敏捷软件开发知识体系

AGILE DEVELOPMENT BODY OF KNOWLEDGE

2011年08月

编委简介

编写组：

组　长：

宁德军

任职于IBM，担任IBM Rational大中国区技术总监、中国石油大学兼职教授、中国敏捷联盟副主席。有超过15年的产品及项目管理和软件工程经验，先后在上海贝尔阿尔卡特比利时研发中心、IBM工作，拥有丰富的跨国项目和跨国公司产品管理经验。曾为华为、中兴、爱立信、腾讯、上海电力、工商银行、交通银行、中国银行等数十家企业提供咨询和培训服务。目前专注于产品及项目组合管理、敏捷开发过程和企业架构等新技术的研究。

副组长：

李春林

任职于东软集团，担任过程改善中心副主任。中国敏捷软件开发联盟副秘书长，资深过程改善顾问，MBA，CSM，A-SPICE Provisional Assessor。1999年加入中国最大的软件解决方案及服务提供商东软集团，拥有12年软件开发和过程改善经验。先后从事嵌入式软件系统研发、测试和项目管理工作，后专职从事过程改善工作，参与了东软集团质量体系文件的编写，曾作为评估组主要成员在2002年和2004年两次实施了CMMI v1.1 5级评估，并作为管理者领导了东软集团2007年和2010年的CMMI v1.2 5级评估。

张 忠

任职于用友软件股份有限公司，1995年开始从事软件开发工作，担任集团开发管理部总经理。目前关注于适合中国软件研发特点的“高效、敏捷、可度量、提升市场和用户价值”的研发最佳实践与研发管理体系，并在2011年提出软件的效益化研发，以期促进本土企业的研发管理创新。

编委成员（按姓氏首字母排序）：

陈志波

陈志波博士目前是Technicolor中国研究院多媒体实验室主任，视频处理/编码/媒体质量分析领域的专家，国际电气与电子工程师学会(IEEE)多媒体技术委员会成员，并是一些国际多媒体会议的组织委员会和程序委员会成员。作为公司首先启动敏捷式研究管理的项目负责人，有四年以上的利用敏捷式(Agile)管理流程管理研究和创新团队的经验。

高　航

任职于用友医疗卫生信息系统有限公司，担任G应用开发部开发经理。从事软件开发5年，精通JAVA系列技术，熟悉Delphi技术。在社保和医疗行业有着丰富的业务建模和系统架构经验。目前专注于软件研发团队的管理、软件研发流程的工具化实践与优化，并积极探索敏捷化开发在工程实践中的应用。

黄　方

任职于Electronic Arts上海公司，担任ScrumMaster/Project Manager。CSP,CSD, CSPO, CSM, PMP，十二年IT工作经验，七年传统项目管理经验，三年敏捷项目管理经验，带领多个Scrum团队从事游戏开发工作。对于Scrum框架，工程实践，全面质量管理和团队Coaching有深刻的理解和实践经验。

刘德意

任职于特艺（中国）科技有限公司（Technicolor China），担任北京研究院质量与项目管理部经理。十年以上软件开发及项目管理经验，自2002年逐渐转入质量管理、过程改进。先后帮助所在公司通过CMM-2、CMMI-3的正式评估，以及公司内部评估。从2007年开始，在Technicolor中国的北京研究院推行Scrum，并逐渐走入正轨。通过多年实践，深信过程改进要以人为本，讲求实效。本人是Agile的积极参与者，曾在2010年的Agile Tour Beijing活动中演讲。

刘曙光

任职于广州畅盟信息科技有限公司，担任IT部门技术总监，有10多年IT行业的工作经验，对软件工程及其技术服务有着深厚的理解和认识，曾为电信、电力等行业多个客户提供软件工程咨询及技术支持服务。目前主要专注于ALM和自动化测试方向。

庞建荣

特艺（中国）科技有限公司机顶盒事业部R&D Office Leader，14年软件开发及项目管理经验，8年质量管理及过程改进经验，曾在大型石油化工行业、移动通讯行业及消费电子产业供职，先后从事软件开发、项目管理、质量保证、过程改进等工作，参与企业的CMM评估、ISO9000内审外审，认证的ISO内审员，、CMM内部评估员、CSM。

钱　岭

任职于中国移动通信有限公司，担任研究院资深研究员。高级工程师，2001年1月毕业于清华大学计算机课学与技术系，获得工学博士学位，主修软件工程方向。毕业后加入贝尔实验室基础科学研究院，参与并负责包括软件质量管理、网络设备、VAS等产品研发。2007年12月加入中国移动通信研究院，历任广告相关项目经理、云计算HugeTable项目经理、大云项目架构师、大云产品研发负责人。在软件工程方法、敏捷开发方法和实践、工具和实践、基于CMMI的过程改进、软件质量管理、软件度量等领域有较多的研究和实践工作。

吴文龙

任职于IBM，担任软件部资深技术顾问。从事于Rational品牌产品线的售前技术支持和咨询顾问工作。长期致力于软件工程领域，具有七年以上软件应用生命周期管理解决方案和产品的售前和咨询工作经验。对于软件工程领域的技术、方法和工具产品具有深入的研究，并具有丰富的项目咨询实施经验。

。

邢　雷

任职于东软集团股份有限公司过程改善中心，任咨询顾问。近十年行业经验，海外工作三年后回国投身于软件项目管理及过程改善工作。曾作为核心人员负责所在组织CMMI五级的过程改善和最终评估工作。目前致力于东软集团的敏捷技术研究、部署等相关工作。

许舟平

任职于IBM，担任软件部Rational高级技术顾问。多年敏捷开发和项目管理经验。致力于敏捷开发在中国软件开发中的推广与实践。《敏捷无敌》一书的合作者。

闫建伟

任职于用友软件股份有限公司，担任银行客户事业部开发经理。多年从事银行业务IT系统的建设工作，具有丰富的银行专业知识和IT系统架构经验。

袁　斌

任职于北京迅思威尔科技有限公司，担任敏捷咨询事业部资深敏捷咨询顾问。工学硕士、MBA。Scrum、AUP、Agile modeling、XP、kanban等的实践者，资深敏捷咨询顾问。15年中就职于全球性公司从事软件和产品的开发。曾任Anoto产品中国区开发总监和Mino中国区软件开发总监，超过8年通讯电信、离岸外包以及互联网产品等多个行业的敏捷实践经验。”落地敏捷”社区的创建人，与超过500名敏捷实践者在社区共同讨论敏捷在国内企业中如何落地。

张克强

任职于上海宝信软件股份有限公司，担任宝信软件技术中心项目总监。系统分析师，Certified Scrum Master，硕士。现在是负责领导咨询团队把业界最佳实践和自身有效实践应用到宝信的各个研发单元中，并组织和协调各单元的过程改进评估和审核。他曾经在Intel的Christea团队担当质量保证经理。在软件工程方面拥有9年经验，在过程改进、质量保证和测试方面有丰富的实践。帮助组织按CMM/CMMI模型进行改进，并通过了CMM4、CMM5、CMMI5的评估，从2007年起为宝信软件引入了敏捷方法，在CMMI的框架下推进敏捷实践，在软件质量和开发效率方面都获得了明显的改善。他参与了2010年中国敏捷发展报告的书写。

赵　静

任职于IBM，担任软件部高级技术顾问。目前主要关注Rational统一开发流程与平台、软件项目配置管理、项目开发管理、项目管理及项目组合管理等。曾经为多个通讯公司、银行、世界500强企业、政府相关部门提供过开发过程管理、项目管理、软件配置管理等咨询服务。

宣传组：

刘　江

全球最大中文IT社区CSDN与权威技术媒体《程序员》总编。知名出版机构图灵公司联合创始人、前总编。曾任华章公司副总编，*Dr.Dobb's Journal* 中文版《软件研发》杂志主编。

高　松

担任全球最大中文IT社区CSDN记者，权威技术媒体《程序员》杂志报道和产品版主编。

秘书处：

黄　群

中国软件行业协会过程改进分会软件业务部副部长，中国敏捷软件开发联盟秘书处联系人，热爱过程改进、热爱敏捷，新浪微博 @CAA黄群。

夏　冰

中国软件行业协会系统与过程改进分会秘书长助理。

赵　倩

中国软件行业协会过程改进分会，负责中国IT服务管理论坛及火炬IT服务创新联盟管理创新专委会的工作。

序 言

对国内企业敏捷成功实践的采集，是年初中国敏捷软件开发联盟确定的7项认领工作之一，确定此工作是基于联盟对中国敏捷软件开发运动发展状态的准确把握：当前正处于为什么做和怎么做的过渡时期，同时存在两批人：一部分人问为什么要敏捷？而另外一些问怎么做？

与此同时，经过近几年敏捷运动发展，国内一批应用敏捷的先行企业，已经有了不少实践，而且很多取得了明确的效果，把这些实践采集和编辑成册，将有助于回答很多疑问：

- 中国的软件企业可以成功实施敏捷吗？
- 敏捷能用于大型软件的交付吗？
- 敏捷实施需要的文化、制度和人才基础我们具备吗？
- 有了 CMMI 和项目管理，还需要敏捷吗？
- 实施敏捷有真正效果吗？

这种时候，我们推出《中国敏捷软件开发成功实践案例集》，并以此为基础，提炼其中的共性方法，制定《敏捷软件开发知识体系》（ADBOK）,无疑是有很大的积极意义。

书中这些案例都是由企业将自身实践中行之有效的实践编制而成，联盟组织编委会进行了简单的审核，把明显非敏捷案例去掉，然后由专家加注评语，集合成册，供其他企业参考和借鉴。

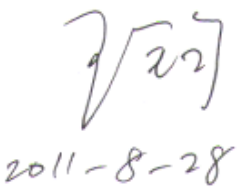
本来原计划进行评选，将其中优秀实践识别出来，但过程中发现若要严肃的开展评选，所需要时间显然不够，因此评选留待以后，此次仅仅是把所提交案例结集成册，以便能借助敏捷大会这个难得的场合尽快发布给大家阅读。

成果的取得，离不开团队的协作，我很高兴向读者介绍这个精英团队：工作组长宁德军（IBM Rational技术总监）、工作副组长张忠（用友股份研发总经理）、工作副组长李春林（东软集团过程改善中心副主任），还有21位很棒的成员（具体名单参见德军序），项目过程中，处处彰显出大家对敏捷的挚爱、对专业的热情、对行业工作的社会责任感，这些年轻的从业者所体现的精神，正是我们整个软件行业生机勃勃、精彩纷呈的原因。

也许，这些工作还很粗糙，但是毕竟我们已经在路上了——不仅学习于国际社区，而

且要在实践中有所创新，勇于分享，争取对国际社区有所回馈。

我也借此机会，代表协会真诚欢迎各路英雄豪杰大侠参与到联盟平台上，共同精化和演进这些成果，推动中国敏捷软件开发运动快速前进，实现我们“以过程改进之能，助企业发展之力”的共同目标。



编者序

在5年前上海举办的首届世界游戏开发者大会（GDC 2007），是我第一次真正领略到了敏捷开发的魅力，数百个来自不同国家、说各种语言的开发者围绕着游戏开发团队如何进行敏捷开发展开热烈讨论，几场敏捷相关的演讲也场场爆满。通过那次的敏捷开发洗礼，骨子里流淌着软件工程的我开始对敏捷开发产生了浓厚的兴趣，上网浏览各种敏捷知识、阅读各种敏捷书籍，从XP、Scrum到OpenUP、精益开发，然而有一段时间我却有些迷失了……，太多的敏捷流派，太多的敏捷实践，我甚至不知道何为真正的敏捷！

后来，带着许多的迷茫，我参加了公司举办的敏捷教练的培训。从各种敏捷的基本知识，到Scrum Master的高级进阶，再到公司内部的各种敏捷转型实战分享，我完成了一次非常系统的敏捷修炼之旅。通过和老师和其他敏捷教练的交流，自己似乎有了种豁然开朗的感觉！正是从那时起，我就有了写敏捷开发知识体系的冲动，因为我知道不会所有的人都像我一样幸运，有如此系统的培训机会；我知道还会有越来越多的人步入敏捷的殿堂；我知道有很多的朋友还在努力学习和感悟着敏捷。而我们能够做的和应该做的，正是联合敏捷领域的爱好者和志愿者，尽快推出中国的《敏捷开发知识体系》，以便帮助更多的朋友能够更快地完成敏捷开发的学习和思考过程！

今天的成绩，只是一个起点，真心希望有越来越多的朋友加入到我们的行列，不断完善《敏捷开发知识体系》，不断提出您的建议和反馈，分享您的理解和思考！路漫漫其修远兮，吾将上下而求索，人生有涯，智慧无限！

在此我要感谢所有为《敏捷开发知识体系》播撒汗水的朋友：李春林、张忠、陈志波、高航、黄方、刘曙光、庞建荣、钱岭、吴文龙、刑雷、许舟平、闫建伟、余晓、袁斌、张克强、赵静、高松、黄晓倩、黄燕、刘嘉、刘江，还要感谢联盟秘书处工作人员黄群、夏冰、赵倩。

目　录

1 敏捷开发知识体系整体框架

1.1 敏捷开发知识体系的核心

敏捷开发是一个灵活的开发方法，用于在一个动态的环境中向干系人快速交付价值。其主要特点是关注持续的交付价值，通过迭代和快速用户反馈管理不确定性和拥抱变更；它承认个人才是价值的最终源泉，强调通过有效的个人激励，提升团队的工作绩效。

敏捷开发知识体系的核心是敏捷宣言（详细内容参见第二章），它们是敏捷开发思想和价值观的集中体现，它直接影响人们的思维模式。因此，正确的理解敏捷宣言，建立正确的敏捷价值观是成功开展敏捷开发的关键。敏捷的价值观更相信通过个人及其有效协作，持续不断的交付价值；通过客户的参与和快速反馈，更好地拥抱变更，提升客户满意度。它充分体现敏捷文化中面向结果、关注价值和以客户为核心的协作创新理念。

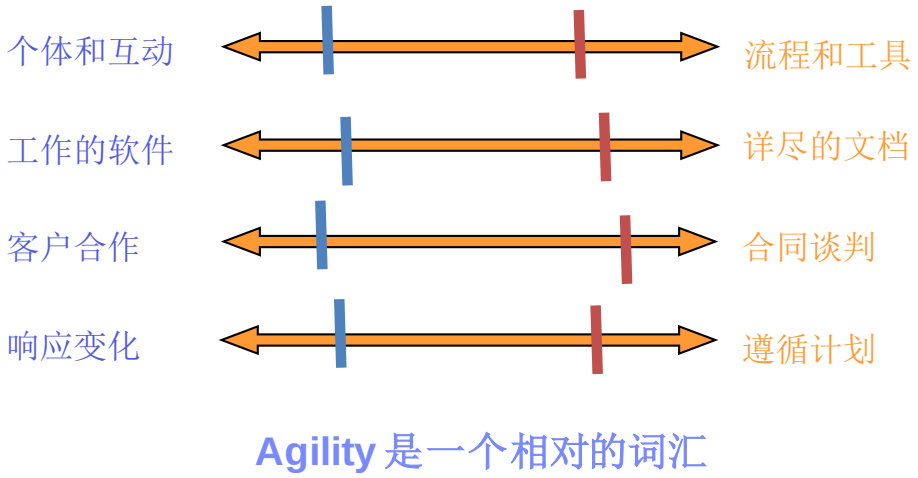


图1-1 敏捷宣言

如图1-1所示，敏捷宣言告诉我们敏捷是一个相对的词汇，具体敏捷程度取决于项目团队的上下文，例如复杂项目由于其团队规模、技术特点和循规要求等，将会要求团队有更严格的治理流程和工具支持、更规范的文档和计划要求，但仍然可以借助敏捷的价值观和各种实践解决开发过程中遇到的问题。因此，在具体的敏捷开发实践中，我们

必须实事求是地采用合适的敏捷实践，以实用主义为指导思想，面向业务结果和价值，切不可为了敏捷而敏捷。

1.2 敏捷开发管理实践

随着敏捷开发运动的开展，如图1-2所示，敏捷的践行者逐渐发展出两类敏捷开发最佳实践：敏捷开发管理实践和敏捷开发工程实践。敏捷开发管理实践泛指用于指导敏捷团队进行敏捷开发实践的开发方法和流程，业界应用最广的敏捷开发管理实践包括：

- Scrum

Scrum包括一系列实践和预定义角色，是一种灵活的软件管理过程。它提供了一种经验方法，可以帮助你驾驭迭代，实现递增的软件开发过程。这一过程是迅速、有适应性、自组织的，它发现了软件工程的社会意义，使得团队成员能够独立地集中在创造性的协作环境下工作。

- 精益开发（Lean）

精益的理念，就是从最终用户的视角上观察生产流程，视任何未产生增值的活动为浪费，并通过持续地消除浪费，实现快速交付、高质量与低成本。因此，对于软件开发而言，在开发者或最终用户的视角上观察软件开发过程，并发现和消除其无益于快速交付的行为，即为精益的软件开发。

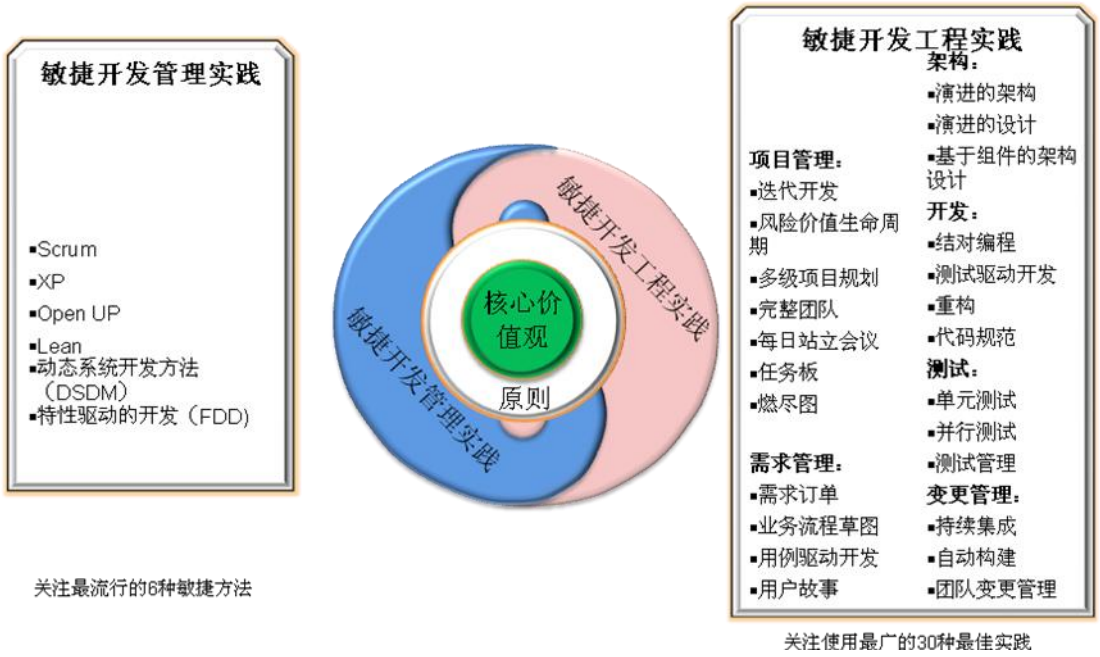
- 极限编程（XP）

极限编程是由Kent Beck提出的一套针对业务需求和软件开发实践的规则，它的作用在于将二者力量集中在共同的目标上，高效并稳妥地推进开发。其力图在不断变化的客户需求的前提下，以持续的步调，提供高响应性的软件开发过程及高质量的软件产品。

- OpenUP

最早源自IBM内部对RUP（Rational Unified Process）的敏捷化改造，它是由一组适合高效率软件开发的最小实践集组成的敏捷化的统一过程。它的基本出发点是务实、敏捷和协作。通过提供支撑工具、降低流程开销等措施，OpenUP方法即可以按照基本模式使用，也可以扩展更多的实践，从而，拥有更为广泛的应用范围。

