏对团队的初始创建阶段、项目启动、软件即将发布以及软件维护阶段的指导价值。

我们要否定Scrum吗？不！在否定之前，我们还需开放性地问问自己，Scrum真的没有对构建初期、发布阶段的工作产生任何作用吗？我的答案是，Scrum方法的确没有对这些阶段的研发有任何指导，可是我们聪明的大脑非常愿意修复这个缺陷，对于所有方法论没有制定或涵盖的部分，我们的大脑延伸了Scrum方法，尤其是四句敏捷宣言。这就可以解释为什么大家在“没法可依”的情况下，仍然默契地、毫无顾虑地在软件完整生命周期的各个阶段坚守Scrum原则。

当团队有勇气和智慧批判地继承权威理论的时候，无论对与错，都要赞许他们追寻真理的愿望。我非常愿意和团队的“明白人”探讨并将经验和心得贡献给IBM敏捷领导团队，也正是因为我不怕犯错，勇敢地挑战了Ken Schwaber的Scrum方法，后来我才能加入IBM的敏捷领导团队，成为2010年的IBM Scrum Fellow。在之后的一年半时间里，我又参与了IBM敏捷方法论DAD、分布式敏捷挑战以及ASM的研发。

在详细介绍DAD方法之前，通过一个例子来找找Scrum的局限性何在，以及我们将如何应对。

某房地产开发公司相信，盖好的房子，就必须按照客户需求不断改进，掌控好用料和工序先后，把握好现金流量，不冒进不头脑发热。按照敏捷主义原则，公司的价值追求是务必让客户满意，这样可以少花费因不按照需求工作带来的成本消耗。

然而，设计部门对于市场预期不准，对于购房者的分析不细，这会花费大量的时间，而这个结果本身又随着时间和格局的变化而不同。设计部迫切需要设计出大卖的房子，单凭经验判断和大胆预期自然不是最好的解决方案，于是依照Scrum的构建方法，设计部门打算先设计最可靠的部分，例如先占地，将地基和房屋的支撑结构构建出来，剩下的就走一步看一步吧。

同时，开发商也没有充足的资金来支撑整个项目，银行向房地产行业放贷的口子缩紧，只能拿到以往项目1/4的资金。但是，项目不能再等，必须开始。项目资金需要通过其他方式募集，或者等待房子构建完一部分再从购买者那里筹备。

尽管充满挑战，这家创新型的房地产开发公司依然认为最重要的不在于是否遵循了原计划或者不逾期不超支，而在于对客户需求的理解和执行到位，因为如果出了问题，将会自己承担责任。这就是以Scrum为核心。

房地产公司开始建房了，设计部门、工程部门和验收部门一起工作，结构设计、施工、验收遵循一个个迭代来完成（见图2-4）。第一个月结束的时候，市场需求基本清楚；第二个月中，利用第一部分投资完成了地基的打造，并且在四方的土地上树立了4个结实而高大的立柱，这4个立柱将支撑起最好的房子：至少团队是这么认为的。当第二个月的工作完成时，房地产公司拿到了准售合同，他们提前售出20套房子，来这里看房子的客户对地理位置和房屋效果图，以及工地上如火如荼开展的工作很有信心，开发商很快将这部分钱用于第二期的投入。就在这20位房主签约的时候，设计团队从客户们那里了解到，他们不再喜欢老式的平顶建筑，而是喜欢复古的尖顶房。第三个月结束时，这个设计理念很快被第二期投入的团队开发出来了，房屋验收以后已经初具规模，有坚实的“骨骼”和漂亮的“房顶”。不但之前的20位房

主愿意付清剩下部分的尾款，而且吸引来10位新的客户，他们也愿意付1/2房款，并提出了自己对于外墙特色的建议：红色的复古砖墙，但是需要有美丽的花纹，有蔓藤在攀爬一样的效果。终于，最后一个月到了，开发商有了充足的资金完成剩下的工程，和客户之间的沟通让彼此的合作更加顺利，外墙工作完成得非常好也非常迅速，并顺利通过验收，于是有了一个皆大欢喜的结尾。

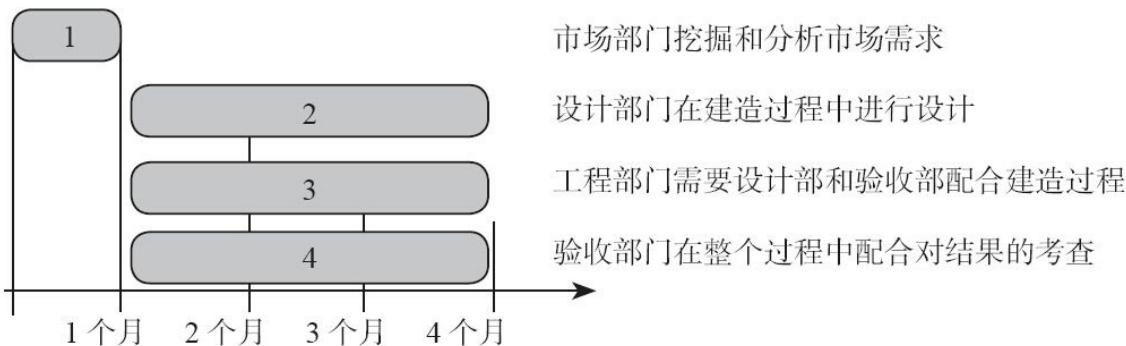


图2-4 使用Scrum建造房子

可是，真的皆大欢喜了吗？不久以后，政府修建的一条高速公路从距离此处100米的位置通过。在高速公路施工期间，房主们纷纷抱怨噪音和光污染，大批住户搬出了住宅。最终，房地产商因为没有提前发现风险并进行规避而损失不小。

事实上，高速公路的修建带来的只是小的损失。当第三个月完成了房顶的构造之后，因为低估了先封顶后建房的技术风险，不得不在第四个月完成外墙修建工作时启用巨型吊车将房顶吊起施工，这个成本是之前没有考虑进去的。突如其来的问题让开发商损失很大。

开发商开始反思，造成损失的原因主要是没有对环境、政治因素进行规划，也没有重视项目中的技术风险。实际项目需要考虑诸多因素，那么如果在开始项目的第一个迭代前，进行初步规划并考虑风险因素，并将其与价值论结合起来，这样的构造方式岂不是更加合理？

其实，IBM敏捷方法DAD的诞生便源自于我们对Scrum和其他核心敏捷方法在复杂环境中的思考和怀疑。这些方法我视为瑰宝，因为让我感动的不仅是方法本身的成功带来的喜悦，更多的是它们见证了一段宝贵的历史。IBM的领导团队心态开放，充分理解和接受新思想，山穷水尽疑无路之时，我们仍不放弃真理，经历了诸多痛苦与挣扎，批判与接受，直到改良落定，柳暗花明又一村。

2.3 传统团队践行

2.3 传统团队践行

迄今为止，社会上有许许多多敏捷培训课程，但是说起敏捷转型，团队既需学习成熟的敏捷方法，又需要资深的敏捷实践者、敏捷教练引导团队循序践行。敏捷的开发者、测试者、业务和需求研发者或者是管理者都需要找到自身的重新定位。事实上，我们在重新构建一套项目研发管理流程和方法，这个过程中，我们或多或少都会有疑惑：项目管理方法PMBOK和敏捷项目管理有什么区别和关系？传统软件研发体系中的项目管理过程和敏捷项目管理过程有什么区别？传统项目的项目经理在敏捷环境中是否需要学习新的技能或面临淘汰？

简单地说，敏捷方法的一切努力都基于一个目标：节省研发成本，降低运维费用，提升利润点。敏捷方法因此才得以推广。需要长期的投入过程、由厚重而具体的计划驱动的传统项目研发方式很难适应变化的市场，因为，再好的设计也会过期，也需要发生变化，计划投入成本的首要考虑是节省。对此，传统项目经理并不会站在对立面，面对新模式，管理者需要做的仅仅是将项目的庞大计划拆分成细小的迭代计划，并且认可调整期待目标的必要性。

2.3.1 挑战传统管理

传统项目经理的华丽转型还需要一点点“破茧而出”的勇气。在敏捷转型完成之前，需要传统项目经理一步步完成自适应的转型，将原来惯于为团队做出裁决、管理团队的高姿态慢慢放下，从“管理”过渡到“支持”，放弃约束和命令，迎合和培养一支能够自适应、自我管理的团队。

2.3.2 PMBOK理论

敏捷和计划驱动的开发方式都符合“项目的金三角”理论（见图2-5），也就是成本、进度和范围的三重约束。但是，传统项目基于的是计划驱动的需求，因为只有这样，进度和成本才可以估算，风险才可以控制。而敏捷方法的项目范围总是在不断变化，即使客户没有要求这个范围发生必然变化，我们也不得不做好变化的准备。在敏捷方法中，由于范围的灵活性，进度和成本不可能保持固定。如果假设进度或成本不变，则这个范围的变化一定是动态平衡的，即变化后又回到了原点。总之，规律谁也逃不过，因为，只有符合规律的项目才不会成为“死亡竞赛”。

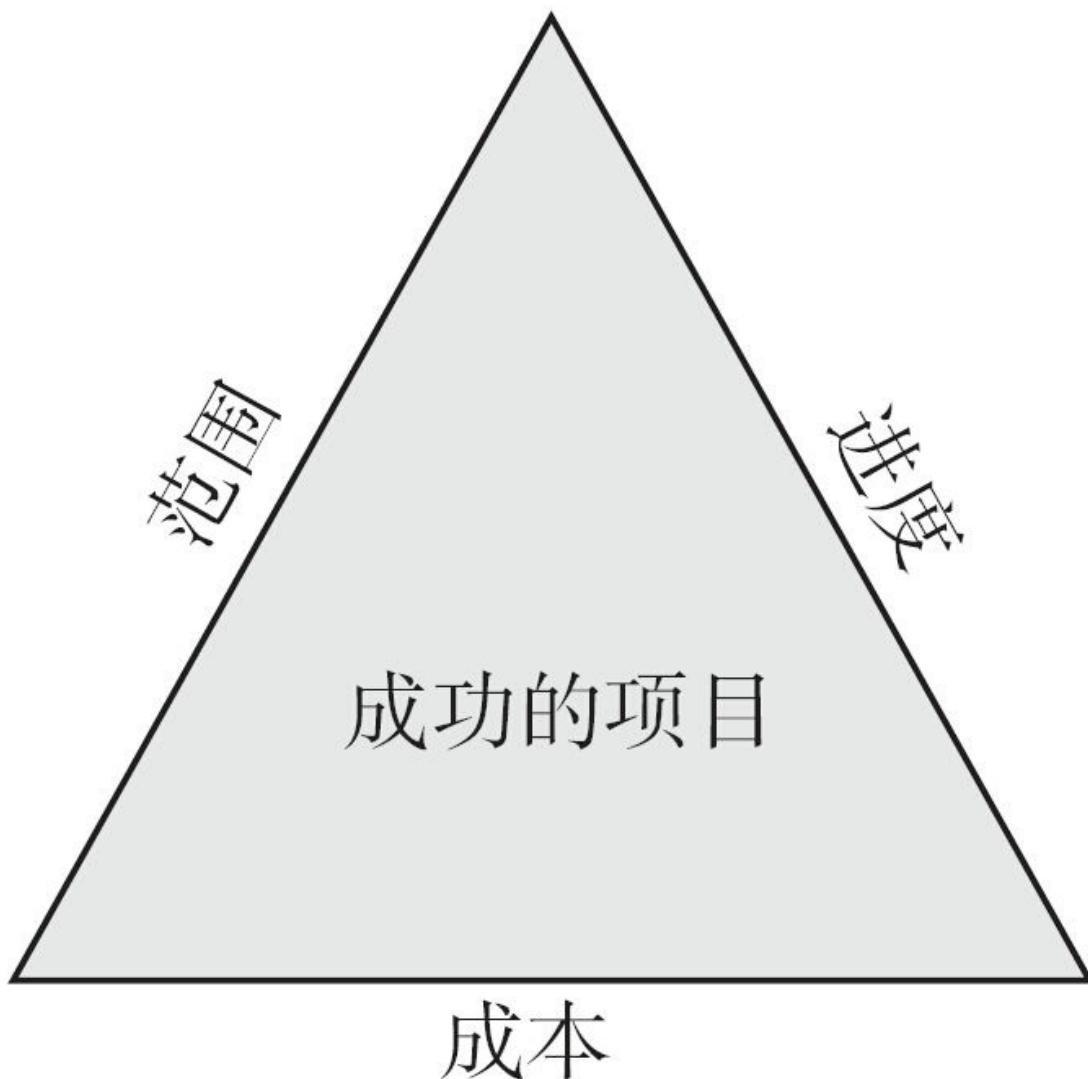


图2-5 项目的金三角约束理论

但是，现实中让项目经理头疼的事情是，利益关系人和客户压根就不买金三角理论的账，项目经理的责任不再是做出独立决策，而是更好地配合持续改进，配合团队在更短的时间内产出更多的价值。用三角理论的方式讲，就是保证范围扩大的前提下，不对进度和成本进行修改。

为了突破金三角的诅咒，项目经理、团队、利益关系人必须做好一件事，那就是提升团队的责任感，增加团队的自我管理空间，界定项目经理的工作，不需要项目经理对太多的事情进行控制，以更好地适应自我管理的团队环境，这也正是核心敏捷原则之一。

项目经理的角色需要从管家型“控制”转变成仆人式“领导”，重点在于促进和合作。下面通过PMBOK知识领域映射到敏捷实践方式，讨论项目经理应该有怎样的专业变化，以及如何使之自然过渡到敏捷软件开发方法。

项目经理要善于观察，并学习如何引导团队在全新环境中可靠地应对变革，而不是强迫团队符合规划的安排。这需要有勇气让团队自己做出决定，而不是被告知该怎么做。这意味着团队成员有了更多的个人责任，并要求项目经理有更强大的协调和配合技能。在开始阶段，最难的就是让项目经理适应新角色。项目经理的责任是培养团队凝聚力，营造良好的团队沟通氛围。虽然并不是每个人都愿意做出转变，但是，我看到了许多项目经理的成功转变，其努力也让团队收获了巨大的回报。而任何时候，我们都必须清楚，总有人不喜欢敏捷，总有人拒绝变化，所以不要强求所有人都参与。

有意思的是，在深知项目管理协会（PMI）和敏捷理念有所差异的前提下，我和PMI的资深顾问，CMMI5的认证评审委员会成员进行了多次讨论，发现项目管理知识体系指南（PMBOK）确定的做法都与敏捷实践可以很好地兼容，当然也有不同。

例如，能力成熟度模型（CMMI）指数指出，PMBOK是最佳的实践方针，组织必须行使自由裁剪的权力，如项目管理、需求管理，团队管理等。事实上，敏捷方法，以及我们提出的纪律和风险与价值驱动模型，都将符合CMMI，就像传统的瀑布方法和CMMI兼容一样。如果非要说有什么不同（除了基本的命令、控制与自我管理），那也仅仅是何时以及如何将这些行为可靠地执行，以及敏捷实践中的新词汇罢了。

PMBOK指出，初始化、反复计划、监管、执行和退出是标准的项目管理流程（见图2-6）。而IBM的DAD更能反映真实的软件项目开发过程，其项目管理流程为初始化、反复计划、迭代审视与改进、迭代构造、迭代交付过程（见表2-1）。初始化阶段兼顾考虑商业视野和初步的功能计划列表，确定构造过程中需要做什么。到了反复计划阶段，这个商业视野定义过的先进产品列表将作为一系列可以实现的功能，并被规划到迭代周期中去。迭代构造是产品通过一个增量迭代过程，从无到有的产生过程。迭代审视与改进过程是项目负责人和团队实际完成项目后，对项目的利益关系人做出总结性汇报的过程，当然还有不间断地对项目风险、结果监控和管理的过程。最后是交付和退出，在这个过程中，团队有机会看到客户对工作成果的真实反馈，了解利益关系人对产品的进一步需求。

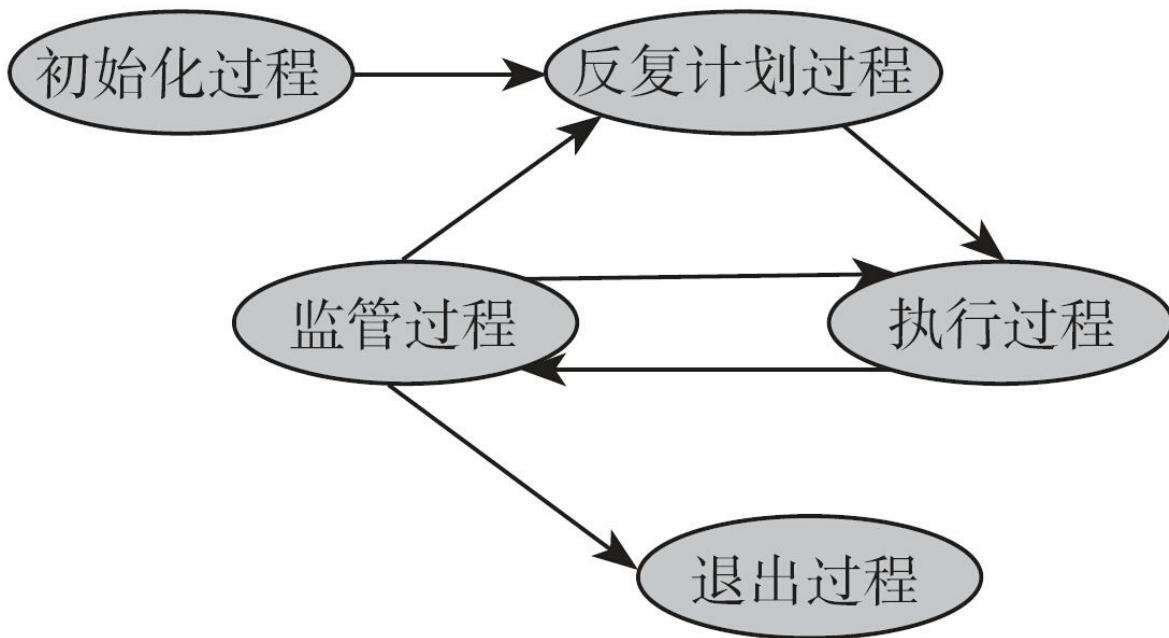


图2-6 PMBOK过程

表2-1 PMBOK与DAD的异同

PMBOK	DAD
初始化	初始化
反复计划	反复计划
监管	迭代审视与改进
执行	迭代构造
退出	交付和退出

阅读PMBOK可以发现，五个主要的知识点在项目管理领域受到了足够的重视，它们是集成管理（Integration Management）、范围管理（Scope Management）、项目时间管理（Project Time Management）、质量管理（Quality Management）和人力资源管理（Human Resource Management）。对于每个知识点，PMBOK所定义的实践活动与敏捷软件研发各阶段的实践活动有着截然不同的定义（见图2-7和图2-8）。

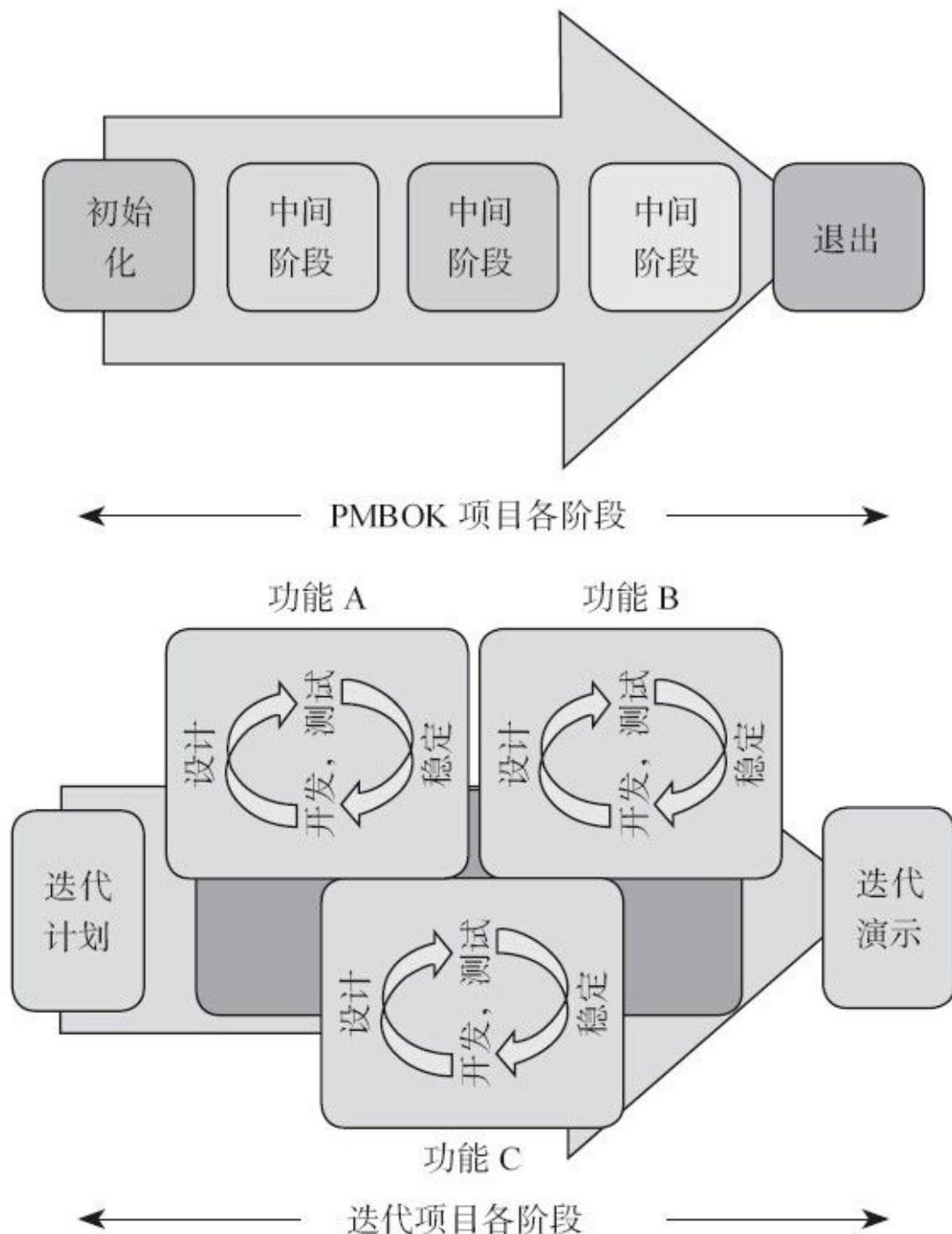


图2-7 PMBOK与敏捷迭代阶段映射

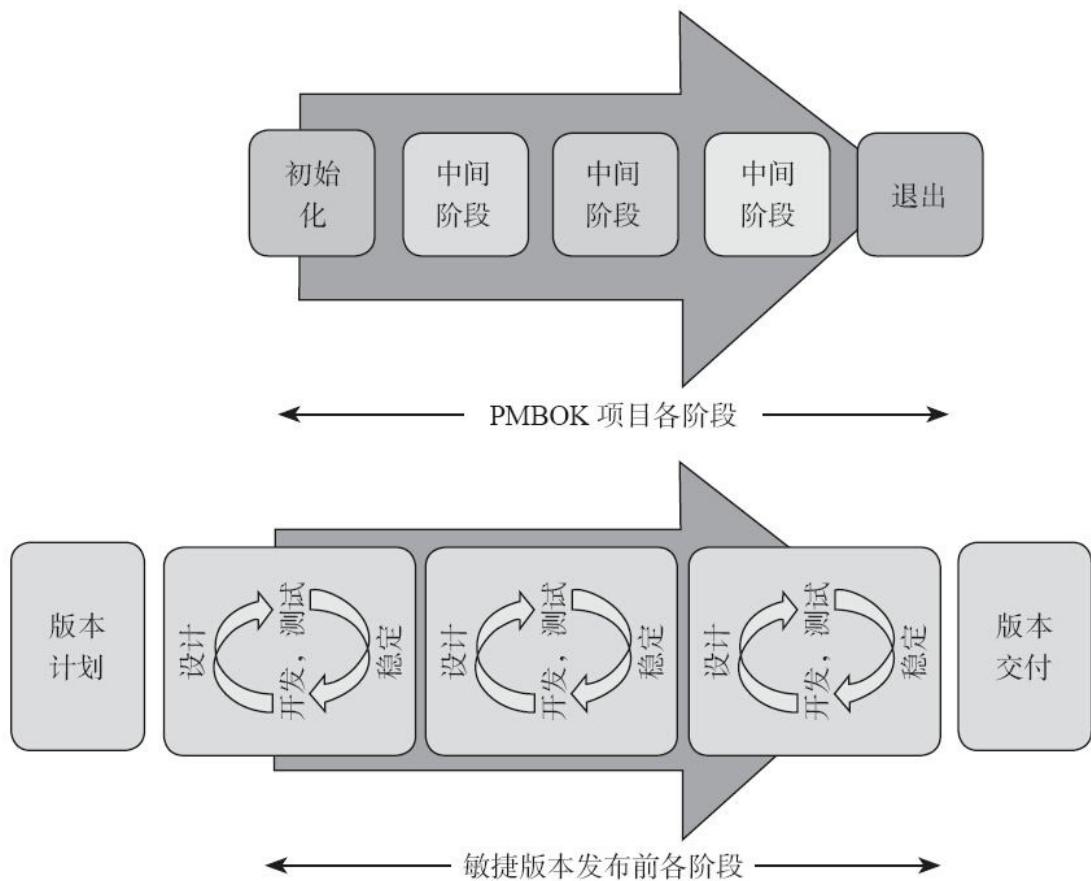


图2-8 PMBOK与敏捷版本发布阶段映射

1. 集成管理vs敏捷多级计划管理

在集成管理领域，最为重要的交付物是项目的计划文档，这个文档由项目经理负责和交付。而敏捷研发过程中，因为尊重仅仅足够和及时的原则，所有的计划和设计都是在团队中历练出来的。计划文档也是有的，但只不过是一个简单的发布计划文档和一些迭代计划文档。敏捷计划文档不会详尽解释范围、工作分解结（WBS）、进度、假设和管制，敏捷项目的项目经理更关注计划的过程而不是结果，因此将轻量级的文档仅仅作为项目团队理解目标，达成一致的手段。为了符合流程上的标准，我们将白板、录音作为“文档”或者“计划”保留项目公共区域里，例如，通过Rational协同工具和工具视图做到“自动化”与“品质”同在（见图2-9）。

再如，只用简单的白板加上有颜色的便签，就可以将计划的关注目标呈现清楚（见图2-10）。这种方法适用于集中的团队开发，在一间封闭开发的房间（war room）将所有重要的资源和信息公开，使大家在行动和思想上达成统一。

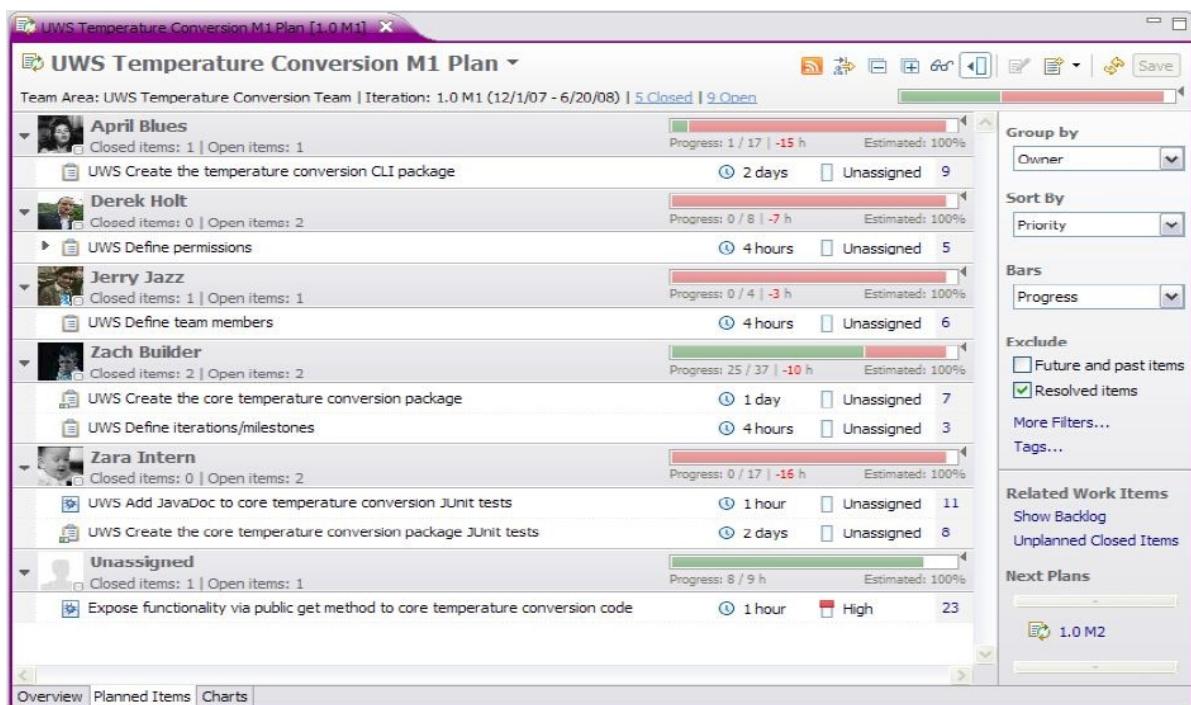


图2-9 使用Rational工具使计划和项目透明化



图2-10 使用白板和各种颜色即时贴做计划和项目跟进

如果既没有足够的资金购进高级工具也没有华丽的war room, 用EXCEL做出简易的产品清单并记录每日团队的工作量, 自动绘制燃尽图, 效果也很好(见图2-11)。

2.PMBOK范围管理和项目时间管理vs敏捷范围管理和时间管理

“规模扩张”一直是传统项目的祸根, 如要求项目经理积极响应客户的业务需求、响应行业的点滴变化、响应突破瓶颈的新技术, 并在开发过程中不断学习和改变, 这样的项目经理一

定要得“变化恐惧症”。在PMBOK中，范围规划、定义、验证和控制都被给予极大关注，并且是计划驱动的，不希望更改的范围变大。不同的是，敏捷方法希望拥抱范围变化，迎接业务需求，响应市场、行业、技术的变革，并对过程和效率有着持续改进的热情。在尽量稳定成本和进度的前提下，努力实施客户定义或者接受的最高价值。

Sprint 5

Sprint start date:		2007年1月3日	Sprint Day
		Day of Week	Day of Month
		Hours	Hours remaining
ID	Sprint Backlog Item	Owner	Estimate (hrs)
ID	Sprint Backlog Item	Owner	Estimate (hrs)
1	No blanks in health table, an icon is required for this case to represent	Austin	10
2	Feedback tracking and verification	Alex	5
3	Hover, infopop, Label and inline help	Frank	20
4	User error, warning and information messages	Frank	10
5	Wireframe refinement and discussion	Matt	20
6	Interaction spec and discussion	Matt	30
7	Wireframes and prototypes	David	30
8	Design specifications	David	20
9	building the table itself implementation	Tao Tao Tian	40
10	scripting the expanding/opening of groups implementation	Tao Tao Tian	15
11	HLD of Heat chart client tier	Tao Tao Tian	10
12	HLD of middle tier of Heat chart client tier	Anne	16
13	refine middle tier from derby to Cache	Anne	24
14	middle tier support for group and database in the same level in health	Anne	20
15	mock data refine	Anne	10

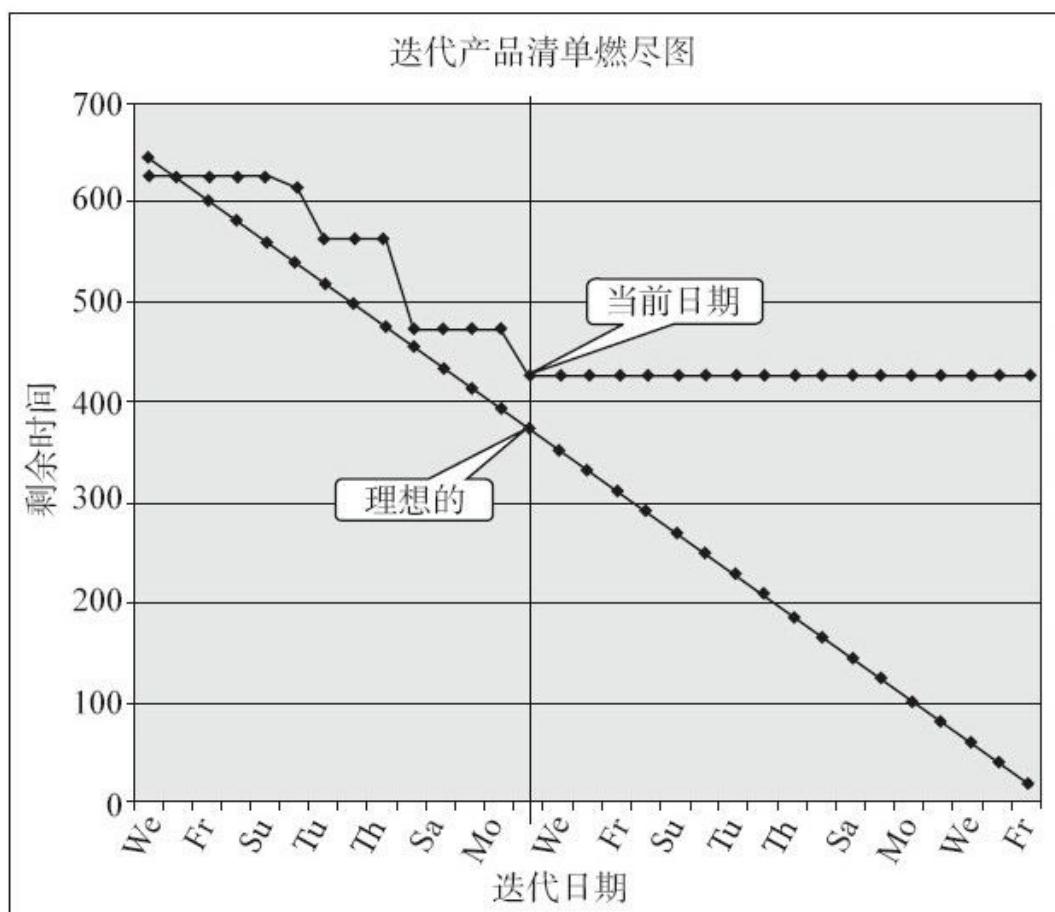


图2-11 使用EXCEL做计划和项目跟进

项目的范围变更和时间管理常常使项目经理陷入慌乱。假设项目是由一个个固定的时间周期——时间盒子构成的，失败的项目管理便是将堆砌功能的“砖头”一个接一个地扔到单一脆弱的时间盒子里，直到时间盒子爆炸。实际上，我们往往会关闭这个对话框，让迭代发生，并生产另一个时间盒子，将“砖头”扔进新的盒子里。由于我们一次只做一个盒子的装载，极致地完成一个盒子的工作，因此我们很难预计在未来的一段时间里能够完成多少工作。事实上，项目经理经常会疑惑：用这种新的、灵活的方法来规划和管理范围听起来很不错，但问题是，当老板和客户问我需要多少时间来完成项目的时候，我该如何回答？

对于时间管理和范围管理，不管使用什么办法，当被问到长远目标或计划时，项目经理往往要扮演先知和算命者。项目经理不可能真正准确地回答这个问题，即使使用传统的项目管理方法，也不过是基于专家判断和历史分析的猜测罢了。罗恩·杰弗里斯，敏捷和极限编程的共同创始人之一，曾告诉项目经理可以这样回答这个问题：

“现在，这似乎是一个200点工作量的项目。根据我们在其他项目上的工作速度和效率（或随机猜测），考虑N位全职程序员的团队规模，并考虑您将亲自参与这个项目，这种规模的项目将需要4~6个月的时间。然而，我们将每两个星期交付一版软件，并以“您满意为止”作为这些功能实现的标准。一个好消息是，如果感到不满意，您可以让团队停下来。还有个更好的消息，如果在所有预计的功能完成之前就已经非常满意，您也可以让我们停下来。而您也有更多的责任，就是与我们积极合作，让团队理解您满意的是什么。最后，也是最好的消息，只要现有功能足以使程序看起来非常有用，您可以要求我们随时准备软件的部署。在项目进展的同时，我们将看到交付效率，且对项目时间估计的准确度也会提高。”

在敏捷软件开发中，范围规划是作为发布计划（Release Planning）的一部分完成的。项目工作范围的定义，和项目时间管理领域中定义的实践活动都是敏捷开发领域迭代计划（Iteration Planning）的一部分。迭代计划对其中的每个功能进行阐述、优先级设定、工作量估计和工作分发。敏捷软件开发与PMBOK定义的关键区别在于，规划和设计工作只是针对独立工作的一个功能或者一段完整代码而言，并不是针对整个系统上某个层面的技术、架构、文档和函数。对应于PMBOK的定义，敏捷开发也有“范围核实”，在迭代结束时，由产品的利益关系人/客户进行审查，用户测试/体验通过即代表工作完成。

3.PMBOK质量管理vs.敏捷质量管理

另一个发生了巨大思想变革的团体是软件研发团队的质量评估（QA）和测试人员。习惯了根据开发者提供的设计文档和详细说明来验证产品正确性的过程，就仿佛置身于无人驾驶飞机的机舱，只能通过感官感受飞机的状态而无所作为。在PMBOK定义下的QA和测试人员所期望的测试是对所交付产品的不规范进行校正，在等待产品交付的时间中陷入焦虑，而在产品交付的最后紧要关头冲锋陷阵。

敏捷测试和质量评估在软件周期时间表上开始渐渐左倾。质量评估和测试人员从产品的需求分析阶段就已经开始参与，一直保持充沛的精力和积极的影响，参与决策整个产品生命周期。由于采取增量开发方式，在每个迭代中，质量评估都在集中测试当下最为重要的可交付功能和有价值的非功能点，而不是等待某事务被“扔过围墙”而开始工作。而且，随着迭代工作和产品功能的累积，质量评估需要不断寻找更为高效和便捷的技术，将自身从低价值的重复劳作中解放出来，关注最有价值和新价值的质量保障工作。因此，必须在GUI、API、系统集成等方面研发自动化测试。

通常情况下，PMBOK要求的质量管理计划应该包括：定义质量保障的目标和范围，确定执行标准质量保证工作的技术和方法。敏捷的质量保障团队成员仍应满足这一要求，并与项目团队一起工作，确定使用的工具和技术，以书面形式提交测试结果。同开发人员一起验证定义并确定执行目标是很重要的，因为单元测试将有助于自动化回归和验收测试。产品负责人也必须参与，因为他们应该帮助团队明确定义并运行验收测试。在敏捷实践中，每个人都责任定义、维护和提高产品的质量保障体系。

质量保证的重点之一是预防缺陷。在敏捷研发中，QA团队与开发团队有着同样重要的决策影响，QA对质量保障工作的承诺在整个项目生命周期中起着至关重要的作用，关乎发布的成败。在项目前期，QA对需求的充分分析可以帮助开发人员编写更好的代码，更多的假设场景被程序员和测试人员作为后续开发、测试计划的有益参考。测试人员站在用户体验的角度，为产品负责人对整个产品的准确规划和定位提供更有效的保证。

质量保证的另一个重点是找到质量缺陷，通过与开发者和程序员的合作将这些缺陷消除掉。敏捷研发中，这类工作也是在迭代内完成的。与PMBOK中定义的缺陷发现和消除工作一样，一般团队会使用自动构建、冒烟测试、自动化回归测试、单元测试、功能性测试、探索性测试、验收测试来完成对缺陷的消除工作。不同的是，敏捷研发中，质量缺陷的发现和消除工作人人有责，没有人能够免于因为缺陷而未能满足客户期望而产生的问责。

4.PMBOK人力资源管理vs.敏捷人力资源管理

一旦收购完毕，公司需要对新团队和每位新员工做出组织规划和团队发展计划，这也正是PMBOK定义的组织规划和团队发展过程。PMBOK组织规划明确指出，需要为员工定义角色与职责，团队发展就是要发展个人技能和个人竞争力（PMI, 2000）。在敏捷领域，团队发展计划的重点是组织一个跨职能团队，使其能够在各种环境下正常运转，实现团队自我管理。

建立跨职能的团队意味着开发团队不再有“门派”区分，而只有角色区分。敏捷开发团队的目标清晰，即创建可工作代码，增量并产生可用的产品。这个过程需要所有关键角色：测试人员、分析师、架构师、技术信息开发者、行业专家、客户/产品负责人，以及团队领导（项目经理，团队负责人等）。这些人具备不同的特定技能，但作为团队一员，以及后续成熟的敏捷团队发展计划的一员，每个人均会更多地了解他人的工作并理解他人的努力，逐渐变得更愿意帮助和自己不同角色的人，进而拥有多项专长。拥有一个跨职能的团队，意味着有一群人致力于让客户满意，并能够通过互助和协作来实现目标。

敏捷团队能够完成自组织的转变，归因于对于团队目标的共识，和持续地、规律地改进产品和产品研发过程。在这个过程中，团队合作紧密，并坚持不懈地寻找解决问题的方案和完成目标的途径。团队负责总计划、总执行，以及审视产品研发和过程改进，自我导向地将团队不断推向新水准。

对于项目经理而言，这又是一个艰难的思想转变，当项目经理认识到不再需要指挥一个团队时，他们开始紧张焦虑，不清楚自己的角色将会怎样变化。实际上，在敏捷团队中，项目经理将从“指挥官”变成以支撑团队为主要工作的“仆人领袖”。团队机制不会自行发生转变，尤其在团队还未熟悉敏捷方法时，需要强有力的领导帮助他们学习如何做出“团队决策”。作为一个“仆人领袖”，需要学习如何促进协作，培养团队持续反思的习惯，帮助团队做得更好。

2.4 敏捷实施法则13条

2.4 敏捷实施法则13条

定义最佳实践需要先明确环境，即明确什么样的组织在什么流程下如何有纪律地交付后，讨论“最佳”或者“误区”才有意义。下面基于IBM的实践和经验教训来分享敏捷实施的13条经验。

1. 及时清理产品工作清单

当团队持续更新产品的工作清单（product backlog）时，产品负责人和团队负责人需要密切关注列表项目中高优先级的工作，并使其比重始终处于一个健康的范围（1：10~1：3）。在迭代计划阶段，需要审阅产品列表，并根据经验和承受能力调整产品清单数量。

根据收集到的需求、用户的最新反馈和经验知识，我们可能决定把一些对当前版本没有附加价值的用户故事从产品清单中删掉。或者，有些用户故事经过重新审阅，尽管对当前版本有意义，但因为不那么紧急，产品负责人仍然可以将这些故事和任务的优先级降低。

产品清单不应该成为一个储藏间，将所有可能会穿但又不想现在尝试的新旧衣服全部堆进去。当产品负责人将一个优先级不高的故事放进产品列表时，必须有充足的理由。

其实，产品清单中用户故事的数量将与这个敏捷团队的生产能力、承受能力密切相关。累赘的产品清单会对团队积极性产生负面影响，进而压抑生产力。当工作趋于轻松、团队趋于稳定时，产品负责人可以加入几个新的用户故事；但当团队的需求产生了比较大的变化、团队工作节奏紧张时，产品列表最好仅仅保留一两个月的工作量，而且，请果断地将不适合留下的工作项从产品列表里删掉，如果它真的是个好故事，你一定会在其他场合再次听到，到时候再把它放回来吧。

2. 迭代规划会议不宜超过1小时

IBM有很多组织迭代计划会议的方式。有些团队会每个迭代组织一次独立的会议，一次性规划好所有迭代任务；还有些团队会每周组织一次会议，或者随时需要随时会晤，清理产品列表并制定迭代计划。后者的优势是让团队经过每周的必经日程，很快养成适宜的工作节奏，减少因调整用户故事大小、细分任务所带来的不可避免的打扰。

规划会议的时长不宜太长，人往往只有40~60分钟的关注度，所以，应该给每次规划会议一个时间限制，保障团队对规划事宜的专注。Scrum的先驱者之一Peter Deemer有一个普遍的建议，就是团队使用5%~10%的迭代长度来设定迭代规划时间。例如，周期为一个月的迭代规划会议长度加起来不要超过1天。

IBM在柏林的STG（System Technology Group）团队每个迭代只会晤一次，用来决策迭代规划：“我们每两个星期有一次规划会议，这样使得规划会议的时间比较短。我们的审查和规划会议在隔周周一举行。在周三之前，有一个可以称为“设计”的会议，但是它不是真正的设计会议。我们聚在一起，打碎用户故事，架构师也会参加这次会议，但他们并不总是参与迭代规划会议。”

HongXue Han，一位杰出的IBM STG的研发工程师，也提及了有关规划和“设计”的会议：“在我们的工作环境中，团队负责人与产品负责人将决策要在迭代中实现什么目标，以及要做哪些工作。然后，他们会召集团队开上一个设计会议，产品负责人将向团队介绍他们想要做的内容。这个设计会议只是为了产出一个高阶设计，旨在了解研发阶段可能的外在依赖和技术风险。当团队知道要做的是什么后，团队会坐在一起讨论迭代规划。”

然而，不是所有的团队都有一个独立的会议来确定迭代计划和调整产品任务列表。事实上，我们是否需要坐在一起讨论迭代规划取决于是否有新的任务和项目将要放入或取出产品任务列表。显然，倘若计划被频繁更改，产品负责人经常召集团队参与会议将会严重影响团队的生产力。

3. 迭代规划会议只做该做的

迭代规划会议的目的是让团队思考在下一个迭代里完成什么工作以及怎样完成这项工作，并达成一个共识。整个团队需要为迭代规划，并为之做准备。这需要我们重新梳理当前的产品清单，了解整个团队的工作效率，并整理在迭代之前收集的信息。在IBM，迭代规划会议由两个连续的会议组成，在第一个会议里，团队决定要做什么，包括：

- 讨论产品愿景。
- 讨论并确定哪些用户故事在清单列表里具有最高优先级。
- 确定哪些用户故事需要在这个迭代里面完成。
- 确定迭代的目标。

在另外一个会议里，整个团队要决定怎样完成这些工作，包括：

- 设计工作途径。
- 细分和估计期待完成的工作项（细分粒度要适当）。
- 同每个成员确认能够完成的工作任务。
- 提交到迭代任务列表。

事实上，团队成员的地理位置分布以及团队规模都会对迭代计划产生影响。IBM的项目规划中，因为全部成员通常不在一个时区，两个会议也并不是紧接着进行的。而对于规模较大的团队来说，IBM的迭代规划也是不一样的。因为有更多的成员来开发产品，我们必须考虑到团队之间的依赖，而且应该比规模小且集中的团队有更多的迭代。规模较大的团队应该促进小组之间的深度合作，以及特殊技术方面的交流和共享。产品负责人，甚至产品负责人团队需要在不同小组的产品清单之间协调优先级。只有这样，小组之间才能配合起来，到迭代结束时，团队才能交付一个可以发布的且稳定的产品。

为了有效地完成这个重要会议，IBM的团队在迭代规划之前都会收集和了解如下信息：

·产品负责人应该有一个需求（用户故事）列表，并且按优先级由高到低排序。这个列表会放在团队可以访问的数据库或者“产品清单”的最前面，供整个团队使用。

·理想情况下，用户故事会有一个基本的行文格式：“作为一位……，我需要做……，因为……”，经过迭代规划会议讨论用户故事的内容以及它们在交付时能够被用户或者利益关系人接受的条件。整个团队必须充分理解用户故事。

·用户故事应该有一个相对准确的工作量估计，如果需要，还应该把用户故事工作量拆解成适合在这个迭代内完成的量，再进入迭代会议进行讨论。

充分的准备是平稳、成功的迭代规划会议的基础，能避免令人沮丧且漫长的会议活动。但是，如果不失去上面这些提前准备的要素，也并不意味着世界末日的到来，只是会使规划会议的时间变得更长，而且会给大家增加更多的挑战。

4. 团队在迭代规划中的自我管理

所有成员必须在一起探讨并承诺能够完成的工作，确定团队的工作任务，以及怎样一起完成这些工作。仅仅由一些代表来承诺团队的工作是不合理的。所有成员都应该自己提交各自能够承诺的那部分工作规划。

对于规模较大的团队，确保每一个对工作有责任感的成员都能参加迭代规划会议是很重要的。无论在不在同一地区，当下最重要的就是团队自己能够获得认领工作的全部权利和责任，当团队成员认同并承诺了这部分任务后，就意味着团队要对这项任务负责。

IBM确实有一些实际的例子，不同时区的小组或许会选择一些代表来参加迭代规划会议，但整个小组都需要参加迭代规划。假设有3个组，每个组有9个组员，从一个一般产品的最高优先级的任务列表清单里选取任务。开始，这3个组委任代表在迭代规划会议上提供了最初的工作估计，然后，代表回去和组内成员讨论这些任务是否合理，并确定是否需要修改。所以对于这次迭代规划会议，确实27个小组成员都参与其中了。

对于一些独立的组，每个组还应该遵从各自的迭代规划会议。

5.不同时区组的迭代规划会议

不同时区的组需要找一个整个组都能参加的时间来进行迭代计划（Sprint Planning）。对于不同时区的组，有时候工作时间是没有重叠的，因此，有一些组员就需要在他们的正常工作时间外来参加这个会议。

开启一个Scrum项目后，通过一个大家都能访问的计划工具可以帮助不同时区的同事沟通。每一个参加迭代计划会议的人都可以看到产品清单、用户故事、迭代任务清单和工作量的估计。另外，通过访问这个分布式的工具，不同时区的组还希望有一个桌面共享工具。一般我们还可以指定一个同事每次记录下组内的讨论或者达成的共识，并且更新到计划工具里。

6.与远程团队的会议

当团队人数超过10人时，我们的每日站立会议就会显得很熬人，一人一分钟陈述，即使不被打断，时间就已经近15分钟。所以，我们提出了采用“Scrum of Scrum”的方式组织会议，即每7~9人组成一个Scrum团队，每个团队再选取1~2名代表参加高阶的Scrum会议。

IBM的研发项目中，超过100人的团队非常之多，尤其是分布式环境下，某一地区可能集中了属于某Scrum团队的20名成员，而Scrum会议的形式基本是全团队参与。

位于分布式环境下的大型团队，让来自世界各地的团队成员参与一个会议，尤其是当不同地区之间没有相互交叠的工作时间时，绝非易事，况且，团队还需要接受文化差异、语言障碍带来的挑战。当很多不同国家、地区的同事参与到项目中来，我们要面对的一个大挑战就是寻找一个最佳的会晤时间，并期望这个会议时间是大多数同事愿意的、或者可能愿意的。

分布式会议需要远程会议设备，这种会议的效果通常远不如面对面的会议，因为除了耳朵以外，还需要眼睛、肢体参与沟通。事实上，有几条分布式团队的电话会议实践经验可以分享：

- 至少提前2天发出会议邀请，如果没有会议系统，那么会议时间的格式最好有完整的年、月、日、时、分等。因为，2014年1月7日，在美国是1/7/14，在日本是14/1/7，而在西班牙是7/1/14。所以，如果时间截是“1/7”，那么将会误导与会者。并且，还要在会议前30~120分钟发出会议提醒，提醒大家会议将准时进行。

- 提前5分钟拨打电话会议，这样不会因为你的拨入打断正在进行中的讨论，也会有足够的空间处理可能出现的技术故障。

·在“Yes/No”之后，跟随一个完整的陈述/否定句来表达正确的用意。实际上，许多国家对于“Yes”和“No”有不同的理解。例如，你问题是：“你不希望开发这个功能，是吗？”如果答案真实，且的确不希望开发，那么，通常美国人的回答会是“No, I don't.”而我们中国人会说：“是，我不希望”。所以，在“Yes/No”后面要跟随一个完整的陈述/否定句来清晰表达你的意愿。

·让语言尽量简单，不要使用过长或俚语、成语等丰富的句子。当我们用非母语沟通时，一定不希望对方使用俚语或者某个电视剧上的笑话来活跃气氛。

·给每个人一个表达机会。在会议中，那些外向的同事常常积极发言，迫切成为中心人物；而内向的同事似乎非常适合做总结，我们应该安排这些同事在最后发言。

·如果几个人在一个会议室，使用一台电话拨打进入电话会议，那么，请在他人讲话的时候避免交头接耳。在必须沟通的时候，也请对准话筒，大声说出他的名字，就像他也在电话另一头一样，这样做会让所有与会的人感到被尊重。而在无需发言的时候，尽量将自己的电话静音。

·当你没有听清楚，或者觉得有必要澄清时，请尽可能地提出来：“我的电话出现了些噪音/我的电话网络应该出了问题，能请您重复一遍刚刚提出的观点吗？”

·使用协同工具，分享开发文件，使用对话机制将某一话题的探讨用文字表达出来。事实上，越是分布式、远程对话的团队，文档和文字的沟通越为重要。

7. 处理工作间的依赖关系

开发团队成员需要寻找合理的方式最小化产品列表中用户故事的相互依赖。一个完整的、优秀的用户故事是独立的，无需任何依赖。在复杂领域和分布式团队项目中，去除依赖性更为重要。如果团队在工作上彼此有了依赖性，就很可能需要一些额外时间和精力来协调不同团队的配合。如果处理不当，还很可能因为彼此工作任务的依赖性导致其中一个团队不能够完成计划任务，或工作遇到障碍。

IBM建议将依赖性划分成三类问题来处理（见图2-12）：

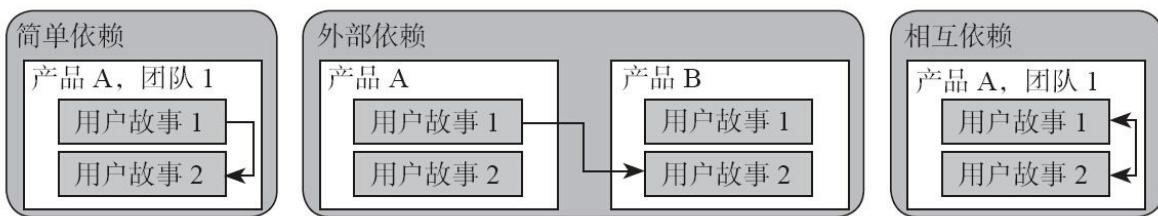


图2-12 三类依赖性问题

- 简单依赖。
- 外部依赖。
- 复合依赖。

简单依赖关系即一个用户故事依赖于另外一个能够独立的用户故事。消除这种单向依赖关系的一种方法是合并或拆分用户故事，使之成为新的用户故事集。如果这么做是不可能的，替代方案是为独立的用户故事设置比依赖于它的用户故事更高的优先级。那么团队将在迭代早期解决独立的用户故事，之后再解决相关的用户故事。

外部依赖关系即一个用户故事依赖于外界团队的工作。在这种情况下，团队需要协商整合和交付日期，并调整该用户故事的优先级。负责该用户故事的小组不应该在外界团队还未

完成其交付的前提下，承诺在当下迭代内完成这个用户故事。在特殊情况下，受依赖影响的团队可以安排虚构的接口，模拟外部组件的数据、数据通信或者Web服务，但这是有风险的。因为系统在真正的外部组件整合进来之前，是潜在不可交付的。外部依赖关系的例子包括：接口、数据馈送和Web服务。另外，在处理外部依赖关系的工作时，团队往往需要在迭代规划前考察和评估外部团队完成工作的能力与时长，这将对我们的迭代规划产生影响。

交织在一起的相互依赖关系即两个或多个用户故事是相互依存的，每个用户故事都不可以独善其身完成而不依靠其他故事的推进。理想的情况下，团队应该想办法打破所有的依赖连接或寻找方法把交织在一起的依赖关系转化为简单依赖关系。实际上，IBM的项目中，产品负责人和团队负责人还有架构负责人肩负了分解复杂依赖关系的责任。

8.有效的团队协作

没有人能独奏交响乐，它需要整个乐团的合作。

——H.E.Luccock

在大规模分布式的企环境境中，对于团队来讲，尽早交付有商业价值的解决方案是极为显著的变化之一。以前，开发团队的成员需要几个月的时间才能完成最后的交付；而现在，他们每两周就要交付出可运行的软件。

产品经理转变为产品负责人，他们每天都要积极地协调利益关系人与开发团队之间的沟通与工作，从而完成每个迭代周期的有价值的软件交付。开发团队、产品负责人、团队负责人以及其他的利益关系人要在整个开发过程中一起认真地考虑产品的商业价值，而不是仅仅在项目的开始和结束。

以前，经理可能会容忍阻碍项目进展的问题持续几天甚至几周的时间；而现在，在每两周的软件交付周期中，这种现象不可能再出现。因为团队负责人和产品负责人提出的问题往往要求他们在24小时之内解决。

所有这些显著的变化促使团队交付的软件可以满足利益关系人最小化失败风险的需求，这个过程也对团队高效协作提出了更高标准。

IBM团队总结出3种非常有价值的实践方法，可促进在迭代周期中的分布式协作，并帮助团队按时完成任务：持续集成（Continuous Integration），自动化测试（Test Automation）和代码审核开发（Code-Review Development）。下一章将会分享一个主人公的故事，通过这个故事详细介绍这3种实践方法，以及它们如何帮助分布式团队快速定位、分析和解决问题，如何提高团队成员之间的协作和效率。

9.用文档克服距离

很少、甚至没有重合工作时间的分布式团队可能需要更多的邮件和其他书面文档来保持成员之间的有效沟通。IBM的团队经验告诉我们，文档能使团队更快达成一致并保持共识。

当团队会议中某一种语言不是所有成员的母语时，书面语言也许是更加可靠和精准的沟通方式。非母语发言人更容易理解的是书面语言，这也是为什么我们应在口头讨论中加入一些文本沟通方式作为补充，如聊天工具或文档。

很多团队会使用在线聊天工具或者屏幕共享工具来记录、分享会议的讨论要点。这不仅能保留讨论内容，也有助于非母语成员更好地理解和掌握要点。随着会议的深入，团队负责人应该以文本方式记录下所有要点，并试着问自己：

·我是否精确地记录了大家关心的问题？

·我是否精准地掌握了大家将要做的事情？

10. 使用合适的工具

Scrum更加重视个人、人与人的协作，而不是流程和工具。尽管工具可以帮助团队完成任务，但它不能创建一个高效的团队。在分布式环境中，虽然工具和有效的实践方法可以帮助团队成员更有效地沟通，但更重要的是要确保引入的工具可以帮助大家完成必须的工作。

比如，一个分布式团队正考虑在日常迭代会议中引入分布式协作工具，首先要评估工具带来的价值与花费的合理性。使用虚拟会议室进行日常Scrum会议虽然看起来很酷，但如果进入会议室都要花上几分钟，那么它也就没有什么使用价值。因为，通常的日常Scrum会议只有15分钟左右。一个简单的电话会议就足以解决问题，为什么还要寻求那么昂贵的方式呢？

所以，设计一种最合适团队协作的敏捷开发工具，需要从活动价值出发，以人为本地进行工作效率改进，将价值作为灵魂，将方法框架融入工具，延展工具的功能，特别是提供定制和扩展的空间，使得工具既能够满足当下工作的需求，保障团队的敏捷性，又可以整合敏捷研发全生命周期中各个有价值的活动交付过程，充分体现项目开发的透明度。

总之，关注的重点应在DAD的核心原则上，千万不要因为拥有工具而一定要使用它们。

11. 重视团队精神

所有的团队成员，包括产品负责人、利益关系人，通过协作共同交付一个产品或解决方案。每一个团队都应是一个完整的多功能团队。这些团队应该分别工作在尽可能多的、各自独立的产品功能点上，而不是产品的组件上。从而可以最小化依赖关系以避免开发进程拖延的风险。

在一个全球化分布式开发环境中，由于沟通的延迟，团队成员无法在同一时间段工作，容易出现状况。这使得大家不得不更加努力地工作以防止出现“我们”与“他们”的观点，造成不必要的分歧。为此，可以尝试问自己一些问题：

- 你是否联系过在不同国家、不同时区的同事？
- 你是否尝试过通过电话交流，而不仅仅依赖延迟的邮件？
- 假如你是一个相对集中分布的小组中的一员，而整个团队还包含一些分布在其他地区的成员，你是否尝试过邀请这些“边缘”成员加入到即将举行的临时会议中？
- 在你决议要缺席某一天的会议时，你是否提前给其他组员提供了有用的最新信息呢？
- 由于时区的问题，你是否让在另一个时区工作的同事总是留到很晚？
- 休息时候的临时讨论，团队的其他成员是否参与进来了？
- 你采取何种措施来验证团队成员是否如预期地掌握了交流中的要点？
- 当谈及到在不同地区的团队成员时，在语言上，是否有使用“其他国家的团队”或者“其他团队”等词汇的习惯呢？
- 是否给团队成员贴过标签（如“在罗马的成员经常迟到”或者“他们从来都没做对过”）？
- 是否不同地区组员之间的依赖关系过于紧密，以至于每天交流过于频繁？

12. 缩短迭代周期

如果利益关系人不断地催促团队并经常打断团队工作，一个比较好的解决办法是缩短迭代周期。在一个周期为4周的迭代中，利益关系人需要平均等待2周，团队才能完成他们的需求（如果这个需求优先级足够高的话）。而在一个周期为2周的迭代中，利益关系人平均只需

要等待1周的时间。

沟通方面的不同步可能会致使分布式团队需要一个稍长的迭代周期。这显然降低了工作效率。而较短的开发周期可以使团队成员把每一次的交流沟通都当成紧要事情处理。

开发周期越短，团队就有越多机会改进整个开发流程，并迅速适应。较短的开发周期同样意味着当利益关系人增加新的需求时，团队完成任务所需的时间更少。

总之，较短的迭代周期不仅能使团队更迅捷的响应利益关系人、产品负责人的需求，还能使团队更迅速地适应变化。

团队应该尽量找到符合自身的迭代周期。当周期过长时，会妨碍讨论新的高优先级任务；当周期过短时，除了成员间的任务说明之外几乎做不了其他事情。选择时最好考虑不产生需求变化的最长时间，即能够保护团队迭代工作不被打扰的时间。

13.思而不学则殆

有些敏捷理论犹如风筝一般，表面高大上，实际却脱离实际，难以自圆其说。因为没有足够支撑其理论的经验和实践，这种会议、课程、咨询活动常常会陷入彼此争论的苦战，浪费了大量时间。记得常有朋友抱怨一些“教练式”的培训，培训师的思想“高度”远远超出了企业、团队能够接受的水平，痛斥组织之臃肿腐败、抨击人性之不古，内容理想有余，但不接地气。一段时间下来，咨询、培训都无法顺利进行。于是，培训师就开始指责和推诿，认为接受培训或咨询的人员素质不高，领导支持也远远不足，所以只能做成现在这个样子……

其实，很多培训师掌握了“足够”的敏捷知识，但是，敏捷培训师更要具备多年的实践经验，不仅做过敏捷开发、测试、业务分析、需求设计、团队管理等，最好也做过传统的研发和管理工作，厚积而薄发才是培训师的根本。而对于希望成为培训师的朋友而言，即使目前没有这么多的经历，至少要在一线技术岗位奋战过，拥有一定的知识并通过不断的学习和实践来累积经验。

关于知识和经验，心理学认为，人的大脑不善于对二者进行区分。换句话说，读书、学习是非常好的事情，但是如果我们没有经历过，学习也未必会有成效，甚至会因为学习阻碍了我们的成长。

举个例子，假设我们在阅读一本书，并且从书中获得了知识。之后，当我们看另外一本书时，虽然文笔不同，但是内容几乎相同，于是我们会这样说：“这些我明白，在另一本书里，谁谁说过……”甚至还会说：“这本书写的什么嘛，不怎么样啊，其实我告诉你是这样的……”无论是否亲身经历过，可能都会有这种反应，这恐怕就是大脑制造的麻烦。我们通过阅读、学习获得了知识，但是却使我们失去了成长的机会，听起来真是讽刺。读书太多，知识和经验的平衡就将被打破（见图2-13）。因此，有一种智慧会要求我们通过增长经验或者输出知识，即“倒空”的方式来重新建立平衡。正如罗曼·罗兰在《约翰·克利斯朵夫》写道：“我们从小到现在被各种谎言灌满了，当他成熟起来的第一个标志就是他要呕吐，重新用理性去认识世界。”



图2-13 知识和经验的失衡

对于敏捷培训师而言，在知识和经验失衡的情况下，培训者的感受恐怕就会像上面那个朋友抱怨的一样了，最终，也只是浪费了公司的时间和金钱。

第3章 大企业的敏捷转型

第3章 大企业的敏捷转型

- 3.1 重新诠释敏捷宣言
- 3.2 新敏捷扩展模型
- 3.3 持续改进转型模型
- 3.4 案例分析：IBM Rational中国的敏捷转型
- 3.5 故事分享：Rational中国敏捷开发者的一天

哈佛商学院教授罗布·奥斯汀曾提出“可靠的创新”，意指致力于为客户提供价值，以创造满足如今客户需求的产品和持续创新流程的力量。创新力量不会在结构化的、独裁的环境中萌芽，而是在适应的文化中，在自我组织和自我约束的原则下产生。可靠的创新不是颠覆性地迈向生死边缘，而是步步为营。

秉承“可靠”理念，为了对产品和企业的变化做出迅速反应，为了提高业务的增长和盈利能力，为了满足未来客户的需求，更为了“更好地生存下去”，我们不需要颠覆性的试错，而是需要可靠的转型，成为具备生存能力的成功敏捷企业。

3.1 重新诠释敏捷宣言

3.1 重新诠释敏捷宣言

2010年，Agile@IBM敏捷社区的讨论热点集中在“软件全生命周期敏捷”和“分布式敏捷面临的挑战”这两个话题，使我对敏捷的价值和敏捷宣言又有了新的理解，也促使我放下对“核心敏捷”的纠缠，以开放的心态研究IBM内部不尽相同的敏捷开发过程。

在此，请大家再次回顾敏捷宣言的四条要义：

个人与交互	重于	开发过程和工具
可用的软件	重于	复杂的文档
寻求客户的合作	重于	对合同的谈判
对变化的响应	重于	遵循固定的计划

我们用敏捷宣言的四条要义来诠释Scrum方法，同时，用Scrum方法中的实践、策略来反思这四句“语法”不甚完整的宣言，通过新的理解得到了12条“敏捷原则”：

- 优先级最高的目标是通过尽早地、持续地交付有价值的软件使客户满意。
- 拥抱需求变更，甚至在开发后期，利用变化为客户提供竞争优势。
- 经常交付可以工作的软件，从几个星期到几个月，周期尽可能短。
- 业务人员和开发人员必须每天一起工作。
- 以人为本，激励个人，同时给予他们所需要的环境和支持，并且信任他们能够完成工作。
- 团队内部最有效的沟通方法是面对面的交谈。
- “可工作的软件交付”是首要的项目成功度量方式。
- 敏捷项目必须注重团队的“可持续性”。投资人、开发者和用户应该能够理解并支撑团队保持一个可持续发展的工作强度。
- 持续关注技能的卓越化，优秀的设计会增强团队敏捷能力。
- 简约是最大化生产力的秘密。
- 最好的构架、需求和设计出自于“自组织团队”。
- 每隔一定时间，团队将坐在一起反思如何才能更有效地协作，并做出相应的调整。

这12条原则不但可以诠释敏捷宣言的价值观，而且是对Scrum的坚实支撑。软件工程理论的发展离不开精确的定义，领导团队也的确认为IBM需要一个精确的定义。事实上，业界对于敏捷软件开发一直没有正式的定义，甚至可能永远没有。但IBM团队给出了一个可能有用的工作定义（来自前IBM首席方法论学大师Scott Ambler）：

“敏捷软件开发是由价值驱动的、有规律地生产高品质软件的、具有成本效益和时间观念的方法，也是一种进化性（迭代和增量型）的方法。它是一个高度协作的、有规则有纪律的、自我组织的、软件利益关系人积极参与的方法。通过利益关系人的参与，确保团队了解并重视解决利益关系人不断变化的需求。使用此方法的团队，即敏捷软件开发团队，将基于其面临的独特环境，采用适量的敏捷仪式持续地交付。”

让我们来展开定义中的一些关键概念：

·价值驱动。将每个迭代目标均设定为一个潜在可交付的产品或解决方案，持续地为产品注入真正的价值，并且每次均以可用软件的形式体现出来。

·有规律地生产高品质软件。敏捷主义者必注重质量。他们更喜欢早期进入且反复测试，更为严格的敏捷主义者甚至会采取测试驱动的开发方法，这意味着编写一段测试代码，然后用刚刚足够的生产代码来通过测试，以实现每项功能和构建（然后迭代）。很多敏捷开发者已经采用了“重构”方式，即对代码或者架构做简单的修改，以实现兼容扩展的新功能，而不改变其原有功能。通过采用以上这些以提高质量为目标的方法和技术，敏捷团队比传统团队更有可能提供高品质的软件。

·进化。敏捷的共性是迭代和增量。“迭代”的意思是在原有代码版本的基础上，通过一系列的重复过程增加内容，完成每个构建、每个版本直至项目完成。但是，这并不意味着工作本身是重复的。在迭代过程中可能要做需求分析、测试、编程、设计，这些工作过程相似但内容不同。“增量”是指将新的功能和工作代码添加到最新的构建，直至利益关系人确定有足够的价值并发布产品。

·成本效益和时间观念。敏捷团队希望实现的功能，其优先顺序由他们的利益关系人（或能够代表客户的人）定义。因此，能够最大限度地提高投入产出比（ROI），也正是如此，他们更关注工作的高价值功能，从而提高成本效益。敏捷团队更愿意在每一次迭代中产出可交付的软件（迭代是一个时间盒子，长度通常为2~4周），这样利益关系人可根据已交付的成果来决定何时进行正式版本的交付，从而提高了时效性。短迭代周期也缩短了反馈周期，提高了发现问题的速度（无论成功、失败都是快速的），从而能够勇于正面解决问题，即使有些情况下会显得缺乏足够的能力和经验。

·高度协作。系统构建或项目开发获得成功的决定因素是“个人和他们的协同工作方式”。敏捷团队致力于尽可能有效、紧密地合作，这种特质必须每个人都具备，即使是领导团队也不能例外。

·有规则有纪律。敏捷软件开发需要从业者、团队拥有更为严格的纪律性，这点胜过传统研发方式对团队的要求。

·自我组织。自我组织的团队意味着从事工作的人需要自己做计划，自己估算工作量。

·利益关系人积极参与。敏捷团队与他们的利益关系人密切合作，这其中包括最终用户、最终付费人、企业架构师、支付技术工程师、运营业务人员以及其他相关人员。在IBM内部，我们将利益关系人分为四类：赞助商（投资人），合作伙伴（商业伙伴和合作的人），最终用户，内部使用产品的人。利益关系人或他们的代表（产品负责人，或一个客户驻场代表）将持续、及时地提出意见和做出决策。

