DOI:10.16080/j.issn1671-833x.2011.10.016

敏捷方法在软件开发过程中的实践

Practice of Software Development Process With Agile Method

中航工业北京航空制造工程研究所 105 室　　隋立江

[摘要] 为提高航空制造企业内部软件开发团队的工作效率，根据这类企业保密要求严等实际情况，采用敏捷开发方法和传统方法相结合的思路，围绕特定项目，经过近三年的探索实践，总结了一些自下而上自发进行软件过程改进的经验，找到了一种适合这类企业内部软件开发的过程模型。

关键词：软件过程改进　敏捷方法　极限编程

[ABSTRACT] In order to improve the efficiency of the software development team within aeronautical manufacturing enterprises which has strict secrecy systems, agile methods and classic software develop process in a concrete project is adopted. After three years' practice, some experiences about how to improve our software process in a bottom-up way is obtained and a software develop process model for this kind of enterprises is found.

Keywords: Software process improvement (SPI) Agile method eXtreme Programming (XP)

我国的航空制造业企业数字化工作在国内各行业中起步较早，通过与国外的交流发现，各种软件工具对提高管理水平和生产效率有着重大的促进作用，因此各工厂研究所都在比较充分地使用各种不同类型不同层次的软件产品，这些软件的普遍特点是与应用企业的实际业务流程密切相关，尤其是在航空制造业中，其普遍使用的大型软件（如 CATIA、DELMIA 和 Winchill 等）的各种模块及相关关系大多是根据美国波音、洛克希德马丁、欧洲空客等公司的情况开发并调整的，国内的航空制造企业的特点与上述公司有比较明显的差异，这导致很多软件的利用率不是很高。一方面由于通用软件的自身特点所限，不能很好地直接满足企业的要求，另一方面，即使软件厂商愿意进行客户化定制，应用企业又不愿意在定制过程中大量公开自己的业务流程等信息，在这种情况下，各厂所的信息化部门（如信息中心、计算中心等）发挥了重大作用，他们熟悉自己的流程，又具有对相关软件的二次开发能力，他们的工作为提高厂所的工作效率作出了很大贡献。这些信息化部门的开发团队一般人数不多，乍看起来，这种团队和一般的小型软件开发企业并无太大差异，但是由于依附于大企业内部，开发团队的管理制度、薪酬机制、职业发展等受所在企业的制约很大，不像一般软件企业那样灵活、有针对性。而且目前国内的航空制造业企业出于安全等方面的考虑，对于保密制度、6S 制度方面有着比较严格的要求，对网络使用、项目管理办法、工作现场布置等方面都有严格的规定。这些限制性因素与目前流行的软件开发管理思想有一定的矛盾，在这样的组织里进行软件过程的改进时直接套用那些完整的、成熟的软件过程改进方法并不合适。

1　软件过程改进的思路与方法

作者所在的团队是航空制造业企业内部的开发组织，为企业提供通用软件的定制二次开发和一些管理信息系统的开发，相关产品在本企业及行业内部分厂所应用。随着企业的飞速发展，开发工作迅速增多，开发内容日渐复杂，开发人员频繁调整，以往粗放的开发模式遇到了很大的挑战，一些常见的软件工程方面的问题开始浮现，开发进度屡屡拖延，产品质量难以保证等，过程改进有了迫切的需求。在软件过程改进过程中来自人的阻力非常大，无论是中间管理层还是一线开发人员，这里面既有打破传统习惯带来的不便，也有既得利益者的损失，而且软件过程改进初期肯定会对工作效率、业绩等产生一定的影响，因此，必须认真分析各种过程模型的特点，结合自身团队的条件，选择并裁剪出适合的改进方法和路线，只有这样软件改进过程才有可能取得令人满意的成果。

1.1　常见软件过程模型

能力成熟度模型（Capability Maturity Model，CMM）是 1987 年由美国卡内基梅隆大学软件工程研究所按照美国国防部的要求，推出的评估软件能力与成熟度的一套标准，2002 年正式发布 CMMI（Capability Maturity Model Integration），CMMI 模型如图 1 所示。CMM/CMMI 建立在“好的软件过程是产生好的软件质量”的假设前提下，通过改进软件过程达到最终提高软件质量的目的，是目前国际上最流行也是最实用的一种软件生产过程标准和软件企业成熟度评估标准。该标准基于众多软件专家的实践经验，侧重于软件开发过程的管理及工程能力的提高与评估，是对软件过程进行评价和改进的