敏捷开发管理实践的具体内容描述，请参见第三章。



注：在本版本中，上图中的敏捷实践未能全部说明或列出，希望能够在广大读者的帮助下，逐步完善和改进

图1-2 ADBOK整体框架图

1.3 敏捷开发工程实践

敏捷开发工程实践泛指用于指导敏捷团队进行敏捷开发的各种工程化最佳实践，业界应用最广的敏捷开发工程实践包括：

项目管理：

◆ 迭代开发

◆ 风险价值生命周期

◆ 多级项目规划

◆ 完整团队

◆ 每日站立会议

◆ 任务板

◆ 燃尽图

需求管理：

◆ 需求订单

◆ 业务流程草图

◆ 用例驱动开发

◆ 用户故事

架构：

◆ 演进的架构

◆ 演进的设计

◆ 基于组件的架构设计

开发：

◆ 结对编程

◆ 测试驱动开发

◆ 重构

◆ 代码规范

测试：

◆ 单元测试

◆ 并行测试

◆ 测试管理

变更管理：

◆ 持续集成

◆ 自动构建

◆ 团队变更管理

部分敏捷开发工程实践的具体内容，请参见第四章。

值得我们注意的是组织实现敏捷的方法本身也应该是敏捷的，应该是面向结果的、关注价值、以客户为核心的实践过程。因此，最终衡量一个团队、一个组织是否更加敏捷的标准，应该是面向结果的，而不是我们采取了哪些实践。从经济学的角度，敏捷与否是由应对变更的单位成本决定的。无论你采用何种实践，如果在实施某个敏捷实践前，你应对变更的单位成本是平均10000元/变更，而实施之后你的应对变更的单位成本是平均9000元/变更，则可以说你的团队更敏捷了；相反，如果实施之后你的应对变更的单位成本是平均11000元/变更，则你的团队变得更加不敏捷了。

在敏捷开发知识体系的其它章节，将具体描述每种敏捷开发管理实践和工程实践，为了方便阅读，我们将采用统一的方式描述其中具体内容。其中，敏捷开发管理实践描述主要包括以下主要部分：

- ◆ 定义和特性说明
- ◆ 主要角色
- ◆ 主要活动和最佳实践
- ◆ 主要输入输出
- ◆ 工作流程

而敏捷开发工程实践描述将主要包括以下主要部分：

- ◆ 定义和特性说明
- ◆ 应用说明
- ◆ 案例说明

2 敏捷开发核心价值观和原则

2.1 敏捷软件开发宣言

2001年2月，17位在当时被称之为“轻量级方法学家”的软件开发领域领军人物聚集在美国犹他州的滑雪胜地雪鸟（Snowbird）雪场。经过两天的讨论，“敏捷”（Agile）这个词为全体聚会者所接受，用以概括一套全新的软件开发价值观。这套价值观，通过一份简明扼要的《敏捷宣言》，传递给世界，宣告了敏捷开发运动的开始。

敏捷软件开发宣言

我们一直在实践中探寻更好的软件开发方法，
身体力行的同时也帮助他人。由此我们建立了如下价值观：
个体和互动 高于 流程和工具
工作的软件 高于 详尽的文档
客户合作 高于 合同谈判
响应变化 高于 遵循计划
也就是说，尽管右项有其价值，我们更重视左项的价值。

注：以上敏捷软件开发宣言摘自敏捷宣言简体中文版官方网站
http://agilemanifesto.org/iso/zhchs/

2.2 敏捷开发的核心价值观

敏者，疾也。对外来的刺激做出迅速、机灵的反应；捷者，獵也。以最短的路径去追赶和实现目标。敏捷开发的核心理念就是以最简单有效的方式快速的达成目标，并在这个过程中及时地响应外界的变化，做出迅速的调整。

敏捷开发的第一条价值观就是“以人为本”，强调“个体（人）”及“个体”间的沟通与

协作在软件开发过程中的重要性。这个顺序不是偶然而为之的，敏捷开发将重视个体潜能的激发和团队的高效协作作为其所推崇的第一价值观。

敏捷开发的第二条价值观就是“目标导向”。同其他众多管理理论和模型一样，敏捷开发认同目标导向是成功的关键，因为没有目标也就无所谓成功。敏捷开发的价值观中清楚地阐明，软件开发的目标是“可工作的软件”，而不是面面俱到的文档。而遗憾的是，很多软件项目已经在纷繁的文档之中迷失了自己的目标。

敏捷开发的第三条价值观就是“客户为先”。虽然敏捷开发强调的第一价值观是“以人为本”，但敏捷宣言的缔造者们并没有忘记客户，相反他们真正的理解客户的需求、懂得如何与客户合作。敏捷价值观里强调的“客户为先”即不是简单的把客户当作“上帝”、刻板的推崇“客户至上”，客户要求什么、我们就做什么；也不是把客户当作“谈判桌上的对手”甚至“敌人”，去斗智斗勇。敏捷价值观里中把客户当成了合作者和伙伴，把自己的使命定位为“帮助客户取得竞争优势”。

敏捷开发的第四条价值观就是“拥抱变化”。人们常说“世界上唯一不变的就是变化”，大多数人也相信事实确实如此。而以往很多的软件项目却忽视了这一点，或者更准确的说是它们不愿意“正视”。它们总是试图用详尽的计划去预先穷举这些变化，然后又试图通过严格遵循计划来控制变化的发生，而结果往往是被不断涌现的变化击垮。敏捷开发价值观中承认变化是软件开发的一部分、并相信正是客户在不断变化其需求的过程中明晰了其真正的需要。因而敏捷开发欢迎变化、拥抱变化，并可坦然应对变化，正是这些变化为客户和项目带来了价值。

最后，我们还应记住敏捷宣言中的最后一句话：“**尽管右项有其价值，我们更重视左项的价值**”——敏捷宣言并未否定或贬损“右项”的价值，在敏捷开发的价值观中承认“流程和工具”、“详尽的文档”、“合同谈判”以及“遵循计划”的重要性，只是两相比较，“**更重视左项的价值**”。

2.3 敏捷开发的原则

2.3.1 敏捷开发的原则

敏捷宣言遵循的原则

- ◆ 我们最重要的目标，是通过持续不断地及早交付有价值的软件使客户满意。
- ◆ 欣然面对需求变化，即使在开发后期也一样。为了客户的竞争优势，敏捷过程掌控变化。
- ◆ 经常地交付可工作的软件，相隔几星期或一两个月，倾向于采取较短的周期。
- ◆ 业务人员和开发人员必须相互合作，项目中的每一天都不例外。
- ◆ 激发个体的斗志，以他们为核心搭建项目。提供所需的环境和支援，辅以信任，从而达成目标。
- ◆ 不论团队内外，传递信息效果最好效率也最高的方式是面对面的交谈。
- ◆ 可工作的软件是进度的首要度量标准。
- ◆ 敏捷过程倡导可持续开发。责任人、开发人员和用户要能够共同维持其步调稳定延续。
- ◆ 坚持不懈地追求技术卓越和良好设计，敏捷能力由此增强。
- ◆ 以简洁为本，它是极力减少不必要工作量的艺术。
- ◆ 最好的架构、需求和设计出自自组织团队。
- ◆ 团队定期地反思如何能提高成效，并依此调整自身的举止表现。

注：以上敏捷宣言遵循的原则摘自敏捷宣言简体中文版官方网站

http://agilemanifesto.org/iso/zhchs/

2.3.2 敏捷开发原则的应用

敏捷开发原则是对敏捷价值观的解释和实践，它将敏捷的价值观落实到具体的可操作的原则之上，严格的遵循这十二条原则，是敏捷软件开发项目得以成功的基石。

可以说这十二条原则已经囊括了软件项目管理的基本流程，而且这些流程足够具体：它告诉我们项目管理的第一步就是要明确目标，而软件项目的终极目标就是“不断

地及早交付有价值的软件使客户满意”；它提示我们软件工程的起始点是需求，而需求的固有特征就是不断变化，敏捷开发过程要“掌控变化”；它强调“可工作的软件是进度的首要度量标准”，因而需要以尽可能短的周期“经常地交付可工作的软件”；它重视相关干系人的协调（“业务人员和开发人员必须相互合作”、“责任人、开发人员和用户要能够共同维持其步调稳定延续”），重视激发个人潜能（“激发个体的斗志”），要求团队使用最高效的沟通方式（“面对面的交谈”）；它推崇技术变革所具备的强大能量（“坚持不懈地追求技术卓越和良好设计”）；它强调精益生产（简洁为本），要求项目采用“自组织”团队管理模式，并指出“定期的总结反思”，是校准团队行为、最终达成目标的有效途径。

成熟的敏捷开发团队，完全可以不再需要任何附加的冗余流程（工程技术流程除外），而基于这十二条原则成功地完成软件的开发和交付。当然，敏捷开发团队也可以以这十二条原则为基础，进一步的细化敏捷项目的管理流程、步骤、和方法工具，以便这些原则能够更好的被团队理解和执行。

3 主要的敏捷开发管理实践

3.1 敏捷开发管理实践之SCRUM

3.1.1 定义和特性说明

术语Scrum来源于橄榄球活动，在英文中的意思是橄榄球里的争球。在橄榄球比赛中，双方前锋站在一起紧密相连，当球在他们之间投掷时，他们奋力争球。1995年，在奥斯汀举办的OOPSLA '95上，杰夫·萨瑟兰和肯·施瓦伯首次提出了Scrum概念，并在随后的几年中逐步将其与业界的最佳实践融合起来，形成一种迭代式增量软件开发过程和敏捷项目管理方法，并在2001年敏捷联盟（Agile Alliance）形成后受到了更多欢迎。

Scrum是一种灵活的软件管理过程，它提供了一种经验方法，可以帮助你驾驭迭代，实现递增的软件开发过程。这一过程是迅速、有适应性、自组织的，它发现了软件工程的社会意义，使得团队成员能够独立地集中在创造性的协作环境下工作。

Scrum采用了经验方法，承认问题无法完全理解或定义，关注于如何使得开发团队快速推出和响应需求能力的最大化。因此，Scrum的一个关键原则就是承认客户可以在项目过程中改变主意，变更他们的需求，而预测式和计划式的方法并不能轻易地解决这种不可预见的需求变化。

Scrum是一个包括了一系列实践和预定义角色的过程框架。任何软件开发过程框架都可以由最基本的三个要素组成：角色（人）、活动及其输入输出。Scrum 框架主要包括以下内容：

角色：

◆ 产品负责人（Product Owner）

◆ Scrum主管（Scrum Master）

◆ 团队成员

活动：

◆ 冲刺规划会议（Sprint Plan Meeting）

◆ 每日站立会议（Scrum Daily Meeting）

◆ 冲刺复审会议（Sprint Review Meeting）

◆ 冲刺回顾会议（Sprint Retrospective Meeting）

主要输入输出：

◆ 产品订单（Product Backlog）

◆ 冲刺订单（Sprint Backlog）

◆ 燃尽图（Burndown Chart）

◆ 新的功能增量

下面我们就从角色、活动和主要输入输出三个维度对整个Scrum过程进行简单介绍。

3.1.2 主要角色

Scrum定义了许多角色，根据猪和鸡的笑话可以分为两组，猪和鸡。

一天，一头猪和一只鸡在路上散步。鸡看了一下猪说：”嗨，我们合伙开一家餐馆怎么样？”。猪回头看了一下鸡说：”好主意，那你准备给餐馆起什么名字呢？”。鸡想了想说：”餐馆名字叫火腿和鸡蛋怎么样？”。”我不这么认为”，猪说，”我全身投入，而你只是参与而已”。

“猪”角色：是全身投入项目和Scrum过程的人，主要包括代表利益干系人的产品负责人，Scrum主管和开发团队。

产品负责人（Product Owner）：代表了客户的意愿，这保证Scrum团队在做从业务角度来说正确的事情。同时他又代表项目的全体利益干系人，负责编写用户需求（用户故事），排出优先级，并放入产品订单（Product Backlog），从而使项目价值最大化。他利用产品订单，督促团队优先开发最具价值的功能，并在其基础上继续开发，将最具价值的开发需求安排在下一个冲刺迭代（Sprint）中完成。他对项目产出的软件系统负责，规划项目初始总体要求、ROI目标和发布计划，并为项目赢得驱动及后续资金。

Scrum主管（Scrum Master）：负责Scrum过程正确实施和利益最大化的人，确保它既符合企业文化，又能交付预期利益。Scrum主管的职责是向所有项目参与者讲授Scrum方法和正确的执行规则，确保所有项目相关人员遵守Scrum规则，这些规则形成了Scrum过程。Scrum主管并非团队的领导（由于他们是自我组织的），他的主要工作是去除那些影响团队交付冲刺目标的障碍，屏蔽外界对开发团队的干扰。”Scrum主管是保证Scrum成

功的牧羊犬”。

开发团队：负责找出可在一个迭代中将产品待开发事项（冲刺订单）转化为功能增量的方法。他们对每一次迭代和整个项目共同负责，在每个冲刺中通过实行自管理、自组织，和跨职能的开发协作，实现冲刺目标和最终交付产品。一般由5~9名具有跨职能技能的人（设计者，开发者等）组成。

“鸡”角色：并不是实际Scrum过程的一部分，但是必须考虑他们。敏捷方法的一个重要方面是使得用户和利益干系人参与每一个冲刺的评审和计划并提供反馈。

用户：软件是为了某些人而创建！就像”假如森林里有一棵树倒下了，但没有人听到，那么它算发出了声音吗”，”假如软件没有被使用，那么它算是被开发出来了么？”。

利益干系人（客户，提供商）：影响项目成功的人，但只直接参与冲刺评审过程。

经理：为产品开发团体架起环境的那个人。

如图3-1所示为Scrum方法中的主要角色。

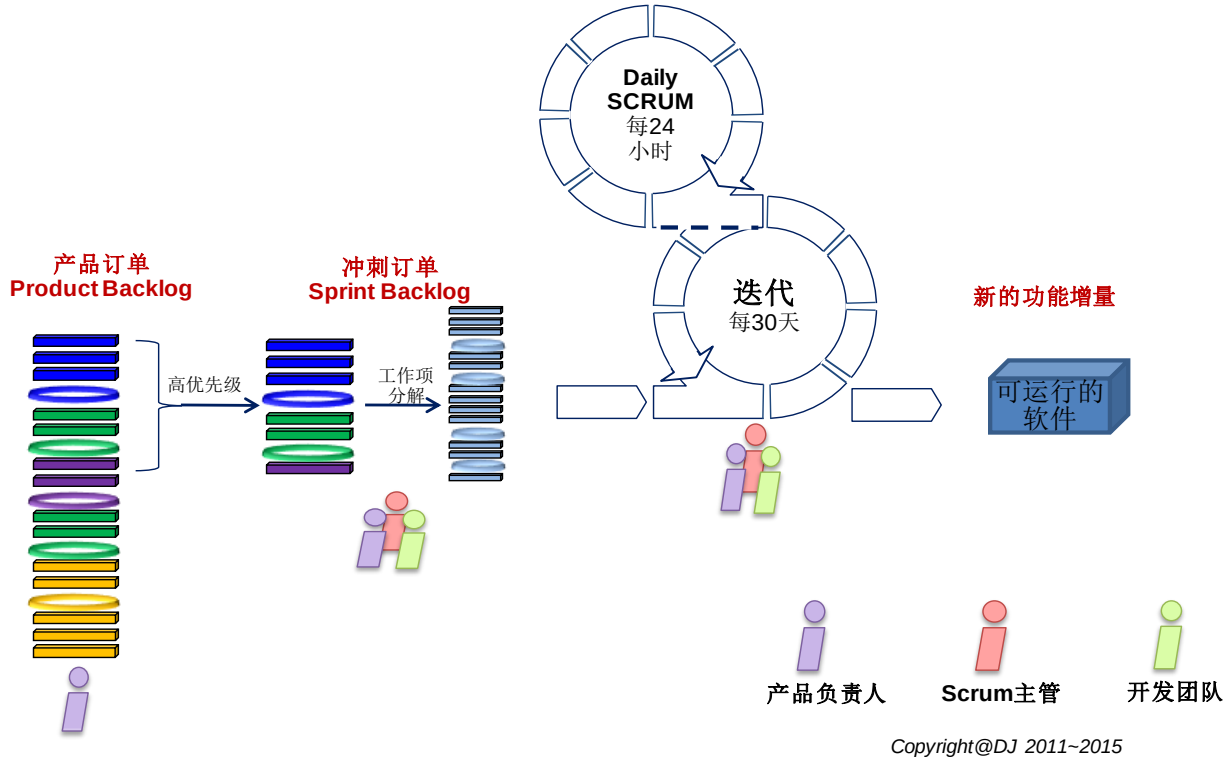


图3-1 Scrum方法中的主要角色

3.1.3 主要活动和最佳实践

Scrum作为软件开发过程框架，它包含的主要最佳实践包括以下几个方面。

迭代式软件开发

通过将整个软件交付过程分成多个迭代周期，帮助团队更好地应对变更，应对风险，实现增量交付、快速反馈。

两层项目规划（Two-Level Project Planning）

基于远粗近细的原则和项目渐进明细的特点，通过将概要的项目整体规划和详细的近期迭代计划有机结合，帮助团队有效提高计划的准确度、资源管理能力和项目的按时交付能力。

整体团队协作（Whole Team）

通过关注保持整个团队可持续发展的工作节奏、每日站立会议和自组织的工作分配，实现团队的高效协作和工作，实现提高整个团队生产力的目的。

持续集成

通过进行更频繁的软件集成，实现更早的发现和反馈错误、降低风险，并使整个软件交付过程变得更加可预测和可控，以交付更高质量的软件。

Scrum的活动主要由冲刺规划会议（Sprint Plan Meeting）、每日站立会议（Sprint Daily Meeting）、冲刺复审会议（Sprint Review Meeting）和冲刺回顾会议（Retrospective Meeting）组成。Scrum提倡所有团队成员坐在一起工作，进行口头交流，以及强调项目有关的规范（Disciplines），这些有助于创造自我组织的团队。

冲刺规划会议

冲刺开始时，均需召开冲刺规划会议，产品负责人和团队共同探讨该冲刺的工作内容。产品负责人从最优先的待开发事项中进行筛选，告知团队其预期目标；团队则评估在接下来的冲刺内，预期目标可实现的程度。冲刺规划会议一般不超过8小时。在前4个小时中，产品负责人向团队展示最高优先级的产品，团队则向他询问产品订单的内容、目的、含义及意图。而在后4小时，团队则计划本冲刺的具体安排。