·利益关系人不断变化的需求。随着项目的进行，利益关系人面前阶段性地呈现了可以工作的产品的部分功能，即使不够完整，他们也将更好地了解自己想要什么。因此，利益关系人很可能在评审过程中改变初衷和需求。而这种可能的变化源自商业环境的不断变化，或者源自团队任务的优先级调整。

·适量的敏捷仪式。“仪式”是指在项目中使用的流程、规范（方法）。例如，结对编程、文档的审阅、对代码质量的讨论。敏捷方法专注于可交付的产品，但不意味着没有流程和规范，不代表破除“仪式”。敏捷团队仍然生产适量的文档、持有恰当的评论，这和传统项目几乎

没有不同。

·持续地交付。利益关系人显然不关心产品团队如何实现、如何解决问题，他们只关心阶段性交付了什么。特别是，他们总是对能够满足当下需求的解决方案有兴趣，因为这使得他们的投资更加理性、智慧。这种持续交付是以固定时间盒子作为周期，以更高质量作为目标的行为。总之，利益关系人关注的是持续的交付结果，而不是方法和过程。

3.2 新敏捷扩展模型

3.2 新敏捷扩展模型

IBM的敏捷实践始于小的、集中的团队开发模型，自2006年起，IBM中国开发中心开始尝试将敏捷方法用于各种各样的系统，包括但不限于基于Web的应用程序、移动应用程序、胖客户端应用、商业智能（BI）系统、嵌入式软件、生命攸关的系统，甚至是大型主机应用程序。尤其是Rational、Websphere、Tivoli和Lotus这样的大型组织，拥有分布在世界各地的、多达上百人的项目团队，需要在各自特有的文化氛围、高复杂度的环境中实施敏捷开发。总之，IBM的敏捷方式不局限于小型的集中团队，而是广泛适用于分布式团队、大型团队、组织级项目团队。

敏捷扩展模型，（Agile Scaling Model, ASD）的诞生为我们在复杂的业务环境中掌控重点信息提供了极大的帮助。敏捷扩展模型是敏捷实践的有效剪裁，以应对任何规模的系统交付团队及其特有的背景方法。图3-1概述了ASM不同敏捷类别之间的关系和区别。

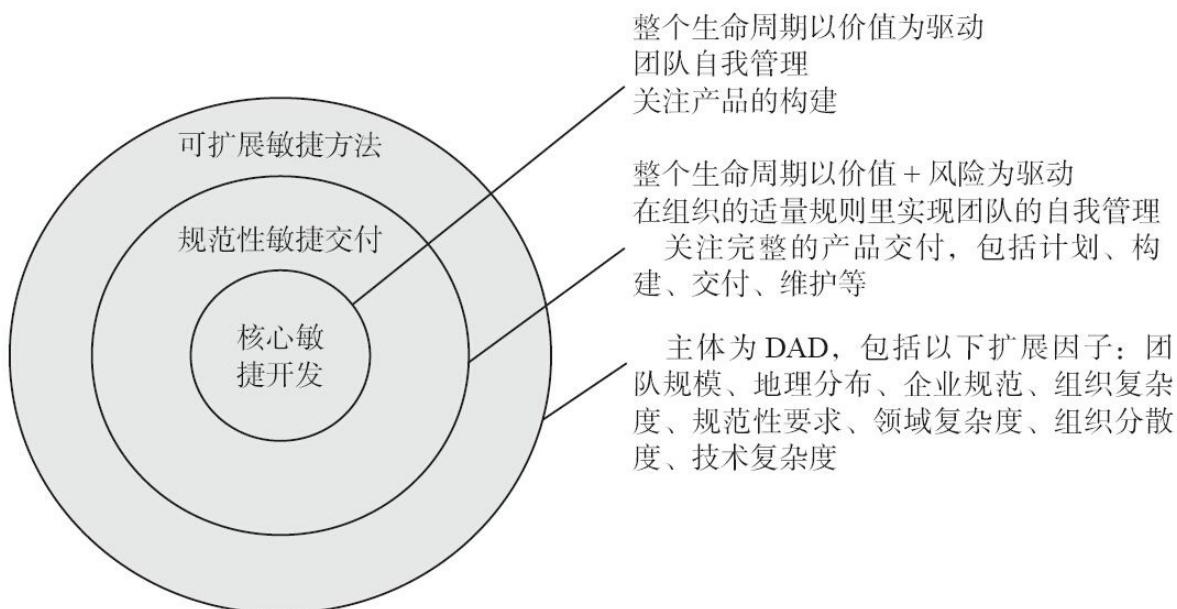


图3-1 ASM框架概述

核心敏捷开发（Core Agile Development）。核心敏捷方法（如Scrum和XP）是自我管理的、以价值为驱动的系统开发方法，解决了开发生命周期中的一部分或某些阶段的问题。这些方法强调具体的实践方式，比如每日站立会议、需求和设计在迭代中细化等，适合小型且集中在同一地点的团队和简单的系统开发及优化。

规范性敏捷交付（Discipline Agile Delivery）。规范性敏捷交付流程包括Scrum、极限编程（XP）、动态系统开发方法（DSDM）和开放统一过程（OpenUP），覆盖从项目开始直至发布的完整的软件开发生命周期。规范性敏捷交付过程是增加了适当的规则治理框架的自我管理模式，既是风险驱动也是价值驱动的敏捷交付过程。同核心敏捷开发方法相同，规范性敏捷交付也适用于小型协作团队和相对简单直接的系统交付。为了实现全交付生命周期的系统交付，需要将几种核心敏捷方法组合起来，采用一种组合式的新方法（DAD），这个方法还采纳了一些能够反映敏捷哲学的改进型“传统”方法，如前期自上而下的需求分析和先行架构设计。

可拓展敏捷方法（Agile at scale）。可拓展的敏捷方法侧重于当一个或多个扩展因子适用

时，对规范性敏捷交付方法的延伸和调整。共有8个扩展因子，分别是：团队规模、地理分布、规范性需求、领域复杂度、技术复杂度、组织分散度和企业规范。所有扩展因子都有一定的取值范围，一个项目可能需要调整一个或者多个扩展因子，但是很少有项目需要调整所有8个扩展因子。所以，需要根据情况灵活地调整有关因子的比例系数，裁剪规范性敏捷交付方法，解决所面临的特定问题。

3.2.1 规范性敏捷交付方法

“规范性敏捷交付（DAD）方法是一个以人为本、学习型、混合型的敏捷方法，是一个风险驱动、也是价值驱动的全生命周期的符合企业规范的过程。”从DAD的定义中可以看到DAD方法的重要构成元素。

1.以人为本

DAD的团队应该是自我管理、自我意识、自律的。传统开发过程中线性传递工作产品，如需求、分析、设计、测试和开发，无论什么原因，当某一环节成为了瓶颈，就将致使后续工作无法完成，产品无法如期发布。而线性传递正是造成时间和金钱巨大浪费的主要原因，人与人之间的交付、信息传递常常造成误解和缺陷注入。在DAD中，我们鼓励由跨职能的人组成跨职能的团队，团队内部不应该有层次，应该鼓励团队成员发展跨职能技能以满足工作的需求，而不是在他们的专业领域内纵深发展。在组织、团队、项目的规范上建立团队的相互理解，将会提升团队生产力，有效利用资源，减少对繁文缛节的依赖。

因此，敏捷方法淡化以技能名称划分角色，DAD角色可以使用多种技能，其中的五个主要角色是：

利益关系人（Stakeholder）。利益关系人是指被解决方案的结果影响的人。利益关系人可以是直接用户、间接用户、内部使用者以及商业伙伴。利益关系人作为团队中最了解业务的人，将帮助开发团队快速达到目标和做出适时决策。

产品负责人（Product Owner）。产品负责人是“客户的代言人”，他们定义产品的功能，决定产品发布的日期和内容，在和投资人的谈判中，规划产品的投入并对产出负责。产品负责人需要始终保持对市场和来自客户的阶段性需求和反馈的敏感度，根据这些变化对产品所需开发的功能模块进行优先顺序排列。在拥有多位利益关系人的团队中，产品负责人尤其需要统一利益关系人对需求优先顺序、团队目标和产品交付的意见。

团队成员（Team Member）。团队成员专注于实际解决方案中的测试、分析、架构、设计、编程、策划、评估，通常以“仅仅足够”的工作完成任务。注意，并非每个团队成员都会跨越多项技能，但他们属于这些技能的一个子集，并将随着时间的推移，努力获得更多的技能。团队成员在核心敏捷方法中被描述为“开发者”或“程序员”，然而，在DAD中，并不是每一个团队成员都必须写代码。

团队负责人（Team Lead）。团队负责人如果是敏捷教练，将有助于保证团队专注于工作项目，并履行其迭代目标。团队负责人直接对产品负责人负责，向产品负责人做出承诺并参与重要的会议，帮助产品负责人果断决策团队的相关事宜。团队负责人是一个真正的团队领导者，擅长交流，赋予团队能力和时间去优化工作流程，并确保团队拥有完成各项任务的资源。同时，团队负责人要在团队遇到棘手问题时及时介入，制定解决方案并实施，直到问题解决。

架构负责人（Architecture Owner）。架构负责人对产品、解决方案的体系结构做出判断和决策，并促进整体解决方案、产品的构建和改进。架构是项目风险的一个重要来源，架构负责人需要尽力减轻这种风险。但架构负责人并没有亲自实现方案、产品中的架构或解决实际问题，而是促成架构的协调者、决策者。

注意，测试人员和业务分析人员并不在DAD过程的主要角色中。相反，一个“通才”的队

员应该能做多件事情，包括业务分析和测试。一开始，团队中曾经专职测试的成员可以志愿协助业务分析工作，甚至可以做团队负责人（对应Scrum中的角色）。这并不意味着每个人都需要成为全能技术专家，但它意味着团队愿意去学习和掌握任何所缺少的技能。

团队成员应该是“通才”——一个在“一个或多个学科”的专家，且在其他学科上拥有一般知识。因为，“通才”相比“专才”更加愿意与他人密切合作，分享他们的技能和经验，并好奇和热衷于学习新技能。由“通才”组成的团队不依赖于人与人之间的传递工作，相反，他们了解不同技术领域的差异和各自的局限性，所以更加热衷于依靠团队配合完成工作，而不是仅仅关注自身的事情。

2.混合型敏捷

DAD的许多战略和实践是根据两个主流的敏捷方法Scrum和XP，以及OpenUP、RUP、Kanban等其他方法而制定的（见图3-2）。DAD扩展了Scrum的构建阶段，为解决全交付生命周期问题而从多个敏捷和精益方法中抽取实践和规则。许多DAD建议中包括我们熟悉的敏捷活动和实践，如持续集成（CI）、每日协调会议、代码重构等，有些是普遍适用的“先进”做法，但是，由于某种原因，在敏捷方法中一般不强调的“需求构想进化”、“架构构想进化”、“即将发布前的测试”等实践活动在DAD中与其他关键实践活动并驾齐驱，下面仅举几例。

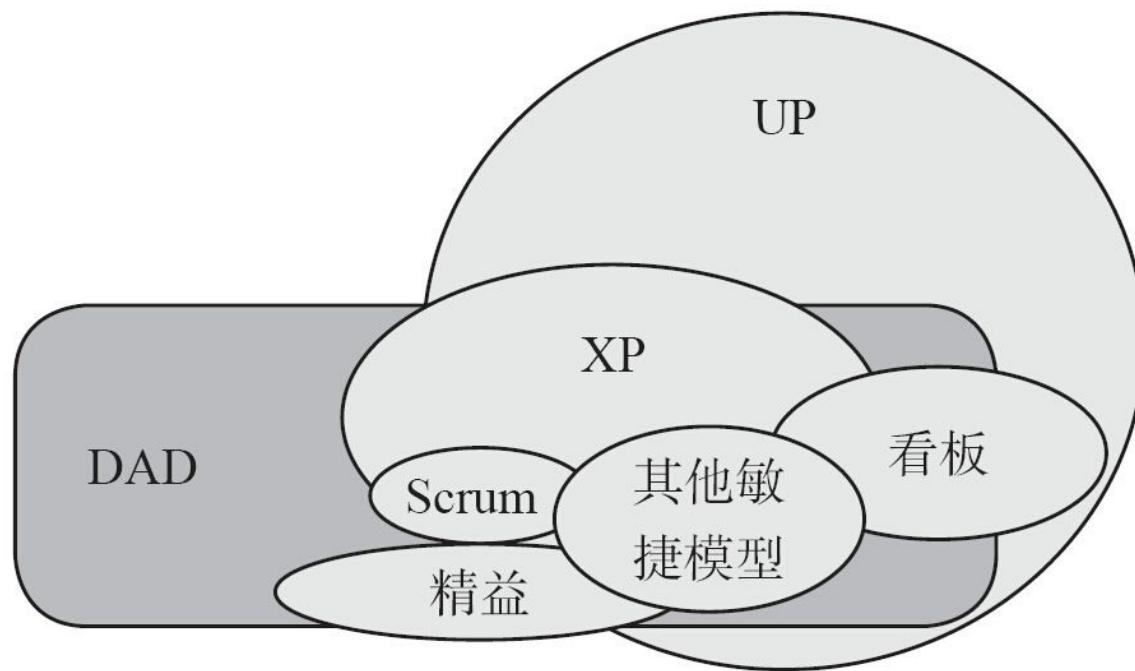


图3-2 DAD是一个混合敏捷方法

Scrum。Scrum的重点是项目领导力和需求管理。DAD采用且裁剪了许多来自Scrum的思路，例如，从堆栈中先提取优先顺序高的工作项；有一个产品负责人代表所有的利益关系人；以及每个迭代都会生产出潜在可发布的产品或者解决方案。然而，DAD放弃了大部分的Scrum的术语，没有Scrum、团队负责人（Scrum Master），也没有迭代（Sprint），只有产品负责人（Product Owner）仍然保留。

极限编程（XP）。XP是DAD开发过程、实践的重要来源，包括但不限于持续集成（CI）、代码重构（Refractory）、测试-驱动开发（TDD）、团队所有制、简单设计、代码规范等。

统一过程（UP）。DAD采用了许多来自UP的敏捷实例，包括OpenUP和Rational统一过程中敏捷特色的实践和项目治理策略，例如，轻量级里程碑和明确的阶段划分，项目的初创

期、构建期和发布期等。DAD同时还使用了降低风险策略，借鉴了UP中的原型和早期迭代架构。

看板（Kanban）和精益（Lean）。DAD采用了两个重要概念，即通过减少看板数量来减少产品的中间存储，通过可视化方式显示产品的库存状态。如果产品库存高，那么即使设备出现故障、不良产品数量增加也不会影响到后续工序的生产，所以容易把问题掩盖起来，延迟处理。而且即使人员过剩也不能解决问题，造成全线停工。看板方法使问题得以立即暴露，从而采取措施及时解决问题，通过改善活动使得生产线的“体制”不断增强，带来生产率的不断提高。看板正是一个精益哲学的方法框架。而精益的七项原则（杜绝浪费、打造精品、加强学习、推迟决策、快速交付、授权团队、优化整体）也注入了DAD的概念。

3.全生命周期过程

DAD延长了Scrum所专注的构建周期，涵盖完整的产品、方案的交付生命周期，从项目开始到发布产品、解决方案或投入生产（或市场）。

DAD关注项目完整的生命周期，从初创期开始实施项目直至解决方案或产品的发布期（见图3-3）。事实上，从一开始直至项目生命周期结束的每一次迭代都是不一样的。项目的工作重点随着时间的推移而发生着清晰的变化。

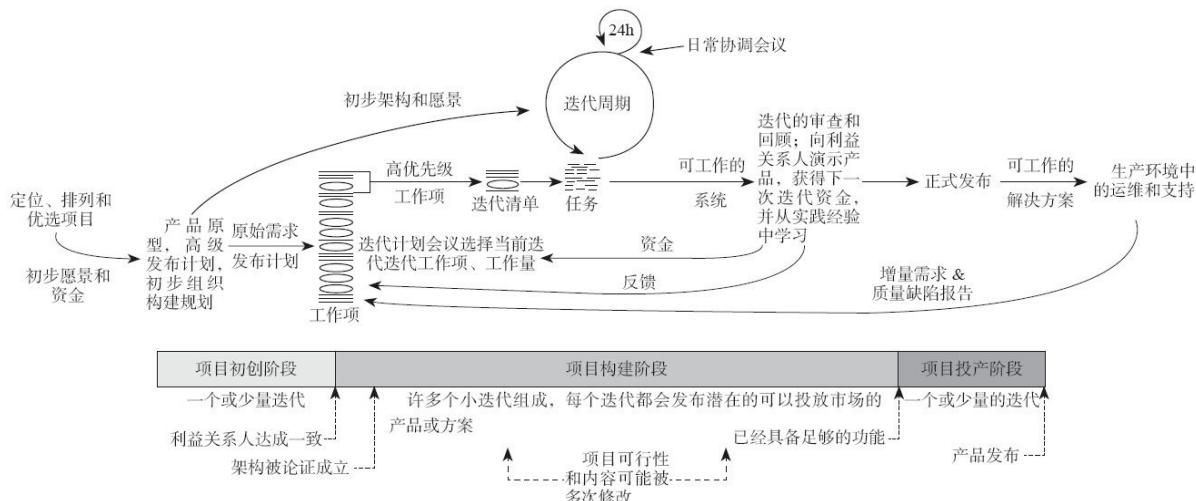


图3-3 DAD过程

为了清楚地描述这一点，我们将项目周期分成多个轻量级的里程碑阶段（见表3-1）以确保我们的重点是在正确的时间做正确的事情，比如最初的愿景是项目规划、建造原型、风险管理以及部署规划。这显然不同于主流敏捷方法，相似的主流敏捷过程发生在项目生命周期中的构建阶段，讨论项目如何开始、产品如何增量及如何发布等。DAD最后还有一个里程碑阶段是关注客户对产品的验收和产品投产。

表3-1 DAD轻量级里程碑及其目标分析

项目初创阶段 (Inception)	项目构建阶段 (Construction)	产品投产阶段 (Transition)
确定项目愿景 确保利益关系人对愿景的一致认同和理解 确定初步的技术实现策略、需求和项目实施计划 组建团队 确保资金到位 风险预估	生产一个有市场的解决方案或产品 满足利益关系人的不断变化的需求 确保产品的质量和稳定性接近可发布的标准 维持或改善质量水平 评估和应对高风险	确保该解决方案、产品可以投产或在客户生产环境中正常工作 确保利益关系人准备好接收将解决方案、产品部署到生产环境中

值得一提的是，DAD强调团队不仅仅需要实现功能需求，还必须修复缺陷（通过独立测试或用户反馈），为其他团队提供支持，或是参加培训课程等。所有这些活动都必须是可见的、可计划的，不只是功能性和非功能性需求。而这些工作项均应该统一存放至一个堆栈，而不是几个栈（可以区分工作项类型）。

4. 符合企业规范

DAD团队由企业组建，是构成企业生态系统的一份子，他们从企业的规范性敏捷开发环境中获取企业的有效资源，把握企业内部系统的各种机会而不断发展。他们与企业的技术架构师和资产重用工程师有着密切的合作，利用并加强现有的技术基础设施构建“未来”的上层技术建筑。企业业务架构师和管理者应该适当地规范团队，监管整个商业生态系统的健康状况；企业数据库管理员、分析员应该能够访问和改进现有的数据来源架构以提升数据库服务效率和吞吐能力；而IT支持人员应当了解并遵守企业的IT原则（如安全性、数据公约等）。换句话说，DAD团队具备胸怀“整个企业”的全局心态。

充分利用企业资产。DAD团队应该充分利用企业资产，或者至少知道有哪些资产，包括公共的代码规范、数据公约、命名规范、安全指南和用户界面标准，但企业的资产实际上远远超过这些。如果使用以资产为中心的方式来构建企业的软件，那么将不难发现，企业维护着一个不断成长的软件库，为了更为高效地工作，DAD团队应力争使用企业批准的技术和开放的数据源，并尽可能合乎企业架构。凭借企业资产提高产品、方案的一致性，从而便于维护，降低开发成本和时间，同时降低运营成本。

加强组织的生态系统。产品研发至少应该融入企业的生态系统，即能够与运营人员、技术支持人员、服务人员紧密合作，将产品的价值以最短的时间传递给客户系统。更理想的情况，则是DAD团队在研发产品的方案计划时就考虑如何优化这个生态系统，进而调整工作目标，我们希望DAD团队不要做象牙塔里的闭门研发，而是更加接地气。要做到这一点，第一步是尽可能利用现有的企业资产，在整个生命周期，特别是发布之前，确保企业了解组织生态的最新状态和方向。第二步，由DAD团队的独立测试团队负责生产集成测试，以确保产品或解决方案适合客户的生产环境，极大地简化产品发布后的部署和实施，减少运营投入。

开放和诚信的监控。虽然敏捷方法是基于信任的方法，但聪明的管理策略是采用“信任但要核查，然后引导”的心态。适当管理的一个重要方面是通过各种手段进行监控。一种策略是相关负责人参加DAD项目团队的日常会议，了解当前的状态，这是Scrum社区所推广的策略。虽然我们极力推荐这一点，遗憾的是IBM没有实施得非常好，因为负责管理的高级经理们经常忙碌得没有时间参加会议。另一种方法是使用仪表盘和协作开发、集成工具，尤其是JazzTM平台，以及IBM的Rational Team ConcertTM软件，该软件可以在项目仪表盘中显示实时指标。

5. 风险驱动和价值驱动

DAD采用了一种被称为风险、价值驱动的方式，十分有效。DAD团队要努力降低项目风险，例如，利益关系人达成共识的产品愿景，经过论证的架构等；还要确保项目持续发展的可行性，是否已经开发完成足够的功能，是否为发布和投入生产环境做好了准备。同时，

DAD也是清晰的价值驱动过程，DAD团队将时间和资源的投入、产出均集中在最有价值的功能和解决方案上，而这项工作的优先级又因新工作项的优先顺序变化而随之调整。

我们依然用建房子的例子（参见2.2节）来阐述DAD的要点（见图3-4），同时，也可以回顾Scrum方法并比较它们的异同。

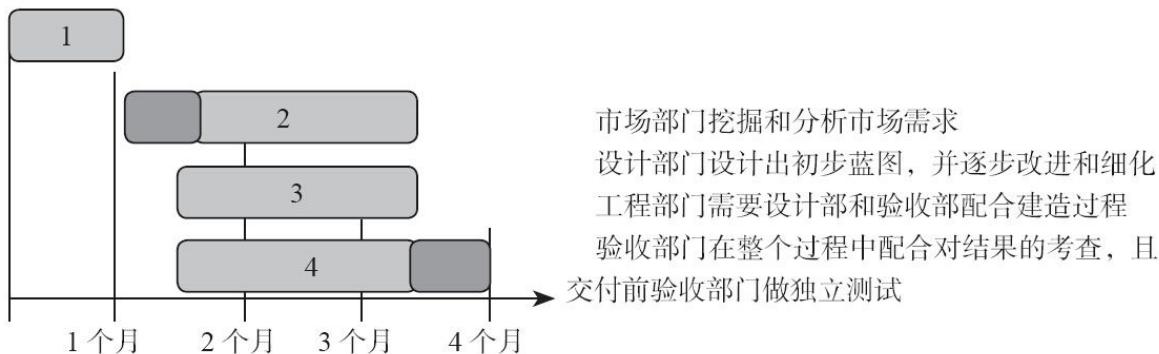


图3-4 使用DAD方法建造房子

某房地产开发公司依然采用敏捷方法，并遇到了市场估计不准和资金不足的问题，但这一次，开发商为了规避风险且降低成本，认为整个团队需要适量的敏捷仪式和纪律规范，这就是DAD思想。

使用DAD方法的房地产公司开始建房了，与Scrum方法不同，在项目工程正式构建之前，DAD团队的设计部门在整体计划上投入了相当的时间，了解该地段未来10年的政府规划，研究土地的地质结构和原有的地下水系统。为了验证新技术的可行性，他们还提前设想并路演了房顶拆装再组装的技术。

与Scrum方法相似的是，建造过程较为顺利，新客户不断加入。因为超期完成工作，开发商奖励团队一笔红利，团队士气高涨。验收部门不仅对之前部分做了再次验收，还将验收工作范围扩大到了整个房屋增量型开发工作的连接部分，因为发现及时，在房屋交付前修复了屋顶接缝的漏水问题，避免了一场纠纷。最终，房屋依然在第4个月的时候准时交付，并因为房屋的设计精良、结构丰富和质量过硬，被业内杂志给予了非常高的评价。

我们用表3-2来对比一下两家房地产公司使用不同方法建造房子的异同。

表3-2 SCRUM和DAD方法的异同

	Scrum	DAD
设计初步蓝图	先设计清晰的需求，后设计其他需求；不会在一开始就设计出整个蓝图	蓝图类似于软件原型，设计原型路演成功后方才开始设计；对于工程先后次序需同时考虑风险和价值
完全自由设计	完全授权团队自由发挥	规范性自由，团队需要了解企业架构、环境和相关社会规则，在确定了界限后自由发挥
全团队参与	全团队参与到完整的生产周期中，互相支持，可能存在职能交错	全团队参与到完整的生产周期中，互相支持，可能存在职能交错

(续)

	Scrum	DAD
独立测试阶段	没有独立测试阶段，随开发工作逐步交付测试成果，遗留下来的缺陷会交付进入下一阶段的开发工作	拥有独立测试阶段，尤其需覆盖迭代间各版本、各模块间的整合部分。而在接近尾声时，独立测试团队将完成最后的冲刺
成本	沟通成本低，但因为风险因素考虑不周，风险成本增加	沟通成本稍高，但因为同时考虑了风险和价值导向的研发，成本控制能力增加
客户满意度	客户十分满意，尤其是服务态度和满足客户需求方面可以打满分	客户十分满意，同时感受到团队的专业。团队帮助客户规避了因为期望值过高或者需求不成熟导致的风险，而对于应该接受的规则和界限，客户并不反感

3.2.2 新敏捷思维的延展

在敏捷的初期，敏捷开发往往是在小范围内进行且项目管理相对简单，小型且集中的敏捷团队管理思想仍然可以在这些情况下指导团队完成任务。而如今，机遇和环境已经显著改变，我们希望将敏捷开发应用于更广泛的项目，并且需要更庞大的团队，为了降低地区经济和市场领域的风险，同时在全球各个地区展开业务，CEO们希望利用分布式方式组织庞大的敏捷开发团队，同时与其他组织成为合作伙伴。因此，需要考虑法规和行业标准，克服巨大的技术或文化环境差异问题，利用超越单系统的思维方式，真正有效地调整跨系统因子——团队规模、地理分布、规范性需求、领域复杂度、技术复杂度、组织分散度和企业规范（见图3-5）。

不是每个项目团队都要面临所有可变维度带来的新问题，也没有任何两个团队会面临同一个问题，但任何一个问题都将增加敏捷交付的复杂性。根据不同的情况，DAD过程需要适应新环境，而团队必须找到策略来面对这些独特的挑战。

团队规模。核心敏捷方法能够很好地适应10~15人的团队，但如果团队大得多呢？50人，100人，或者1000人？随着团队规模的增长，团队沟通和协调会更加困难。书面沟通反而比核心敏捷所提倡的面对面沟通更加有效，但有时书面沟通以及面对面的沟通都会显得无力。

地理分布。如果团队成员位置分散，他们不在同一间会议室中，或许在同一建筑物内的不同楼层，或许在同一个城市的不同位置，甚至在不同的国家，将会发生什么有趣的事情？如果允许一些工程师在家工作，或是团队成员在不同的时区，又会发生什么？团队的联结被断开的可能性越来越大，有效的协作变得更加具有挑战性。

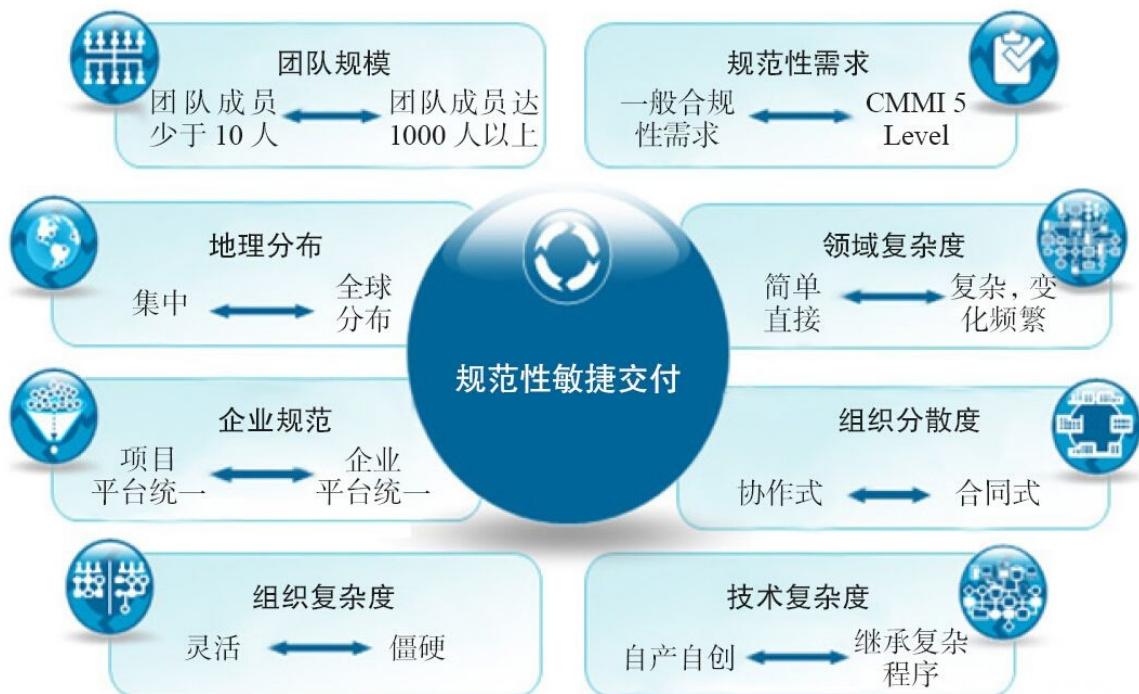


图3-5 ASM是DAD模型的八个维度延展

规范性需求。如果行业或地方监管法案出台，所有的移动设备均需送检过关后方能投入运营网络，又或者企业要求所有流程必须符合ISO9000认证，那么这些将带给产品交付过程什么影响？其实，往往是外部强加的、客户驱动以外的许多规则、需求增加了团队的工作流程和工作内容。规范流程往往彼此相似，而业务的日常运行至关重要。发展新客户是运营项目的一个例子，建立一个新的服务客户，无论是作为一次性客户或持续性客户，对项目进展都是有益的。但是，一般的工作流程对于客户基本上是无差异的。合同制造是这种类型的项目环境中的另一个实例，虽然每个产品可能是唯一的，但制造产品的流程却几乎完全相同。

领域复杂度。有些项目团队要解决的是一个非常简单的问题，如开发数据录入、查询应用程序或以信息公开为主的网站。而有些问题是复杂的，如监控医疗药品的生物化学工艺或通过大数据采集、存储和分析以辅助交通管制决策等。还有些问题涉及快速变化的信息，如金融系统的前后台转换、端到端的电子服务平台，需要较高的安全保障且兼顾易用性。有人认为，领域内的变化速度、系统的临界性，以及商业模式是影响软件开发过程的关键因素。更复杂的领域需要更加注重探索和实验，以降低风险，这其中的活动包括原型验证、建模和仿真等。

组织分散度。一个项目团队可能包括来自不同部门的成员、不同的合作伙伴，或由外部服务公司雇用、委派的成员。对于组织分散的团队，更适合的关系将是合同联结的方式，而不是协作。例如，一些项目成员虽然在需求、架构、设计，甚至是代码上产出，但实际上出于安全原因，他们不能访问源代码所属空间，不能获得资源执行测试，对真正的产品的理解有极大的偏差，且缺乏组织凝聚力，这将大大增加项目实施的风险。由于缺乏信任，从而减少合作意愿，甚至可能会增加知识产权（IP）斗争的风险。因此，许多组织都在努力调整人力资源采购流程和合作合同，以便更好地在分布式环境中实现产品、解决方案的成功交付。

技术复杂度。有些开发项目是从头开始的，易于提升整体质量和满足客户需求；但如果开发工作与现有的遗留系统和遗留数据源存在极大的耦合需求，产品就有可能变得不完美。同理，使用一个单一的平台来建立系统会相对容易，如果系统需要基于多种平台，或兼用几种不同的技术，这一过程就会相对困难。另一方面，技术复杂度是相对主观的因素，例如，在敏捷计划中，每个人对工作复杂度的理解会因为个人经验和技术水平的不同而不同。换句话说，不同的团队成员评估同一项工作会有不同的结果，所以针对技术复杂度，我们在思考交付过程时需要一个复杂的解决方案。

组织复杂度。有些组织结构和文化可能直接反映了瀑布式交付的价值理念，这将为采用和推广敏捷策略增加难度。更糟的是，组织内的不同团队可能有不同的看法，因为他们需要考虑自身的需求。而差异性策略即使是相当有效的，但作为一个整体，因为他们并不是向同一个方向努力，因而可能大大增加项目风险，也可能闭门造车般地重复投入资源开发着同一个轮子。

企业规范。大多数企业希望利用共同的基础架构平台，以降低生产成本，缩短产品上市时间，提高一致性，并促进可持续发展的步伐。如果项目团队只专注于他们的迫切需要，理念和实际就会脱离。为了充分利用公共平台和资源，项目团队需要采取有效的企业架构、企业业务建模、战略性重用和资源组合，必须同心协力以更好地实现规范敏捷交付流程。但是，这不是免费的，敏捷开发团队需要招募专业人士——如企业架构师和企业资产重用工程师——如果这些人士属于非可调动的资源。该企业的专业人士也需要学习规范敏捷交付的工作方式，因为未来的工作相比于他们在传统的团队的工作方式将有非常大的不同。

需要指出的是，每个扩展因子代表的复杂度才是关键，而每个项目团队都将面临这些复杂度的不同组合，这意味着团队需要根据实际情况做出调整。实际上，前四个扩展因子（团队规模、地理分布、规范性需求、组织分散度）是相对简单的，能够通过工具，采用适当的技术来解决。而其他四个扩展因子（领域复杂度、技术复杂度、组织复杂性和企业规范）则更难解决，许多企业正在努力寻找解决方案。

3.2.3 案例分析：“唯衣定制”——ASM框架下的制衣品牌

有些产品开发受到“合规性”的约束较多，且必须按常规的频率运行这些项目，而项目之间通常十分相似，对于新技术的需求并不大，因此对业务的日常运行模式没有必要经常更新和颠覆，此时有效的工作模式更应偏向传统的计划性开发模式。这好比“工厂式”的工作流程，面对每一个客户时流程基本上是相同的。而“合同制造”也正是这类项目的另一个特点，虽然每个产品可能是“唯一”的，但该过程构建出的系统通常是普遍的。相反，如果合规性要求降低，产品对于新技术本身的需求增加，例如互联网行业，则开发模式就应该向敏捷开发方向偏移，图3-6所示为扩展因子对开发流程的影响程度。

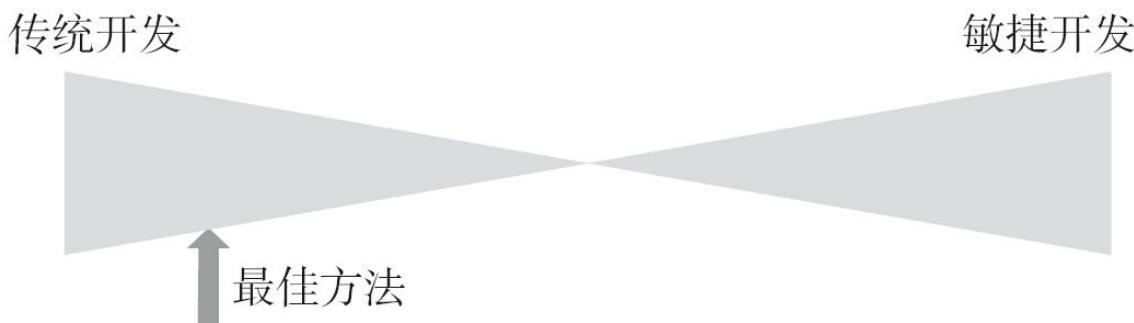


图3-6 扩展因子对流程的影响

对于合规性要求很高、不需要技术革新、相对流程类似生产线的工厂式项目，不妨分享一个制衣行业的故事。

“唯衣定制”是一家个人品牌店，这家店是由两兄弟在北京创业发展起来的，店面不大，但是工作量不小，几乎每天都有客户上门要求量身定制衬衣、西服等。大部分客户是男性，多在35岁以上，有了一定的经济基础，但同时身体也开始发福，市面上常规规格的服装不再适用，因此他们非常需要量身定制。此外，这家店还吸引了一些讲究的客户，这部分客户对外型比较挑剔，例如领子的形状、扣子的排列、衬衣和西服的材质、口袋设计的位置以及袖口的设计等，一件男士定制衬衣价位200~1000元不等。从产品的多元化方面来看，这家店是非常有吸引力的。价格合适、品质出众，又受到“私人定制”风潮侵袭，我果断地订了一件。

从下订单到通知取衣服的时间大约是15天，超出了老板承诺的10天。说实在的，如果再不来电话，我可能都忘记了。但店主很客气，一边道歉一边解释说，年前由于遇到了订单高峰期，加上工人提前回家过年，没有能够按照承诺的周期交付。看老板态度不错，加上我并不着急，也就未予计较。

取衣服时和店主聊了起来，这么一聊才知道，制衣的所有工序都是固定的流程，鲜有例外，每个订单都有详细的要求记录，一张订单会发送到工厂，然后工厂的工人依次“组装”。生产线上有足够的工人，而且工人的操作都非常熟练。“组装”就是一个瀑布式过程，一共5个环节，“前道、中道、后道、检验、大烫”，每个环节只完成一个或两个衣服的组成部分，如领子、上衣、扣子、领口的加工等。

有意思的是，“唯衣定制”的返工率只有不到10%，有时候100件衣服只有2~3件需要增肥或者减瘦。我比较惊诧，难道没有像我们估计那样，瀑布式的传统开发会因为过度的遵循计划，而忽视市场的变化吗？原来，除了第四道工序是整体检验、核对订单和需求以外，每道工序的工人都已经自然地对前道工序的结果做出了判断。事实上，这不仅仅是一个“等待和递交”的传统开发过程，每道工序都是“需求理解+设计+开发+测试”的完整过程。从另一个角度看，每个环节都可以交付成品，如成品上衣、成品袖口等。这分明是敏捷开发的迭代过程！

通过更进一步的了解，才发现“唯衣定制”真是团队建设的高手。团队采用自我管理方式，员工是“跨职能”的，也就是能够在各个环节工作，尤其是组长，更是各个环节的好手。员工的技能培养方向也是全面的，如果某个环节的工人输出明显慢下来，产生了积压，则组长会立即安排其他环节的工人来帮忙，保证所有工人忙闲适度。而每个环节的工人都有熟练和不熟练的，一起干就是培养新人的最好过程。组长和工人干一样的工作，同时还需要协调生产线上的人力分配，是不折不扣的拥有最令人信服的“能力+协调力”的“领导”。

从薪酬角度来看，工人除了底薪以外还有按照完成成衣件数计算的提成。因此，团队会很认真地做新员工入职考核，帮助筛选效率低下的人员。同时，大家的配合更多的是一种良性竞争，之间的利益关系不是此消彼长，而是配合越有效率，成品越多，得到的回报也越多。而且这个提成机制将员工的个人利益和团队利益捆绑在了一起。

那么，如果因为制衣问题返工了，工人会受到惩罚吗？老板的回答是：“不会的，成衣只要经过了以上五个环节就会安全出库，我们的衣服非常少会因为制衣质量问题而返工，绝大多数返工是因为太瘦、太肥，而这部分工作有专门的团队来处理，所需的成本在我们的报价中已经包含，并不是返工多少，我们赔多少，而是返工少了，我们赚得更多”。

“唯一定制”就是一个规范敏捷开发过程，因受到“合规性”因子影响，在一开始就已经将规划、成本预算、架构搭建好，和客户“合同”达成一致。在之后的开发过程中，按照增量型的迭代交付成品，每个迭代过程都有完整的“需求+设计+开发+测试”过程。在发布期之前，又会做一次完整的验证，确保产品和需求一致，最后将“按需定制”的商品发送给客户。

3.3 持续改进转型模型

3.3 持续改进转型模型

敏捷模型的核心就是通过人（people）、过程（process）、工具（technology & tools）三者的配合实现循序渐进的转型（见图3-7）。首先确定企业和团队的发展需求和盈利目的，面向目标，基于自身的情况设定可以量化且能够公正、客观地描述转型绩效的度量矩阵集，并与利益关系人在绩效度量手段上达成一致。再以绩效度量为导向，选择和借鉴成熟转型的敏捷实践方法，积极鼓励企业的自主创新与敏捷原则相结合，通过试点项目的转型实验验证和体现转型成果。试点转型过程将由度量矩阵持续监控，并向利益关系人报告转型的阶段性成果和度量结果。当试点项目转型获得认可后，方可将转型方案推广至3~5个部门，最终推广至整个企业。与此同时，需将“人”的观念的根本性转变过程提上日程，或许组织结构形态的变革也需要深入考虑。这就是我们所推广的“可靠敏捷转型”，企业将通过自律、自我管理的成长使整个生产效率在迭代增量的生产过程中持续提高；更为重要的是，企业在转型前所确定的目标均将得到实现。

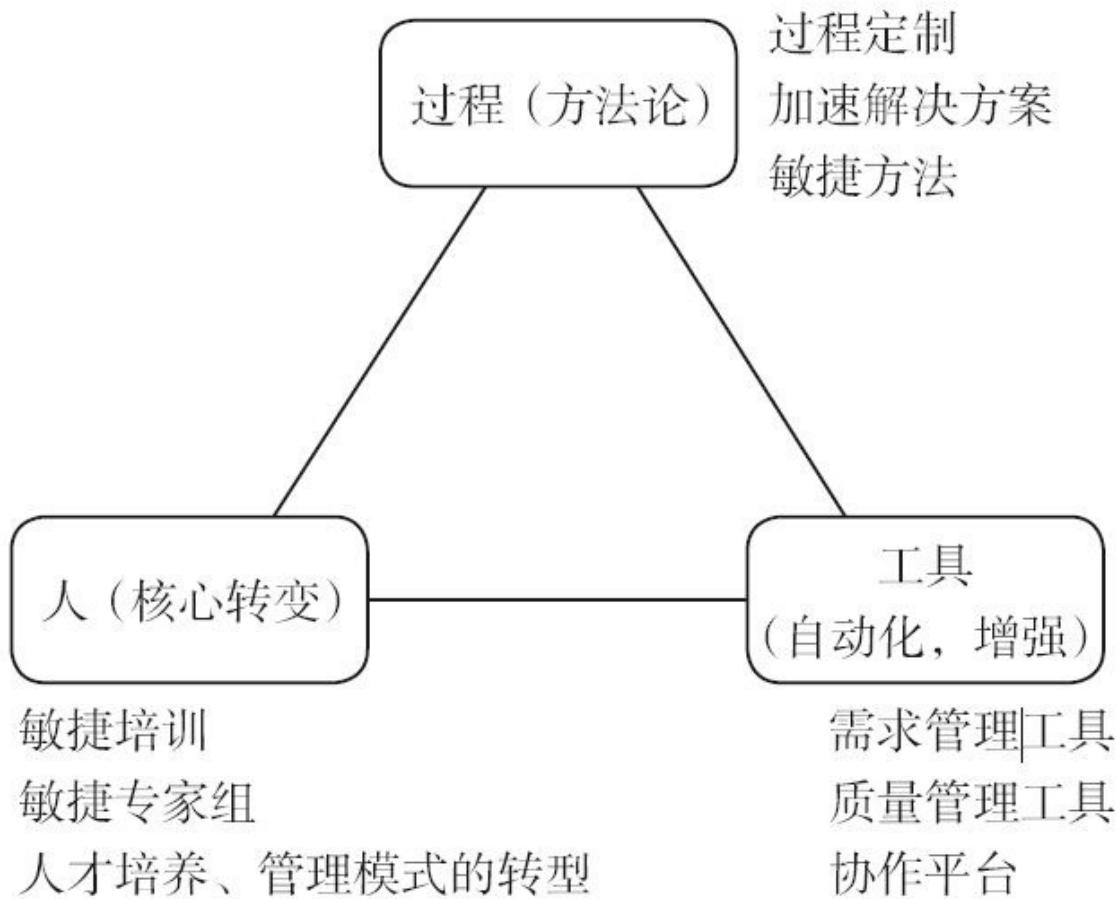


图3-7 敏捷转型的是人、过程、工具三者的协同转型

转型必然带来过程的改进，甚至可以说过程改进是转型最明确的、也是最关注的目标。IBM的DAD方法和ASM框架都是值得借鉴的，40人以内的团队使用DAD方法，在规范中找到团队敏捷和项目透明度的双重目标；100人以上的团队会受到团队规模、公司文化、企业架构甚至地方政策的限制，如果要突破瓶颈并在利润上得到更进一步的提升，需要控制团队的内耗，因此需要合理地使用ASM方法。

过程改进绝不是方法的照搬或者系数的微调，准确地说，那些成功的案例和经验都只是顺其自然达到的目标。对于过程改进，我们需要的是为了明确的、可度量的业务和运营目标，持续性地提高过程质量。所以，以下指出过程改进中的几项关键工作。

1) 需要满足利益关系人的需求，除了客户所表达的，还需要了解需求背后的真实需求，或者开发出客户所未考虑到的需求。

2) 设定过程改进目标，并为这些目标设定优先级；明确验收标准，对于可预计的风险、利益进行评估；针对一致需求、目标，建立过程改进度量矩阵（见图3-8）。

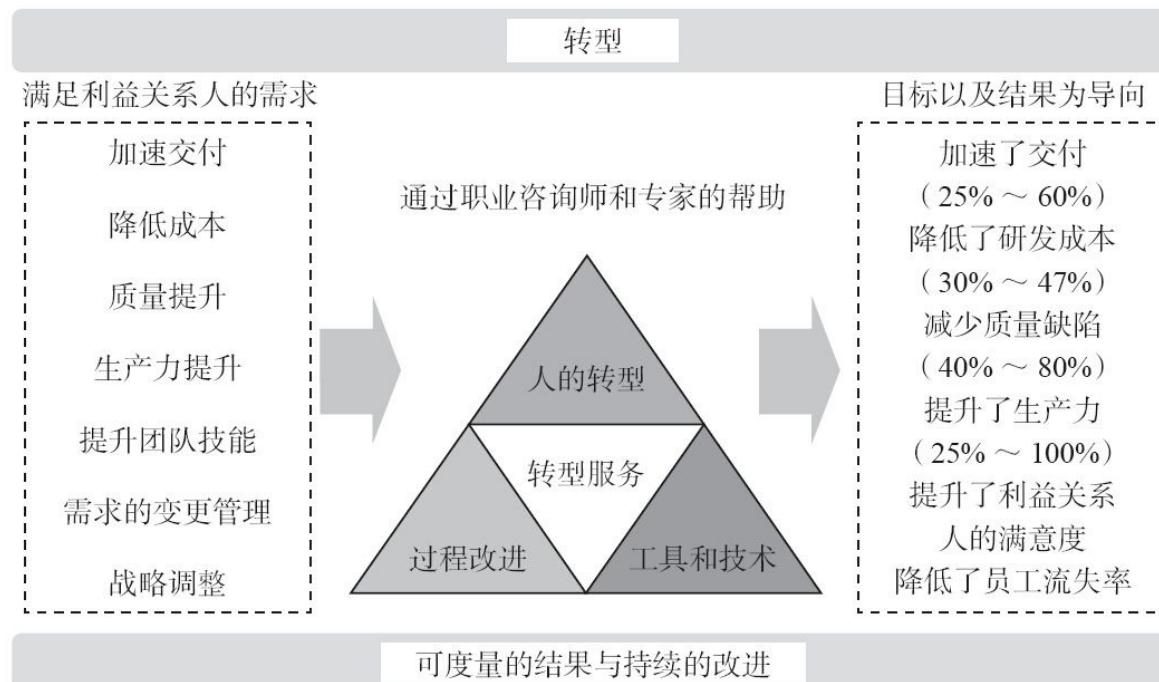


图3-8 可度量的多维度敏捷服务和以结果为导向的敏捷转型

3) 在项目实践中，以度量监控的方式控制风险，导向目标，开展项目工作（见表3-3）。

表3-3 敏捷转型的规划步骤及目标设定

项目规划	目标
敏捷工作坊：针对项目负责人和骨干	建立起敏捷研发的基本概念、潜在价值，以及组织机构对于成功实施敏捷转型的期望
总监们的敏捷顾问：针对高阶领导	针对高阶领导（总监）对敏捷的困惑提供顾问、咨询服务，帮助其了解敏捷对其自身及其所管理的组织或者团队所产生的价值和意义
总监们的业务价值工作坊：针对高阶领导	建立并明确敏捷转型目标、过程改进的价值、转型战略、项目实施的重点，对系统改造计划进行交流，达成一致 输出结果：转型服务同意书，初步服务规划范围、目标以及变更策略等
敏捷部署的战略和路线图 • 健康评估 • 诊断 • 度量 • 建议 • 路线图	理解客户面临的困境，根据环境因素描述转型前的度量基线； 通过实践提出建议和调整战略部署 输出结果：协作式的敏捷部署战略和路线图
战略执行：启动项目	可靠地持续化过程改进，并以度量监控转型过程

4) 阶段性地请利益关系人进行验收，考核过程改进成果，并修正目标和过程改进矩阵，提高实践的可行性或者提出更有价值的实践方案。

5) 将过程改进的最终结果和度量结果呈现给利益关系人。

至此，过程改进项目宣告完成。

以上敏捷转型五步法能够解决大部分中小型企业40人以下团队的转型过程，但是对于100人以上或者更大型的企业和团队，则需要在试点项目完成转型后，增加一个企业级推广过程，这个过程还需要扩展到企业文化和发展结构，为了说明这一过程，我们将复杂企业的转型计划用图3-9表达出来。

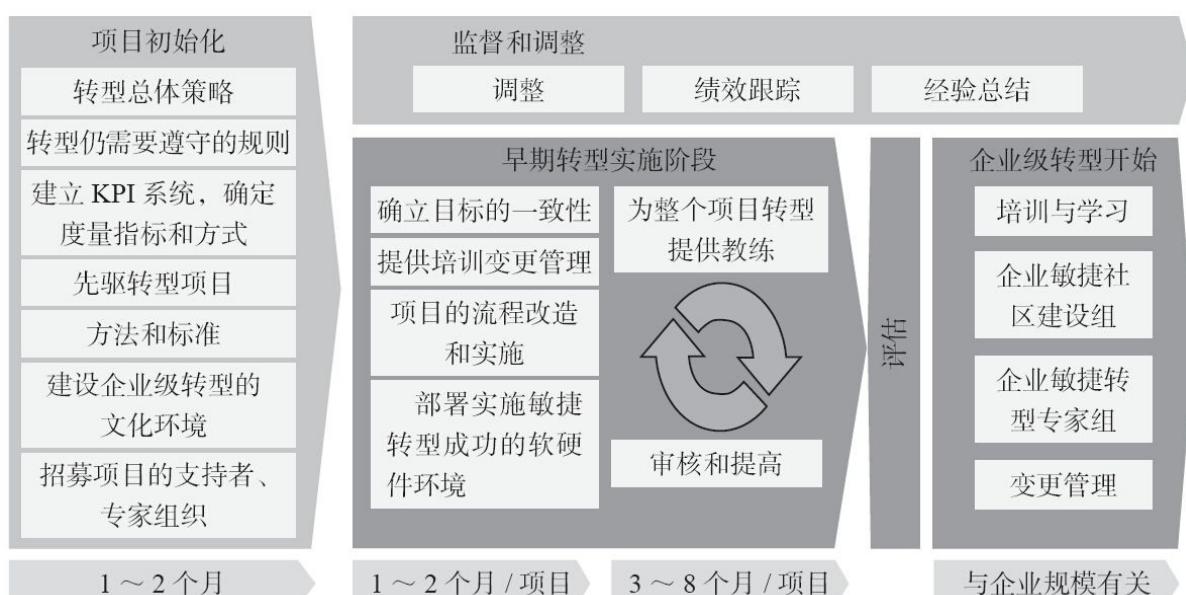


图3-9 典型大型企业敏捷转型规划

讲了这么多的敏捷转型理论，到底什么是敏捷方法、什么是传统方法，DAD和ASM又该如何实践？我们来讲个故事。

3.4 案例分析：IBM Rational中国的敏捷转型

3.4 案例分析：IBM Rational中国的敏捷转型

2010年春，IBM开始着手准备整理Rational的敏捷开发过程，其目的并非统一所有产品团队的IBM式Scrum开发流程，这既不可能也不合理，而是需要将原来的Scrum落地成为一个像IPD（IBM Process Development）那样具有具体指导意义的、对于初步使用这套方法的各种角色人员都有直接指导作用的方法论。但是我们不能忽视一个客观事实，即各种团队实施Scrum这些年来形成的良好的差异性实践活动。

这项工程的价值还在于为Rational团队定义一种“框架”和“规范”，因为我们始终相信，只有在有“边界”的空间，团队才能充分实现自我的价值和发挥其创造力。

“规范”最好不是操作级别的手册（不得不说，某些时候手册却更有价值），而是一个“组织级的战略”加上有高度的“规划”。因为，在企业里，项目的透明度和团队的自由、敏捷一样重要，我们相信：

·敏捷需要纪律、需要过程，XP就是拥有各种纪律的敏捷方法，因此敏捷绝不是无政府主义。

·敏捷不需要管理是一种错误的认识，团队自我组织、自律自持、自知之明，但仍然需要管理，否则无法实现敏捷转型。

·为团队定制可持续性开展活动的过程或者方法，是真正运用敏捷核心价值实现敏捷框架的过程。

·敏捷方法、实践过程的设计、使用以及过程的管理均是敏捷转型的重要环节之一。

在着手设计、撰写Rational中国的敏捷开发过程的同时，我和我的团队踊跃地参加到全球敏捷社区领导团队的各项工作，其中一项对我形成巨大影响的是和IBM的CoC团队以及首席方法论学者Scott Ambler对即将出版的一本新书《Disciplined Agile Delivery》进行评审。

对于我的导师、前辈Scott Ambler先生的书，我是缺乏勇气自诩“评委”的，但是令人嫉妒也罢，走运也罢，我真是太荣幸地被邀请加入此书的评审活动中了。Agile@IBM领导团队成员全部参与了评审，因为所涉及的章节内容非常丰富、细致，我们被分成了若干组，各自集中对其负责的一些章节开展讨论并提出评审意见，大家和Scott先生的探讨也时常发生，Scott其实是个幽默的人，这让针锋相对的风险几乎降低到零。

与其说我给书中章节提供了一些建议不如说我彻底地向导师、专家学习了IBM的敏捷方法“Discipline Agile Delivery（DAD）”和“Agile at Scale”，这些工作对我最终完成Rational敏捷开发过程大有裨益。

在敏捷转型之前，Rational在中国的研发产品有CC、CQ、RQM、RPE、RTC等，而这些产品都有着浓厚的“地方特色”，CCM产品族CC和CQ从瀑布式、RUP、敏捷开发到今天，在中国已有十多年的基础和历史，产品结构和功能非常强大，对其中任何一项功能的改动都会影响到过去十几年开发者留下的历史代码，既是对团队技术的挑战，也为产品的后续发布带来较大的风险。

所以，CCM团队相比其他产品开发团队具有更严谨的开发流程，尽管也采用迭代式开发并召开每日的迭代会议，但团队在选择敏捷实践、改进过程和任何产品的改动上都表现出尤其谨慎的态度。当我希望他们展示流程和方法中最为“敏捷”的部分时，我感觉实在是难为这支团队。我非常清楚，对于这样稳定的产品，新功能的开发、架构的改变、新问题的解决均需

要开发者有非常强的整体把控，任何一次变更都需要尤其谨慎。因而无论如何我们都希望敏捷转型对团队的测试、开发、支持小组产生太大的负面影响，而非常理解维持目前稳定的开发过程、保证团队继续成长和技术深入发展才是重点。这也再次证明，并不是纯粹的敏捷、100%的敏捷就是对的，敏捷是一个框架，我们开发出来的流程一定要有包容性，对于产品的复杂度、团队的组成结构和规模、行业特点以及已成既定事实的做法，应该更多地去观察它的优点并加以继承，而不是一味地完全推翻重来。

RPE产品刚刚跟随Teleoic的并购进入IBM中国，产品的研发流程、方法看起来有着很好的可塑性。中国团队刚刚经历了两年的建设和成长，从测试团队发展成了开发支持团队，团队的配合比较默契，团队成员的技术、能力、工作经验也相对过去丰富和成熟了好多。但是，RPE产品在中国的市场还不成熟，同时，RPE的“元老”团队拥有最好的技术，对产品的话语权也最重，而不幸的是，这个元老团队是有控制欲的、坚持自己原有流程和方法的典型，在一年前，所有的开发工作都是在家、实验室、PC上完成，项目的透明度几乎为零，可以说，产品完全取决于个人能力而不是团队协作。元老中的每个人都是独当一面的专家，除了架构师本人，没有人对产品有全面的理解。他们拒绝在工作中履行“不认同的任何一套方法”，并且坚持自己掌控绝对领导力。整个团队约40人，结构复杂，分散于世界各地，产品经理在加拿大、架构师和元老们在罗马尼亚、测试团队在印度、需求团队在美国、支持和主力开发团队在中国、项目发布经理在英国，这个团队想要建立同一个目标、适应新的开发方法和流程的挑战绝对不小。这个时候让团队参与流程改进一定得不到足够的支持，也会受到元老们的挑战。因此，在不影响大局面的情况下，我们只能围绕团队本身的成长和成功来实施敏捷，先“齐家”再“治国”，取得阶段性的成绩，等到这些成绩得到大家的普遍认可后，再来考虑系统级的改进测试、设计、产品清单优先级排列等较弱的环节和过程。在此之前，我们期望可以获得本土团队的支持、配合，并让员工体会到敏捷的优势和好处，以实际表现来说话。

开发中心还有RTC和RQM团队，这两支团队在Jazz统一的敏捷模型下工作，但因为Jazz团队非常庞大，中国的队伍所占比例很小，参与具体的开发和测试工作比较多，而鲜有参与产品团队建设和过程改进、架构设计的讨论。团队埋头于细节工作，很难看到全局，因为没有对比，也感觉不到敏捷实施前后以及将来会对他们有何影响，对于整合一套适用于Rational中国的统一开发过程显然缺乏兴趣。如果真的以他们做为过程改进工作的切入点，恐怕团队会因此力不从心，反而招致不必要的来自投资方的挑战和压力。

3.4.1 统一的研发流程

Rational中国团队的高级经理层希望建立中国实验室的领导力品牌形象，指出我们需要开发出、至少是发掘出一套适用于中国团队的开放的、统一的开发流程，在此基础上建立相应标准和分析、度量手段，就可以从大量看似无序的数据中分析出有序的规律，例如，预测项目的健康程度、更快地预知可能的风险并建设性地提供解决方案等。我们希望通过团队在大型项目上的经验优势、数据分析上的工具优势，以及高端技术人才优势发掘出更有深远意义的研发理论，这个理论将指导我们解决企业级应用结构调整不够灵活的问题，降低企业级软件的整合难度，预知下一代企业级应用开发过程，揭示软件全生命周期中尚未被人发现的规律，具有诸多意义。

这项工作的开展必须有愿景设计并充分理解当前环境下“存在即合理”的最佳实践模式，而这些实践必须可以从这个完整的过程中摘取出来，经过改进后普及于一般团队的活动。例如，敏捷开发的持续化集成、测试驱动开发等实践活动，以及更多等待我们去发现、总结的智慧和规律。

每个团队对自己的优秀实践活动的发掘、抽象、提升已经潜移默化地对团队自身产生了有益的影响。这些输出不仅直接提升了团队的工作热情、工作认同感和工作效率，也让团队有了更多的使命感。我们不仅仅在写代码、找缺陷，我们是在研究和指导企业级应用开发，让越来越多同我们一样的工程师更卓越、更有成就感。

优秀的过程设计、方法论设计能够指导不同成熟度的团队，适用于不同习惯、规则、复杂程度的领域，以及不同分布程度的组织结构。这个过程有意忽视企业软件研发、甚至是一

般软件研发价值体系中的细枝末节，用“最少的步骤实现最大的价值流（Value Stream）”，即促使产品安全、快速投入市场的商业价值。同时，可以通过调整“过程参数”来满足团队各自的差异性附加价值，即在公式中调整某些数值后，就能够实现“额外的期望”，例如“团队的成长”。我们把这套寄予厚望的方法称为“Rational敏捷之路”。

3.4.2 流程开发的利器

开发“Rational敏捷之路”需要绘制流程图，定义角色、阐述职责，还需要绘制框图、流图、包图等，以各种形式方便大家一目了然地看懂流程，流程是一个有层次、有进入和退出的过程，在组织和开发流程的同时，还要以粗细不同的粒度展示流程图，换句话说，为了更好地向不同层面和角色的人展示流程，必须做到既能够通过一组节点和线条的关联来说明这个流程，又能够只用一个节点特写出这个流程在大流程框架下所处的位置和作用。因此，在没有得心应手的“省力”工具前，我们迟迟没有开工。

我的“零起步”开始于Rational Method Composer（RMC）的入门（见图3-10）。RMC的设计思想是将所有的过程、方法定义成“方法内容+流程=方法”（见图3-11），即RMC的描述既可以提供工作描述又能给出工作顺序（见图3-12）。



图3-10 使用RMC工具进行过程开发的工作一览

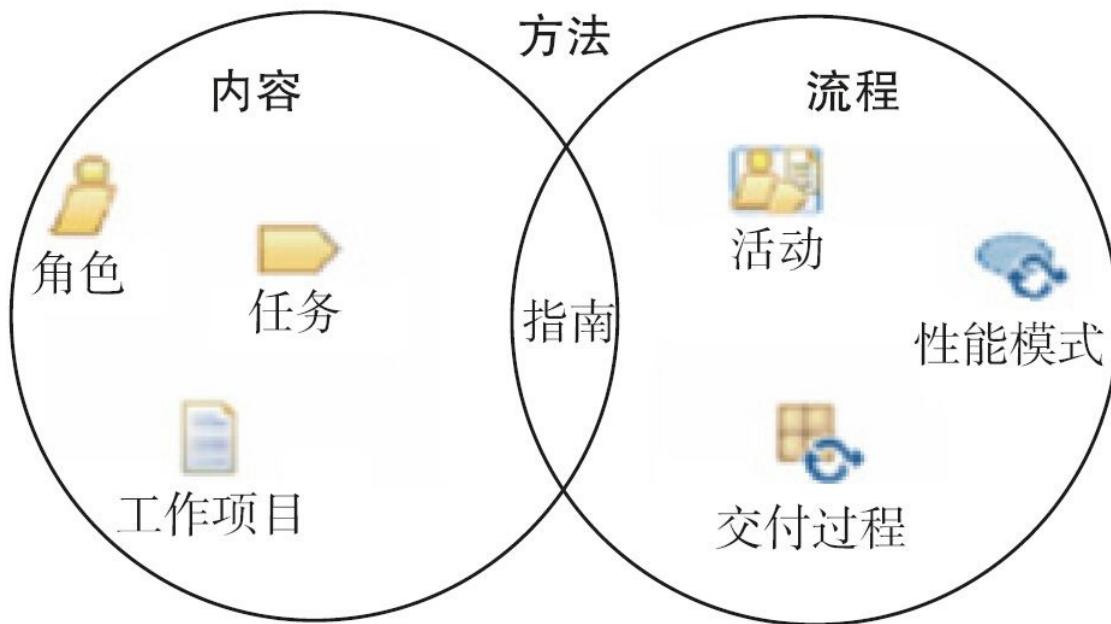


图3-11 RMC实现方法和过程的描述所依赖的元素——方法内容+流程

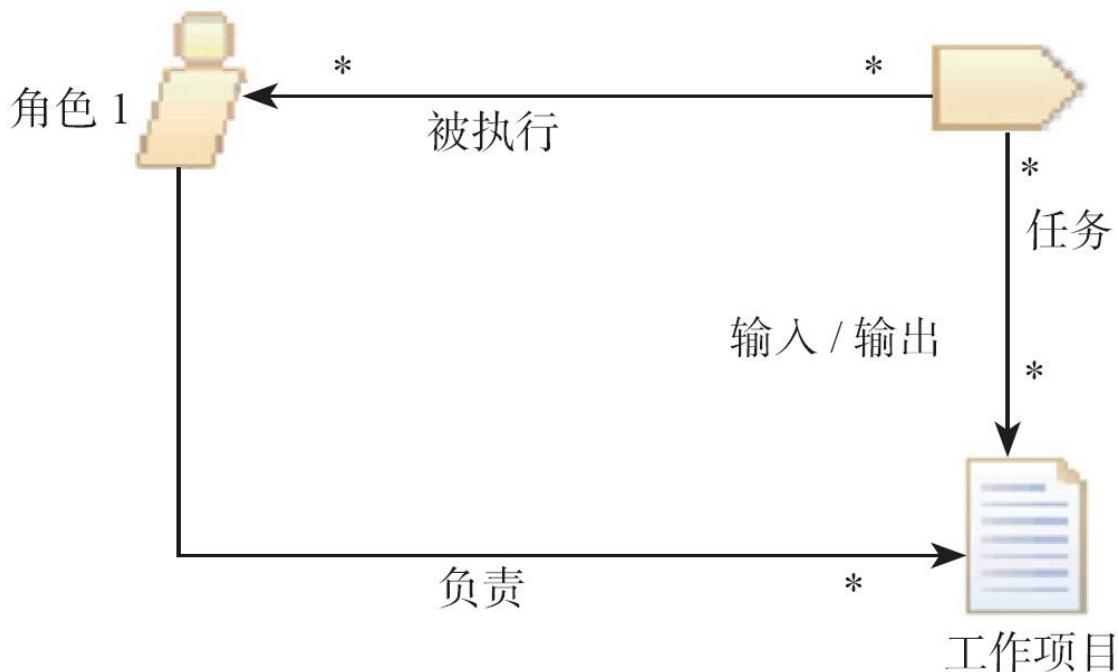


图3-12 RMC视图中基本要素之间的抽象关系

方法内容。方法内容是对工作的描述，它作为关键方法构建的块可以重复使用。方法内容描述任务、角色、工作输出和指南等包含在完整工作中的静态元素。

流程。流程表示将以上定义的静态元素及其内容组织成为一个个鲜活的、有生命周期的动态流，并且为内容提供了一个有序的结构。

任何一个方法或方法单元（局部方法），一定是由角色、任务和工作输出以及之间存在关系视图构成的，RMC可以将任意复杂或简单的方法展示为宏观或微观的视图。

在“Rational敏捷之路”的开发过程中，RMC团队开发出的“开箱即用”等方法插件（Method

Plugin) (见图3-13) 帮助我们减少了许多重复劳动，也激发了我们的创作“灵感”。这些插件可以独立存在，也可以依赖和继承，其设计的过程类似于从一些父类和接口中选择生成子类和实例的代码过程。

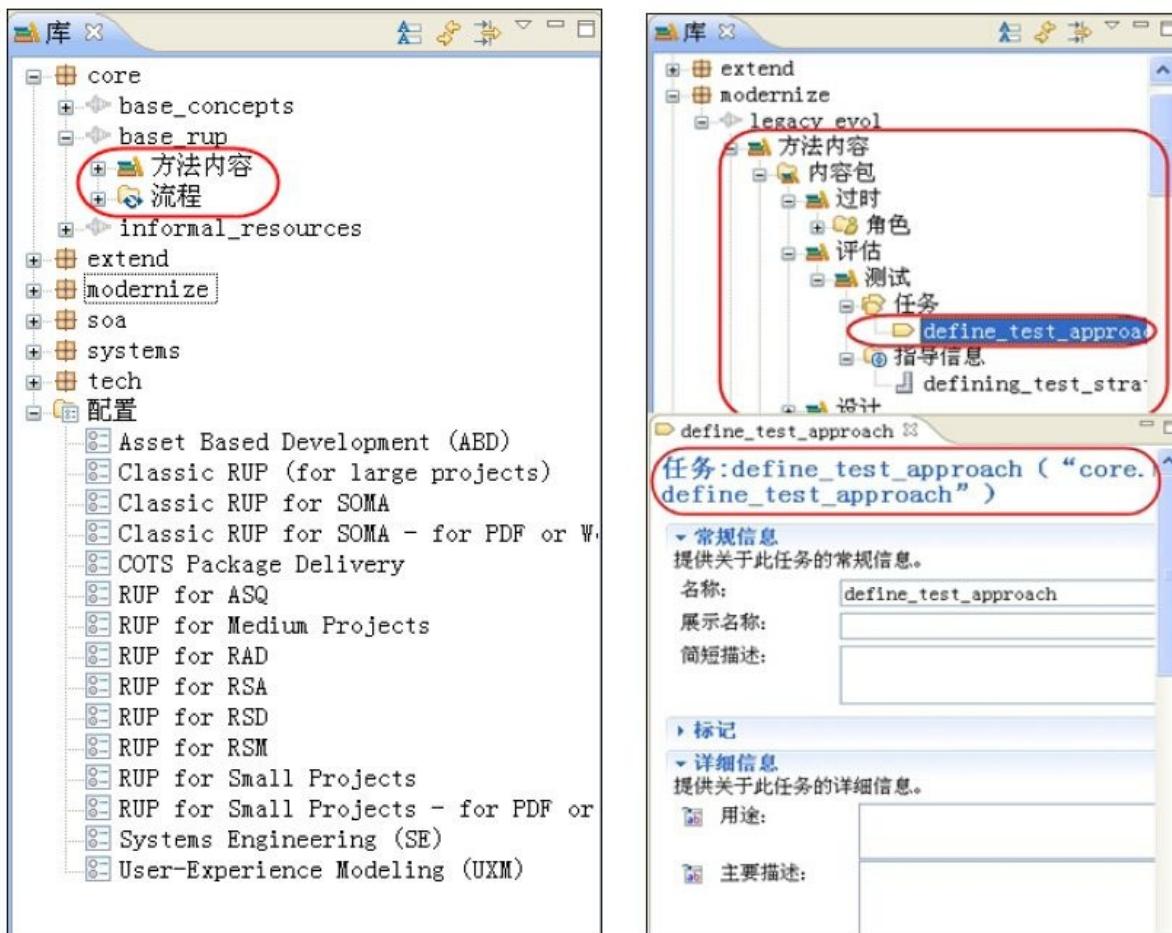


图3-13 RMC提供了可以继承和实现的大量方法插件

3.4.3 研发转型的困境

即使CCM在中国已经开始向成熟产品的轨迹偏移，但产品研发团队仍然要解决许多独特的问题。而类似RPE这样的新型产品，产品的研发流程还在形成期，而且因为内部人员对新过程的接受程度和对目标的期待不统一，敏捷转型也未能像我们想象的那样大刀阔斧般地成为“敏捷变革之楷模”。