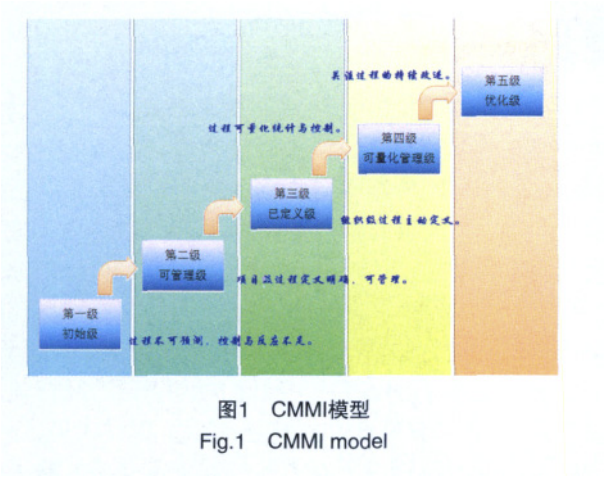

图1 CMMI模型
Fig.1 CMMI model

行动指南。国内相关部门和学者制订的GJB5000/5000A标准与之类似，经过在不同行业的多期的推广工作，取得了一定成果，为国内涉军软件开发组织的软件过程改进提供了指引。CMM/CMMI只提供了一个庞大的目录，详细地指出了进行软件过程改进时要在哪些方面入手，要完成哪些目标，要达到什么样的状态，但是并没有说明要如何做才能实现，没有办法作为直接的参考。

相比较而言，Rational统一过程(Rational United Process，RUP)提供了很容易直接应用的模型[1]。RUP最初由Ivar Jacobson等人设计，由原Rational 公司(现属于IBM公司)开发的实现统一过程(United Process，UP)方法的一个很成功的商业化的产品。RUP模型如图2所示。RUP提供了详细的文档、表格等资料，覆盖软件全生命周期的方方面面，使用者几乎只要照着做就可以了。但是也正是由于这种全面性，直接使用RUP指导软件过程改进所需的额外的工作是非常多的，在没有经验丰富的咨询师的帮助下又很难进行有效合理的裁剪，这很显然超过了一般小型开发团队的承受能力。

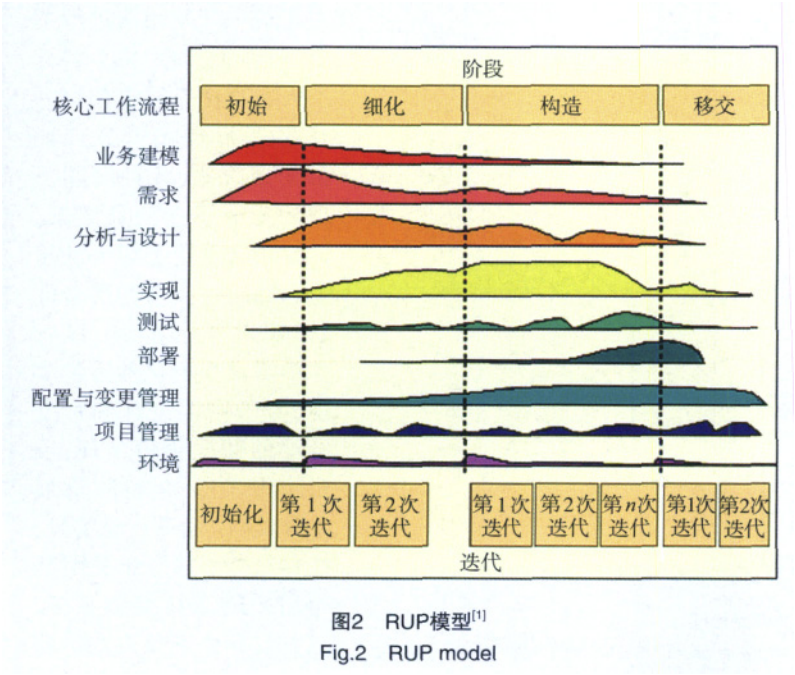

图2 RUP模型[1]
Fig.2 RUP model

其他诸如微软解决方案框架(Microsoft Solution Framework，MSF)、团队软件过程/个人软件过程(Team Software Process/Personal Software Process，TSP/PSP)等方法也都因为文档繁杂或者过程变化太大等原因没能得到团队的认可。在对这些方法的学习和了解过程中，尤其是通过学习CMM/CMMI和RUP相关案例，整个团队对于并行、迭代、版本控制等概念有了比较明确的认识，并产生了很强烈的在工作中进行尝试的愿望。