每日站立会议

在冲刺中，每一天都会举行项目状况会议，被称为Scrum或”每日站立会议”。每日

站立会议有一些具体的指导原则：

1) 会议准时开始：对于迟到者团队常常会制定惩罚措施（例如罚款、做俯卧撑、在脖子上挂橡胶鸡玩具等）。

2) 欢迎所有人参加，但只有”猪”类人员可以发言。

3) 不论团队规模大小，会议被限制在15分钟。

4) 所有出席者都应站立（有助于保持会议简短）。

5) 会议应在固定地点和每天的同一时间举行。

6) 在会议上，每个团队成员需要回答三个问题：

- 今天你完成了那些工作？
- 明天你打算做什么？
- 完成你的目标是否存在什么障碍？（Scrum主管需要记下这些障碍）

冲刺复审会议

一般4个小时，由团队成员向产品负责人和其他利益干系人展示Sprint周期内完成的功能或交付的价值，并决定下一次Sprint的内容。

冲刺回顾会议

每一个冲刺完成后，都会举行一次冲刺回顾会议，在会议上所有团队成员都要反思这个冲刺。举行冲刺回顾会议是为了进行持续过程改进。会议的时间限制在4小时。

如图3-2所示，表示Scrum方法中的主要活动和交付件。

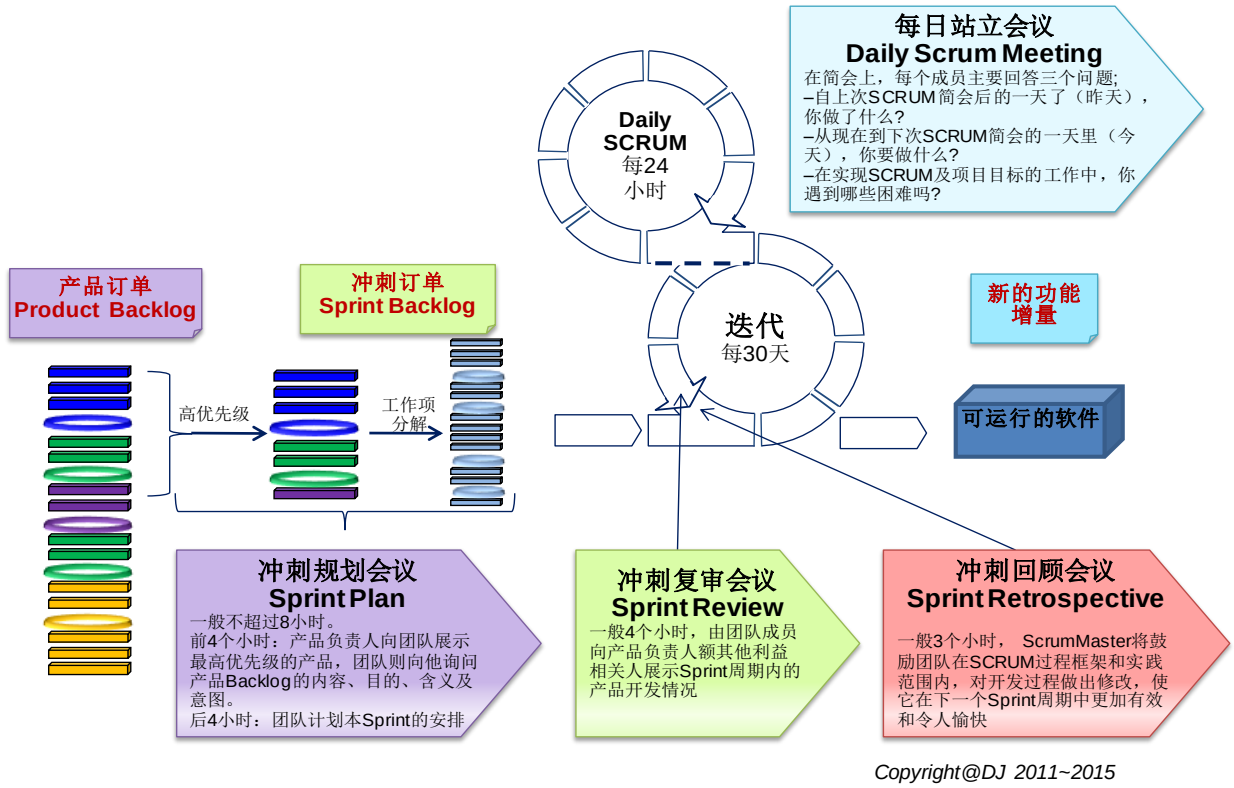


图3-2 Scrum方法中的主要活动和交付件

3.1.4 主要输入输出

产品订单

产品订单（Product Backlog）是整个项目的概要文档，它包含已划分优先等级的、项目要开发的系统或产品的需求清单，包括功能和非功能性需求及其他假设和约束条件。产品负责人和团队主要按业务和依赖性的重要程度划分优先等级，并作出估算。估算值的精确度取决于产品订单中条目的优先级和细致程度，入选下一个冲刺的最高优先等级条目的估算会非常精确。产品的需求清单是动态的，随着产品及其使用环境的变化而变化，并且只要产品存在，它就随之存在。而且，在整个产品生命周期中，管理层不断确定产品需求或对之做出改变，以保证产品适用性、实用性和竞争性。.

冲刺订单

冲刺订单（Sprint Backlog）是大大细化了的文档，用来界定工作或任务，定义团队在Sprint中的任务清单，这些任务会将当前冲刺选定的产品订单转化为完整的产品功能增量。冲刺订单在冲刺规划会议中形成，其包含的任务不会被分派，而是由团队成员签

名认领他们喜爱的任务。任务被分解为以小时为单位，没有任务可以超过16个小时。如果一个任务超过16个小时，那么它就应该被进一步分解。每项任务信息将包括其负责人及其在冲刺中任一天时的剩余工作量，且仅团队有权改变其内容。

燃尽图

燃尽图（Burndown Chart）是一个公开展示的图表，纵轴代表剩余工作量，横轴代表时间，显示当前冲刺中随时间变化而变化的剩余工作量（可以是未完成的任务数目，或在冲刺订单上未完成的订单项的数目）。剩余工作量趋势线与横轴之间的交集表示在那个时间点最可能的工作完成量。我们可以借助它设想在增加或减少发布功能后项目的情况，我们可能缩短开发时间，或延长开发期限以获得更多功能。它可以展示项目实际进度与计划之间的矛盾。

新的功能增量

Scrum团队在每个冲刺周期内完成的、可交付的产品功能增量。

3.1.5 工作流程

在Scrum项目管理过程中，一般产品负责人获取项目投资，并对整个产品的成功负责。他会协调各种利益干系人，确定产品订单中初始的需求清单及其优先级，完成项目的商业价值分析和项目整体规划，并任命合适的Scrum主管开展项目工作。如图3-3所示表示Scrum方法的全景图。

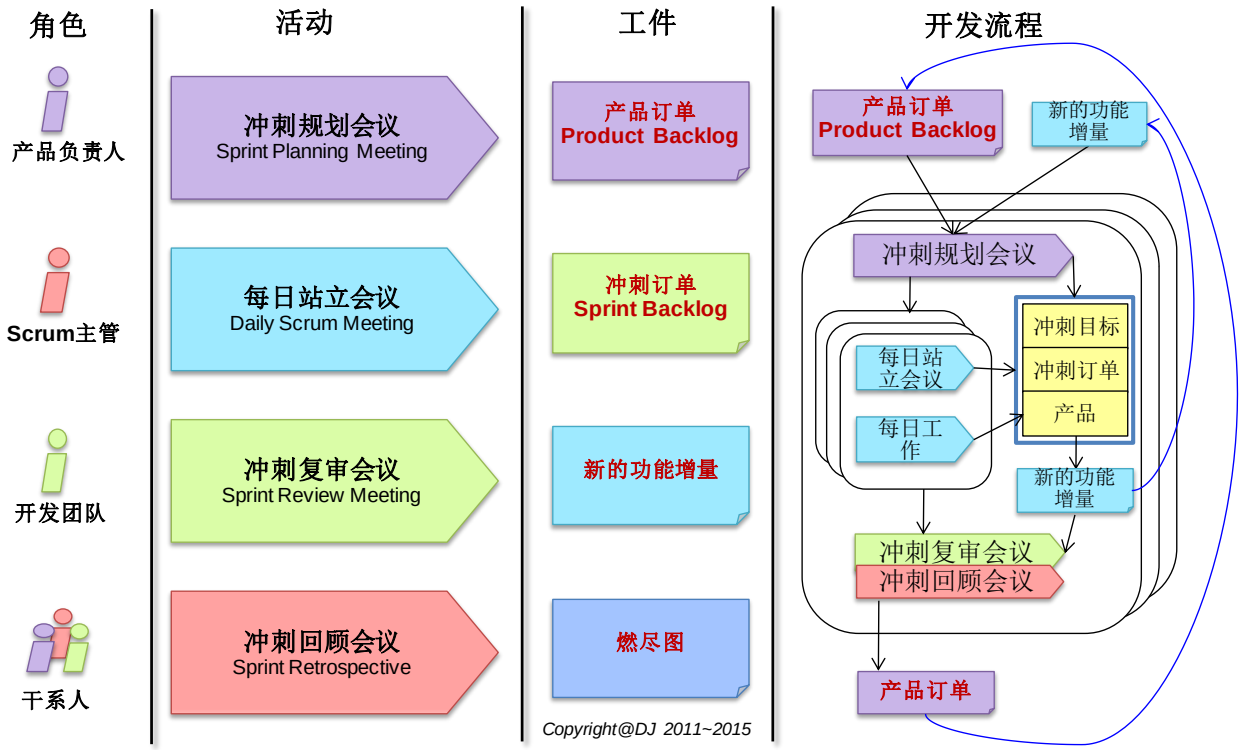


图3-3 Scrum方法全景图

在Scrum软件开发项目中，每个冲刺就是较短周期的迭代，通常为2~4周。在每个冲刺开始时，产品负责人和Scrum主管通过召开冲刺规划会议和“两层项目规划”的最佳实践，制定冲刺订单（类似于迭代计划），明确将在本次冲刺中实现的需求清单，相应的工作任务和负责人。在每个冲刺迭代中，团队强调应用“整体团队协作”的最佳实践，通过保持可持续发展的工作节奏和每日站立会议，实现团队的自组织、自适应和自管理，高效完成冲刺工作。在每个冲刺结束时，团队都会召开冲刺复审会议，团队成员会在会上分别展示他们开发出的软件（或其他有价值的产品），并从产品负责人和其他利益干系人那里，得到反馈信息。

在冲刺复审会议之后，团队会自觉召开冲刺回顾会议，回顾整个项目过程，讨论那些方面做得好，哪些方面可以改进，实现软件交付过程的持续、自发的改进。

3.2 敏捷开发管理实践之极限编程（XP)

3.2.1 定义和特性说明

XP(eXtreme Programming)极限编程是由Kent Beck提出的一套针对业务需求和软件开发实践的规则，它的作用在于将二者力量集中在共同的目标上，高效并稳妥地推进开发。

其力图在不断变化的客户需求的前提下，以持续的步调，提供高响应性的软件开发过程及高质量的软件产品。

XP提出的一系列实践旨在于满足程序员高效的短期开发行为和项目长期利益的共同实现，这一系列实践长期以来被业界广泛认可，实施敏捷的公司通常会全面或者部分采用。

XP开发体现如下特征：

◆ 短周期的开发形式，以支持尽早的、持续的反馈

◆ 递增地进行计划编制，总体计划在整个生命周期中是不断发展的

◆ 依赖口头交流、团队和客户共同测试及代码产品不断沟通并进化系统结构和需求

◆ 依赖于团队成员间的紧密协作

XP方法的价值观：

◆ 沟通 (Communication)

◆ 简单 (Simplicity)

◆ 回馈 (Feedback)

◆ 勇气 (Courage)

◆ 尊重 (Respect)

XP方法的实施原则：

◆ 快速反馈 (Rapid feedback)

◆ 假设简单 (Assuming simplicity)

◆ 包容变化 (Embracing change)

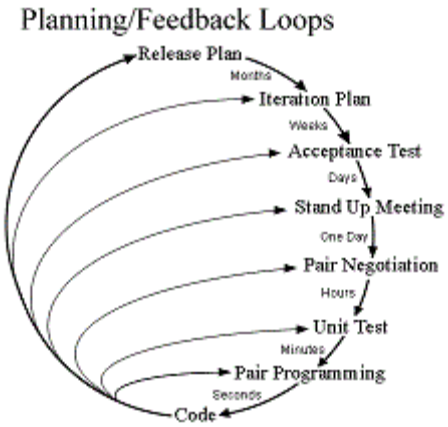


图3-4 XP怎样实现Feedback

3.2.2 主要角色

XP倡导“统一的开发团队”（Whole Team）

1) 程序员是项目开发小组中必不可少的成员；小组中可以有测试员，帮助制订、实施验收测试；有分析员，帮助确定需求。

2) 这个小组中必须至少有一个人对用户需求非常清晰，能够提出需求、决定各个需求的商业价值（优先级）、根据需求等的变化调整项目计划等。这个人扮演的是”客户”这个角色，当然最好就是实际的最终用户。和所有方法一样，客户是提出需求并最终获得产品的角色。XP方法的特别之处，明确提出希望客户陪伴团队一起工作，随时能够给出需求细节信息支持和反馈意见。

3) 通常还有个Coach（教练），负责跟踪开发进度、解决开发中遇到的一些问题、推动项目进行；还可以有一个项目经理，负责调配资源、协助项目内外的交流沟通等等。

项目小组中有这么多角色，但并不是说，每个人做的工作是别人不能插手或干预的。XP鼓励每个人尽可能地为项目多做贡献。

3.2.3 主要活动和最佳实践

主要活动

XP主要活动归结为“编码、测试、聆听、设计”四种。

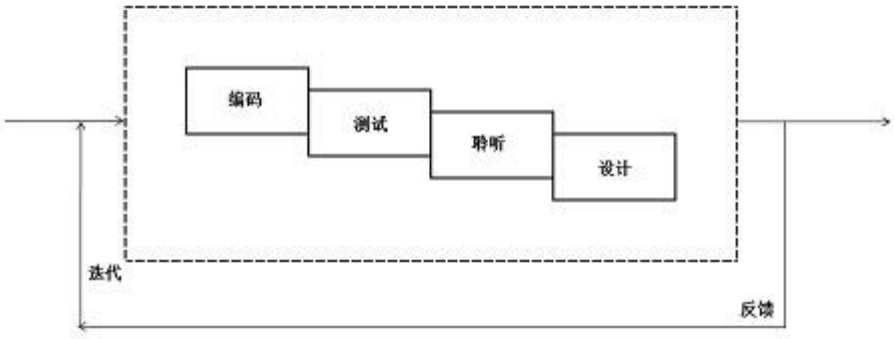


图3-5 XP主要活动

◆ 通过编码实现机能；

◆ 测试保证按质量标准按时达成编码任务；

◆ 聆听保证了知道为什么编码以及要测试什么；

◆ 设计是为了不断地支持编码、测试、聆听。

最佳实践

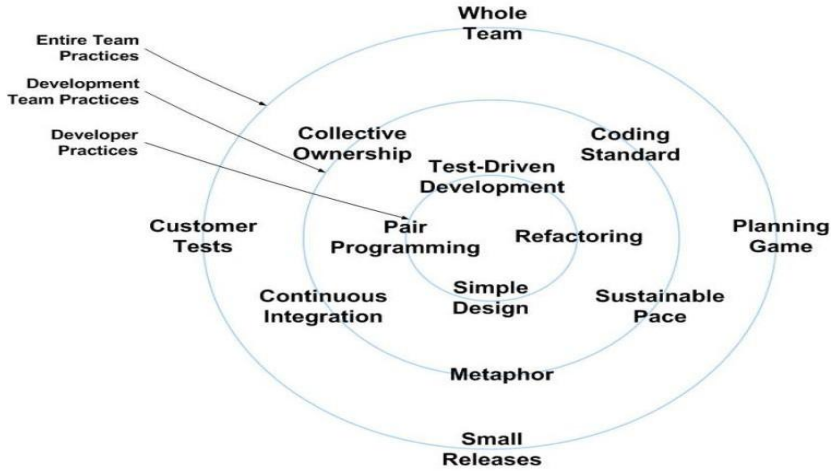


图3-6 XP最佳实践

按照整体实践（Entire Team Practices），开发团队实践（Development Team Practices），开发者实践（Developer Practices）三个层面，XP提供如下13个核心实践：

1) 统一团队（原名：在场客户）——Whole Team（On-Site Customer）

XP项目的所有的贡献者坐在一起。这个团队必须包括一个业务代表 -“客户”- 提供要求，设置优先事项。如果客户或他的助手之一，是一个真正的最终用户，是最好的；该小组当然包括程序员；可能包括测试人员，帮助客户定义客户验收测试；分析师可帮助客户确定的要求。通常还有一个教练，帮助团队保持在正确轨道上；可能有一个上层经理，提供资源，处理对外沟通，协调活动。一个XP团队中的每个人都可以以任何方式作出贡献。最好的团队，没有所谓的特殊人物。

2) 策划游戏——Planning Game

预测在交付日期前可以完成多少工作；现在和下一步该做些什么。不断的回答这两个问题，就是直接服务于如何实施及调整开发过程；与此相比，希望一开始就精确定义整个开发过程要做什么事情以及每件事情要花多少时间，则事倍功半。针对这两个问题，XP中又两个主要的相应过程：”软件发布计划（Release Planning）”和”周期开发计划（Iteration Planning）”。

3) 小型发布——Small Release

每个周期开发达成的需求是用户最需要的东西。在XP中，每个周期完成时发布的系统，用户都应该可以很容易地评估，或者已能够投入实际使用。这样，软件开发不再是看不见摸不着的东西，而是实实在在的价值。XP要求频繁地发布软件，如果有可能，应每天都发布新版本；而且在完成任何一个改动、整合或者新需求后，就应该立即发布一个新版本。这些版本的一致性和可靠性，靠验收测试和测试驱动开发来保证。

4) 客户验收——Customer Tests

客户对每个需求都定义了一些验收测试。通过运行验收测试，开发人员和客户可以知道开发出来的软件是否符合要求。XP开发人员把这些验收测试看得和单元测试一样重要。为了不浪费宝贵的时间，最好能将这些测试过程自动化。

5) 集体代码所有——Team Code Ownership

在很多项目中，开发人员只维护自己的代码，而且不喜欢其他人修改自己的代码。因此即使有相应的比较详细的开发文档，但一个程序员却很少、也不太愿意去读其他程序员的代码；而且因为不清楚其他人的程序到底实现了什么功能，一个程序员一般也不敢随便改动其他人的代码。同时，因为是自己维护自己的代码，可能因为时间紧张或技术水平的局限性，某些问题一直不能被发现或得到比较好的解决。针对这点，XP提倡大家共同拥有

代码，每个人都有权利和义务阅读其他代码，发现和纠正错误，重整和优化代码。这样，这些代码就不仅仅是一两个人写的，而是由整个项目开发队伍共同完成的，错误会减少很多，重用性会尽可能地得到提高，代码质量会非常好。

6) 程序设计标准 / 程序设计规约——Coding Standards/Conventions

XP开发小组中的所有人都遵循一个统一的编程标准，因此，所有的代码看起来好像是一个人写的。因为有了统一的编程规范，每个程序员更加容易读懂其他人写的代码，这是是实现Collective Code Ownership的重要前提之一。

7) 恒定速率（又名：40 小时工作）——Sustainable Pace / 40-hour Week

XP团队处于高效工作状态，并保持一个可以无限期持续下去的步伐。大量的加班意味着原来的计划是不准确的，或者是程序远不清楚自己到底什么时候能完成什么工作。而且开发管理人员和客户也因此无法准确掌握开发速度；开发人员也因此非常疲劳而降低效率及质量。XP认为，如果出现大量的加班现象，开发管理人员（比如Coach）应该和客户一起确定加班的原因，并及时调整项目计划、进度和资源。

8) 系统隐喻——Metaphor

为了帮助每个人一致清楚地理解要完成的客户需求、要开发的系统功能，XP开发小组用很多形象的比喻来描述系统或功能模块是怎样工作的。

9) 持续集成——Continuous Integration / Build

在很多项目中，往往很迟才把各个模块整合在一起，在整合过程中开发人员经常发现很多问题，但不能肯定到底是谁的程序出了问题；而且，只有整合完成后，开发人员才开始稍稍使用整个系统，然后就马上交付给客户验收。对于客户来说，即使这些系统能够通过终验收测试，因为使用时间短，客户们心里并没有多少把握。为了解决这些问题，XP提出，整个项目过程中，应该频繁地、尽可能早地整合已经开发完的USERSTORY（每次整合一个新的USERSTORY）。每次整合，都要运行相应的单元测试和验收测试，保证符合客户和开发的要求。整合后，就发布一个新的应用系统。这样，整个项目开发过程中，几乎每隔一两天，都会发布一个新系统，有时甚至会一天发布好几个版本。通过这个过程，客户能非常清楚地掌握已经完成的功能和开发进度，并基于这些情况和开发人员进行有效地、及时地交流，以确保项目顺利完成。

10) 测试驱动开发——Test-driven Development

反馈是XP的四个基本的价值观之一——在软件开发中，只有通过充分的测试才能获得充分的反馈。XP中提出的测试，在其它软件开发方法中都可以见到，比如功能测试、单元测试、系统测试和负荷测试等；与众不同的是，XP将测试结合到它独特的螺旋式增量型开发过程中，测试随着项目的进展而不断积累。另外，由于强调整个开发小组拥有代码，测试也是由大家共同维护的。即，任何人在往代码库中放程序（CheckIn）前，都应该运行一遍所有的测试；任何人如果发现了一个BUG，都应该立即为这个BUG增加一个测试，而不是等待写那个程序的人来完成；任何人接手其他人的任务，或者修改其他人的代码和设计，改动完以后如果能通过所有测试，就证明他的工作没有破坏原系统。这样，测试才能真正起到帮助获得反馈的作用；而且，通过不断地优先编写和累积，测试应该可以基本覆盖全部的客户和开发需求，因此开发人员和客户可以得到尽可能充足的反馈。