这一系列的困难使我们意识到，不能成为“理想主义者”或者“敏捷狂热之徒”，为了改革而蛮横地强迫团队服从个人意志，破坏了团队生态，忽视团队自我改良的愿望；也不能以粗放的方式任由团队走入误区却视而不见。任何不切实际地追求形象、模式而不是追求卓越的做法都不应该成为过程改进的目标，相反，我们应该基于数据分析，采用循证型方式将团队引导进入敏捷的领域，在改进团队外围环境（领导层充分信任与支持）的基础上，在团队充分得到所需的资源和支持的前提下，在团队产生自我持续改进的愿望后，系统地规划团队的敏捷转型。在转型环境建立的同时，努力完成Rational敏捷开发过程才有实际意义。

然而，我们的大多数项目和团队的敏捷转型要么处在传统迈向理想敏捷的某一中间状态，要么处在一种“存在即合理”的真实敏捷状态。这个状态的下一个转型绝不是“理想敏捷”、“核心敏捷”，而是另一种由独特需求和环境决定的“多态性敏捷”。

数据说明我们的推测和预期是正确的。IBM中国敏捷社区的年度调研显示，IBM中国开发中心76%的员工认为自己的组织内采用了超过一种的敏捷方法做项目开发。那么，Rational敏

捷开发流程该如何构建？是抽象还是具体？会被大家接受吗？

回忆2009年春和Scott Ambler的一次座谈，我记录了这样的一段话：“随着敏捷像风暴一样进入了各行各业，敏捷正变得更加普遍。敏捷项目团队相对于传统的项目团队享有更高的成功率，提供更高质量的产品，利益关系人的满意度更高，提供更好的投资回报率（ROI），并提供系统能更快进入市场的保障。由于这些优势，才使敏捷如此声势浩大。”但是，我们均认可的一个观点是：平均意义上，敏捷团队比传统团队更成功，并不意味着所有的敏捷团队都成功，也不意味着所有的组织都能相同程度地实现敏捷的价值。敏捷既有显著的优点也有很大的改进空间，这为我们完成Rational敏捷开发过程增添了一股动力。

3.4.4 定制企业开发过程

我们尝试和各个团队合作，试图建立统一的、既有包容性又能够反映出各自需求的Rational敏捷开发过程，这一过程将是未来持续改进的基础构建。目标很理想，但不可避免地遇到了不少问题，其中最主要的是人的惰性和对变化的恐惧。

大家均表示还是愿意继续使用和开发当下的流程和工具，尤其是已经认可了先前的成功经验后，他们相信现有流程已经“成熟”，不愿意接受外来的改变。而我们要做的正是将DAD和ASM的概念输送给这个组织，事实上我们需要更多的资源来完成这项工作。

除了请IBM全球社区中经验丰富的敏捷领导人帮忙出谋划策和提供信息支持外，我们还积极地邀请公司内的敏捷支持者、传播者一起参与过程改进的规划。同时，找到各个部门经理，了解他们对过程改进、敏捷开发方法的期望。最后，就是寻求RMC团队开发经理的技术协助，将流程的定制技术难度降到最低，最可贵的是，我们的流程定制也发布在RMC的官方Wiki上，所有IBM员工都可以看到，这也增加了流程的置信度。

最终促成敏捷具体方案设计的步骤，可归纳为以下的重要事件：

- 1) 邀请好友、《Disciplined Agile Delivery》这本书的中文译者、IBM中国区的社区负责人Kevin Yan，向部门中做过程改进和流程设计的同事和管理层介绍DAD。
- 2) 安排组织内部的几次头脑风暴，由部门经理选送部门代表，与其他部门交流当下流程的“利与弊”，将大家共同的重点问题作为下几次会议的主要主题进行讨论和经验分享。
- 3) 联结IBM全球敏捷社区领导团队，寻找可借鉴的转型案例，并且从领导团队中寻找方案的支持者和反对者，将话题提上领导团队的日常会议日程，使得各种意见都能够被充分地探讨，即使是反对者的意见，也能够帮助我们提升工作质量。
- 4) 申请RMC开发团队的技术支持，将过程、方法、实践活动的流程发布到公司内网，接受来自各团队的意见和建议，并将更新后的版本再发布。
- 5) 制作组织内的新闻栏目，每隔一定时间发布工作总结和团队的阶段性收获，分享国内外有影响力的专栏内容转载、国际敏捷社区的文章和经验等。这些内容帮助组织内部在转型和统一流程之前，从组织外的成功案例中获得信心。
- 6) 将工作的创新点、成绩和组织目标结合，在组织内部寻找高阶的支持者，通过他们的长期影响力，为组织内的过程统一和转型寻找资源和长期支持。
- 7) 当然，很难让所有的关键人物认同我们做的事情，所以，我们与组织内部几位有影响力的技术、管理人士进行了一对一的沟通，在他们希望“干掉”这个想法之前，获得他们的支持或者“宽容”。
- 8) 在组织内的各个团队中寻找对敏捷感兴趣的、愿意做深入研究的人才组成“敏捷学习小组”，参与过程改进和开发，通过他们影响周围的同事、上级和管理团队，并让他们从这项工作中受益，例如，帮助其发表文章、参与跨团队的经验分享，或者通过过程改进中随之而

来的个人和团队效率、质量的可度量的提升影响个人绩效评分。

正是因为把上面的工作做足了，后来公司才顺其自然地接受了DAD方法。针对各自独特的环境，团队调整了有影响的几个缩放因子，很快形成了各自成熟的敏捷过程。我们把这种“私人定制”的过程保存在各自团队的研发资产中，将大家共同遵守的“团队定制”，即基于DAD的Rational敏捷开发过程发布在了企业内部的知识平台上。

在图3-14中，软件解决方案全生命周期过程可以分成三个阶段，从左至右依次是“构想、构建和发布”阶段。在构想阶段我们组建团队、初始化项目、设定高层的目标和计划，对于新产品研发（如RTC和RPE），这个过程还意味着架构和原型产品的定义。对应于传统项目的CCM产品线，这个过程意指一个发布版本的计划阶段。

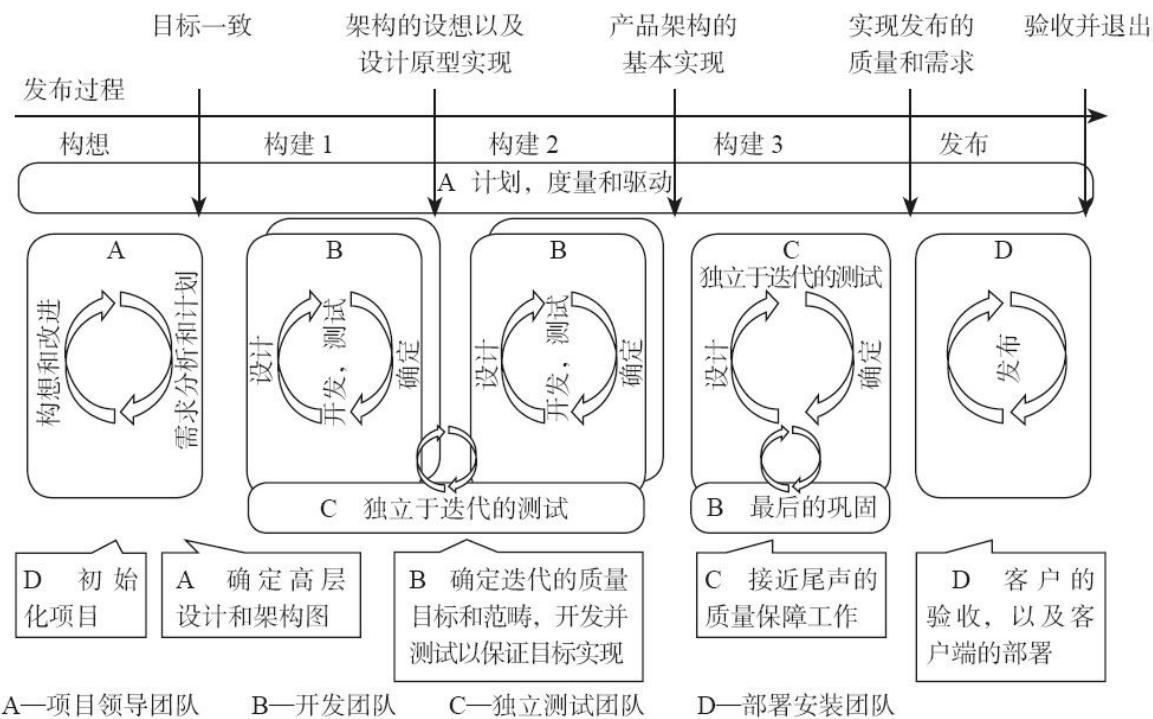


图3-14 基于DAD的Rational敏捷开发过程

3.4.5 规范性敏捷管理

将DAD方法运用在IBM Rational中，除了支撑团队的敏捷性以外，对于项目的透明度，如风险预估和控制、架构的原型需求，以及构建过程中的迭代管理、测试管理、变更管理等，都需要融入到开发和测试流程中。为了保证管理与控制的透明度，又不打扰团队的专注和协作，不破坏团队自我管理的氛围，我们需要做的是通过工具集成，渗透到项目的细枝末节，再将统计和组合的数据分析和归纳为尽可能直观的方式进行传递，而团队（包括管理者）可以通过定制视图来达到对项目的方方面面了然于心的目的。

在构建阶段，即核心敏捷所关注的高优先级产品清单的实现、测试、验收过程，这个阶段由多个迭代组成。每个迭代均有可以工作的产品交付，最后的几个迭代交付的产品将是潜在可发布的。

既然流程定制与开发已经在RMC平台中完成，下面通过RMC的生成过程视图来分析每个阶段的详细流程设计。例如，图3-15为构建阶段流程图。

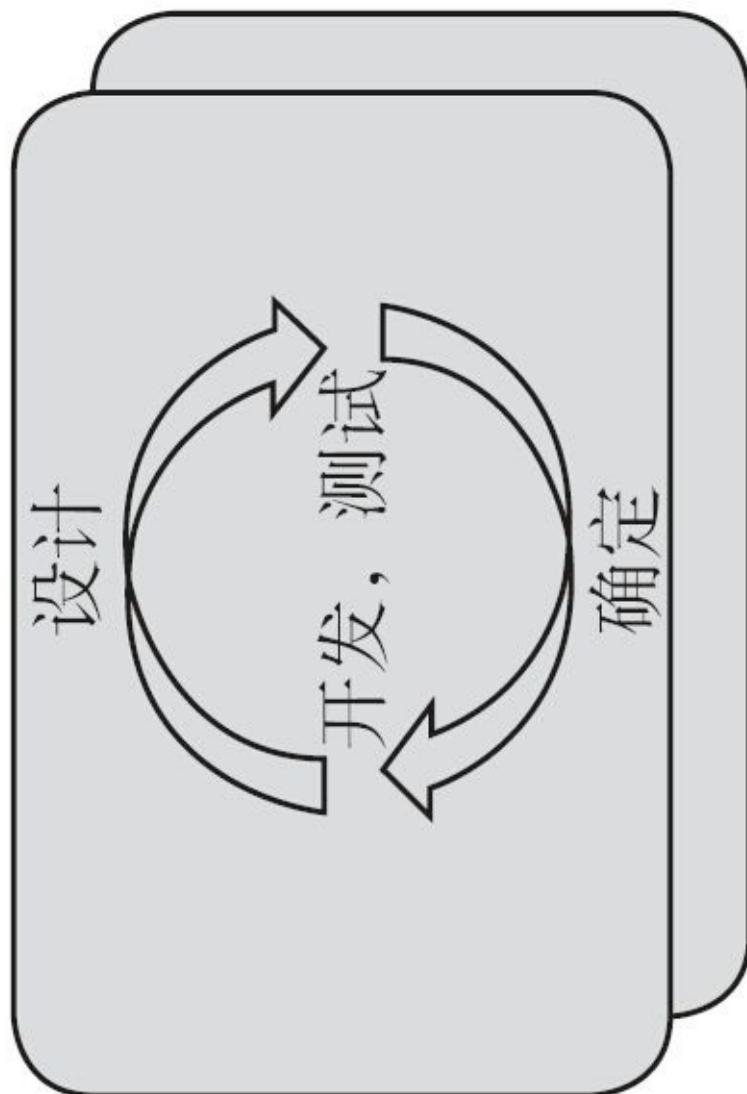


图3-15 构建阶段

图3-16中，通过一个迭代[1..n]的图标元素来代表多个迭代过程，这个元素是可以展开的，单击这个图标展示迭代的内部流程图，如图3-17所示。

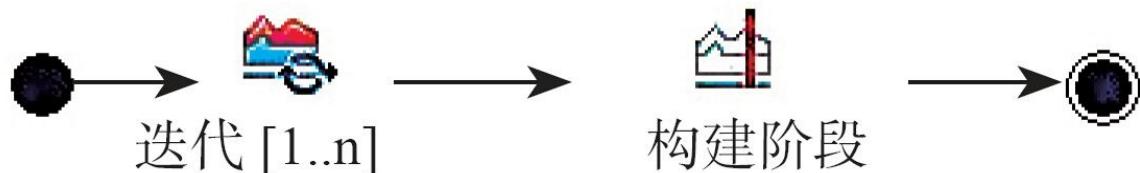


图3-16 构建阶段流程图

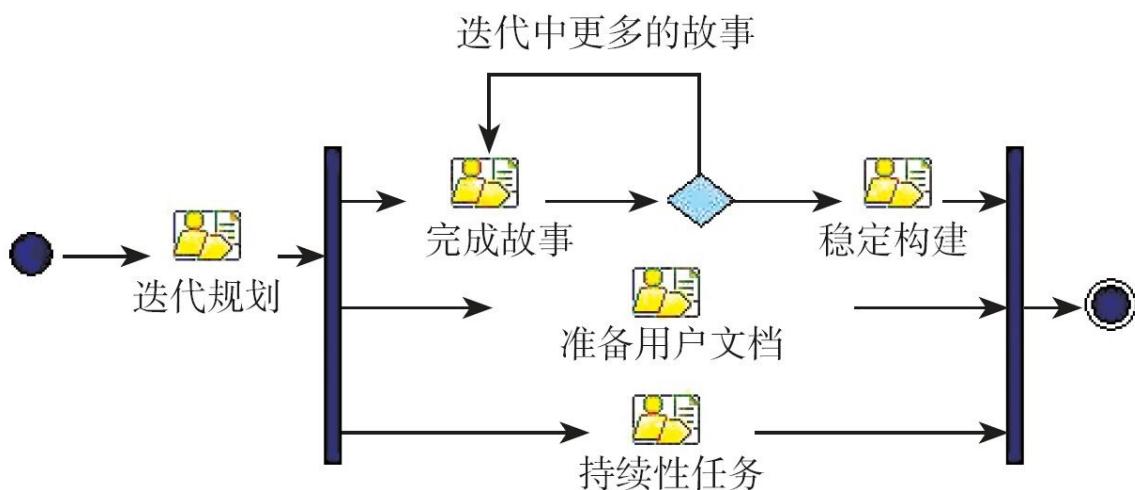


图3-17 构建阶段迭代内部流程图

在一个核心构建迭代中，团队需要完成“迭代规划（Plan Iteration）、完成故事（Complete Story）、稳定构建（Stabilize）以及准备用户文档”四项主要任务。同时，在这个流程结构中，有一个特别的任务是“持续性任务（ongoing tasks）”，它记录的是在项目过程中，那些支持团队完成故事、稳定构建等一系列核心过程所需的“辅助、支持”工作，包括管理、计划和各种报告的输出。在分析迭代规划和完成故事之前，我们先看看持续性任务是如何通过“迭代管理（Manage Iteration）、风险管理（Manage Risks）、监控和测试管理（Monitor&Control Test）、变更管理Request Change”四部分工作来支撑团队的（见图3-18）。

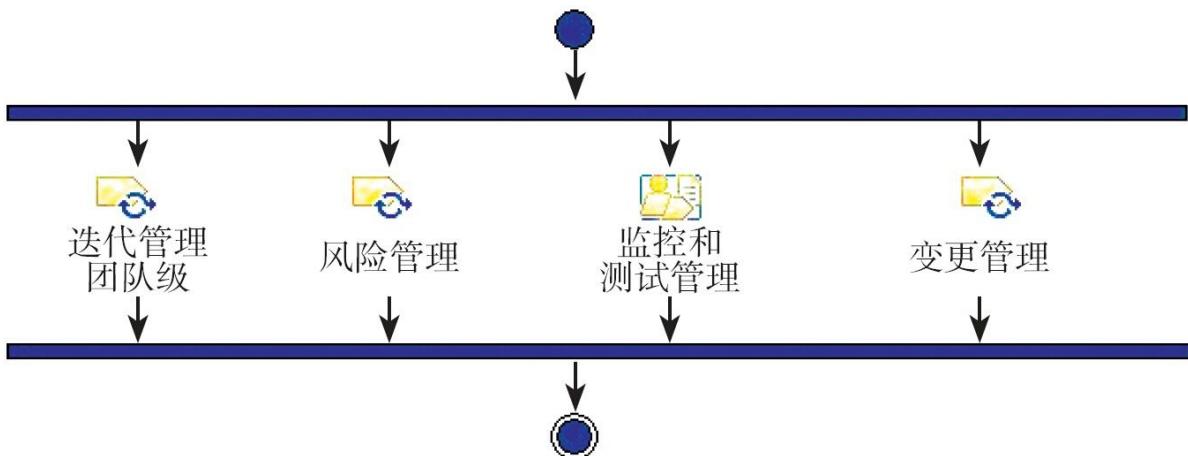


图3-18 构建期迭代内的持续性任务过程图

1. 迭代管理

发展团队其实也是迭代执行的一部分。如何提高团队成员之间的协作和新人的培养，以及如何更好地将“团队建设”融入到项目开发的活动中来，这正是迭代管理需要考虑的问题。所谓迭代管理是一个以目标驱动而不是随着迭代结束而结束的事务，在项目内外，我们都对这个过程予以坚持。

迭代管理事务试图通过系统对项目进度、工作项剩余时间的监控，帮助团队去除“政治化、部门墙”等制约因素，而达到团队的迭代目标（见图3-19）。当团队落后于预期进度时，系统将会报警，此时项目管理团队需要帮助团队评估如何通过减少工作量而仍然可以满足迭代的目标。例如，利益关系人，如产品负责人、团队负责人将在关键时刻参与决策，审批项目范围变更、工作项延期、任务删减的计划和行为。总之，就是通过危机响应，帮助团队解决影响重要目标的关键问题和风险。

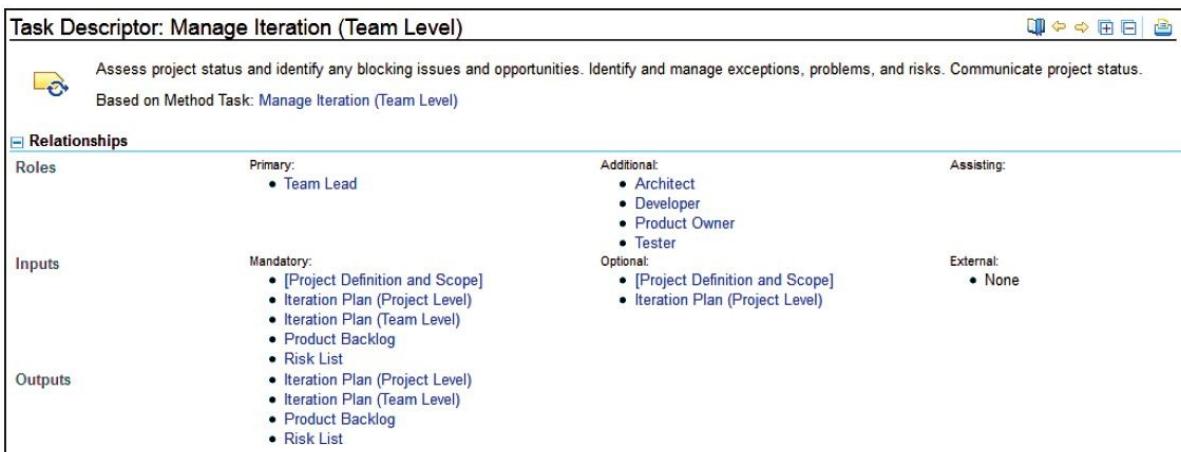


图3-19 展开RMC的任务节点图——持续性任务迭代管理

理想的迭代管理是由系统自动收集数据、自动度量并提交预警报告。系统将通过事先定义的几个度量尺度或者度量矩阵，实时监控项目过程。当然，在不满足全自动化的条件下，也可以靠人工收集和处理，或两者兼而有之。

2. 风险管理

在早期项目的迭代中，我们和团队坐在一起，讨论并创建该项目的潜在风险清单。为了减小风险列表的大小且将注意力保持在高风险上，我们将类似的风险合并，且按照风险对项目的影响和发生可能性由大到小的顺序排列风险清单。要确定风险大小需要估计以下信息。

- 风险的影响。计算如果风险发生，对项目计划进行调整会对日程、人力资源、成本规划造成的影响。
- 风险发生可能性。该风险实际发生的概率（通常以百分比表示）。
- 风险值。“风险的影响”乘以“风险发生可能性”，表示风险的实际大小。

根据风险大小优先排序制定风险应变计划，典型的策略应遵循如下。

- 风险的缓解计划。通过关键问题的解决，减少发生风险的影响和可能性。
- 风险的应急计划。通过预留额外的时间、资源的B计划，以应对风险。
- 风险的回避计划。重组项目，将风险的行为减到零，以消除风险。

在软件开发中，我们制定的风险应对策略如下。

- 为了减少产品X和Y不能整合的风险，需要在项目初期就建立一个原型用以研究整合的难度及风险。设计一系列的测试点（通过列表形式），以确保整合是成功的。
- 为了减少数据库A不正常运作的风险，需要模拟目标软件在并发访问、压力的环境下访问数据库，可能还需要虚拟出数据流，在目标软件没有建成之前实现数据库的验收。
- 为了减少测试工具Z无法有效执行应用程序回归测试的风险，需要在即将到来的迭代中使用它。

风险评估实际上是一个持续的过程，而不是一个只发生在特定时间的过程。至少，应该在迭代或一个里程碑抵达时评估风险，重新聚焦迭代目标时重新评估风险，尤其是以下几点。

- 消除已经完全缓释、不在存在的风险。
- 寻找、发现新近引入的风险。
- 重新评估风险值，重新排列风险列表。

如果可能的话，最好每周都重新审视风险清单，看看它们发生了什么变化。让最靠前的十大风险可以在整个项目过程中保持足够的曝光度、可见度，并坚持采取、执行风险应对的计划和行动。通常，应该将风险状态评估报告附加到周报、月报（如果有）中去。

3. 监控和测试管理

既然采用敏捷方法，为什么还需要监控和管理呢？在企业环境中，适当的纪律和度量是有益的。正如DAD方法中，构建迭代的质量管理和测试工作经由核心团队和独立测试团队负责，显然在不同时期（迭代、软件生命周期的不同阶段）均有不同的测试关注点。为了使测试工作条理清晰、资源充分利用，团队目标的适时调整需要从全局测试的角度进行规划。为了在测试工作中赢得宝贵时间并准确地调整测试策略，需要对测试工作进行监控和管理。

测试工作量的监控。这个任务的重点是监控测试工作的进展状态和工作量，以提高测试有效性为目的进行适时调整。应该评估工作的质量和完整性，并将其用于后续质量评价工作。如果可能的话，使用检查表，以验证质量和完整性是否足够好（见图3-20）。



图3-20 随时更新的测试进展状态报告

寻找积极的迹象，如发现缺陷的消除，或随着时间的推移持续增加或减少的规律。寻找尖锐波峰和波谷的出现点，这表明测试团队可能会遇到的过程或协作方面的压力，我们需要适时给出支持和问题的解决方案。

看看缺陷闭合的趋势。确定当下情况有没有测试关闭待验证不足的问题，或者开发修复问题不充分性的问题。对问题进行量化和分析，例如，开发者标记缺陷为“不可再现（Not reproducible）”而关闭问题的趋势，或“代码按设计工作（work as design）”而关闭问题的数量和趋势等。请注意，在这些问题凸显的时候，关注是否因为过度的开发工作量、或者因为害怕审查工作而采取的行动所致。此外，还要分析修复缺陷又再次引入新缺陷可能导致的后续版本中的问题。总而言之，从趋势分析可以发现团队工作效率降低、个别队员超负荷工作的问题，并据此改善团队的工作状态。

测试覆盖范围监控。这个任务的重点是对迭代、版本测试工作的覆盖范围、测试组织进行充分的监控，以及判断跨迭代的独立测试团队与迭代内团队自主测试的配合是否恰好充分，测试的增量计划是否满足了增量型开发持续增长的需求。

测试覆盖包括功能点覆盖、配置覆盖和测试环境的覆盖，有效地了解测试的覆盖程度需要类似RQM（Rational Quality Manager）的工具，借此设计测试覆盖规划，以及监控测试实际覆盖状况（见图3-21）。

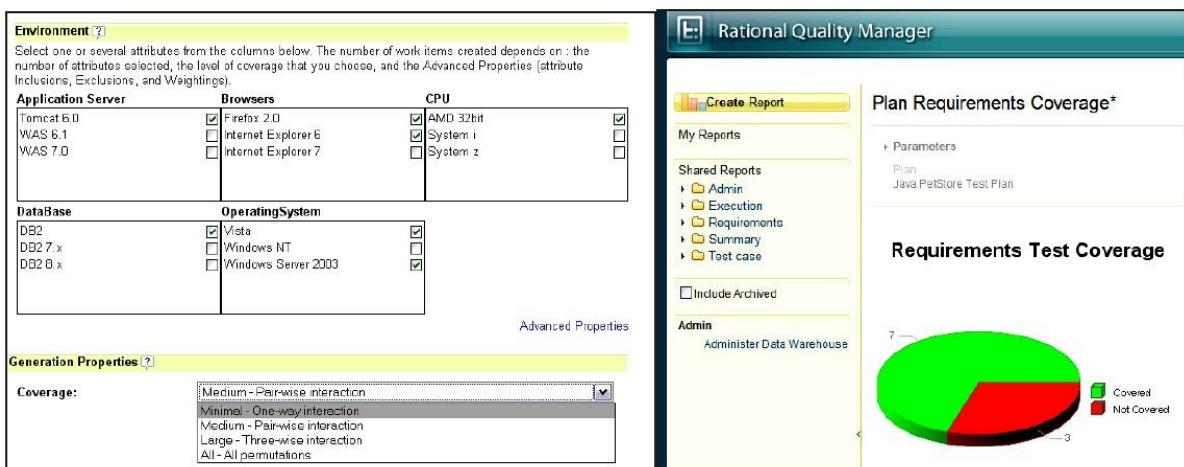


图3-21 设计最佳的覆盖面和最高效的执行路径

将度量测试覆盖率的公式集成为工具，例如RQM和RTC，可以回答“测试完成了百分之多少？多少设计的测试案例还没有被测试过？多少需求和功能点还未和相应的测试用例关联？”等问题。最为常用的测试覆盖度量矩阵是“单元测试的代码覆盖率”。

任何有计划的测试任务至少基于一个测试覆盖策略，覆盖策略指导着测试者的测试用例设计。基于需求的测试覆盖率可能足以产生满足测试完整性的量化评定。例如，如果所有的性能测试要求已经确定，且所有测试执行的结构可以关联需求，则很可能得到性能测试要求的75%已被验证的结论。

类似地，如果应用基于代码的覆盖技术，这种类型的测试覆盖率就可能足以说明系统的白盒测试覆盖率，这也是目前大多数企业所认可的。

这两种测试覆盖策略可能需要人为干预，而最佳且最科学的方式是通过系统自动化抽取数据形成报告，尽量避免引入人为因素。

测试工作个人仪表。个人仪表盘的首要工作是让个人在工作中以最小的代价了解自己的工作进展、合理安排工作计划。个人仪表盘是根据以人为本、团队自我管理的初衷来设计的，将质量和测试监控的度量指标和个人仪表盘的视图规则联系起来，个人就能够更有纪律地完成自己的工作，配合团队和项目的整体进度。每个人也非常清楚自己的工作对整体所产生的影响，了解自身的价值，有效建立自我认同感（见图3-22）。

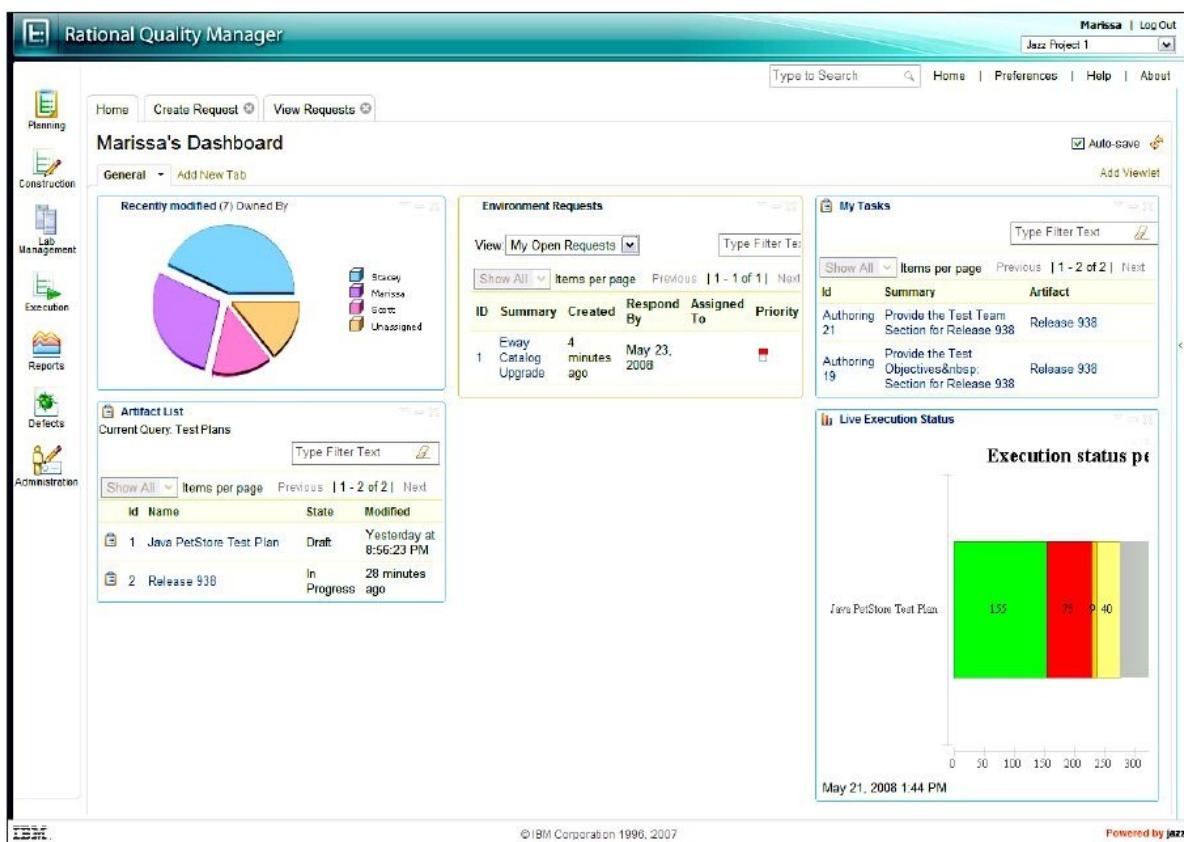


图3-22 测试者根据团队规则和个人意愿设计的个人仪表盘

测试的持续性。在DAD的流程图中，无论是构建阶段的测试还是发布阶段的独立测试都需要在整体的测试策略下，以阶梯式覆盖、节省资源和时间的方式合理安排测试。应当有效地组织已有资源将测试覆盖范围尽可能扩大，如果有时间甚至可以尝试不依照测试用例做随机测试。但是，毕竟我们不可能做完整的测试，需要优选最需要测试的资源，再来考虑测试覆盖的完整性，这就需要对整个生命周期内所有阶段的测试活动采取统一规划。

统一规划不一定要详细到测试用例，而应该依照项目的发布计划对测试做一个完整的规划。例如，我们将产品生命周期分为4个阶段，即初建阶段、构建阶段、产品发布阶段和产品维护阶段（见图3-23），在产品的建设阶段中，展开测试策略的讨论，从测试方式上来看基本可以分成两大部分，即合规性测试（Confirmatory Test）和调查性测试（Investigative Test）。

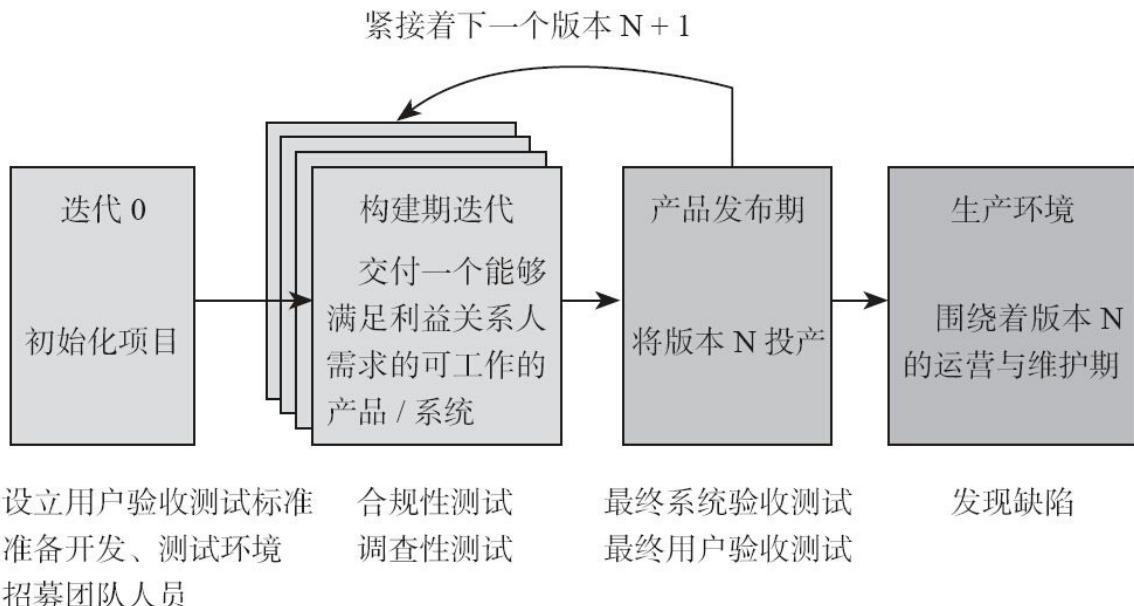


图3-23 Rational敏捷开发流程测试的生命周期

合规性测试的目标是对每个潜在发布构建的有效性和关键功能进行验证。测试者将依据项目的背景和时间合理安排工作，不注重测试交付物，基本不用考虑详细的测试用例和测试脚本的输出，可以简单地使用表格、图片、录音等多种形式描述测试计划和规则。合规性测试所追求的是客户对迭代交付物的“足够”认可：足够证明迭代工作的成功，但可以有些小瑕疵。Rational敏捷开发流程中，测试策略规定合规测试包含了开发人员的单元测试和被称为敏捷验收测试的两部分工作，这两种测试用来保障迭代的必要基本输出，重心放在按时完成迭代目标上。测试者与开发者的工作密切相关，测试者其实可以是开发者自己，这使得测试非常提前，除了参与需求、设计等静态测试的工作外，还有白盒代码级单元测试和“自动化测试”等。在测试交付时间上需要严格要求，测试必须提前，甚至可以做到在代码完成前，测试就已经可以独立运行和工作的程度，使合规性测试和开发工作保持一致的节奏。

值得一提的是，在这个阶段所涉及的测试可能是功能的，也可能是非功能的；但凡是在迭代计划中承诺的，也是对用户需求的理解，哪怕是很繁琐的细节，都要在测试中充分验证。这部分的测试工作存在很大的可变性（可能在下个迭代就被重新改写），测试用例、计划、脚本形式都很灵活。

调查性测试是对合规性测试的丰富，合规性测试关注的是迭代成功，而调查性测试更关注产品的成功，所有调查性测试从整体功能、文档完整、系统测试和与其他产品的整合角度等来验证产品是否达标（见图3-24）。这部分工作在Rational的敏捷开发流程中属于“独立测试”，由“独立测试团队”负责完成。

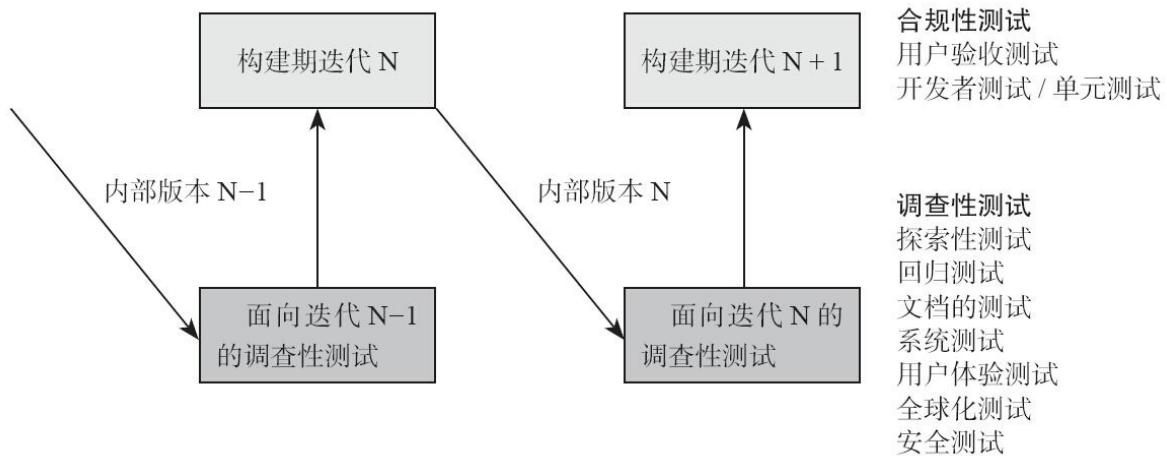
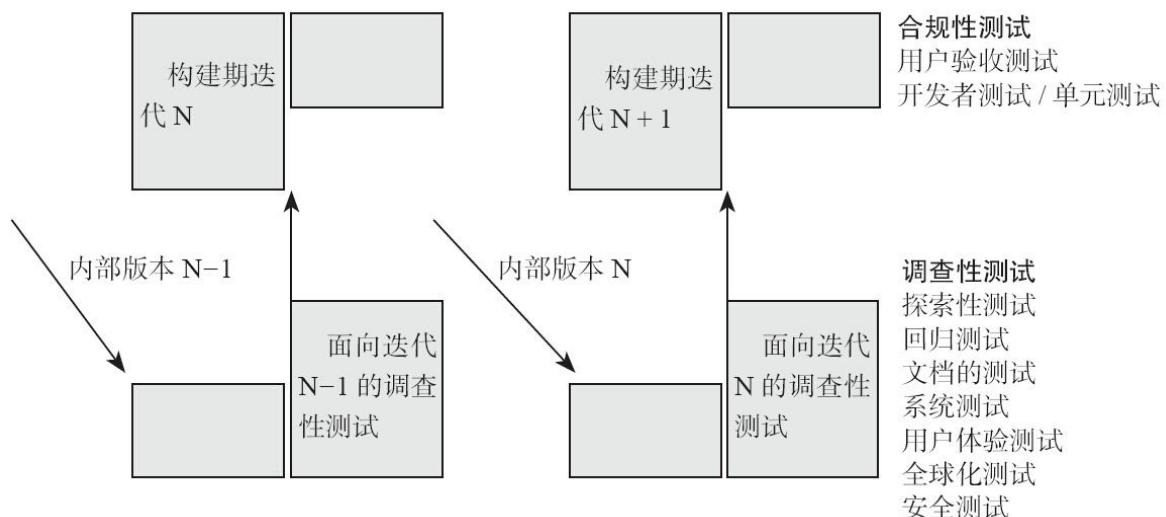


图3-24 Rational敏捷开发构建期的测试流程

调查性测试需要测试计划、用例、脚本，因为这段测试工作周期长、跨度大，包含功能测试、文档测试、系统测试，以及和其他产品、环境之间的整合测试。所以，最好使用工具帮助管理，我们的独立测试团队会使用得心应手的自动化工具，以减少自己在执行工作和人机交互上的等待。除了RQM外，RFT（Rational Functional Tester）可以进行图形化自动化测试，RPT（Rational Performance Tester）可以进行性能测试，RBF（Rational Build Forge）可以进行远程跨多平台、并行、多线程指挥测试机执行测试等工作。

Rational的CCM团队采取的就是这种策略，产品进入8.1*版本后趋于成熟，因为客户的基数非常之大，核心开发团队正向开发-支持型团队转型。CCM团队选择了2周为一个迭代的长度，在每个迭代的第一个周一进行产品开发任务的规划，团队90%的成员都会做开发性工作，例如，选择产品清单中的质量缺陷、客户需求来进行开发、单元测试等，只有10%的人会专注于测试环境的准备、规划和自动化测试的支持以及上一迭代完成工作的手动、自动化测试；第二个周一做调研性测试的规划，100%的团队成员会转职成为测试者，为满足测试规划的要求，对产品做系统的、有计划的测试，在这一周，几乎所有人都在努力验证，除了严重的缺陷（导致产品挂起、宕机或者客户的紧急事件）外没有人会递交新代码。

在Rational项目中，还有些项目团队灵活地调整了测试比重的分配（ASM允许在可变因子的维度上根据因子复杂度调整过程）。例如，RPE团队的独立测试团队是后来建立的，从经验上和技能上都还需要时间成长；团队也很小，与产品核心团队的比例是1：4，所以无法承担这么多的独立测试；再加上独立测试团队位于印度，在沟通上和核心团队存在时差和语言差异的障碍。RPE团队最后决定将大量的调研性测试工作纳入合规性测试，由核心团队主力完成。同时，将模型和流程调整如下（见图3-25和图3-26）。



在独立测试团队初建期，RPE核心团队中的2名开发人员会在迭代中花费近50%的时间转职成为测试负责人，负责独立测试部分的测试用例设计和测试团队新人的培养。因为这个时间还只是个短暂的过渡期，我们没有对流程做更多的定制。

RPE团队的合规测试、调研性测试的范围与Rational的构建器敏捷测试中的两部分测试有很大差异。事实上RPE的测试工作许多都依靠核心团队来完成，因为项目的技术复杂度、行业复杂度比其他产品要轻，核心团队规模15人，团队整体规模约40人，比其他Rational的团队要小。换句话说，项目级的产品、解决方案研发，团队不超过40人的情况下，我们可以将主要资源和投入放到核心团队的建设上；相对地，如果团队规模很大，超过100人，产品研发任务很多，技术复杂度和行业复杂度已经上升到企业级大规模的应用场景，那么我们需要用Scrum of Scrum的方式划分团队外，还需要投入较多资源发展独立测试团队，培养专业的测试人员，加强测试的监控和管理等，因为测试是一项持续性的工作。

4. 变更管理

一个项目从开始就处于不停的变化中。用户需求变了需要调整计划或者设计；测试发现了问题需要对错误代码进行变更；甚至人员流失了，也需要对项目进行一定的调整以适应这种情况。缺陷管理、需求管理、风险控制等本质上都是项目变更的一种。传统的变更管理是为了保证项目在变化过程中始终处于可控状态，并随时可跟踪回溯到某个历史状态。而在敏捷和Rational的敏捷开发过程中，变更管理是保证项目永远可持续化地追寻最高商业价值的产出。

为了保证项目可控，项目领导团队需要充分了解变更的价值，衡量变更实施对项目的冲击，以决定是否要修改。比如，质量缺陷是否严重到必须马上得到修改，缺陷的修改是否复杂到会牵扯很多方面。在项目接近尾声时，团队会启动变更管理严格管控的流程，即未批复的代码不得进入产品，而项目领导团队需要对这个变更做100%的批复，代码才能够递交，这个批复团队的名称为CCB（Change Control Board），由开发、产品、售后、测试、发布工程师代表组成。团队一旦进入CCB控制，即产品即将发布期间，开发团队需要在将私有构建流（private stream）更新到系统主要构建流（main stream）之前，在RTC中相应的代码变更工作项上留下如下几个问题的答案。

步骤一 提交CCB所需问题答案（见图3-27）。

- 这个修复是针对当下客户所遇到的严重问题吗？
- 这是一个质量回退的问题吗？
- 这个修复会改动前台吗？
- 这个修复对应的变更设置是什么？

什么是质量回退（regression）？作为一个软件产品，从一个版本向另一个版本发展，总是会有些风险，某些功能虽然可以在先前版本中工作，却可能在以后的版本中被意外破坏。



图3-27 RTC工作项中描述问题和答案

步骤二 在RTC工作项里创建一个“批复列表（approval list）”，命名为“CCB approval for code delivery”，并且选择一个时间点作为批复的最后时间，系统将在最后时间发送提醒给未完成批复的审阅者，并将所有CCB的批复人录入批复人列表，且等待批复（见图3-28）。



图3-28 RTC工作项中的批复列表

反之，开发者认为此时不应该交付变更所引入的代码，则也需要通过CCB提交延期交付的申请，这也可以通过RTC来完成，只是开发者需要回答的问题有所不同：

- 从客户角度来理解：这是一个对客户影响很小的问题吗？
- 能否提供一个变通的解决方法以降低该影响？
- 从技术角度来理解：这个问题的修复非常复杂吗？会导致质量的回退吗？

虽然变更管理的形式是“控制”，上面的CCB例子看起来也是在“控制”；但DAD的变更管理目的却不是“控制”而是支持，是有规范的、不随意破坏原有成果的过程。针对需求的变更也

是一样，我们需要通过变更管理保护团队不在迭代内产生变更，尽量减少可能推翻前面的工作重新开始的状况，减少浪费。而正是从减少浪费的目的出发，我们的变更管理需要更“快”完成。

回顾1.4.4节的价值流模型，带上“客户”的眼镜来看这个价值流，相对于实际价值工作交付所需的时间，括号内的时间是指在价值流中浪费在等待、整理、文档、会议等不直接输出价值的事务上的时间，显然实际花费的时间超出了价值工作所需的时间。如果将整个软件开发、软件交付的时间轴作为分析参照物，那么开发工作效率只有9%~33%（从客户需求至产品发布：3/（3+30）；从客户需求至产品部署到生产环境：15/（15+30））。

整个Rational敏捷开发过程变更管理的故事，也都在一定程度上重复着这个价值流中的各个环节，而我们调整和解决浪费的方式就是在努力降低等待、整理、文档、会议的时间，这才是和普通变更管理的差异，也是Rational开发过程高效、敏捷的关键。而除了缩短浪费时间外，团队也同时要掌握“延期承诺”的哲学。

何为“延期承诺”的哲学呢？在DAD方法里，团队、设计、需求均在一个迭代内完成，这意味着不是所有的需求都已经清楚，需要甄别和细分需求的过程，在需求完善之前，尽量不做详细的设计、开发、测试工作。这种观点可以归纳为：

- 明确需求前，团队不得承诺将其作为高优先级迭代任务，直至需求细节讨论清楚，方可承诺至计划中。

- 所有的需求产品清单中，优先开发、设计理解清晰的功能点，后开发正在探讨和完善的功能点。

持续性任务是DAD核心构建中的持续性敏捷管理过程，它同时展示了ASM、DAD是如何将企业项目管理与敏捷团队的敏捷性相结合的。总之，ASM可以理解为通过系统自动化渠道的、基于数据分析的循证型管理方式，在团队的管理和纪律性上形成明确的定性和定义，在定性和定义的范围内，团队将有足够的自由来完成设计、开发和测试等工作。

作为补充，我们再展开迭代内部构建的“计划”和“完成故事”两个任务。

在之前的RMC迭代内部流程视图中，单击“迭代计划”即可打开迭代计划的阶段流程图，这个过程是敏捷迭代计划阶段团队从产品清单选择工作项、评估工作量的过程。一般来说，团队会根据环境因素和自身可容纳的最大工作量来评估开发和测试工作。

Rational项目中，一般定义“史诗”是可以包含“史诗、故事和任务”的大颗粒用户故事，而“故事”又可以包括“故事或任务”。通常，在我们的项目中，故事（Story）和任务的区别在于，故事是那些需要源代码递交的任务，任务仅仅是任务，例如文档、管理工作等。“完成故事（Complete Story）”描述了对一个包含源代码的典型性工作项构建的全过程（见图3-29）。

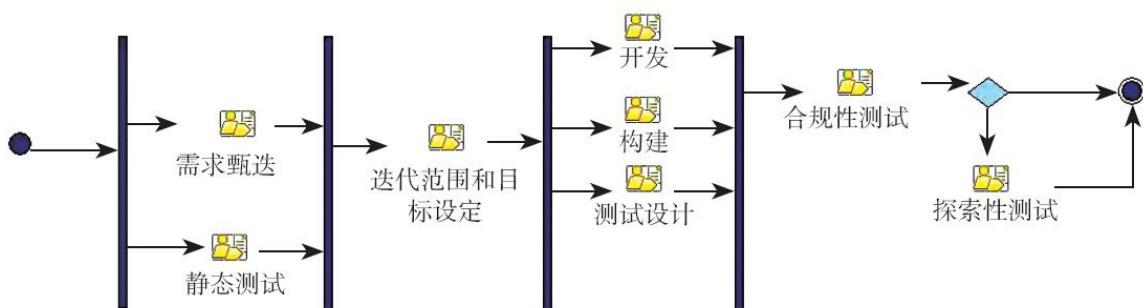


图3-29 构建期迭代内完成故事过程图

当用户故事进入迭代后，团队负责人将领导团队做的第一件事情就是细分这个故事，将故事的需求和目标定义清楚，业务人员和团队的其他关系人将一起探讨设计的细节，这个过

程将持续约1/4个迭代时间，在这个工作完成后，也就形成了团队一致的“迭代范围和目标设定（图3-29中的迭代范围和目标设定）”。紧接着团队开始做增量型的开发、构建和测试工作，直至有一个可以运行的构建，然后便开始执行测试，直至产品发布。

3.4.6 满足客户

就像大多数IT企业所面临的挑战一样，我们需要面对许许多多的，越来越聪明、越来越挑剔的客户。有的客户乐意参与IBM的设计、评估和发布决策，但有的客户不愿意付出额外时间、精力帮助我们完成宏远的目标。在市场越来越多元化的细分发展中，客户期待越来越模糊，同时越来越喜爱简洁的产品，IBM软件部旗下所有品牌都因为自身的辎重巨大而倍感危机。对于任意一个产品，平均每个月、每个国家或者地区的客户会提出10个左右的新需求。在这种强大的内外压力下，在客户和我们没有对产品的使用方法达成一致的阶段，我们仍然要关注产品和方案的发展方向，使产品或者方案能够在增量型研发中最快、最灵活地真正产生最大的商业价值。

如图3-30中2008年的调查报告和右侧的柱状图不难得出：固定投入或者固定时间都不是最重要的，最重要的是我们的团队能够产出高质量的产品并快速发布，要更加关注那些正好满足需求的正确功能，做到比传统开发团队输出更好的“投入产出比”。

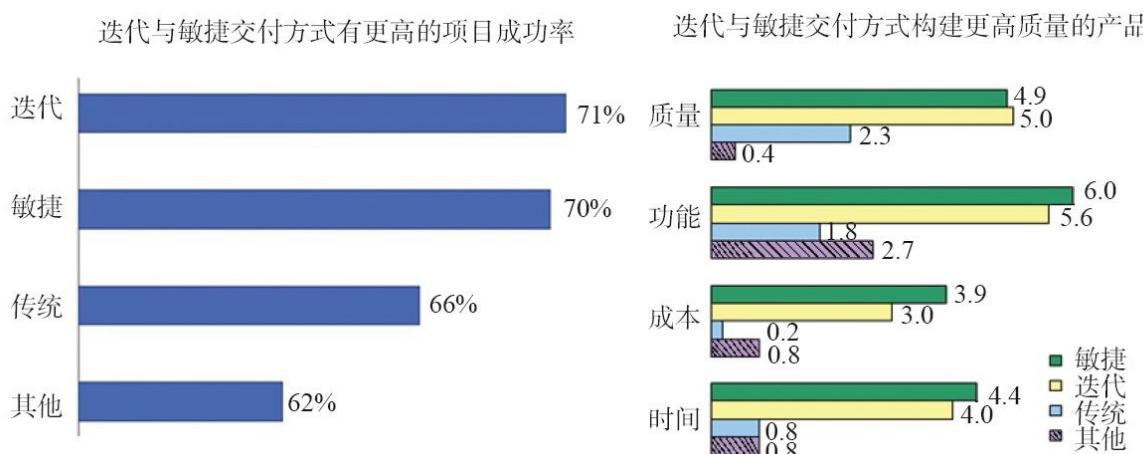


图3-30 Dr Dobb 2008项目成功率统计

从左侧柱状图可以看出，迭代和敏捷的方法使得项目有更高的成功率，而越来越多的人认为，项目的成功已经不仅仅是“对时间、预算的控制，以及产品与项目设计合同的完美一致”。有83%的人认为项目中满足客户当下的需求比一丝不苟按合同执行更重要，82%的人认为关注产品本身质量比时间和预算的控制更重要，70%的人认为将优先级放在最好的投入产出比上比成本控制更重要。

而敏捷项目开发快速、质量高，迎合需求以实现最大化的投入产出，比传统开发给企业带来更多价值。所以，我们和客户不是单从甲乙双方的合同约束来履行责任，而是双方需要培养共患难、同享福的战略合作伙伴关系。

3.4.7 培养客户关系

在后台工作的6年里，我基本上不见客户。我畏惧客户，因为前端带来的需求变更、客户的抱怨总会让我加上一阵子班。我领受的第一个有关于客户的理念就是IBM的价值观“客户为要”。今天，尽管我对客户立场有了更理性的认知，但心中敬畏犹存。

我会对客户的声音极度紧张和关注，面对大量的客户及其个性化的需求，我不得不承认很难平等对待所有客户。从2013年起，我开始涉及许多与客户打交道的工作，工作重点是将客户带入到企业产品研发的需求、设计、开发、测试等各个环节，使客户成为产品的忠实使

用者，愿意将自身将来的成功与公司产品的成功绑定到一起。开始，我负责大中华区Rational旗下所有产品的客户，2013年下半年，我又承担了大区售后Rational品牌的所有客户关系。这项工作颠覆了我的一些最基础的认识，帮助我将对待客户的态度和对Rational未来的思考结合起来，而卓有成效的客户管理也带来了良好的业绩。

从简单地把客户的事情做好而创造价值到管理客户关系，我的工作从满足“朋友的需要”向“将客户看成公司一员的立场”来解决客户困扰、满足客户需求而转变。一次面谈、电话或者为客户解决实际问题，我都理解为是向客户提供着或者新颖、或者改善产品和服务性能、或者定制产品和服务以满足具体细分客户特定需求的重要工作。在与客户的交互中，始终把产品和服务提供给客户的可达性作为目标，在可控风险范围内给客户承诺并严格完成。因为我的一部分工作性质，与客户的不愉快经历时有发生，我的“超理智”思维让我顶住了当时的压力，尽管后来花费了好几天的时间倒空“不愉快”。

经过一段时间的工作，我更加理解客户，他们许多人都在忍耐着比我更大的压力，无论是从内部发出的的合规、业务需求，还是部门关系、舆论、或者来自上层的绩效考核的多重压力，他们表现出的对于乙方的苛刻和低忍耐都是可以理解的。即使换上我们披挂上阵，也未必能像他们那么抗压和从容。而我需要通过我的工作，帮助IBM与客户保持着积极的联结力。

我的工作，事实上也是软件交付全生命周期中对“客户参与”这个原则的具体实现。客户只有通过客户项目才能真正参与并直接影响IBM产品。我们来看看客户项目的内容。

客户项目。邀请客户直接参与到产品生产的全生命周期各阶段中来，帮助提高IBM产品质量和客户满意度。这即是我的工作，在工作中，我挑选有战略合作价值的客户展开深度合作。

客户项目之顾问团项目（BOA）。小规模、动态的组织模式下，重要客户的总监、高级总监将参与指导Rational品牌的技术和业务方向，通过与Rational总经理及其资深领导团队成员的合作，帮助客户了解Rational定位的市场和商业目标，并引导客户提出意见和反馈以对Rational的发展策略以及方向产生影响。

客户项目之Beta项目。通过对早期发布版本的试用和免费培训，客户将获得新发布产品的使用权，了解之、评估之，客户将通过这个项目直接向开发、产品团队提供产品质量评估（质量缺陷、新需求）报告，而同时客户将从使用过程和产品培训中受益。

客户项目之设计合作伙伴项目。设计合作伙伴的项目允许客户参与Rational的产品/解决方案的设计。IBM与选定的一群客户建立长期的战略合作伙伴关系，专注于产品系列版本的设计活动。客户的技术经理、架构师、卓越的实践者将和Rational的技术专家、产品研发的决策者一起，在建立Rational产品/解决方案的未来上提供优秀的设计理念。参与该活动需涉及到常规的小组网络、电话会议。

客户项目之客户声音项目（VoiCE）。通过参与IBM面对面的基于区域的客户声音项目，客户们将有机会听到IBM的产品发展战略部署，并有机会向IBM开发团队提供改进意见，并接触到Rational的广大客户群体。此项目旨在收集客户在产品路线图和Rational产品战略计划方面的反馈信息。

客户项目之倡导者项目（Lab Advocate）。Rational的倡导者项目提供一条便捷的与实验室团队的快速沟通途径，与IBM的一位甚至多位关键领袖、经理或者品牌总监建立长期战略合作关系。倡导者将与IBM客户团队合作确保客户在自身环境中充分发挥出IBM产品的全面能力，很好地理解IBM产品的发展趋势，以及IBM解决方案将如何促进业务的持续成长。客户项目与研发周期的关系如图3-31所示。

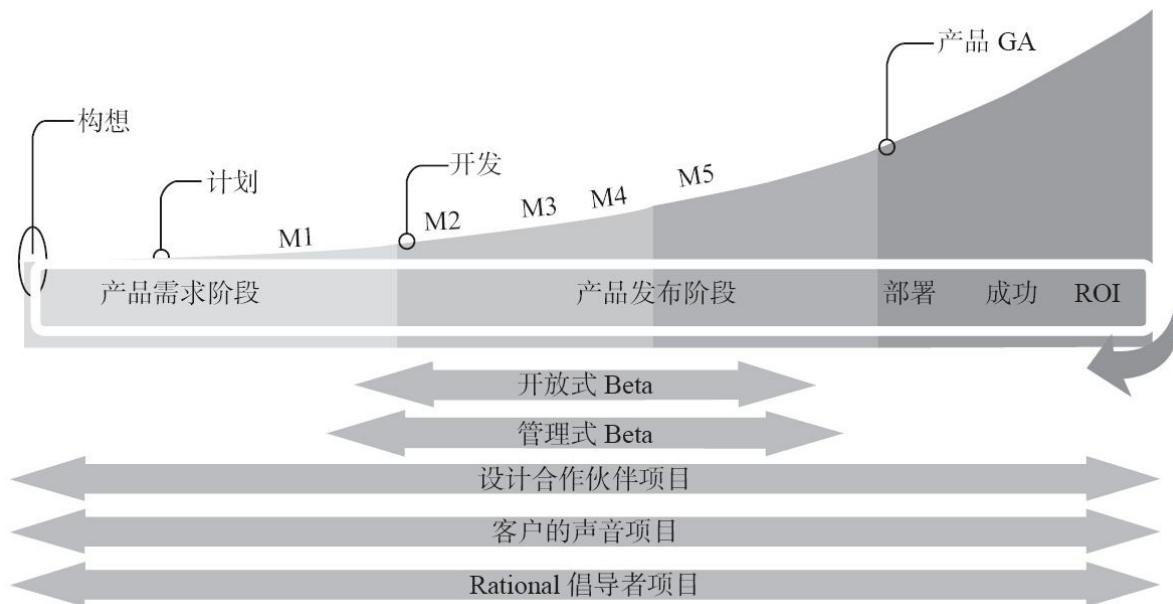


图3-31 客户项目与研发周期的关系

IBM还设有商业价值研究中心，会根据行业专业顾问和市场的预期，对各行业未来趋势进行展望和战略设计。在此基础上，各大事业部也同样有自己的战略研究部门，帮助事业部设计未来产品，例如产品的路线图、投资规划等。

与此同时，各事业部分布在各国家和地区的市场部门也会通过收集产品销售数字，按照区域、产品和行业来划分统计并分析产品的受欢迎程度。每年各地还会举办大小展会，通过新媒体和社会化的普及宣传等市场活动收集客户对产品的直接反馈。

尽管如此，未来IBM的产品更多地还是掌握在客户们手中，顾问规划和市场预期对产品战略的影响是宏观的，具体的产品设计仍然基于客户声音项目、开放式Beta或者设计合作伙伴项目所收集到的客户反馈。客户对于产品的想法、建议、不满都会反馈到相应的客户专员或者经理手上，并将输入数据库系统，从而对其进行跟踪，而产品经理将根据客户反馈内容的商业价值和市场影响进行优先排序。仅2013年中国区一家大型国企为IBM相关产品提供的需求就多达15条，产品咨询和建议达55条。因为产品的全球化，客户数量很大，通过客户项目组渠道收集的需求和建议不小于每月一百条。仅排序、优选且回复客户的工作就会让产品经理忙上一阵，而产品经理最为重要的工作还根据这些不同声音的反馈设计出产品的未来。

一旦客户需求对于产品的影响逐渐降低，或者产品客户经理、客户代表不再积极收集客户的需求，通常只有一个原因，那就是产品将进入生命周期的最后阶段——维护期。IBM是一家对客户负责的企业，尽管没有新的改进计划，但对于产品的维护和各种服务仍然会持续3~5年，并在此期间为客户提供换代产品和版本的升级服务。

采用这种做法的还有知名的国内企业华为，带领华为的任正非总裁曾经说IBM是最好的老师，但是在我看来华为已然成为传奇，这家企业甚至比IBM更加珍视客户。华为的企业文化中，第一条也是“以客户为中心”。“华为作为一家百分百的民营企业，26年来生存不是靠政府，不是靠银行，客户才是我们的衣食父母。”华为的第五位员工，现任三位轮值执行董事之一的郭平接受采访时如是说。

华为可以因为一通电话就立刻派工程师到达现场，与客户一起解决问题，不像其他企业为了节省成本，多半用远程视频遥控，即使这个产品是在乞力马扎罗火山出现了问题，华为也毫不犹豫。满足客户需求和客户服务到位是华为胜出的关键，华为是一家典型的会为了客户利益甚至牺牲自己一到两年短期利益的企业。能让员工甘心为公司、客户卖命，除了配股分红的激励机制外，也与华为强烈的企业文化有关，“你们脑袋要对着客户，屁股要对着领

导”，这是任正非反复对员工说的话。他认为，大部分公司会腐败，就是因为员工把力气花在讨好上司方面，而非思考客户需求。

3.5 故事分享：Rational中国敏捷开发者的一天

3.5 故事分享：Rational中国敏捷开发者的一天

这是一名普通开发人员一天的工作和生活，它真实地反映了团队使用了敏捷研发方法后可持续化的协同开发过程。

·主角登场

我是华英雄，IBM开发中心某著名产品的非著名开发工程师，早上7：30送孩子去幼儿园，8：00和太太在家自制Home早餐，9：00电话开机，10：00坐在座位上打开电脑。

·紧急通知

10：00，第一件事查邮件（见图3-32），发现第一封紧急的邮件来自系统通知，是美国团队昨晚交付代码后，CI系统自动编译和制作build的过程出错了。

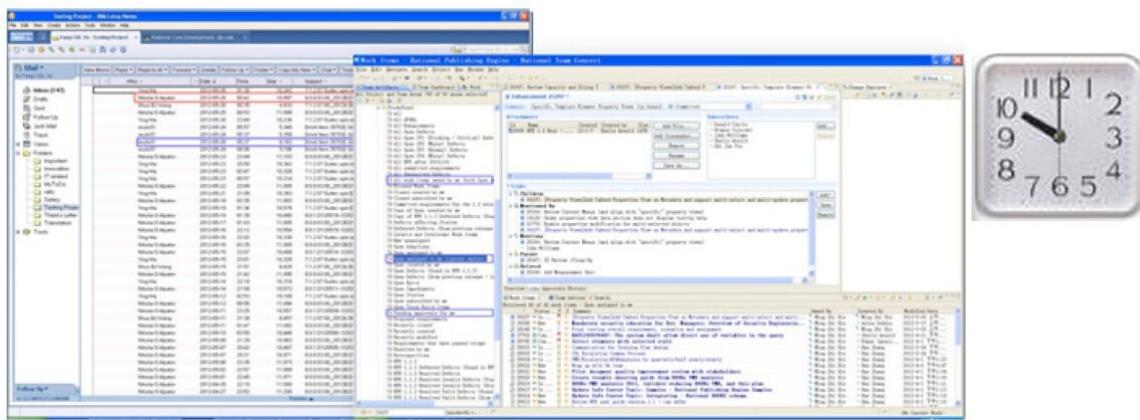


图3-32 华英雄打开邮箱发现来自RTC的紧急邮件

·跨时区协同

现在是美国时间22：00，立刻打开Sametime，还好Jared在线，让我跟他沟通下，嗯很好，Jared说这个问题他也收到了系统通知，正在处理，应该很快解决……30分钟以后，Jared告知编译问题解决，简单道别后Jared便离开休息了。

实际上，在此之前，也就是在华英雄还在家中的7：00，自动化部署和编译、自动化测试的持续化集成系统已经在运行了（见图3-33）。

·永不疲倦的CI系统

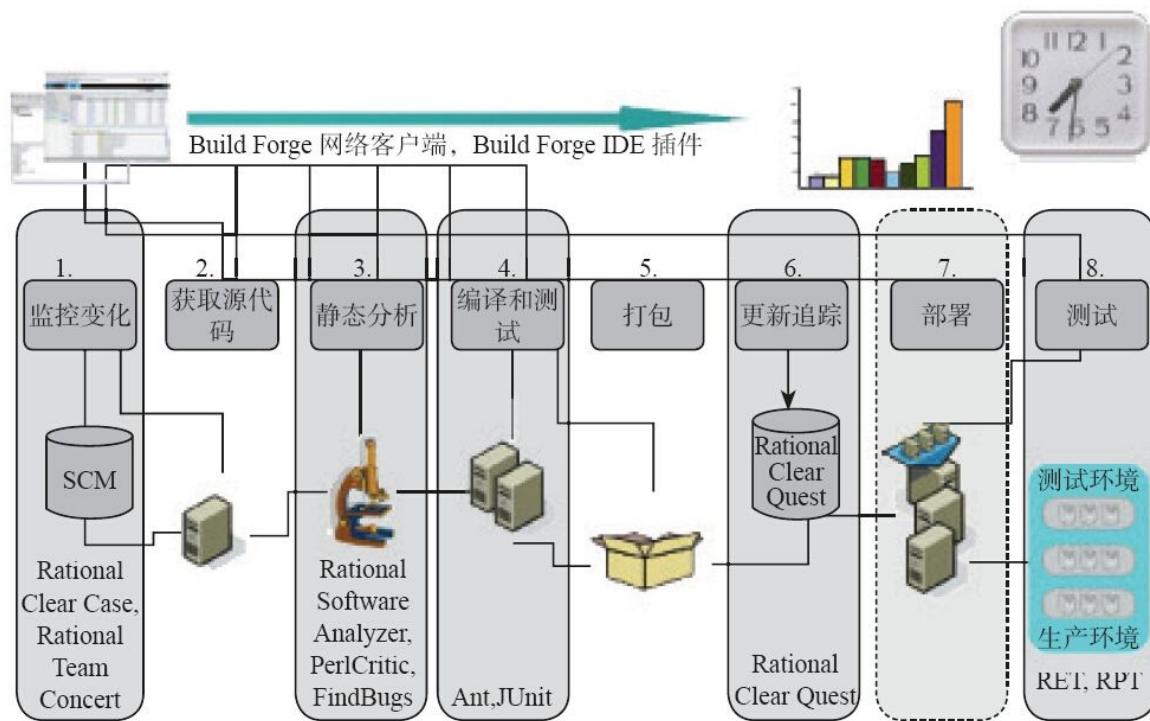


图3-33 CI系统可以确保自动化构建、编译和发布

·Todo总会有“人”提醒

10 : 15，在Jared解决构建问题时，发现有一个代码审核工作项需要我完成，这个工作项的描述里提到了我，需要我的响应。还好都不紧急，让我开完例会再说。还有一个测试用例设计的任务需要我完成，测试大使（testmister）需要在迭代最后一个星期确保我在RQM里为新功能写好测试流程和测试点。这个没问题，让我在RTC的工作项里建立一个测试用例设计的任务，然后设一个期限，这样就能够收到系统的邮件提醒，不会因为错过时间而耽误迭代的测试工作了。

·制定计划

我们鼓励开发人员自己选择工作项，并且制定迭代计划，而一开始我们一起制订了版本发布计划（Release Plan），这和个人的迭代计划有很大区别。版本发布计划相对高级、抽象，而迭代计划（Iteration Plan）则非常具体、详细（见图3-34）。