1.2 敏捷方法

敏捷方法(Agile Method)是一系列以快捷、轻便的思维方式面对各种变化的新软件工程思想的统称，其中包括极限编程(eXtreme Programming，XP)、Scrum、水晶方法(Crystal Methodologies)等。其中，极限编程是由Kent Beck于1999年正式发布的一种“轻量型”的软件开发方法，是众多敏捷方法中最具代表性、最为流行的一种。极限编程是一种轻量、高效、低风险、柔性、可预测、科学而充满乐趣的软件开发方式[2]。极限编程中有独具特色的核心价值观：沟通(Communication)、简单(Simplicity)、反馈(Feedback)和勇气(Courage)，很适合小型团队在面临很多不确定因素的情况下进行软件开发时采用[3]，其中12个最佳实践(Best practices)无疑是最吸引人的部分。这12个最佳实践都是前人经验的总结，我们从中选择了容易理解、方便实现的几个进行了尝试，在项目的开发中取得了不错的效果，进一步增强了大家主动进行过程改进的愿望和决心。

12个最佳实践包括：

- 规划制订(Planning Game)
- 系统隐喻(System Metaphor)
- 简单设计(Simple Design)
- 结对编程(Pair Programming)
- 测试驱动开发(Test Driven Development)
- 持续集成(Continuous Integration)
- 重构 (Refactoring)
- 小版本发布 (Small Release)
- 共同拥有代码(Collective Code Ownership)
- 现场客户(On-site Customer)
- 编码规范(Coding Standard)
- 40h工作制 (40-hour Week)

2 软件过程改进实践与经验

开发团队首先对现代软件工程相关理论进行了比较全面的学习，搜集并分析了大量成

功和失败的案例。团队内部统一了认识，根据自身特点对软件过程改进模型进行了比较和裁剪，确定了在不同的组织结构、制度产生较大影响的前提下，自下而上、先易后难的改进原则，征得了管理层的认可，在某软件产品的开发过程中软件进行了初步的过程改进实践。

该软件产品是某基础研究项目的最终成果之一，同时还是该项目研究过程中的重要工具。在这种开发模式中，需求一直在逐渐进行调整，直到项目的最后阶段才能完全确定，但软件产品需要在项目初期就提供部分功能给项目中的其他团队使用。传统的开发模型明显不能适应这种要求。根据软件过程改进领域的现有成果，软件开发团队大胆地引入了在 CMM 和 RUP 中都有所体现的迭代式开发。除了在需求、设计、实现、测试和发布各个阶段中进行迭代之外，还在这几个阶段之间进行了迭代，使得软件产品具有了足够的可扩展性，并确保了阶段性可用版本的及时发布，改进后的开发模型如图 3 所示。

在过程改进实践中，开发团队充分感受到了在当前这种管理模式下使用灵活极限编程最佳实践的阻力和困难，由于场地、资金等各方面的限制以及涉军项目对节点和阶段性成果的严格要求，不得不对改进内容进行反复调整和删减，尽管有些内容还是一些经典极限编程指导资料中强烈要求进行的，我们还是决定在尽可能确保项目顺利进行的情况下，实事求是地运用相关工具，力所能及地进行过程改进。经过 3 年的开发过程，项目顺利完成，相关软件产品完整地展示了项目成果，而且为项目其他团队提供了有力的技术手段支持。除了得到软件产品外，开发团队还初步掌握了在传统软件开发流程中引入极限编程方法中若干最佳实践的方法，形成了有自己风格的软件开发过程模型。以下分别介绍几个典型的敏捷方法最佳实践在实际应用中的经验和教训。