11) 重构——Refactoring

XP强调简单的设计，但简单的设计并不是没有设计的流水账式的程序，也不是没有结构、缺乏重用性的程序设计。开发人员虽然对每个USERSTORY都进行简单设计，但同时也在不断地对设计进行改进，这个过程叫设计的重构（Refactoring）。重构主要是努力减少程序和设计中重复出现的部分，增强程序和设计的可重用性。概念并不是XP首创的，它已被提出了近30年，一直被认为是高质量代码的特点之一。但XP强调把重构做到极致，应随时随地尽可能地进行重构，程序员都不应该心疼以前写的程序，而要毫不留情地改进程序。当然每次改动后，都应运行测试程序，保证新系统仍然符合预定的要求。

12) 简单的设计——Simple Design

XP要求用最简单的办法实现每个小需求，前提是按照简单设计开发的软件必须通过测试。这些设计只要能满足系统和客户在当下的需求就可以了，不需要任何画蛇添足的设计，而且所有这些设计都将在后续的开发过程中就被不断地重整和优化。在XP中，没有那种传统开发模式中一次性的、针对所有需求的总体设计。在XP中，设计过程几乎一直贯穿着整个项目开发：从制订项目的计划，到制订每个开发周期（Iteration）的计划，到针对每个需求模块的简捷设计，到设计的复核，以及一直不间断的设计重整和优化。整个设计过程是个螺旋式的、不断前进和发展的过程。从这个角度看，XP是把设计做到了极致。

13) 结对编程——Pair Programming

XP中，所有的代码都是由两个程序员在同一台机器上一起写的。这保证了所有的代码、设计和单元测试至少被另一个人复核过，代码、设计和测试的质量因此得到提高。看起来这样象是在浪费人力资源，但是各种研究表明事实恰恰相反。——这种工作方式极大地提高了工作强度和工作效率。项目开发中，每个人会不断地更换合作编程的伙伴。因此，结对编程不但提高了软件质量，还增强了相互之间的知识交流和更新，增强了相互之间的沟通和理解。这不但有利于个人，也有利于整个项目、开发队伍和公司。从这点看，结对编程不仅仅适用于XP，也适用于所有其它的软件开发方法。

尽管每一种实践都可以当做独立技能来学习和运用，但是其相互制约及促进关系，致使综合运用才有望达成最大价值：

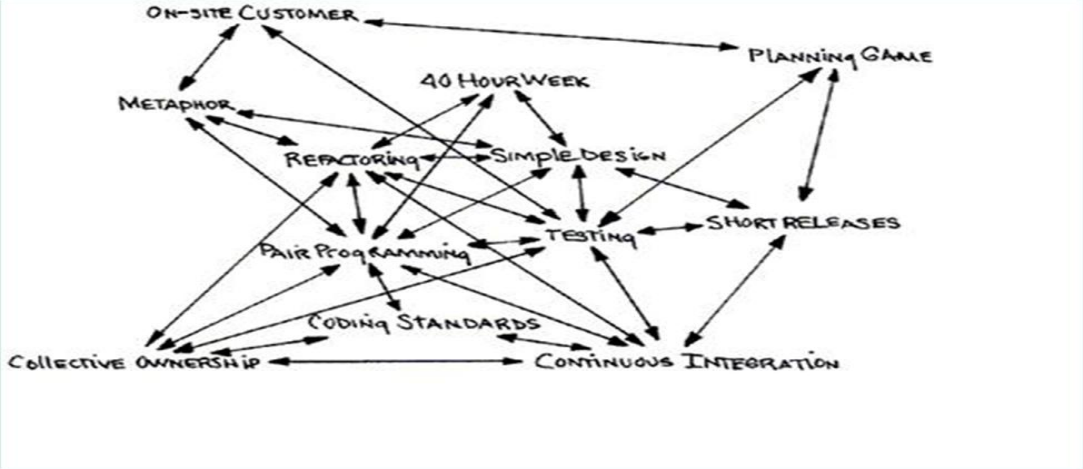


图3-7 XP实践

3.3 敏捷开发管理实践之OpenUP

OpenUP方法最早源自IBM内部对RUP（Rational Unified Process）的敏捷化改造，通过裁剪掉RUP中复杂和可选的部分，IBM于2005年推出了BUP（Basic Unified Process）和EPF（Eclipse Process Framework）。此后为了进一步推动UP族方法的发展，IBM将BUP方法捐献给Eclipse开源社区，于2006年初将BUP改名为OpenUP。此后，在20多名业界敏捷实践专家的共同努力下，结合大量现有敏捷实践，于2006年9月发布EPF1.0和OpenUP 0.9。2007年中

发布OpenUP1.0版本，目前最新版本是2011年6月发布的1.5.1.2版本。

（http://epf.eclipse.org/wikis/openup/）

3.3.1 定义和特性说明

OpenUP是由一组适合高效率软件开发最小实践组成的敏捷化统一过程。OpenUP方法的基本出发点是务实、敏捷和协作。通过提供支撑工具、降低流程开销等措施，OpenUP方法即可以按照基本模式使用，也可以扩展更多的实践，这样就拥有更为广泛的应用范围。

OpenUP虽然受到SCRUM、XP、Eclipse Way、DSDM、AMDD等各种敏捷方法的影响，但是主体仍然是RUP，即在一组被验证的结构化生命周期过程中应用增量迭代研发模式。OpenUP基于用例和场景、风险管理和以架构为中心的模式来驱动开发。

OpenUP中包含了四项基本原则：

1) 通过协作来统一认识、均衡各方利益

这项原则的目的在于打造健康的团队环境，激励团队协作并建立项目共识。

2) 平衡团队竞争优先级以最大化利益干系人的价值

此项实践的作用在于允许项目参与人和利益干系人开发一种即最大化利益干系人收益，又符合项目各项约束的方案。

3) 在项目早期就重点关注系统体系结构，以减低风险并组织研发

4) 通过持续获得反馈和改进来演进系统。

此项实践的意义在推动团队通过不断向利益干系人展示收益，而尽早、持续获得反馈。

这四项原则均支持敏捷宣言中的对应声明：

OpenUP原则	敏捷宣言中的声明
通过协作来统一认识、均衡各方利益	强调个人和实践，而非过程和工具
平衡团队竞争优先级以最大化利益干系人的价值	强调客户协作而非合同协商
在项目早期就重点关注系统体系结构，以减低风险并组织研发	强调可用的软件而非完善的文档
通过持续获得反馈和改进来演进系统	应对变化而非执行计划

OpenUP的主要特征如下：

1) 四阶段的软件研发：启动、细化、构建、移交；

2) 三个领域：利益干系人领域、项目团队领域和个人领域；

3) 逐层细化的增量迭代式开发；

4) 体系结构设计优先；

5) 风险价值评估；

6) 用例驱动；

7) 关注结果。

图3-8显示了OpenUP的总体组织架构图，清晰地反映出上述关键特征。

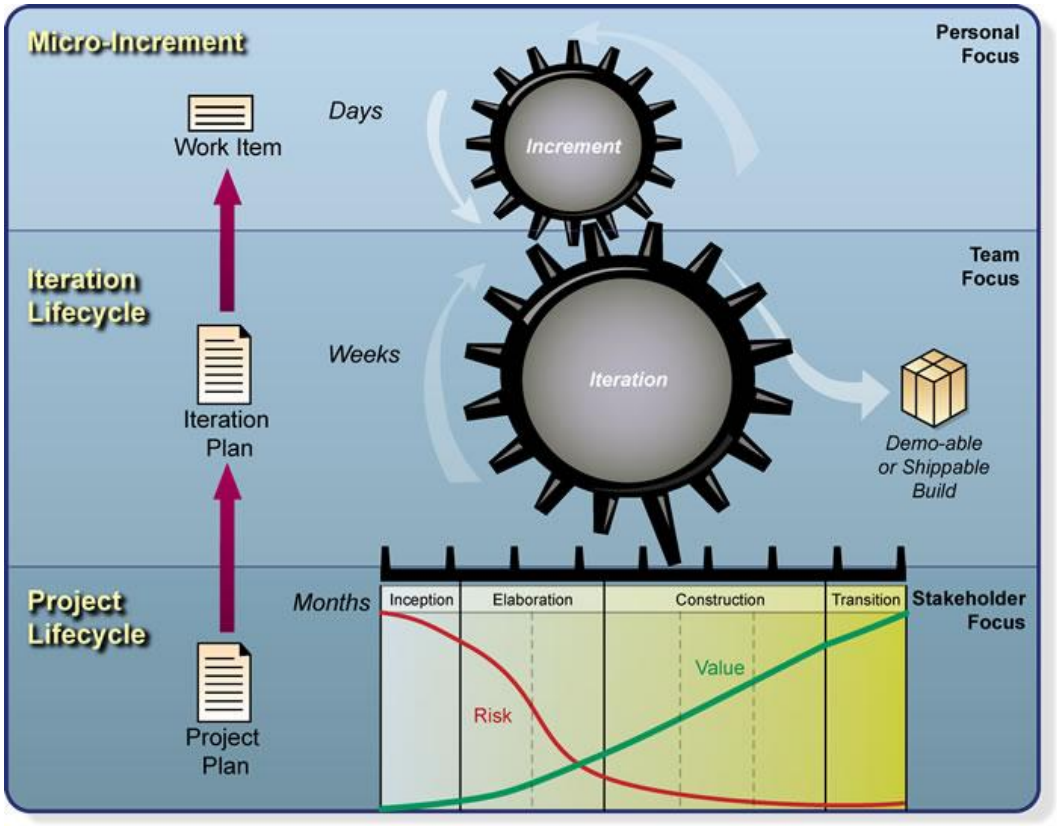


图3-8 OpenUp总体组织架构

3.3.2 主要角色

OpenUP方法中包含了7类主要角色，这些角色通常由一个成员少而精的本地团队构成，以便高效地开展工作。

利益干系人

代表了产品必须要满足的目标人群，这个角色可以由任何会被产品输出影响到的人担任。

分析师

通过从利益干系人处获取输入，分析师代表了客户和最终用户的利益。他们需要理解问题、捕获并设置需求的优先级。

架构师

负责设计软件体系架构，包括对关键技术作出选择，这种选择将约束和限制整个项目的设计和实现。

项目经理

通过和利益干系人、项目团队的协作，项目经理领导并策划项目的实施。在实施过程中，项目经理还需要和利益干系人持续协调，以保证项目团队专注于项目目标。

开发人员

负责系统研发，包括依据架构进行设计、实现、单元测试和组件集成。

测试人员

负责核心测试工作，包括确认、定义、实现并执行所需的测试；记录并输出测试结果；对结果进行分析。

通用角色

负责完成团队中的通用任务。这个角色可以由任何人担任，完成的通用任务例如评审和审计、提交变更申请、维护版本管理库等。

3.3.3 主要活动和最佳实践

如前所述，OpenUP方法包含三个层次/领域的实践活动，分别是针对利益干系人、项目团队和个人。

利益干系人领域

利益干系人通过项目周期计划来获知产品的进展情况，项目周期计划被分成4个阶段，每个阶段都是一个里程碑，在里程碑处重点关注风险和交付。在每个里程碑处都需要进行下列工作：对上一个里程碑的评审、对下一个里程碑的认可，风险识别和规避。

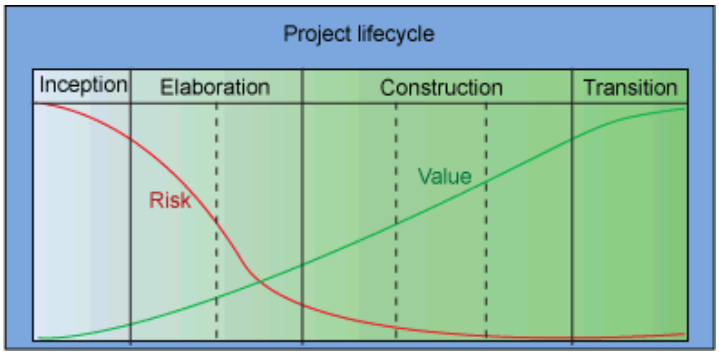


图3-9 项目生命周期

项目团队领域

项目团队需要以周为单位完成产品迭代开发。通常以2-6周为一个迭代周期。每个迭代周期起始时需要进行估算并完成周期迭代计划。通常以输出可演示版本为目标。每个迭代的策划工作通常以小时为单位完成，而不是传统意义以天为单位的估算。同时需要以天为单位完成本次迭代的架构细化工作。此后进入实际研发，建议在迭代中定期完成每周Build。在迭代结束时完成稳定的迭代交付Build。同时花费少量时间（以小时为单位）完成迭代评审和反思。

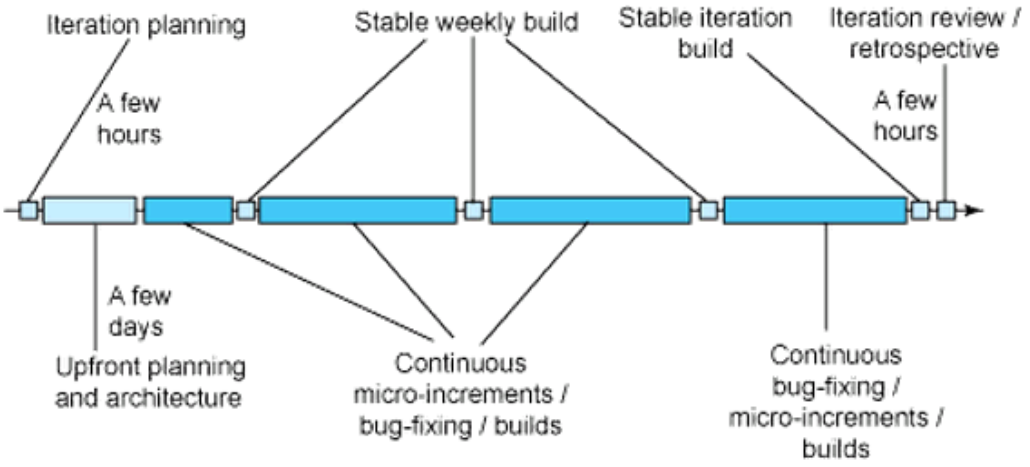


图3-10

个人领域

个人领域采用被称为微增量的研发模式。一个微增量周期在几个小时~几天不等，通常由一个人或者几个人完成。引入微增量可以将工作分成更为细小、更易于控制的部分。微增量可以是Vision的定义、也可以是模块设计、还可以是具体的研发和Bug修复工作。通过

每日团队会议、团队协作工具，团队成员之间可以分享各自的工作进展和成果，同时也可以演示自己的微增量成果来加深团队沟通与理解。OpenUP并不明确定义研发中需要哪些微增量，这个部分可以结合项目实际情况或者其他模型加以确定。

3.3.4 主要输入输出

OpenUP主要输出按照领域分类如下。

项目管理领域

项目计划：项目计划用于将各项收集到的信息汇总为一个策略层面的计划。这个计划是一个粗粒度的计划，其中标明了项目迭代周期和迭代目标。

迭代计划：描述每个迭代的目标、任务安排和批判标准的细粒度计划。

工件项列表：此列表记录了项目中所有计划内工作，以及可能影响后续项目产品的工作项。每个工作项可以包含完成工作所需的参考信息。收集工作项信息可以用于后续的工作量估算、进度跟踪。

风险列表：一个按照重要性排序列的风险项列表，每个风险项还需记录对应的补救行动和应急方案。此项工作由项目经理负责完成。

需求领域

Vision：定义了从利益干系人角度对技术解决方案的观点。文档中定义了利益干系人需要的关键特性和需求，通过系统核心需求提纲的形式提供。

词汇表：定义项目中使用的重要术语，这些术语的集合成为项目词汇表。

系统级需求：此项工件记录系统层面的质量需求和功能需求。包括用例中无法描述的部分、运营和服务层面需求、约束设计的质量属性要求、用于选择设计方案的规则。

用例模型：用于描述系统的功能和环境，可以看做是系统和客户之间的一个约定。

用例：此项内容用于捕获可为客户观察到的系统行为。这是用例模型的细化，由分析师完成。

架构设计领域

架构设计说明书：此项产品描述了系统架构，以及相关的设计思路、假定、详述、和设计决策。

项目研发领域

设计说明书：对系统功能如何实现的描述，可以看做是源代码的一个抽象。这样其他开发人员无需阅读代码即可理解系统。

系统实现：软件源代码文件、数据文件或者其他支持文件（如在线帮助）。这些部分是软件产品的原始数据。

开发人员测试：用于核实特定已经实现部分是否符合需求。通常建议采用自动化方式加以测试。也可以采用手工测试或者基于特定技术的测试。

构建：指可以运行的系统，或者具备最终产品部分功能中某个可以运行的子系统。

测试领域

测试用例：一组描述了测试输入、执行条件、和期望输出的测试规格说明书。其中每个部分都被用于验证应用场景中的某个特定方面。

测试脚本：让测试按步骤执行的一项工件。即可以是人工执行是需要参考的文档化手册指南，也可以是自动执行的程序脚本。

测试日志：收集了某个测试、多个测试或者某轮测试运行后的原始输出。

3.3.5 工作流程

OpenUP 将项目生命周期分为四个阶段：启动、细化、构建和移交。项目生命周期为利益干系人和团队成员提供可见度和决策点。这将更有效的进行管理，并且允许适当的时间做出是否继续的决定。项目计划定义了生命周期，最终结果就是一个可发布的应用程序。