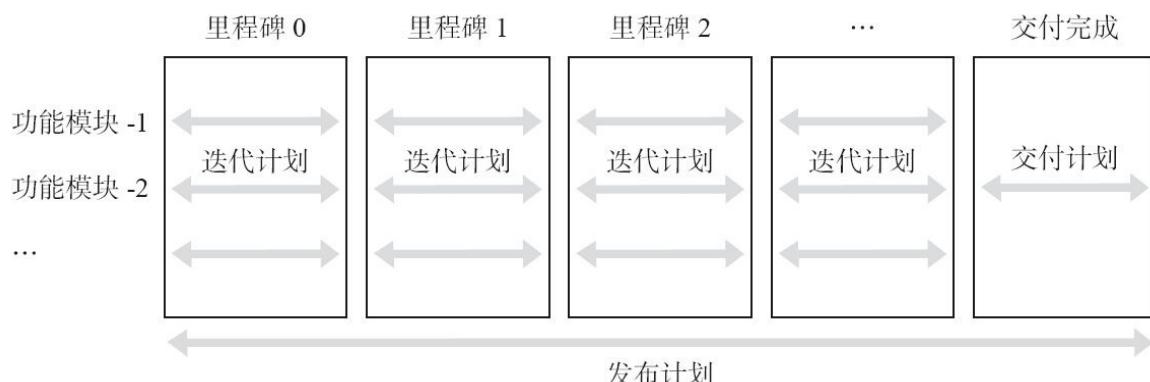


图3-34 版本发布计划和迭代计划

·效率的会议

10:30, 小组开了个本地团队的Scrum例会, 大家分别分享了几个主要问题: 你都做了什么, 你将要做什么, 你现在遇到了什么问题。轮到我了, 我告知团队负责人: “我今早和Jared发现代码交付出了问题, Jared刚解决, 大家可以放心地接受美国团队的代码提交; 我还要做一个同事的代码审核, 以及回复一个工作项关于需求的讨论; 剩下的时间我都会集中在Multi-template editing的工作项上, 这是个很重要的工作项, 目前有个刚生成的子工作项记录了关于数据源view与Editorview的不确定问题, 我想在开发核心代码的同时和架构师Allan于今晚继续讨论。另外、这周我是Jazz论坛的新闻大使(Newsmister), 我今天将花两个时段去回复客户的问题。最后, RTC刚刚发布一个补丁, 但是我不确定如何下载和安装, 这个可能会花费更多的时间。”(见图3-35) 团队负责人马上回复: “华英雄, 我希望你能更好地利用自己的时间, 我会和服务部门的Terry了解下正确的升级步骤和规则, 我将把这个问题记录在我们的问题列表里, 希望明天能够得到满意答案并尽快升级成功。”

10:45, 散会, 回到座位上打开需要我回复的工作项, 写回复, 因为是回给美国人, 不在同一时区, 所以尽可能地把我的想法写得详细些, 希望尽快完成讨论, 达成一致(见图3-36)。

·只做最需要做的审核

11:00, 开始代码审核, 我打开团队负责人放在RTC仪表盘(Dashboard)上的代码审核检查条例参照表, 用RSAR和RTC的工具联合做了静态扫描, 嗯, 没有问题, 再打开代码视图, 逻辑正确, 判断条件比较全面, 目前没有发现什么问题, 好, 我给他批准通过(见图3-37)。

Backlog

Closed Items: 6 | Open Items: 53

- Provide project level read permissions
- Load Scalability improvements
- Provide RESTstyle APIs and resource formats for work items
- Add a Visual Studio Client
- ▶ • Support to bundle the RTC components in different ways
- Remove dependencies on the Eclipse SDK
- Project restructuring on the same and across servers
- Dashboard support that viewlets can show data from other repositories
- Work items functional parity between eclipse and web UI
- Provide a web UI for the RTC build functionality
- Extend the available process preconditions, follow-up actions
- Support planning in the web UI
- Support process configuration in the web.
- ▶ • Reports administration/deployment in the web UI
- Web UI User Assistance: context help, local help for the web
- ▶ • SCM: Better control of what & where content can be loaded
- Improve agile planning support
- Improve work item customizability
- Support reports across project areas
- Support work item queries across project areas
- ▶ • SCM: Logical model support
- SCM file locking
- ▶ • SCM: Cross repository deliver
- ClearQuest connector set-up improvements
- Provide ClearQuest Importer
- CC importer
- Query editor usability improvements
- Add support for higher ceremony process templates

图3-35 典型版本发布计划视图：RTC的发布计划

不一会儿，同事峰从座位上挪过来，表示已经收到系统通知，他刚刚做了代码提交。

·成为新闻大使

11:30，打开Jazz论坛，喔，有三个客户给我们发了问题，让我看看，第一二个问题很好回答，让我来回复一下……第三个有点难度，让我问问团队负责人有什么好主意。5分钟后，没有答案，看上去不是RTC的问题，倒像是RPE的问题，让我发给Alex，他是RPE开发经理，请他帮我回复，好，邮件发出去了（见图3-38）。

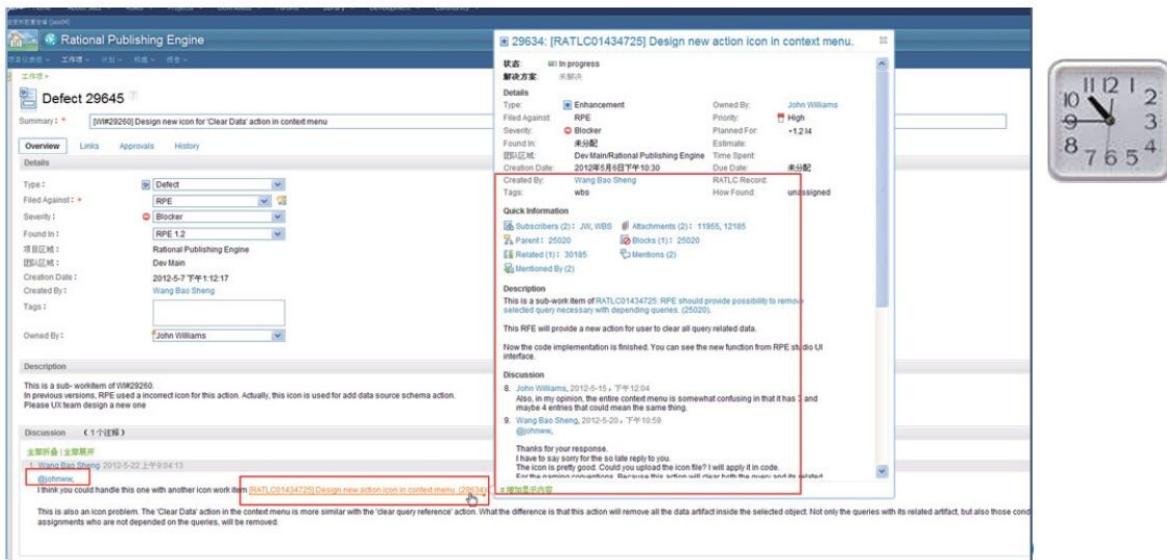


图3-36 协作平台上的异步沟通

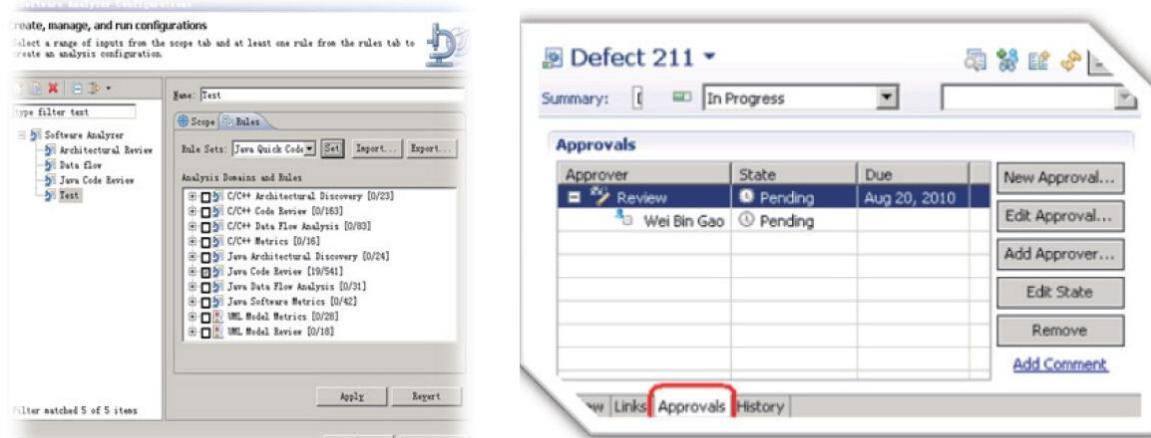


图3-37 使用协作平台工具辅助代码审核

·午休

12:00, 午餐。

13:00, 重新回到座位上。

·开始代码工作

13:10. 来杯台湾的蓝山咖啡，放松放松，嗯，开始代码了。

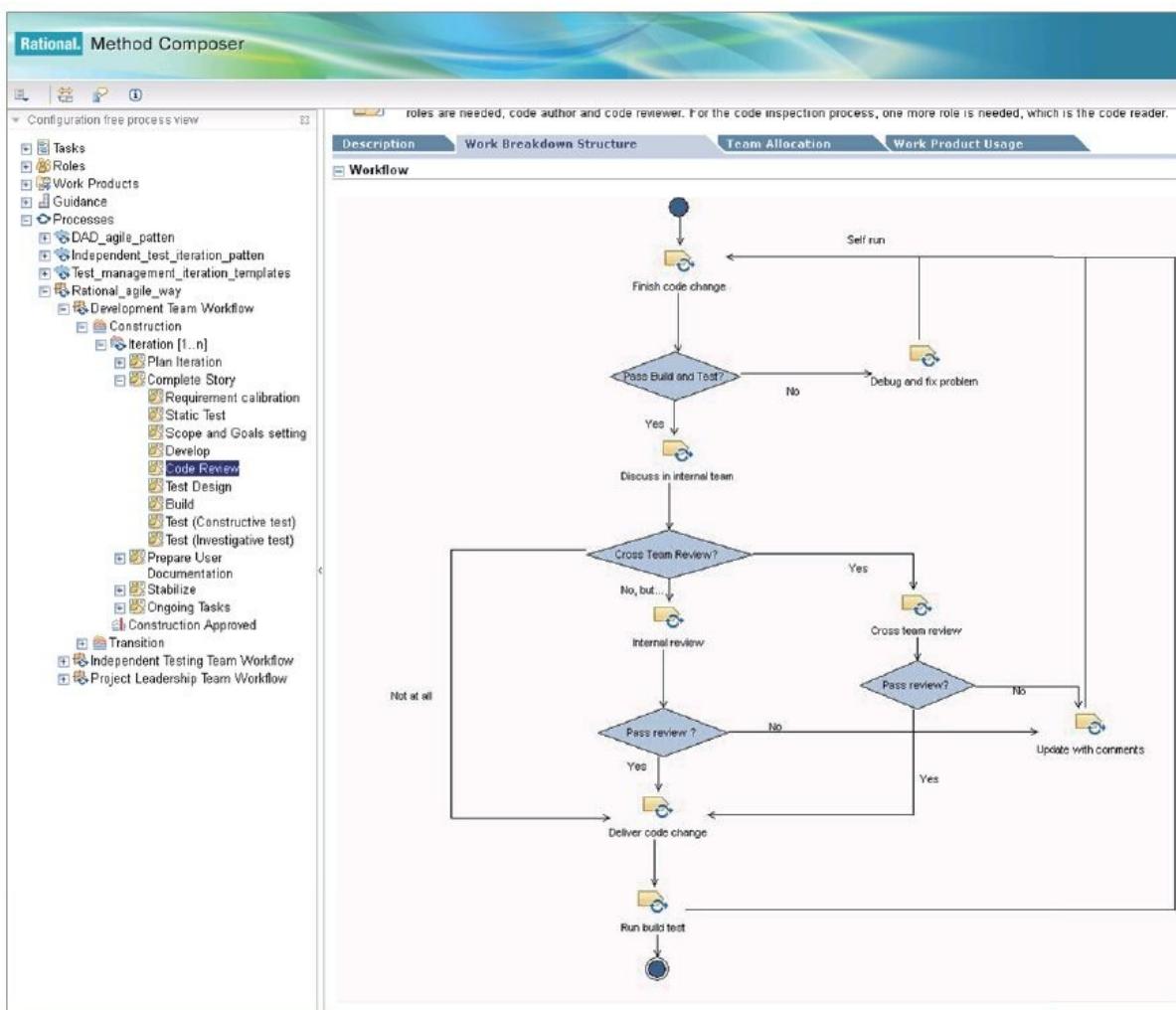


图3-38 RMC下的Code Review工作流展示

2个多小时过去了，现在是15：30.....进展不错，除了Editor和Datasourceview还没有完成，基本所有的属性视图、编辑视图、工具箱视图都可以通信了，加载/保存也都能够正常工作。再运行一次单元测试，不错，所有脚本都返回正确，再局部使用RSAR扫描代码，很好，没有新增加警告或者异常，我提交代码吧。（需要补充说明，持续化集成工具将自动于19：00开始集成代码生成新的安装包，并自动开始更全面的代码级安全、版本控制、静态分析等扫描，自动化开始Build Verification Test并上传测试结果。）

·更新工作进展

16：00，休息一下，顺便在RTC的Multi-template描述中记录今天做的工作，将工作时间减去3小时并记录在RTC剩余时间项上。

·学习和成长

16：10，我决定去师傅Gary那里走一走，看看能否向他了解一些有关GEF、GMF技术的重要细节，以免将来在进一步开发中出现纰漏，顺便也要请教下其他几个小问题。

·突然增加的会议日程

16：30，我回到座位上，倒了一大杯水，一边喝水，一边欣赏窗外的风景。这时团队负责人线上找我，问我是否有时间和他一起参加独立测试团队的系统测试用例审阅，我同意了。（全团队概念下的开发人员不仅仅关注自己的工作，和测试人员一样，他们是一个团队，关注的是让产品更快更好地进入市场。）

17:30, 回到座位上, 收到测试部门的一封感谢信, 心情很好。我在RTC里测试用例的任务下加入了一项“审阅系统测试部门测试用例”的子任务, 然后标记为“完成”。

·社团工作

此时, 又收到了Elizabeth全球敏捷社区负责人的一封邀请函, 邀请我参与9月的一个号称24小时的敏捷大会评委组, 我兴致勃勃地回复Elizabeth, 表示对此次大会一定竭尽全力。

·下半部新闻工作

17:40, 我再次打开Jazz论坛的主页, 又有一个客户留言咨询工具与RRC的整合方法, 这个问题再熟悉不过了, 我即刻回答。这个时候, 也收到了Alex更新论坛的消息, 他已经帮我做了回复, 道过谢, 我起身准备离开公司。

·返家

19:00, 到家时, 父母已经准备好了可口的饭菜, 一起享用美好的晚餐。之后, 带着孩子读书、弹琴、拼拼图。呵呵, 每一分钟都是那么开心。最后给孩子洗个澡, 准备让孩子休息。

·跨时区的联结

22:00, 孩子应该睡觉了。孩子妈妈哄着孩子上床。我, 华英雄, 还是回到书房, 打开电脑做今天要做的最后一件事情, 与架构师讨论一个问题。这时候Allan已经在线上了, 我们就编辑视图的不确定问题进一步讨论……

·一天的完结

23:00, 关机, 休息!

第4章 初创企业的敏捷转型

第4章 初创企业的敏捷转型

4.1 初创企业的敏捷思维

4.2 以客户为中心

4.3 如何解决人事危机

21世纪伊始，在互联网/移动技术兴起的大潮中，企业的经营者已经寻找和开发了属于自己的管理方法和产品导入方式。10年过去了，互联网大潮造就了数个十亿级、百亿级的新型企业。腾讯、阿里巴巴、京东、小米如今已是名声显赫的弄潮儿，其影响力和声势甚至超过了苹果、Google、微软和IBM等千亿级公司。今天，已经有很多作者和作品在介绍互联网行业的新兴概念，其中，企业管理领域出现的一些卓越管理模式和精益管理模式，与传统而经典的德鲁克管理思想几乎并驾齐驱，成为中小型企业特别是初创公司的至宝。而已经成为企业首选，特别是初创企业首选的产品导入方式——敏捷开发（一种增量的迭代工程交付方法），可以让产品或者服务的开发始终专注于最高价值，积对应对客户和市场反馈。

这里的初创企业不包括两种，第一种是依靠现有优势资源，倒买倒卖发上一笔横财，然后出售企业的；另一种是满足于山寨或者克隆，没有核心竞争力，不断跟在别人后面走的企业。我们要谈的是那种有自创品牌，发展着自己的核心技术，极致地打造产品的创新企业，即使面对金融危机、市场疲软等诸多不利因素，这种企业也可以通过精品战略存活下来。

史蒂夫·布兰克与鲍勃·多夫在他们的《创业者手册》中指出，所有创业都坚信自己的道路与众不同，然而事实并非如此。成功的故事都是一样的，世上必然存在一条通往成功的真实且可重复的路线图。而成功的创业者同时需要善于抛弃传统产品管理和开发流程，善于结合敏捷工程和客户开发方法，以不断迭代的方式建立、测试和寻找商业模式，从而实现从“未知”到“已知”、从“不确定”到“确定”的转变。

4.1 初创企业的敏捷思维

4.1 初创企业的敏捷思维

对于初创企业来说，如无必要，勿增实体，秉承极简主义，即最简单的就是最好的。极简主义也是乔布斯最为推崇的产品设计模式和组织管理模式，乔布斯认为他无法记住100名员工的姓名，他喜欢的工作模式是可以关照到方方面面，而如果组织超过100人，则将不再是称心如意的工作环境。从初创企业的角度，我们非常认同这个观点，如无必要，不要将企业组织和产品设计模式做大。

4.1.1 使用瀑布式的初创企业必亡

传统的大型企业使用的是瀑布式生产模式，而且为了精细化流程，确保质量和对生产线的严格管理，一件产品需要经过成百上千道工序，这种流程复杂、分工严密的传统管理模式，是由著名的管理学家弗雷德里克·泰勒提出的，后来经福特的实践推广开来。标准流程对工人素质要求不高，只需要他们简单地重复即可。这样一来，可以大幅度提高工人的工作质量和效率。传统的软件行业生产也正是取证于此，使用瀑布式模式进行软件开发，每个角色都有严格的分工和定义，他们所要做的就是加工和生产已定义好的内容。

但是，瀑布式开发有很长的生命周期，从原始构想到投入工作，从设计和需求的确定到测试结果的交付，一旦开始了瀑布式开发，就需要用一个信念支撑整个周期：“我们很清楚客户需要什么”，接着传递这个信念和构思，大胆地设计并要求研发团队按照图纸——完成，从此永无回头之路。而为了完备地交付工作，公司需要成立各个职能部门，将研发周期相关的所有活动和责任尽可能清晰和细致地分配给各个团队。这又增加了管理和人力成本，同时局限了团队的“主动思考”。无论管理层或投资者，都将不再关注“客户需求”而是高度强调发布日期的重要性，并对造成延误的“错误行为”严惩不贷。对于初创公司，这往往就是致命的错误，因为企业创始人一开始灵光一现，就将这些想法写了下来，并努力将其变成“核心思想”灌输给团队，而当他们遇到反对声音的时候，通常不会放弃或者重新思考。创业者通常有很强的控制欲，摒除一切反对声音也要“前进”。这使得初创企业的团队愿意妥协并相信和跟随创始人，“你说干嘛，我就干嘛”。市场部会收集有利的证据，而主动忽视自己的真实想法；研发团队已经不再回头，每天谁打扰我完成手上的任务，谁就是和公司过不去。所以，大家根本不去怀疑“客户需要什么”。当产品即将发布时，董事会和创始人约定好继续投资做渠道和市场，大笔资金开始流向媒体、渠道商和各种铺天盖地的小广告；最好，还要聘请公关公司，做一个“首次客户体验的”线上线下活动，再加上记者招待会。产品交付的日期终于如期而至，可是，此时我们的创业者才首次从客户体验的结果中读到了“盲目乐观，不符合实际”的差评。显然，每一家初创企业或公司都希望把产品推向市场并展开销售，但若不了解销售对象以及对方的购买原因就盲目行动，仓促扩张，结果肯定“凶多吉少”。强势的瀑布式开发模式忽略了迭代的作用，失去了寻找可复制、可升级和可盈利的商业模式的过程，单从一腔热血出发就在未经验证的初始假设上执行方案、交付产品或服务，这样做无异于自取灭亡。

初创企业绝对不是大企业的微缩版本，初创企业与成熟企业、大企业最关键的区别在于，后者往往已经明确客户群体、产品特征、市场范围和竞争对手；而前者有的是明确的愿景、假设的客户群体、未经验证的商业模式。

过于纷繁复杂的流程和制度必定扼杀员工的创造性和积极性，流水线的生产方式将导致大家各自为政、没有协作精神、关注内部而不是外部需求，在产品中心化思想即将淘汰的今天，如果产品不能有效地服务，不能打动客户，那么初创企业便将面临极大的挑战。

显而易见，那些将“顾客是上帝”的口号高高悬挂的企业，事实上无法抵御内部“事不关己高高挂起”的官僚作风，而只能昭然若揭地自欺欺人。只有真的把客户需求当回事，才能不断发掘客户信息，寻找并明确商业定位，比起商业计划，一些草根客户的意见可能更有价值。

在研发中，尽早将真实客户的体验和试用报告反馈给研发团队，这种做法表面上会带来额外的成本，但避免了因为错误开发和理解差异而导致产品失败的巨大损失。

那么，减少一些流程、降低一些要求，或者减少一些分工就可以成功了吗？理论上，风险小一些，但是仍然以产品导入为主的按部就班式方法，只是跟踪相关资源的消耗、过分关注里程碑的传统方式，即使所造成的错误可以挽回，实际上对一个产品就决定一家公司前途的初创企业或公司来说，也有着致命的缺陷。

4.1.2 开发新商业模式

每个初创企业都坚信自己的道路与众不同，但有的能够成功，有的只能沦落到廉价清仓的下场，虽然运气起了一定的作用，但是好运好比“东风”，只有万事俱备后，东风到来时，才能带来好成绩。等风到的过程中，最重要的事情就是在不断迭代中验证和推演商业模式，以及使用敏捷交付方式交付产品和服务，从而实现可盈利的转变。

在导入产品之前，初创企业最早思考的应该是商业模式。商业模式是初创企业创造价值、传递价值和获取价值的基本原理，同时也是分析组织和竞争对手、构建新战略的共同语言。如果对画布法展示商业模式的思路感兴趣，请访问商业模式中心 www.BusinessModelHub.com（英文）或 www.BizModel.org（中文）。画布法是通过客户细分、价值主张、渠道通路、客户关系、收入来源、核心资源、关键业务、重要合作、成本结构9个模块来表述商业的4个主要方面：客户、提供物（产品/服务）、基础设施和财务生存能力，参见图4-1。商业模式就像一张战略蓝图，可以通过企业组织结构、流程和系统来实现。

重要合作	关键业务	价值主张	客户关系	客户细分
有些业务需要外包，而另外一些资源需要从企业外部获得	关键业务活动	通过价值主张来解决客户难题和满足客户需求	在每一个客户细分市场建立和维系客户关系	企业或机构所服务的一个或多个客户分类群体
	核心资源		渠道通路	
提供和交付先前描述要素所必备的重要资产			通过沟通、分销和销售渠道向客户传递价值主张	
成本结构		收入来源		
由上述要素所引发的成本构成		产生于成功提供给客户的价值主张		

图4-1 画布法的9个模块即商业模式描述

对于商业模式的探索，首先，在特定的环境中需要深入理解和审视周围的变化，分析日益复杂的经济环境（如互联网化的商业模式），以及更多的不确定性（如创新技术的发展和市场的混乱），还有经济危机和免费的思维。理解商业环境的变化趋势能有效地帮助我们适应不断变化的外部因素，我们要从市场影响因素、行业影响因素、重要趋势、宏观经济影响因素等方面深入分析，并且可以利用互联网检索文件或者借助有效的分析工具。在选择商业模式的前提下，选择创始人比较熟悉的行业和技术领域，这样的初创企业成功率较高，只有对环境比较熟悉，对商业环境的洞察和分析才比较准确。

简单的可执行、可复制、可规模化的商业模式便是好模式。但需求存在未必代表客户愿意付费，例如有免费邮箱，推出收费模式的263邮箱只能退出这个市场；或者有很多流量的团购网站，当商户提高价格以后，网站的粘滞度为零，所有人立刻远离。这就需要商业模式的验证（参见图4-2），从理论上验证其合理性、科学性，尤其是对成本构成、渠道、收费模式、定价的验证，以及对如何将投资风险降到最低的验证，听着都会幻灭的商业模式自然不算好模式。然后要尽早接触用户，在用投资人的钱烧流量之前，确认这种做法可以被接受。

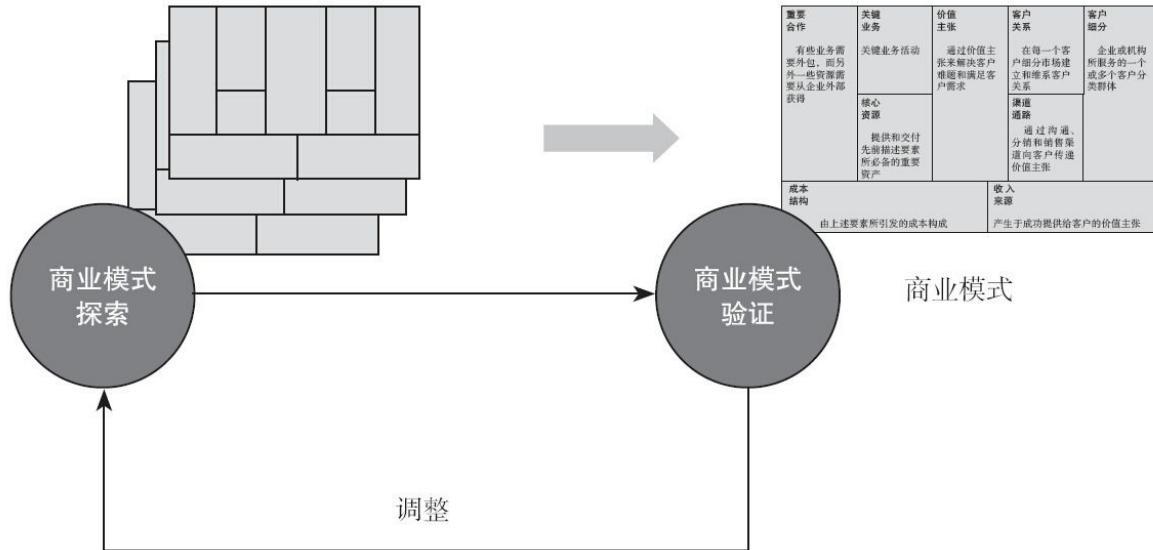


图4-2 应用画布法展示商业模式的动态调整

例如，我们开发一款移动社交产品，起初希望服务于大龄未婚人士，通过线下体验，发现20岁出头的少男少女更愿意为此付费。所以，必须更改商业模式，渠道建设、主要服务人群、价值主张等先前的假设都得推翻。其实，失败本身不可避免，初创企业恰恰就是在试错中生存下来的。这也注定了初创企业始终在走两步：退一步，验证且推翻原来的假设，修正再验证，直至商业模式被证明为可复制、可升级、可盈利；迈一步，之后，我们才可以投入大量的金钱和时间进行规模性开发。

4.1.3 以客户为中心的价值主张

商业模式中，“价值主张”是客户转向一家企业而非另一家企业的原因，它解决了客户困扰或者满足了客户需求。每个价值主张都包含了一系列可选产品或者服务，以迎合特定客户细分群体的需求。在这个意义上，价值主张是企业提供给客户的收益集合，所以最先需要考虑清楚的就是价值主张，即我们应该向客户传递什么价值？

请注意，这里的关键词是价值，价值的中心是客户。我们是帮助客户做好某些事情而简单地创造价值，还是通过客户使用和展示某些特定品牌而发现价值？或者，帮助客户削减成本、降低和抑制风险，从而创造新价值？

有人说，我们去了解客户需求吧。No！我想说的是，先了解客户，客户才是核心，需求的准确程度取决于对客户理解的准确程度。

通过一段时间的接触和对困难的研究，我发觉客户往往不甘心被忽视，也不愿意屈服于产品规则，他们真正关心的就是自己。任何传统软件、硬件、集成商的“以产品为中心”的策略和方式，在今天，无论提供多么“友善的服务”，还是无法确保客户因一时兴起而挥袖离去。80后、90后的崛起，让“客户”这个概念发生了很大的变化，他们需要的不是更好的服务或者产品，而是参与感、存在感。

客户有“脾气”的时候就像一个被娇纵的孩子，他们认为自己是“中心”，是唯一的“重要人

物”。但不得不说，这样的客户往往很清楚自己需要的是什么。因此，他们表现得对任何其他事情都没有耐心，仅仅希望立刻得到想要的东西。如果未能把握机会满足他们的需求，他们会毫不犹豫地离开。

客户越来越不能容忍以产品为中心的模式，越来越需要以顾客为中心的模式，这种模式的关注点不在于以何种方式开发和交付产品，而在于对客户真正的需求的深切理解，考虑顾客愿意用什么方式来获得和交换。

传统的开发模式太重视资源、时间、范畴，而这些一定会惹恼客户。也许企业拥有先进的技术与雄厚的资金，拥有在地域维度远离竞争对手的得天独厚的优势，拥有当地的大部市场份额，但是时间和空间的瓜分逐渐被无处不在的互联网、大数据等先进技术打破，固有的优势会很快消失。

虽然以产品为中心的优势和主体没有分崩离析，但是源自其内部和基础的腐蚀已经摧毁了以产品为中心的优势，难以适应未来的需求。世界其实很小，今天的发明天就会被仿造，后天就会充斥大街小巷。无论是国际公司还是本土崛起的企业，产品的竞争优势将在短短几个月内甚至数周内就消失殆尽，如果没有做好变革准备，那么请准备变卖公司吧！

企业的价值主张不可独立研究，在不断开发商业模式的迭代过程中，我们会通过接触客户来验证所主张的价值是否成立，发掘哪些客户才是有刚需的客户，通过找到真正细分的客户不断修正价值主张，这就是以敏捷思维开发商业模式的过程写照。

4.1.4 抛弃以产品为中心

我们做个小测验，列举三家耳熟能详的企业：星巴克、沃尔玛、苹果，然后投票选出哪个是以客户为中心的企业。大家的结论会不会是这些企业均是以客户为中心的样本呢？例如，星巴克的待客之道、沃尔玛提供的丰富且价格合理的商品以及苹果超炫的用户体验。

实际上，在美国行为数据专家彼得·法德尔的论述中，这三家均不是以客户为中心的企业。首先，星巴克的客户至少有百万人，他们每天都在距离自己方便的星巴克点上一杯同样的饮料，几乎从不例外。如果是老客户，星巴克咖啡师会奉上“照老样子点吗？”而当客户外出旅行，在另一家星巴克点餐时，恐怕没有得到过同样的关照。因为星巴克只是在简单传递产品价值，并觉得这已经足够。

沃尔玛也存在相同的问题。沃尔玛在全球15个国家拥有近9000家店铺，拥有巨大的销售额并在零售业占据主导地位，但是对客户知之甚少。尽管沃尔玛的产品不是自己研发的，但沃尔玛对产品本身的关注远远多于对忠实客户群体的关注，在收集客户数据、了解客户如何挑选产品并据此设计自己的主营产品和仓储物流方面，沃尔玛完全没有比同行表现得更好。他们只是做出了总体决策，即在巨大的销售市场上决定卖什么，而没有精准地对单个客户进行分析，或提供精准服务。因为对于沃尔玛而言，以销售规模和流量来降低成本是主要盈利策略，至少目前还无法考虑到精准营销的价值，同时也不划算。

最后来说苹果，这家公司可能会获得最多的投票，因为他们在用户体验上做出了无与伦比的创新，在获得用户喜爱的设计、制造、渠道以及推广上，苹果公司远超对手。无论来自哪个国家，苹果粉丝都是最忠实的粉丝，苹果的时髦、流行在年轻人当中最受欢迎。而苹果专卖店的维修，尤其是美国的维修服务绝对令人无欲何求。每个月在各地都有数以百万人购买苹果产品，这家公司有无数的机会与客户进行交互，了解他们、理解他们，但是苹果在这方面做得很少。苹果从根本上并不渴望了解客户对新产品的想法。例如，iTunes中的Genius引擎可以利用以往的购买记录判断顾客对其他歌曲的喜好，但这项功能直到2008年才出现，这种缓慢的市场反应速度令人难以置信，而亚马逊早在十年前就推出了原理基本一致的功能。

苹果的设计师和产品经理并不是不了解自己的客户，在产品的设计和使用上他们简直是天才，但是这样的模式并没有什么创新，他们依然遵循和信任长时期统治行业的以产品为中心的模式。在5年或10年以后，是否还能够像今天一样有效地获取利润？我们只能期望如此出

色的以产品为中心的苹果公司能够与时俱进。

4.1.5 过硬的产品

以客户为中心，不是说只要鞍前马后地让客户“满意”就行了，好产品、过硬的产品是以客户为中心的，对于好用的、耐用的产品，客户也将不离不弃，这是我们粘滞住客户的基本原因。同时，好产品就是质量过硬的产品，维护成本低，即使客户在使用产品时遇到了问题，也可以快速解决。总之，没有好的、过硬的产品，再好的商业模式也是空谈，再好的客户也无法挽留，所以“以客户为中心”也就失去了意义。

这里分享一个故事。有一位顶级家居品牌的销售人员，一天她谈了一个客户，前后5个小时陪伴，她的态度很好，笑容、专业都很到位，而客户隔天告诉她因为没有更低的折扣要退货。她是如此重视和小心应付这个客户，实在是因为客户太苛刻和挑剔，才导致销售失败……

其实，这位销售人员用心帮助客户值得表扬，正因为她体贴客户、用心沟通才能够留住客户5个小时，也正是这种友善待客才能够暂时成就这笔销售。但是，理智的客户往往不会为专业笑容买单的，如今区域销售的概念已经被电子商务打破，销售产品几乎不受地域的限制，购买者也越来越聪明，各个环节也愈发透明，产品真正从各个角度满足他们自身的需求才是成交的理由。而这个需求到底是什么则需要企业用心去理解，客户是否因为价格和折扣的问题要求退货都与这次销售无关，而是与产品有关，因为折扣问题的根本是产品问题，是产品在当下价位上的品质逊于竞争产品的问题。既然，产品品质达不到客户需求，就有客户消费后后悔的风险。所以，与其自怨自艾，不如思考产品到底哪里出了问题，从品质上去改进，直到合理满足客户的需求。

4.2 以客户为中心

4.2 以客户为中心

如果无法做到乔布斯那般极致的产品和准确的市场预期，那么不如换一种思路。“以客户为中心”是一种战略，是将产品和服务的研发以及交互与一部分选中客户的当前及未来需求相结合，从而使这部分客户对公司的长期经济价值最大化。只有少数明智的企业，例如IBM、亚马逊等，正在把客户中心化模式付诸实施。而其他企业没有实施的原因很简单，因为客户中心化需要公司愿意并且能够改变其组织架构和生产流程、业绩衡量指标以及员工和渠道的结构，从而集中打造这个创造和交互的过程。

对于“客户阻碍敏捷推行”的论断和立场，我们必须认识到以“研发”为中心的团队与生俱来地认为“研发高优先级”，重视过程而不是结果的职业特性，是不是有一种因为错误地强调了过程而忽略目的的嫌疑呢？敏捷的实质是更好、更快地提供客户满意的产品，以客户为中心的思考和行为需要更有立场和委婉地与客户沟通和协作，不能理解为“为了你好”就可以剥夺客户选择的权力。

在IBM的敏捷转型方式中，影响敏捷实施的有几大因素，这些因素正是我们需要思考清楚和勇于面对的挑战和机遇，而不是敏捷推行的障碍，因为没有一种“银弹”可以解决所有团队的敏捷需求。

所以，不要让客户立场和需求成为我们实施敏捷转型的不可抗拒的对立面，我们所做的一切转型和敏捷推行都是为了更好地服务于客户，想当然地理解客户并不是敏捷的要义。

4.2.1 敏捷交付

除非产品开发团队能够快速灵活地迭代式开发增量型产品，否则新商业模式、盈利模式的转型毫无意义。所以，初创企业也在使用敏捷交付的方式，永远只将资源配置给那些目前看起来最为重要的客户需求，并毫不吝惜地放弃已经确定没有价值的功能的开发。快速成功或者快速失败的方式，使得董事会和公司可以在最短时间达到最大商业价值。定制后的DAD方法和ASM框架在初创企业模式中作为产品交付和研发体系的根本思想值得借鉴。

基于DAD模型，改变ASM可扩展因子，如团队规模、地理分布、规范性需求、领域复杂度、技术复杂度、组织分散度和企业规范，再结合创业公司的期望，我们开发出了极简主义组织和产品的开发和交付模式。

为了说明方法的适用性，假设我们有一支初创公司团队，其规模少于10人，为精简分工，省去了营销，专心做产品，团队成员组成如下。

1名CEO，既是产品经理也是公司总经理；1名技术总监，除了参与关键技术的实现工作以外还需要处理很多团队管理、客户管理的工作；2名资深开发工程师，超过80%的时间做项目开发，此外还负责给其他工程师做导师，培养新员工；1名高级工程师，90%的时间用于项目开发，还要预留项目学习时间；2名初级开发工程师，需要努力学习技术和项目，追赶前辈成为独当一面的人才，大概一半的时间可以投入到维护性项目和协作式开发项目中；而前端开发和测试工程师（可能不足1名）则哪里有需要就在哪里工作，属于机动部队。

组织采取扁平化管理方式，所有人直接报告给CEO，并全部遵守企业内部规则和流程，对于任何一个项目和产品的研发，所有参与人员能够快速沟通、目标一致。公司使用简单的即时信息沟通平台，所有人可以在同一个开发平台上工作，并且对其他人工作项目可见。员工关系简单，组织灵活，沟通起来自然便捷有效，协调性好。核心技术和核心产品由自己来开发，虽然前期投入大，但是组织对产品的可控性、技术掌控能力，以及风险抵抗能力高，

新品发布和技术机密也能够很好地掌控在少数几个必要的人手中，一定程度上降低了被抄袭和盗用的风险。

麻雀虽小，五脏俱全，创业型公司的研发流程和团队构成相比Rational的DAD流程只是简化一些，所以可以采用相应的模式。我们效仿DAD的流程（差别仅在于每个构建的内部取消测试活动），将所有创业公司的人员分组：

- 项目领导团队（CEO+技术总监+测试工程师）。
- 开发团队（开发工程师+前端开发工程师）。
- 独立测试团队（测试工程师）。
- 部署安装团队（所有技术团队成员）。

由于创业公司人少事多，测试资源也严重不足，在尽可能不影响团队成员专注度的前提下，需要将机动部队资源划分到不同的组，以一敌多；同时，为了保证交付的速度，至少要将开发团队划分成两支并行的增量型开发团队。以不足10人的创业公司为例，我们可以做到并行开发两个模块，其人员构成大致如下：

- 项目领导团队：1名CEO+1名技术总监+0.2名测试工程师。
- 开发团队A：1名资深开发工程师+1名初级开发工程师+0.5名前端开发工程师。
- 开发团队B：1名资深开发工程师+1名初级开发工程师+1名高级开发工程师+0.5名前端开发工程师。
- 独立测试团队：0.8名测试工程师。
- 部署安装团队：所有技术团队成员。

4.2.2 建设敏捷团队

敏捷交付是任何一支自我管理的团队都可以做到的。而培养一支健康的团队，需要经历的四个阶段分别是：招聘和组织建立，人员磨合，漫长的成长期，最后才是高效期。对于敏捷团队建设，我们讨论以下几个方面。

1.招聘团队

使用敏捷交付的团队，大家情绪高亢、信心百倍，要有“只要给我一个支点我能撬动地球”的信心。核心团队一定要大气、上档次，不在乎短期内没有回报，但是，虽不愿意被人约束，却知道服从大局，服从创始人的决策；可以个性化，按照自己的方式工作，甚至为代码加上自己的署名。

对于个人来讲，如果一开始钱不是最重要的，那就要构建一个愿景，且符合每个人的心愿。愿景可以是股份、期权、额外的收获等，但不是无限期、无条件的给予，要有退出标准。招聘人才时，这一点很重要，要求创始团队能清晰描述自己的愿景，中间可能会遇到什么问题，有怎样的退出方式等。这才能吸引优秀人才加入，如果志不同、道不合，就不要招进来。

团队有共同的目标，相互尊重，一起创业是为了享受这种“愉悦的团队环境”，期望自己的价值能够通过团队的合力被放大。

所以，一开始就要跟团队讲清楚，敏捷交付意味着要清楚了解所要做的事情，不清楚绝不要冒然进行，因为不清楚的地方有可能马上发生变化。可是，一旦变化确定下来，就要执

着行动，全身心接受任务并持之以恒地完成任务，以个人能够做到的最高标准尽快交付。

2.建设合力文化

敏捷交付组织的团队文化，即合力文化，从创建期就要开始建设，就好比个人的习惯，需要在孩童未懂事前就开始培养。在研发进行中，初创企业团队需要学会节制，没有必要花费的时间不要花，没有必要做的功能不要做，一切向着实现CEO构建的伟大目标而努力。要有事业心，把工作当做生命一样珍惜，做事情要周全，设想得更远、更多。每个人都要保持开放的心态，不了解的就去学，了解的也不炫耀，诚实坦荡，让别人理解你，并在别人有需要的时候主动帮助，帮助别人成长就是帮助自己，也是自身价值的最好体现。

开例会既是个人才华得以在团队展示的机会，同样也是培养团队合作精神的机会。让其他团队成员知道自己尽最大力量完成擅长的工作，而不是状态的汇报更新，不再需要那么多时间交代过去和将来做什么，只需要关注“什么阻碍了你的工作”和“什么将会影响其他人的工作”。每天都在系统里更新完成工作的情况，对待工作认真负责。遇到合作问题首先不是去求助，而是先想方案再提供给对方选择，所谓合作就是100分的工作你我共同承担，我会主动做好55分再交给你完成剩下的45分。

团队需定期组织回顾会议，注意发言顺序，请外向的人先发言，内向的人后发言，让每个人都勇敢地提出建议，无论对错，都要对勇敢提出不同观点的人给予鼓励，尤其鼓励自省和自我总结，不但可以积累个人和团队经验，还能提高团队自我修养。还要分享知识，充分挖掘团队的聪明才智，在团队内部鼓励有价值的探讨，不安于现状，为团队的成长不遗余力。CEO还应该了解和帮助团队中的每个人建立自己的个人发展目标，并鼓励他们不停地向新的高度突破。

同时，在企业提倡志愿服务精神，提倡“欣赏他人、与人为善、有爱无碍、平等尊重”的友爱精神。鼓励公司员工加入公司志愿者团队，甚至可以联系其他志愿者组织开展各种志愿服务活动，帮助那些困难和处于危机中的人们。当代商业社会发展的核心命题——人们为什么而工作，一直被众多研究者不断提及。秘密就是利益驱动往往会适得其反，而充分发挥志愿者精神，却有释放无穷力量的可能。如果公司有这么多的志愿者，那么，长期和真善美的人在一起，并从事志愿者工作，一定能得到真知，学会照顾弱小，而不是自以为是；学会利他利己，而不是损人利己；学会获取荣誉感，而不是单纯地受利润驱使。这样的团队很大意义上将减少管理成本，而如今的管理模式在某种意义上就是管理者能在多大程度上激励多少人成为公司的志愿者。

3.组织的发展

下面将讨论自我管理和个人自律的培养，这是一个深远的话题，而自我管理的团队建立也是一个过程，它需要经过4个发展阶段，从团队的“启动期”，经过“磨合期”，然后是“休整期”，最后是“高效期”。

- 启动期：新鲜、有趣、令人兴奋。
- 磨合期：挑战、冲突、渴望放弃。
- 休整期：团队一致，成员定位。
- 高效期：团队生产力爆发。

(1) 启动期

团队初创成功，成员虽兴奋不已，但面对公司实施敏捷交付要求掌握的方法和实践操作，则表现出不同的反应。第一种是很平静地等待和观望，不去理会，但是也不拒绝，做做表面功夫，应付一下领导，自己该做什么还是做什么。第二种是被动地反击，思想方式不希望发生改变，叛逆，甚至暗地里斗争，对于敏捷实施的细则不予遵守。第三种是无条件接

受，兴奋、高调地响应，对新的方法、流程充满了期望，借此机会批判所有老旧的过程。一开始的迭代体验让大家表现出一定的挣扎，但因为仍然有强有力的支持者，某些敏捷活动会表现得比较顺利，实施起来比较灵活。

(2) 磨合期

敏捷实施很快进入了第二个阶段——团队的磨合期。经历了一两个迭代后，保持沉默的和本来就逆反的成员不能继续无视团队走向“不可控”的一面；而支持和充分接受敏捷的团队成员也能够看到敏捷方法中那些不尽人意之处，例如依赖彼此的信任和配合，需要自己担当和考虑比以前更多的事情。“怀疑”在团队中弥漫开来，“危机”开始爆发，而团队成员之间的磨合期恰逢其时。此刻，有意见的成员都会直接或者间接表达出来，团队之间开始分立，冲突对团队合作的冲击越来越大，不能置之不理，但是又不能过多干涉。就好像一支球队，面临一场球赛，仅仅依靠个人的成功是不可能的，团队的成功靠的是团队里每位成员的配合与合作。个人能力再强，没有队友的配合也无法场场取胜。而此时，团队中的内耗让团队开始丧失取胜的机会。如果组织或者团队试图忽略和隐藏暴露出来的问题，团队中间的指责和不自信就会膨胀，团队的自我管理氛围就会瓦解，所有人将恢复到原来的样子。所以，当下最为重要的事情是承认客观冲突，尽管有的冲突表现得较为温和。团队建设的“磨合”阶段是一个必经过程，教练、导师、CEO要清楚这个过程，不是回避而是期待它，并强有力地坚持使用敏捷方法和实践体系，遇到问题就地解决，如果实在是有很多问题，那就先解决容易解决的问题或者紧急的问题，必须打好“攻坚战”。

(3) 休整期

经过4~8个迭代，团队将会挺过磨合期，可以说到了一个相当舒服的阶段，此时基本的敏捷过程和实践被团队接受、理解，团队和组织已经能够为自身量身做一些实践方面的裁剪，例如，开会的时间和频度恰到好处，会议内容也精简节约，每个人学会了从创造性的工作中找到乐趣，对新环境下的工具操控、技能掌握也趋于纯熟。虽然使用敏捷实践仍然有许多障碍，甚至团队当下的生产率与老方法下的生产率相比没有明显的改善，但是发展已经步入正轨，磨合期中看似庞大的困难和挑战已经攻克，团队成员间的磨合也到了风平浪静的阶段，与利益关系人的关系基本顺畅，团队成员之间的配合已经达成一定的默契，沟通良好而自然，定期（提前）竣工的交付已经实现。

(4) 高效期

团队进入“高效期”这一最佳阶段的直接表现就是，团队成员相互尊重、高度信任、彼此和谐，就像一对已经生活多年、感情深厚的夫妇那样，高效的团队成员总能迅速而准确地了解彼此的想法和情感。而从外部来看，这个团队的领导者往往能够得到各方的支持，能够营造开放和信任的组织氛围，同时也是很棒的培训者和指导者。从个人成长来看，每个队员都明白了什么才是有意义的“工作完成”，并逐渐把这个理解扩大到团队从零到满交付的每一次迭代。

4. 带头人

管理团队要有平常心，平时不要把自己当领导，更不可以藐视他人，尽可能为团队提供支持与服务，将管理金字塔倒过来理解，CEO在最下面，所有员工在第一位。要用动态的眼光看人，不要只看学历、工作经历等，对于他人的错误要宽容，要允许团队犯错，定期与团队的每一个成员面对面交流，充分了解团队核心成员的心理、优势、劣势，并努力帮助其成长，一个人走100步，不如让团队每个人走10步。

团队带头人除了要做到得体、大方以外，还要公正，最怕的就是一碗水端不平，厚此薄彼，对于那些喜欢的下属无理由地偏爱。整个团队需要讲诚信，答应彼此的事情就一定要做到。团队的利益高于一切，一切个人目标服从于团队目标，CEO也要服从。

既然是CEO、管理者，就要将团队激励作为一项长期的、持之以恒的工作来做，并用领导者的人格魅力整合一个团队，然后激发他们的热情，全身心地投入工作。这才是一个合格

的团队领导者必须具备的素质。

敏捷团队的管理中有一个关键的利害关系，就是如何界定“界限”，我们鼓励团队共享目标、共享利益，团队自我管理意味着除了下放责任和工作，还要下放利润和权力。团队应该自治，如开发团队A，所有人无论是开发、测试还是设计都应该承担不能如期完成交付的责任，为任何一个延期交付的严重缺陷负责。所有人都应该对团队的荣誉和耻辱感同身受，而不是一个人的功过得失，这就是团队的界限。但是团队间彼此是合作关系，也是竞争关系，作为团队的最高管理者，需要协调团队之间的问题。例如团队A和团队B分别负责模块A和模块B，当A和B需要整合的时候，让哪一支团队来修复整合部分的缺陷呢？尽管这部分工作看起来是组织内工作，但对于任何一支团队而言都是额外的工作，管理者不能以“我们都是一个团队”来简单粗暴地对待，应该责任明确地分工，以如果做了就是额外的奉献，如果不做也不是你的错的心态来知会两支团队。事实上，这项工作往往能够看出哪些是人才，真正的人才会自己站出来，以老板的立场考虑问题，痛快地利用业余时间解决这个问题，我们必须耐心等待这样的人才出现。

很多管理者和CEO都希望部下既有狼一样的执着坚定，又有羊一样的温顺。但是这个愿望实现不了，我的亲身体会是，如果充分授权，就一定会受到部下的挑战，做好心理准备，他们是狼就要有狼性，只要能够充分投入战斗并解决问题，就不需要管理者插手。对于管理者来说，知人善用、大度和自信是成功的品格。

4.3 如何解决人事危机

4.3 如何解决人事危机

初创公司需要建设高效的团队，需要培养CEO、管理者、执行力很强的成员，我们都希望招募最合适的人才，并且成功地影响和留下他们与公司一起持续发展。

高效的运作方式，即实施团队的敏捷交付，自律、自知的优秀成员很快会成长为公司独当一面的人物，而对于企业而言，最大的危机就是这些人才的流失。某些专业的HR机构证明，企业人员的流动低于10%是正常的，但是，对于“人才”，无论流失多少，都是极大的损失，而最有可能的原因就是和“直接上级”的矛盾。马云说过，一个发展了几年的员工要离开这家公司，“直接上级”关系较大。仅因为蹩脚的管理导致了人才流失，这一点着实遗憾。

近些年由于经济危机的影响力还在，一些行业受到严重影响，许多大公司的业务支撑也越发困难，沃尔玛和苹果公司关闭了一些门店和工厂，IBM也迅速地裁掉了成本极大的产品线，例如一年耗资69亿美元的低端服务器部门。与此同时，各种负面的声音中，大公司所面临的“人才流失”也很严峻，而越是危难的时候，越需要英雄人物挺身而出，越需要挽留住“人才”。大公司员工比较多，流失一些还会填补一些，只要将人才流失比例控制在可以接受的范围内，未必会出现负面影响。但是初创公司就不一样了，很多初创企业一开始海誓山盟地集中了一批热血沸腾的年轻人，在遇到问题的关口却不能挺住，人才的流失将直接影响创业公司的生存。

最近，一个朋友说起自己创业5年的公司就在今年刚过完年的时候，几个骨干陆续要离职，这几个骨干公司都委以了重任，且培养过至少一两年。不用我说，你们一定体会得到我这位朋友的痛苦。成天“不务正业”，忙于“是非”，刚刚解决一个员工的离职，又有员工要求离开，同时还要处理员工心猿意马不认真对待工作的态度的问题。总经理的时间总是不够用，每天有开不完的会、见不完的人、处理不完的事，疲于奔命、忙于救火、生活不规律、透支健康、没有时间陪伴家人，当然更没有时间发展自己的兴趣和爱好，每天除了工作还是工作。因为好几个业务活动几乎陷入“冬天”，连其岳母都参与了救赎行动。后来，这位朋友跟我讲，这几个骨干该给的都给了，但是呢，稍有风吹草动就要离职，骂也骂不得，哄也哄不来，最终也没有留住。受其委托，我来到这家公司和他们一起工作了两天，希望能够帮助找到问题的症结。

4.3.1 快乐才是我的目标

在观察期间，我开始从朋友的口中听到了“控制、限制和制定规则”的论调，有种“听我的才是对的”的管理思想。显然，他忽视了人都有被认可、完成目标、追求高绩效的愿望，就好像孩子希望体验“跌倒后爬起”一样，团队也希望靠自己的力量完成工作，尽管并不完美。所以，就像不应该剥夺孩子体验的权力一样，为什么要剥夺年轻人“体验、成长”的权力呢？而持“控制论调”的不是他一个人，在观察期间，我竟然发现有管理层对员工讲：“做好员工是你的本分，你只管按照我说的去做，方法、策略那不是你的工作！”

对此，我只是匆匆做了记录，没有立刻打断这些谈话，因为我要看看到底还有没有更深层次的问题。我的理解是，即使让员工拿着丰厚的薪水，他们也未必愿意继续留在这里。原因很简单，剥夺员工参与、学习的权力，就是剥夺了员工的成长机会，而成长机会对于80后、90后来讲不是第一位就是第二位考核去留的因素。

在一家优秀的公司，好的老板都是积极帮助下属成长的，下属会觉得在这样的老板下面工作是件快乐的事情，当然不会三心二意。好老板通常很有人情味，能够让员工感受到被关注、被尊重和被认可，让员工产生强烈的归属感，管理者如果能照顾好员工，员工就会将感激和愉悦投入到为公司尽心尽力的工作中去。这就是员工快乐积极的心态的缘由，它将会为

公司创造更大的价值，推动部门和企业的可持续发展。

“人力”是一种资源，但不仅仅是一种资源，我们不能忽视“人力”的人性、内心价值、潜意识、愿景对团队生产力、组织成长的作用。有种说法是“人力资源是利用率最低的资源”，企业自动化、机械化提升的只是产品研发中很少一部分的效率（efficiency）问题，并非效能（effectiveness），自然采购、流程设计、战略规划却不能通过机械化实现提升，只有“人”才能发掘被隐藏的能力，只有“人”才知道怎样用这种能力实现企业效能的提升。

新时代就要具备新思想，现在80后、90后已经成为骨干，我们还用传统的企业管理制度，即所谓的“外儒内法”那一套就落伍了。70后常常敢怒不敢言，让加班就加班，鲜有怨言。但是，80后敢怒敢言，要让他们加班且俯首称臣，就得解释清楚加班为什么拯救了世界，或者准备好他们给你上课吧。而90后根本懒得理你，他们就是看不惯加班，只要谁让他们不爽便拔腿就走。00后马上登场了，这些孩子可能生下来名下就有了三四套价值千万的房产，凭什么要在这里跟着你的小指挥棒转呢？他们缺10万年薪吗？他们凭什么仰视领导？你都不尊重、不关心我，我为什么还要在这里和你玩呢？许多年前，我们把“员工价值”四个字看得一文不值，但是今天，老板们、经理们需要好好思考了。

那么，什么才是80后、90后留下来好好工作的理由呢？如今，年轻的一代已经比我们更快、更多地吸取着各种信息，他们获取信息和学习的热情远远高于我们，除了经验外，知识和存储知识的脑容量都将超过我们这一代人。很可能你的手下能力超群，甚至在你之上，却桀骜不驯、不服管束，而这些人却又是主要的价值缔造者。他们需要的是认可和关心，需要感受到平等和存在感。是否尊重他们的意见，是否为他们提供了更大的舞台，是否可以让他们施展才华？“坐冷板凳”的思维方式是绝对不合拍的，他们认为命运是靠自己决定的，而不是靠天、靠地、靠爹、靠老板，尽管许多孩子家境一般，但是花钱从不吝惜，10分钟车程却打车上下班，请客吃饭出没于高级餐厅，这都是常有的事。如果你还以“扣钱”或者“涨薪”来引导他们的价值导向，那就大错特错了。所以，我们需要投其所好，这些年轻人能够留下来好好工作的主要原因是可以学习和能够不断成长的空间，必须要让他们体会到在这里可以不断提升自己，并且不断认可他们所取得的成绩。或许，你觉得他们什么都不如你，但是，请不要把他们当孩子，他们较起劲来不好收场，况且年轻人并不认为你比他们强多少，至少在他们这个年纪你不比他们强多少，他们相信年轻、积极总有一天会超越你。

总而言之，如果没办法招募到和企业发展方向高度一致的人才，就需要寻找能够配合团队人才发展的企业发展方式。因为他们最终能否陪伴在你左右，完全由快乐与否、自身成长需求满足与否而定。

4.3.2 人才流失

我们很喜欢分类，将自己贴上适合分类模式的标签：“大公司”、“小公司”、“初创企业”、“打工仔”、“创业者”、“传统行业”、“互联网金融”等，为什么呢？因为其中有许多的共性，大家在一起可以很快进入有价值的议题，而且都是“苦难出身”，增加了一些认同感和同理心。有人曾说，公司越大分工越细，工作安排就越“局限”，在大公司里除了做事以外，最重要的是做人。而公司越小，工作安排就越“繁杂”，要学习并处理许多事情，除了“按期交付”，还要尽可能学成“多面手”，所以小公司还是做事为要。大公司高管往往有充足的时间和大量的分析师帮助其进行商业决策，但是创业公司的CEO需要每日做出大量抉择。“领袖和英雄”是小公司和大公司里都需要的，但是英雄主义在小公司里更有发展空间，而在大公司里，前提是协同与配合，有竞争但是因为更合理的内部机制而表现得较为公平些。

对于成熟的大公司来说，有智囊团在谋划，而对于初创企业而言，就得自己思考企业发展方式。初创企业绝非大型企业的微缩版，它是一个寻找可升级、可复制和可盈利商业模式的临时组织。成立伊始，初创企业的商业模式只是一块描绘着创意和猜想的画布，没有深度的客户开发，甚至没有细分过客户，自然也没有任何可用的客户信息。但创业者一定思考过如何赚钱、如何盈利，也考虑了如何选择自身的优劣势并“一条道走到黑”。简单地说，想成功的创业者一定是认定了目标，然后极致地去实现这个目标。百度公司董事长兼CEO李彦宏在演讲中曾说过：“人生之中存在着诸多让我们妥协、分心乃至屈从的事，我们应该舍弃那些不必

要的迂回和避让，遵从内心最真实的渴望，并为此竭尽全力。”成功的创业者往往质朴，甚至极端偏执。这让我们想起乔布斯——苹果公司的灵魂人物，他始终保持着坚定的信念，极致追求，在形形色色的选择面前永远迅速决策并把握方向，勇往直前。

无论是大型企业、中小型企业还是初创企业，首先要留得住“人才”，做经理和老板的要反省，是否真正关注和关心员工，让员工有归属感。其次，良好的工作环境、愉悦的人际关系也是“人才”乐意在公司长期发展、实现自我价值的原因。

4.3.3 构建人文环境

我观察过一些企业，其中人才不少，可是都有个毛病，就是“人人相轻”：当地人轻视外地人、技术轻视市场、市场轻视技术。有些“大牛”喜欢出言不逊，瞧不起技术能力差的员工，觉得与他们合作就是浪费时间和精力，甚至不能接纳、包容和欣赏他人的差异，当然就不能接受不同意见，不能坦诚沟通，不能在工作中真诚配合、互相帮助，把事情做好。其实，每一个人都有自己独特的“强项”，世界上不存在两个绝对相同的人，让团队充满了各色才干的人才，让每一个员工都能够发挥自己的才干，才能让团队成员的协同效率倍增。

要维护好的“团队环境”，一定要做好“内部协调者”。当团队内部发生冲突时，对和错的一方通常会抱怨和倾诉，要成为他们倾诉的对象，一定要沉住气，不要武断批评或让其改过，否则只能失去员工的信任，以后，就算是有事情他们也不愿意再找你，那样你对问题只能视而不见，听天由命。所以，当员工在工作中表现得非常焦躁时，我会有意经过他的座位，稍许停留，直到他主动找我说话。我的团队中有些特别内向、不够自信的员工，一整天无论发生什么事情，他都不会主动找我沟通，这种情况下，我会主动用即时通信的方式向他打招呼，询问他关于某件事的看法，以获得沟通的机会。还有些员工非常外向，只要自己内心受到创伤就一定要找人倾诉，我自然也就成为了他的倾诉对象。最戏剧性的事情是，这类特别愿意倾诉的员工往往是我花费最小沟通代价就可以达到彼此内心一致性的人。如何做到呢？其实非常简单，就是“静静地倾听和附和”，我只需要“嗯”、“是”、“就是”就可以安全过关，他会在抱怨一通以后很快扫去不快，并很认真地对我说：“其实，我觉得在……上我做得不好……”接着，我们很快完成了沟通，再聊聊家常，我什么也没做，他什么都已经清楚。

最难的工作要算是协调两个能力相当、相互不买单账强势“大牛”。这类人通常较为敏感，虽然在很多情况下有很强烈的反应，但通常不愿意将个人情感表现出来，而做事却有很强的原则性。对此，无论说什么都很难达成一致，确实棘手。因为这类人通常没法通过“沟通”、“讲道理”互相协调，只有自己悟出了道理才能改变。所以，只能尽量制造一些机会允许“犯错”，让他们认识到对方的价值，或者让他们做一些最合适两个人相互配合的工作，让他们体会到双方能力互补的魅力，才能够让他们真正冰释前嫌成为队友。

关于团队配合有一个非常好的故事，一个橄榄球队的教练手下有两个出色的队员，一个前锋、一个中锋，但是，因为过去某些不愉快的小事，这两个人一直不和，常常连球场上必要的配合都打不了，在公开场合甚至还打过架。教练并没有对其分别做思想教育工作，而是静静地等待机会。直到有一天，一场联赛，教练将两个人列入首发阵容，但两人因为不快，就是不好好配合，观众因为他们糟糕的表现唏嘘不断，无法理解球队为何如此不堪一击，与过往的表现大相径庭。场下，两名队员都非常沮丧，因为他们都是球队的主力队员，立下过汗马功劳，在前几场比赛中积累了不错的得分，而今天却连连失利。就在下半场上场前，两人非常肯定教练一定要换下他俩时，教练却仍然让他们首发。两个人此时很难平复自己内心的波澜，面面相觑，似乎在说：“我们配合试试，就为了这场比赛……”结果超乎想象地好，中锋在第一时间传球给前锋，在两人势如破竹的突破下，攻入一球，很快比分逼平，最终赢得了胜利。当两人互相击掌庆祝胜利的时候，发现内心已经冰释前嫌。此时，教练才说：“终于让我看到了这一天。”

缺乏团队合作也是现在国内许多软件企业在执行敏捷开发时的阻碍，因为组织架构和项目架构呈矩阵网络关系，项目经理对项目中员工的影响力远不及部门直接领导，而在日常沟通中，团队的直接部门领导常常会有意无意地制造“部门墙”，这堵深墙使得工作的开展、团队的协作受到影响。如果无法判断是否存在部门墙，只需要观察团队在开会的时候是不是总会

说“他们部门”和“我们部门”。总而言之，好的人际关系就是满足人的更高层需求，在马斯洛的五层需求理论中，在满足了温饱、安全以后就是社交与自我价值的体现。从心理学或者管理学交叉角度上来说，企业本身就是一个生态体系，有自己的独特运作方式（经营模式和管理流程），有自己的心智模式（企业文化和团队氛围），这一切取决于“人”。

4.4 人格与职业

4.4 人格与职业

在担任企业顾问的过程中，我发现采用人格测试方法进行面试的企业越来越多，大家开始认同人格与职业发展和职场定位的密切关系。有的人奋斗了大半辈子都不明白自己到底适合做什么工作，还有的人从事一项工作很久后才突然发现自己并不是这块料。那么，到底怎样才能证明“天生我材必有用”呢？

我的职业生涯虽然只服务于IBM一家公司，但在IBM我尝试了不同的岗位。而随着我对荣格心理学研究的加深，我发现自己原来真的是一个适合做当下工作的人。虽然人格没有好坏之分，但是当一个人面临职业抉择时，也会隐隐有适合不适合、容易不容易的感觉。例如，如果你不喜欢与人打交道就不必强迫自己去做公关工作（确实有些原本内向的人，在心情好或环境适宜时，非常喜欢沟通和聊天，因此错误地认为自己是个很善于交际的人）。

弗洛伊德认为最真实的自己是本我，而不是别人所期待的我——自我。人的根本期望和需求源自本我，本我与生俱来，在出生后不久便慢慢形成并稳定下来。荣格后来的研究表明，一个人如果能够看到自己的本质（本我），并懂得尊重和满足本我的需求，那么他将享受到更多的幸福和快乐。而自我可以看作是因为社会环境的影响而形成的“面具”，带上面具的我会在后天成长起来，并且不断和本我抗争、制衡。例如，如果本我很耿直，那么在交往中就容易对他人造成无意的伤害。于是，自我就选择带上体贴的面具，不断模仿那些很有亲和力的人的工作和交谈方式，以获取对亲和力技能的满足感。但是，这绝对不是本我，而是自我。相对而言，若一个人只能满足自我的需求，那么即使取得了值得骄傲的成绩，也会发现快乐转瞬即逝。本我的确很难通过测试或者其他简单的方式来发现，而在没有压力的自然情况下，当人们面临选择时，往往会不由自主地倾向于本我而非自我。

成功与本我人格有关系吗？IBM的第一位院士王云曾说：“成功源于协作，协作源于信任，信任源于沟通。”人与人之间通过坦诚沟通才能建立长久的信任，而沟通方式直接受人格的影响。所以，人格与成功是有关系的。作为企业的管理者，懂得采用不同的方式与不同人格的人进行沟通才能达到事半功倍的效果（下一章会具体讨论不同的沟通技巧）。但是，仍然有很多人不曾真正探寻过自己的本质，而是将外界压力所塑造的“我”当作了所谓的自己，将他人的目标作为自己的目标，在不断朝着他人目标努力的同时，内心的分裂和痛苦不断累积且不得释放。这就是“面具统治自身”的诟病所导致的后果，“出于强烈的上进心而笔直地向下坡走去”。

荣格人格心理学指出，团队的行为方式往往透露着其领导者的品性，CEO的人格也将影响整个公司的文化。迈尔斯·布里格斯人格分类指标（Myers Briggs Type Indicator, MBTI）是人格分类理论模型的一种，其基本理论依据是荣格于1921年出版的书籍《心理类型》（Psychological Types）。下文将讨论荣格人格类型和MBTI人格测试对职业选择和人才招聘的影响。

4.4.1 荣格人格类型理论

荣格认为，在与世界的联系中，人的精神有两种态度：一种态度是指向个人内部的主观世界，称为内倾型（或内向型）（Introversion）；另一种态度则是指向外部环境，称为外倾型（或外向型）（Extroversion）。前者爱思考、孤僻、退缩、常提防戒备、不愿抛头露面；后者开朗、爱交际。每个人只是更倾向于外倾型或内倾型，而并非绝对的外倾型或内倾型。所以，荣格指出，只有当外倾倾向占优势时，我们才把一种行为模式称为外倾型，反之亦然。

根据人们收集信息的不同方式，可将人分为感觉型（Sensing）和直觉型（iNtuition）两类。感觉型的人关注由感觉器官获取的具体信息：看到的、听到的、闻到的、尝到的、触摸

到的事物。例如，关注细节、喜欢描述、喜欢使用和琢磨已知的技能。直觉型的人关注事物的整体和发展变化趋势：灵感、预测、暗示、推理。例如，重视想象力和独创力，喜欢学习新技能、但容易厌倦，喜欢使用比喻，跳跃性地展现事实。

根据人们作决定和作判断的方法，可将人分为情感型（Feeling）和思考型（Thinking）两类。情感型的人常从自我的价值观念（对或错，有意义或无意义）出发，做出一些自己认定是对的决策，同时比较关注决策可能给他人带来的情绪影响，对语气、语调、情绪的微妙变化敏感，人情味较浓。思考型的人则比较注重依据客观事实的分析，一以贯之、一视同仁地贯彻规章制度，不太习惯根据人情因素变通，哪怕做出的决定并不令人舒服。例如，一些讲话较为直接的人通常是思考型。

在荣格人格类型理论的基础之上，美国心理学家凯瑟琳·布里格斯和她的女儿伊莎贝尔·迈尔斯经过长期观察和研究提出了第四个维度，即依据人们应对外部世界方式的人格划分。喜欢以一种较固定的方式生活或作决定的人属于判断型（Judgement），而喜欢以一种更自然、随性的方式生活的人属于觉察型（Perception）。判断型的人目的性较强，喜欢有计划、有条理的世界，更愿意以比较有序的方式生活，往往是制度和层级体系的拥护者。觉察型的人好奇心重、适宜性强，他们会不断关注新的信息，喜欢变化，也会考虑许多可能的变化因素，更愿意以比较灵活、随意、开放的方式生活。

4.4.2 16种人格类型及其特点

荣格说：“感觉告诉你存在某种东西，思维告诉你它是什么，情感告诉你它是否可以接受，直觉告诉你它从何处来、到何处去。”就像我们将光分成赤、橙、黄、绿、青、蓝、紫七种颜色一样，人格也可以分类并打上“标签”。七种颜色之间存在不计其数的过渡色彩，同样，人也存在倾向于某种人格或者多重人格的特点，尤其是面具人格的存在，使得人格很难精细化分和准确度量。即使在同一种人格象限里，比如都是内向型，有的人更愿意表达自己的情绪和观点，而还有的人却内向得近乎封闭。

将人格进行分类并不是为了判断一个人的好坏，而是帮助我们更好地理解一个人、描述一个人。同时，也帮助自己了解更真实的本我，了解为什么会有差异需求，为什么容易产生某一类错误，以及哪些学习方式更适合自己，哪些朋友更容易和自己发生共鸣。

不过，有些所谓的人格分类是错误的，例如，“南方人通常……”或“处女座的人一定……”，这样的分类和标签过于笼统或极端，甚至当我们用“你就是一个……的人”的句式来判断和打标签时，这种方式本身就是错误的。

荣格人格心理学和MBTI测试是现有的经验证实相对科学的分类方法。例如，对于一个多愁善感型的人，在与他沟通的过程中，道理和逻辑可能行不通，要满怀同情，尝试使用和缓的语气和略低的语调，比如：“你不是一个无可救药拖延症患者，只是需要吸收更多的信息才能做出决定，你也一样能够按时完成工作。”MBTI是一种人格归类的好方法，它能够帮我们读懂自己，了解他人。

MBTI将人格极其精准且个性化地分为16种类型，对此，我尝试通过真实的对象进行样本分析实验，以验证理论的可靠性。至今，172人完成了MBTI测试，结果只有4位研究对象表示不完全准确或“人格倾向的劣势”不准确，其他实验对象全部认可其人格类型，以及对应人格类型的每一句描述。幸运的是，我只花了3个月时间便找到了MBTI所有16种人格对应的样本。后来，我又将实验对象的MBTI人格分类与乐嘉人格色彩心理学和九型人格心理学分类做了对比，并且惊讶地发现，后两者的所有分类在MBTI分类中都可以找到一个或多个原型。