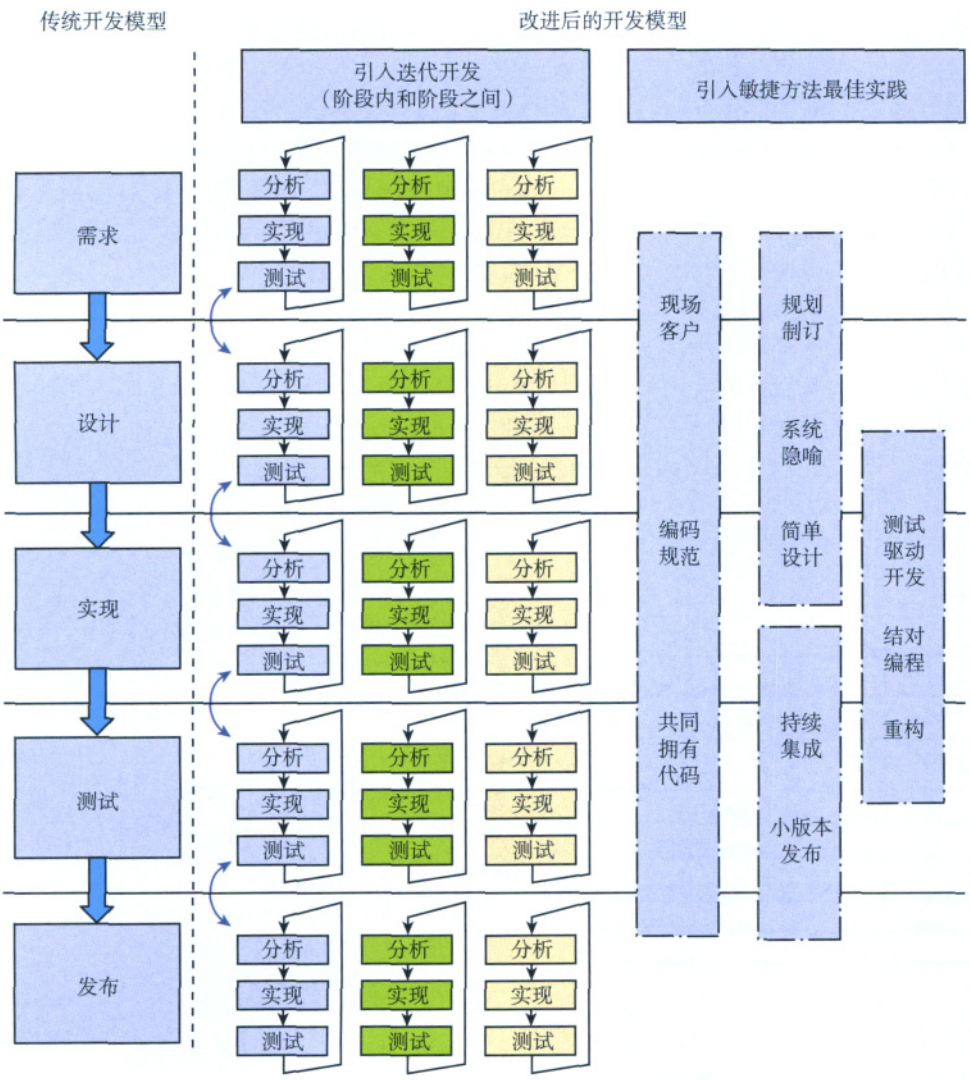

图3 具体实践中经过改进的开发模型

Fig.3 Improved development model in practice

2.1 现场客户(On-site Customer)

“现场客户”强调了在整个软件开发过程中客户参与的作用，包括持续地对需求的确认、对中间版本的使用反馈等。此次项目的合作双方在同一个大企业内，物理距离不远，基于这种内部开发的特点，在实践中开发团队进行了更大胆的尝试：将 2 名开发人员直接安排在用户办公室进行现场开发。项目初始阶段，各种问题出现的十分迅速而且繁杂，开发团队承受了很大的压力。经历过痛苦的磨合期后，效果开始显现：尽管各种需求变更仍然经常出现，由于错误设计和过度设计的情况很少出现，项目少走了很多弯路，进展十分顺利。需要注意的是，在传统模式下，开发团队与用户的直接接触不多，双方仅有个别负责人员在

为数不多的若干次讨论、确认以及评审会议中直接面对面交流。而运用“现场客户”这一方法时，双方要在日常工作中长时间地合作交流，尤其是最前沿的开发人员与最直接用户的互动很多，双方在企业文化差异以及个人行为等非技术领域产生的各种问题和矛盾不可避免，解决不好会对项目的顺利进行产生巨大影响。双方都应该做好足够的精神上和制度上的准备，避免产生不必要的纠纷。这一潜在的困难在之前的资料中强调的并不多，但是在实际应用中，尤其是国内航空制造业企业内部确实要注意这个问题。

2.2 编码规范(Coding Standard)