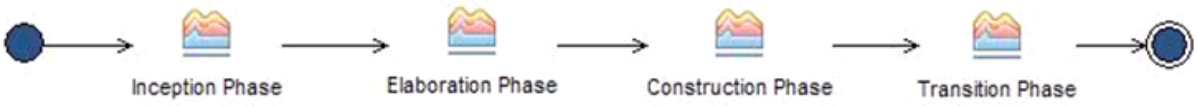


图3-11

1) 启动阶段

启动阶段的核心任务是完成项目启动、需求制订、技术方案制订和评审以及项目生命周期计划制定。

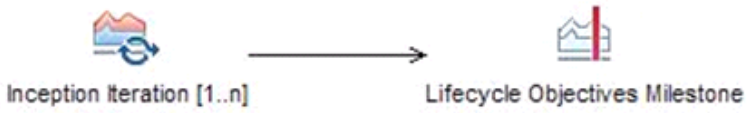


图3-12

2) 细化阶段

细化阶段的核心任务是进一步分析需求、设计系统架构、完成增量式开发计划、完成测试方案、完成迭代计划并准备启动、完成其他任务。

图3-13

3) 构建阶段

构建阶段完成系统的研发。核心任务包括进一步细化需求、完成增量式研发方案、完成测试方案、完成迭代计划并准备启动、完成其他任务。

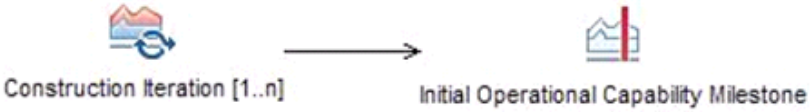


图3-14

4) 移交阶段

移交阶段的核心任务是项目交付。主要工作包括增量式研发、测试、迭代管理和其他任务。

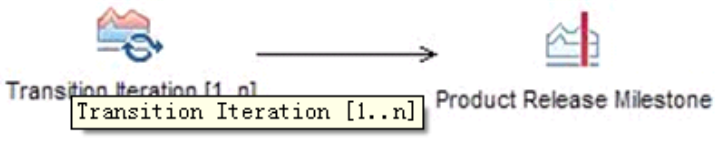


图3-15

3.4 敏捷开发管理实践之Lean软件开发

3.4.1 定义和特性说明

基于Toyota Lean生产概念 http://www.poppendieck.com/

3.4.2 精益的历史

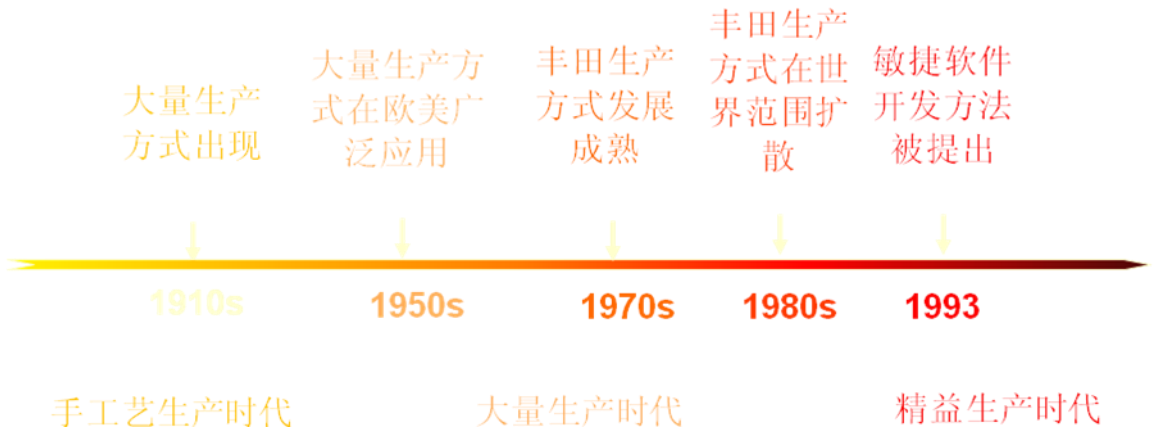


图3-16 精益的历史

3.4.3 精益的概念

在终端用户的视角上观察生产流程，视任何未产生增值的活动为浪费，并通过持续地消除浪费实现快速交付，高质量与低成本。

对于软件开发而言，在开发者或最终用户的视角上观察软件开发过程，并发现其无益于快速交付的行为，即为精益的软件开发。

3.4.4 精益五原则

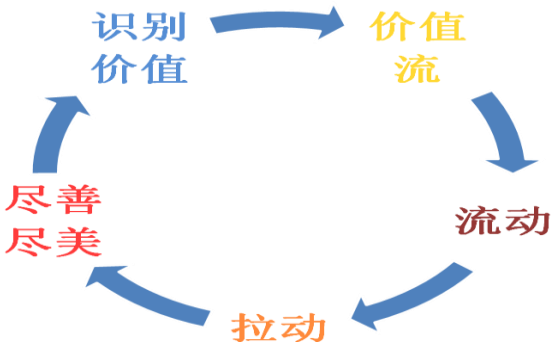


图3-17 精益五原则

3.4.4.1 敏捷和精益(Lean)是互补的

敏捷

- ◆ 一套关注于交付对客户有价值的东西的体系
- ◆ 避免对客户没有价值的东西
- ◆ 不相信“大而详细的计划”

精益(Lean)

- ◆ 出发点是流水线作业的管理方法
- ◆ 避免所有的浪费
- ◆ 在尽可能早的时候让客户介入

3.4.4.2 减少浪费

无用的功能：20%的功能往往能够提供 80%的价值，你需要的是这样的一个流程。

变来变去：如果出现需求变来变去的情况，那么是你太早的文档化需求；如果出现专门的测试和修复的阶段，那么是你测试得太晚。

跨越边界：组织边界通常会增加超过 25%的成本，是降低响应时间和妨碍沟通的隔离。

3.4.4.3 减少浪费的方法

延迟决策：抛弃那种认为只有具备完整的规约之后才能开发的想法。

消除依赖：系统架构应该允许在任何时间增添任何功能。

保持可能：把代码当成验证的机会 – 同时使代码易于变更。

在最后一刻才做出决策：做决策之前尽可能多的了解信息。

3.4.4.4 Lean软件开发的原则

消除浪费：价值流图 完整的解决方案

构建质量：基础规程 持续验证

延迟决策：保持可能

快速交付：排队理论

关注学习：产品&流程

尊重个人：团队 伙伴

优化整体：系统思考 Set-based design

4 主要的敏捷开发工程实践

4.1 迭代式开发

4.1.1 定义和特性说明

敏捷迭代开发是指每次按照相同的开发方式短期的开发软件的部分，或前期开发并不详尽的软件，每次开发结束获得可以运行的软件，以供各方干系人观测，获得反馈，根据反馈适应性的进行后续开发，经过反复多次开发，逐步增加软件部分，逐步补充完善软件，最终开发得到最后的软件。其中每次反复开发叫做一次迭代，在 Scrum 中称为 Sprint，中文常译为“冲刺”。

需要说明的是，狭义的迭代开发和狭义的增量开发是不同的。狭义的迭代开发中，每次迭代处理的范围是不变的，这与数学中的迭代算法相同；狭义的增量开发中，每次开发是增量地扩大范围。在目前的大量软件开发实践中，软件的规模都比较大，不可能一次迭代的范围就能覆盖软件全部，多数迭代处理的范围是需要增加的，而在业界书面材料上，有“迭代增量式开发”、“迭代化增量开发”，”迭代式增量开发”等的提法。在敏捷开发领域，为简便计，采用“敏捷迭代开发”的提法。因此在敏捷迭代开发中，每次迭代处理的范围是可以增加的，也可以保持不变，甚至可以缩减范围。

在敏捷软件开发中，敏捷迭代开发需满足 3 个条件：

1) 迭代的时间长度，也称为迭代周期，是有短迭代周期的要求的，一般的，敏捷迭代周期不超过8周，推荐的迭代周期是2周到4周。

2) 迭代的产物是可运行的软件。

3) 获得迭代的反馈，并处理反馈，反馈作为迭代开发中至关重要的一个方面，必须得到足够的重视。

迭代开发的优点：

1) 降低风险，在进行大规模的投资之前就可以进行关键的风险分析

2) 得到早期用户反馈，各次迭代为各方干系人提供了一个机会以对进行中又可运行的软件进行评论、反馈，同时能够对未来的开发趋势产生影响。每次迭代都能回顾了

一个能够表明各方需求决定以及开发团队对这些需求理解的软件版本，可以决定如何修改项目方向或是划分剩余需求的优先次序。

3) 对过程的测量是通过对实现的评定（而不仅仅是文档）来进行的，更加直观，更加体现用户价值。
4) 能够自然的处理变更，快速的适应新情况。快速的开发周期，可以通过后续的迭代来纠正前期迭代的误解、失误，在迭代之间自然的、平滑的处理变更。
5) 可以对局部的实现进行部署，建立团队交付能力的信心。

4.1.2 应用说明

1) 迭代的时间长度是否每个迭代都可以调整？敏捷开发的迭代周期选择和项目类型、复杂度、敏捷规模化程度有关，敏捷开发讲究固定的节奏，建议按照固定的节奏开发，因此某个迭代碰到特殊情况，尽量保证迭代时间窗不变，在不得以的情况下可以调整迭代周期长度。
2) 敏捷开发时，上个迭代结束后是否可以安排一段缓冲期后，再开展下个迭代吗？敏捷开发讲究固定的节奏，强烈不建议安排缓冲期，相关任务可以安排在下个迭代中。
3) 是否可以将原瀑布生命周期的阶段作为迭代？比如将需求分析阶段改称为需求迭代，迭代的目标产物就是需求规格说明书。这种做法不符合敏捷开发方法。敏捷迭代的目标产物应当是可运行的，不能仅仅出文档。

4.1.3 案例说明

◆ **项目基本信息**

一体化监控指挥平台-iCentroView， 是宝信软件历经多年自主研发的适用于智能化、自动化等多个业务领域的综合一体化智能监控指挥平台，起步于 2004 年。他集存储、监视、控制、报警、联动、指挥等众多功能为一体，具有”集中管理、分散控制、全面监控、安全联动”等特点。该产品从技术、设计、开发、维护等各个方面保证系统的先进性，为广大客户提供了完整、全面的综合监控管理方案，是一体化监控的有力工具。

- 团队规模：23 人，

- 环境：主要是 C++。

◆ **组织结构**

3 个子普通项目组织+独立测试组

◆ **面临的问题**

用户的功能需求不断扩大，工期要求高，产品本身有缺陷

◆ **解决方法**

采用敏捷迭代开发；迭代周期：4 周主要活动：迭代计划、迭代架构工作、功能增量的开发工作、每日构建。

◆ **主要收益**

通过迭代确立项目的“心跳”，迭代支持客户的快速反馈，迭代支持快速发布，拥抱变更

4.2 持续集成

4.2.1 定义和特性说明

持续集成是指当代码提交后，马上启动自动编译、自动部署和自动化测试来快速验证软件，从而尽快地发现集成错误。持续集成的英文是 Continuous integration，在敏捷语境下缩写为 CI。

在持续集成中，团队成员频繁集成他们的工作成果，一般每人每天至少集成一次，也可以多次。每次集成会经过自动构建（包括自动测试）的验证，以尽快发现集成错误。许多团队发现这种方法可以显著减少集成引起的问题，并可以加快团队合作软件开发的速度。

持续集成的目的是自动化软件或部件之间的集成，并持续改进，从而使工程师能够尽早发现和解决问题，交付高质量软件。

持续集成是在短时间迭代中交付稳定、可用软件的关键。

持续集成原则：

1) 所有的开发人员需要在本地机器上做本地编译，然后再提交的版本控制库中。

2) 需要有专门的集成服务器来执行集成,每天可以执行多次集成。

3) 每次集成争取都要100%通过，如果集成失败，马上修复，修复失败的集成是优先级

最高的事情。

4) 每次成功的集成都可以生成可运行的软件。

5) 修复失败的构建是优先级最高的事情。

持续集成能够帮助开发团队应对如下挑战：

1) 软件构建自动化

使用 CI，只要按一下按钮，它会依照预先制定的时间表，或者响应某一特定事件，就开始进行一次构建过程。如果想取出源码并生成构件，该过程也不会局限于某一特定 IDE、电脑或者个人。

2) 持续自动的构建检查

CI 系统能够设定成持续地对新增或修改后签入的源代码执行构建，也就是说，当软件开发团队需要周期性的检查新增或修改后的代码时，CI 系统会不断要求确认这些新代码是否破坏了原有软件的成功构建。这减少了开发者们在手动检查彼此相互依存的代码中变化情况需要花费的时间和精力。

3) 持续自动的构建测试

这个是构建检查的扩展部分，这个过程将确保当新增或修改代码时不会导致预先制定的一套测试方案在构建构件后失败。构建测试和构建检查一样，失败都会触发通知(Email, RSS 等等)给相关的当事人，告知对方一次构建或者一些测试失败了。

4) 构件生成后续过程的自动化

一旦自动化检查和测试的构建已经完成，一个软件构件的构建周期中可能也需要一些额外的任务，诸如生成文档、打包软件、部署构件到一个运行环境或者软件仓库。通过这样，构件能更迅速地提供给用户使用。

实现一个 CI 服务器需要的最低要求是，一个易获取的源代码仓库(包含源代码)，一套构建脚本和流程和一系列围绕软件构建的可执行测试。

4.2.2 应用说明

建立持续集成环境

硬件和软件的准备和安装：

- 硬件包括用于建立持续集成系统的设备；

- 软件包括持续集成工具（Jenkins（原Hudson））、代码管理工具（CVS、SVN、GIT等）、代码检查工具（PCLint、Checkstyle、Findbugs等）、自动构建工具（ant、nant、shell、bat等），测试工具（CppCheck、Junit等）、结果展示工具（各种Report plugin）等。

建立持续集成过程

根据不同的项目类型，建立不同的持续集成过程，包括：

- 代码的更新方式
- 代码的构建方式
- 代码的检查/测试方式
- 构建产品的发布方式
- 构建结果的呈现方式

持续集成结果的管理

- 持续集成结果通知
- 持续集成结果数据收集
- 代码质量问题数据收集

代码质量度量网站

- 代码稳定性度量与分析
- 代码质量问题度量与分析

并行持续集成有效实践

通常，持续集成由源代码提取（在大型项目中，代码可能由不同源代码库管理）、编译、单元测试、自动部署、构建验证测试等步骤完成。由于步骤间或步骤内可能崔在依赖关系，持续集成只能串行完成，从而增加了时间成本。

这里介绍的是一种并行持续集成的成功实践和思路。Apache Ant 提供了并行执行嵌套任务的功能，但是它不拥有任务间依赖检查的能力。而在实际开发中，Ant 任务之间经常会有依赖。通过定义和配置任务间依赖拓扑，并行执行嵌套任务，从而扩展 Ant 任务的并行能力，支持实现并行持续集成。

1) 并行源代码集成

静态源代码模块之间没有依赖，源代码提取可以很方便地做到并行进行，以提高生产率。

2) 并行自动化构建

模块之间在编译时可能会存在依赖关系。构建工程师通过定义工程间依赖拓扑，以实现并行编译。

3) 并行单元测试

定义一个单元测试框架，为每个模块工程建立单元测试工程，从而实现并行单元测试。

多个应用开发的并行持续集成。对于一个项目组，以上方法可以从单个应用开发的并行持续集成扩展为多个应用开发的并行持续集成。当应用 A 在进行步骤 1 和 2 时，其他应用等待。当应用 A 在进行步骤 3 时，从等待列表中唤醒应用 B，应用 B 开始进行步骤 1 和 2，等等。构建工程师可以管理不同小组，通过定义线程数量，来进行多应并行持续集成。

4.2.3 案例说明

案例1：宝信软件iCentroView项目

◆ **项目基本信息**

一体化监控指挥平台-iCentroView， 是宝信软件历经多年自主研发的适用于智能化、自动化等多个业务领域的综合一体化智能监控指挥平台，起步于 2004 年。他集存储、监视、控制、报警、联动、指挥等众多功能为一体，具有”集中管理、分散控制、全面监控、安全联动”等特点。该产品从技术、设计、开发、维护等各个方面保证系统的先进性，为广大客户提供了完整、全面的综合监控管理方案，是一体化监控的有力工具。

- 团队规模：23人
- 环境：主要是C++

◆ **组织结构**

3 个子普通项目组织+独立测试组

◆ **面临的问题**

代码修改后的质量不能保障

◆ **解决方法**

采用持续集成结合每日集成的方法，主要活动包括：单元测试，建设持续集成测试集。

◆ **主要收益**

保障了代码修改的质量

案例2：来自IBM的某项目

◆ **项目基本信息**

在一个在线报价和订单项目中采用敏捷软件开发，实现快速高质量交付。

- 团队规模：10人
- 环境：Java, Web

◆ **组织结构**

标准 Scrum 组织架构

◆ **面临的问题**

在敏捷实践中，持续集成的时间成本过大。

◆ **解决方法**

实现并采用并行持续集成的方法和工具

主要活动：持续集成

◆ **主要收益**

持续集成时间缩短至原来的 1/3，实现快速交付， 提高了软件质量。

案例3：东软某事业部

◆ **项目基本信息**

具备多种项目情态，人数过千的某东软事业部

◆ **组织结构**

- 矩阵型组织：事业部下，多个开发部，每事业部下存在多个项目
- 多项目：标准Scrum团队+以传统PM为核心的团队

◆ **面临的问题**

项目在开发过程中，每次版本 Release 前的编译和检查工作需要花费较长的时间，并时常伴随的问题的发生。

- 存在问题的代码经常被提交到代码库中；
- 项目代码的稳定性无法度量和保证。

◆ **解决方法**

引入持续集成工具，对代码进行自动构建，提升构建效率，并结合代码检查工具，如PClint、Checkstyle、Findbugs等，自动对代码进行检查。

对代码管理工具进行二次开发，在开发人员向代码库中提交代码时，触发代码检查工具对提交的代码进行检查，如果提交的代码中存在问题，则不允许向代码库中提交。

在事业部层面搭建持续集成服务器，配置专门技术人员，供所有有需求有条件的项目选用。

建立代码质量度量网站，用于收集项目的持续集成结果和代码中的问题，并对代码的稳定性和质量进行度量和分析，作为后续代码质量改善的输入。

◆ **主要收益**

成本节约：持续集成工具每天/每周会自动对版本库中的代码进行构建（包括从代码版本库中取得代码、开发包和动态库的最新版本，代码的编译，代码的静态工具检查等），整个构建过程不需要人工的参与，单次持续集成的效益已比较明显，并且聚少成多，多个项目为事业部带来的效益非常可观。

代码质量提升，并利于持续改善：代码库中的代码质量有了显著提升。项目中代码的稳定性有了显著的提高，版本发布前编译过程基本不会出现问题。项目中代码的稳定性得到了量化，通过持续集成工具，可以看到项目中的持续集成过程成功了几次，失败了几次，成功的次数越高，项目稳定性越好。

为整个事业部建立集成工作的高效管理：通过建立代码质量度量网站，并通过图表的方式清晰的展现项目持续集成的成功率，随时了解项目中代码的稳定性。给事业部层面的管理，提供了简洁有效的管理视图。

4.3 多级项目规划

4.3.1 定义和特性说明

图4-1 项目/产品规划图

多级项目/产品规划，在软件开发领域，是指以迭代开发为基础，形成多层次的、逐步细化的项目或产品计划。这些层层相关的项目/产品规划包括：项目/产品愿景、项目/产品路线图、版本发布计划、迭代计划、每日实现（Daily Scrums）。如图 4-1 所示。

完整的多级项目规划包含如上 5 个层面，仅包含版本发布计划、迭代计划有时也被称为两级项目规划。

4.3.2 应用说明

项目/产品愿景

在该计划阶段，首先，项目 Stakeholder、项目/产品负责人将参与并组成工作组，他们负责阐述项目的重要性、给出项目成功失败的关键标准以及项目整体层面”完成”的定义；在过程中，可以利用形成项目愿景的一些工具，包括愿景盒子（Vision Box）、业务收益矩阵（Business Benefits Matrix）、项目范围矩阵（Scope Matrix）、滑动器（Slider）、成本收益矩阵（Cost/Benefit Matrix）等；最后，项目愿景需要使用尽量简要的文档固定下来，并保证项目团队成员都能了解。该文档需要包括：

1) 当前的问题

2) 机会描述和理由（描述项目的重要性）

3) 项目的价值

4) 项目如何和组织的战略目标达成一致

5) 解决方案综述

6) 项目包含的关键功能

7) 项目必须服从的技术和约束条件

8) 项目范围

9) 项目的关键时间线

10) 项目收益分析

11) 项目和其他项目的依赖性

12) 项目的相关风险以及如何消除。

项目/产品路线图

项目/产品路线图主要描述为了达到产品愿景而需要交付的关键功能和特性，这些特性基本处于 Epic 和特性层面，不包括 User Story（用户故事）。它从时间的维度来表述对愿景的支持和实现。当项目/产品需要发布多个版本时，项目路线图就非常重要，否则，它和发布计划相同。项目/产品路线图由项目负责人和项目经理维护，并保持更新。通常，会形成路线图文档或幻灯片，使用大图标显示重要的里程碑、包含的功能和发布日期等，让所有项目/产品相关人员都清楚产品各个组件的可能发布日程。

版本发布计划

版本发布计划由团队成员和项目/产品负责人共同制定，并通过版本发布计划会议讨论通过。它包括了当前版本需要交付的、达成一致的关键功能，并经过优先级排序，可以包

含 Epic 和 User Story。版本发布计划中常使用的概念包括：故事点、迭代、团队速率和优先级排序。通常，项目/产品负责人提出本次版本发布的目标，团队成员根据目标和功能特性的重要性对故事进行排序，并依据团队速率决定本次发布需要包含的故事点。前几次版本发布使用估算值，其准确度随着项目/产品的时间持续而逐步精确。版本发布计划是具备适应性且可调整的计划，会随着项目演进而改变。

迭代计划

迭代计划是对版本发布计划的再次细化，同样由团队成员和项目/产品负责人共同制定，并通过迭代计划会议讨论通过。迭代会议负责两件事情：根据当前状态确定是否需要对版本计划做出更新；为当前的迭代制定迭代计划。迭代计划中常使用的概念包括：拆分 Epic 和 User Story、任务、任务估算。在迭代会议上，成员首先根据当前的项目变化对发布计划进行更新，然后根据更新后的、重新排序过的故事制定当前迭代需要完成的故事，并对这些故事进行详细的任务拆分。成员在认领完任务后，会对任务的实现时间做出估算，估算值需要具体到这些估算信息可以方便任何成员追踪任务的进度。