当然，MBTI和其他人格心理学方法一样，它们不是塔罗牌、卜卦或水晶球，不能预测未来会遇到什么样的人生磨难和生活考验，也不能断定陌生人之间是否会被彼此的外貌或经历所吸引，更不能阻止恋人的相爱或分手。MBTI只是帮助人们接受不同的人格倾向，承认且尊重彼此的差异，理解自己进而更加懂得理解他人，它为我们提供了一个看待其他个体或团队的新视角。例如，有趣的是，美国沃顿商学院的学生普遍是ISTJ（检查员）、ISFJ（照顾者）

和INTJ（专家）类型；而斯坦福大学的学生，无论是硕士生还是博士生，几乎都是NT和NF类型，只有绝少数是S类型（详见表4-1~表4-4），因此，斯坦福成为创业和学术人才的摇篮就不奇怪了。对于一家公司或一个团队，如何增进员工间的交流，如何给客户带来更好的信息和服务，如何组建优秀的团队，如何让有不同优势的员工都能如鱼得水，人格分类将成为其中的关键因素。一些人喜欢将自己习惯的沟通方式作为最佳沟通方式，想当然地以为只要用这种方式就可以同所有人顺畅交流。这种一厢情愿的做法忽视了一个事实，即人们习惯使用自己擅长的倾听方式而不是对方擅长的表达方式来接收信息。个人的沟通习惯与人格类型有很强的关联性，例如，外向型的人喜欢头脑风暴，而内向型的人喜欢一对一的深入交谈；思考型的人总是对错误的逻辑极其敏感，而情感型的人却对语气、情绪的微妙变化更加敏感。不同人格类型的人在一起交流时，处理不当就会造成沟通障碍，甚至导致人际关系出现裂痕。因此，了解自己的人格特点，理解不同人格类型偏好的交流方式，对于成功的沟通和协作至关重要。

4种基本人格分类是E-I、N-S、T-F、P-J，通过任意组合即形成四维人格，如ENTP，共计16种组合方式。每一种组合都代表一种人格类型，稍后将给出这16种人格的特点描述，我们先来看看这4种基本人格。如何判断一个人的基本人格呢？请看以下4张表格，其中每一列对应一类人格的特点，也是判断人格类型的方法。例如，朋友聚会时，有个小伙伴每次都是最热情的，总能和其他人打成一片并聊个没完，而且爱出风头，这样就可以将其判断为E型人格，即外向型人格。从而可以推测出这个人在独处的时候会比较难受，应该是个急性子，平时话语多，有时难免言多必失。但恐怕在任何场合他们还是愿意多说，也喜欢别人多听他们说，所以，和这种类型的人交往的策略是尽量多给他一些回应，对其所言表示兴趣和好感。

表4-1 E型人格和I型人格的特点

E型	I型
与他人在一起时感到振奋	独自一人时感到振奋
希望成为注意的焦点	避免成为注意的焦点
先行动，再思考	先思考，再行动
喜欢边想边说出声	在脑中思考
易于被了解，愿与人共享个人信息	注重隐私，只与少数人共享个人信息
说的比听的多	听的比说的多
热情地交流	不把热情表现出来
反应迅速，喜欢快节奏	思考之后再反应，喜欢慢节奏
较之精深更喜欢广博	较之广博更喜欢精深

再如，朋友聚会时，一个小伙伴总是特别愿意聊起时事、政治、历史方面的话题，那么可以将其判断为S型人格。对照表4-2可以了解S型人格的特点，即只相信自己看见、听见的东西，对精神、灵感和推理则不感兴趣。这种类型的人比较相信经验，极有可能过分地以自我为中心，认为别人提供的任何信息都不如自己的经验来得靠谱。因此，与这类人合作时，要多了解其支持、维护的立场，并赞许其过往所赢得的荣誉。

表4-2 S型人格与N型人格的特点

S型	N型
相信确定而有形的事物	相信灵感和推理
喜欢具有实际意义的新主意	喜欢的新主意和新概念只来源于自己
崇尚现实主义与常识	崇尚想象力和新事物
喜欢运用和琢磨已有的技能	喜欢学习新技能，但掌握后又感到厌倦
留心特殊的和具体的事物，喜欢给出细节	留心普遍和有象征性的事物，使用隐喻
循序渐进地给出信息	跳跃式地以一种绕圈的方式给出信息
着眼于现在	着眼于将来

依然以朋友聚会为例，有个小伙伴说话直来直去，虽然讲得很有道理，但是着实有些不考虑听者的感受，这个小伙伴属于哪种人格呢？对，一定是T型。这类人比较诚实，拥有行必信、信必果的人格。T型人格有担当，对于任何事情都倾向于冷静和客观地分析因果和逻辑，不太会仅仅因为个人喜好就草率地做出决定，很可能会成为一个公正的考官。那么，对于这么直接的类型，一定不好沟通吧？是的，他们没有那么平易近人，打动他恐怕只有靠真凭实据了。记住，千万要讲信用，这种人讨厌言行不一。

表4-3 T型人格和F型人格的特点

T型	F型
后退一步，客观地分析问题	向前看，关心行动给他人带来的影响
崇尚逻辑、公正和公平，有统一的标准	注重感情与和睦，看到规则的例外性
习惯发现缺点，有吹毛求疵的倾向	习惯让别人快乐，易于理解别人
可能被视为无情、麻木、漠不关心	可能被视为过于感情化、无逻辑、脆弱
认为诚实比机敏更重要	认为诚实与机敏同样重要
认为只有合乎逻辑的感情才是正确的	认为所有感情都是正确的，无论是否有意义
受获得成就的愿望的驱使	受被人理解的愿望的驱使

最后一个例子是关于J和P的故事，小J和小P忙碌了一天后回到宿舍，发现床上很乱，小J的做法一定是先把杂物规整好，将床打扫干净再睡；而小P则一定是倒头就睡。人格没有极端情况，所有人都只是偏向J或者偏向P多一点。当一个人表现出有原则、喜欢按常理和常规办事的时候，基本可以判断其为J型人格，但同时也要清楚，这种类型的人对于自身处境非常关注，如果处在不确定性中就会非常焦虑，因此，与这类人合作一定要让其放心，在事情的方方面面都要有B计划，并且最好像承诺的那样付出行动。

表4-4 J型人格与P型人格的特点

J型	P型
做出决定后感到快乐	因保留选择的余地而快乐
工作优先，先工作后玩	玩优先，先玩再工作
确定目标并按时完成任务	当有新的情况时便改变目标
想知道自己的处境	喜欢适应新环境
着重结果	着重过程
通过完成任务而获得满足	通过着手新事物而获得满足
把时间看成有限的资源	把时间看成无限的资源

4.4.3 团队“人格”

快乐的工作需要拥有一个良好的团队，而这个团队绝不仅仅是建立在技术互补的基础上，人格因素将深深地影响着团队的氛围。例如，一个急性子的老板一定不喜欢一个慢性子的下属，一个有创造力的开发者一定不喜欢每天被逼着做没有价值却必须花去大量时间的重复操作。团队的真正领导者往往决定着团队的特点，从而使团队表现出一定的“人格”。

E型团队。这类团队拥有丰富的技能和经验；沟通对外延伸，习惯性地与利益关系人保持着紧密的沟通；对于外界的响应非常及时也非常到位，而遇到了难题则更多地表现为寻找外界的帮助。

I型团队。这类团队会在领域的深度方面比较有建树，而且常常会从他们信任的少数人那里获得经验和意见；比较关注内部事宜、内部协作，当遇到难题时，通常考虑自己内部协调解决。

S型团队。这类团队非常愿意按照详尽的、提前设计好的程序来做的事情；对内部的奖励往往都能落在实处，对细节成绩的奖励比一般团队要多；更信任有经验和有阅历的前辈的判断，不喜欢更不推崇抽象而不实用的理论培训。

N型团队。这类团队充满想象力，喜爱创造和发明，可能有不少成员都是多项专利的拥有者；对内部的奖励往往难以落实，如晋升和加薪的许诺；对未来的关注通常多余当下，在市场分析、股市预期和风险评估等方面多有杰出表现；更相信灵感的指引。

T型团队。这类团队非常有原则性，据理力争，办事公正、清晰且务实；喜欢听取不同意见，不怕反面批评且能够化压力为动力不断提高自己；更喜欢设定逻辑和游戏规则，遵守规则，不喜欢越界；原则性很强，内部管理规则化而不近人情，会给人以刻板、冷血的印象。

F型团队。这类团队比较热情和友好，以价值为导向；对于功劳、苦劳一并看重，即使没有取得成绩，为之付出的努力也应该得到大家的掌声；原则性不强，有时会讲态度而不讲是非，习惯性地考虑法外开恩；工作和个人情感纠结不清，会给人以人情味浓但没有原则的印象。

J型团队。这类团队坚持例行流程，工作和生活绝对分开，先工作再玩，喜欢一次做好一件事情，踏实肯干；习惯定好目标、做好规划，并坚持严格执行，不接受例外和无计划状态；目标感很强，给人的印象是固执和守旧。

P型团队。这类团队具有极佳的并行处理能力，喜欢协调多头绪的工作，工作、生活、玩可以同时开展；认为灵活性和适应性是最重要的，不仅可以让工作充满惊喜，也消除了不确定性所导致的焦虑，反脆弱也是这个道理；内部只保留最小集合的例行流程，过程简化，拒绝复杂；喜欢的工作类似于开放接口供用户随意使用，通过灵活地调整参数和配置解决可能出现的问题；随机性太强可能是团队的弊端，给人的印象是不按常理出牌，常被认为是异类。

4.4.4 择“优”录取

所谓的择“优”录取在这一节即为择“人格”录取。在此，我做一个假设，假设这家企业的创始人是ENTP型，先看看这个类型的特点。

ENTP型人格。这类人有很强的首创精神，擅于运用创造冲动；反应迅速，睿智，有激励别人的能力，警觉性强，直言不讳；在解决新的、具有挑战性的问题时机智而有策略，善于找出理论上的可能性，然后再用战略眼光进行分析；善于理解别人；不喜欢例行公事，很少会用相同的方法做相同的事情；好奇、多才多艺、适应性强，在解决挑战性和理论性问题时善于随机应变。

如果说这个ENTP型创业者影响着团队的基因，那么对他而言，所需要的合作者——一个能够直接帮助他协调好具体工作且带领好技术团队，能够理解他且在重大决策上形成互补的人——一定是ISFJ型人格。

ISFJ型人格。这类人有责任感且行事谨慎，能坚定不移地承担责任，做事贯彻始终、不辞劳苦且准确无误；忠诚，替人着想，细心，记得他所关注的人的种种微小事情，关心别人的感受；努力创造一个有秩序的、和谐的工作环境，意志清醒，十分务实，具有强烈的职业道德。

我们再来分析一下这个ISFJ型管理者，ISFJ是中层领导，期望自己的职位稳定，上下级沟通顺畅。想要做到这些，除了领导者自身的项目管理和人员管理能力必须过关以外，还需要懂得沟通的技巧。对于下属，有效的沟通能够赢得他们的支持和信任；而对于上级，这位中层领导一定要和创始人形成默契，即双方都懂得如何沟通才最有效。因此，我认为选择ISFJ型的人作为中层领导非常合适。相对而言，创始人的工作是指明方向，而中层领导的工作是跟进项目，事无巨细地率领团队解决问题，需要更注重细节（S）和执行力（J）。

接下来，在招聘管理者下属的一线员工时，就不能仅仅从与管理者互补的角度来考虑问题。因为从管理者的工作安排和时间分配情况来看，与团队一起工作时，沟通和协调内部资源占了较大的比重，也就是说，沟通的代价很高，沟通成本是除了员工技术和资历之外最需要优先考虑的因素。那么，能够与ISFJ类型的人良好沟通的人是谁呢？沟通中最为重要的是收集信息和做出决策的方式，与ISFJ类型的人具有相同方式的人格类型可以是ESFJ、ISFJ、ESFP、ISFP，我们来看看ISFJ以外的三种人格的特点。

ESFJ型人格。这类人有爱心，有责任心，通过积极直接的行动来参与合作；希望周围的环境温馨而和谐，并果断地营造这样的环境；喜欢和他人一起精确并及时地完成任务；忠诚，即使在细微的事情上也如此；能体察到他人在工作中的需求并竭尽全力提供帮助；希望自己和自己的所作所为能受到他人的认可和赏识；很现实，实事求是且安排有序；支持现存制度，往往是委员会或组织机构中积极主动和乐于合作的成员。

ESFP型人格。这类人外向、友善、包容，热爱生活，追求物质上的享受；喜欢与别人共事，工作上讲究常识和实用性，能够使工作富有趣味性；灵活、即兴、自然、不做作，易接受新朋友和适应新环境，与别人一起学习新技能时可以达到最佳学习效果；乐于参加各种活动和节目，而且通常能同时应对几种活动；现实的观察者，按照事物本身的属性去对待并接受它们；天真率直，很有魅力和说服力，喜欢意料不到的事情，喜欢寻找给他人带来愉快和意外惊喜的方法。

ISFP型人格。这类人沉静、友善、敏感、仁慈，能够欣赏身边所发生的事情；喜欢有自己的空间，做事时能够把握时间；忠于自己的价值观，忠于自己所重视的人；不喜欢争论和冲突，不会强迫别人接受自己的意见或价值观；完全生活在当下，所以准备或计划往往不会多于必需，是很好的短期计划制定者；喜欢享受目前的经历，而不继续向下一个目标兑现，所以他们对完成工作感到很放松。

从敏捷思维中接受挑战和拥抱变化的角度来看，领导者和员工都必须具备一定的P型人格，而这正是人格学有趣的地方。一个ISFJ类型的人也可以具有一定的P型人格，因为并不存在绝对的J型或者P型人格；另一方面，一个项目经验丰富的职业领导者，在特定时候表现出的其实是面具人格，如果面具人格里有P型成分，那么，即使ISFJ型人格会表现出传统的本质，也不影响其带领团队完成敏捷交付任务。

4.4.5 SJ型与SP型人格的任用

根据大卫·凯尔西的气质与性情理论，SP类型的人属于“经验主义者”，他们关注和相信那些可以测量和证明的东西，同时喜欢面对各种各样的可能性，喜欢自由随意的生活方式，是反应灵敏和积极主动的一类人。经验主义者富于冒险精神，他们最显著的特点是足智多谋，一想到某件事情就有立即去做的冲动，而且喜欢一气呵成。

而SJ类型的人则属于“传统主义者”，他们相信事实、经验、已证实的数据和“五官”所带来的信息。相比于SP类型，SJ类型的人更喜欢有结构、有条理的世界，喜欢做决定。他们尊重权威、等级制度和权力，而且一般具有保守的价值观。他们很有责任感，而且会努力去做正确的事情，这使得他们值得信赖和依靠。

人格是一个完美的综合体，人格平等，每种人格都具有自身的优势也同时存在着不同的潜在问题。例如，SP类型的人不喜欢长时间做同一件事情，只有很少的事情（如很喜欢的事情）能够长期坚持，其他多数情况下则难以做到坚持不懈；而SJ类型的人可能不擅长变通或缺乏想象力，不能很快做出改变和适应。

总而言之，对于不同的人格类型而言，没有好与坏之分，每一个人都是一個独一无二的个体，都有其特别的优势和劣势，问题的关键是认识这些优势和劣势，在职场中选择适合自己的位置，实现事业的长期发展，学会通过职业规划、成长计划来“扬长避短”。

实际的MBTI测试结果也会存在误差。其中最主要的原因是参与测试的人在做题时会出现三种心态：过去经验的我、真实现在的我、理想完美的我。以过去经验的心态去做题容易得出过去的职业人格，以理想完美的心态去做题容易得出想象中的职业人格，而只有以真实现在的心态去做题才会得出真实的职业人格。许多人在做题时会在三种心态之间不停地切换，甚至几次测验出现不同结果。而网上的MBTI测试题也不是十分完美和系统化，容易造成理解偏差。例如，我们来看下面这道MBTI测试题：

你愿把（ ）作为朋友？

- A. 常有新观点的人
- B. 脚踏实地的人

从这道题看，“常有新观点的人”代表N型加分，“脚踏实地的人”代表S型加分，但是，和他们交朋友的人属于N型还是S型就属于个案了。许多人都拥有人格多样的朋友，和N型的人交朋友不代表自己就一定是N型。因此，从这道题能否得出正确的结论，将取决于每个人理解的角度。不熟悉MBTI精要的人进行测试时，最好能与MBTI专家谈话并做全面的测试调查，或者在做测试题之前进行相应的心理辅导。

我的建议是，在进行招聘甄选和岗位匹配时，如果没有心理学专家参与指导，不妨使用网上的MBTI测试题来测定候选人的人格类型，虽然有些繁琐，但是值得尝试。经过数十年的研究和几代人的努力，MBTI已经成为比较普遍的测评工具，其目的是积极的，即用便捷的方式推动人与人之间的了解与沟通，帮助我们选择更好的职业道路。即使有人对结果存在质疑，但有益的尝试和体验也会带来收获。

第5章 敏捷型人才的培养

第5章 敏捷型人才的培养

- 5.1 如何培养自我管理的秉性
- 5.2 如何培养自知之明的秉性
- 5.3 如何培养自律的秉性
- 5.4 提供上升空间
- 5.5 团队的沟通

每每遇到朋友们谈论敏捷转型，最深入的问题就是如何培养敏捷人才，因为敏捷的核心价值以及产品和项目的成败、效率、产出、度量、改进，一切的一切核心都是“人”，一个好的团队可以做出一流的工作，而一个二流的团队做出的工作可能不到三流的水平。

合格的敏捷开发（测试）工程师应该具备独立完成本职工作的能力，能担当，敢于挑战，并且有拥抱变化的胸怀；尽量少给别人添麻烦，又乐于帮助团队其他人来完成工作，面临困难时知道如何利用各种资源发动解决问题的引擎，但也清楚自己不是一个人在战斗。这类人才通常有三个共性：

- 自我管理。自己做自己的工作计划，能够独立开展工作。
- 自知之明。能够发现过程、技术、产品的不足，并有果断寻找改进方法的勇气和能力。
- 自律。不需要主管的控制和监督，就能自发地完成自己的工作，并且只要承诺的工作就能够很好地完成。

我常常将这三个特点作为人才培养的目标，但是选择培养员工成为能自我管理、有自知之明和自律能力的人，还是选择招聘这样的人呢？客户和朋友们常常发出这样的疑问，基于自身的经验和立场，我会选择培养人才而不是从外边招聘。这样的人才其实要求并不高，不会从内部找不到或者培养不出来。

从另一个角度来看，招聘高级人才的做法值得一试，可是同样有风险，如果花很长的时间才能了解一个人的真实品质，恐怕为时太晚。一次招聘失误将带来1.5倍薪水损失，此外还有过去所花费的面试、培训的金钱和成本，你是否十分相信自己的慧眼呢？而即使招聘到人才，学习能力、态度，一切皆好，又该如何留住这样的人才呢？员工愿意继续与你在一个战壕里战斗往往不取决于工资、福利，而取决于你是否真正关心员工。

而敏捷方式的选择正是要将团队培养成才，让员工成为更优秀的人，这也是员工自我管理、自知、自律良好意愿的表现。我们不应该剥夺团队成长的权力，要给他们足够的关心，帮助他们快速成长起来，有多大的能力就提供多大舞台，这是现代企业竞争环境对管理者的新考验。

培养人才非一朝一夕的工作，也不是光有决心、善待下属就能做好。但是，培养人才也绝非一件登天的难事。我曾经在研发团队中培养出有企业影响力的团队，他们负责全球发布产品的主要功能架构和开发，负责最困难的国际客户现场支持工作，负责管理全球客户关系，而这只用了8个月时间。在此之前，他们有的是来自其他企业的编程人员，有的是留学归来的待业海归，还有的是本土刚刚毕业的学生。他们在本职工作中均做出了良好的成绩，此外他们还能自主学习，不断攀登高峰。更可贵的是，当团队中其他人遇到困难时，他们既尊

重战友又能在需要的时刻伸出援手，每个老员工（6个月以上）都学会了予人鱼不如予人渔，他们之间是值得信赖的队友关系。这个团队在开发中心很快建立了自己的口碑，在得到更多海外团队、经理、架构师的认可后，团队成长非常迅速，两年内从一个外围队伍发展到核心队伍，拥有自主研发、独立设计能力，并担负全球高级技术支持工作。这样一支初具规模的队伍能够在两年内和国际上有20年发展经历的先进技术团队、产品的创始者团队齐头并进，我们应该感到欣慰。

抛去大企业的资源优势，我们仅讨论对于培养敏捷人才通用的策略，看看如何培养具有敏捷交付精神的敏捷人才，如何培养敏捷主义者的基因。

5.1 如何培养自我管理的秉性

5.1.1 如何培养自我管理的秉性

我遇到过一个年轻人，聪明好学，而且非常帅，虽然没有傲人的教育背景，但是毕业于国家211重点大学，科班出身，最大的优点就是事情交代清楚以后，能够明确地按照要求做好，有时候还有让你吃惊的表现。但这个小伙子的最大问题是什么事情不喜欢自己做主，习惯了服从和不主动思考的解决方法，大事小事总要请示，总要讨论，总是将工作说得会遇到各种各样的困难，这怎么办，那怎么办。如果每次跟他沟通都给出答案，我会发现是他领导我，而不是我领导他，解决方案不好他还要跟我较劲，讨论后的心情且不论，时间花了不少，可如果这样带人，再过半年他还是老样子，而且我也没办法抽出时间做我该做的事情。于是，我拜服了他这种自然天成的“能力”，看着他天真无邪的装着各种问题的大脑袋，我换了一种方式——“忍”。忍着先不给出答案（我是个思考很勤快的人），默默地等他说完，只是简单附和。一开始我觉得很机械，不自然，他有时也会突然顿住，而过招几回，我发现他的确有难处。这家伙有个高频问题：“这邮件我写不出来”，我开始意识到是英语表达的问题（IBM的邮件用英语表达是惯例）；接着，从他的问题里我又发现，这家伙虽然看着很阳光，其实就是一个愤青，习惯性负面思维，如果你不让他发泄发泄，他就不停地叨叨个没完；再过几分钟，我发现我对他的赞美过少，导致他总想从我这里得到嘉奖，虽然不是为了“表扬”活着，但在企业中，这是一种非常需要的给养，表扬是促进合作的肥料。

在认真聆听完他的话以后，我迅速组织我的想法，并确认我有信心在短时间里结束这场会谈。

我：“Edward，这些都是你自己想到的吗？”

Edward：“是的，我昨晚都没睡好，这些问题很纠结。”

我：“你发觉没有，你的系统思考能力很强，方方面面都考虑过，这样足以让我觉得把事情交给你很放心。”

Edward：“我……”

我：“项目的确有风险，你考虑到了，我今天也听见了；但是既然我已经把重担交给你了，你打算放弃呢，还是打算还给我？目前没有人比你更加合适做这件事情。”

Edward：“我并没有说不能做，只是困难很多。”

我：“写邮件是需要功力，我也是慢慢积累过来的，一开始会慢一些，但是邮件还是有章法可循的，我给你推荐一个培训，下半年我提名你去，你看好不好？”

Edward：“谢谢老板，那是挺好的，我觉得如果有你教我更好，培训我也需要。”

我：“Edward，如果你的邮件是代表我们团队的，你可以找我帮你review；如果大多数邮件是内部沟通，哪怕是跨团队沟通，我还是希望你自己抉择，你可以找你的Mentor，我会和她说说你的学习动机，这是个很不错的向你的美女师傅请教的机会，她的邮件相当棒！”

Edward：“谢谢老板，这倒是，我怎么没想到。”

我：“Edward，我发现你有了新的进步。”

Edward：“啊？什么进步？”

“我发现你开始变得谦虚了，”我用眼睛直视他，接着说，“这工作是对你各方面能力的挑战，说心里话，我知道这对你有难度，但是，你要是能努力挑战自己，你的成长会在这个项目快结束的时候体现出来，我坚信这点。这个项目非常适合你，能够发挥你的长处，你要是放弃了我们目前不会有更好的机会。”

Edward：“我知道，这是个好项目，我会努力的。”

我：“对于项目风险的把控，你确实做得很好，我承认我都没考虑得如此周全。这很好，你是怎么考虑的？”

Edward：“呵呵，我每次拿到项目都是……略去100字……然后，再来找老板，听听专家的意见。”

我：“Edward，这很好，看样子你掌握了我们的GROW方法，不过有点小出入，你知道吗？”

Edward此时非常认真看着我：“哪里呢？”

我：“GROW方法在向客户反馈阶段性结果时，是反馈当前一致性目标执行后的结果，而不是对执行的结果各种猜测，你说是吗？你的执行力一直表现得很强，你是否需要我更多的授权呢？”

Edward：“我知道了，你的授权足够了，我就是觉得你会有更好的想法，其实我应该更勇敢些……”

我：“去做吧，做错了也不要紧，有你的态度，即使错了，还有我呢，我可以为你的错误负责，因为我相信你会做得比我更好！”

Edward：“谢谢老板，我明白了。”

省去5分钟的家常闲扯……

自我管理的养成策略

将身边的人变成“自我管理”的人才，否则管理者将疲惫不堪。在体育赛事中，双方都会展示各种各样的招数，但是他们都有一个清晰的目标——得分。必须记住，没有目标，就不能成功。作为领导者，我们希望将时间分配给关键性的工作，做高回报价值的工作，那么培养员工的计划能力一定是高回报价值的工作，因为这是一个可以复制出多个“你”的办法。

1. 目标的分享和沟通

建立“自我管理的团队”绝不是让团队无的放矢，无政府主义。自我管理团队的目的是提高团队成员的生产力，将增加的生产力和时间投资在能使团队产生更大、更长久效益的长期战略上。提高团队成员的生产力将会在他们完成组织目标的同时，自然而然地提高他们的个人能力。同时，也减少了达成团队目标的时间。提高团队成员生产力的最有效策略之一就是让他们适度地参与到组织目标的计划当中。牢记组织目标分享是一个双向沟通的过程，在你和你的团队成员就组织目标进行沟通的时候，也要征求他们的想法和意见。要强调组织和部门目标积极的方面以及产生的效益，让团队成员真心接受那些为提高生产力而制定的目标，做出可以达成的承诺，并规划出个人的行动步骤，为团队的进步做出贡献。这样，团队成员就能清楚地了解，如何对组织的目标做出贡献，同时达到自己个人的目标。

在敏捷项目中，我们的发布目标通常很抽象也很有魅力，和团队最大的沟通工作就是制定团队的发布目标计划和迭代目标计划，在大家清楚地认识、认同了团队级、组织级的目标计划后，我们会让每个成员自己独立思考，并要求其完成书面的个人目标、个人周计划甚至个人日计划。

2.认真制定书面目标

没有计划和目标，所有能唤起人类无限潜能的愿望，就会像夏日暴风雨中的闪电一样被浪费掉。闪电在瞬间挥霍了巨大的能量，然后消失在虚无的空间中，没有目标和方向。具有讽刺意味的是，闪电瞬间的光芒，似乎使得闪电过后的天空变得更加暗淡。

相反，在认真计划的基础上进行的目标设定能够提供明确的方向。目标使人长期保持兴趣和动力，使生活井然有序，能唤起所有人最优异的才能。成功必须依靠目标的不断设定和成长，即“逐步实现”的过程。尽管许多有价值的成就来自于其他活动或目标的连带效应，然而，这些成就仍然是追求预先设定目标的直接结果。目标实现后的最终效果不一定现在就很清楚，但重要的是，成就和个人生产力的增进，均是追求预先设定目标的直接结果。

写出来的目标计划能帮助我们明确最有意义的成就。锻炼员工将目标时刻写出来，能够帮助员工澄清和具体化思路。书面的目标还是衡量进展的基础。所以，当我们和员工坐在一起开会议讨论目标计划时，最高效的方式就是双方对认同的、明确的书面目标计划的讨论。

写出目标，可以让每个人有明确的轨迹可循，将干扰值减至最低，并使得员工专注目标。与之配合的其他团队成员明确了彼此的优先顺序，可减少冲突。同时，一旦员工将目标计划写出来，能让每个人清楚自己的目的，加强对达成目标的信念。写出的目标能提醒员工自己的使命和目的。写出目标计划需要列出需要实施的特定行动步骤，将目标由梦想转化为扎实的行动，直至成为现实。

而拟定目标计划的最大挑战，就是要找出一段不受干扰的时间。各种干扰，如会议、例行公事、处理大小危机，都要花时间，或把你的时间切碎，使得有效计划所需的连续性思维变得很困难，甚至是不可能。员工需要对目标计划的时间有所考虑，克服困难，安排时间来做好目标的规划。

3.专注于优先顺序较高的执行

鼓励员工专注于优先顺序高的活动。提高团队生产力最快、最有效率的途径，就是将时间用在完成重要目标的工作上。要确认团队、员工的时间花在自己认为重要的工作上，否则时间就会被一些琐事完全荒废掉，甚至花费数个小时去解决本来别人完全可以自己解决的问题。针对团队成员的疑虑，可以授权他们解决。这种方式能够节省许多宝贵时间，同时让员工有机会培养责任感、自我归属感和解决重大问题的能力。帮助员工将时间用在优先顺序较高的活动上，集中时间和精力达成自己设定的目标，将会爆发强大的执行力。

4.坚持与不完美

鼓励你的员工将注意力持续专注于某项工作，直至工作完成。设定自己的优先顺序，不要让任何干扰或事件打断员工正在进行的工作。严格限定自己花在某项工作上的时间，不能超过该工作所值的时间，或将工作授权给团队中合适的人选，这两种方式都要进行认真的评估和考虑。对于员工中的典型完美主义者，要帮助其领会将过多的时间和精力执着地投入一项工作会减少价值而非增加价值。对于自己就是完美主义者的管理者，在员工关注自己的工作时，不要去打扰和指挥，既然已经将特定的工作交给了你认定的适当人选，尽管他无法做得十全十美，只要能做到自己的75%，或者只要能够达到基本的品质标准即可，否则员工如何能学会做此项工作呢？在这种情况下，完美主义者必须学会接受不完美的失误，以增加他人的贡献，创造新的机会，并保持整体的效率和生产力。

鼓励员工自己提出规划和设定目标，鼓励员工建立达成目标的决心，帮助其理解实现目标后的益处，这对于提高员工和自己的生产效率都很重要。这些要素可以使你和你的团队持之以恒，而持之以恒是成功人士必备的特征。许多人在接受新的工作或任务、负担起新的责任后，开始时兴致勃勃，全心投入，进展很快，但不久就失去热情，结果是工作无疾而终。相反，有生产力的人会设定明确的目标，认真地规划并专注达成目标所需的行动。有恒心的人会牢记自己的目标，不屈不挠，努力追求，直至尝到成功的果实。当团队成功时，管理者的关注力就能够放在更为重要的工作上，成功价值就将持续放大。

5.允许失误

工作中的失误以及从失误中学到的东西是非常有价值的经验。如果你从未有过失误，那只能说明你从来没有接受过新的挑战。失误告诉我们什么是行不通的，然后我们会选择新的方向去达成目标。在组织中架设一道“安全防护网”，使团队成员在为了组织的目标而努力工作的过程中导致了无心的失误时，能够得到保护。这样，团队成员就能放下自我保护主义的包袱，充满热情地工作。为了鼓励团队自主制定计划和设计方法，我会告诉他们：“我们会有不同意见，我希望你们能够勇敢提出自己的想法，如果有一天，在座的各位所持意见与我相左，而我又不得不为结果买单，那么我会认同各位的做法并一并承担结果。”这样做的好处是让员工能够专注于目标而不被打搅，经过长期的实践，他们很勇敢地做目标设定、发展领导力和管理时间，这也对所有团队成员的职业发展非常重要。我的团队是个民主的团队，我允许团队表达不同意见，哪怕是让领导“没面子”，哪怕会有失误，我权当在做对构建有效组织、组织持续进步、生产力提高和解放我的手足去从事我认为最有价值的工作的长期投资。

5.2 如何培养自知之明的秉性

5.2 如何培养自知之明的秉性

人最难理解，人的性格甚至是多变的。从事客户项目已经快一年了，从接触诸多国内外客户的体会中，我逐渐总结出客户的价值取向、行事风格和做决定时最重要的考虑因素。但是，如果不经过深入了解，很难把握得准，为了更为准确地把握客户的心理，我研习了心理学、性格学，对于一些简单的行为，比如谈话时不经意地看手机、抓鼻子、眼球向左右不自主地转动，我能够了解其影射出来的心理特征。清楚一个内向型、务实人格的人，给他上再多的课、演示再多都没法打动他，而是需要创造机会让他自己体验，看到不如摸到，感知不如行动。尽管如此，我依然发现，在我面试许多性格特征迥异的人时，每个人都或多或少地希望“深宫锁院”，不希望在人前表现得认真和简单。真的简单的人，总会说，“不全对，我过去是这样，但我现在不是……”真的社会阅历丰富的人总会表现得略有些吃惊：“还行，都有些，但视情况而定……”后者的面谈通常花费略长的时间，排除掉客户不希望在人前暴露自己真实一面的因素外，他已经记不得“本来的我”，而是染尽铅华，成为“意识中的，那个应该是这样的我”。的确，人的阅历越多，经历越沧桑，性格特征也就越发模糊，仿佛已经砍掉了棱角，打磨光滑。因此，培养自知之明的秉性一定要从年轻时做起。

我遇到过这样一位员工，她是个标准的“理想主义者”，关注一件事情是否有意思，关注周围的人是否从一开始就很友善、是否愿意无偿地帮助她，对于符合这个价值体系的人，她称之为“朋友”，否则皆为“对手、敌人”。做任何判断时，她都会基于自己的价值观念，而不是客观证据和因果关系。

虽然理想主义者是一类关心个人成长和非常易于理解他人与自我的人，但因为他们非常崇尚人与人之间和各种关系中的真实、正直，所以非常痛恨“欺骗”等破坏“底线”的事情，无论是自己的事情还是别人的问题，只要触及价值底线，便容易纠结其中。记得有一次，我们私下向她了解最近提早离开公司的缘由，她怒不可及地表达了对于同事“出卖”她的愤慨。还有一次，在年中绩效考核时，当我发现超级理想主义的她竟然给自己全部考核项以满分时，顿时僵化，而后又面对面谈及年终评定，依然没少头痛。

从心理学角度来看，“自知力”的缺乏，会使得我们过于理想化地透视自己，理想化地认为周围的人也采用同样的价值体系，这会让我们很容易为了失败开脱责任，陷入责备他人的行为模式，纠结于往事不能自拔，几乎责怪身边的一切。当我们继续将人生所有的过错和不幸全部归结于恶劣的环境、不良教养或者遗传基因时，我们的身与心就越发远离梦想和幸福了。而人如果要幸福、要成功，必须先认清“我是谁”，发展自知力，建立真实“自我”和“本我”的同一性，从内心到现实构建这种一致性，创建一个自己决定、协调一致、更为公众所熟知的独具“统一风格”的自我。在理想、职业、价值观、人生观等的思考和选择中，统一的自我会如沐春风，拥有愉悦和内心平和的处事态度，随着愉悦在身体和思想的深入，就会对周围的人产生吸引力，因此会带来更多的机会，我们将会明了自己应从事的事业，学会自发地、出色地完成工作，并为社会创造价值。大多数杰出的人都坚持追求于自身十分重要的事情，追求意义重大的人生目标。企业自始至终都是由一个个人组成的，无论在世界的哪个角落，无论出身、容貌、信仰如何，你都在与这个企业的客户、员工发生着信息和感情的联系。如果没有健康的身心和出色完成工作的能力，那么，经由这些员工工作所拼凑出来的结果将无法帮助企业构建一个成功的故事。

具备自知能力的员工，可以帮助管理者看清旧的体制与新管理流程设计的切合点，使得管理者更有探索性、更成功地在新的变革中找到平衡。我们可以找出30名来自各个相关部门的同事，将他们分为3~4组，重点关注正处于积极重建中的管理流程，问问每个小组：

- 谁是这个流程的所有人，谁能够改变它？

- 这个流程的服务目标是什么，它的最大绩效表现于何处？
- 谁是流程的参与者，他们的意见是否得以体现在流程的改进中？
- 这个流程中，关键的决策者是谁，谁的意见最为重要？
- 什么事件将驱动这个流程的进行，是主观驱动还是客观执行？

通过小组讨论，我们要求每个小组给出一个改进所选流程的重点优先顺序的清单，并将其做出的具体行动目标计划和目标放在公司的内网上，激励公司其他员工继续参与活动。这样，“管理者”所涉及的流程、策略才能够更好地实施。难能可贵的是，即使在传统的国内企业，这样的活动开展仍然非常有益，许多员工提出的改变清单都直接挑战着那些长期令人不堪和恐惧的管理流程。这一点最值得鼓励，因为顽固的、令人纠结的不适宜“基因”即将以最优的方式被剔除出去。可以通过民主、头脑风暴的方式让员工参与流程改变，那么，如何做到鼓励而不是压制对方的声音？如何使流程对来自一线的需求和考虑有更强的响应？如何使基层的智慧和经验在政策和战略形成时产生更大的影响？

自知力的养成策略

要建立自知力，可以分如下几步。

1) 发现自己，了解自己原生态的一面，了解自己究竟是谁。

发现自己，找寻自己真实的“性格”。为什么强调真实呢？原因是一经走入社会，在社会上总会学会伪装自己和保护自己，但是往往也容易迷失了自我，不再理解“我”，使得“我”的潜能无法很好地释放。而了解自己，寻找真实的“我”，有益于发现自身与他人的价值区分，清楚自我实现的不同途径的诉求。在人生陷入停顿，必须重新开始的时候，这些自我发现能够帮助我们站稳脚跟，调整心态适应全新的自己和审视自己。

应当将人生目标建立在自己的核心价值上，你是你，因为是你，所以成就了你。如果你是内向的人，你不可能一夜之间变得外向，但可以理解为什么你不善于在公众场合左右逢源，而他却可以。你不会嫉妒“俗不可耐”的社交行为，你可以用平常心看待那些场面上称兄道弟、觥筹交错的逢场作戏。你会看到自己的优点是能够通过自身的力量从低谷中走出，而无需寻找一个可靠的“肩膀”来哭诉。你了解你可以同时交几个非常好的朋友，但是你不可能是所有人的朋友，你不必羡慕那些左右逢源的外向的同事，因为你结交的朋友定是专注的深交。而外向的朋友一定不要辜负自己强大的行动力，你习惯通过行动-思考-行动达成目标，不用因为外人眼里那个未经深思熟虑就立马横刀、独步天下的冲动的你而失落。

性格的差异构建了我们这个丰富的社会群体，性格虽然没有好与坏、多与少之分，但上帝是公平的，当他给予你一件令人钦佩的能力的同时，也会赋予你一个你不希望尽人皆知的缺点，每一种性格都有潜在的优势以及可能的劣势。尊重他人在沟通、决策、接受信息上表现的种种不同，选择适合自己和对方的方式合作才是自知力的充分体现。

你可能要问了，怎样才能更好地认识自己？事实上，很少有人真正了解自己，苏格拉底曾经说过“未经反省的人生毫无意义”，指的正是此意。

小说或电影中通过人物的经历、心理斗争、对白以及行为来诠释主人公的内心，我们对他的情感、愿望感同身受。那么，可以用同样的态度客观地观察自己，透视自己的人生价值，把它们写下来和你的几个亲密且坦率的朋友分享，看看他们如何认知你，以及大家的认知与你自己的认知是否一致。思考你的成长环境和经历，过往中的家人和朋友如何影响了你的处事风格，人生中最有意义的事情是什么，这一切都可能帮助你找到“原本的自我”。

2) 撕下标签，切勿习惯或顺从负面的情绪给自己贴上不良标签，不要怀疑自己。

想象一下：你有一个3岁半的孩子，他每天都会赖着你抱他一会儿，虽然你很想让他自己

走路、自己上楼，但你不得不钦佩孩子的毅力远在成人之上，他的果断和坚持让你无法拒绝。可是，你却说他是个“娇气包”，每每孩子需要爱的时候，他都会赖着不走直到你抱起他，原本的天伦之乐因为你的标签改变了味道，从此以后，孩子开始怀疑甚至相信自己就是个襁褓里的娇气包，除非家长不在身边，一切问题都不需要靠自己，即使当时不喜欢“娇气包”的称呼，他还是最终承认自己就是个“娇气的孩子”，和其他“好孩子”不一样，认为自己是个“有错的人”。“娇气包”就是一个标签，生活中还有许多自我设定的标签，诸如，“我就是个非专业人士”，“我怎么可能和年轻人再去竞争写代码”“我是个受害者，没人受过这等折磨，我要崩溃了”……如果对这些负面的标签照单全收，我们就丧失了好好倾听自己内心声音的机会。

“天使就在身边，而你却听信了魔鬼的话！”

有些父母说，你一无是处，永远不可能出人头地。贴上这个标签，你就很难提升自己的人生。对此，很多人无所适从，因为没有人教会你走出来，不知道该做些什么，甚至笃信了这些标签，直到有一天你意识到，这世界上并非只有你一人贴有标签。而撕下标签的方法，就是不要让他人来定义你。你必须定义你自己——只要你知道选择的方向，就一定能够做到。

年轻的时候，我也曾血气方刚，读到这里，有些人或许会猜测下面将是我犯错的故事。非也，我非常感谢上帝在我年少的时候赋予我坚持、自信和迎难而上的品质，在从事科研工作和研发工作的几年中，我似乎一直都在抗争，我最不愿意听到的，但也是最喜欢听到的一句话就是“你不能……”20出头、有些桀骜不驯的我只要听到这句话就一定要发愤图强，不但学习赶上，还要成为专家才能放手。我经历了许多磨难、吃了许多苦，有一天我对自己说，真怕对不起我受过的苦……但是，我就是这样从标签中走出来，相信你也能自己走出来。

3) 改变意愿，并接近那些能够帮助你、支持你的人。

相信自己和独立思考是可贵的，在艰难改革过程中遭遇艰苦、波折、失败的时候，我们能够有勇气依靠自己走出来，但却又可能在成功喜悦的背后，盲目笃信以至于某天成为了一个固执己见、过于执着的人。必须承认，在前进的道路上，我们几经陷入僵局，而世界已经发生了变化，并且这种变化从未停止。我们停下来思考，在需要意见和方向的时候，去找找能够帮助我们、支持我们的人，而最好的支持者并不是直系亲属，他们与你的浓厚的亲情和长时间的共同生活很可能阻碍他们接纳崭新的你。因此，我通常会去寻找直系亲属外的朋友、上司、长者帮助我重新建立起目标和勇气。事实上，扩大帮助和信息的来源，同那些事业上有过重大转变的人保持联系，关注他们与自己不同且值得学习的地方，都非常有帮助。

大部分人害怕变化，在组织中尤其容易形成一种基于经验的保守主义。尽管我们看到了外部环境的变化，但仍然采用老旧的思考方式去理解新的问题和挑战，就像一位心理学家说过的，我们的大脑总是愿意理解那些我们希望理解的事。所以，这种组织惰性使得我们无法适应变化的世界。

开放心态对待周围发生的新情况、新问题。不要让主观偏见妨碍你做出正确的判断，不要急于做出判断，要认真聆听他人的意见，设身处地地为他人着想，不要只想着争辩。自我反省，明确自己的反对是基于客观事实还是个人偏见。

趋势和潮流是不断变化的，而且不一定是不适宜的。同个人一样，企业也会追逐潮流而动。通过观察和学习新的趋势，才能使自己从专家的角度进行思考，成为变革的主人，这会帮助你更加自如地掌控未来的变化。

乐观地对待变化，把变化看成是机遇和挑战，而不是一种威胁。考虑自己可以从中获得哪些益处，而不是会失去哪些东西，关注自己可能遇到的机会。

4) 不断学习，活到老、学到老。

“口袋空空并不会妨碍人们前进，只有头脑空虚、心灵空无一物的人才会退步不前。”——诺曼·文森特·皮尔（美国著名商业思想家）

学习是一个长期的过程，而不是一次次短暂的事件。做终身学习者需要我们每天都学习新的知识、掌握新的技巧。比起为了学习而学习，学习的另一种更有价值的目的是提高自身素质，追求更高效、更完美的生活。学习非常重要，所以我们应该培养学习的习惯，用开放的思想和心灵去迎接新的学习内容，要意识到：

- 人可以活到老、学到老。
- 学习是生活中一种相当自然的过程。
- 可以将挑战和压力变成学习的动力。

人的性格影响了学习的方式，所以，如果是内向型人格，你会喜欢学习-尝试-学习，如果是外向型人格，你会习惯尝试-学习-尝试。总而言之，保持旺盛的求知欲和从错误、失败中学习的态度尤其重要。

如果不清楚自己哪些方面做得不好，可以向支持你的长者请教，在他人发言时注意倾听，不要产生排斥心理，必要的话，还可以进行记录，以表现你重视他们的意见。无论你是否同意他们的观点，都要对他们提出的建议以及在此事上花费的时间表示感谢。

如今，很多公司越来越倾向于让员工个人为各自的学习和技能提高负责。很多公司都在缩减用于员工培训的开支，并将培训视为对员工的一种额外奖励，而不是将培训作为提高全体员工工作技能的机会。所以当我们有机会和上司进行发展计划探讨时，需要谨记几个原则：

- 学习计划应该与工作计划密切联系。集中精力，学习那些有助于工作发展或者提高客户服务水平的技能。
- 学习计划应该详细具体。你不可能笼统地学习概念，因此在计划中应该详细列出需要掌握的具体技能。比如“沟通技能”就比较笼统，“如何倾听”、“如何在公众场合演讲”就比较具体，如果可以，应该在计划中标记何时参加了哪些培训课程。
- 学习目标和内容应该实用。如果学习到的技能不在一个月内使用，大多数人都会因为过于忙碌而忘记这种技能。
- 学习目标应该面对现实。过高的目标会给自己带来超额负担，在没有掌握一项技能之前，迫于压力学习另一项技能完全是一种浪费。

5) 慎于自视，善识人长。

必须慎于自视，初涉社会之时尤其如此。人皆自视甚高，而往往越是精神层次空无的人越严重。谁都会梦想自己是奇才、人中龙凤，指日飞黄腾达。但是，望之心切却到头来一事无成。目标甚高固然带来动力，但是万不可达到荒谬的地步。除了理性、对客观事实因果关系的尊重以外，没有什么灵丹妙药能够根治这种愚蠢。每个人必须清楚自己的能力和处境，对自己的认知必须符合实际。

三人行必有我师，世上无不能成为人师之人，在我很年轻的时候，曾经因为各方面成绩优秀总是自我感觉良好，常常因为得意过头、说话带刺而伤害了同学感情。回想我的高中、大学生活现在都有些后悔，好在同学们都曾年轻过，如今即将迈入不惑之年，我发现大家都没有将这些小事放在心上，而我深深忏悔，有时常常希望自己重新回到大学一年级再来一次。在经历了挫折、感悟和清醒之后，我发现那些成绩优秀的同学，有些并没有因为这些能力而成就理想，而如果拥有良好的人际关系，我们将会更简单地做好每件事。

一开始，我非常害怕重新陷入无休止的勾心斗角之中，希望我的周围有更多的朋友而不是对手。后来我心生一计，每每大家聚会的时候，我有意表现得特别谦虚，像那些人缘很好的同事和朋友学习如何说话更讨人喜欢，如何倾听和恰到好处地响应。尽管心里觉得有些别

扭，但当时只是想换来更多没有敌意的朋友。奇怪的是，当我逐渐花费时间在倾听上时，我不仅从他们身上看到很多优点，很多快乐、幸福的东西，而且渐渐地产生了兴趣，连我自己也忘了自己是表现得低调还是真的低调了，很快我成为了能够在一旁鼓掌、快乐的人。而这个时候，我周围的人更愿意向我倾诉，愿意向我索求帮助。当我们自己放下架子的时候，开始对别人的事情感兴趣的时候，朋友们非常愿意倾诉同时也乐于鼓励和赞赏你侃侃而谈的才华，而更加真心愿意听你分享见解。

做一个在路边鼓掌的人对我来说并不难，而现在已经是一种习惯和品质。也许有人会说：“我不愿意在路边为别人鼓掌，我不愿意以别人为中心，我对别人的事情不感兴趣，我压根就不想打听别人的事情。只要能赚钱、生活，我愿意关心的只是我自己的目标，管那么多干嘛？”诚然，你可以做出这样的选择，但是如果你认为你对了，那么我们的处世哲学就错了。人不能只看到自己，在过去、现在和将来，你的能力远不能达到思想的高远，如果没有与人合作的能力、没有信任的基础、没有倾听的心态，你只能做很少很少的事情，此生都会留有遗憾。交新朋友、和陌生人产生摩擦，进而了解发展成为知己似乎已经被社会的浮躁和现实慢慢磨灭，所谓以利交友者利尽人散，以色交友者色衰则疏，以貌交友者久之则腻！唯有以心交友，方能永恒！以心交友，就要懂得欣赏和关心。

在团队中，真正懂得认识自己、欣赏别人才能建立良好的合作氛围，无论是谁都有过人之处。在IBM的团队管理理论中，帮助团队成员学会关心他人、赞美他人是高级管理的必修课程。

学会了赞美，学会了发现他人的优势，而同时了解了自身的不足，我们方可从持续化改进中学习、提高，而不会误入歧途或是讳疾忌医。

5.3 如何培养自律的秉性

5.3 如何培养自律的秉性

自律，即自制力，通过自律我们知道在面对问题时如何以坚毅、果敢的态度从学习与成长中获得益处。我对自律的理解直到研习了美国作家M·斯科特·派克的《少有人走的路》方才全面突破，简单地说，自律是以积极而主动的态度解决人生痛苦的重要原则，主要包括四种技巧：推迟满足感、承担责任、尊重事实、保持平衡。而学习自律的关键要素在于态度，如果无法直面痛苦却总是逃避，那么这些原则和技巧永远不会起作用，你也不会从中获益。

自律的养成策略

缺乏自律能力的人，在工作中表现为缺少自尊、自爱和推卸责任。我们都不希望与我们合作的同事表现得这么不靠谱，因此，我们应该培养自己的自律能力。从心理学角度分析，自律同样是解决我们人生问题的首要要素。

1. 推迟满足感

还记得我们是如何改掉拖延作业的习惯吗？我有个同学，总是拖延作业，每天早晨学习委员收作业的时候，总是在他那里花费很长时间听取各种理由然后才可以拿着半本作业离开，而我们总希望帮助这个同学。

“放学了，我们一起做作业吧。”我说。

“放学了，脑子不好使了，我要运动下，运动完再做。”他回答，“你要做，就自己先做吧，回头借给我……”

我扭头回去做作业了，可我并没有带着作业来找他，此时，他显然已经离开了球场，在放学的路上打起了玻璃球。

我问：“你作业做了吗？”

他没动静。

“我作业已经做完了，”我接着说，“我们回家吧，我可以去你家帮助你完成作业。”

“有你帮助就好了，我正好再玩会儿。”他回答。

我开始和他一起打玻璃球，很晚才一起回家，因为太晚我没有去这个同学家里。第二天早上当我问起他的作业情况时，他说：“作业没做完我就困了，一觉睡到第二天早上。”

就这样，这个同学很少能够完成作业，直到最后自己选择留级，我们没能再成为同学。

如今，我回到家乡偶尔会听到他的近况，据说在一些地方打打杂，已婚，且有了小孩，日子过得一般。每每回忆这段往事的时候，我总觉得如果当时能够意识到“贪图暂时的享受”对他的人格形成的重大影响，他一定不会像今天这样。

从小就可以学习自律的原则，避免只图眼前安逸带来的不利。例如幼儿园里，有的游戏需要孩子们轮流参与，如果一个孩子能够多些耐心，暂时让前面的同伴玩游戏，而自己等到最后，就可以享受到更多的乐趣，他可以在无人催促的情况下玩到尽兴。对于家庭作业，也是实践“推迟满足感”的最佳手段。孩子满12岁时，无需父母的催促，首先做完功课，再去玩。到了青春期以后，就会学会先处理棘手的问题，而不会拖延工作。

而很多青春期的少年缺少这种健康的心态，许多孩子也不懂得推迟满足感的好处，缺乏“先吃苦，后享乐”的意识，而是恰好将次序颠倒过来，他们很容易成为“问题学生”。尽管他们的智商与别人相比毫不逊色，甚至超出一般孩子，但他们不肯用功学习，成绩永远落在别人后面。他们说话和做事全凭一时冲动，只要心血来潮，或者翘课，或者出去打架生事。司空见惯的“先享受，后付费”即使在成年人中也不在少数。

这种心理是如何形成的呢？心理学家推测是因为家庭教育的原因，其一是错过了教育孩子学会自律的最佳时机，例如做完作业再去玩；其二是自身本来就很难自律。而在年幼的孩子心中，父母位高权重，孩子缺乏模仿对象就把父母亲处理问题的方法全盘接受了，并视为金科玉律。父母懂得自律、自制和自尊，生活井然有序，孩子就会心领神会，并奉之为最高准则。父母的生活混乱不堪、恣意妄为，孩子同样照单全收，并视为不二法门。如果父母在家动辄恶语相向、拳脚相加，在孩子面前缺乏起码的自制力，缺少长辈的尊严和理性，那么如何希望孩子能够自律、自尊、自持呢？

在工作中，不拖延工作，遇到问题直视困难，不退缩、不逃避，对于棘手的问题先入手解决，让后面的时间过得更轻松，这样规划才科学。我们都有习惯性的最佳工作时间，这段时间中我们头脑更清晰、体力更充沛，有的人是在上午，有的人是在下午，我们应该把这段时间留下来处理最优先、最棘手的问题，以克服拖延工作的坏毛病。并且，要求团队也是如此，这样工作绩效和团队自律的培养都能够收到很好的效果。

2. 承担责任

不能及时解决难题，难题将会像山一样压在我们面前，阻断我们前行的路。

很多人忽略了其中的道理，遇到问题的时候避之唯恐不及，认为“这不是我的问题”，显然，指望别人解决问题不是聪明的做法。最好的方法是，我们应该勇敢地说：“这是我的问题，我希望现在开始从……入手解决。”相当多的人只想逃避，他们宁愿这样自我安慰：“出现这个问题，不是我的而是别人的原因，是别人拖累了我，是我无法控制的第三方因素造成的，应该由别人替我解决，这绝对不是我的问题。”

趋利避害，逃避责任，这种心理趋势令人遗憾，有时甚至达到了可笑的程度，这种人只要有可能就会嫁祸别人，或者推卸给上司。我遇到过这样的员工，当一切顺利的时候，他总要来问你，这么行不行，那么行不行，一切都需要你明示或者暗示他才能够履行职责；而遇到难题的时候，他会告诉你这有多严重，他有多么恐惧做出选择。我只好一次次重申，授予他权力的同时，我的确知道我需要花更多的时间处理他可能遇到的本应该由他来解决的问题，但是，我希望他相信自己能够面对或者改变现状。其实，仔细分析这种力图把责任推给别人或组织的行为，这意味着他永远甘愿处于附属地位，把权力和自由拱手交给上司、组织。为了远离责任带来的痛苦，数不清的人甘愿放弃权力，实则是逃避自由。

不含糊地说，每个人不同程度地都有推卸责任、逃避风险的人格。人在职场，正确评价自己的角色，判定自己该为何人、何事负责，是我们的责任，也是无法逃避的问题。评估责任归属，必然与我们理解的“权力和地位”有直接关系。我遇到过IBM公司的一位技术团队负责人，据说在客户那里他以IBM技术负责人自称，有一天客户现场发生了很严重的状况，技术团队同时派出了他和其他相关部门的员工，而这个技术团队的负责人所表现出来的行为是，并不组织各个团队协同工作，而是用客户的口吻对各个团队施加“政治”压力，将“责任”和“问题”完全转嫁给与其合作的其他人。当问题开始步入分析和研究阶段时，各个部门各司其职，一个重要的环节是客户、IBM各个团队的信息需要达成一致，我们必须派驻团队领导参与每周的阶段性总结和讨论，尽管客户多次邀请这个技术负责人参与会议，但都被以各种理由婉拒，而此后因为沟通环节的错位，某团队竟然在一天内收到来自客户的感谢信和内部上层压力两封邮件。承担责任，否则让有能力的人来行使权力吧！

3. 尊重事实

尊重事实，如实看待现实，杜绝虚假，因为虚假与事实完全对立。我们越是了解事实，

处理问题就越是得心应手；对现实了解得越少，思维就越混乱。虚假、错乱和幻觉只能让我们不知所措。我们每个人都有一本“字典”，从懂事起，就用这本字典来解释看到的、听到的事情。这本字典并非与生俱来，我们在人生中不断谱写和丰富这本字典。凭借这本字典，我们同人生的生词、语法错误和补充信息打交道。当这本字典准确无误时，我们就能够正确地寻找到真理，得心应手地处理问题。

而很多人或是不善于使用字典，或是字典内容非常概括和模糊，在通往彼岸的途中，原有的字典狭小、模糊、粗略，对世界的认知狭隘而偏激，但却自以为神圣不可侵犯，世界观没有任何瑕疵，听不进去新的意见，对于新的信息和资讯没有兴趣。而只有少数幸运者得以保持充分的开放性继续努力，不断探索和开辟对世界的认识，直到生命的终结。

我们从弱小的、依赖性很强的孩子，一点点地成长为强有力的、被他人依赖的成年人；我们也会生病和衰老，力量再次消失，变得虚弱和更有依赖性。我们的字典也应该随着外在环境和内在力量的变化而丰富起来，才能成就人生观、世界观。当我们不再抗拒新的信息，不再害怕异端邪说，不再考虑如何捍卫迂腐、过时的字典，我们就开始有了开放心态和尊重事实的成熟。

逃避现实的痛苦和不幸是人性使然，只有通过自律，才能够克服现实的痛苦。我们必须尊重事实，尽管这会带来痛苦，但远比我们的个人利益和暂时的舒适更为重要。我们必须淡化暂时的不适感，追求真理而不是幻想，并愿意承担痛苦，让我们的心智走向成熟。

团队领导者在鼓励团队自律、尊重事实、承担责任的同时必须身先士卒，以身作则。你希望别人怎样行事，自己也得怎样做，但是不要期望别人一定会向你看齐。如果他们这样做了，要表示赞赏。如果没有，也要坚持自己，因为毕竟你自己的行为是无可指摘的。在团队会议上扮演一个积极的角色，团队会议是相互沟通、解决问题、制定计划，以及做出决定的大好机会。这些会议中你的表现对于团队培养自律、尊重事实和承担责任的人格非常有利：

- 在会议中自愿承担责任，例如担当会议秘书、记录员或者计时员。
- 始终围绕会议主题进行讨论，避免打断别人，不要指手画脚，避免支配会议讨论。
- 鼓励与会者与他人一起分享他们的想法，倾听他们的想法，对意见保持开放心态，要接受他们带来的影响。
- 不要掩盖问题，或装作视而不见，如果你感到不高兴，就应该告诉别人，不要让问题比实际情况恶化或扩大化。

4. 接受挑战

固步自封、逃避挑战，可以说是人性的基本弱点之一。不管现实如何变化，我们都有自我调节的能力。逃避挑战是人类的本性，但并不意味着它是恰当的态度，不意味着我们无法做出改变。在敏捷要义中提出了“要拥抱变化”，在IBM的员工手册中也有同样的观点。

在工作中，我们可能不允许别人质疑，可能会对员工说：“我是你的老板，我让你做你就去做。”“你竟然有胆怀疑我，还要向我挑战，你最好想清楚，最好别让我知道，不然就卷铺盖滚蛋。”生活中，我们会对朋友说：“说点好听的好吗？别再说我的不是，不然我和你翻脸啦！”而当我们年纪大了，又会对儿孙说：“我都活这么大岁数了，吃的盐比你吃过的饭都多，你为什么要和我过不去，你凭什么对我指手画脚？”

你希望成为这样的人吗？如果你的选择和我一样，那么请相信，从现在开始接受他人的尖锐挑战，将磨练我们的意志。当我们虚心聆听别人带有讽刺性的意见，不抗拒批评和指责时，表面上，这种做法显得非常懦弱和没有勇气，但实际上，我们远远比那些没有勇气直面挑战，甚至使用针锋相对、欺骗的方式来回避问题的人有勇气得多。

然而人接受的更多挑战却并非来自外部，有多少时候你会受到外界的挑战呢？我的工作

是每天面对客户和一些稀奇古怪的发散性的问题，还有种种夹杂着大量负面情绪的质疑和挑战，仅仅如此而已，我感到更多的挑战来自于“自省”。在敏捷会议中有一个叫做“自省”的会议，不过没有受到高度重视，这个会议首先就设定了一个“过于和平”的基调，对意见没有保持开放的态度，也没有考虑让参加会议的人充分发挥主观能动性。甚至，有人讽刺这些喜欢提出意见的人为“刺头”，从来不做自我怀疑。从我经历过的几个案例来看，这种压抑的会议非但没有让团队和谐，反而制造了一个个症结使得团队成员彼此沟通不畅通，相互猜疑。“自省”会议往往成为另一个极端，那就是口诛笔伐团队中的异己，当大家彼此足够直接但政见相左时，这种斗争尤为凸显。而会议的最佳方式是，大家分别“自省”，自省的方法是，当说到与他人相关的方面时，只做客观陈述，但说到与自己相关的部分，一定要深入剖析，做思想和行为方式的自省，哪些曾经的误会需要澄清，哪些好心做坏事的举动或者失误需要承认，如何在以后的工作中得到提高并请团队其他成员共同见证和帮助自己提高。出色的团队会从自省中飞速成长，而绝不会去指责这种近乎陷入自我怀疑的行为，人优于其他动物的特征是，我们有特殊的“大脑额叶”帮助我们通过自省获得生存能力。自我反省和自我审视对我们造成的痛苦，大于观察外在世界的痛苦，所以，很多人逃避前者、选择后者。实际上，当我们成熟地面对和献身真理时，痛苦将显得微不足道，自我反省的快乐将远远大于痛苦。

让团队学习接受挑战的有效方法有如下几点：

·让团队“自由联想”、“说出真相”，“想到什么说什么”看似没有意义，没有关系，有些事情会引起很多深入的自省。

·管理者或者组织会议的人只需要静静地聆听，让团队充分发挥，不要对所有的内容进行详尽记录，让大家在没有心理压力的环境里“喋喋不休”地倾诉。

·鼓励诚实，在诚实的气氛下团队才能够学会接受自我挑战，接受自律，才会说出真话，如果发现有人满篇谎言，立刻请他闭嘴，让其他人发言。

·感同身受，如果有人努力说出诚实的、深刻的问题，你真的能够感受他的痛苦，还需要鼓励他的勇气。

5.4 提供上升空间

5.4 提供上升空间

无论企业规模大小，只要职能健全，通过转换岗位和承担不同的工作就能够培养员工多方面的能力，这是大家都认可的。所以，往往初创公司有更多的机会，有更大的舞台。作为明智的管理者，一定会从分配有价值的工作给最需要的人当中受益，既减轻了工作负担，对于员工而言又有了绚丽的舞台和上升的空间。

在我带领Rational Publishing Engine（RPE）团队的那几年，尽管权力微不足道，但我仍然承诺：“你有多大能力，我就给你多大舞台。”这被后来的成绩证明是多么明智的决策！发挥他人才能，为员工找到适合的领域，并让他通过完成自己有兴趣的工作，达到个人绩效的同时实现团队目标，是我执掌团队以来一个响亮的成绩。几年来，除了一名成员因为移民新加坡离开团队外，其他所有人均非常稳定地为团队尽心尽力。团队也在短短的两年内，从一个萝卜头成长为能够独立设计和研发，并支持全球客户 的全能技术团队。

这里需要说明的是，要帮助团队发挥其才能而不是帮助团队提升绩效，这两者有着根本的区别。以绩效为目标是站在管理角度，以结果论英雄，而发挥其才能是站在培养角度，以成就自身成长、对比自身过去的成长空间大小论英雄。绩效需要公平，而成长需要差异对待。

为了准确定位差异性成长空间，我将大量的精力和时间用于分析团队成员的优劣潜质，帮助团队建立各自目标、寻找成长舞台。我的团队总能在期望的时间内感到自身的提升，品尝成功后的果实，即使是一件非常小的事情，例如克服了赖床、开始使用流利的英文说明工作，也成为我的团队成员产生满足感、成就感的“兴奋剂”。一个不断感受到自身成长的队员，无论多么辛苦，压力和负面情绪都无从而入。组织同样如此，因为组织就是个人的集合，许多组织出现问题大多是由于组织停止了成长。

2010年秋至2012年春，不到两年的时间里，RPE中国团队就从一支功能测试团队迅速成长为独立设计、开发并承担100%全球客户支持和60%~75%新功能开发工作的成熟团队（参见图5-1），成长速度和团队潜在发展空间让美国的上司和罗马尼亚的总架构师感到惊愕。团队中每个人在两年间均获得“杰出贡献”和“快速成长”奖项。除此之外，中国团队领导了提升RPE产品的“客户可感知质量”的竞赛，在整个Rational的产品族里，唯有RPE产品在9个月内将产品的PMR（客户提出的问题、缺陷）降低了50%，将其中与产品整合相关的PMR降低了30%。当我离开Rational开发部的那一刻，我仍然是IBM Rational全球产品整合质量提升组织的领袖。

我目前的工作多为拜访客户和解决线上的问题，很少回到后台实验室接触本部的同事了。但是，每当我和新老同事坐在一起吃饭、聊天的时候，我们都会为过去卓越的成绩、精彩的时光而兴奋不已。

过去那段日子里，我们很少谈论绩效话题，因为无论如何，团队中的个人绩效其实是一个相对的结果，不代表个人的成长空间和结果，在我的能力范围内，我也尽可能地将绩效差异带来的负面影响降到更低。

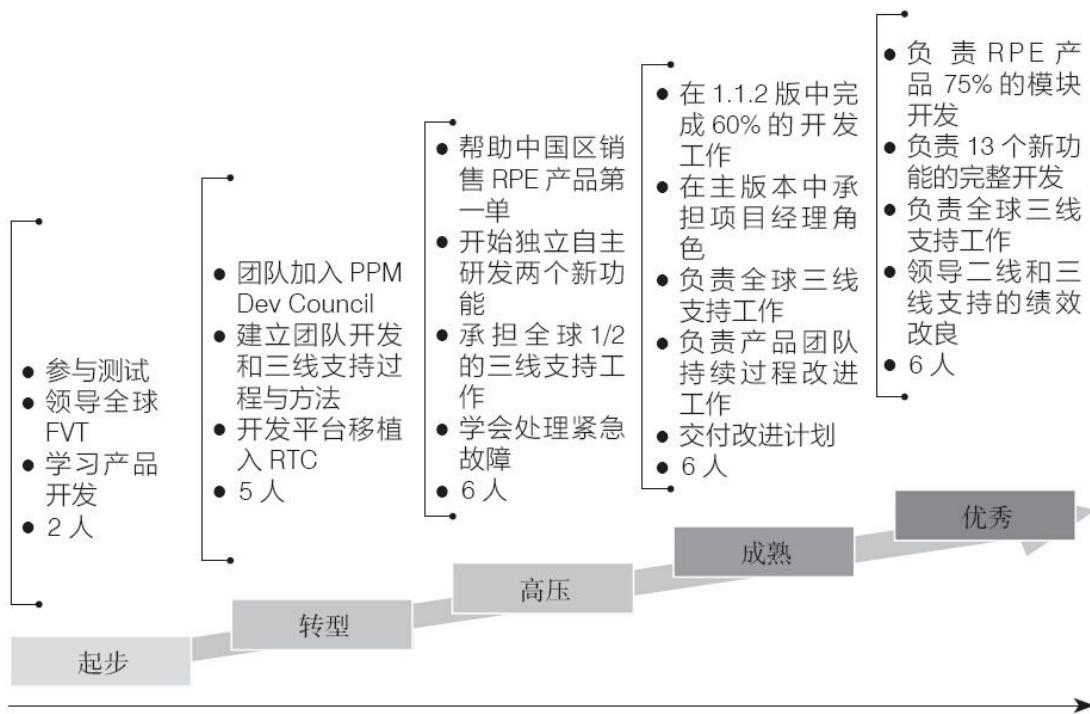


图5-1 RPE团队两年内的快速成长轨迹

相比那些特别看重绩效、看重升职、为了升职而努力工作、为了表现自己而学习的团队，我的团队反而是最少让老板操心、破格升职比例最高的。其实，最终你我都将离开这家或者那家公司，位置、工资水平什么的都是浮云，关键是个人能力是否达到下一个平台的需求。而现实中，绩效常常成为个人成长的绊脚石，因为绩效是以组织目标建立起来的，是没有人情味的，不是量身定做的，绩效考评甚至可以毁了一个人的信心，一个人对企业的一片赤诚。

一个高明的做法就是将团队绩效和个人发展道路联系起来，团队领导者“制造了梦想”并做好了“个人目标设计”。就像马云所说，我们统一不了每个人的思想，但是我们可以统一目标。这个做法的关键是如何将团队的目标与个人目标进行重合，如何能够让员工满足于管理者的指引。实践证明，当个人目标和组织目标重合时，个人在组织中就更容易成功；如果个人目标、组织目标、社会目标重合，则会更加成功。

5.5 团队的沟通

5.5 团队的沟通

如果从事过敏捷实践或接触过敏捷团队，你一定会发现，敏捷交付的环境中，无论线上线下的沟通都要比平时多很多，沟通成本数倍于传统交付。事实上，理由很简单，因为层级拉平了，每个人所需要面对和协调的角色也多了，自然需要更多的沟通环节并花去更多的时间。这里讨论两点：第一，如何高效地“听”；第二，如何魔法般地“说”。

5.5.1 “听”的哲学

“听”谁不会呢？你一定会觉得这个话题未免有些幼稚。

我们参考一下有关沟通和信息流失的科研成果，即图5-2所示“沟通中的信息损耗”。一般的沟通中，平均约有20%的信息没有有效传达，而倾听者只能关注声音信息的60%左右，紧接着大量的信息在大脑中接受处理，通过大脑筛选并理解的只占原有信息的50%。这就是听或许很容易但理解却很难的道理。