这一最佳实践的重要性和意义很容易理解，开发团队在一开始就接受了这个观点，结合以往的工作，并参考一些公开的编码规范，经过多次讨论和征求意见，制订了符合我们习惯的编码规范，分为 C++ 版和 Java 版两种。但是在实践中遇到了很大挑战，在实际编程过程中，由于习惯的作用，不同的开发人员在开始阶段对编码规范执行的力度明显不同。因为同时在实践“持续集成”和“小版本发布”，在集成发布中经历过几次因为书写习惯不同而产生错误且很难追踪之后，执行编码规范的认真程度大大提高。编码规范在增强程序可读性以及改善遗留系统可修改性方面的好处非常明显，以至于在随后的另外几个项目的开发工作中尽管没有强制规定，大家都能主动地按照这一规范编写代码。在整理制定代码规范时一定要充分考虑到即将使用这些规范的团队的习惯，从而减少推广阻力。

2.3 共同拥有代码(Collective Code Ownership)

在学习了 CMM/CMMI 相关知识后，开发团队一直使用开源工具 Subversion 进行了源代码版本控制，并在随后的改进过程中引入了“共同拥有代码”这一最佳实践。当项目规模不大，开发团队人员少，每个开发人员都要承担多种不同任务时，实施“共同拥有代码”并辅以“编码规范”，互相配合修改代码会变得容易，项目的人员调配也比以前灵活了很多。但是在进行保密相关技术改造后，由于开发团队成员密级不同，这一最佳实践的使用受到了很大的限制。

2.4 测试驱动开发(Test Driven Development)

当前比较流行的各种软件过程方法论都很强调测试的重要性，从最基本的单元，测试到最后的验收测试，测试工作贯穿整个软件开发过程始终。但在该项目的开发之前，团队长期使用传统的开发过程，形成了先写代码后统一测试的习惯，缺乏相关测试的经验和工具。在软件过程改进实践中没有强制进行过多的先行测试，而是由开发人员自行决定使用的范围和力度。

2.5 其他实践

规划制订(Planning Game)、结对编程(Pair programming)、重构(Refactoring)、隐喻(Metaphor)、简单设计(Simple Design)、小版本发布(Small Release)、持续集成(Continuous Integration)和 40 h工作制(40-hour Week)是极限编程中推荐的其他几个最佳实践，我们在项目进程中由个别开发人员分别进行过尝试，大部分结果还不错，但也有一些人不适应，需要做更多的调整。

软件过程改进要做到以人为本，让开发人员在改进过程中逐步认识到以前工作模式中的不足从而自发地进行改进，远比强制他们按照某种“经典”或“传统”的标准执行要有意义得多。开发人员才是软件开发过程中最具有能动性的因素，充分调动和发挥人的积极性才能产生真正好的产品和服务，这才是进行软件过程改进所要达到的根本目的。

2.6 实践效果

根据所开发的项目和开发团队的特点来看，CMM/CMMI 和 RUP 尽管是关于软件过程改进比较全面和成熟的理论，但实施成本比较高，而且其理念与传统企业的冲突比较明显，不利于得到组织的支持。而在传统开发模型基础上，在各开发阶段内增加迭代并允许在阶段之间进行迭代，在具体开发过程中引入比较容易进行的敏捷方法，这样能够在对现在的工作模式变动不大的情况下明显提高工作效率。我们的项目尽管需求变化频繁，仍然按时优质地完成了软件开发工作，确保了项目顺利完成，取得了比较明显的效果，从而得到了上级领导对继续推动过程改进工作的支持。

3 结束语

航空制造业企业内部的软件开发团队往往已经具有多年的开发经验，已经基本适应了传统的软件开发模式。在当前这种需求快速多变的环境下，这样的开发团队可以通过类似方法，将敏捷方法的最佳实践灵活运用到现有开发过程中，对软件过程进行自下而上的改进。这样做一方面能够提高开发团队对先进软件工程理论的认识，一方面可以以真实效果逐渐在制度和资源上获得企业对软件过程改进的支持，从而最终实现增强团队的实力、提高开发效率、保证产品质量的目的。

参 考 文 献

[1] Ivar Jacobson,Grady Booch,James Rumbaugh著. 统一软件开发过程. 周伯生,冯学敏,樊东平译. 北京:机械工业出版社,2002.10.

[2] Kent Beck. Extreme programming explained: embrace change.2nd ed. Addison-Wesley Professional,2004.

[3] Stephen R Schach著. 面向对象与传统软件工程.6th ed 韩松译. 北京：机械工业出版社，2006. 40-42.

（责编 泰山）