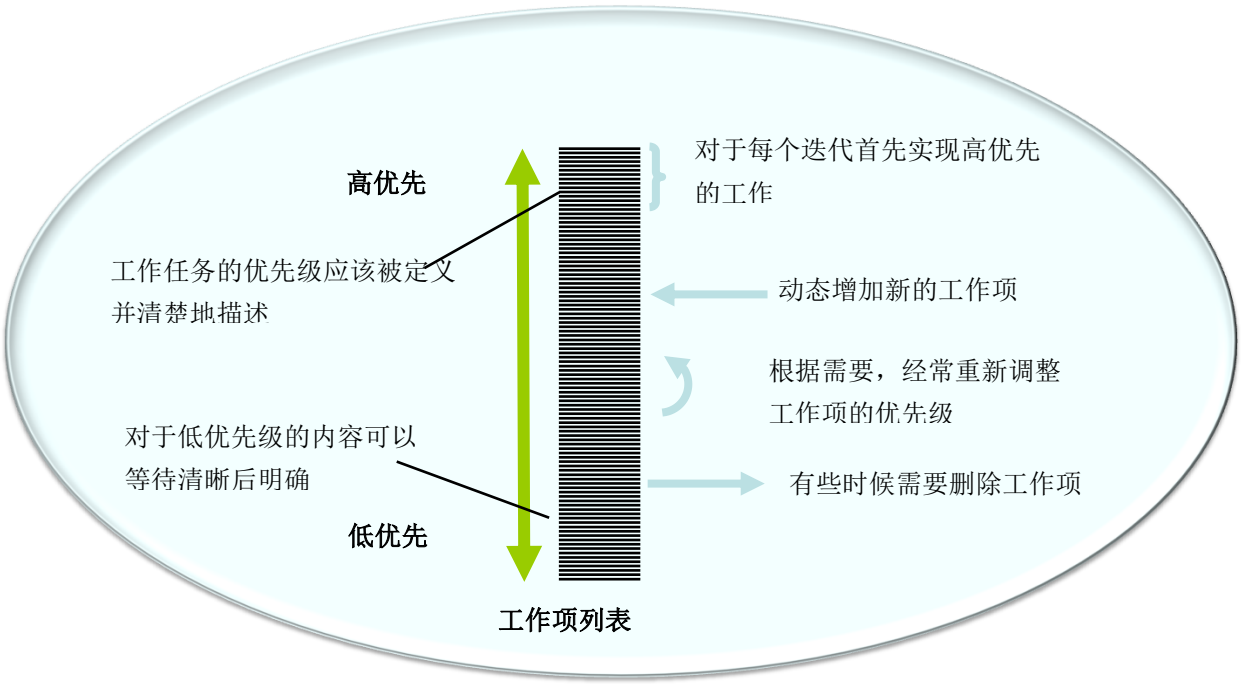


图4-2

每日实现

每日实现是团队成员完成任务的具体过程，它依据任务估算值并根据任务最终实现情况更新该值。在敏捷方法中，使用每日站立会议来报告进度，通过 15 分钟的站立形式，团队成员报告故事或者任务的完成、未完成状态，而解决层面的问题则在会议之后处理。

4.3.3 案例说明

案例：以医疗基层卫生服务系统产品研发为例

◆ **项目基本信息**

- 团队规模：20人
- 环境：JAVA、用友UAP

◆ **组织结构**

矩阵式组织架构，包括需求、UE/UI、开发、测试来自不同部门的人员组成产品团队参照 Scrum 组织架构，产品项目经理为 Scrum Master。

◆ **面临的问题**

产品研发计划总是不能得到正确执行，任务分配可能会有遗漏或责任人不清。

◆ **解决方法**

采用多级项目规划，每到一个阶段，将计划和任务分配细化并调整，产品研发进度与计划能够匹配，任务分配明确清晰。具体步骤：

1) 项目启动初期，由产品经理定制一级需求目标，描述产品要实现的业务场景或业务点，每一块完整的业务为一个独立内容。

2) 一级需求目标评审通过之后，由架构师或主设计师分解系统功能，给出二级开发计划，它描述对应的系统功能，并对每一个功能评估工作量，给出整体的开发计划，二级开发计划不指定每个任务的负责人。

3) 二级开发计划确定后，对其再进一步细化，明细到某一个菜单功能、按钮或后台算法。对每一个任务给出相对准确的工作量评估和完成时间（可以对二级开发计划的评估有调整），接着为每一个任务分配负责人员。

4.4 完整团队

4.4.1 定义和特性说明

Scrum 团队是基于功能开发而组成的跨职能、自我管理团队，在组织方式、管理模式和开发过程等方面与传统的开发团队有着重大改革：

- Scrum团队中不再有传统意义上的产品经理、项目经理、开发经理，而是引入了Product Owner、Scrum Master和Scrum团队等新角色，团队中通常会包含多个职责的成员，比如由需求设计、开发和测试人员共同组成的团队。
- 传统开发团队通常是项目经理下达工作内容，而Scrum团队提倡自我管理，按照兴趣和能力挑选任务。
- 从前的开发团队通常是接到任务后分头工作、独立完成，而现在Scrum团队需要相互配合工作，相互协作完成任务。
- 传统开发过程中，通常是经历一个大时间段的开发过程才完成一次产品发布，而Scrum开发过程中，产品是迭代增量发布的，通常是在每个Sprint结束时交付可发布软件。

4.4.2 应用说明

Scrum 团队必须具备以下三个完整性，才能算是一只完整的敏捷团队。

1) Scrum团队职责完整性

在一个标准的 SCRUM 团队中分为 3 种角色，分别是 Product Owner、Scrum Master 和 Scrum 团队。他们的职责如下：

Product Owner：需要确定产品的功能和完成时间，并对产品的收益负责，需要根据市场需求确定产品功能的优先级。Product Owner 主要工作是根据市场需求确定产品功能，将其列入 Product Backlog 中，并为这些功能确定优先级。

Product Owner 的角色通常由市场部门的人员来担任。

Scrum Master：负责监督整个 Scrum 项目进程，调整项目计划，确保开发团队成员的能力能够胜任开发，促使团队中不同角色能够进行充分沟通和交流，并负责为项目进程扫清障碍，保障每日的开发进度及一些日常开发管理工作。

Scrum Master 通常由开发组组长担当。

Scrum 团队：一般由 5-10 个全职的成员组成，团队成员横跨多个职能，一般包括开发、需求、测试人员，大家在弱化分工，每个人都参与设计、开发与测试中，这也是团队职责完整性的一种体现。

2) Scrum团队素质完整性

首先，要具备很强集体协作精神。敏捷开发提倡的一个重要思想就是集体协作，即使个人能力再强，不懂得集体协作，这种人也不是敏捷开发团队所需要的。

其次，Scrum 团队的成员需要具体良好的沟通能力。敏捷开发中最强调的就是沟通，最有效的沟通方式就是面对面交流，那种只会埋头工作而沟通能力不强的员工，在敏捷开发团队里也是吃不开的。

第三，Scrum 团队的成员必须能积极主动的接受新的事物，要具备创新能力。敏捷开发与传统的开发模式最大的优势就是拥抱变化，面对时时变化的客户需求，那些不能及时转变思维、墨守成规的成员是不能胜任的。

最后，Scrum 团队的成员要具备极强的自我管理能力和积极主动的精神。自我管理是敏捷开发团队中不可或缺的素质，那些工作时只会被别人引导而不能主动自我管理的人也是不适合的。

3) Scrum沟通完整性

Scrum 团队除了日常的工作之外，有三个重要的会议也是要坚持的，这三个会议召开的好坏，直接影响到 Scrum 开发过程的效果。这三个会议分别是 Sprint 启动会、每日站立会议和 Sprint 回顾。

首先来介绍一下 Sprint 启动会,这个会议主要目的是要根据 Product Owner 制定的产品或项目计划在 Sprint 的开始时做准备工作，通常是由 Product Owner 根据市场的前景和商业价值制定一个排好序的客户需求列表，这个列表就是 Product BackLog。当 Sprint backlog 确定后，Scrum Master 带领 Scrum Team 去分解这些功能点，细化成 Sprint 的一个个任务. 这

些任务就是细化的来实施这些功能点的活动。 Sprint Planning 的这个阶段需要控制在 4 个小时。

其次，每日站立会议，顾名思议是每天开站会。这个会议是让团队成员能够面对面的在一起，同步目前的工作进度和状态。通常每个成员需要轮流回答”昨天我做了什么”、”今天计划做什么”和”遇到了哪些困难”三个问题。这个会议一般要严格控制在 10 分钟左右，不宜过长。

最后，在一个迭代周期结束的时候，要召开迭代回顾会。这个会议首先要演示本迭代完成的工作内容，需求、开发和技术人员要做相关的评审工作，由 Product Owner 来确定完成了哪些功能，哪些工作还没有完成。另外，整个团队还要回顾本迭代内哪些工作完成的好，继续发扬和保持下去，哪些工作完成的不好，需要进行什么样的改进。

这三个会议是 Scrum 开发过程中不可缺少的重要组成部分，一定要持续的将这些会议坚持下去，才能真正体会到敏捷开发带来的好处。

4.4.3 案例说明

案例：来自用友软件银行客户事业部的案例

◆ **项目基本信息**

用户软件银行客户事业部，为 XX 银行开发一套资金交易系统。面临困难：

1) 开发过程周期短，只有3个月时间；

2) 客户很难一次提足需求，由于对类似的系统没有概念，对系统的功能定位及业务描述很难到位；

3) 业务逻辑复杂，对产品的正确性和稳定性要求高。

按照以往的经验，在这么短的时间内，开发出一套这样的系统几乎不可能完成，需要我们能够抛开以前的开发模式，按照敏捷开发的模式组建开发团队，迅速交付产品。

◆ **组织结构**

采用标准SCRUM架构，团队由需求、设计、开发、测试等角色共计6人组成。

◆ **开发过程**

1) Product Owner需要确定产品的功能和完成时间，确定整个开发过程分成12个单周迭

代。并把需求分解为Product BackLog,维护进Scrum Works；

2) 每个迭代开始的第一天上午，所有团队成员参加Sprint启动会，Product Owner首先从Product BackLog中挑选定本迭代需要完成的Backlog。然后由Scrum Master带领团队成员来细化任务，分配任务；

3) 开发过程中每天坚持召开站立会议，会议上每个成员通过回答“昨天我做了什么工作”、“今天计划做什么工作”、“遇到什么问题”来同步项目进度和解决工作困难，会议还可以根据最新的开发状态来及时调整开发计划；

4) 开发人员在开发过程中，及时更新ScrumWorks的任务状态。团成员可以通过燃尽图来了解本迭代的任务还有多少没有完成；

5) 每个Sprint结束的最后一个下午，召开Sprint回顾会。会上首先演示本迭代完成的功能，由Product Owner来确定本迭代的目标是否已经完成。如果有未完成的任务，把这些任务排进下一个迭代计划。同时回顾本迭代中有哪些工作完成的比较好，哪些不足的地方需要改进；

6) 每个迭代结束后，把这个迭代完成的成果交给客户做需求确认，如果不满足客户需求及时在下一个迭代中修改，避免在项目后期产品交付的时候与客户需求有大的偏差；

7) 每个迭代持续进行，直到产品完成开发结束，完成交付。

◆ **面临的问题**

1) 敏捷开发提倡先完成市场收益率最高的工作，这样容易导致敏捷团队过于看重眼前的利率，而忽视了产品发展的长远目标；

2) 敏捷开发团队对团队成员素质要求很高，要求每名团队成员都要非常专业并且要能够积极主动的自我管理。但面临的现状往往是团队成员水平参差不齐，工作的态度也是因人而异，往往不能达到敏捷开发要求的标准。

◆ **解决方法**

1) 敏捷团队应该放眼长远目标，不能只关注对眼前收益率大的目标；

2) 敏捷开发的价值观提倡谦逊和勇气，团队成员互信互助，而不是互相指责批评；自己能够认识到自己的不足，并且主动要求进步，遇到困难的时候主动寻求团队帮助。

◆ **主要收益**

1) 团队自我管理，团队成员的责任感和主观能动性增强；

2) 团队成员不再各自为战，团队联系更加紧密；

3) 每个迭代都将成果与客户做确认，避免了项目后期较严重的需求变更的风险；

4) 不断的自我反醒，自我激励，成员能力得到提升。

4.5 ATDD (验收测试驱动开发)

4.5.1 定义和特性说明

ATDD 是测试驱动开发核心理念的一个扩展，是指在用测试驱动开发实现某个具体功能之前，首先编写功能测试或验收测试用例，从系统功能角度驱动开发过程。

ATDD 能有效促进客户、测试人员、开发人员之间的交流协作。通过增量式开发软件，用面向客户的测试作为讨论和反馈的基础，ATDD 能使整个团队紧密协作。依照客户指定的优先级，在整个项目过程中不断开发出可用的具体功能，赢得客户信任并增强自信，交流也因相互之间有了共同语言而更加高效。

4.5.2 应用说明

ATDD 周期包含挑选用户故事，为故事写测试，实现测试以及最后实现故事这四个步骤。

1) 挑选用户故事

迭代开始前的计划会议，这个会议中，客户会决定下一个迭代该做哪些故事，及给故事排优先级。一般情况下可先选择较高优先级的用户故事。

2) 为故事写测试

在为故事编写测试的时候，客户是最适合写验收测试的人，而对于实际情况因为客户参与度和技能问题，可行的方式是由客户、开发人员、测试人员共同参与编写验收测试，给故事拟出一系列测试。在编写测试用例时只关注完成故事必须要做的几件事情，形成简单清单，以后在具体实现故事或者验收测试时再细化测试，添加更多细节，讨论各部分如何工作，确定客户对界面是否有特别的要求等。根据功能的类型，对应的测试可以是一组顺序操作，或者是一组输入及对应的输出。另外在故事的实现过程中还难免会发现原有测试

外的新的测试，合理的做法是，在征求客户意见后，我们应该把新的验收测试加入到清单中。

3) 实现测试

在完成验收测试编写后，就需要把验收测试转化成可执行的验收测试，这时候，团队常会引入一些相对成熟的测试框架及商业化自动化测试工具。这些框架和工具一般可划分为API 级别功能测试（如采用 Fit 和 FitNesse 框架）和自动化 GUI 测试（如可采用关键字驱动框架和商业自动化工具等），团队可以根据开发的应用及团队自身情况选择使用其中一种或组合使用。API 级别功能测试，使团队能够在不牵涉 UI 层的情况下测试业务逻辑。而自动化 GUI 测试在实际应用中，团队所采用的自动化测试框架（通常需要团队自行开发）针对 GUI 测试的成熟度尤其重要，良好框架可通过测试用例、对象、数据的分离，大大减少因为频繁的界面变化而对已有自动化脚本的影响。

4) 实现故事

实现新功能，让新加入的验收测试通过。ATDD 本身并不会限制实现功能的方式，不过使用 ATDD 的人通常倾向于在实现过程中采用 TDD 方式。这样在代码层面，采用 TDD 方法以测试驱动的方式编写代码。在软件的特性和功能层面，使用 ATDD 方法以测试驱动的方式构建系统，这两个不同层面上结合使用测试驱动，既能保证软件的内部质量同时能保证开发出的软件能满足正确的功能需求。

4.5.3 案例说明

管理平台、B/S 架构、使用迭代式软件开发，迭代周期为 3 周，要求测试团队能迅速反馈系统质量。

◆ **项目基本信息**

- 团队规模：10人
- 环境：Java

◆ **组织结构**

标准Scrum组织架构

◆ **面临的问题**

难以有效开展系统功能测试，每次迭代后界面变化比较大，传统自动化工具录制的脚本维护工作量大，难以快速反馈系统质量状况。

◆ **解决方法**

采用验收测试驱动开发实践，引入针对业务功能的自动化 GUI 测试（关键字驱动框架+IBM Rational Functional Tester）。

◆ **主要收益**

1) ATDD保证故事实现质量的及时反馈，保证开发出的软件满足正确的功能需求；
2) ATDD通过编写验收测试清晰定义了故事完成标准，让团队“知道我们在哪儿”及“知道何时停止”；
3) ATDD让团队（客户、开发人员、测试人员等）为一个目标努力，创造了协作性更强环境。

4.6 结对编程

4.6.1 定义和特性说明

在敏捷软件开发的各种实践中，结对编程（Pair Programming）是特别有争议的，尤其是在国内的软件企业或团队，几乎看不到有真正实践、施行，并带来效果的。相比于其他敏捷软件开发方法的火热及它们带来的成效，结对编程可以说备受冷落。它看上去很美，但想要成功的实践，可谓障碍重重。那么实用为王，我们是不是可以秉承结对编程优秀的核心理念和思想，而将它的实践框架加以改造，让它更适合我们的工作流程和习惯，最终成功实践并提升研发效率和质量呢？

传统结对编程定义及特性

两位程序员形成结对小组，肩并肩地坐在同一台电脑前合作完成同一个设计、同一个算法、同一段代码或同一组测试。

结对编程可以看作是一种敏捷化的代码检查（Code Review），其最终目标是提高软件产品的质量。代码检查工作是公认的提高软件产品质量的重要活动之一，但在实践中，成功应用的也并不多见，原因是传统的代码检查方法只能在某个功能完全开发完毕后才进行检查，效率不高。而更大的问题是，检查人对被检查人的代码所实现的业务功能并不十分

清楚，所以只能检查代码的格式、语法是否规范，对实现逻辑是否满足业务需求往往无法检查。这导致了很多企业和团队的代码检查都流于形式。

结对编程就解决了这一问题，它不需要代码都写完才开始检查，而是在两个人相互协作过程中，实时的进行多次交叉检查，大大提高了效率，而且因为两人同时完成设计和代码，对业务需求也都了解，所以检查时可以验证代码是否满足了需求、是否符合业务逻辑。但是，传统的结对编程模式也有一些显而易见的问题导致其难以实践，例如人们担心两个用一台电脑做同样一件事情，是不是浪费了人力资源？两个人的个性、习惯、水平都不同，在对等的地位上同时做一件事情会不会产生冲突和矛盾？他们之间的沟通和协作会顺畅吗？会不会感到不自在？

新结对编程定义及特性

两位程序员形成结对小组，每人一台电脑，坐在临近的工位上，两人合作完成一组功能（可以是两个或多个独立的模块）的设计、代码实现。但对于某一个模块来说设计和代码是分开的，一个人负责设计，另一个人负责写代码，对于其他模块则反之。当某个功能阶段性完成后，由其设计人员对代码进行检查。目前，很多敏捷开发的倡导者们都在积极的寻找可实践的结对编程的变形体，以打破结对编程这种看上去很美，摸上去扎手的情况。而上述这种形式的新结对编程，也是我们几经探索总结出来的一种可实践的结对编程形式，并在产品团队中成功应用。它继承了结对编程的核心理念，让代码检查更加高效、深入。也在两人合作模式上，更适应实际工作情况和工作习惯，以做到真正的可实践、可应用。

4.6.2 应用说明

在可实践的结对编程中，两名开发人员结对形成一个小组，临近而坐，共同负责一组功能模块，在具体的某个功能或任务上，两个人所扮演的角色是有承接关系的，一个是设计者、另一个是开发者，一个是审查者、另一个是被审查者。在单个点上，两人互不干扰，独立运行，都有自己的发挥空间。而在在总体上，两人又形成一个整体，要分别对对方的设计和代码熟悉了解、审查及负责，而且两个人的关系又是平等的，因为设计与开发、审查与被审查对于两个人来说都是相互的。这样就能够让两人在工作中共同协作，提高效率和质量，而且又同时避免了一些可能会产生的冲突与矛盾。

结对编程中的设计环节，也是敏捷化的。敏捷开发与传统瀑布型开发一个很重要的区别就是：在研发活动中各环节的流转的承接时，瀑布型开发是以文档为主，思想的交流与沟通为辅。而敏捷开发是以思想的交流与沟通为主，文档为辅。所以在瀑布型开发中，对设计文档的格式和规范等要求比较高。而敏捷开发中的设计文档，只是要描述清楚设计思想，并作为一个辅助手段，将关键内容存留作为依据即可，不需要规范的模板。可以根据设计者的习惯，以最高效的形式描述清楚设计内容，并且清晰明确即可。在结对编程的实践中，所涉及到的设计文档的书写，都是以此种方式进行的。