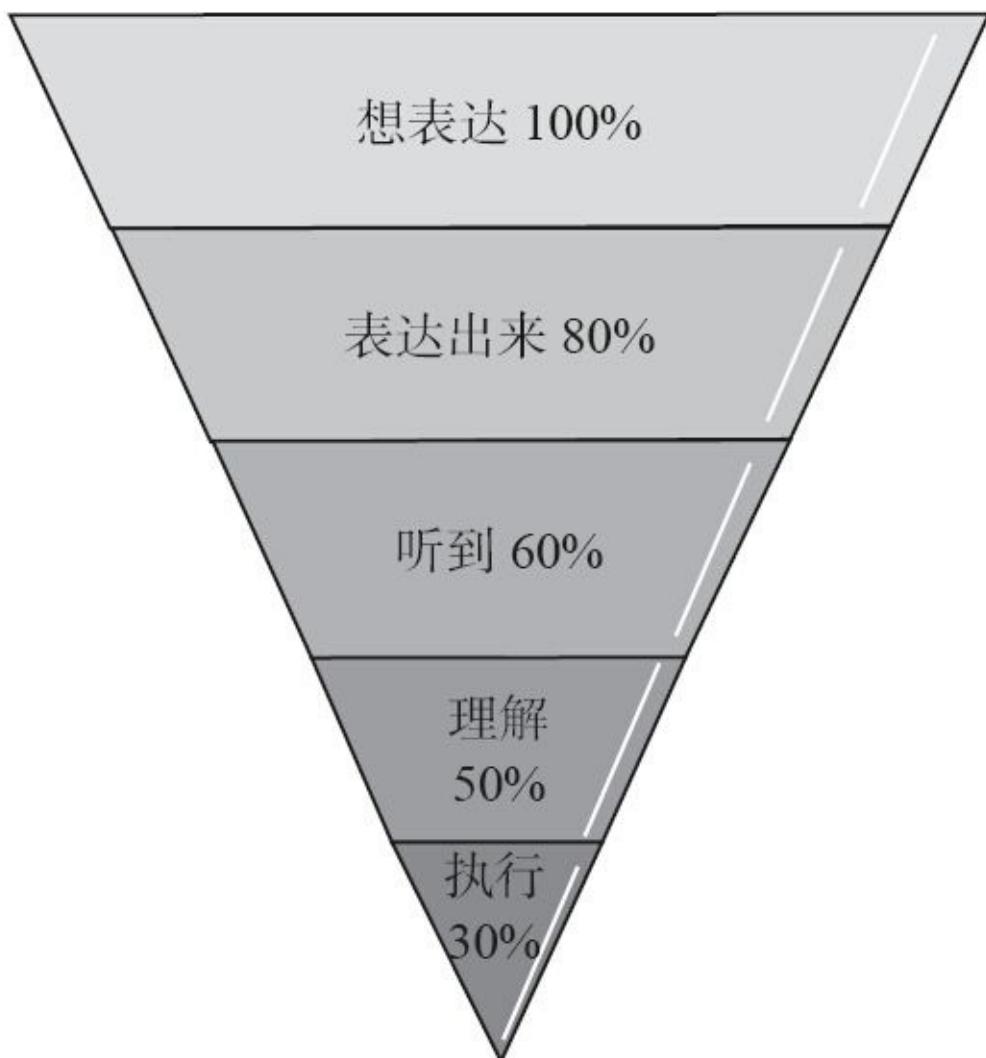


图5-2 沟通中的信息损耗

在工作中，我遇到的能言善辩、巧令雌黄的人不少。会说是必需的，你做的工作需要汇报，遇到了问题需要解释，与人合作或者达成一致都需要说。而表达过程中所损失的大量信息也是可观的，需要绝对重视。

“听”有许多种，有盲目地言听计从的“听”；有左耳朵进右耳朵出的“听”，也有抛去表面、听到深层思想的“听”。你的“听”到底是哪一种呢？

如果悉心观察每个人的倾听方式，或者简单回顾你认为有特点的倾听，一定不难发现“听”有这么几种不同的方式：

- 不用心的倾听：忽视式。
- 外表看起来在倾听：假装式。
- 只听自己有兴趣的一部分内容：选择式。
- 专注对方所说的话，并与自己的经验作比较：留意式。

忽视、假装的“倾听”非常让人恼火，这是一种比较低级的错误倾听方式。选择式或者不由自主选择式的倾听多半由两个原因造成：不期望聆听，迫于无奈只能装着应付了事；有意忽视以达到个人目的。这两种倾听都将不遗余力地宣告合作的终止。例如下面两段对话。

·对话一：

同事A：“艾利克斯，我认为今天我们谈判的最终目的是……”

艾利克斯：“都饿了，我们吃饭去吧。”

同事A：（这小子真过分，太不把我当回事了……）

·对话二：

同事A：“艾利克斯，我认为今天我们谈判的最终目的是……你觉得A方案还是B方案更合适？”

艾利克斯：“好，你说的很对。”

同事A：“（这小子又开小差了，和他沟通真累。）我问的是两个方案你觉得哪个更好？A方案是……”

据心理学分析，未经训练的大脑将做一些“主观的预处理”，“强大”的大脑愿意理解那些能够理解的部分，会强迫我们相信那些部分就是正确的。此时，我们把大脑的工作叫做“违规”，我们的行为也会表现出这样几种“违规”状态：

·联想和假设对方希望说出的下半句。我们的大脑将先入为主地认为说话人一定带有强烈的主观态度，愿意自作聪明地揣测对方的用意。例如，当对方反复说一件事，或者我们认为对方在重复简单观点时，大脑会出于好意站出来，期望尽快解决与对方的谈话过程。

·时刻准备着将自己的观点表达出去。我们担心自己失去表达观点的最佳时机，大脑时刻准备着寻找“最佳插话时机”，而不是关注对方传递的信息。因此，只要对方提到了关键字，或者出现一个停顿，就立刻抢过发言权。

最后一种“违规”的倾听方式是“留意式倾听”，在听取意见和新观点的时候，希望对方的发言内容与自己的经验匹配。大脑出现这种违规时，也有那么一点“主观选择理解”的意味，一旦发现经验、立场与对方的立场相左，就希望打断对方。如果对方是一个说话不怎么精炼的人，或者采用“欲扬先抑”的沟通方式，这种打断将会让对方比较痛苦。

我们模拟一次领导和艾利克斯有关团队建设的谈话，来体会一下“留意式倾听”和领导的感受。

领导：“艾利克斯，管理最重要的是人心，一直想着往前冲而忽视了员工的感受，一定会伤到人心……你们的部门压力一直很大，先前，有人已经受不了这种强大的压力而制造了许多不利于团队稳定的言论……”

留意式倾听模式下的回答：“（痛苦，老板又在挑我的问题了，他们怎么说我能管得住吗？防民之口慎于防川啊！）经理，请你放心，我的部门有再大的压力，我都会关注他们的负面情绪，一定让他们知道管住自己的口，不影响团队士气。”

领导：“艾利克斯，我还没有说完，我的本意不是指你的压力大，或者你的员工说了敏感的事情，我是希望给你提个建议……你从团队成长角度多给团队制造一些与他们工作相关又高于工作的机会，设定一些切实可行的远大目标，用成就感来缓释压力后遗症。”

艾利克斯的回答，是因为敏感地捕捉到了领导对自己团队的批评。艾利克斯向来是一个遇到问题就立即采取修正行动的人，所以，艾利克斯在老板还没有完整表达清楚之前，就已经异常激动地要征得老板与自身在“缓解压力解决方案”这一问题上的一致性。艾利克斯在脑中准备打断老板，非常急于表达自己对于处理此事的观点，而忽视了老板表达的真正重点是建议艾利克斯关注个人发展需求。

老板需要反复修正艾利克斯的认知直到他理解自己的真正用意。不管艾利克斯解释或者尝试改变什么，只要老板的谈话被打断，从主动变成被动，就不会有心情和意愿站在艾利克斯的角度为属下着想。通常，这也会导致沟通失败，双方的这种沟通将产生情感间隙。

我们当然不期望这样的倾听方式。最好的倾听是“同理心式倾听”，用心倾听及回应来了解对方的信息、动机和感受。

领导：“艾利克斯，管理最重要的是人心，一直想着往前冲而忽视了员工的感受，一定会伤到人心……”

艾利克斯：“是的，您说的我非常认同！”

领导：“你们的部门压力一直很大，先前，有人已经受不了这种强大的压力而制造了许多不利于团队稳定的言论。”

艾利克斯：“（我感到压力，让我深呼吸，听听老板想要说什么。）最近的例会上，有人抱怨过工作强度的问题，您一定听到了这些，我也正在思考团队稳定的问题。”

领导：“嗯，我就想说说团队稳定的问题，如果是个人能力不足，承受不了压力，证明他并不适合这里，他要走就让他走吧。但是，我们需要让团队看到发展和成长的空间，毕竟我们是一个成长空间很大的公司，我希望你从团队成长角度多给团队制造一些与他们工作相关又高于工作的机会，设定一些切实可行的远大目标，用成就感来缓释压力带来的后果。”

艾利克斯：“好的，经理，您说的远大目标我深有同感，我会回去为我的团队做一个有针对性的成长计划，因为需要有目标，我希望从您这里得到更多的资源，比如，我希望您能多给我两个项目管理、压力管理培训的名额；还有，我期望您能够参加我这周的例会，就我的两个主要工作汇报做些正面评价，以鼓励他们分享在工作中获得的经验，以及肯定他们的成长。”

领导：“好的，这个没问题，我会通知秘书安排课程的事情；另外，你将例会的时间发给秘书，我会参加。”

“同理心式倾听”不仅仅让谈话过程变得和谐，更重要的是双方易于通过谈话产生共鸣和合作。小结一下，我们列举了四种不合格的倾听方式，也举例说明了“同理心式倾听”方式。我引

用IBM王云院士的一句话：“我们将来无论在大企业还是小公司，无论是打工还是创业，成功一定源于协作，而协作源于信任，信任源于沟通，沟通源于倾听！”

5.5.2 “说”的魔法

在敏捷流程中，任何人都需要跟许多人打交道，沟通是否愉悦将直接决定你是否能够有效说服他人或传达信息。而每个人性格不同，要根据对方的性格做不同的沟通策略。

在刚刚进入IBM的前两年，因为公司转型、业务重组，我所在的普适计算大部门从有到无，先是项目一个个移出普适计算，接着团队一个个被分拆到这个或者那个老板名下，这种变动足足持续了两年，最后，普适计算部门的名字终于被抹去了。在这两年中，我因各种理由被调动，先后跟随6个不同的老板，直至最后一个我希望称之为“独裁者”的老板，我选择了义无反顾地离去，第一次主动选择了一个新的老板，加入了一个新部门。

在此，我无意对比哪个老板更好或者更坏，我必须感激所有老板们，如果没有这些经历和与我遭遇的老板们，我不会有今天，也不配拥有这些谈资。换句更有魅力的话，他们每个人都可以说是上帝安排来与我相会，并且教会我一件事或者几件事的“贵人”。但我的成长之路也许是我不曾经选择或无法复制的，我的“苦”有些也是你不必尝试的，所以，分享这些经历是想帮助你少陷入这些不必要的“痛苦”，尽可能地趋利避害，获得“向上空间”，获得“老板赏识”。

在企业中，如果你拥有IBM一样的开放空间，我建议你选择一个好的老板，至少，他会有助于你的事业发展。何为“好”老板，即和我们发生“相容化学反应”的老板，喜欢我们和我们喜欢的老板。如果两个人“化学反应相斥”，那么最终分道扬镳只是时间的问题，理由也只是借口。据调查，大多数员工离职的原因都是因为和主管不合，所以，这个问题不容小觑。

化学反应很奇妙，但是我的分析却很简单，化学反应是否相容，就是指双方的采集信息方式和决策方式是否相同。

感觉-直觉。我们每个人都在不断接受着信息，这是我们跟上外界节拍的必要前提。但不同类型的个体接受信息的方式不同，这便有了感觉型与直觉型之别。首先，面对同样的情景，两者的注意中心不同，依赖的信息通道也不同。感觉型的人关注的是事实本身，注重细节；而直觉型的人注重的是基于事实的含义、关系和结论。感觉型的人信赖五官听到、看到、闻到、感觉到、尝到的实实在在、有形有据的事实和信息；而直觉型的人注重“第六感觉”，注重“弦外之音”，直觉型的人的许多结论在感觉型的人眼里，也许是飘忽的、不实在的。注重细节的结果是感觉型的人擅长记忆大量事实与材料，他们有时候像本“词典”，能清晰地讲出大量的数据、人名、概念乃至定义，常使其他人感到吃惊；而直觉型的人更擅长解释事实，捕捉零星的信息，分析事情的发展趋势。感觉型的人对待任务，习惯于按照规则、手册办事，比如照着手册使用家电，看着地图辨认交通路线；而直觉型的人习惯尝试，跟着感觉走，他不习惯仔细地看完一大本说明书再动手，结果可能比感觉型的人更快地完成了任务，也可能因为失败而重新开始。感觉型的人习惯于固守现实、享受现实，使用已有的技能；直觉型的人更习惯变化、突破现实。简言之，感觉型的人注意“是什么”，实际而仔细；直觉型的人则更关心“可能是什么”。

逻辑-情感。这是从做决策的方式来对比。仅看这个维度的名称，也许你会觉得，逻辑型的人是理性的，而情感型的人是非理性的，事实上并非如此。两类人都有理性思考的成分，但做决定或下结论的主要依据不一样。情感型的人常从自我的价值观念出发，变通地贯彻规章制度，做出一些自己认定是对的决策，比较关注决策可能给他人带来的情绪体验，人情味较浓。逻辑型的人则比较注重依据客观事实的分析，一以贯之、一视同仁地贯彻规章制度，不太习惯根据人情因素变通，哪怕做出的决定并不令人舒服。

因此，这两种维度组合成了4种不同性情，即感觉+逻辑、感觉+情感、直觉+逻辑、直觉+情感。如果我们了解不同风格的特点，懂得和不同风格的人如何沟通，那么我们就更加容易建立“良好的化学反应”。这四种组合方式又可以总结为：

·直觉+情感=理想主义者。

·直觉+逻辑=概念主义者。

·感觉+情感=关爱主义者。

·感觉+逻辑=严谨主义者。

(1) 沟通对象是理想主义者

“理想主义者”感兴趣的是事物的意义、关系和可能性，并基于其个人的价值观念做出决定。这是一类关心个人成长和如何理解他人与自我的人。

“理想主义者”做人的原则是“真实地面对自己”，是精神上最具哲理性的人。“理想主义者”乐于接受新的思想，善于容纳他人。

“理想主义者”好像永远在寻找生存的意义。他们非常崇尚人与人之间和各种关系中的真实和正直，容易将别人理想化。许多“理想主义者”本能地喜欢帮助别人成长和进步。“理想主义者”是很好的传播者，被人们认为是促进积极变化的催化剂。他们天生能够理解别人的情感，关心他们在生活、工作中碰到的人（如同事、病人或客户、雇员）的需要。

对于“理想主义者”而言，一份好的工作应该是对他们个人很有意义的工作，而不是简单的常规工作或只是一种谋生手段。“理想主义者”崇尚和谐，不愿意在一种竞争激烈或四分五裂的环境中发展。他们喜欢民主、能够激励各种层次的人们高度参与的组织，会被那些促进人性价值的组织或那些允许他们帮助别人完成工作的职业所吸引。

如何说服理想主义者呢？

·跟他们解释这件事能够加强彼此之间的关系。

·申明其有助于团队成长和个人发展。

·关注他们的善举。

·向他们展示新的见解和观点。

·表明如果他们接受这件事，人们会喜欢这个结果，而且暗示人们也因此会喜欢他们。

·指出这件事将帮他们找到价值的意义。

·告诉他们这件事是多么有趣。

(2) 沟通对象是概念主义者

“概念主义者”自信、有智慧、富有想象力。他们的原则是“所有的事情都要做到最好”。

他们是最独立的一种人，工作原则性强，标准高，对自己和对别人的要求都很严格。他们不会被别人的冷遇和批评干扰，喜欢以自己的方式做事。

“概念主义者”天生好奇，无论是由于工作本身的需要还是出于长远的考虑，他们很喜欢不断地吸取知识，因此常常能看到同一问题的多个不同方面，习惯于全面地、概括地思考问题和一分为二地看待问题。

他们很善于发现事物的可能性，理解事物的复杂性，喜欢进行逻辑分析，从而对真实或假设的问题构思出解决方案。

“概念主义者”喜欢能够提供自由、变化和需要较高的智力才能完成的工作，喜欢看到自己的想法得到实施，不喜欢简单、重复的工作。

“概念主义者”会对那些他们认为不够能干的人不耐烦，喜欢与很有能力的上司、下属、同事共事。许多“概念主义者”推崇权力，易被有权力的人和地位所吸引。

如何说服概念主义者呢？

- 讨论这件事有研究价值的一面。
- 高亮其理论基础。
- 展示其如何满足价值体系和战略。
- 展示其如何增强竞争力。
- 指出其广泛和深远的可能性。
- 指出其有趣和引人入胜的可能性。
- 证明自己掌握着可靠的信息来源。

(3) 沟通对象是关爱主义者

“关爱主义者”关注五官带给他们的信息，而且相信那些可以测量和证明的东西，同时喜欢面对各种各样的可能性，喜欢自由随意的生活方式，是反应灵敏和自发主动的一类人。

“关爱主义者”在做决定时，把与别人的关系和人放在首要位置，本能地努力寻求与他人更和睦的关系，同时不断寻找使他们能够通过有形的方式帮助他人的机会。

“关爱主义者”与人交往时较为敏感，谦逊而少言、善良、有同情心，喜欢关心他人并提供实际的帮助，他们对朋友忠实友好，有奉献精神，虽然在很多情况下会有很强烈的反应。

“关爱主义者”与人沟通的时候通常非常友好，比较关注当下的需求和过去的经验，关注细节和事实。

如何说服关爱主义者呢？

- 表明这件事带给人们的实际业绩。
- 清晰地展示其如何让他们受益并使他们关心的事情有益。
- 向他们展示其立竿见影的效果。
- 将判断设置在个人角度上。
- 在讲座中让他们感受到尊重。
- 明确声明他们将得到的好处的细节，不要只是暗示收益。

(4) 沟通对象是严谨主义者

“严谨主义者”非常相信事实，和“关爱主义者”一样，他们关注那些已证实的数据、过去的经验和“五官”所带给他们的信息，喜欢有结构、有条理的世界，喜欢做决定，是一类既现实又有明确目标的人。

“严谨主义者”是坚定、可靠、可信的。他们重视法律、秩序、安全、规则，得体且本分，

被一种为社会服务的动机所驱使。他们尊重权威、等级制度和权力，而且一般具有保守的价值观。他们很有责任感，而且经常努力去做正确的事情，这使他们可以信赖和依靠。

“严谨主义者”需要有归属感，需要服务于别人，需要做正确的事情。他们注重安稳、秩序、合作、前后一致和可靠，而且严肃认真，工作努力。“严谨主义者”在工作中对自己的要求十分严格，而且他们希望别人也是如此。“严谨主义者”喜欢那些与他们一样具有奉献精神、尊重权威和尽自己的本分的同事。

“严谨主义者”沟通时关注细节、务实，组织信息时表现出逻辑、有序，基于因果关系和效果，也关注过去和现在，并且具有严肃、没有废话的风度。

如何说服严谨主义者呢？

- 向他们展示工作原理或流程。
- 指出其如何节约时间和金钱。
- 向他们展示良好的投入产出比。
- 指出度量方法。
- 让他们先试用，然后他们才会考虑买。
- 提供具体的应用和好处。
- 回答他们所有的问题。

对于沟通和说服，最好的方式就是运用同理心。学会用同理心沟通，就需要站在对方立场上思考问题，决定对方思考和判断方式的并非只有“职位、角色”。冥冥之中人的选择倾向、偏重是性格、价值观和成长经历决定的，而性格的影响最为深刻。性格分析以及使用性格学改善团队沟通的管理课程是IBM培养职业经理人的两天封闭培训课程，在我职业生涯的两次管理培训中，这也是最有价值和给我最多帮助的一门课程。后来，许多朋友慕名而来，向我咨询和了解性格学用于管理、团队建设、人与人沟通的话题。也有许多在十字路口徘徊、对生活和自己产生怀疑的朋友请我分析其性格的优势和潜在的风险，找到走出束缚的出路。

在我的信念里，性格没有好与坏，了解自己的性格特征、优势劣势，目的是寻找弗洛伊德所谓最真实的自己——本我，而不是别人所期待的你——自我。人的根本期望和需求源自于本我，而自我是后天成长起来不断去和本我抗争、制衡的。但是，当你在没有压力的情况下，面临选择、判断、思考、描述、研究、行动时会不由自主和不假思索地选择本我而非自我。所以，了解性格的意义在于让每个人方便地了解自己与别人，促进人与人之间的坦诚沟通，而坦诚沟通将决定信任，信任将决定协作，协作最终决定成功。性格分析能够找到满足“本我”的快乐方式，因此为了快乐而工作，也是企业提升效率和个人追求幸福的源动力。

第6章 非凡的测试

第6章 非凡的测试

- 6.1 故事分享：从测试工程师成长为敏捷测试工程师
- 6.2 决策影响力
- 6.3 合理投入
- 6.4 “等待”和“浪费”
- 6.5 自动化敏捷测试
- 6.6 故事分享：Rational中国敏捷测试工程师的一天
- 6.7 选择自动化工具
- 6.8 案例分析：改进自动化工具

为什么敏捷开发或者管理模式要讲测试呢？其一，我从做测试起步，在测试一线工作，从技术骨干到资深人士，我对测试有很深的情结，一直希望能够指导新员工在测试职业路线上好好发展；其二，测试的价值不仅仅是通过尽早暴露产品缺陷以换取研发和运维成本的降低，更是产品质量、品牌形象的把关。据不完全统计，越是重视品牌形象、越是在风口浪尖的企业越倚重测试，而不过多考虑由此会增加成本，或测试投入没有带来期望回报；其三，敏捷方法比传统研发方法的开发周期更短，这使得任何拖延都会将测试推向两难的境地，如何有效测试、何时开始测试、如何面对不断累加的增量型产品的回归测试需求，是近些年来我们听到的国内软件企业的疾呼。所以，我们讲讲测试！

6.1 故事分享：从测试工程师成长为敏捷测试工程师

6.1 故事分享：从测试工程师成长为敏捷测试工程师

从1995年IBM中国开发中心成立到我进入公司的2004年的这段时间，公司主要以接外国成熟产品的测试项目为主，那时候除了少之又少的几个团队在做些边角料的开发工作外，其他团队几乎都在做测试。我们招募了非常优秀的人，却让他们做着不怎么有技术含量的工作，但是，大家都很敬业，每到年终考核大家都攒了一些合作团队的某某开发大牛、某某资深架构师对中国测试团队工作成绩的表彰。新项目一个接一个地在中国落地开花，开发工程师人数急速膨胀，不断有管理职位、高级技术岗位空缺。随着IBM对中国市场和研发的投资兴趣的激增，那时最开心的事情就是去美国出差一两个月做技术交接，不但可以晒晒从奥特莱斯（outlets）买到的打折奢侈品，还能很快迈入更大的发展空间和平台。我们那些年的茶话会的主题通常是到国外出差或是旅行的各种见闻，谁谁谁又要去了，谁谁谁快回来了。而我们的桌上总是不断摆满各国巧克力或者小纪念品，记得有个美女秘书的工位上总有那么多的各国玩偶和各色高级巧克力包装盒，很是令人羡慕。

我感激我的第一任老板，除了把我招进来给了我丰厚的起薪外，还给了我成长的空间，入职一年后便计划送我去美国两个月接手新的工作。我的部门在当时叫做普适计算，其实做的是IBM软件部的“尖峰”项目，我们也是最早把控开发的团队之一，记得我们手上总有各种最新的手机设备，我们使用SMF技术写各种Bundle，并在模拟和真实手机平台上跑。而我们做的产品是公司的雏形项目，通常非常快地通知你要开始，也会非常快地通知你这个项目的生命周期到此为止，或者将要移交给另外一个部门的团队。我在这个部门的日子其实并不算好，不停地换项目，有时候随着项目的迁移还要换老板，有时来不及换老板，就直接借调出去帮忙做项目。

我也是从这个时候起特别崇拜技术好的前辈，特别是我的师傅，从华为跳槽过来的资深程序员，他带着团队大展拳脚，总能搞定别人以为攻克不了的问题。当他走向领奖台领取IBM中国区第一个突出技术贡献奖（OTAA）时，我不禁感动万分。我真心希望认真跟他学习技术，沾染些逢题必解的“好运气”。可是，试用期过后，师傅建议我做测试，原因是他觉得我更适合。

半年后，我不但掌握了产品测试的主要技能，还做了产品图形化测试的自动化测试负责人。因此，我的工作满满当当，经常需要写脚本、写代码。因为对技术的“憧憬”，我抓住一切机会学习新知识、新技能，并乐此不疲，以至于老板几次问我走技术路线还是管理路线的时候，我都执意说：“技术路线！”

那时候，因为开发中心高调扩充实力和规模，管理路线确实发展比较快速。我记得当时真的后悔过，为什么不选择做管理，比起技术路线看似简单、升迁也快得多。而后来我客观地意识到，我若是没有当时的一股子傻劲，做各种苦活累活，怕是不会有今天面对困难的自信。在经历了这么多项目和事情的考验后，在信念深处，我相信任何技术问题没有得到解决只是时间的问题，总有一种针对性的解决方案可以解决它。

那时值得骄傲的一件事就是领导常常表扬我，说我负责的模块他是不用操心的，而工作中比较有成就感的是总可以花费工作时间学习新技术，无论是产品相关的移动平台开发技术、OSGI、SMF即插即用的概念，还是2层或者3层的基于RFT的自动化测试框架等，或者其他几个我现在已想不起名字的新技术。每每我能够使用一种技术成功调通一个“Hello World！”或使用新技术改进了我原有的工作成果时，我那略微自恋的情绪就开始膨胀了。

自从进入公司开始，大小培训不断，而每次能够得到新培训机会会让我变得非常开心。

IBM的培训课都是昂贵的而且宝贵的，不仅课程含金量很高，对于我们这些每天埋头苦干的人来说，这是一个帮助我们“抬头看路”的机会。除了在培训中结识和我一样勤奋但做完全不一样事情的朋友外，我特别喜欢的事情就是和培训课上的特邀嘉宾进行“面对面”的交流。在一次“测试基础”的课堂上，我很有幸结识了我的偶像Karan，他是软件部QSE技术社区负责测试的Lead。通过Karan我认识到了QSE自动化测试社区、测试社区和质量社区的Lead和追随他们的一群优秀的人。

记得初次见到Brent和Karan时，因为他们年纪很大了，也很胖，给我留下的是典型的美国式肥胖症的印象。但是，听着听着，你的思想就会被他们睿智的眼神和富有激情的腔调所贯穿，那么让人敬畏而又羡慕。美国人喜欢活跃的课堂，因为我较为外向，加上英语还不错，所以很快就和Brent、Karan混熟了。课堂上，我总是积极地响应、做游戏或者回答问题。尽管学员有时会出错，但是他们都不会难为学员，相反会让你觉得不论任何观点都是有价值且受尊重的。

如果凑巧说到某个非常有价值的想法，与大师之间的沟通就不仅限于课堂了，他们都会很主动地给你写邮件和你深入探讨。有时候，我们会一起署名文章发表在QSE的社区，就这样，我很荣幸地加入了第一个QSE的社区——测试与自动化，我伴随社区多年，一直积极参与社区的建设。慢慢地，我也就融入了QSE这个圈子，这才发现我们平时努力思考的问题也是很多人努力钻研的，你没有想过的和想不明白的问题，有许多人已经做得很成功，早就有了可以照搬过来的思路和方法。通过在社区里不断积极分享和虚心学习，我渐渐觉得自己的价值在社区里被认可了。之后，我连续发表了多篇有关测试自动化、质量管理、度量以及开发过程的文章。

经过大师们的调教，我工作中的一些关键问题也迎刃而解。我越来越感到，无论是改进方法后的工作效率提升、发现更多的问题，还是原来看似平淡的工作都有了新的非凡意义，都让我兴奋不已。因为每每QSE社区希望大力宣传一种新方法、新技术时，他们都愿意找我这个爱说“实话”和“有观点”的人提供意见；而当我们的想法被充分尊重，我们也会同样尊重其他人。我组织了国内志同道合的同事参与到小范围的社区活动中，我们更为贴近一线的思考方式和实践成果为QSE社区继续添砖加瓦，我也举荐了不少优秀的同事参与到QSE的其他社区活动和团队中，例如开发社区、TDD社区和质量社区。

也许正是因为结识了国内外这些不错的朋友，他们不断给我传递正能量，所以至今我都没有离开IBM。我在这里经历过的磨练和辛苦都没有白白浪费，无论是愉快的还是悲伤的回忆我都会保存下来与人分享，我知道没有过去就没有未来。

虽然我在社区中的影响力和知名度有了显著提升，但我不得不承认这些并没有给我带来丰厚的物质回报，薪水、奖金这回事毕竟还只是取决于你的顶头上司。相反，有时超过别人期望的知名度反而成为我们对立面的放大镜，会更加暴露自己的短板，所以，如果你希望和我一样扩大影响力，请同时保持低调。

上司交给我的工作无论多艰苦、多有难度，我都理解为对我的关照和我必经的磨练。做事和论坛“炒概念”比较起来更需要“务实”，而我也确实静下心来做到了老老实实做人，踏踏实实做事，扎扎实实做工作（必须感谢我的母校西北工业大学对我的教育）。我那时候的状态现在看来仿佛就是怀揣梦想、刻苦用功的一段日子，在心中为自己画了一个大饼，直到很多年后这个饼才得以实现；而在那个青涩年龄，则显得略微“太有想法”、“太刻苦”。

不过我的付出还是有回报的，在IBM国际技术社区里我展露头角，同时，我成为了中国区测试、自动化测试的专家，领导过整个开发中心做自动化测试的研究，后来成为敏捷社区中有IBM认证的Scrum Fellow和社区领导团队中国区代表。

在IBM国际技术社区中遇到的同事，总是在我需要的时候给我带来帮助和希望。在这个社区里遇到的令人兴奋的事情就是能够和许多思想犀利、卓有成绩的人做深入的探讨。这个过程让我体会到，在公司这个以商务为首的大环境下，仍然能够坚持追求真理和懂得顺势而生的大智慧。

很多人历经二三十年甚至更久的时光，几经沉浮，洗尽铅华，最后终成大器；所谓读万卷书不如行万里路，行万里路不如和有智慧的人聊上一聊。我真心感谢我的导师和朋友们，在我最初几年的IBM生涯中点醒我、帮助我，以至于没有因为浮躁和焦虑离开这家公司，在这里我学习智慧而不仅仅是知识和技能。

我在从事测试工作时，有一次，我所在的部门出现了严重的问题，而我真心烦透了频繁的重组，领导不管项目了，但是各种杂事不少，还碰上了不喜欢的“独裁型”老板，而项目却依然一个接一个地忙。我记得名下曾经挂靠过108个实验室设备，我们整个团队每天都在封闭的实验室中，在一堆堆的仪器间穿梭。我给自己充实更多、更重的工作来防止大脑去思考那些烦透的事情，而设备天线发出的哔哔声让心中的那股凄凉不时刺痛着我。

我在我希望那样的日子早些结束，也认识到必须主动求变才是尽头。那一年是我的测试生涯中最难捱的一年，某天我终于因为过重的思虑和一点风寒刺激生病了，一躺下就是两个星期高烧和喉痛不止。当我重新回到团队上岗时，便毫不犹豫地离开了这个团队。

在做管理者之前，技术是我唯一可用、可控的方式，以减少低头做手动测试的时间。而我最纠结的是，无论技术如何改进似乎都达不到100%的自动化，而即使实现了自动化，一些测试环境的异常依然会使测试的可靠性降低。

但是当我经过一番挣扎而重获新生之时，我非常清楚自己不愿意成为经理的窥视下用冰冷的手指拍打键盘的僵尸，也不愿意成为因为某些人推卸责任的一句“看，我们需要更多的测试”而经常加班的员工。我希望成为决策“为何要做”的人。

自此，我开始理解测试的价值体系，我的价值观又发生了一些变化，那时我做了3年测试，几乎已经做过各种类型的测试，经验丰富、技术过关，但现在回想起来，当时的我还只是在搬砖，只是山脚下徘徊的登山者。

一次，社区的Brent老兄推荐了一篇发表于《计算机世界》的文章（James Bach, A Framework for Good Enough Testing, IEEE computer society, 1998），读完这篇文章，我狠狠地拯救了自己一把，为什么我没有早点意识到呢？这篇文章主要以“足够好而且有限的测试”来挑战“无限制测试”价值体系，作者并没有直接给出如何做到“足够好而且有限的测试”，或如何判断我们当前的测试已经到达“足够且刚刚够”，但是却从定义上进行了解释和回答：“足够好而且有限的测试指的是我们能对所测对象提供足够的质量评估，以合理的投入，帮助决策团队做出明智和及时的决策。”

于是，我这几年的历程所形成的对于测试的情感不知不觉地移情给了“自动化测试”。自动化测试不仅仅解放了不断重复劳作的我们，更重要的是，自动化测试能够更好地解决“配置敏感的”、“容易因为人为因素造成返工的”的测试过程，更容易地捕获First Error并提供充分的证据，更有效地减少人工交互。

6.2 决策影响力

6.2 决策影响力

如果我们仔细思考决策影响力，会发现通过合理地测试投入，科学地选择测试方法，快速地反馈和准确地评估产品和项目质量，所提供的信息越准确、越及时，对于管理者决策时的帮助也越大，他们就越有信心做出正确的判断。那么，如何提供准确的质量评估，如何使测试的结果和过程对决策产生深刻影响呢？

测试团队的质量评估报告是最有影响力的。因此，大多数团队都是由资深的工程师或者部门领导来完成测试的质量评估报告。我翻出了多年前还在测试团队工作时所做过的一些测试质量评估报告样本，其中很具代表性的一个报告模板是这样写的。

Regarding to the <name> test status, we can claim the sign-off info:

- How many defects has your area opened against the product core features in <spring/release name>?
- How many defects are currently still open that you would consider stop-ship? (should be zero upon sign off)
- Testing completion status? What percentage of your testing is currently complete? (ex. 100%)
- Has your area given sign-off on the project?

以一个版本为v96Fp7的项目作为参考，翻译后如下。

鉴于目前V96Fp7功能测试Qcert的测试状态，我们可以声明正式退出测试，原因如下：

- 在产品V96Fp7版本的核心组件Engine中发现多少质量缺陷？[xmz] : 0
- 你认为目前存在多少可以停止发布的质量缺陷？[xmz] : 0
- 测试完成的状态？已完成了测试的百分之多少？[xmz] : 100%.100%
- 是否有把握正式声明测试已可安全退出？[xmz] : Yes

印象里，我按照这种报告方式工作过不短的时间，或许当前IBM团队仍然在使用这个模板。IBM的项目实际上希望通过“项目核心组件的零问题+整个项目零严重缺陷”指标来标定质量达标。而在理想情况下，这两个问题的正面答案隐含着一个推断：客户不可能发现产品的核心组件问题或是严重缺陷。但你现在是不是和我一样，开始有些怀疑这个推断的充分性。所以，如果报告的模型稍稍更改成下面的样子，可能会使得我们的测试报告更有说服力。

鉴于目前V96Fp7功能测试Qcert的测试状态，我们可以声明正式退出测试，原因如下：

- 在产品V96Fp7版本的核心组件Engine中发现多少质量缺陷？[xmz] : 0
- 你认为目前存在多少可以停止发布的质量缺陷？[xmz] : 0

·预计多少个遗漏的严重问题将在产品交付后被客户发现？[xmz]：1

·测试完成的状态？已完成了测试的百分之多少？[xmz]：100%.100%

·是否有把握声明测试已可安全退出？[xmz]：Yes

理想情况下，客户会遵循使用说明来使用产品，在产品发布前，使用说明书所涉及的所有例都已经被反复验证过，绝对没有问题。而我们计划的测试范围绝对要大于这个范畴。由此可以推断出，一旦客户发现了我们测试中未发现的缺陷，直接原因是测试代码没有包含对应的部分，或者说用户没有按照我们期望的方式使用产品。随着产品功能的增加和扩展性的加强，按照产品使用说明来使用产品的用户恐怕是凤毛麟角了。我自己在使用IBM产品前也从来不会仔细阅读说明。

这其实不难理解，一行代码很容易做到“零缺陷”，但开发了100或者1000行代码后，就很难避免“引入缺陷”。有人提出，可以从编程人员每百行代码的出错率来估算产品总共含有的潜在缺陷数量，也有人从经验数据出发来推测质量缺陷数量，而我们真的无从考证哪个方法更为精准，这显然是一个百家争鸣的话题。

我们不妨选择一种，就选百行代码出错率的方法来推测吧。如果团队从事开发工作半年或者更长时间后，则不难得到一个经验数据，即团队或者个人的“代码出错率”，假设这个值平均为20%。我们借此可以推测，当他们新编写或者更改100行代码时，会引入20个质量缺陷。

如果我们对产品的测试是充分的（执行全部的测试用例），假设有10万行新增的代码，第一次测试完成应该能够发现 $10万 \times 20\% = 2万$ 行代码出错，开发人员因此要修复所有错误，相当于更新了2万行代码，这又产生了代码变更，则按规律将会有 $2万 \times 20\% = 4000$ 行代码可能引入缺陷。于是，我们决定做第二次测试（回归测试），假设本次测试还是充分的，那么第三次对代码的修改和测试依然存在800行错误。以此类推，第四次测试还有160行错误，第五次测试还有32行错误，第六次测试只有6行错误，第七次测试几近1行错误。

我们取的样本数据足够大，因此需要测试的次数多达7次，才能够真正将“顾虑”从产品中移除出去。而假设每次迭代所更新和影响的代码行数是1000行，则测试执行4次以后就能够让“产品预计缺陷数”趋近于零。

上述测试报告中的第三个问题是：“预计多少个遗漏的严重问题将在产品交付后被客户发现？”从理论上来讲，这个数字可以通过计算给出，先根据开发团队的代码出错率推算出总共多少个缺陷，再看我们计划和已经执行的测试次数，二者相减就可以推测出将遗漏多少问题，如果按照我们发现的严重缺陷占所有缺陷的比例，就可以推测出究竟会有“多少个遗漏的严重问题将在产品交付后被客户发现”。

例如，在V96Fp7的测试中我们取开发团队的平均出错率为20%，本次开发团队一共新增和更新的代码行数是10万，经过3轮测试我们发现质量缺陷数为790个，此时就可以得到，如果此刻交付产品，我们预计有 $800-790=10$ 行潜在的代码行错误，而假设每行错误都折射新的质量缺陷，那么还有10个错误将留给客户，而已经发现的790个缺陷中有79个缺陷是严重缺陷，那么我们推算还有1个严重缺陷未被我们识别，这很可能被客户发现。

基于“平均出错率”的估计的准确度主要依赖于经验值，通过我们的观察，在较为频繁的迭代和有压力的环境下，经验值往往和预期有很大出入。有的迭代表现得很低，而有时却偏高。采用新技术时，又会增加引入质量缺陷的风险；而在开发人员工作于较为熟悉的模块时，“代码出错率”很低。

这样看来，开发人员的心理状态也对质量缺陷的数量有相当的影响，如果开发团队的氛围比较轻松，出错率就相对低，反之就会高；而如果开发团队非常紧张出错率所带来的不良后果，这种条件下出错率仍旧很低。但介于这两者之间的心理状态往往产生相对高的“代码出错率”。

随着开发经验的累积和自查（单元测试、代码静态扫描）工具的运用，出错率会得到控制。其实，预测遗留问题的最好方法因人而异。例如，你的习惯是善于从宏观入手，对于抽象概念的理解好过对具体规则和实际情况的总结，则不妨凭借直觉给出预测；而那些性格很严谨、客观，喜欢关注细节和寻找问题根源的人，不妨试试以上的精确计算方法。

那么，目前的报告充分了吗？正如我所担心的，一些人质疑“足够好而且有限的测试”，却相信“我们需要做各种类型的测试，直到找到所有缺陷”。但是当我在过去的某个时段猛然意识到这种想法的严重性时，我已经没有更好的词汇来形容思想的解放和价值体系的重新构建带给我的欢欣雀跃了。

通过测试验收报告，我们告知管理层测试进度是100%尝试，并未发现重要的问题，也给出了可能流向客户端的错误数量的预期，好吧，我们长嘘一口气，报告终于交付了，报告提供了准确的质量评估且宣告评估过程完成，我们应该休息休息了……

但是，请等一等，还不能就这么划上句号，虽然我们的报告已经很好，但是报告真的给出了完备和正确的信息以做出产品发布的决策吗？我们有没有误导决策者呢？

下面要关注的问题出在这个“100%尝试”上。在测试执行的计划阶段、甚至更早的策略规划阶段，测试工程师会依据测试目标、测试环境、资源决定哪些测、哪些不测，哪些是有时间、有资源再测等策略。通常，这个过程在测试执行前就要得到批准，在敏捷开发中，测试计划和规划放在团队的产品清单中，测试工程师还要组织团队审阅测试规划和测试用例，以得到大家对评估目标的一致认同。

测试验收报告中对测试覆盖范围的补充说明并不是必须部分，但是作为专业的测试工程师，我们相信只有这样才可以合情合理地说明测试报告中“100%的完全尝试”所代表的意义，它涵盖了我们对承诺工作的严格履行，也显示了我们严谨和负责的工作态度。

鉴于目前V96Fp7功能测试Qcert的测试状态，我们可以声明正式退出测试，原因如下：

- 在产品V96Fp7版本的核心组件Engine中发现多少质量缺陷？[xmz] : 0
- 你认为目前存在多少可以停止发布的质量缺陷？[xmz] : 0
- 预计多少个遗漏的严重问题将在产品交付后被客户发现？[xmz] : 1
- 测试完成的状态？已完成了测试的百分之多少？[xmz] : 100%.100%
- 是否有把握声明测试已可安全退出？[xmz] : Yes.

补充测试点覆盖范围报告：

· 平台覆盖（如图6-1所示）。

	W2003EE	W2003SE	W VISTA	W XP	SUSE10	RHEL5	AD5.3										
Ex. FVT Functional Scenarios	EM64	IA32	AMD64	EM64	IA32	AMD64	EM64	IA32	AMD64	EM64	IA32	AMD64	EM64	IA32	AMD64	PPC31	PPC64
IE6	✓	M	H	H	✓	H	L	L	L	L	✓	M	H	H	M	✓	X
IE7	✓	M	H	H	M	H	L	L	L	L	✓	✓	M	H	H	✓	X
Firefox 2.0	✓	M	H	H	M	H	L	L	L	L	✓	M	H	H	M	✓	X

H High priority 高优先级

M Medium priority 中等优先级

L Low priority 低优先级

X Supported out-of-box 开箱即已经支持

n/a Not available 不适用

✓ The path is selected for testing 测试路径被选定

图6-1 平台覆盖测试计划

- 功能点覆盖（如图6-2所示）。
- 测试领域覆盖（如图6-3所示）。

Coverage		OS/Database/Webserver								Number of Scenarios
		W2003 EE/EM 64T/D B2LU	W2003 SE/IA3 2/DB2 LUW	Vista /AMD6 4/ DB2LU	W XP EM64T /DB2L UW	SUSE 10/EM 64T/D B2LU	Redhat 5/AMD 64/DB 2LUW	AIX5.3 PPC64 /DB2L UW		
Tooling	Startup Page	✓				✓		✓	✓	1
	RAS PD&R	✓				✓		✓	✓	1
	Conn. Mag	✓			✓	✓		✓	✓	2
	Welcome Page	✓				✓		✓	✓	1
Monitor	Monitor	✓	✓		✓	✓	✓	✓	✓	9
PD&R	PD&R	✓	✓			✓	✓	✓	✓	6
E2E	E2E	✓				✓	✓	✓	✓	4

Not tested 未测试
✓ Tested 已测试

图6-2 功能点覆盖测试计划

Functional Scenario Test					Security Test	Globalization Verification Test	Accessibility Test
	Tools and Infrastructure	Monitoring Components	Problem Determination & Resolution	End to End Functional Scenarios			
Correctness	✓	✓	✓	✓	✓	✓	
usability	✓	✓	✓				
Reliability		✓		✓			
Portability	✓	✓	✓	✓		✓	
Maintainability		✓	✓	✓			
Accessibility							✓

Not Applicable 不适用
✓ Tested 已测试

图6-3 领域覆盖测试计划

至此，这个测试退出报告就非常棒了。不但提供了准确的质量评估，而且提供了质量评估的证据。管理团队应该可以根据这个报告做出选择，我相信测试工程师的专业性和工作态度也将得到认可。

6.3 合理投入

6.3 合理投入

每当我被迫从事繁琐而没有意义的执行工作，或者偏执、无休止地做回归测试时，我总希望这种日子尽快结束。而工作必须进行，我只能选择靠双手改变这种工作方式，或是提升一些满足感。

当我整理团队的工作流程时，我发现我们竟然没有认真考虑过测试投入的合理性。改正测试发现的问题并没有直接增加产品的价值，而我们却时常给领导施加压力要求给测试部门投入更多资源。“测试无法穷尽所有的缺陷”是一句推诿责任的借口还是公理？如何规划测试人员的职业道路，激发其对工作的热情和认同感？自动化测试降低了人工干预，但是自动化测试的投入本身成为成本，如何核算并规划具有合理投入产出比的自动化测试？带着这些问题，我曾反复思考，过去许多年的实践经验或许可以给出答案。

6.4 “等待”和“浪费”

6.4 “等待”和“浪费”

当我面临手指头数不清的质量缺陷，而计划退出测试的时间所剩无几时，我会毫不犹豫地告诉团队，请利用你们仅剩下的所有时间尽可能地多做测试，这样做有些隐蔽地推卸责任，但我们确实对产品没有信心。我的团队非常配合，然而这种穷尽所有时间来测试的状态一直没有好转，甚至激增了测试团队的焦虑情绪，甚至弥漫开一些抱怨和责难。我开始认真思考这个问题，问题到底出在哪里？我们为什么始终不能从容应对产品的质量评估工作？

首先，一定是“浪费”。我们没有在最好的时机进入测试，因此浪费了最佳排除问题的时机，直到最后留下一大堆复杂问题的时候，才发现问题多得没有时间来排查也更没时间修复。

其次，就是“等待”。工作流程中，不仅仅是测试相关的流程，一定有一些不能弹性变化的刻板固执或者必须上级领导批准的流程让我们“一等再等”。为什么要屈从于实则不合理的流程、浪费大好光阴呢？我们一定要找到问题，将其根除。有了这个想法后，我们开始看到一些并不容易发现的、但存在已久的问题。

1) 活动的价值部分从发现问题开始，直至解决问题，而往往我们实际所支配的时间远远超出了这段价值活动合理时间十几倍，包括消息的传递至消息被阅读的时间，问题被认领的时间，以及因为繁琐的流程所耗费的时间。例如，当我们从客户那里获取需求后，需求工程师要将其转化成需求文件并写入需求文档管理工具的数据库，然而需求工程师正在休假；又例如，当需求工程师通知团队，需求已经进入数据库，需要寻找一个大家合适的时间来审阅需求，但因为重要的决策者或者审阅参与者有更重要的会议和工作要处理，这个会议邀请被安排在两天后；再如，因为需求不详细，前期测试工程师和开发工程师对需求的理解有很大出入，因此在开发工程师交付工作和测试工程师开始测试时，测试工程师发现准备好的测试环境已经不适用于新代码、新功能的测试，测试环境必须重新处理，而这个过程需要IT部门花费8小时时间进行软件更新和各种参数的重新配置。

2) 活动的价值部分被重写，之前付出再多再有价值的工作，此时都毫无意义，而我们已经追悔莫及。例如，因为测试后期发现了极其严重的问题，产品需要重新做大改动，这个改动不得不推延了整个发布时间；或是需求工程师因为种种原因不得不在开发已经开始了一两个星期后推翻之前的理解和用例设计，且团队此时无论是接受新设计还是拒绝配合，这个活动价值部分已经不复存在，之前做的工作均没了意义。

3) 活动开展前忽视了风险的存在，或者盲目应对潜在的风险，因而，风险一旦触发，就像太平洋上空的蝴蝶效应，倾刻引起大西洋海面海啸。当产品的功能很大程度上基于第三方软件的能力时，而这个第三方软件并不像提供方所承诺的那样稳定和完整，我们的工程师不得不陷入紧张的焦灼期，产品所有的工作不能按时退出，直到这个第三方产品的问题得到解决方有所改善。有时对产品大动干戈地替换整个类，测试工程师不得不因此重新审视测试计划，并做较大范围的回归测试，以确保先前的功能一切正常。

4) 除了这些外因，我们也发现由于自身局限性造成了不良后果。对于“测试可进入条件”的过分依赖，对责任的过度负责，使我们成为了流程改进和提升活动价值的阻挠者。例如，我们害怕过早进入测试，在没有完整代码发布时，我们挑剔地选择了忍耐或等候更好、更稳定的版本。

综上所述，测试的投入或者团队任何人的投入都需要考虑如何利用现有可控资源迅速抵达目的地的问题，而不是如何更快地完成计划的问题。所以，为了实现合理的投入，我们（从测试角度出发）需要建立起大局观，了解团队的最终目标、完整的开发流程以及阶段性

的成就和目标，了解可能存在的风险，包括来自天灾或者人祸的风险，并且一定要做好计划B，以备不时之需。

最后，我建议测试团队做出如下改进：

- 1) 拒绝“等待”，如果能够提前开始工作，可以模糊测试和开发的工作界限，甚至参与到更早的需求分析中以直接发现和评估产品的质量。
- 2) “静态测试”与前期需求的分析和拆分细化非常重要，不为不完备的需求和缺失设计部分做出假设，并延期我们的“承诺”。
- 3) 尽管不用读懂开发工程师的每行代码，但需要和开发工程师充分沟通接口技术，理解代码集合、函数具有的意义和业务逻辑中所承载的工作的定义和发生顺序，以知道如何“捷径”测试。
- 4) 尽早尝试未执行过的测试用例，一经发现测试过程受阻，及时报告问题，自动化“关键路径”测试的做法一直以来都值得称赞。

6.4.1 仅仅足够好

合理的投入，需要有严谨和科学的测试方法，以及“仅仅足够好”的态度。好的测试路径设计更重要，最佳路径好比在扫雷游戏中用最少的点击数找到真的地雷一样，测试需要仔细选择尽可能不重复路径的测试方式。例如，一般的测试环节，我们将“登录组件”和“业务组件”分开设计测试用例；根据每个组件的可输入数据特点，用“边界值数据、非法数据、常用数据、最大数据和最小数据”等方法设计测试方法；而测试“业务组件”又必须经历“登录组件”，因此如果将两个组件有机结合起来就会发现，一半的测试用例就可以完备地覆盖两个组件的测试了。

又如，测试平台操作系统、语言版本、后台数据库平台的组合选择本身就是一个特别浪费和重复的过程，不可能在穷尽一种组合环境的测试后，将n倍放大的测试工作量分配给团队来满足对多个平台、语言、后台数据库兼容性的验证，而采用一般性平台的验证方法可较为灵活。

- 1) 交给平台测试团队，或者自动化测试团队，将关键测试用例自动化运行起来，使得发布多版本后基本可以保障产品的环境兼容性和稳定性测试需求。
- 2) 如果有条件可以将功能测试点随机或者按照环境权重比例分布到所有测试平台和环境上，像“梯田”一样，对一般性的测试用例不需要重复地分割到各个平台和环境中，对于重点和关键路径，测试用例可以重复分配。

在此基础之上，可以优选一两个重要的组合方式，例如，尽量选择与客户环境接近的环境组合方式，在IBM的行业术语中，我们把这种系统的拓扑结构叫做“Golden Topology”，以实现较为完备的测试，测试工程师不用频繁切换工作环境因而测试效率提高。同时，在一些并不重要的平台上可以做“技术忽略”，也就是放弃测试。再有，就是针对特殊需求，例如用户流量逐步增加但目前仍然小众的平台和系统，我们可以只做关键路径测试，或者更小的冒烟测试。

6.4.2 理性规划

合理的投入，需要谨遵可持续性发展的战略来竭尽资源达到工作目的。在过往的多个项目中，我们因为不能合理计划工作量，错误估计或者少估计工作量，太过乐观地评估产品的质量，致使测试工程师不得不避重就轻，选择更多的容易执行而不易发现错误的用例。在根本完成不了数量指标的前提下，测试工程师会尽可能少地尝试那些容易发现错误的测试方法、路径。通常容易过于乐观估计的工作量的环节如下。

1) 测试准备工作，环境的搭建。原以为搭建好一个环境就可以了，但是往往会出现一些非常奇怪的问题，只有这个环境可以再现，而仅仅是重装或者换一个环境就可以解决。我们很少能够从这种错误中找到有价值的答案，因此最好在计划测试准备工作时，增加一些重建测试环境和修复测试环境错误的时间。

2) 测试用例的执行时间。一般我们会使用平均时间乘以测试用例的数量来规划我们的测试执行时间，而需要指出的是，发现一个问题时，不是执行一遍而是执行至少三遍，因为第一遍发现错误后需要跟踪和研究，势必再至少执行一遍，当问题得到解决以后，又得执行一遍来验证这个错误。所以，往往因为发现缺陷数量超出预期使得测试用例的执行时间远远超出预期。

3) 被阻断的测试时间。我们虽然建立了良好的沟通渠道、及时的反馈机制，每当发现问题，一定第一时间报告，但是一旦“满身虱子就不再怕咬了”，许多问题长期得不到解决或者不能完整地解决，因此在这些问题发生后，而又没有解决的前提下，我们继续测试相关功能点就失去了意义，因此，我们选择停止这个角度的深入测试工作，测试因此被阻断。如果出现了最关键和最低层的错误，那么整个测试团队都可能被阻断工作，其后果将给后期的测试带来非常大的压力。

4) 报告、开会、被打扰的时间。很少有人去计算我们每天花在报告、计划、其他文书上的时间以及开会的时间，我们总喜欢乐观地估算自己每天的工作量是饱满的，没有打扰（也经常不愿意被打扰）。很久以来，我们一直在处理又紧急又重要的事情，偶尔不得不被迫参加一些领导安排的会议，接着继续追赶进度，永远有一大堆事情拖着、等着，如果是这样，那么一定是我们过多地承诺了工作和背离了“可持续化发展”原则。我发现合作意识越好、越在乎别人的情感和想法，以至于总是给予别人额外的关心和帮助的人，越容易过高承诺工作量。

6.5 自动化敏捷测试

6.5 自动化敏捷测试

虽然已经多次提到了IBM的自动化测试社区、专家和各种工具，我们还是看看自动化测试的发迹史，这对许多公司都有借鉴意义。

或许是对代码工作的热恋、对点击鼠标无技术含量的担忧，或许是大量测试工作的累积使我们不得不考虑效率的提升，甚至是片面、客观而非自主地选择了自动化测试这份工作，大量的工程师入职不久后就开始了自动化测试之旅。

2008年之前，IBM测试团队的人数占据开发中心总人数的60%，我们每天为开发中心数以百计的产品进行测试。我测试过的产品大概可以分成三类：一是面向终端用户的消费产品，对于这类产品的测试，我的个人经验是更需要在用户体验、用户并发访问、系统的安全和健壮上多多用心寻找缺陷；二是面向企业的中间件产品，中间件产品的测试更需要关注在大数据量、压力、长时间运转下系统的稳定，以及在异常恢复时数据和状态的健康，因此重点是对API的测试；三是面向解决方案的测试，这一类包括了前两类，除了在上述重点上多用功，好的测试需要尽量真实地再现客户环境。这三个方面的测试都需要大量的用户操作、输入、监视输出，自动、半自动地根据输入和输出产生系统的下一步动作，这些系统中每个对象的变化状态、过程的演示以及数据流的准确都是需要关注的重点。

现在难点来了，很难依靠肉眼和手指的跳动实现完备性的测试。例如，一些数据的中间状态不借助工具就不能展示出来，就无法知道数据从发生到结束是否出现了异常。

那么，全自动化的测试是否能够解决所有测试问题呢？也不可能，因为没有100%的把握证明自动化测试工具和测试方法本身不带来问题，此外还需要考虑自动化测试脚本的开发所需的代价。很有可能在我们的测试脚本还没有完成的情况下，产品的研发已经不得不结束了。

下面，针对自动化测试落足点的选择、自动化测试的完备性、自动化测试工具的选择以及自动化测试开发额外开销的时间对测试进度和测试成本的影响等方面分享我在IBM的自动化测试经历与我对自动化测试的认识。

6.5.1 自动化的落足点

我初次接触自动化测试时还在普适计算部门工作，部门的项目非常多、工作也很繁重，因为更多的测试工作都是针对新功能、新模块开展，自动化测试的工作并没有像后来在测试中心做自动化测试时那么备受重视。

测试工程师每天遇到的都是新问题，许多测试需要“跑”。自动化测试为什么要做、什么时候开始我已经忘了，而自动化测试的项目我依然记得，那是一个最新的Nokia手机平台上的自动化办公软件的测试，因此自动化测试主要是针对GUI界面展开，自动化测试的成效很差，以至于我印象里从来没有有效的问题是通过自动化测试发现的，而是在自动化测试脚本编写过程中发现的。但是大家还是“被”要求完成一定程度的自动化测试，为此我也算是舍本逐末地学习了如何使用Rational Functional Tester这个有趣的工具。

2008年我加入了测试中心，主导的工作就是如何改造原有测试模式，利用较少的资源完成高效的测试。其实，解决方案很简单，就是将那些可以重复的、标准化测试过程独立出去，交付给专业的独立测试团队来做测试。

按理说，任何相似的产品和项目都可以使用同样的测试环境、测试标准和测试策略，但实际工作却没有那么简单。可配置参数的自动化测试脚本开发、运行时负载均衡等多个问题

成了时下的焦点。而测试本身占去了团队大量的时间，对自动化测试框架的改良只得见缝插针，我和团队将改良工作划分成一步步细粒度的工作，虽然分身无术，但是这种做法还是可以逐步推进工作的。

我记忆里做得最多的工作就是不断提高自动化测试执行的首次执行成功率，提高可执行脚本的测试点覆盖率，提高测试平台和工具的鲁棒性，减轻环境搭建、配置工作等。在这一年里我和测试团队的同事也同时发表了许多有关测试能力、自动化测试、如何写好测试脚本、如何有选择性地实现自动化等多篇文章，并了解了诸多出色的测试规划、执行、管理工具，例如Rational Build Forge、XDE、Manual Tester、Rational Performance Tester、Rational Quality Manager、Rational Asset Manager等。

在自动化测试框架的逐步改进过程中，除了能够解放出部分人手外，我们的测试框架能够完成更多的测试点，并且定制了一套开放的标准，产品团队按照这个标准就可以提供新的自动化测试脚本并在这个平台上运行。而公司又提出了更有挑战的目标——希望执行周期缩短一半……面对新的要求，我们希望团队利用各种资源培养自动化测试的能力和技能。下列是我们采用的一些做法：

- 最大化地从IBM社区、成熟测试团队那里学习全面的测试方法、自动化测试工具。
- 选择容易操作、容易编写测试脚本的工具来建立尽量统一的测试平台，并极大地鼓励测试团队之间的测试资源共享和测试资产的重用。
- 最大化地使用本公司的工具和平台，在开发过程中获得官方的帮助和专业的技术支持，同时制造了跨团队合作和学习的机会（而这是IBM个人发展的重要维度）。

我们一边不断提高测试自动化率，但同时意识到我们的工作绝不能100%由机器代工，设计一套能够自动化所有测试用例的平台并非没有技术可能，而是开发所耗费的时间远远超出我们的期望，这种工作到了后期让我们觉得自己像是在努力使用计算机取代人类的智慧那么愚蠢，更不用说自动化测试做测试规划、测试缺陷这样比较少涉足的领域有多少艰难险阻。

测试无法取代完全的手动测试、黑盒测试，所以我们需要自动化测试更多地集中在最有价值的部分，因为所有的自动化测试都不是免费的午餐，无论自己开发还是由其他团队开发，这部分的开发工作量都将划入“成本计算”。因为我们是专业的测试团队，而不是自动化测试工具开发团队，所以我们从来没有想过为了自动化而做自动化测试。我们认为最佳的自动化测试策略是做重复度高、关键的测试，包括：

- 重复的测试用例。在可见的、将来最多被测试和执行测试的部分需要做更大比例的自动化，之后每执行一遍测试，就会从自动化中多一分收益。
- 敏感、复杂的配置和安装过程。我们之所以需要这部分的自动化是因为，电脑好过人脑，在重复、复杂的配置操作中，如果做错一步都可能带来极大的损失，因此这部分的自动化最重要。
- 测试工程师认定最没有价值的部分。当我们觉得手上的工作无趣、疲劳、简直是降低个人价值时，考虑自动化这些工作会给团队带来生气和活力。

并且，我们认为自动化测试的工具应该具备以下特征：

- 自动化测试产品需要更“容易”使用。
- 自动化测试需要更“适用”于“你”的环境。
- 自动化测试产品需要规范编写、系统设计并且尽可能保留可扩展的能力。
- 自动化测试产品的单元、任务最好容易被获得。

而为了做好自动化测试的工作，我们还需要培养工程师的如下技能：

- 需要良好的软件工程理论和方法指导我们的自动化测试开发工作，例如设计、代码审阅、开发、测试等。
- 自动化测试产品需要有清晰的文档、注释，能够系统地组织这些说明和文档，并用版本控制工具管理这些资源。
- 懂得自动化测试领域“分享”和“重用”的价值，通过向社区发布最佳实践经验，有能力和渠道向咨询领域专家请教相关问题和寻找解决方案。

6.5.2 ROI指标判断

维基百科对ROI的解释为，在金融学中，ROI被理解为投入产出比（Return on Investment），指投融资的回报率或者投资回报，表示相对于投资额的金钱获得或者损失的比例。金钱的获得或者损失可以理解为利益、利润或者损失、浪费。投资额可以理解为资产、本金。

ROI的最简单的计算公式是

$$r_{\text{arith}} = \frac{V_f - V_i}{V_i}$$

其中 r_{arith} 是产出、收益， V_i 是初始投资， V_f 是最终价值。

测试的价值是通过找到更多质量缺陷（Defect）使得产品后期投入（Cost）减少而实现的。因此，自动化测试和手动测试一样， V_f 最终的价值取决于暴露质量缺陷的数量。

6.5.3 自动化测试的ROI

自动化测试的 $autoV_f$ 取决于最终发现的质量缺陷数量，而自动化测试的初始投资 $autoV_i$ 由自动化工具开发、部署，自动化脚本、程序的开发和维护，以及执行自动化测试所需的成本决定。

因此， $autoV_f - autoV_i$ 越大， $autoV_i$ 越小，ROI比例将越大，也就证明自动化测试越成功。

而为了证明自动化测试的优势，我们喜欢将它和手动测试的ROI进行对比，得到公式1，从而证明自动化测试比手动测试更有效。

$$\frac{autoV_f - autoV_i}{autoV_i} > \frac{manuV_f - manuV_i}{manuV_i} \quad (\text{公式 1})$$

如果无论自动化测试和手动测试达到的最终价值都是等量的质量缺陷的暴露，也就是

$$autoV_f = manuV_f = V_f \quad (\text{公式 2})$$

可以推得

$$\frac{V_f - autoV_i}{autoV_i} > \frac{V_f - manuV_i}{manuV_i} \quad (\text{公式 3})$$

进而推得

$$\text{manu}V_i - \text{auto}V_i > 0 \quad (\text{公式 4})$$

经过以上推算，我们可以得到结论，如果要使得自动化测试优于手动测试的投入产出比，需要证明同样质量的测试结果前提下，自动化测试的投入成本要低于手动测试的投入成本。

为了进一步实现自动化规划的最优化，我们来分析一次成功的测试案例，首先需要一些假设和前提。

假设1：手动测试用例/自动化测试脚本的粒度是一个用例/脚本足以且仅能发现一个质量缺陷。

假设2：自动化测试不发现质量缺陷，只能发现日志错误，在人工检查日志错误时，需经过分析日志（约等同于一遍手动测试）、人工再现场景以确认产品问题（等同于一遍手动测试），从而定位和暴露真实质量缺陷。

假设3：无论是否采用自动化手段，在当前质量缺陷被修复后，仍然需要花费一遍手动执行测试用例的同等执行时间来验证此修复。

假设4：自动化过的脚本仅执行所花费的时间和投入可以忽略。

如果用100个测试用例来测试一个隐藏了20个质量缺陷的产品，而且每个测试用例的执行时间约为a小时。

当这100个测试用例完全手动测试时，我们最少的测试时间是80a（80个可以完全通过的测试用例的执行时间），加上20个平均3a小时的处理时间，才能完全暴露质量问题（参考假设2, 3）并最终完成测试。因此

$$\text{manu}V_i = (80 + 20 \times 3) \times a = 140a \text{ (小时)}$$

而经过这次测试，可以得到20个质量点的提升 $\text{manu}V_f = 20\text{Defects}$ 。

为达到同样的测试效果，我们仍然对同样的产品采用自动化测试手段。那么基于前提假设，最优的自动化测试比例，也就是将80个可以完全通过的测试用例都生成自动化测试脚本，那么本次测试的 $\text{auto}V_i = (x + 20 \times 3 \times a)$ ，其中，x为开发和维护80个自动化测试脚本的代价。同样， $\text{auto}V_f = 20\text{Defects}$ 。

根据公式4得到的结论，最优自动化规划将满足以下不等式

$$x < 80a$$

结论：如果自动化后的测试脚本在项目中只运行一遍，那么自动化脚本本身的开发成本不应大于手动执行一遍对应测试用例的执行时间，换言之，完全没必要自动化。

6.5.4 合理的投入

真实的场景将比以上案例更复杂，自动化测试脚本可能执行多次，产品中的质量缺陷无法确切得知。为了进一步得到更合理的场景下自动化测试规划的建设性分析，我们需要声明几个变量。

声明1：d%为平均出错率，即一个产品的代码中如果有d%的出错率，对产品质量缺陷的修复仍将带来d%的新质量缺陷。

声明2：质量达到95%的测试通过率后，产品可以发布。

声明3：n为迭代次数，自动化脚本将被执行n次，或者叫做迭代测试n次（n≥1）。

声明4：TotalEf为完成一次完整的手动测试时，测试用例完全通过的执行成本。

声明5：x为自动化的开发和维护成本的上限， $x=manuVi-autoVi$ ，也就是说实际的脚本开发维护成本应该大大低于x，这样的自动化测试ROI才是合理的。

利用公式4推导出来的结果 $manuVi-autoVi>0$ ，可推得

$$manuV_i = TotalEf (1 + 2 \times d \%)$$

$$autoV_i = TotalEf (3 \times d \%) + x$$

因此推导出

$$x = TotalEf (1 - d\%) \quad (公式 5)$$

结论：d%的值越大，质量缺陷越多，我们依赖于手动测试的可靠性更大。

因为新测试用例开发出来很快投入了FVT（功能检验测试），之后测试人员才投入开发自动化测试，所以，此阶段我们必然依赖于手动测试。只有在以后的回归测试阶段，两种测试手段，即自动化测试和回归测试才具备可比性。而真正的自动化收益也来自于测试的回归测试阶段。同时，以上公式表明，TotalEf越大，我们越能够倚重于自动化测试。回归测试的用例重复性使用频率越高，自动化的投入必要性越大。下面就以回归测试作为背景来分析自动化测试的合理规划。

假设一个团队回归重复测试相同用例的出发点是为了确保先前成功的功能不会失效，那么成熟的测试团队应该不断根据现有的产品质量缺陷报告，对回归测试的覆盖范围和计划做出正确估计和调整。在开发团队修复代码的过程中依据d%错误率的平均工作有效性来估计，我们对每个迭代的质量缺陷数量将是上一迭代数量的d%。那么，n次迭代将总共产生的质量缺陷数量为

$$d + d^2 + d^3 + \dots + d^n = \frac{d(d^n - 1)}{d - 1} \quad (公式 6)$$

结论：回归测试产生的质量缺陷是产品出错率的等比数列之和。

6.5.5 自动化回归测试

同理，科学规划的回归测试的覆盖范围也是等比递减的。例如，第一次迭代所花费的成本为 $TotalEf \times (1+2 \times d\%)$ ，第二次迭代所花费的成本为 $TotalEf \times [1+2 \times (d\%)^2]$ ，第三次迭代所花费的成本为 $TotalEf \times [1+2 \times (d\%)^3]$ ，以此继续推导，可以得到第n次迭代总共花费手动测试成本为

$$manuV_i = n \times TotalEf + TotalEf \times d \% [(d \%)^n - 1] / (d \% - 1) \quad (公式 7)$$

而引入自动化测试后，第一次迭代所花费的自动化测试执行分析成本为 $TotalEf \times (3 \times d\%)$ ，第二次迭代所花费的成本为 $TotalEf \times 3 \times (d\%)^2$ ，第三次迭代所花费的成本为 $TotalEf \times 3 \times (d\%)^3$ ，所以n次迭代后总花费自动化测试执行分析成本为

$$3 \times \text{TotalEf} \times d \% (d \%^n - 1) / (d \% - 1) \quad (\text{公式 8})$$

因此，根据公式7和公式8以及声明5，我们得到自动化开发和维护的成本上限x为

$$x = n \times \text{TotalEf} - 2 \times \text{TotalEf} \times d \% (d \%^n - 1) / (d \% - 1) \quad (\text{公式 9})$$