4.6.3 案例说明

案例：来自用友医疗基层卫生服务部门的案例

◆ **项目基本信息**

用友医疗基层卫生服务系统产品

- 团队规模：20 人
- 环境：JAVA、用友 UAP 平台

◆ **组织结构**

矩阵式组织架构，包括需求、UE/UI、开发、测试来自不同部门的人员组成产品团队，参照 Scrum 组织架构，产品项目经理为 Scrum Master。

◆ **面临的问题**

产品开发人员对相互业务不了解，产品代码质量不高，如果团队人员变动，开发活动接续困难。

◆ **解决方法**

采用变形的新结对编程方法。举例说明，两个开发人员甲和乙，形成结对编程小组，共同负责两个功能 A 和 B。工作开始，甲进行 A 的设计工作，并给出设计文档。乙进行 B 的设计工作，并给出设计文档。两人的设计工作完成后，互相阅读对方设计文档并进行交流，两人要保证对对方的设计内容了解清楚，并指出对方文档中描述不当的地方让其修改，最终保证设计内容是清晰明确的，而且甲、乙两人对 A、B 功能的设计都熟悉明确并思想一致。接下来开始代码开发工作，甲按照乙的设计文档开发功能 B，乙按照甲的设计文档开

发功能 A。代码开发工作进行到某个段落后，甲对乙开发的功能 A 代码进行审查，检查代码格式和规范以及代码实现是否符合甲设计的业务逻辑，如有问题，则请乙进行修改并复查，乙同样反之对 B 进行审查，最终达到代码开发完毕，A、B 两个功能都进行过多次审查和修改，而且甲、乙两人都确认两个代码是合格的，并符合业务需求的。这样，功能 A、B 的设计和代码实现就是有甲、乙共同协作完成的，两人共同对两个功能负责，并且两人对两个功能的设计和实现都清楚，更为重要的是甲、乙两人的工作是相互并行配合的，不会产生冲突，更为容易推广施行。

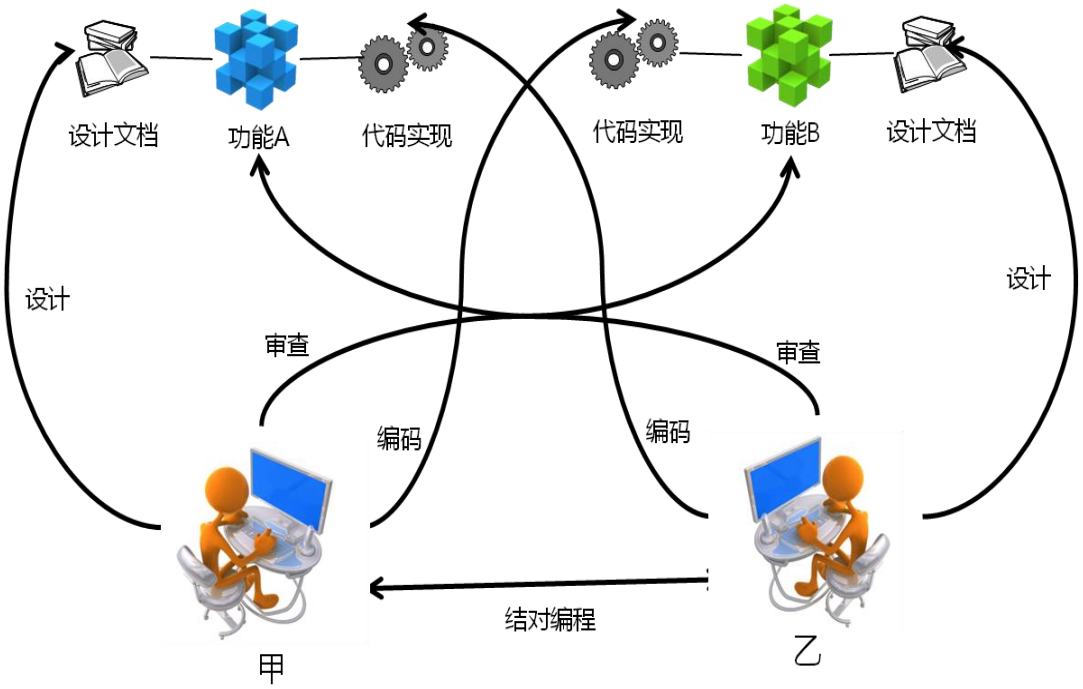


图4-3

◆ **主要收益**

按照可实践的结对编程模式完成软件开发，不仅高效的完成了代码检查，提升了软件质量，还带来了一些额外的好处：

1) 每个模块至少两个人熟悉，这样在有人请假的时候都有backup，不会造成严重的耽误工作进度。如果有人要离职，交接工作通常也会非常轻松，几乎没有什么需要特别交接的。

2) 进行结对的两个开发人员相互阅读设计文档和代码实现，实际也是一个相互学习的

过程，所以结对编程也起到了一些知识传播、培训的作用。

4.7 确定冲刺计划

4.7.1 定义和特性说明

冲刺会议的目的：Scrum 团队和产品负责人（Product Owner）共同决定在接下来的冲刺周期内的目标以及哪些功能和任务需要完成。

冲刺会议参加的最主要角色：Scrum 团队，产品负责人以及 Scrum Master。

冲刺会议的主要输入：Product Backlog、团队的能力。

冲刺会议的主要输出：Sprint Backlog。

4.7.2 应用说明

冲刺会议最主要的参加人员是 Scrum 团队，产品负责人以及 Scrum Master，同时感兴趣的管理层或者客户代表也可以参加。

整个冲刺会议分为两个部分：

第一部分：解决本次冲刺要完成哪些需求；

第二部分：解决这些选择的需求如何被完成。

在第一部分的冲刺会议中，产品负责人向团队描述最高优先级的一些功能，团队成员详细询问功能涉及的各个细节，并决定哪些功能可以从 Product Backlog 放入在 Sprint Backlog 中完成。产品负责人和团队共同定义本次冲刺的目标，并最终在冲刺结束的评审会议中以冲刺目标是否被完成作为本次冲刺是否成功的标准。在第二部分的冲刺会议中，Scrum 团队集体讨论放入本次冲刺的需求如何实现，并决定最终有多少需求可以承诺在本次冲刺内完成。

4.7.3 案例说明

◆ **项目基本信息**

互联网新产品的研发，典型的 B/S 应用系统，每个月发布一个新的版本，得到客户反馈后调整需求列表并金星下一次的迭代。

◆ **组织结构**

8 个人的团队，1 个 DBA，1 个架构师，5 个开发人员，1 个测试人员。

◆ **面临的问题**

每次的冲刺会议可以让团队更好的了解需求，并提出更加合理的承诺。

◆ **解决方法**

1) 冲刺会议的第一部分

- PO会向团队介绍半年之内的Roadmap；
- 团队内部进行分组，明确至少两个人对某个或几个功能重点关注（对相关业务熟悉的和不熟悉的组成一组）；
- PO向团队介绍优先级最高的一些需求背后的用例，包括正常流程、异常流程。此时团队有任何细节问题都可以向PO提问以获得答案。界面原型也是讨论的一个方面。以报表为例，此时就增加了如何排序等细节。同时完成各个需求的最主要验收标准。

2) 冲刺会议的第二部分

- 按照Sprint目标中的各项需求拆分出相应的任务，确保考虑到工作中三个关键的细节：编码、测试和文档；
- 如果任务需时超过一天，尝试将此任务分解为几个小任务，任务的分解周期是2～8小时。如果有多个任务非常小，例如Bug修改，可以几个合在一起作为一个较大的任务。任务完成的时间是在以没有任何外界干扰为前提，例如额外的技术支持等。
- 通过分解任务判断需求（Story）最后的工作量评估；
- 功能通过认领的方式落实到人。关键的里程碑、check point也已经确认，目的是让团队所有人对项目的进展有总体的把握；
- 团队确认Sprint的承诺（commitment）。

◆ **主要收益**

团队的承诺更加科学，同时和 PO 目标一致，可以形成更多的信任

4.8 故事点估算

4.8.1 定义和特性说明

故事点是表述一个用户故事，一项功能或一件工作的整体大小的一种度量单位。当使用故事点进行估算时，我们为待估算的每一项设定一个数值。这个值本身的数字并不重要，重要的是这些项之间通过各自数值对比体现的相对大小。例如，一个被赋予”2”的用户故事，其大小应当是一个被赋予”1”的用户故事的两倍。

当使用故事点来估算用户故事的大小时，并没有固定的公式来规定如何计算故事点的数值。故事点估算用于评估为了交付一个用户故事所包含的所有努力（team effort），用户故事的复杂度(complexity)，风险，以及所有其他需要考虑的元素。

— Agile Estimating and Planning, Mike Cohn

4.8.2 应用说明

在 Scrum 中，最典型的使用故事点做估算的方法是计划扑克（Planning Poker）。同时也有另一种更新的估算方法－敏捷估算 2.0（Agile Estimating2.0），也正被越来越多的 Scrum 团队采用。

计划扑克（Planning Poker）

计划扑克由 James Grenning 在 2002 年首次提出。计划扑克集合了专家意见（Expert Opinion），类比（Analogy）以及分解（Disaggregation）这三种常用的估算方法，使团队通过一个愉快的过程快速而准确的得出估算结果。

计划扑克的参与者是团队的所有成员，包括了程序员，测试，数据库工程师，需求分析，用户界面设计师等等。典型的敏捷团队一般不超过 10 人，如超过则一般可以拆分成为两个小团队各自独立进行估算。Product Owner 也需要参加，但不参与估算。

计划扑克开始时，每个参与估算的组员都会得到一副计划扑克，每一张牌上写有一个 Fibonacci 数字 （典型的计划扑克由 12 张牌组成：？，0, ½，1, 2, 3, 5, 8, 13, 20, 40, 100,∞，其中？代表信息不够无法估算，∞代表该用户故事太大）。

开始对一个用户故事进行估算时，首先由主持人（通常是 Product Owner 或需求分析师）介绍这个用户故事的描述。过程中 Product Owner 回答组员任何关于该用户故事的问题。展开讨论时组持人应注意控制时间与细节程度，只要团队觉得对用户故事已经了解到可以作出估算了，就应当停止讨论。

所有问题都被澄清后，每一个组员从 扑克中挑选他 / 她觉得可以表达这个用户故事大小的一张牌，但是不亮牌，也不让别的组员知道自己的分数。所有人都准备好后，主持人发口令让所有人同时亮牌，并保证每个人的估算值都可以被其他人清楚的看到。

经常会出现不同组员亮的牌差距很大的现象。当出现有很多不同分值的时候，出分最高的人和出分最低的人需要向整个团队解释出分的依据。主持人需要注意控制会议氛围，避免出现应意见不一导致的攻击性言论。所有的讨论应集中于出分者的想法是否值得团队其他成员进行更深入的思考。

随后全组可以针对这些想法进行几分钟的自由讨论。讨论之后，团队进行下一轮的全组估算。一般来说，很多用户故事在进行第二轮估算的时候就能得到一个全组统一的分值，但是如果不能达到全组意见一致，那么就重复的进行下一轮直到得到统一结论。

敏捷估算 2.0（Agile Estimating 2.0）

计划扑克是 Scrum 团队应用最广泛的敏捷估算方法，但是有时候计划扑克玩起来耗费比较多的时间，尤其是在十人左右的团队中。Ken Schwaber 在他的书《Agile Project Management with Scrum》中指出，在进行 Sprint 规划时，Sprint Planning 环节应该控制为一个 4 小时的时间盒，但是从战术的角度看，如果一个会议持续 4 小时，大部分的参会者会有精疲力竭的感觉，并且很难保持持久的注意力。

为了解决这个问题， Brad Swanson 和 Björn Jensen 在上海 Scrum Gathering（2010/4/19）上介绍了 Agile Estimating 2.0 技术。这个新的估算技术同样基于专家意见，类比和分解，同样适用 Fibonacci 数列，但是它可以显著的缩短会议时间。

第一步是由 Product Owner 向团队介绍每一个用户故事，确保所有需求相关的问题都在做估算前得到解决。

然后整个团队一起参与这个游戏。只有一个简单的游戏规则：一次仅由一个人将一个用户故事卡放在白板的合适位置上：规模小的故事在左，大的在右，一样大的竖向排成一列。整个团队轮流移动故事卡，直到整个团队都认同白板上故事卡的排序为止。

第三步，团队将故事点（Story Point）分配给每个用户故事（列）。最简单的做法是使用投票来决定每个用户故事分配到哪一个 Fibonacci 数字。

还有最后一步：使用不同颜色来区分影响估算大小的不同方面，并且重新考虑是否需要修改估算值。例如，我们使用红色来表示那些无法被自动化测试脚本覆盖的用户故事，因此那些用户故事需要一个更大的数字来容纳手工回归测试的代价。

在多次实践 Agile Estimating 2.0 之后，我们对于结果还是相当满意的。团队对于估算的准确性更有信心了，并且只耗费原先的 1/2 时间。Agile Estimating 2.0 的原版材料如下：

http://properosolutions.com/wp-content/uploads/2010/05/agile_estimation_2.0-for-pdf.pdf

4.8.3 案例说明

案例1

- **项目基本信息**

B2B 电子商务系统，集成的开发 / 测试团队（7 人），一周的迭代，标准 J2EE 开发技术。

- **组织结构**

标准 Scrum 结构

- **面临的问题**

Sprint 计划中估算不准确。

- **解决方法**

使用 Planning Poker 进行故事点估算。

- **主要收益**

故事点估算较时间估算更准确，得到团队 velocity 的数据，全组目标一致每个 Sprint 提高 velocity。

案例2

- **项目基本信息**

在线教育平台，大型团队（共七个团队），三周的迭代，.NET + ETL + Reporting。

- **组织结构**

Scrum of Scrum

◆ **面临的问题**

Sprint 计划会议时间太长。

◆ **解决方法**

使用 Agile Estimating2.0 进行故事点估算。

◆ **主要收益**

对于单个团队运行的三周的 Sprint，Sprint Planning 会议时间由 3 小时缩短到 1.5 小时，估算更准确。

4.9 需求订单

4.9.1 定义和特性说明

需求订单，也可称为产品订单，在 Scrum 中，称为 product backlog。是一张记录用户需求的列表，包括产品所有需要的特征。需求订单中的每一项通常包含了需求标题、描述、重要性以及故事点数或其它表示大小的数字。

需求订单是开放的，团队每一个成员都可以编辑和维护。

在整个项目开发生命周期内，需求订单需要不断维护，管理与跟踪需求变更。

4.9.2 应用说明

建立一张准确的需求订单有助于团队成员了解项目概况，理解具体需求。同时各相关人可根据需求订单了解项目进展，以及产品的概貌。

在项目开始初期,通常只记录最粗粒度的需求并给出大致的优先级。优先级是一个数字，数字越大代表优先级越高，初始给出的优先级间应留有一定的空间，便于日后添加介于两者优先级之间的需求。在项目初期不细化需求的原因在于避免将时间用于暂时不开始的需求，以及避免日后需求变更所导致需求重新分析所带来的时间浪费。

在迭代开始前，按优先级选择需求，如果需求粒度太粗，则先细化需求。确定一个需求粒度是否太粗，可有以下标准：

1) 开发人员无法确定其大小；

2) 其大小大于某个特定数字（以故事点为单位的某团队认为6是最大可接受的需求粒度）。

需求细化选定之后，还想讨论和澄清更多的细节。通常一个 4 周的迭代前，需要约 4-8 小时的讨论，以达成共识。需求讨论得越细致，产品最终和用户的需求越接近，也就避免了由于需求理解偏差而产生的返工，变更所消耗的时间。最终，需求细节推荐记录在需求订单中。

在迭代过程中及迭代后，如有新需求，可将需求加入需求订单。

4.9.3 案例说明

案例1：来自汤森路透的某项目

◆ **项目基本信息**

在某自动化处理配置应用中，通过制定、维护需求订单，分解了需求的同时，使得开发人员对项目有了总体的了解，避免了需求理解偏常以及需求变更所造成的浪费。

◆ **组织结构**

标准 Scrum 组织架构

◆ **面临的问题**

项目大，需求粒度大，变更多，需求需要进一步挖掘。

◆ **解决方法**

维护需求订单；需求早期为粗粒度需求，迭代开始前细化高优先级需求，基于细粒度需求详细讨论。

◆ **主要收益**

由于有维护良好的需求订单和充分的沟通，项目开发工程中基本没有返工，以及不必要的资源浪费，及时满足了客户的需求。

案例2：来自石化盈科的离线油品调合软件

◆ **项目基本信息**

在离线油品调合软件开发项目中使用 Scrum 模型，每次迭代版本周期为 2 周左右。项目基本信息如下：

- 团队规模：9人
- 环境：vs2010 & sqlserver2008

◆ **组织结构**

Scrum 组织结构：1 名 Product Ower,1 名 Scrum Master 及 7 名团队开发/测试人员。

◆ **面临的问题**

早期需求不稳定，用户意见零散，业务目标不明确。传统的需求文档表述繁琐，后期维护的工程较大。

◆ **解决方法**

用 Product Backlog 作为产品的需求规格。周期性收集用户的业务需求并表述为用户故事。与团队成员一同建立和细化用户故事。每一次迭代结束后，与用户确认，对需求订单进行维护。

◆ **主要收益**

用需求订单（用户故事）取代传统的需求规格说明书。表述清晰明确，团队成员易于理解，易于维护。

4.10 燃尽图

4.10.1定义和特性说明

燃尽图是在项目完成之前，对需要完成的工作的一种可视化表示。燃尽图有一个 Y 轴（工作）和 X 轴（时间）。理想情况下，该图表是一个向下的曲线，随着剩余工作的完成，”烧尽”至零。燃尽图提供工作进展的一个公共视图。一般的，如果没有特别说明，燃尽图指的是反映一个迭代（或 Sprint）之内的情况。

燃尽图主要有 2 类，第 1 类是燃烧剩余工作量；第 2 类是燃烧剩余工作多少，常见的是燃烧故事点。

燃尽图的更新频率一般是每天一次。

燃尽图具备以下几种效果：

1) 具备可视性，直观展示进度情况；

2) 具备风险预估，提醒团队目前完成情况；

3) 具备可估量，直观显示当前还需要的时间。

也可将燃尽图用于跟踪整体宏观情况，比如每个迭代有一个点，这时可以称呼此燃尽图为迭代燃尽图或项目整体燃尽图。

4.10.2应用说明

燃尽图的绘制需要与迭代估计和定期跟踪（一般是每日，也有与每日站会一起）配合使用。

燃尽图的常见问题：

1) 无法正确的填写燃尽图

在敏捷施行之初，这个问题较为严重，首先是谁去填写这个燃尽图的问题，燃尽图无法反应出任何一个人的情况，往往无法明确真实的完成情况，而仅凭讲解与说明得出结论，因此源头在于团队成员没有完全把握工作完成情况与估计。这在初期时比较明显，甚至对于新员工来说。

因此这个问题不可避免，要做的让所有成员能够认真对待完成情况，并且进行反思，树立自己的对完成情况估计的信心，学会预计风险，特别是对于团队核心成员来说，这样可以让管理层对整个团队树立信息，认为成员反馈的信息是正确，并且会正视对待看到的燃尽图，这样的燃尽图才真正有意义的。

缓解这个问题的一个办法是利用用户故事或其它相对客观的条目，配之与完成的定义（Definition of Done），当具体条目满足完成定义时，可以把相关任务或故事点燃尽。但这个办法对于具体条目满足完成定义之前仍然没有好办法，仍然需要依靠上文所提的提升成员把握工作完成情况的能力。

2) 只关注开发工作量，还有测试工作量

在一个开发团队当中可能包括一个测试人员，并且开发本身有自己的测试工作量，因此燃尽图统计的应该是所有成员所有工作的工作量与完成情况，甚至有临时的 Task，因此图表随时在变化，而往往这些造成某些任务无人认领的情况，这些都应该认真考虑与完全考虑，并且填入工作量。