图6-4为公式9所呈现出来的各单元之间的关系。

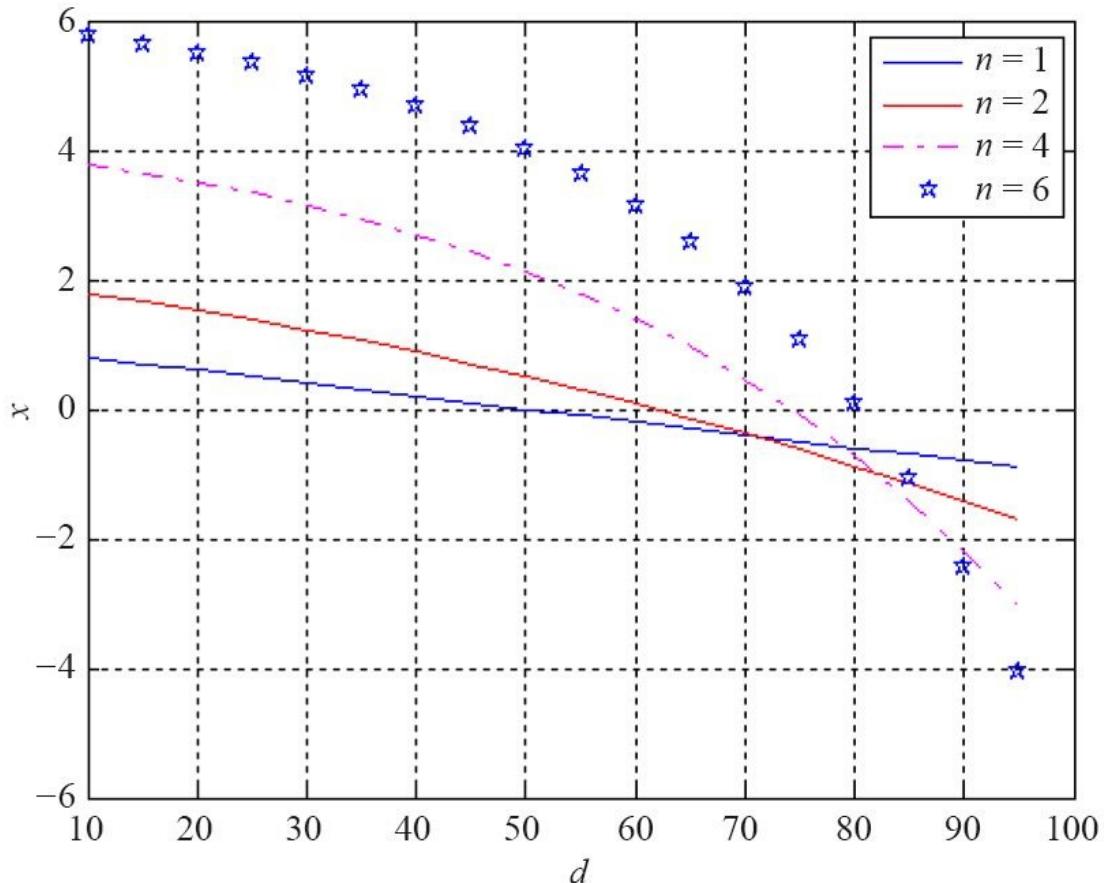


图6-4 自动化测试脚本开发与维护成本规划

结论：当考虑自动化测试成本收益时，我们应该先考虑对那些可能迭代次数更多、运行次数更多的测试用例进行自动化脚本开发。而对于产品的质量缺陷，质量缺陷越少、质量越好的产品，自动化开发成本收益也会比较大。反之，则致使自动化开发并不合算。例如，当前测试用例很可能最多执行2次，但产品中的遗留问题可能使得60%的测试用例不能通过，这时没有必要考虑自动化测试。

而且，即使产品中质量缺陷很少，但是测试用例可能被使用的次数非常少（少于3次），那么自动化测试的开发成本也只允许极少量投入，或许并不可行。

调换n和d的关系，我们再来看看图6-5中x的曲线。

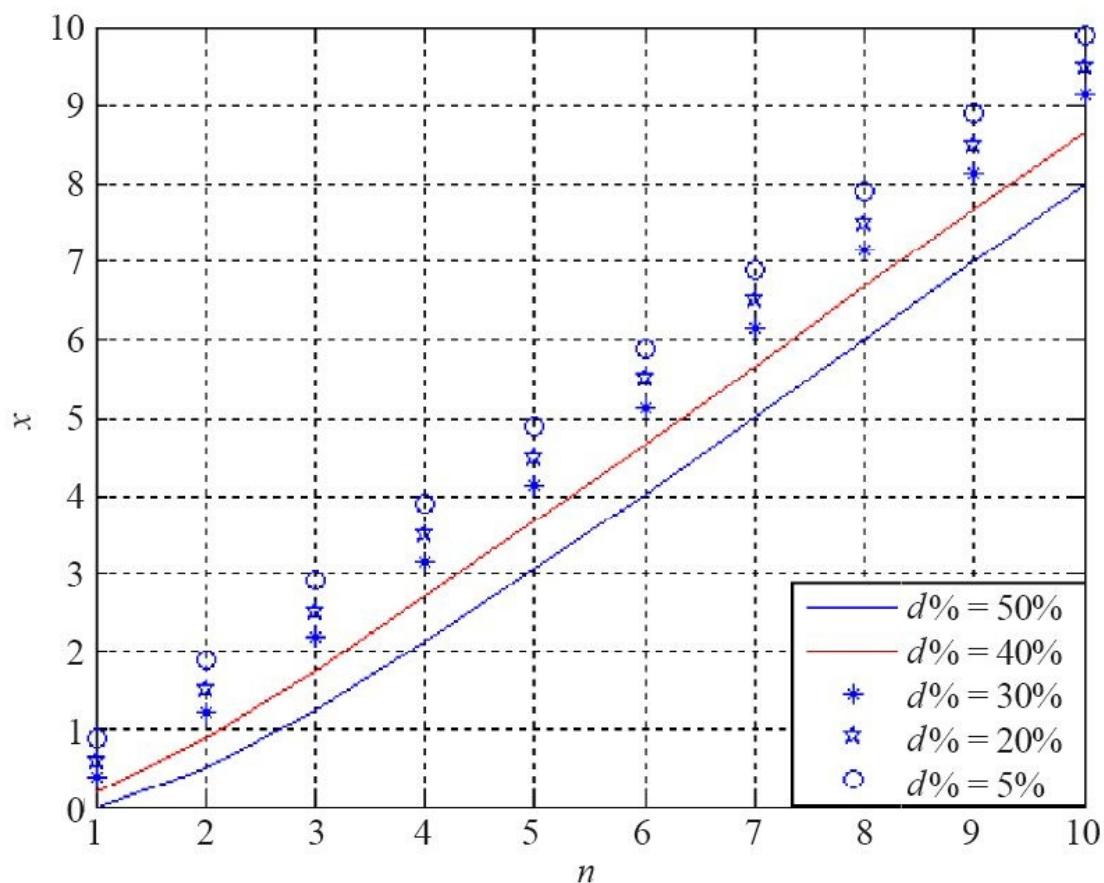


图6-5 自动化测试脚本开发与维护成本规划

结论：继续对自动化测试脚本开发与维护成本规划进行分析，我们还能得到另一个信息，那就是迭代次数，即测试重复性基本与可允许投入的自动化开发成本成线性关系。当测试需要迭代的次数是 $n=k$ 时，允许投入的自动化开发、维护成本可以是 $n=2$ 时的 k 倍。

6.5.6 敏捷测试的自动化

自动化测试规划的复杂性已窥见一斑，敏捷测试、Scrum开发模式下的敏捷测试隐约比我们分析过的情形要更加扑朔迷离。众所周知，Scrum模式下的开发、测试遵循了独特的规则。

迭代模式使得代码逐步累加起来，每个迭代周期又是如此短暂，还没来得及验证最后几个质量缺陷，突然需求变化了，一半以上的测试用例和测试脚本不再有用。

由于需要对每个迭代的新功能和新需求进行测试和研究，而迭代周期又很短，因此不容我们考虑诸如测试代码复用这样的问题，也没有时间来计算多少基于上一迭代的测试用例和脚本需要再度测试。

简单地说，测试节奏一下子加快了，在这样的压力下，更加追求迭代价值，以使每项开发、测试、研究活动一定能够服务于当前迭代即将诞生的代码和产品。更多的人开始谨慎选择自动化策略，缩减自动化投入。因而，更多的问题指向敏捷开发专家，我们是不是需要自动化测试，何时开始自动化测试，多少投入自动化测试工具和脚本的开发是值得的。

关于要不要做自动化测试，我们的回答是肯定的。迭代模式使得代码量逐步累加的同时，越靠后的迭代，我们所面临的整合测试压力、测试任务就越重。而且敏捷测试需要测试人员能够随时启动自动化的回归测试对马上发布的迭代代码进行快速验证。

我们借鉴传统测试中自动化测试规划的细则，基于敏捷迭代的特点对自动化测试规划做

进一步分析。

6.5.7 与传统自动化的对比

迭代次数 (n) 、测试覆盖次数越多，自动化测试带来的好处就越多，而产品开发中出错率 (d%) 越小，自动化测试效果越好。这是我们在之前的分析中得出的结论。

但是，因为迭代目标和对象的差异性，自动化脚本可能需要大幅修改。而重新构造和改变测试脚本带来的代价和成本也非常大。

特别是在敏捷测试中，产品变化之大给自动化测试带来的难度也陡然增加。因此，敏捷测试中，我们要以更精细的单位来计算自动化测试的投入产出比。即决策敏捷自动化测试的投入产出应该基于每个迭代自身的收益，而不是整个产品开发周期。

在每个迭代的2~4周的时间内，我们将经历从测试策略的选择、测试脚本和用例的撰写到即刻的测试执行和短暂的回归测试。我们不能保证回归测试完全自动化，因为我们可能没有足够的时间投入自动化开发，而正是因为测试脚本和用例的生命短暂，我们开发的成本也如期望的一样缩减了。

因为敏捷模式在不同项目中的实践存在较大的差异，所以在进一步分析敏捷测试下的自动化测试规划之前，我们先来描述一个理想的敏捷测试模型。

在这个模型里，我们的迭代周期为4周，团队共有7名成员，其中只有1名是测试人员。在这4周的活动中，我们将讨论敏捷测试的自动化测试规划方案。如图6-6所示。

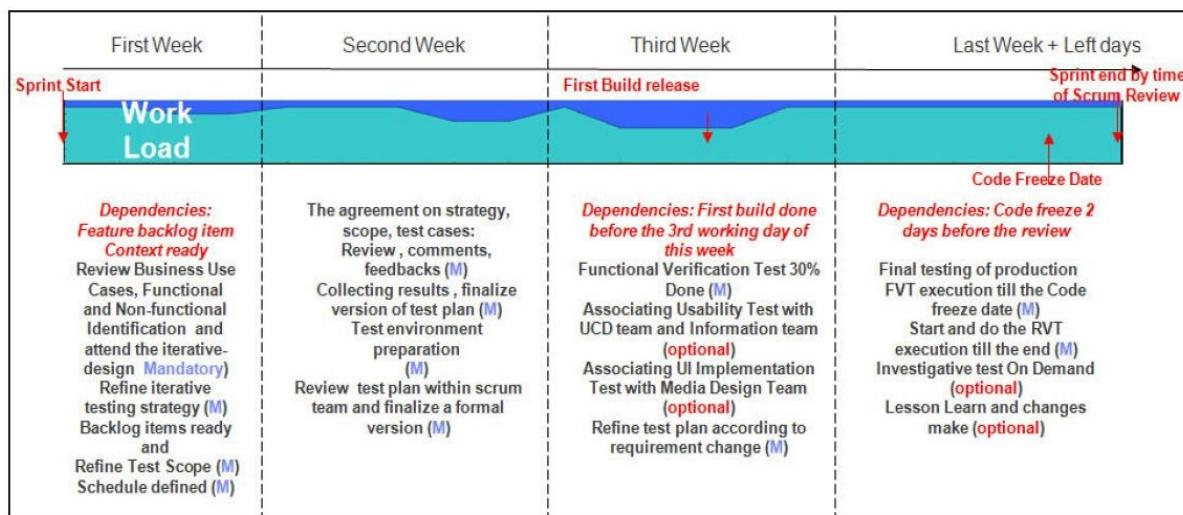


图6-6 敏捷测试工作流最佳实践

迭代的第一周，测试人员集中确定测试策略，制定可行的测试计划并划定充分的测试范围。

第二周，准备开始测试的执行，因此要准备好测试用例和测试环境。需要注意的是，测试人员是根据需求和团队讨论而设计出的用例来开发测试用例的。

第三周的第三天，基于约定，第一个可以执行的产品代码已经能够发布了（这个约定的履行需要整个团队的配合），之后我们便可开始功能测试。直到第三周末，一部分功能测试已经完成，大部分关键的新功能得到验证。

第四周，我们将要结束迭代的测试，在这一周中，不但要完成新功能和新非功能需求的测试，也要顾及持续性开发带来的老代码的质量风险。也就是说，产品的回归测试将在这周

内开始并结束。同样，基于约定，我们将迭代最后的两天或者更少时间作为回归测试阶段。正如基于Scrum的模式拥有严格的周期制度一样，一旦进入了回归测试阶段，所有人都要配合、遵守Code Freeze（代码冻结）等时间约定。

6.5.8 自动化的ROI

我们对自动化的规划也将融入这个4周计划。同样，为了更好地分析，我们需要几个假设和声明。

声明1：n为进入回归测试后我们仍然需要对测试用例反复测试的次数。

声明2：d%为这段时间内测试的平均出错率。

声明3：TotalEf为执行一次回归测试所花费的执行成本。

假设1：产品迭代周期为4周。

假设2：在此次迭代中，第一个可测的产品发布于第三周的第三天。

假设3：回归测试于第四周的倒数第一或者第二天进入，在此之前测试人员要完成新功能测试。

假设4：一个迭代达到退出的标准是 (d%) $n < 5\%$ ，即95%的测试都通过了，这样的迭代才能使客户满意。

因此，迭代次数n就必须满足

$$n \geq \frac{\lg(d\%)}{\lg 0.05} \quad (\text{公式 10})$$

根据测试的最佳实践，我们假设测试人员将用迭代的最后1~2个工作日完成回归测试。

因此，如果TotalEf是执行一次回归测试用例所花费的代价的话，那么

$$\frac{1}{n} \geq \text{TotalEf} \leq \frac{2}{n} \quad (\text{单位：人天}) \quad (\text{公式 11})$$

根据公式10和公式11，我们可以得出

$$\text{TotalEf} \leq \frac{2 \lg(d\%)}{\lg 0.05} \quad (\text{公式 12})$$

将公式12代入公式9得到公式13（不是十分严谨，不过不影响分析）

$$x \leq \frac{2 \lg(d\%)}{\lg 0.05} [n - 2d\% (d\%^n - 1) / (d\% - 1)] \quad (\text{单位：人天}) \quad (\text{公式 13})$$

绘制其曲线时，n为横坐标，取值为1：1：10，x为自动化工具、脚本开发维护的最大投入。

如图6-7所示，我们可以得到以下基于敏捷测试的自动化测试规划的规律：一般产品的测试错误率高于20%，因此，为了达到质量标准使测试足够退出，回归测试至少需要3次，在这种情况下我们允许投入到自动化开发的成本不多于3人天。

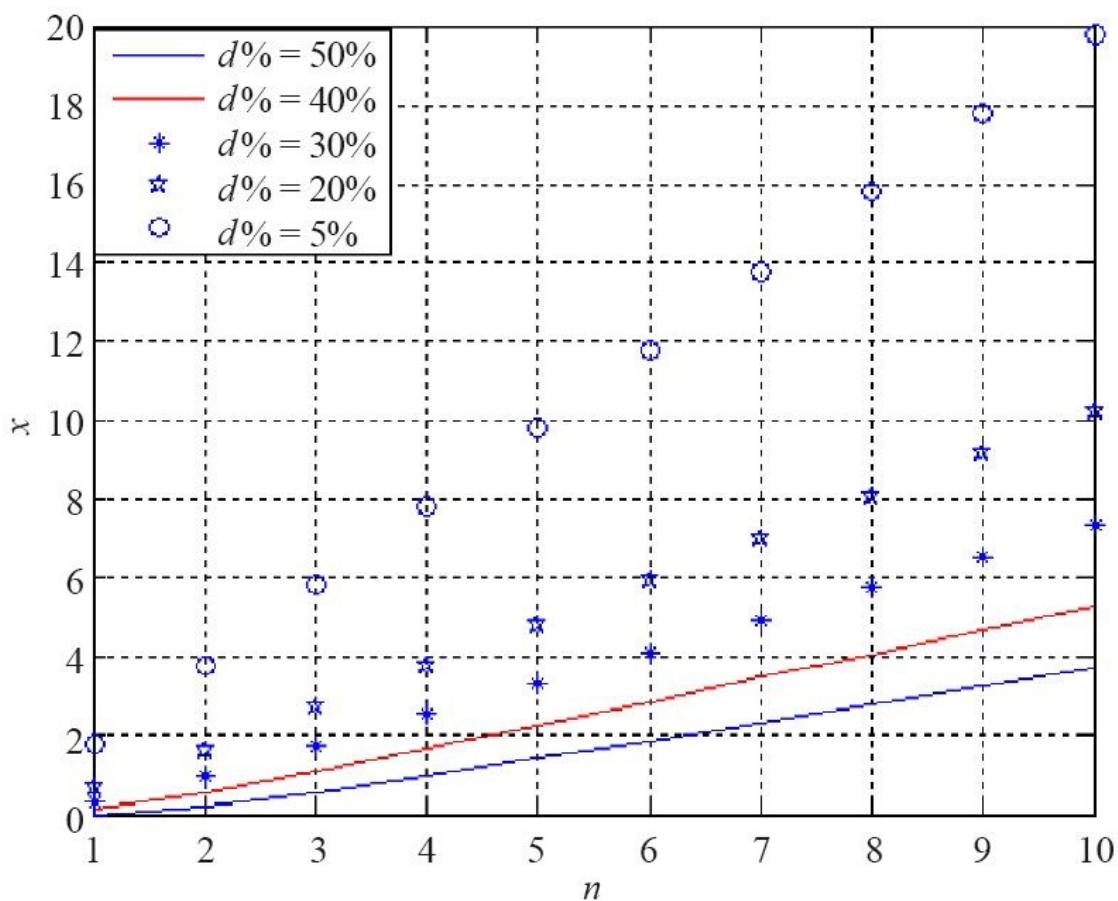


图6-7 迭代中自动化测试脚本开发与维护成本规划

而当产品质量非常好，错误率低于20%时，因为不需要经过多次测试便可达到退出标准，可以省去自动化测试的步骤。

最后是一点经验之谈，当我们从事敏捷测试活动时，在以4周为周期的迭代测试中，测试人员在第二周的开始进入自动化脚本开发，开发活动不宜超过3天。

6.5.9 小结

基于假设的数学公式所指示的结果和客观规律令人吃惊，但我们的的确感同身受，投入自动化测试的研究和调查越久，越客观地感受到自动化测试的好处和弊病。自动化投入并非越多越好，自动化测试对产品的质量提高作用也不是很大（目前不是，希望将来可以）。我们并不指望自动化测试可以取代手动测试，但是测试人员合理、科学地使用自动化测试，是提高测试成效的途径之一。ROI的自动化规划将非常适合敏捷测试、传统测试的最佳原则。

而成功的自动化测试除了拥有良好的规划外，自动化成本越低，开发工具越简易，自动化维护和管理复杂度越小，自动化测试也越容易驾驭。因此，在同等自动化规划下，测试人员应当采用更成熟的自动化测试工具，积极参与自动化测试经验交流，以不断提高测试自动化开发的生产率，以有限的投入获得更大测试收益。

6.6 故事分享：Rational中国敏捷测试工程师的一天

6.6 故事分享：Rational中国敏捷测试工程师的一天

·主人公登场

我叫英才，IBM开发中心一名极其普通的测试工程师，2年前进入IBM，现在已经是资深的小band6（一般硕士毕业生进入IBM的级别）。我去年进入Rational团队，曾经做过FVT、GVT、自动化测试开发，如今正在做敏捷测试。

8：10，和女朋友道别后，我步行到离家仅仅100米的333路车站，像往常一样，一边发着微信向群里的朋友问候，一边查看着最新的微博，只要10分钟左右，这班车一定到站。我有些机械地顺着人群，依着不是很规矩的人流上了车，站在距离中门很近的位置，我喜欢这里，因为这里总是人少些。我将音乐开大些，慢慢享受这一路摇摇晃晃的旅程，我在心理准备着，因为这一天紧张的工作即将开始。

·开始一天的工作

8：55，我来到公司座位，顺手点开显示器，简单地扫了一眼满屏正在自动化运行的自动化测试任务，确认测试昨夜正常运转后，打开写测试报告的php页面，再次确认大部分的结果均已上传成功后，我锁定了屏幕，坐下来，打开了我的笔记本，等待它的启动。

9：00，用最大的水杯打上一大杯水，用小咖啡杯打上一杯公司的免费咖啡，坐在桌前，第一件事就是查看邮件，第一封邮件是RTC……第二封……第三封……

我喜欢做好每天的计划，这是Alex多次给我们介绍的时间管理方法之一，于是我在座位的白板上写下了几个简单的词汇，算是用来提醒吧。

·温顾而知新

9：10，我决定对近来Alex给我们讲的敏捷测试知识做个温习，因为我此时的头脑最好。因为初步进入敏捷项目，我还在寻找跟随步调和锻炼职业操守，我非常高兴Alex为我们团队准备了敏捷“测试手册”，这非常实用，让我看清敏捷测试的工作内容、时间切入点，Alex告诫我切勿等待，更加主动地、聪明地工作，多和其他职能的同事（比如开发人员、需求设计者）沟通。

·参加Scrum例会

9：30，Scrum团队的会议召开了，来自美国的Matt主要沟通了新近一轮讨论中需求和设计的最终方案的改版，Alvin这个天才的开发团队队长则告知他已经更新了这个迭代的代码原型，这简直太神速了，我在考虑这家伙一定在晚上开夜车了，不过我很难理解他为什么每天都这么精力充沛。轮到我了，我首先向大家介绍今天已经是第二个星期的第三个个工作日了，我的测试用例和测试计划即将完成，我友善地提醒了团队负责人以及我的开发者同事们，下周也就是3天后就需要进行代码测试。我很窃喜，因为这次没有听到任何人的抱怨，他们似乎已经习惯了我这个小碎催。第二件事，我向团队汇报的是我的自动化测试版本经过了一夜的正常运转，现在大量的测试报告已经上传到了网上，开发团队昨天发布的最新版本已经经过了足够的验收测试和回归测试，这个版本可以作为新的里程碑继续进行增量型开发。当然，这是个很好的消息，我看到Alvin的一丝得意表情，自然是Alvin在团队的代码审查工作上取得了质的飞跃，不然不能有这么好的测试结果。这对于我来说更是个好消息，我可以更多的精力集中思考新功能的测试方法和新性能指标的度量。

小提示：在Scrum会议中，测试人员需要规划测试活动和更新进度状态。如图6-8所示，测试活动，如测试策略、测试计划、测试环境的搭建、自动化开发等都可以在全团队的产品清单中登入，可以由自己更新进度，在Scrum会上尤其需要关注并主动提出测试进度如何能够顺利开展、所要完成当前工作的前提条件和所需帮助等。测试人员同时还要承诺，何时可以完成环境搭建、配置管理工作，或将要花费多少个小时完成新迭代的功能和非功能部分测试，或需要多频繁地做回归测试以确保已开发功能和稳定产品的增量开发。

ID	Sprint Backlog Item	Owner	Sprint Day				
			12	13	14	15	16
			Day of Week	Mo	Tu	We	Th
			Day of Month	15	16	17	18
			Hours remaining	475	431	431	431
23	scripting the drop-down menu itself implementation	Austin	40	40	35	35	35
24	scripting the expanding/opening of groups within the menu tree implementation	Austin	15	15	15	15	15
25	HLD of Context/navigation menu client tier	Austin	10	5	0		
26	Dojo tree v 3 investigation	Austin	10	16	16	16	16
27	refine middle tier	Anne	16	10	10	10	10
28	feedback from UX and QA team	Austin	10	10	10	10	10
29	Test strategy and initial test plan for Context menu	Alex	10	3	0		
30	Test plan for Context menu	Alex	9	7	7	7	7
31	Test plan integration for Context menu	Alex	1	1	1	1	1
32	Test environment for Context menu	Alex	6	6	6	6	6
33	FVT for Context menu	Alex	17	17	17	17	17
34	Regression Test for Context menu	Alex	4	4	4	4	4
35	scripting the expanding/opening of groups within the menu tree implementation	Tao Tao Tian	20	20	20	20	20
36	scripting the first- and second-level overlays implementation	Austin	20	15	15	15	15
37	HLD of Alert/monitoring overlays/popups client tier	Tao Tao Tian	10	10	10	10	10
38	refine middle tier	Anne	20	20	20	20	20
39	feedback from UX and QA team	Anne	10	10	10	10	10
40	Test strategy and initial test plan for Alerts	Alex	10				
41	Test plan for Alerts	Alex	9	1	1	1	1
42	Test plan integration for Alerts	Alex	1	1	1	1	1
43	Test environment for Alerts	Alex	6	6	6	6	6
44	FVT for Alerts	Alex	17	17	17	17	17

图6-8 测试工作计入了产品清单

·透过仪表盘看团队

9：25，我从会议室回到了座位上。我很想知道为什么在今天的会议中团队负责人极少干涉，我很快打开了RTC客户端，访问了我们团队的Dashboard，就目前看来进度很好，不得不说这个Team的队长经验丰富，知道在该说话的时候表达犀利，而不说话的时候也能感到他有只无形的手在掌控着一切，不对，这只手是在身后托着我们，让我们更快地向山顶攀登而去。

·团队讨论设计

10：00，我和Mat、Alvin像往常一样拨入会议，虽然我和Alvin在一个办公室，但是Alex建议我们各自通过电话拨入网络电话会议，通过视频和电话来沟通，因为只有这样，我们才不会在开会时耳语，不会让电话线那边的同事觉得无理。团队的网络会议录制开始后，我们接着上次的设计进行讨论。

我：“Mat，你说你需要在用户点击窗口的时候弹出一个浮动窗口，是吗？”

Mat：“是的，英才。”

我：“请问如何关闭这个浮动窗口？这个浮动的窗口相对于屏幕多大？或者有准确尺寸吗？这个窗口应该是固定的还是可以拖拽的？再有就是，如果出现内容较多，是否需要一个滚动条，这个你有考虑和设计吗？”

.....

·做哪些测试

每每和需求、设计人员沟通的时候，永远都要考虑如何将业务需求尽可能准确地变成可以编程、可以测试的技术性需求，我们的经验是将提问的角度延伸至验证逻辑、多线程时用户可以接受的延迟时长或者安全隐患等。这可能会带来一系列的分析和再次融合统一，但这个讨论非常有意义。因为这个过程不但使团队对迭代将要开发的目标、验收指标达成一致认识，还可以大量减少由于需求不明确、需求模棱两可而产生假设理解，进而导致产品在迭代验收中失败的问题。这个过程我们称为静态测试，通常发生在迭代前半部，随之而来的是需求和业务流程的明晰和确定，测试策略、计划和测试用例的完成。最佳的实践是测试人员与团队代表们一起做1~2次的测试用例审阅，以达到大家对目标和细节的一致性，推动团队工作的有效进展。

经验表明，静态测试将减少产品在发布时期20%左右的质量缺陷，导致这些质量缺陷主要是因为需求不全、需求不清。静态测试的开展没有统一的执行步骤，测试人员需要带着原则和足够的想象力来测试这个还未构成的产品，逻辑型判断、缜密地捕获需求和设计中遗失的细节，这些都是静态测试中的重要品质。

静态测试需要测试切入点。如何全面测试？如何查漏补缺？第一步，我们从产品发展的趋势、从宏观的视角理解产品的定位、产品族的竞争力，了解产品当前版本的重要价值以及同类产品的差异。第二步，了解当前迭代所关注的最重要的功能指标、非功能指标（如性能指标）。第三步，进一步细化基于用户体验和行业惯例的用户操作流程，推测最有可能被用户挑战的组件或者操作流程，找出用户最有可能需要而还没有开发充分的操作流程，反复思考、假设如果用户操作出错，能否简单地返回出错前状态，还需要考虑多用户并发操作对数据造成修改冲突的可能性、竞争资源的可能性等。敏捷测试的系统层次结构如图6-9所示。总而言之，对产品长期战略和业务的熟悉可以帮助测试人员更好地理解团队的用例设计、视图、模块等，也能够通过对比分析，直接找出某些设计中的缺陷，提高设计的质量。项目的开发前期阶段，设计占有非常重要的比例，而设计是直接影响产品质量的环节，因而确保这一阶段的质量对提高产品的品质起决定性作用。针对开发出来的用例，设计者对用例的描述通常比较简单，缺乏完备性和缜密性。而开发和测试需要的是详细的设计，需要将用例定义得清晰、明确，例如边界条件、例外条件、数据的格式和其他性能指标的界定等。因此，测试人员应该帮助团队明确分析更多的细节。有原则，即宏观战略指导微观行为；有立场，即用户习惯与满意优于技术实现代价，并以此甄别需求和设计中的缺陷。

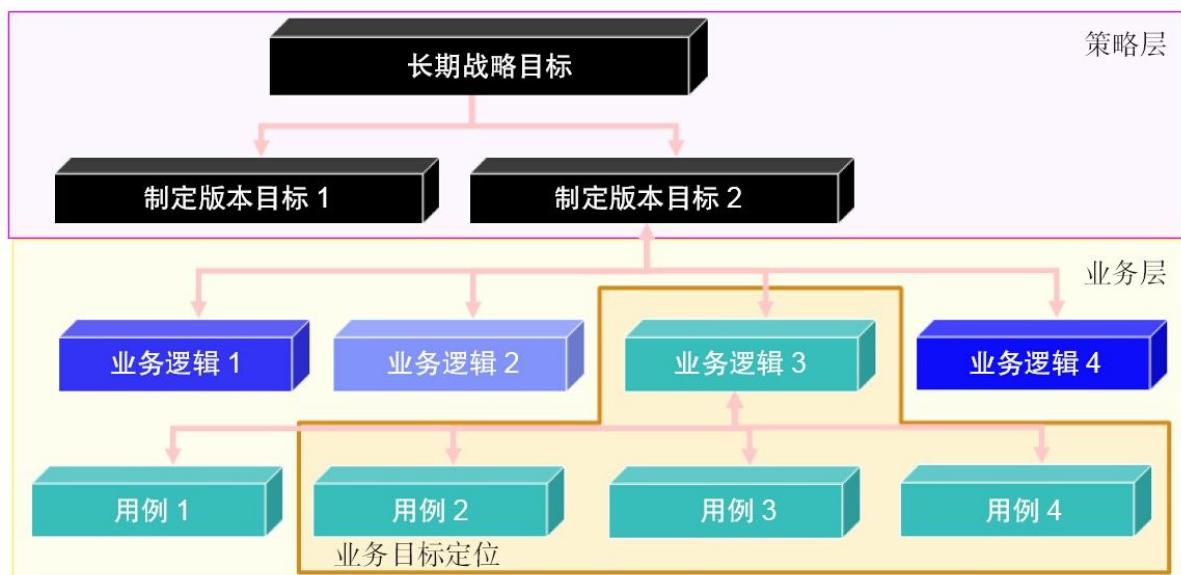


图6-9 敏捷测试的系统视角

这部分工作是测试团队与开发团队和设计团队合作最默契的阶段，交流非常频繁，正是通过积极的沟通和及时的修正，团队方向更加明确，更有凝聚力，更利于发挥团队的最佳战斗力。通常，这个阶段需要约1/4迭代周期的时间，这同时也说明了静态测试在敏捷测试类型中的重要性。

11:30，会议结束，我放下耳麦，很快地将几个重点细节记录在RTC的工作项里面，将本次会议录音的链接也发到了RTC管理的相关工作项下面。锁定电脑，和测试团队的同事去楼下食堂吃午餐。

·午休

13:00，回到工位，沏上一杯红茶，细细品品……开始更新测试用例和测试计划。

·编写测试用例

测试用例的撰写和传统测试基本没有差异，按照已有的模式撰写就好了。有些测试团队使用XML文件格式保存所有测试用例，主要原因是便于复用和版本控制，同时也因为XML元素可以很灵活地被查找和更新，而且在自动化测试中还可以通过程序抽取测试用例的模块信息形成自动化的测试报告、需求测试跟踪报告等。后来，这种方式被更加智能和强大的Rational Quality Manager (RQM) 取代，RQM可以更好地和需求工具、工程配置管理工具集成，自动化生成报告、进度跟踪。

·使用QM管理测试

在QM的角色设定中，有测试管理者、测试领导者和测试人员三种，每种角色具有不同的工作职责和权限，对此，在敏捷团队的每次迭代中，测试人员实际上完成了三个乃至更多角色的工作，这个过程正是自我管理的体现，也是自我成长的需求。通常在RQM工具配置时，要赋予敏捷测试人员更全面的角色和更高的权限。RQM适用于敏捷、迭代、瀑布式开发模式的测试用例管理和进度跟踪，工具是方便我们工作的利器，决不可成为过程和方法的代言。图6-10为使用RQM管理敏捷测试各项工作的示意图。

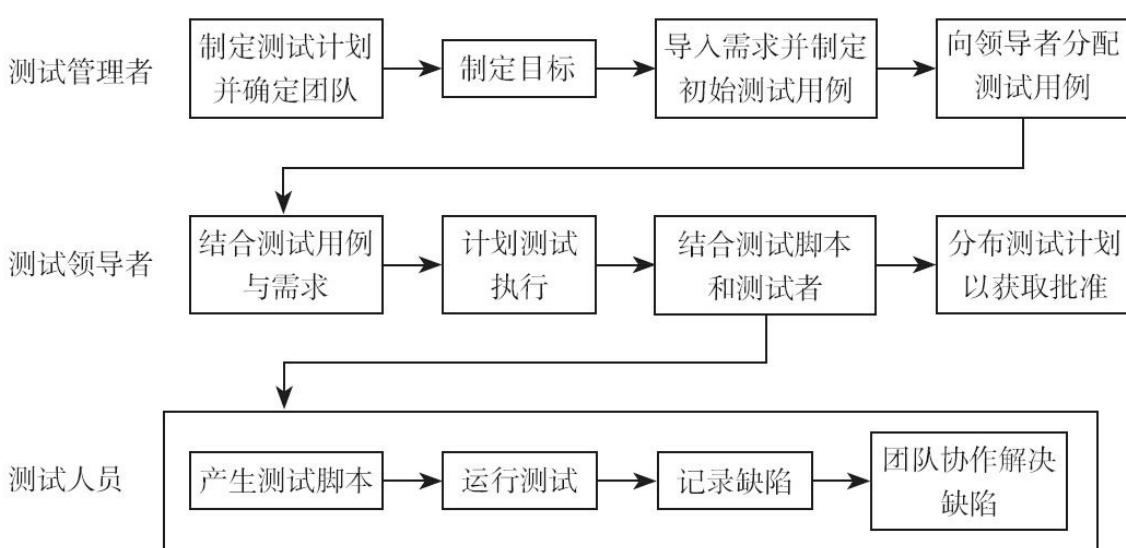


图6-10 使用RQM管理敏捷测试各项工作

·自动化完成执行

16:00，自动化测试已经马不停蹄地工作了近20小时，开发团队的新代码即将检入（Check in）到代码流，新一轮的版本验证自动化测试即将开始，至此，所发现的昨日代码所带来的的重要问题、关键路径问题已经在回归测试和版本验证测试的自动化测试系统上跑了一

遍以上（为了避免因为电子系统受周遭化学、物理条件变化而产生的不稳定测试结果，我们设计了自我检测式脚本错误，以及在新环境跑相同任务脚本时的自动化纠错逻辑）。此时，我很快地检查了这个老版本所遇到的各类问题，通过自动化测试报告的日志链接点击进入查看错误的缘由。

如图6-11的测试报告中，第一列为测试日志的URL；第二列关键平台和环境定义；第三列为测试版本的甄别编号；第四~六列标记测试任务的完成程度与结果（第六列以颜色标记执行状态）；第七列为测试机对应的主机名，用于区分是哪个具体机器或者虚拟机运行了当前任务；第八、九列为测试脚本、测试任务的分类，例如数据源Teradata相关、或者informix10相关等；第十列为自动化测试脚本的状态标记，即正在运行、准备运行和完成运行等。

Results	Real_Release	Level	Successes	Failures	Slave	Class	SI_Class	Status
ccased1_lb_rt_tera_e113701	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
cdtabd1_lb_rt_tera_e113343	dt2_v10ifp1_ax64	s120619	6	1	regres1@maradona	server_tera	teradata	completed
creed1_lb_nt_infx10_e113712	dt2_v10ifp1_sun64	s120719	3	4	regres1@grape	server_ifx	informix10	completed
crithd1_lb_nt_odbc_e114699	dt2_v10ifp1_sun64	s120712	0	0	regres1@grape	server_odbc	odbc	badrun
ddtransd1_lb_nt_tera_e113349	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
djcn_lb_nt_infx11_e104510	dt2_v10ifp1_sun64	s120719	16	3	regres1@grape	server_ifx	informix11	completed
djcn_lb_nt_tera_e113340	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
djcn_lb_nt_odbc_e114972	dt2_v10ifp1_linuxamd64	s120715	20	2	regres1@ronaldo	server_odbc	odbc	completed
djco_lb_nt_syb125_e113934	dt2_v10ifp1_sun64	s120614	0	0		server_syb	sybase125	not_run
djct_lb_nt_tera_e113734	dt2_v10ifp1_linuxamd64	s120624	13	2	regres3@cherry	server_tera	teradata	completed
djct_lb_nt_tera_e114645	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
djjob_lb_nt_tera_e115002	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
djjob_lb_nt_tera1a1_e115272	dt2_v10ifp1_sun64	s120614	0	0		server_tera	teradata	not_run
djjobud_lb_st_ora10_e13107	dt2_v10ifp1_linux390x64	s120626	109	1	regres5@zlnx5	server_ora	oracle10g	completed
djjobud_lb_st_ora11_e114979	dt2_v10ifp1_sun64	s120719	106	4	regres1@grape	server_ora	oracle10g	completed
djrp_b_asa_v5_3_ax64_e00_112900	dt2_v10ifp1_sun64	s120712	0	17	regres1@grape	server	_asa4tcpip	completed
dmlj_lb_nt_odbc_e113650	dt2_v10ifp1_sun64	s120719	190	3	regres1@grape	server	null	completed
dmlj_lb_nt_infx11_e108568	dt2_v10ifp1_sun64	s120719	6	179	regres1@grape	server_ifx	informix11	completed
dmlj_lb_mva_v8_1_mvs_e110119	dt2_v10ifp1_linuxamd64	s120621	188	2	regres3@db2cc113	server	_mvsdbpp	completed
dmlj_lb_nt_odbc_e114975	dt2_v10ifp1_linuxamd64	s120621	188	2	regres3@db2cc113	server_odbc	odbc	completed
dmlj_lb_nt_syb15_e104014	dt2_v10ifp1_linuxamd64	s120621	191	2	regres3@ronaldo	server_syb	sybase15	completed
errdj_lb_nt_odbc_e114949	dt2_v10ifp1_ax64	s120712	18	1	regres3@maradona	server_odbc	odbc	completed
fed2p1b_lb_st_ora11-odb_e113582	dt2_v10ifp1_ax64	s120712	17	1	regres3@cookie	server_ora	oracle10g	completed
fed2p2b_lb_nt_ax400-odb_e114448	dt2_v10ifp1_linuxamd64	s120705	21	1	regres3@ronaldo	server_mssql	mssqlserver	completed
fed2p3b_lb_st_informix-odb_e111457	dt2_v10ifp1_sun64	s120719	5	14	regres1@grape	server_mssql	mssqlserver	completed

图6-11 可以实时监控的测试执行状态报告

小建议：自动化测试的成功率、测试脚本不依赖环境和机器的鲁棒性，这其实是有效自动化测试错误分析的基础，可以想像，如果自动化脚本换一个机器跑就会出错，测试人员的精力将大量投入到维护环境中，这是非常没有价值的事情。所以，最佳的做法是：很好地设计自动化测试脚本，做好对物理环境或者操作系统环境的兼容；自动化测试执行引擎需要有好的逻辑判断，当脚本在当前版本上失败，而在之前版本成功时，测试引擎应该自动激发测试重新开始，以避免因为测试环境的偶发因素导致的错误测试；每一个平台的测试都需要准备多台测试机，以确保实在没有时间处理测试机的环境问题或者测试机本身出现明确故障时不至于耽误自动化测试进度。

一天工作的回顾

17：45，也是即将下班的时间。因为今天是迭代的中期验收测试用例的日子，我非常清楚这个时机应该做两件事情：第一是确认开发团队从现在起就要发布代码，尽管或许不完整，及时发布代码是迭代质量工作得以完成的保障之一，也意味着我的正式测试执行期到来了；第二是测试用例需要团队的审阅和最终定稿，所以我利用最后的15分钟发送了会议邀请，邀请开发人员Dave和用户设计团队的Mat于明天10：00开会20分钟审阅测试用例，然后我做了RTC工作项的工作进度更新，就这样为我一天的工作画上句号。

小建议：敏捷测试、敏捷开发要有很好的时间观念，到了该发布的时间或者完成工作的时间就一定要尽最大努力完成，以确保团队的整体进度和他人的工作进度不受影响。我会建议工程师们都准备一个台历，在每个关键的日子的方块里标出要完成的工作；标出迭代起止日期，在大概1/2的时间点完成测试计划、用例、测试环境搭建，3/4的时间点完成60%的功能测试和关键的用户验收测试等。图6-12给出了工作台历的示意图。

July

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1	2 迭代计划 会议	3	4 美国团队的 节日	5
6	7 测试策略 完成	8	9 详细测试 完成计划	10 用户设计 工作 DCUT	11 测试用例 初稿完成	12
		$2 * 8h$ for Detail Test Plan		→		
13	14 Lucy 的 生日	15	16	17 测试用例 完成	18 第一个可用的 Build 发布	19
20	21 UCD 参与 FVT FVT 开始	22 召开测试 会议	23 FVT 80% 完成	24	25 调查型测试 (可选)	26
		$4 * 8h$ for FVT		→		
27	28 回归测试	29 迭代工作 评审会议	30 享受两天 的假期	31		

图6-12 人手一份的计划和提醒利器——台历

6.7 选择自动化工具

6.7 选择自动化工具

大多数人都曾经倾向于自己研究一套“很有效”的测试工具，并努力证明其在目前和将来的项目中可有效运行且节约成本，单从这个角度来看这是件好事，但是从开发中心6000人的总体投入出发来看则并非如此，过多的自动化测试工具在维护和开发着，大家都为了能够争取更多的“用户”和“关注”而展开了一场无休止的“口水战”。仅我们的自动化测试社区、测试社区半年来不断组织的大小讨论，都在讲一个又一个貌似独创但又有几分相似的“最佳工具”。

为了将自己的工具做到“最好”，一些团队不惜投入将工具做得十分“强大”，而此时我们发现，投入已远远超出工具本身带来的收益，甚至影响到了中国区的财报，是时候为自动化测试指出更合理的发展方向了。

借此机会，我即刻拟定了当年开发中心测试社区的首要目标是调查开发中心自动化测试的发展状况；再者就是找到自动化测试的新的指导方向；最后希望可以找到一种科学的方式，在“好”工具中寻找“合适”中国研发中心的自动化工具。这项计划获得了批准。

6.7.1 度量尺度

我们首先重组了测试社区自动化测试的领导团队，选择现有“自动化工具”分布相对集中的三大领域，即自动化执行API测试、自动化执行GUI测试、自动化执行环境配置领域。然后在这三个领域上集结了相关自动化测试专家分成三个团队，分别为API测试、GUI测试和环境配置团队，而每个团队的成员都是相应领域的自动化测试专家，至少都参与设计了一款自动化测试工具。团队的使命是在有限时间内提出尽可能丰富的、可量化的指标对团队内所有人所代表的工具的指标打分，指标一定是具体的，可以但最多有2~3层细分的指标，所有叶子节点指标定义一定是明确的而不是模棱两可的。最后我们鼓励精确的分类技术，每个团队都有7~8人每两个星期利用1~2小时完成对一个工具的详细分析和集体打分过程，被打分工具的代言工程师将竭尽所能向大家展示自己工具的特性，在陈述中只要是之前没有讨论或提及的特性都会转化成一项性能、功能度量指标留下来，相类似的功能或者性能指标又将和以后的性能指标做整合，这样对所有工具都基于一套相同定义、相同尺度的指标打分，每项指标可以是1分或者0分的“有和无”，而不是0~10分的程度分值。

例如，我们研究了GUI自动化测试工具后得到的度量指标矩阵有10大簇集：

- Test Functionality（测试功能）。
- Test Environments Building（测试环境支持）。
- Test Case Creating（测试脚本创建能力）。
- Test Asset Infrastructure（测试资产管理能力）。
- Deployment（部署形式）。
- Test Case Execution（测试脚本执行能力）。
- Test Result Display/Analysis（测试报告展示/分析能力）。
- Test Case Maintenance（测试脚本可维护性）。
- Integration with 3rd party tools（与第三方工具集成能力）。

·Other（其他）。

在10大簇集上分1~2层细分规则，如图6-13所示有关测试功能度量矩阵的细分和对相应工具的打分，我们可以了解FTX、SandBox和PACAS均支持Web2.0JavaScript的测试，除了PACAS都支持Windows操作系统上的测试，以及只有WCM支持全球化测试环境自动化测试。这个簇级指标包含三层指标，其中，第二层分别是Supporting OS（支持的操作系统）、Supporting Browser（支持的浏览器）、Web 2.0Support（是否支持Web 2.0技术）、Support Globalization execution environment（是否支持全球化测试环境），第三层则是各操作系统、浏览器和Web2.0对象。

Test Functionalities ^②	Supporting OS ^②	Windows ^②	FTX ^②	SandBox ^②	MTP ^②	TMC ^②	WCM ^②	Jester ^②	Squash ^②	LA ^②	PACAS ^②
②	②	Windows ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	②
②	②	Linux ^②	②	②	Y ^②	②	②	②	②	②	②
②	②	UNIX ^②	②	②	Y ^②	②	②	②	②	②	②
②	②	AS400 ^②	②	②	②	②	Y ^②	②	②	②	Y ^②
②	②	z-OS ^②	②	②	②	②	②	②	②	②	Y ^②
②	②	Other OS ^②	②	②	②	②	②	②	②	②	②
②	Supporting Browser ^②	IE 6 ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②
②	②	IE 7+ ^②	②	②	②	②	②	②	②	②	②
②	②	Firefox ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②	Y ^②
②	②	Safari ^②	②	②	②	②	②	Y ^②	②	②	②
②	②	Mozilla ^②	②	②	②	②	②	②	②	②	②
②	②	Other browsers ^②	②	②	②	②	②	②	②	②	②
②	Web 2.0 Support ^②	JavaScript ^②	Y ^②	Y ^②	②	②	②	②	②	②	Y ^②
②	②	AJAX ^②	②	②	②	②	②	Y ^②	②	②	②
②	②	Flash ^②	②	②	②	②	②	②	②	②	②
②	②	Feeds ^②	②	②	②	②	②	②	②	②	②
②	②	Mashup ^②	Y ^②	Y ^②	Y ^②	②	②	②	②	②	②
②	②	Others ^②	②	②	Y ^②	②	②	②	②	②	②
②	Support Globalization execution environment ^②	②	②	②	②	②	Y ^②	②	②	②	②
②	②	②	②	②	②	②	②	②	②	②	②

图6-13 基于设计的矩阵为工具打分

有了这个功能矩阵后，团队设想出了有意思工具选择故事。如图6-14所示，当用户在为自己的项目挑选自动化测试工具、测试工具时，只需要将右表中的功能点（Feature 1~100, etc）拖入对应的“必选”（Mandatory）、“最好有”（Best to have）、“可选”（Optional）三类筛选池，这样系统将会从现有的工具列表中，筛选出满足“必选”（Mandatory）的工具给用户。



图6-14 系统根据用户对项目的描述筛选工具

我们来模拟下结果的输出，当用户完成了筛选后，系统自动绘制了当前项目需求下的雷达图，系统会从现有的工具库中选择并展示出满足条件的所有工具的雷达图，用户可以选择任何一个结果集中的工具作为项目的备选自动化测试工具。如果用户还希望选择更好的工具或完全符合需求的工具，那么我们将建议用户选择一个能够尽可能包含项目雷达图的工具作为最优工具。不同雷达图的示例见图6-15。

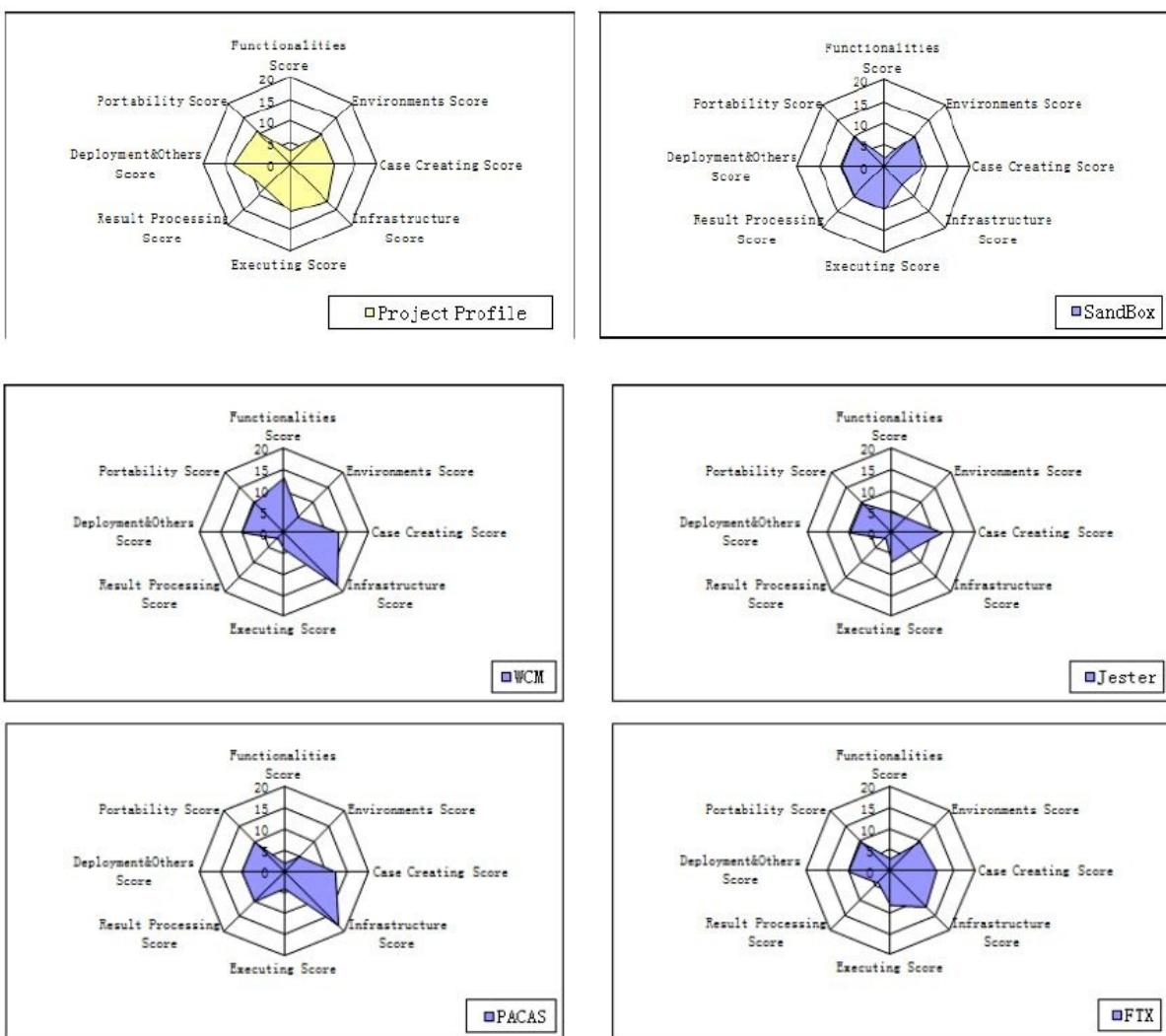


图6-15 不同雷达图示例

6.7.2 量化指标

因为自动化测试工具相似度很高，或者有太多的选择，所以我们推荐另外一种优选方式，即通过工具成熟度、支持能力和投入回报比做进一步筛选。为此，我们研究了行业最好的技术文章，探访了专家，与研究部门合作找到了评价工具的另外三组指标。

·Maturity & Customer Base (MCB)，工具成熟度。通过工具的使用时间、用户数量，以及这个工具被运用于有相当大小的项目团队的数量综合计算获得。

·Tool Support (TS)，工具支持能力。当工具在使用中出现错误，通过工具支持团队解决问题的平均时间；用户查询相关文档要点的平均时间以及在文档查询时遇到文档错误、误导问题的几率综合计算获得。

·Estimated Return On Investment (EROI)，投入回报比。考虑投入使用自动化工具前后，对于测试进度、发现质量缺陷数量的对比，投入维护和开发实践的综合计算获得。

这三个度量指标和计算方法的选择源于两篇非常精彩的文章，分别是James Michael的《Metrics for Measuring the Effectiveness of Software-Testing Tools》和Daniel Rowley的《An Automated-Test-Tool Evaluation and Selection Technology》。而同时度量矩阵还经过了自动化专家组的讨论和修正，下面介绍这些指标的定义。

$$MCB = M \times CB + P$$

M (Maturity) 是自从第一个发布至今的整年数，单位[年]。例如，一个工具有三个版本发布了，第一个版本发布的时间是2000年，假设今年是2013年，那么M=13。

CB (Customer Base) 是指使用工具超过1年的经验用户的数量（可以统计在中国开发中心获取、了解、支持和被支持的用户数量）。例如，有20个开发中心的项目团队使用过这个工具，每个团队平均10名工程师，他们都已经使用这个工具超过1年，那么CB=20×10。如果你并不确定有多少团队正在开发中心使用这个工具，则只需估算有多少可能的用户使用这个工具超过1年之久。

P (Project) 是以“人年”为参考单位，计算在开发中心总共有过多少人年的使用经验。例如，开发中心有3个项目团队已经采用了这个工具，平均人年数是1.2py，那么P=3×1.2。再如，开发中心有2个项目团队已经采用了这个工具，其中一个项目有3py，另一个项目有2py，那么P=3+2。

$$TS = (ART + ARTAH) / 2 \times DS + ATSD \times (1 - DS)$$

ART (Average Response Time in Test) 指用户刚开始试用测试工具时，工具的支持团队平均解决用户发现问题的时间，单位[天]。例如，在用户测试工具阶段，一个工具错误发生了，并且花去了用户约0.5天来解决这个问题，那么ART=0.5。

ARTAH (Average Response Time After Test Schedule) 指当用户经历过测试工具阶段，正式使用工具时，支持团队平均解决用户发现问题的时间，单位[天]。例如，在论证了工具的可行性后，如果用户使用时发现了工具的错误，并且花去用户1天时间来解决这个问题，那么ARTAT=1。

ATSD (Average Time by only Search Documentation) 指当用户发现一类问题时，这类问题能够通过查询工具文档、在线帮助解决的平均时长，单位[天]。例如，如果用户花去0.25天查询和阅读文档解决一个问题，那么ATSD=0.25。

DS (Document Insufficiency) 指的是多少比例的问题用户无法通过查阅说明文档、在线帮助，而只能通过客服、支持人员解决，单位[无]。例如，如果用户曾发现10个问题，平均评估有4个问题不能够从文档中获得帮助，必须从用户团队、支持团队那里获得帮助才得以解决，那么DS=4/10。

$$EROI = (EPG \times ETT \times ACTH) + EII - ETIT \times ACMH + (EQG \times EHCS \times ACCS)$$

EPG (Estimated Productivity Gain) 是生产力的增长点，单位[无]。即过去100个真实日历日内，度量项目平均测试的生产力的增长点。若采用了新的工具后，团队或者个人的工作量为过去的120%，那么EPG=0.2。

ETT (Estimated Testing Time without tools) 是在没有工具帮助的前提下，以原有的测试生产能力计算可以完成多少个工作量的测试工作，单位[人时]。例如，如果没有这个工具，团队在过去的100个真实日历日内，可以完成300个人时的工作量，那么ETT=300ph。

ACTH (Average Cost of Test per Hour) 指测试工程师每小时的生产成本，单位：[\$/h]。例如，开发中心主力测试工程师约28.6\$/h，辅助测试工程师约18.7\$/h，如果一个团队的构成是3个主力测试工程师、2个辅助测试工程师，那么ACTH=24.6\$/h。

EII (Estimated Income Increase) 是使用了新工具后直接产生的盈收，单位[\$]。例如，当我们使用这款工具后，除了生产力得到提高，工具本身给我们的工作带来了直接的盈收\$50，那么EII=50（如果并没有直接的盈收，EII可以为0）。

ACMH (Average Cost of one tool Maintenance Hour) 是使用新工具后在后台维护工具、二次开发时所需花费的每小时费用，单位[\$/h]。我们基本认为ACTH=ACMH。

ETIT (Estimated Tool script Implementation & maintenance Time) 是估计的自动化测试脚本开发、工具维护所需花费的时间，单位[人时]。例如，如果过去100天内总共花费了80人时，则ETIT=80。

EQG (Estimated Quality Gain) 是使用工具后对比之前工作的质量盈收，单位[无]。例如，使用了新工具后，发现了多余原来10%的质量缺陷，那么EQG=0.1。

EHCS (Estimated Hours of Customer Support per defect) 是指每发现一个缺陷所需提供技术支持的平均时间，单位[小时]。例如，每个缺陷平均需要5小时支持服务，则EHCS=5。

ACCS (Average Cost of hour of Customer Support) 是每个测试阶段遗留下来并流入到用户手里的质量缺陷将带来的平均用户支持代价。例如，每个用户支持小时都将付出62.5\$的代价，则ACCS=62.5。

基于以上算法，我们的团队对所有的142个自动化工具做了综合评分对比，得到每个工具的MCB、TS和EROI，我们将其中几个最有特色的工具进行对比，如图6-16、图6-17和图6-18所示。

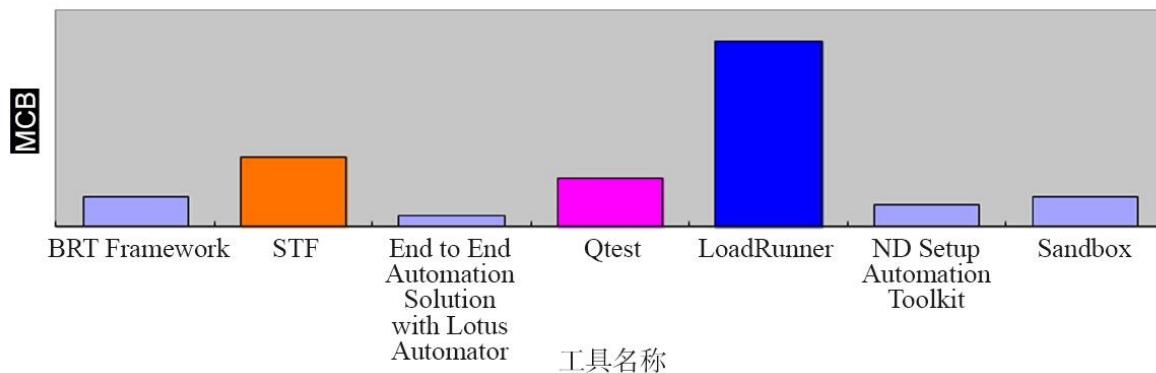


图6-16 成熟度最高和用户基础最多的自动化测试工具排名

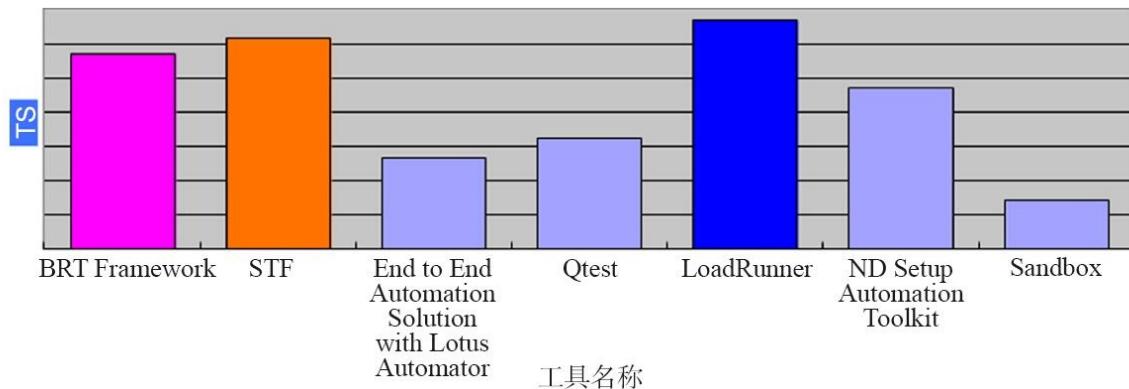


图6-17 技术支持最优的自动化测试工具排名

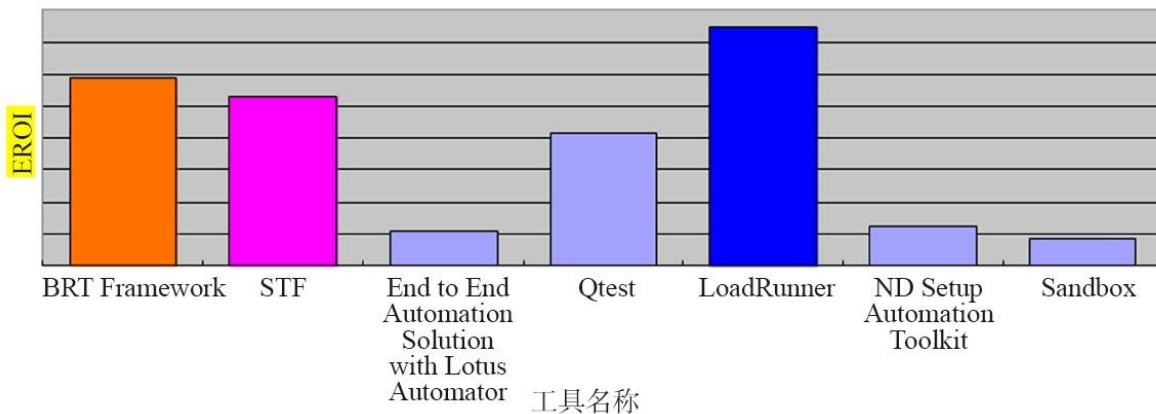


图6-18 投入产出比最优的自动化测试工具排名

如图6-19所示，我们将每个工具的三位属性以一体化、直观的方式展示，用户在挑选其工具时将更为简单。



图6-19 功能矩阵与属性尺度一体化的展示

至此，我们分享了2008年开发中心的一项调查和研究成果，即“工具能力矩阵”和“可量化指标”，而这个方法的提出本身，其意义已经超出我们最初的期望，它可以指导自动化工具开发回归理性。这项工作从规划到完成，得到各个部门领导和团队的支持，近40名技术专家集思广益保证项目顺利进行。

我们分享这个故事和方法，是希望让更多的同仁有所收获，能够从这群可敬、可爱的人身上感受到向上的精神。这个成果属于IBM中国开发中心自动化测试社区，属于参与其中的每位志愿者。时过境迁，物是人非，这中间的许多人或许已经认不出来了，但我希望留下每一位贡献者的姓名，谢谢这些可爱的人们，与你们共事非常愉快！

姓名随机排列，成绩不分先后，他们是：

Zhang Yan, Li Chen, Jian Xiao Xiang, Cheng Jun Qiang, Chen Yong Jie, Tao Rui, Zou Dian Guo, Yang Peng, Zhang Guo Jun, Zhao Hou Fang, Bian Jiang, LV kai, Liu Kang, Mercury Li, Zhang Shi Jia, Li Yun Li, Yang Zhi Bao, Deng Jun Ning, Gao Xin, Xie Ming Zhi, Chen Xiao Kang, Li Lu, Gao Yu Hang, Xu Guang Xin.

6.8 案例分析：改进自动化工具

6.8 案例分析：改进自动化工具

在开发自动化系统RECBIN（垃圾箱）之前，我们的DB3项目版本为v9，小版本v95，Fixpack为FP6，产品每年都发布3~4个Fixpack，而当年我们同时面临主版本v10的压力，一半左右的开发和测试人员被抽调到新版本的开发上，剩下不到一半的工程师将对老版本的工作遵循维护和修复的策略，因而在v9的FP项目中，回归测试是工作重点。

不久以后，我们被告知测试人员还要减少，但仍需完成同样甚至更多的测试工作。同时，新版本的开发工程师因为曾在v9的工作中承担重要模块的开发工作，虽然要求平缓交接工作，但仍然会负责修复很多遗留下来的质量缺陷并检入到v9的版本中。也许是因为不够重视，在v9上检入的策略带来了大量的回归问题，这给我们的测试团队带来了不少往复的工作量，v9的内部发布版本在很长一段时间内都表现不佳，测试团队压力非常大；而且随之而来的v10的投入逐渐增加，v9上的工作越来越表现出测试在拖后腿，疲惫不堪地在抱怨声中哀怨度日。

我就是在这时加入了这支9个人的测试团队，我们的主要任务是负责v9和v10的冒烟、功能、回归测试，我们需要用两个月的时间对原有测试工作做出改进。

v9和v10的功能测试主要面对的是已经开发的约220个测试用例，部分用例使用C++、C语言编写，部分用例使用Java语言编写，其余大多使用perl语言编写，中间不乏有链接数据、JDK等外部工具包调用，这些脚本和用例都是前辈工程师花了数年时间开发出来的，被认为是可以覆盖整个产品所有功能点（当然会有重复覆盖）的“系统工程”，也就是在脚本的文件、文件夹上做了合理的“分类”，每个脚本都包装成了“Bucket”，每个Bucket中都包含执行文件、结果文件、引用包链接文件等，每个执行文件的编写都很规范，有不少注释信息，新来的测试人员还是比较容易通过阅读每个Bucket的Readme文件以及执行文件的前几行注释上手工作的。不过，因为这些Bucket有的太过庞大，测试文件少说有几千行代码，一个重量级的Bucket光计算机运行就达到20个小时。好在还有在产品线上工作了近20年的老工程师，否则这些测试脚本真会成为我们的重负。

在接手项目时，资深的美国测试工程师与我们分享的不仅仅是这些测试用例的阅读方法，也给我们做了有价值的自动化监测运行时错误的自动化报告功能，这个功能也是迄今我发现的最有“范”的自动化测试，对我启发很大，在以后很多场合我都会自豪地向自动化测试工程师推荐这种方式。我们看看如图6-20所示Bucket的构成。

```
[12] [iifvt13@db2|cc113] /home/iifvt13/rtest
=> ls
cnickcre.run cube.crv cube.drv cube.trace err exe exp res run run_log test xnickdrp.run
```

图6-20 Bucket的构成

- 测试用例文件：cube.crv和cube.drv。
- 执行日志文件：cube.trace。
- 测试输出文件：run文件夹下名为cube.rrn的文件，同名的clp文件是入口执行文件即cube.clp。
- 标准答案文件：exp文件夹下正确的命名为cube.rxp的文件。

· 错误检查文件：错误发生时，run文件夹与exp文件夹下同名的文件对比出的“误差”打印到err文件夹下以.err结尾的同名文件。

· 测试结果统计文件：res文件夹下为cube.res的文件。

· 可执行文件：test文件夹下许多被调用的测试文件，如图6-21所示。

```
[26] [iifvt13@db21cc113] /vbs/test/fvt/standalone/cube/test
=> ls
census.xls      cube.clp      cubedrpor.clp  cubeinsudb.clp
cnickcreas4.sqc  cubecreas4.clp  cubedrprb.clp  cubeinsvm.clp
cnickcreexl.sqc  cubecreinf.clp  cubedrpsyb.clp  cubemapdt.sqc
cnickcreflat.sqc  cubecremsq.clp  cubedrpter.clp  flatdatacensus.txt
cnickcreinf.sqc  cubecremvs.clp  cubedrpudb.clp  flatdatasales.txt
cnickcremsq.sqc  cubecreor.clp  cubedrpvm.clp  flatdatat1.txt
cnickcremvs.sqc  cubecrerb.clp  cubeexl.clp  sales.xls
cnickcreor.sqc  cubecresyb.clp  cubeinsas4.clp  t1.xls
cnickcerb.sqc  cubecreter.clp  cubeinsinf.clp  xmldatacensus.xml
cnickcre. sqc  cubecreudb.clp  cubeinsmsq.clp  xmldatasales.xml
cnickcresyb.sqc  cubecrevm.clp  cubeinsmvs.clp  xmldatat1.xml
cnickcreter.sqc  cubedrpas4.clp  cubeinsor.clp  xnickdrp.sqc
cnickcreudb.sqc  cubedrpinf.clp  cubeinsrb.clp
cnickcrevm.sqc  cubedrpmsq.clp  cubeinssyb.clp
cnickcrexml.sqc  cubedrpvmvs.clp  cubeinster.clp
```

图6-21 Bucket目录Test下文件列表

不同类型文件的内容示例见图6-22~图6-24。

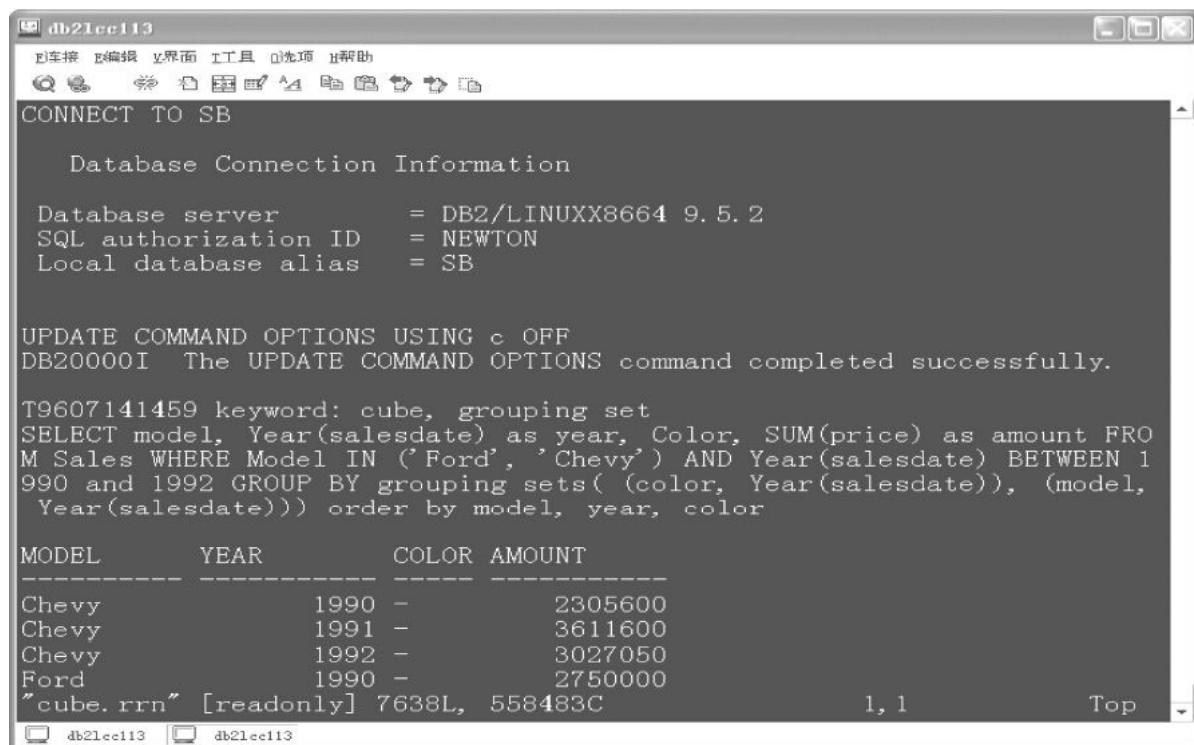
当测试产生错误，即run和rxp文件出现误差时，err会显示出“缺少”、“错误”、“多增加”的输出结果，这是我们定位错误的简单办法。err文件的内容见图6-25，测试结果见图6-26。

```
ORDER BY year, month, day;

-- Same query using Microsoft syntax.
SELECT Year(salesdate) as year, month(salesdate) as month,
       day(salesdate) as day,
       SUM(price) as amount
  FROM Sales
 WHERE Model IN ('Ford', 'Chevy')
   AND Year(salesdate) BETWEEN 1990 and 1992
 GROUP BY Year(salesdate), month(salesdate), day(salesdate) with ROLLUP
 ORDER BY year, month, day;

-- The query after semantic rewrite must be the following. It should
-- produce the same answer.
WITH agginput as (select * FROM Sales
                   WHERE Model IN ('Ford', 'Chevy')
                     AND Year(salesdate) BETWEEN 1990 and 1992),
     YMD as (
       SELECT Year(salesdate) as year, month(salesdate) as month,
              day(salesdate) as day,
              SUM(price) as amount
        FROM agginput
       GROUP BY Year(salesdate), month(salesdate), day(salesdate)
     ),
     161, 1
     8%
```

图6-22 可执行文件clp的内容



```

db2lec113
连接 编辑 离线 工具 选项 帮助
CONNECT TO SB

Database Connection Information

Database server      = DB2/LINUXX8664 9.5.2
SQL authorization ID = NEWTON
Local database alias  = SB

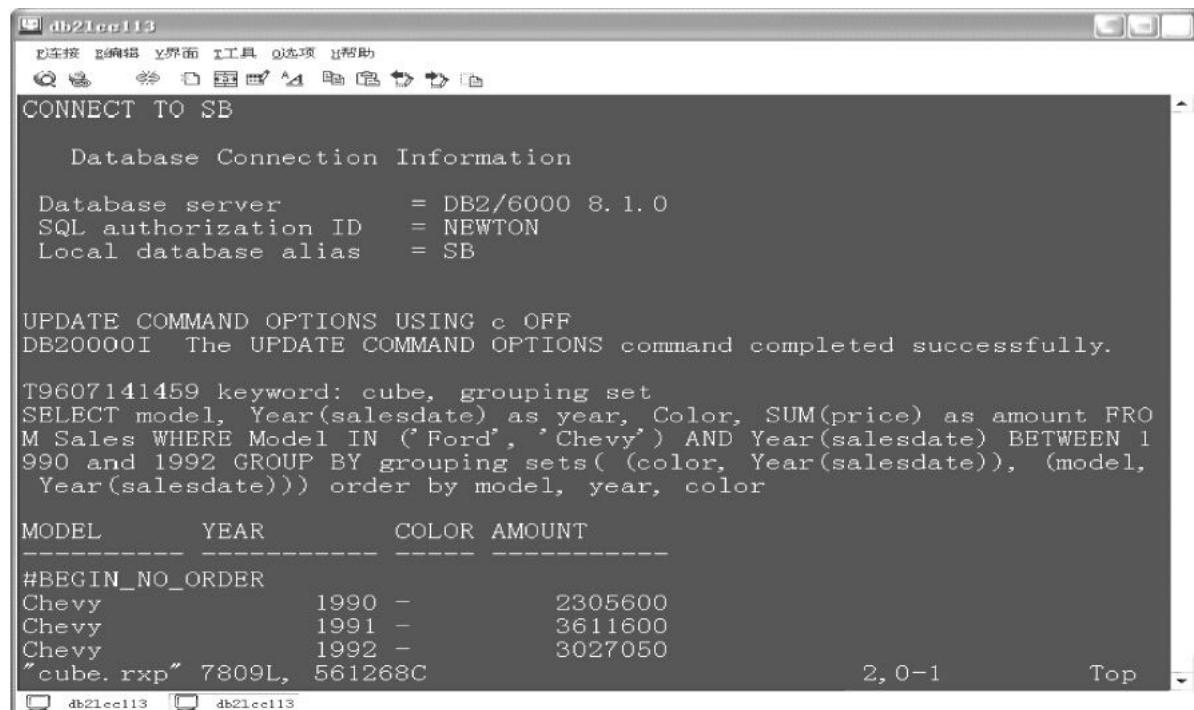
UPDATE COMMAND OPTIONS USING c OFF
DB20000I The UPDATE COMMAND OPTIONS command completed successfully.

T9607141459 keyword: cube, grouping set
SELECT model, Year(salesdate) as year, Color, SUM(price) as amount
FROM Sales WHERE Model IN ('Ford', 'Chevy') AND Year(salesdate) BETWEEN 1
990 and 1992 GROUP BY grouping sets( (color, Year(salesdate)), (model,
Year(salesdate))) order by model, year, color

MODEL      YEAR      COLOR  AMOUNT
-----  -----  -----  -----
Chevy      1990      -      2305600
Chevy      1991      -      3611600
Chevy      1992      -      3027050
Ford       1990      -      2750000
"cube.rrn" [readonly] 7638L, 558483C           1, 1      Top

```

图6-23 可执行文件run的内容



```

db2lec113
连接 编辑 离线 工具 选项 帮助
CONNECT TO SB

Database Connection Information

Database server      = DB2/6000 8.1.0
SQL authorization ID = NEWTON
Local database alias  = SB

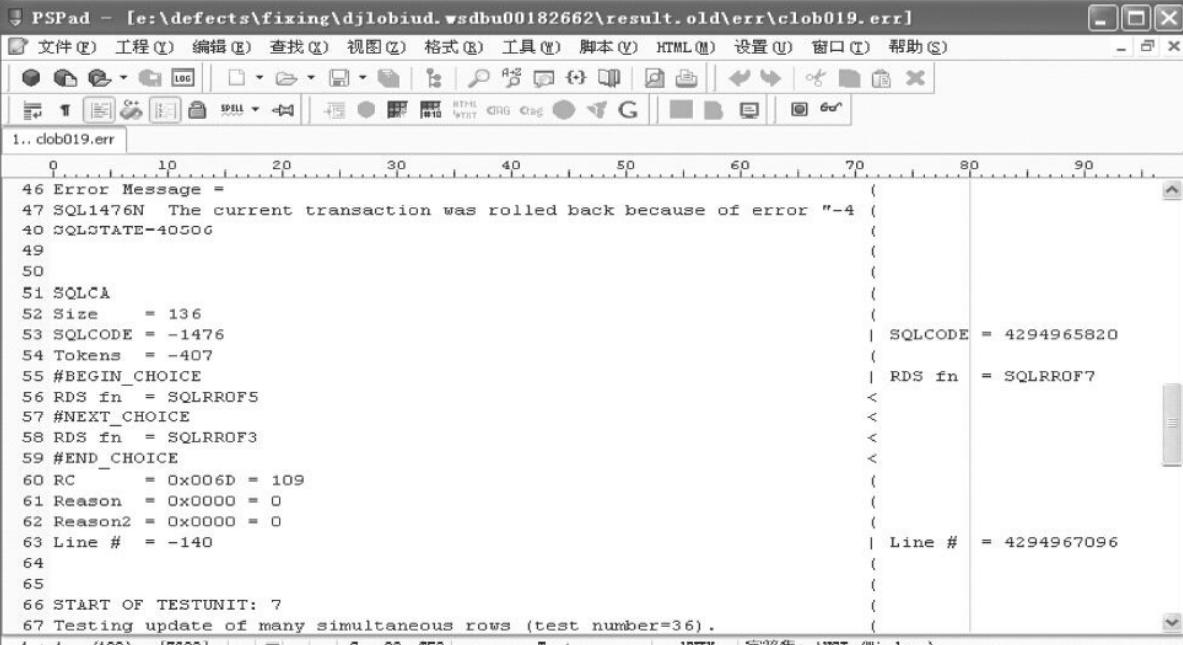
UPDATE COMMAND OPTIONS USING c OFF
DB20000I The UPDATE COMMAND OPTIONS command completed successfully.

T9607141459 keyword: cube, grouping set
SELECT model, Year(salesdate) as year, Color, SUM(price) as amount
FROM Sales WHERE Model IN ('Ford', 'Chevy') AND Year(salesdate) BETWEEN 1
990 and 1992 GROUP BY grouping sets( (color, Year(salesdate)), (model,
Year(salesdate))) order by model, year, color

MODEL      YEAR      COLOR  AMOUNT
-----  -----  -----  -----
#BEGIN_NO_ORDER
Chevy      1990      -      2305600
Chevy      1991      -      3611600
Chevy      1992      -      3027050
"cube.rxp" 7809L, 561268C           2, 0-1      Top

```

图6-24 可对比的期望值文件exp的内容

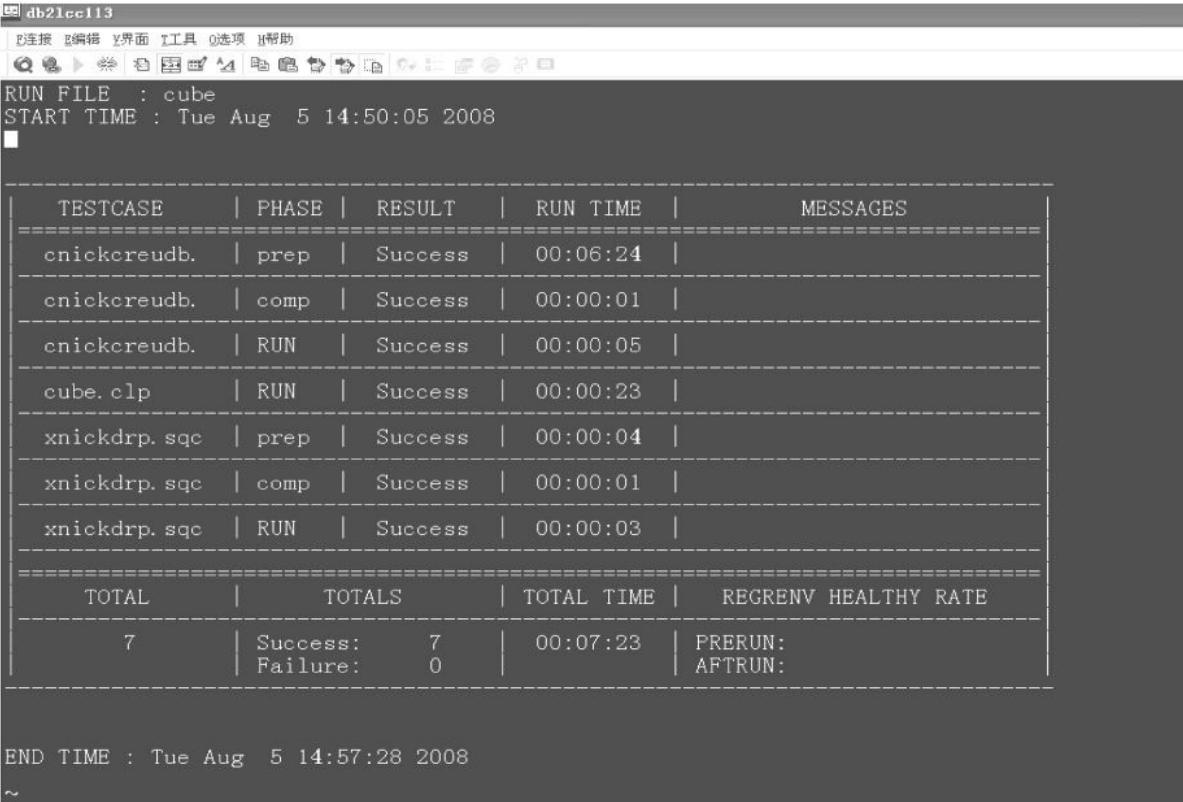


```

46 Error Message =
47 SQL1476N The current transaction was rolled back because of error "-4"
48 SQLSTATE=40506
49
50
51 SQLCA
52 Size      = 136
53 SQLCODE   = -1476
54 Tokens    = -407
55 #BEGIN_CHOICE
56 RDS fn    = SQLRROF5
57 #NEXT_CHOICE
58 RDS fn    = SQLRROF3
59 #END_CHOICE
60 RC        = 0x006D = 109
61 Reason   = 0x0000 = 0
62 Reason2  = 0x0000 = 0
63 Line #   = -140
64
65
66 START OF TESTUNIT: 7
67 Testing update of many simultaneous rows (test number=36).

```

图6-25 当RUN文件与EXP文件产生差异时生成的ERR文件内容



TESTCASE	PHASE	RESULT	RUN TIME	MESSAGES
cnickcreudb.	prep	Success	00:06:24	
cnickcreudb.	comp	Success	00:00:01	
cnickcreudb.	RUN	Success	00:00:05	
cube.clp	RUN	Success	00:00:23	
xnickdrp.sqc	prep	Success	00:00:04	
xnickdrp.sqc	comp	Success	00:00:01	
xnickdrp.sqc	RUN	Success	00:00:03	
<hr/>				
TOTAL	TOTALS		TOTAL TIME	REGENV HEALTHY RATE
7	Success: 7	Failure: 0	00:07:23	PRERUN: AFTRUN:
<hr/>				
END TIME : Tue Aug 5 14:57:28 2008				
~				

图6-26 测试结果生成

在测试中，可以说封装好的Bucket是一个拥有完整故事的测试用例，而手动执行这类测试需要上百步的操作，包括创建数据库、执行表单动作等，每一步都存在人为错误的风险，即使不出错，因为人为的干预远远比计算机的工作效率低下，测试工程师在这样的自动化环境下可以在450个计算机运算小时后完成一次完备的测试，这包括了100多组测试用例(bucket)，以及不同的配置模式：

- 122组使用Perl和C++编写的复杂用例

- 9个平台 (linux, unix, windows, etc)
- 8个数据源 (DB2, Oracle, Teradata, Sybase etc)
- 2个模式 (SER, MPP)
- 2个隔离模式 (DJ_FENCED=YES, NO)

自动化的平台不但可以自动化部署build，也可以从部署的build中自动化地执行回归测试，这个阶段称为API自动化测试阶段，经历半年的工作环境熟悉和技能的成长，测试团队的主力可以并发1~2个环境，发起指令执行自动化测试的清理环境、建立数据、测试、收集结果，而并发度很快成为瓶颈，资源的竞争也是问题。最重要的是，精力的分散使得结果并没有像设想的那样，增加测试机器并没有提高测试效率。我们找到了真实的原因，即bucket的执行依赖环境和配置的敏感度极高，一旦测试的准备和环境的清理工作不够到位，或者遗漏了一个步骤，我们的测试只能推倒重来。就这样，在自身技能和熟练程度的不断提高和并发能力的不断提升下，我们的效率和测试的进度在一定程度上一直保持着一个令人不怎么满意却很疲惫的水准，与此同时，来自管理层的压力和项目进度的压力迫使我们做出更大的调整。

API自动化测试阶段所遇到的最大挑战是：

·如果因为产品出现回归，我们将在两个月以后发现问题。

·我无法知道这两个月间哪个build质量最优。

·测试人员需要反复执行相似但又非相同的参数配置，而测试人员专注于测试配置以至于很难多线程并行，测试效率较低。

·测试人员将很多时间用于测试环境问题的诊断和排除，需要手动清理的频率很高。

·First Catch错误常常很难重现。

于是我们改进了自动化测试方案，特别是环境部署和配置，这部分工作极易出错，我们增加了perl语言写的autodeploy.pl+系统参数（例如linux64+20110301），这样可以跨平台安装和配置。为了在干净环境里运行测试，我们写了cleanmode.pl来删除无用临时文件，恢复数据库空间，将系统环境参数恢复到初始状态；此时，我们已经做到了相当程度的持续化集成+持续化测试。

·因人为失误和环境原因造成的测试错误显著减少，无效defect数量减少，测试的速度有了明显的提高。

·每个测试用例可以减少30分钟的等待和人工干预的时间，一年可以减少1180人时的工作量，相应地，完成一次覆盖的测试时间也减少了（但如果平均到每个版本发布中，这种提升还不够明显）。

那么持续化测试阶段的测试模式是怎样的呢？如图6-27所示，我们将bucket的执行进行了封装，这样可以使用计算机的批处理来批量执行而无需人为干预。为了统一输出和便于发现问题，我们将执行的输出也进行了一定的标准化，DJrun就是这样一个工具。通过这些操作，屏蔽了各种不同的语言完成的bucket在测试时使用不同方式执行的差异性（当然还是有些bug，测试首次执行成功率依然是一大挑战），在终端，测试人员只需要找一个slave测试机，当数据库、测试所需调用的外部其他资源均允许时，即可运行auto.pl。

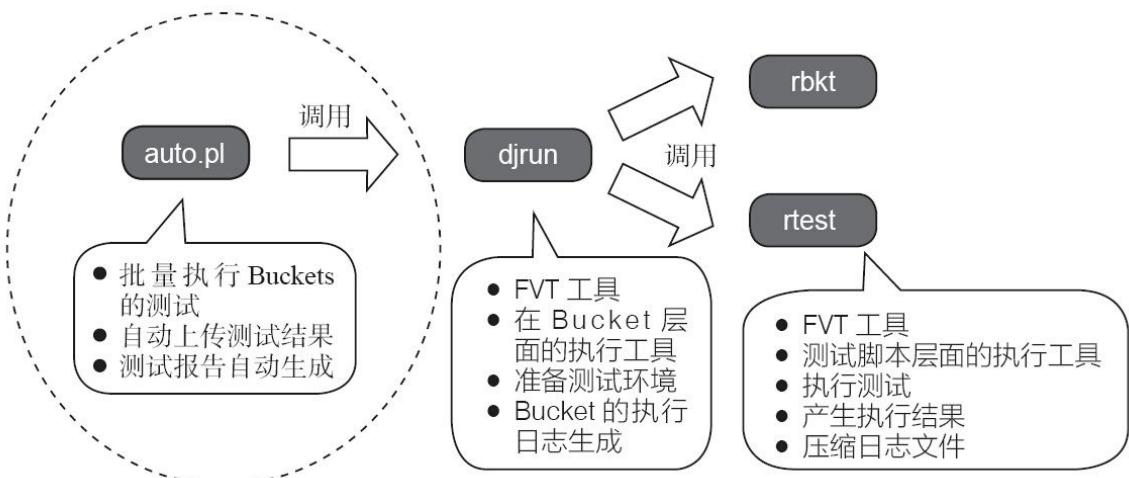


图6-27 AUTO.PL实现自动化加载测试脚本、准备环境和发出执行指令

我们设计了一套验证自动化测试产出比以及有效性的调研方法，包括5个问题，将其分发到每个测试成员手中，请他们如实填写，这5个问题是：

- 1) 如果使用100作为您过去执行日常测试工作所花去的工作量（没有当前测试工具），那么，如果使用当前测试工具完成这个工作，需要花费的工作量是多少（1~100）？
- 2) 有多少工作量花在了维护和开发这样的自动化工具上（1~100）？
- 3) 一个新员工需要花费多少时间掌握自动化测试的技能并独立完成工作（人周）？
- 4) 在最近的一次测试项目中，有多少测试用例已经被自动化了（百分比）？
- 5) 有多少质量缺陷是在自动化测试中被发现的（百分比）？

经过两个小组的参与，我们取每个小组的平均值，得到的结果是，使用持续化测试的自动化测试后，相比原来的API测试增加了约15%~30%的产出，见表6-1。

表6-1 方案1改进效果反馈调查结果

参与者	使用工具来完成 100 工作量需要的工作量	使用工具需投入的维护 (1 ~ 100)	学会使用工具花费的时间 (人周)	自动化执行比例 (Rtest, Qtest)	自动化测试暴露缺陷占所有测试的比例
组员 1	70	15	1.5	75	15
组员 2	70	5	1	70	20
校对	70	0	0.4	90	72
均值	70	0	0.4	81	0

这种持续化集成+持续化测试所遇到的挑战是：

- 多人并行测试时，经常会出现资源冲突，例如数据源同时被操作造成了死锁，或者是同一台测试机器被两个人同时操作造成崩溃等问题。
- 测试机器一旦出了问题，恢复起来所面临的问题比起先前更加复杂和耗时。
- 开发环境发布build以后，从美国到中国同步这个build需要近10小时，而不同平台上的版

本发布时间也不一致，需要人为kickoff清理环境和安装，从build同步完成到真正开始测试环境的部署，中间的时间浪费也不可小觑。

在进一步改进现有的持续化测试方案后，再次统计团队在使用新方法后的体会，我们使用了同样的5个问题，他们给出的答案如表6-2所示。

表6-2 方案2改进效果反馈调查结果

参与者	使用工具来完成100工作量需要的工作量	使用工具需投入的维护(1~100)	学会使用工具花费的时间(人周)	自动化执行比例(Rtest, QTest)		自动化测试暴露缺陷占所有测试的比例
组员1	70	10	0.05	80	80	12
组员2	20	10	0.04	85	85	15
组员3	20	10	0.07	80	80	0
均值	30	10	0.053	81.6		9

总而言之，不但总体的效率和投入产出比API自动化测试提升了60%，系统利用率、自动化覆盖范围和人力资源的介入频率也得到了明显提升：

- 系统利用率提高50%。
- 自动化能力和覆盖范围提高50%。
- 敏捷性和快速测试能力提高90%。
- 系统的大规模测试灵活性提高60%。
- 其他资源利用率提高了80%。
- 管理和人为介入的消耗减少了70%。
- 安全性提高60%。
- 资产以及人力资源消耗减少了90%。

这个改进的持续化测试系统是如何设计的呢？我们找准了需要自动化的重点区域，加强了自动化的程度，重新在原有系统的外部设计了自动化BI系统，也优化了内部的测试结果。

在外部使用Build Forge工具和编写的负载均衡Java程序联合起来自动启动build下载安装，分配workload。我们能找到的所有计算机，高性能的、低性能的，横跨9个平台共20多台计算机变成Slave测试机，而如果某台机器因为故障下线，系统的测试负载均衡功能可以将测试任务平均分配给剩下的可用测试机。

因此，一旦同步build被检测已经生成并复制完整，测试系统将立刻下载并配置测试机。工作完成以后，Build Forge的任务逻辑将做出条件选择，是继续测试或者将当前不佳状态的测试机退出测试计划系统，因此彻底解决了人为交互所造成的资源冲突问题，解决了build同步造成的时间等待以及bad测试环境影响测试进度的问题。可以说真正做到了持续化集成和持续化测试的所有工作。

而在自动化分析错误方面，我们在BI中增加了对典型错误的自动判断，将典型错误所带来的log关键词作为监控对象，一经发现即直接生成可读性的跟踪和质量缺陷报告。同时，Slave测试机往复运行，永不停息，每执行完一个循环均会寻找build同步系统中的最新build，

并进行自动下载和配置。

而测试的报告都将上传至服务器，测试报告的状态以测试时间和版本排序，这样很容易判断和排除偶发的无效质量缺陷。

下面给出改进的持续化测试系统的设计架构——RECBIN系统架构（如图6-28所示），希望对各位实现自动化测试框架有所帮助。

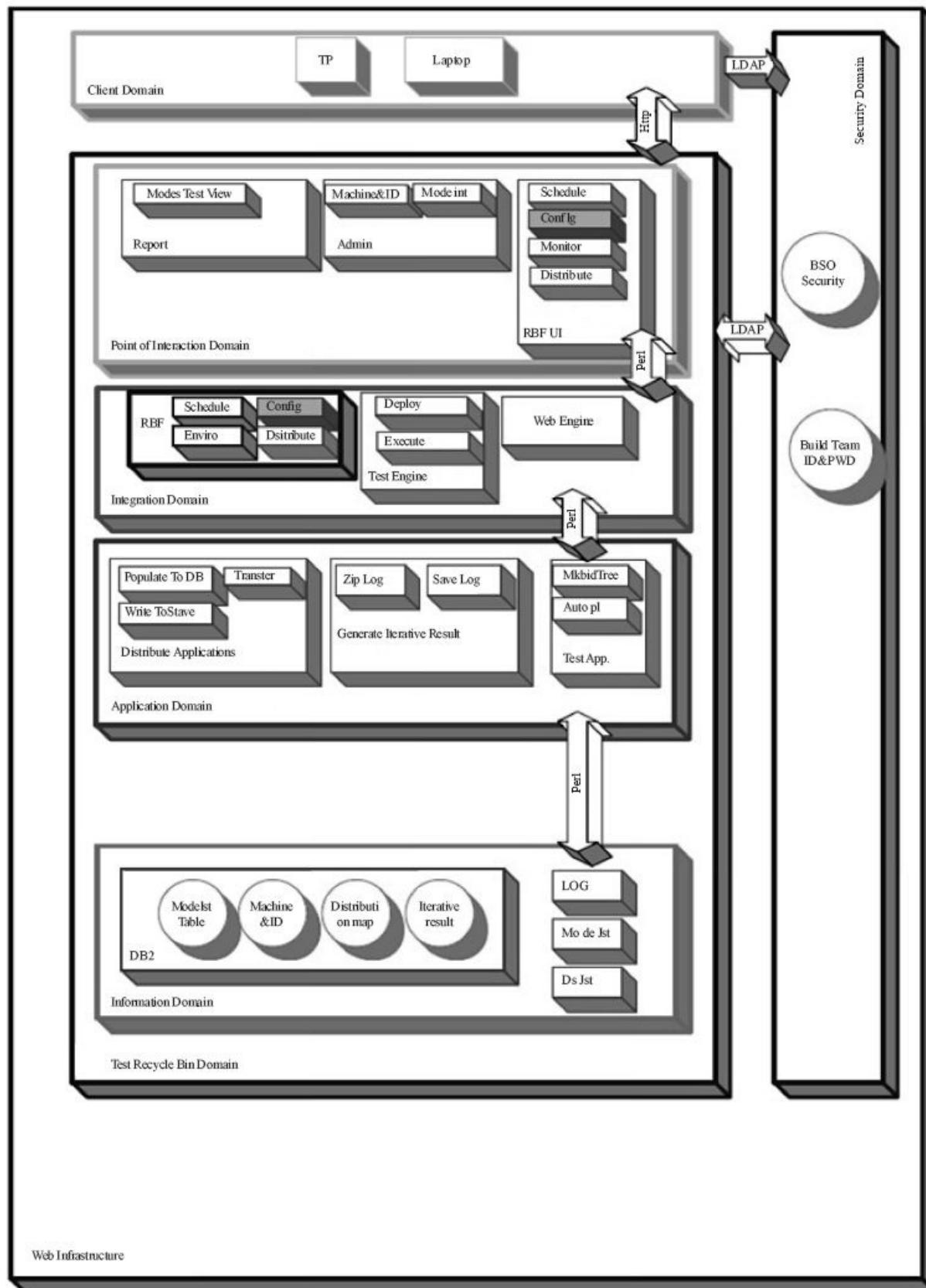


图6-28 RECBIN系统架构

附录A 有关敏捷思维、敏捷转型的常见问题和回答

附录A 有关敏捷思维、敏捷转型的常见问题和回答

以下收集了和朋友、客户、同事的探讨中常见的一些问题以及解答。

1.有关自动化测试和测试ROI的探讨

Q：读了你在IBM developerworks发表的《敏捷开发的最佳实践第4部分：自动化测试的ROI》一文，有一个问题想请教您：自动化ROI有标准吗？比如大于50%？如果没有标准，那么应该怎么衡量这个值呢？

A：亲爱的Sydney，据我看来，没有一个放之四海皆准的指标，在不同情况下，你可以灵活设定目标。例如，当自动化测试团队刚刚成立时，不建议设立严格的ROI要求。同时，如果团队为了自动化而自动化，甚至有重复发明同一个轮子的行为时，或许可以给一个严格的ROI指标来进行控制。我的ROI文章仅仅讲述当我们计划投入自动化测试时应该考虑的因素，而远非自动化测试的技术和自动化测试的覆盖率本身。所以，我给你的建议是逐步提高你所监管的自动化测试团队的ROI目标，以持续提高其效率和作用。

最后，我还想补充两个观点，第一，在考虑度量必要性时要从为什么度量出发，也就是你的目标；第二，在设计任何指标作为度量标准时，要考虑清楚是否会有伤害团队的可能，例如团队之间的互信，过往经验告诉我们，过多的度量会丧失团队的信任，一个没有信任力的团队所提供的数据以及度量结果往往是用来隐藏问题而不是帮助改进的。

Q：衡量自动化测试指标的问题，能否建立关于自动化ROI的指标，如何制定这个比例比较公平呢？

A：可能“公平”不是我们衡量的重点，ROI是体现投入产出是否合理的指标，虽然计算起来也较复杂，中间加入了人为设定的标量，但我们仍可以认为它是一个客观指标。要根据团队职能、技术条件、项目复杂度等因素差异调整你的指标，而不设定一个一百分的标准，但是可以要求他们不断改进ROI，要求工作效果不断提高，团队生产率不断提高。

Q：开发自动化测试脚本和执行手动测试比较起来，自动化价值不大，请问如何衡量自动化测试的收益？哪些是自动化测试的重点？

A：这也是我最早选择研究自动化测试ROI的一个初衷，而回归测试应该是自动化测试的重点之一。

Q：单元测试（Unit test）是自动化测试的一部分吗？

A：单元测试多用于开发工作，不论是测试驱动开发（TDD）还是单元测试，它们都将更早地发现缺陷，不至于让其遗留到软件开发后期。而关于自动化测试指标，还是与单元测试分割开来看比较合适。

Q：如何用自动化方式发现更多的缺陷？

A：经验告诉我很多问题发生在集成部分，每个组件的独立测试似乎都很完备，但是一旦

集成接口、数据、标准，各种问题就暴露出来了。在敏捷测试中，我会提倡在软件项目进展后期完成一个独立测试，以发现中间集成的问题。而自动化测试期望做好自动化BVT和问题解决后的回归，发现新问题的工作还需交给测试者自己。

Q：对于反复验证的问题是否需要自动化？

A：这点非常赞同，当我们发现某个问题多次被检查出来时，为了将测试者从程式化的重复性工作中解放出来去做真正有价值的工作，我们就应该用自动化的方法来解决。另外，当第三方代码出现问题时，我们通常期望第三方来修复，而这部分又需要通过软件才能验证，为避免沟通成本，我们也可以将自动化脚本发给第三方，让其修复后自行检查一遍。还有，当客户经常发现某一类问题而测试者没有发现时，这种测试就需要补充到原测试计划中，并最好实现自动化。

2. 有关文档的探讨

Q：敏捷就是少文档、去文档吗？

A：敏捷开发强调working software重于light document，然后便产生了两种甚至更多的理解，这两种极端的理解是：

1) 保持一定细节的文档，例如一个登录组件测试设计，只要写明测试一组有效的用户名和密码登录成功，并且测试另一组无效用户名和密码无法正常登录即可，而留给测试人员更多的空间在测试时选择“有效和无效”的数据。

2) 压根就不需要测试设计和测试用例，仅仅用用户故事（user story）就可以，因为如果需求都只是简单的记录，为什么希望测试用例是详细的呢？

这两种理解我认为不能完全推翻，都有合理的地方，而且第一种相对来讲要合理些，但两种极端的理解都必然会带来一些弊端，一个不完整的、模棱两可的测试设计是不能被执行的，你期望你的测试人员怎么工作呢？另外，如果对场景（scenario）的要求过粗，不同的人将采用不同的数据和方式进行测试，得到的结果将大相径庭。而且，这样的测试将给产品带来漏测的质量风险。

我建议的做法是light documenting而不是light document，传统的测试文档，工件（artifacts）可能过于繁多，但我们宁可有一个繁多的文档也不愿意缺少足够的信息。但是我们也希望文档的更新工作可以足够light weight，我们的精力应该放在真正核心的测试设计和增加测试覆盖率等工作上，而把文档的更新、组织交给计算机系统。现在Rational Publishing Engine可以帮助实现light documenting。

总之，在敏捷开发过程中我们仍然需要文档、需要存档、需要通过文本的形式做些必要的沟通。我们应该让工具来解放documenting工作，把宝贵时间投入到必要的事情上去。

3. 有关监控与管理的探讨

Q：传统团队通过审查监控的手段来确保项目的顺利运行，而敏捷团队更多的是依靠什么呢？

A：审查和监控的手段对于项目的顺利进行有帮助，这一点毋庸置疑，但是如果这是一个“垃圾产品”，为什么需要顺利进行呢，能不能不生产“垃圾”，让“垃圾”快速“失败”，让我们的有限生命投入到真正对社会有价值的项目中去呢？这就是敏捷的真谛。在敏捷项目里，我们很少需要“Manager”的监督和控制，因为员工可以做到三点：“自我组织、有自知之明、自律”。但我们关注产品本身传递的价值，产品负责人（Product Owner）需要“控制”项目的开发优先顺序，确定交付和开发的进度表，拍板决定何时能够发布产品。对于复杂的产品，如大型企业应用开发、大型用户群行业应用开发，则不仅仅是一个产品负责人和一个小团队能够

完成的，我们既要保障团队的敏捷能力，又要降低企业在项目中的投资风险，因而我们需要项目有足够的企业透明度，这就是IBM Rational敏捷的服务宗旨。

Q：如何对一个项目需求并不清晰的新系统进行敏捷需求管理呢？

A：敏捷需求管理可以分成简单和复杂两类，如果团队很小，可以集中管理，建议使用普通的“白板note便签”方式；如果项目不得不牵扯到更多的人、更多的团队，则建议你选择 Rational企业级应用，RTC/DOORS/RRC/CLM解决方案都可以帮助你驾驭敏捷需求管理。

Q：在顾客强调不能接受项目逾期、不能够接受项目超支的前提下，如何保持敏捷的鲜活呢？

A：没有人能够提前预估一个未知领域的产品是什么样、将来的市场在哪里、盈利点在哪里，也绝对不清楚我们需要花费多少时间才能完成最有价值、最重要的产品组件。我能够理解投资人从风险角度的考虑，如果客户在完全未知的因素里，还要勇敢地投入一笔资金，说实话，那样风险太大了。因此，无论如何，保守型的客户选择了固定项目周期，限制项目支出，甚至会干涉你的研发、管理、财务来确保这两项投入的精确或者最小化。但是，项目管理的基本元素——资源、范围、时间有着相互牵扯的金三角关系。即一旦任何一项发生变化都将带来其他两个维度的调整。因此，在顾客强调不能逾期或者项目超支的前提下，又需要保持足够的敏捷和项目需求变更的灵活，我们的客户需要接受一个事实，那就是项目的范围必须调整。而越是采用新技术、越是与第三方团队和个人产生依赖关系，项目范围的调整越可能超乎你的想象。

4.有关敏捷转型的探讨

Q：对于整个软件项目主线采用瀑布式开发，对于每个功能的开发采用敏捷开发是否可行？

A：对于你的问题，我想加一个问题，为什么要在局部采用敏捷开发呢？

Q：我比较习惯传统方式，在我的团队里应用敏捷开发，我希望从局部开始尝试，视其发展，再全面应用。

A：我也很喜欢传统模式，在我先按部就班做事情的时候，产品线固定，不期望创新和改变，或者没有新技术的引入，使用传统模式极具帮助，因为传统模式源于“制造业”。而敏捷开发的真谛是如何做到始终给客户传递最高价值的交付，从局部开始创新，得先保证我们的进度是可行的，我希望你不要考虑是否在用敏捷方法，而是考虑以下几个方面：如何让团队众志成城；如何将项目的透明度提高，即弱化Command&Control的传统方式带来的诟病；有效地使用敏捷方法中的具体实践解决你当下最为关心的问题。

Q：为什么使用敏捷开发模式？优点是什么？

我在很多场合说了敏捷思想、敏捷开发的优点，多从企业、组织角度出发，之前在微博里也多次论及。对于你的问题，我希望从人成长的角度来回答。很多人曾认为，敏捷是高大上的，不是一般人能够做好的。但是，从我在公司带领的多个开发、测试的敏捷项目和敏捷团队来看，这些说法未免言过其实。敏捷的成功的确和人才这个要素分不开，但是，任何企业的成功难道不都取决于人才吗？人才素质和职业修养左右着企业成本，直接影响着企业的运转效率，甚至决定着企业的存亡。但是，我们看到的敏捷成功案例、本书中说到的故事，对其人才的要求也不过三点：自我管理、自律和自知之明。做好敏捷教练，除了流程，更多的工作是需要用教练的方式培养人才，敏捷转型为人才的培养提供了良好的平台，而人的成长确实需要经验的累积，所以，我鼓励所有国内的企业都学习并推广敏捷模式。

Q：国外的理论是高手才能玩的东西，一般的团队是不行的，你们这是什么节奏？

国内企业的软件富强之路还很长，敏捷思想在国内的发展还很短暂。我参加过这几年的年度敏捷中国大会，看到过数以千计的敏捷粉丝和嘉宾，这些让我感到学习敏捷并不是抄袭国外的理论或者说敏捷只是高手的玩意儿，更不是只有大企业在谈。但是，的确在一些场合听到朋友们谈起敏捷实施遇挫的遭遇，有的是因为摸着石头过河，没有专业的教练和系统的考虑，导致中途而返；有的是没有学习敏捷的道和术，为了模仿敏捷而敏捷，因为一知半解而自己耽误了前程。我想提问者所指应该是后者。我是一名有着多年敏捷研发、测试、咨询、管理经验的实践者，从2006年开始做第一个敏捷项目，我已经练就得将敏捷思想深入骨髓了，我想将我的理解、经验和国内的朋友分享，确实想帮助那些想学习敏捷、想学习成功经验的国内企业正确地实践敏捷。

Q：敏捷团队容易过分关注眼前的短期目标，而忽视长期的战略目标。

A：这是一个陈述句，我却希望这是一个疑问句。目标应该是可度量、可抵达的，否则目标不成立。长期的目标在那里吗？我们要问问团队，这个目标，大家的理解清晰一致，可抵达吗？随着环境的变迁和行业趋势的发展，我们的长期目标是不断修正的。IBM的RTC产品就是使用敏捷开发的产品，我们的产品经理已经为产品规划了极致的未来，一种将体验、协同、过程融于一体的未来工具，我们因此做了非常高级但完全可能被取缔的发布计划。而为了最快将产品价值产生出来，我们为这个产品的长期计划增加了一些小周期的、目标明确的里程碑。经过客户和使用者对迭代发布的体验反馈，通过当下时刻对趋势的预测，我们将当下判断为最有价值的功能作为短期目标先实现。如果以上情景作为我们下结论前的一个前提，我认同提问者提出的观点。

5.有关实践的探讨

Q：在实际业务中，事先可能对需求框架没有理解，也就无法确定一个大的框架，这个问题该如何解决？

A：在Rational敏捷方案里，我们有两级敏捷计划。第一步，我们对预计的范围、时间/预算、资源会有个预估，叫做发布计划，这个计划实施之后误差会很大。预估是为了更好地和项目的上线沟通，以获得批准；而到了第二步，也就是产品清单展开细节，团队选择任务做迭代计划的时候，产品负责人一定要给出一个先后顺序，价值高的先做，价值低的后做，在保障价值前提下，技术风险高的先做，这样我们便有了迭代计划，迭代计划是一个具体的承诺，一旦承诺就需要付诸行动，确保完成，这里所有涉及的范围/时间/预算/资源都是具体和清晰的。但等到第一个迭代完成，客户反馈后，我们发现还未做过的特性或功能里有许多调整，那么产品负责人需要重新调整第二个迭代的产品清单优先级，团队又从新的产品清单里选择重要的工作放到第二个迭代，以此类推。

Q：我看到您介绍的那个全家出游的故事，他是按照a-b-c顺序的过程，但实际开发过程往往有些并行进行的开发，那么敏捷开发如何平衡顺序开发、并行开发呢？

A：这里的“并行开发”是个敏感词汇，平行如果是指一个人同时在做两个项目，这就不是敏捷开发了。我们允许将复杂项目拆分成多个独立的、可耦合的小单位，分给不同的团队，它们是可以并行开发的。

Q：产品清单达到所要求的功能确实时间会较长，且无法再拆分，如何解决？

A：其实，拆分产品清单并没有你想象的那么难，先别给自己设限。工作项既可以按照功能点拆分，还可以按技术、结构、过程拆分。原则上，迭代时间越短，单个工作项能够分配的最大时间越小。试着和开发者沟通，既要理解拆分工作给开发者带来的额外压力，又要讲清楚敏捷开发重在协同，1个人走100步，不如100个人走1步，拆分工作项是每个人都需要做的，设定一个团队成员、团队负责人都能接受的工作项最大粒度，做成团队规则，团队会接受的。

Q：还有一个问题，敏捷模式下是不是要求客户充分参与？

A：是，但这不容易做，在与客户沟通和交互时，对于项目层面来说，客户说了算，那么频繁接洽，请客户参与验收测试和“审阅&反馈”顺理成章；但是对于行业应用和产品开发，你面对的将是100个客户，IBM除了请以上客户参与外，我们还有一些客户项目和反馈机制，这个是我今年的工作重点。

Q：敏捷方法与传统方法的利弊？如何甄别何时何地需要采用敏捷而不是传统方法呢？

A：在纠结变化的利弊之前，我想分析下敏捷和瀑布，当你的客户要求你建一座大楼，蓝图、效果图都出来了，你为什么不用瀑布开发方法按部就班呢？可是当客户连自己要什么都不知道，比如一个方舟或者绝世地堡，此时，你还是得用敏捷，因为没法瀑布。这么说来，是否你对变化的利弊就不再纠结？

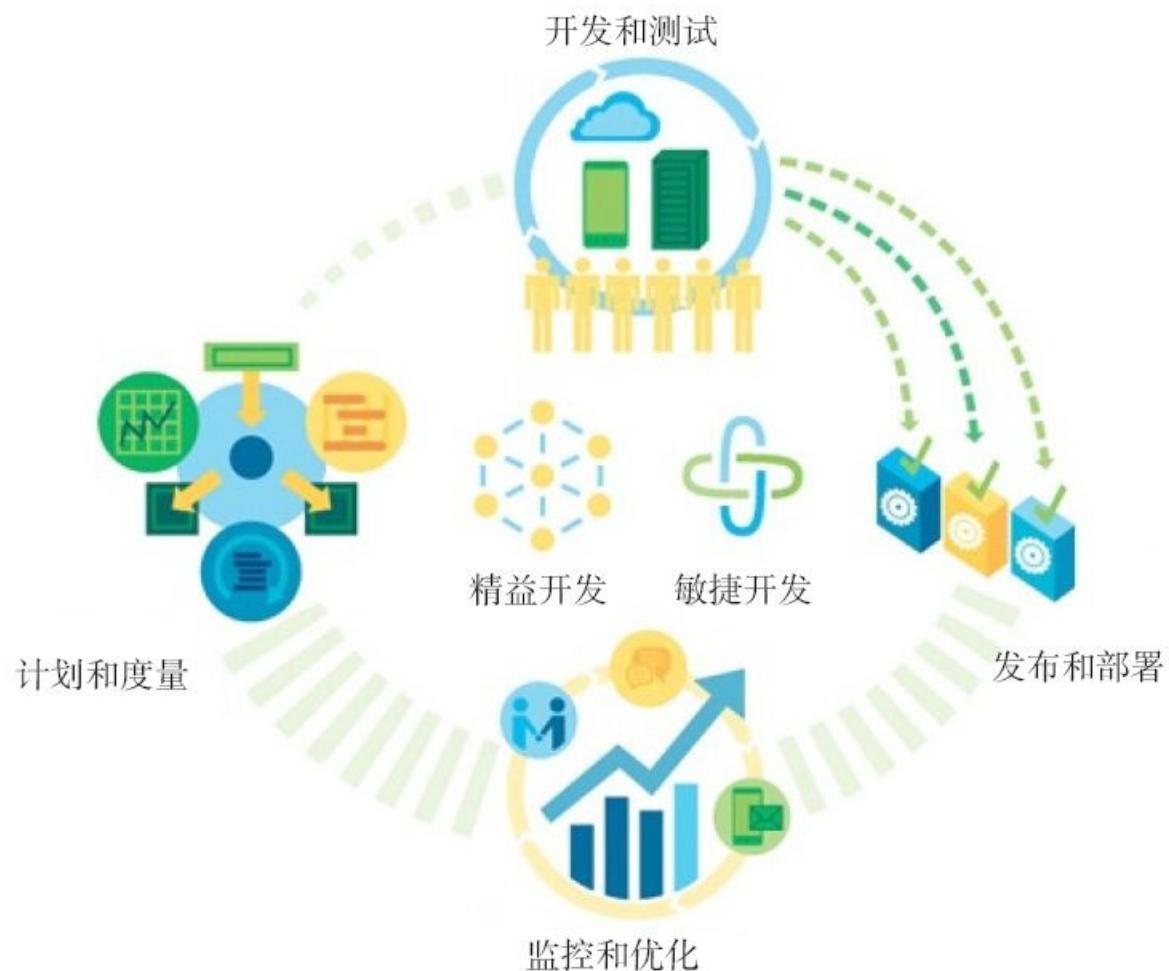
6.有关新技术的探讨

Q：敏捷开发如何应对云时代？

A：我感到云服务、企业移动、大数据和敏捷开发很快将成为一体方案，针对云与敏捷开发，在2009年~2011年的项目中，我自己所从事的开发和测试工作就已大量运用。在项目里，云提供了轻量级的环境构建、配置管理服务，将敏捷开发中的规模测试部分缩减了近90%的硬件和时间成本。

Q：敏捷开发的目标是提高开发效率，和现在的DevOps概念的区别大吗？

A：敏捷开发目标和DevOps的目标均是在加速从产品初步想法到最终价值交付的过程，并在过程中平衡投入、时间、质量和风险。而DevOps除了横跨产品研发的完整生命周期外，还涵盖了“交付”后的“部署、监控、优化”，见图A-1。



图A-1 DevOps贯穿产品的完整生命周期

附录B 第二代敏捷猜想

附录B 第二代敏捷猜想

我们正处在从传统IT面向知识IT转变的时代，当下的工业社会即使满负荷也不能够适应将来以信息化经济为核心的社会需求。而下一代的经济将是知识经济，下一代的IT就是知识IT、大数据、云计算、移动和社交领域的。

随着软件发展的提速，软件将逐渐取代部分硬件，例如小米电视、乐视电视这些设备将取代传统机顶盒，迎来第二代数字电视。而随着网速的稳步提高，计算终端已经可以简化到极致，所有的存储也可以放置到远程，“0”终端的设备露出水面。

B2B、B2C互联网电商时代黑马盛行，而下一代互联网经济将是C2B，即先有消费者需求产生而后才有企业组织生产。通常情况为消费者根据自身需求定制产品和价格，或主动参与产品设计、生产和定价，针对产品、价格等彰显消费者的个性化需求，生产企业进行定制化生产。长尾效应继续发挥，佼佼者胜利得益于客户中心化的商业模式。

可穿戴设备、移动应用广泛用于企业、汽车、医疗、家居，一切以“人”为重点，获取的“人”的感知、情绪、行为信息都将成为有价值的信息保存在云端。除了亚马逊、谷歌、微软、IBM等大厂商争相追逐企业云和私有云的市场以外，个人云的专项服务也会越来越多。未来5年的软件将不仅仅挑战硬件和服务器，软件还将虚拟出网络、存储、流量，几乎无所不及。

随着C2B的崛起、互联网思维的“极致”和为了满足“以人为本”的产品交付，敏捷研发自然是首选的方式。但在交付后，如何保障产品质量、安全部署和平稳交接则是需要思考的新问题。

特别是下一代语言的发展，软件企业的轻量级化，开源软件和免费软件的浪潮会使得原有企业管理软件不再是某些品牌一统天下，而是大集群，任何可以互联以及整合的软件只要是满足应用需求，都可以在网络、云平台上进行对接。而软件开发商可以是一个大型企业也可以是一个程序员。软件的生产环境进一步分布式，以获得更加优惠的地区补贴或者低廉的人力资源，小而精的团队之间需要通过开放的标准和合理的规则相互合作，如PMO，总线服务、开发、运维需要强强联结。而这些就意味着我们的产品必须有更加优化的研发过程，还要有更高的开发效率，尤其是APPs和互联网应用的开发需要更加快速地交付。

如图B-1所示，企业能否将敏捷软件交付能力作为自己的核心竞争力，将决定企业能否驾驭新技术时代。而云计算、移动、社交、大数据、可穿戴设备等产品的要求空前高涨也是导致企业交付软件的方式发生显著变化的原因，需要企业实现新标准的协作和创新。发展了多年的传统IT行业，和基于此建立的软件交付方法会经历更大的风险和挑战，固守围城将导致企业竞争优势的丧失，企业的生存也可能受到威胁。



图B-1 软件交付是所有技术趋势的核心

总之，新模式下仍然要持续解决的问题是：

- 外包开发50%的项目未能如期达到目标。
- 生产线生产和更新移动App或服务的时间太长，45%的企业生产延期。
- 面对快速的应用程序发布，系统的运维稳定性受冲击，70%的预算投入到运维中。
- 更快业务需求与后台的缓慢的开发测试差距加剧，即使是很小的移动App也需要花费4~6周时间完成开发测试工作。

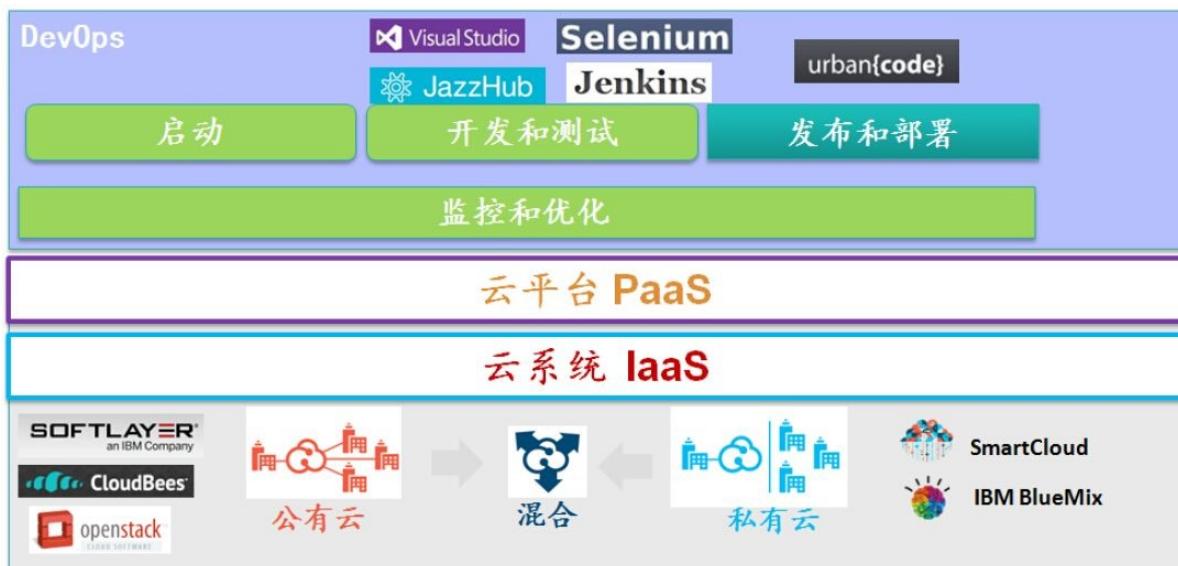
结果表明，目前的软件交付方式是需要改变的，即使是敏捷交付方法也需要改进，因为它们无法满足利益关系人或业务需要，同时缺乏持续的交付。特别是新需求引入后，或者外来系统整合现有ERP或者CRM系统时，方案的复杂性大大增加。而我们需要产品代码编译打包更加频繁，版本控制和环境部署、自动化测试、回归测试更加快速。这就是我们寄予厚望的第二代敏捷——DevOps，持续交付软件驱动的创新。

DevOps是连续的软件交付，可以帮助企业抢占市场先机，更迅速地对客户的反馈做出反应，并且平衡速度、成本、质量和风险。通过在整个软件交付生命周期应用精益和敏捷原则，DevOps帮助企业提供具有差异化和吸引力的客户体验，实现更快的时间值，提高创新能力。

事实上，在分析核心敏捷方法的局限性时，我们已经意识到DAD方法需要在运维上做得更好，即将开发测试与运维的通路打开，这将大大降低运维成本。以DAD来实现DevOps需要团队做出以下变革：

- DAD团队必须承认自己是一个更大的组织生态系统的一部分，他们需要与企业架构师和其他专业人士密切合作，利益关系人、运维和支持人员被明确列入DAD的解决方案。
 - DAD的目标是一种潜在的、可以消费的，而不仅仅是潜在可交付的产品或方案。
 - DAD团队要有大局观，除了关注产品和整个解决方案，还要关心硬件的更新，业务流程的变化，甚至组织变革，并能够在发生异常时治理生产运维环境。
 - DAD团队中进行监控和指导的高级管理人员及其利益关系人在营运环节被明确列入，他们参与了完整DAD生命周期的解决方案。
- 而DevOps必须依靠云计算技术，实现更快的持续化集成、测试和部署安装。特别是需要

将运维环境与生产环境打通，需要一系列的工具和平台来实现DevOps。或者说，下一代敏捷DevOps将打造云端工具链。图B-2给出了DevOps持续创新的示意图。



图B-2 DevOps持续创新

附录C 敏捷大赛评分规则

附录C 敏捷大赛评分规则

直到我了解到敏捷的真谛、学习敏捷方法并开展敏捷实践的时候，我才真正理解当年有个朋友的一句“玩笑”。当年我们在一个项目组，我刚刚进入，他是我的二线斯丹。一天他跟我们似玩笑又非玩笑地说：“艾利克斯，你知道什么是瞎忙吗？看看你现在，我每天只需要工作两个小时就能够打败你了……”

“我们现在从事敏捷交付、研发、测试，将自身的目标与团队的目标结合，可以自己做自己的主，计划和接受资源来完成任务，我们可以做到个人高效率的工作，避免了“瞎忙”，因为使用了高效的方法，我们的私人时间则可以用于生活和其他有意思的事情。”

这是一段我在一场敏捷大赛总结会上的致辞，我要向更多的朋友推广“敏捷”，不仅仅因为它是一种企业发展的必经过程，而更为重要的是，敏捷对于个人的成长极其有利。

2012年8月IBM软件技术峰会刚刚落下帷幕，IBM中国在国内开展了一系列技术比拼活动。主题之一的“敏捷开发”作为“寻找中国的好CTO敏捷大赛”的命题模式，要求参赛企业以真实业务场景中所面临的技术难题和实际业务需求为方案目标，且深入电信、银行、证券、保险、能源、医疗等六大行业，从企业中选择优秀的团队，并经过合理组合选出了20支完整的敏捷团队，包括产品负责人、团队负责人、团队成员等参与敏捷大赛。我接受了为最后一轮入围选手决赛的命题任务：既要考核团队对敏捷开发过程的理解，考核团队在实际操作中是否具有足够的灵敏度，又要检查团队在项目进行中是否满足了企业项目透明度的深度要求。

此次大赛旨在推进完整的IBM敏捷方案和敏捷实施体验，帮助团队深入了解IBM规范性敏捷的真正价值，体会敏捷开发所带来的多方面收益。严格要求在一个月的时间，也就是4周的4次迭代活动中完成项目，除了我发布了敏捷培训视频可作为入围选手的学习资料以外，最重要的是决赛的评分标准。

大赛将考核团队以及团队成员对敏捷发展理念的理解，驾驭工具、过程的实力，以及灵活面对挑战、解决问题的能力。所以，我将总评分规则设置为：

·工具。占总分值8%，JAZZ平台Team Concert是必选工具，只要正确使用了这个工具，用其做了过程要求的计划、拆分用户故事和任务、生成个人仪表盘就可以获得满分。而其他Rational需求管理、配置管理、质量管理工具均作为可选项。

·过程。占总分值65%，考核团队在软件交付完整生命流程中，对敏捷计划、用户故事、仪表盘和燃尽图，以及需求跟踪、版本控制、敏捷实践等方面的理解和应用。

·敏捷实践。占总分值22%，考核对敏捷实践活动的掌握程度，其中最重要的得分环节有：TDD、持续化集成、结对编程、回顾会议、计划、演示和迭代验收几个方面。

·面试。随机抽选一位团队成员，检查其对团队目标、迭代目标的正确理解。

·结果。占总分值10%，考核迭代里程碑是否有可工作的产品交付，是否增量型开发最终交付；通过提交敏捷迭代的中间和最终结果的抓屏图像和视频得分。

·灵敏度。占总分值17%，检查团队响应一次客户需求变更的速度；在约两周的时候，项目中一个正在开发的需求突然发生变化，检查应变能力。

总分构成

·工具 5分

·过程 41分（其中实践部分14分）

·结果 6分

·敏捷 11分

工具部分得分构成

·RTC的使用（必选） 5分

过程部分得分构成

·敏捷计划

·工作项评估（故事点/人时） 3分

·发布计划/迭代计划（两级计划） 3分

·产品工作清单视图 3分

·故事点拆分 3分

·仪表盘

·燃尽图 2分

·严重缺陷跟踪 2分

·新需求跟踪 2分

·版本控制 3分

·面试 6分

·实践

·测试驱动开发（非强制） 1分

·持续集成（非强制） 3分

·结对编程（非强制） 2分

·回顾会议 2分

·计划会议 3分

·演示 3分

结果部分得分构成

·迭代成果的抓屏（或视频） 6分

敏捷Agility show

·重新评估产品清单的优先顺序 5分

- 增加新工作项/删除旧工作项 3分
 - 第三周的演示中体现变化（可选） 3分
- 总计 63分

结束语

结束语

在我即将收笔之际，我服务10年的公司IBM也正在抛售旗下低端服务器业务，而世界范围内的金融危机和中国区的悲惨数字也导致公司裁员的流言四起。前些日子，甚至还看到圈子里朋友转的一则“一个IBM人的离职泪”，总之是各种唱衰。也罢，媒体总有猎奇的、胆大的言论，这些都不会影响我的判断和心情。这家公司在过去的百年中遭遇了多次致命的危机，但也通过其发人深省的变革重新站起来。

我必须诚实地说，IBM不是一家传统的大公司，相反，IBM是由成百上千持续创新的，为客户提供领先的硬件、软件、企业咨询和技术服务的中小型公司组成的集团公司。从校园迈入IBM直至现在的10年里，我认识到IBM有非常人性化的性格，也正是这些人性化的性格，才造就了IBM不可思议的成就。IBM的基本行为准则就是尊重个人，而任何人都不能违反这一准则，至少，没有人会承认他不尊重个人。毕竟在历史上许多文化与宗教戒律也一再呼吁尊重个人的权利与尊严。虽然几乎每个人都同意这个观念，但列入公司信条中的却很少见，更难说遵循。在IBM，所有的管理人员对公司里的任何员工都必须尊重，同时也希望每一位员工尊重顾客，即使对待竞争对手也应同等对待。

10年来，从工作伊始，无论在研发、测试、服务、支持还是管理团队与客户项目环节中，我都体会到IBM的特别训令：IBM是一家“顾客至上”的公司，也就是IBM的一举一动都要以顾客需要为前提。为了让顾客感觉到自己是多么重要，无论顾客有任何问题，一定在24小时内解决，如果不能立即解决，也会给予一个圆满的答复。在对待自己的工作、责任上，IBMer追求优异，对任何事物都保持追求最理想的理念，无论是产品或服务都要永远保持完美无缺。当然，完美无缺是永远不可能达到的，但是目标不能放低，否则整个计划都将受到影响。IBMer都努力争上游，每个人都认为任何有可能做到的事，都能做得到，这种态度让我的IBM生涯留下了许多令人振奋的故事。

所以，我希望这家公司可以继续续写辉煌，继续光耀神州。

我的抒情源自于IBM这10年身体力行的快乐、痛苦、坚持。也许我们头上的光环已不再耀眼，但我会谨记自己是一名曾有着纯粹蓝色梦想，在这个充满魅力的平台经历了公司最好年华的IBMer，我以成为IBMer为荣。这十年，公司给我的栽培和锻炼使我对其恩情没齿难忘。随着社交商务活动的愈发频繁，我也愈发感到我承载的使命和社会对我的需要，遂花费一整年的时间来完成这本书。本书以敏捷转型这一专业话题为主线，但是中间点滴注入了我对人生、处事的观点，对过往经验的总结，流露了对年轻人的告诫和期望。书中所描述的故事都是真实的故事，无论你能否记住，这些事情都已烙印在这本书里，这本书将是我对于已逝去的10年光阴的最好交代。而怎么想写这本书呢，让我最后讲给你听。

2012年，我经历了个人最大的蜕变，曾经将更高的职位、更多的薪水作为自己的目标，从来都是将生活和健康放在第二位，即使取得了更好的位置，却似乎并没有让我有多么大的愉悦。虽然一心向学，认真经营自己的职业发展路线，但是，和大多数人一样，我也曾遭遇过一些挫折、失败。有人说过我们终归慢慢学会“放下”，懂得自己需要什么的同时也清楚自己需要放弃什么。

而那年冬天，诸多优秀的企业也向我投来了橄榄枝。可是，当我人生中的一个贵人——来自美国的崔西——出现后，我才坚定了自己的信念。尽管其他机会的薪水和职位更加耀眼，但在和崔西的一次愉悦的面试后，我决定不离开公司，仅从原来的管理岗位调整到客户工作，开始了工作与生活平衡的日子。此刻，我放弃了某些事情的“执念”，转而“追求快乐、愉悦和内心的平和”，这成为了我至今的信仰。

我与许多人谈起过这个故事，我都信心满满地告诉他们通过这个选择，通过发觉自己的真正需求，我似乎打开了上帝给我的一扇窗，开启了我对“生命”的新的探索。事实上，我的确是通过这个选择从“象牙塔”的研发工作转向了更加“接地气”的客户工作。记得在我做这个决定之前的那个冬天假期，也就是2012年年底的一个早晨，当我从丽江的瑟瑟寒冷的清晨中醒来，踏上了湿漉漉的青石板路，一缕温暖的阳光撒向我的胸怀，我瞬间释怀了，我猜，我的心智在那一刻也成熟了。

意想不到的是，通过这个“接地气”的平台我走了出来，来到了这个现实而令人兴奋的大千世界（真是有这种相见恨晚的感觉）。我参加了CTO健走俱乐部、西山读书会等，不断拓宽了我的人际圈。就在我紧张地开始这份工作和兴奋于人脉慢慢变宽的时候，我也结识了各行各业的朋友，其中不乏精英和大人物。

在畅谈、小酌之间我忽然有了写作的欲望，为什么不坐下来写写东西，把以前的经验、知识分享出来呢？况且，再过两年，客户和市场活动更多了，我的心将更乱，我还能做研究吗？作家若莹（小说《西桥》的作者）是鼓励我写书的贵人，若没有若莹的鼓励，恐怕这本书到今天只是一个大纲……

终于，在“字里行间”书房的一角，我找到了安宁、静心的感觉。书房非常适合我写作，所以一有时间，整个周末我都会在那里伏案、奋笔疾书。心累了，就会找找有意思的读物，沏一杯清茶，伴随着耳机中一曲淡淡的音乐慢慢品味，让自己熔化在当下。不知不觉中这本书已自成体系，虽然文字不多，但是字字句句都是我的亲身经历、所思所想，期望你喜欢我的书。

如今只要一到周末和节假日，我通常是泡图书馆、咖啡厅，比如“字里行间”、“星巴克”和“太平洋咖啡”，我很快发现自己爱上了读书，一个月下来怎么也读了一两本，这可是我从事研发工作时想都不敢想的事情。随着出差在路上的时间越来越多，我也学会了捧一本书好好打发等候的时间。在完成这本书的同时我至少完成了40本图书的阅读，其中的收获不言而喻。

从研发部门跨越到今天的IBM客户工作，本身是很大的挑战，不少人在这个时候或许会选择一个新平台，而我没有。不是因为外面的诱惑不够多，也不是给的薪水不够高，而是我认为跳出去的职业路线只能越来越窄，我真心觉得还需要尝试一下客户工作，而且我认为自身还有一些不足有待改进。

我知道不少人都感兴趣跳槽这个问题。无论当时、还是现在，我所希望的工作环境是能够好好学习和提升自己的平台，并拥有高素质的工作伙伴，在大家的互惠互助之中感受幸福的工作氛围，并在平台上自然地发掘自身的潜力和展示个人价值，这一切，IBM当年都可以提供给我。同时，我深知，跳槽就失去了许多资源，你的新公司一定是因为你既有的能力而聘用你，所以你不能够选择性就业，只能做“老本行”。通常新公司不会考虑你的兴趣和发展，跳槽可以让薪水短期内飞跃，但是赌的是职业发展，尤其是干不到三年就跳槽，并且以不断跳槽来获取高薪水的朋友，其职业发展道路往往越来越“窄”。总而言之，我视就业如“嫁娶”，大家跳的时候需谨慎。

最后，10年的经验和历程也罢，成功、失败也罢，快乐、不快乐也罢，我们今天所正在经历的才是需要真正好好体会和珍惜的，因为错过了就不再有。无论你感受到人生的积极还是消极，你我均需要理解和接受，我们应该折服于曾经的苦，享用今天的一切。因为，生命中所有的事情、所有的人都是必然的，都是上帝为了教会我们一件事而使得我们必须经历的。

参考文献

参考文献

- [1]杰夫·戴尔, 等.创新者的基因[M].曾佳宁, 译.北京 : 中信出版社, 2013.
- [2]艾森·拉塞尔.麦肯锡方法[M].张薇薇, 译.北京 : 机械工业出版社, 2010.
- [3]肯尼思·法尔曼, 等.色彩物语 : 影响力的秘密.[M].谢康, 等译.2版.北京 : 人民邮电出版社, 2012.
- [4]史蒂夫·布兰克, 等.创业者手册[M].新华商都学院, 译.北京 : 机械工业出版社, 2013.
- [5]James Bach.A Framework for Good Enough Testing[J], IEEE computer society, 1998.
- [6]James B Michael, Bernard J Bossuyt, Snyder B B.Metrics for Measuring the Effectiveness of Software-Testing Tools[J].Software Reliability Engineering, IEEE, 2002.
- [7]Daniel Rowley, Dr Sita Ramakrishnan.ATTEST : an Automated-Test-Tool Evaluation and Selection Technology.
- [8]彼得·法德.沃顿管理精要[M].邓峰, 译.北京 : 中信出版社, 2013.
- [9]Louis Cauffman.The Solution Tango : Seven Simple Steps to Solutions in Management[M].Marshall Cavendish Limited, 2006.
- [10]加里·哈默, 等.管理大未来[M].陈劲, 译.北京 : 中信出版社, 2012.
- [11]塞·查耐.世界500强工作规范[M].郑海娟, 译.北京 : 世界文化出版公司, 2004.
- [12]木村英一.零压力工作法[M].汪婷, 译.北京 : 电子工业出版社, 2014.
- [13]2008IT Project Success Rates Survey Results[EB/OL].<http://www.ambyssoft.com/surveys/success2008.html>.
- [14]Manifesto for Agile Software Development[EB/OL].<http://www.agilemanifesto.org>.
- [15]Modeling and Documentation Practices on IT Projects Survey Results : July 2008[EB/OL].<http://www.ambyssoft.com/surveys/modelingDocumentation2008.html>.
- [16]Poppendieck M, Poppendieck T.Implementing Lean Software Development : From Concept to Cash[M].Boston : Addison Wesley, 2006.
- [17]The Agile System Development Life Cycle (SDLC) [EB/OL].<http://www.ambyssoft.com/essays/agileLifecycle.html>.
- [18]Ambler S W.The Object Primer : Agile Model Driven Development with UML 2.0[M].3rd ed.New York : Cambridge University Press, 2004.
- [19]Ambler S W.Agile Modeling : Effective Practices for Extreme Programming and the Unified Process[M].New York : Wiley Press, 2002.
- [20]Kroll P, Macisaac B.Agility and Discipline Made Easy : Practices from OpenUP and

RUP[M].Boston : Addison-Wesley, 2006.

[21]Information Technology Infrastructure Library (ITIL) Official Site[EB/OL].<http://www.itil-officialsite.com>.

[22]IBM Tivoli Unified Process (ITUP)[EB/OL].<http://www.ibm.com/software/tivoli/governance/servicemanagement/itup/tool.html>.

[23]Fowler M.Refactoring : Improving the Design of Existing Code.Menlo Park[M].CA : Addison-Wesley Longman, 1999.

[24]Ambler S M, Sadlage.Refactoring Databases : Evolutionary Database Design[M].Upper Saddle River, NJ : Addison Wesley, 2006.

[25]Measured Capability Improvement Framework (MCIF)[EB/OL].<http://www.ibm.com/software/rational/mcif/>.

[26]Ambler S M, etc.The Enterprise Unified Process : Enhancing the Rational Unified Process[M].Boston : Addison Wesley, 2004.

[27]Ambler.Product Backlogs at Scale, 2009[EB/OL].<http://www.ddj.com/architect/217701202>.

[28]Kroll P, Krebs W.Introducing IBM Rational Self Check for Software Teams, 2008[EB/OL].http://www.ibm.com/developerworks/rational/library/edge/08/may08/kroll_krebs/index.

[29]Stober T, Hansmann W.Agile Software Development : Best Practices for Large Software Development Projects[M].New York : Springer Publishing, 2010.

[30]Larman C, Vodde B.Scaling Lean&Agile Development : Thinking and Organizational Tools for Large-Scale Scrum[M].Upper Saddle River, NJ : Addison Wesley, 2009.

[31]Kessler C, Sweitzer J.Outside-In Software Development : A Practical Approach to Building Stakeholder-Based Products[M].IBM Press, 2007.

[32]Pixon P, ect.Stand Back and Deliver : Accelerating Business Agility[M].Upper Saddle River, NJ : Addison Wesley, 2009.

[33]Ambler S.W.Agile Database Techniques : Effective Strategies for the Agile Development[M].New York : Wiley Publishing, 2004.

[34]Beck K.Extreme Programming Explained-Embrace Change.MA : Addison Wesley Longman, Inc, 2000.

[35]Palmer S R, Felsing J M.A Practical Guide to Feature-Driven Development[M].Upper Saddle River, NJ : Prentice Hall PTR, 2002.

[36]Beedle M, Schwaber K.Agile Software Development With Scrum[M].Upper Saddle River, New Jersey : Prentice Hall, Inc, 2001.

[37]Ambler S W.The Agile Unified Process, 2005[EB/OL].<http://www.ambyssoft.com/unifiedprocess/agileUP.html>.

[38]Velzen T.Skillful and Maneuverable : OpenUP and the Eclipse Way, 2008[EB/OL].<http://www.ibm.com/developerworks/rational/library/edge/08/jul08/vanVelzen/>.

[39]Duvall P, Matyas S.Continuous Integration : Improving Software Quality and Reducing

Risk[M].Upper Saddle River, NJ : Addison Wesley, 2007.

[40]Kruchten P.The Context of Software Development,
2009[EB/OL].<http://www.ddj.com/architect/201804241>.