3) 中途加班

燃尽图关注的是剩余工作，加班后，需要重新估算后续所需的工作是多少，并且把新的估算反映到燃尽图中，极有可能出现燃尽曲线不降反升的情况。

4) 进度变动如何控制

对于讲究时间箱的敏捷而言，如果在燃尽图上判断出时间节点到达时可能无法开发完所有原计划的功能，首先推荐的是把预计无法完成的功能放到下一个迭代继续开发，这样做的话，在功能变化前后，燃尽图会出现快速下降两个点。如果但是如果有外部事件的重要时间节点，可以适当调整时间窗使得某个迭代结束时与外部吻合，这样做的话，燃尽图会在 X 方向上延长，绘制方法不需要变化，但需要关注根据燃尽图预测完成的时间点是否符合要求。

常见工具：

1) 白板手绘；

2) Scrumworks：提供了较强大的可视化报表功能，整个产品的燃尽图可以通过任务板中任务状态变化自动形成。

4.10.3案例说明

案例1：用友承担的某银行资金交易系统

◆ **项目基本信息**

在 XX 银行的资金交易系统开发过程中，需要一种能够及时展示产品开发完成情况与整体进度的管理工作报告，我们在开发过程中使用了燃尽图作为解决方案。通过使用燃尽图来展现项目进度和完成情况，使团队管理层很容易地通过燃尽图了解实时的进度以及细节，管理层更能够实时把握产品的进度并做出正确的决策，并且预计风险，同时随时调整计划。

- 团队规模：6人
- 环境：NC5.6平台 & ORACLE10G

◆ **组织结构**

标准SCRUM架构

◆ **面临的问题**

无法正确的填写燃尽图，燃尽图的表象不一定能反映项目进度的真实情况。

◆ **解决方法**

绘制燃尽图，推进各成员掌握。

◆ **主要收益**

通过使用燃尽图可以使团队成员更容易容易了解项目进行的真实进度和细节，最大程度上实现了项目的可视化。

案例2：EA的某企业应用程序开发

◆ **项目基本信息**

企业应用程序开发(Customized Ent App) + ETL + Reporting

- 团队规模：23人, 3个Scrum团队
- 环境：.NET, ETL (Informatica), Reporting (Cognos)

◆ **组织结构**

Scrum of Scrum

◆ **面临的问题**

基于 Ideal Hours 的燃尽图不能准确体现真实状态

◆ **解决方法**

1) 使用故事点估算；

2) 为每个故事定义清晰的完成标准(Definition Of Done)；

3) 使用基于故事点的燃尽图。

◆ **主要收益**

准确，清晰的进度度量。

4.11 每日站立会议

4.11.1定义和特性说明

团队每天站着召开的短时间会议称之为每日站立会议，也简称为”每日站会”。

每日站立会议旨在让团队统一目标，协调团队内部问题的解决，绝非进度汇报。

会议主持人（比如 Scrum Master、轮值者、教练、团队协调者）确保会议的举行，并控制会议时间，团队成员进行简短有效的汇报。

会议上每个成员需要回答 3 个问题：昨天都完成了哪些工作？今天准备完成什么？工作中遇到了什么问题？

回答的形式与目的不是向领导汇报工作，而是团队成员之间相互交流，以共同了解项目情况和共同解决问题。

每日站立会议的时间一般不超过 15 分钟。

团队外成员也可以参与，但没有发言权。

每日站立会议是在 Scrum 中的明确的要求，在 Scrum 中，也被称为 Daily Scrum。

通过每天面对面的沟通可以：

1) 快速同步进度，让组内成员相互了解彼此进展，从而了解本项目的整体进展；

2) 给团队成员一种精神压力，要对每日的工作目标信守承诺；

3) 培养团队文化，让每个人意识到我们是一个团队在战斗。

4.11.2应用说明

每日站立会议要求有效、快速，团队需要全部的参与。在每天的固定时间固定地点来召开每日站立会议。成员在回答三个问题时目光要注视着大家，而不是只向会议主持人，避免变相为向领导汇报工作。对每个人回答的问题有疑问，其他成员都可以提出，而不是只有会议主持人一个人在问。成员提出的问题或困难其他成员要认真倾听，确定相关的负责人和协作方式，但问题细节不在会议上讨论。每日站立会议结束后，会议主持人要知道哪些问题需要帮助团队成员解决。

会议主持人需要具备有效组织会议的技能。有效的开展会议，能够体现出组织者、主持人在控制会场，协调问题的解决，积极推动项目进展和管理团队方面的综合实力了。并且，当敏捷项目分布在不同国家地区、时区时，这类会议的开展更有难度。

给会议主持人的几点建议：

1) 自信

2) 声音洪亮

3) 在会议时适当走动

4) 观察团队每个人，了解每个人的特点

5) 不做无准备之发言

6) 保持礼貌，感激，尊重

7) 避免情绪化

8) 平衡发言时间

9) 请外向型的先发言，请内向型做总结

10) 保持正常语速

11) 可定期改变发言顺序

12) 并非3个问题都需要问

如果是在电话会议中则还需注意：

1) 发言者需独占话筒；

2) 未发言者需保持缄默。

4.11.3案例说明

案例1: 来自用友的医疗基层卫生服务系统产品研发的项目

◆ **项目基本信息**

医疗基层卫生服务系统产品研发

- 团队规模：20人
- 环境：JAVA、用友UAP

◆ **组织结构**

矩阵式组织架构，包括需求、UE/UI、开发、测试来自不同部门的人员组成产品团队，参照 Scrum 组织架构，产品项目经理为 Scrum Master

◆ **面临的问题**

产品研发小组内部或小组之间沟通交流不够，导致需求已完成但设计人员不清楚迟迟不开始的情况，或设计的内容与需求不符，但开发出来才发现，导致修改工作量大。

◆ **解决方法**

采用每日站会，加强沟通，团队成员互动，了解相互的情况、问题、计划，及时发现协作之间不一致的问题。

◆ **主要收益**

产品研发流程运转效率大大提升，研发过程中的问题和困难都能及时得到解决。

案例2：来自IBM的分布式项目

◆ **项目基本信息**

在分布式敏捷项目中，团队因区域、文化、时间的差异，每日 Scrum 会议仍能够顺利开展。

- 团队规模：20人
- 环境：Java

◆ **组织结构**

标准 Scrum 组织架构，但成员分布在不同国家。

◆ **面临的问题**

与会人员不同角色、不同性格、不同文化、不同经验、不同关系，一开始每日会议非常焦灼。即使在同一个会议室内每日成功开展会议也未必是件易事。

◆ **解决方法**

1) Scrum Master学习组织会议的方法和技巧；

2) 避免错误的行为和方法。

◆ **主要收益**

1) 提高团队会议效率，节约50%会议时间；

2) 提高团队会议出席率30%；

3) Scrum Master 更加专业和自信。

案例3：来自石化盈科的某企业能耗评价项目

◆ **项目基本信息**

在企业能耗评价系统二期开发项目中使用 Scrum 模型，每次 Sprint 周期相对固定，约为 2 周左右，要求团队快速响应。

- 团队规模：7人

- 环境：VSTS2010 & SQL server

◆ **组织结构**

Scrum 组织结构：1 名 Product Ower,1 名 Scrum Master 及 5 名团队开发/测试人员。

◆ **面临的问题**

团队成员彼此间不熟悉，为第一次合作。工位相对分散，公司会议室资源紧张。

◆ **解决方法**

每日上午召开站立会议，一般选择刚上班的时间，地点随机，相对灵活。各团队成员分别汇报自己的工作情况，轮流发言，内容围绕着 3 个方面（1、我昨天完成了什么？2、我今天准备做什么？3、我当前工作遇到了哪些问题？）。

Scrum Master 负责主持会议，记录并跟踪大家提出的问题。面临来自团队外干扰时由 Scrum Master 出面协调。

◆ **主要收益**

1) 通过每日站立会议增进了团队成员之间的了解；

2) 有效的提高了团队的信息沟通；

3) 有利于及时发现项目问题；

4) 提高了团队开发效率。

4.12 任务板

4.12.1定义和特性说明

任务板（Task board）一箭双雕：为项目团队提供一个便利的工具，用于管理他们的工作；使团队成员对本冲刺剩余的工作一目了然。（Agile Estimating and Planning, Mike Cohn）

任务板通常设立于项目团队日常工作的公共空间的一面墙上，可以是一大块白板、软木板，或者一块干净的墙面。任务板上的信息包括该冲刺计划完成的用户故事和相应的任务，分别写在卡片上，按照一定的方式贴在任务板上（To Do，In Progress，Done）。团队成员通过调整任务卡的位置和上面的信息反映任务的执行情况。任务板其实就是一个能够互动的，墙上的 Sprint Backlog。

4.12.2应用说明

任务板的基本构成

任务板中的信息包括：

1) 用户故事卡（Story Card）

每张卡片记录一条用户故事（Story）。卡片上除用户故事的描述外，还可记录该用户故事对应的点数（Story Point，是由团队成员一起估算出来的）。

2) 任务卡（Task Card）

每张卡片记录一条任务，每个用户故事卡通常对应多个任务卡。任务卡上通常记录任务的内容，以及到目前为止完成该任务所需的工作量（通常以小时为单位）。工作量随任务的进展实时更新，更新的历史直接反应在卡片上。

任务板的布局：

第一列：用户故事，每个用户故事卡占一行

第二列：已计划的任务（To Do），与相应的用户故事同在一行。

第三列：执行中的任务（In Progress），与相应的用户故事同在一行。该列的任务卡上可以标注执行该任务的组员的标记。

第四列：完成的任务（Done），与相应的用户故事同在一行。

任务板的附加信息

除了以上列出的基本信息外，任务板还可包括其他一些附加信息，比如：燃尽图、计划外任务列表等。

任务板的使用

任务板的初始化：任务板中的初始信息（故事卡、所有任务卡，包括估算信息）应该在Sprint Planning 会议之后，第一次站立会议（Daily Scrum）之前准备好。

任务板的更新：团队成员随着任务的进展随时更新任务卡。包括：

1) 当开始一个新的任务时，将该任务从“To Do”列移到“In Progress”列，并且加上自己的标识，如与初始的估算不同，则更新到目前为止完成改任务所需工作量（小时）；

2) 在每次站立会议（Daily Scrum）之前或当中，更新自己所执行的任务的剩余工作量（注意：由于初始估计的偏差，或遇到新的问题，此时的数值有可能比之前的数值

大或者相近）；

3) 当某个任务完成后，将该任务卡移到“Done”一栏中。

任务板的跟踪：通过任务板，团队成员可以直观地看到该冲刺中所有任务的执行情况，是否有某些有些优先级较高的任务一直没人认领、是否某个任务出现了什么问题因为它长期处于“n Progress”状态。团队成员时刻都能直观地看到本冲刺还有哪些任务需要完成。

Story	To Do	In Progress	Done
As a user, I can … 5	Code the … 8 Code the … 6 Test the … 4	Code the … Tom ~~8~~ 4	Code the … Mike ~~6~~ ~~4~~ 0
As a user, I can… 2	Code the … 8 Test the … 6		
As a user, I can… 3	Code the … 6 Test the … 4	Code the … Lucy ~~5~~ 3	

图4-4 任务板 （Agile Estimating and Planning, Mike Cohn）

4.12.3案例说明

案例1：Technicolor Task Board

◆ **项目基本信息**

Research 项目，研究项目”不好做计划”，项目基本信息如下：

- 团队规模：4-5人
- 主要交付物：算法、实验用代码、demo、技术调研报告等。

◆ **组织结构**

标准 SCRUM 组织架构

◆ **面临的问题**

在使用 Task Board 之前，由于 research 工作的相对独立性，站立会议中讨论的进展有时相关性不大，也不直观，使得站立会议的效率不高。Retrospective Meeting 中讨论的结果经常忘记实施。

◆ **解决方法**

Task Board + Sprint Burn-down Chart + Retrospective Result

◆ **主要收益**

Sprint 的任务及进展全部显示在公共区域的墙上，使得团队成员对 Sprint 的进展了解得更加直观，有效提高了站立会议的效率。

Retrospective Result 也贴于明显的位置，提醒大家在当前 Sprint 中需要注意的事项。

案例2：英孚教育　任务看板

◆ **项目基本信息**

整个团队都能随时看到的大白板

◆ **组织结构**

团队所有人员

◆ **面临的问题**

开发的过程中，不光是管理着项目的领导还是团队的成员都希望有一个地方能让他们简单而快速的发现项目的进度，并且能在困难发生前，看出问题，并提早解决。

传统的 mpp 管理方式，不光在更新上费时，同时也并不是每个人都会想到去看，也不是每个开发人员都看的懂。

◆ **解决方法**

项目看板。它被选在一个项目组成员都能方便看到的地方，可能是小办公室门口，也可能是必经的路口，上面贴满了任务表，并且随着进度的更新，变换着状态。无论是谁，都可以轻而易举的结合看板上的三个模块，获得项目当前的状态。

◆ **主要收益**

1) 团队成员间，作为一个记录板，很容易看到别人正在进行的任务；

2) 任何问题，方便的找到责任人；

3) 帮助Scrum Master预估所会遇到的困难与阻碍；

4) 让团队时刻能更新手头的障碍，并有预期的看到障碍被解决的时间，方便合理安排自己的时间；

5) 老板无需和团队沟通，就能知道团队的进度；

6) 让团队意识到自己当前的进度，调整开发速度；

7) 瓶颈的消除，变的积极，因为谁都不会想在action board看到自己的名字。

案例3：用友 银行客户事业部 任务板

◆ **项目基本信息**

在开发 XX 银行资金交易系统的过程中，使用 SCRUM WORKS 里的任务板来模拟任务板，来展现任务、分配任务。团队成员接收到分配的任务后，即时更改任务状态。

- 团队规模：6人
- 环境：NC5.6平台 & ORACLE10G

◆ **组织结构**

标准 SCRUM 组织架构

◆ **面临的问题**

1) 成员容易挑选那些容易实现或者自己感觉兴趣的任务；

2) 任务板状态更新不及时，造成项目进度失真。

◆ **解决方法**

1) 任务分配由原来个人挑选改变为master根据实际情况指定；

2) 任务板的进度状态在每天的站立会议中进行核实。

◆ **主要收益**

通过使用任务板，团队成员能够更加明确自己的开发任务，同时项目管理人员也能够通过任务板的状态变化更加直观的了解项目的进度和开发情况，使项目的可视化大在加强。

4.13 用户故事

4.13.1定义和特性说明

用户故事（User Story）是敏捷开发方法的基础，它描述了产品/项目实现的具体功能。它真正以用户的视角来表述希望产品/项目提供哪些功能或特性，通常由客户、产品经理或者需求人员来编写，开发人员也可以编写一些非功能的用户故事，比如安全、性能、质量等层面。为了辅助用户故事的编写，建议使用标准的格式：

作为<某个角色>，我希望<实现何种功能>，以便<带来何种价值>

如：作为一个用户，我希望在每次退出系统前得到是否保存的提示，以便所有内容都被成功存储。

用户故事使用贴近用户的语言来描述需求，简洁易懂。而每个用户故事的商业价值则为确定用户故事的优先级提供了量化的参考。和用户故事配合使用的是确认测试（Acceptance Testing），编写用户故事的人员同时负责测试点的编写，这些验证点方便后期衡量该故事是否被正确实现。另外，和用户故事相配合使用的概念还有：故事点（Story Point）、用户故事的优先级、团队速率等。

4.13.2应用说明

尽量在系统的开始阶段，尝试找出所有的用户故事。为每一个用户故事评估故事点（Story Point）和优先级。

在版本发布计划会议上，由产品经理和团队共同负责，从产品订单中选出高优先级的、数量合适的（由团队速率表示）用户故事，制定迭代计划。

在迭代计划会议上，将本迭代需要实现的用户故事分解为工作任务（Task)，由团队成员认领并承诺在规定的时间内完成。每一个迭代完成后，召开评审会议，与 Product Owner 进行用户故事的确认（结合确认测试），对于确认通过的用户故事，可以将关联的工作任

务关闭；对于确认未通过的用户故事，考虑是否将本次迭代计划延迟或将这些用户故事放到下一轮迭代计划中。

补充说明：可以把用户故事写在一张张小卡片上，同时在卡片上标明它的优先级和预计完成时间，张贴在墙上（通常称为故事墙），以方便敏捷团队实时查看和沟通。

4.13.3案例说明

案例1：

◆ **项目基本信息**

一个产品的新版本的开发：

- 团队规模：200+人
- 环境：Java, Web2.0, C, DB2

◆ **组织结构**

Scrum of Scrum 组织架构

◆ **面临的问题**

开发周期确定，需要严格按照计划的时间发布新版本。新的技术架构具有不确定性。行业变化快，要求产品能够迅速响应市场需求。

◆ **解决方法**

围绕用户故事进行计划、估计。产品负责人根据市场需求，随时修订产品订单的优先级，确保敏捷团队在每一个迭代周期能够交付相应的用户故事。

◆ **主要收益**

1) 能够及时调整计划，相应市场变化；

2) 迅速交付高优先级的用户故事；

3) 避免部分完成的工作带来的浪费。

案例2：

◆ **项目基本信息**

在开发 XX 银行资金交易系统的过程中，需要有一种有效的沟通手段，既能让需求分析人员和开发人员都能够明白客户的需求和意图，这就要求这种沟通手段即不能太偏重于业务背景又不能太偏重于技术名词，通过用户故事能够很好的解决这一问题。

- 团队规模：6人
- 环境：NC5.6平台 & ORACLE10G

◆ **组织结构**

标准 SCRUM 架构。

◆ **面临的问题**

USER STORY 难以编写，并且会占用开发人员的工作量。

◆ **解决方法**

1) 用户故事的形式要简单，这样才可以很容易地掌握编写它的方法；

2) 应该由与项目相关的领域专家们来写用户故事，而不是开发人员；

3) 用户故事有一个标准的格式：作为<某个角色>，我可以<做什么>，以完成<什么目的>。

◆ **主要收益**

通过使用用户故事来描述用户需求的每一个用例，使业务需求人员能够更好的表达客户需求，开发人员也能够最大程度的理解需求人员的意图，使沟通效率大大加强。

案例3：

◆ **项目基本信息**

在企业能耗评价系统二期开发项目中使用Scrum模型，每次Sprint周期相对固定，约为2周左右，要求团队快速响应。

- 团队规模：7人
- 环境：VSTS2010 & SQL Server

◆ **组织结构**

Scrum组织结构。1名Product Ower,1名Scrum Master及5名团队开发/测试人员

◆ **面临的问题**

1) 团队成员对业务知识掌握不够；

2) Product Owner公务繁多，还负责其他项目，不可能实时支持项目的全过程。

◆ **解决方法**

每次冲刺前先召开计划会议，由Scrum Master主持，会议之前Product Owner确认本次冲刺所涉及的用户故事。计划会议一般选择在周二的上午进行，时间为半天到一天。

计划会议上，Product Owner向项目成员详细解说某一个用户故事，确保团队成员准确把握用户需求；Product Owner通常还会在计划会议上给团队成员培训业务知识，各成员进行学习与交流。

◆ **主要收益**

1) 在冲刺计划时，团队成员与Product Owner就用户故事的理解达成共识，降低了开发过程中由于误解需求造成返工的风险；
2) 团队成员聚集在一起学习业务知识，有助于业务能力的提高，加强彼此之间的交流与沟通。

4.14 TDD (测试驱动开发)

4.14.1定义和特性说明

测试驱动的开发是敏捷开发的核心最佳实践之一，这种方法本身是一种开发方法，强调在开发的过程中，首先开发测试用例，然后再开发代码，而开发代码的目的或者说目标就是通过这些测试用例。

测试驱动的开发方法通过减少集成和稳定构建所必须花费的时间来减少产品的上市时间。这种方法通过在错误引入之际就能发现和修复这些错误来提高生产力，以及通过保证在代码检入之前，所有新的代码都被测试过，和所有老的代码都经过回归测试，从而保证软件的整体质量。

测试驱动开发方法的一个显着优点是，它使您能够采取小步骤编写软件，这不仅是安全也是生产力远远超过大步骤编写代码。举个例子，假设你添加一些新的功能代码，编译和测试。机会是相当不错，您的测试将被新的代码中存在的缺陷所打破。这些错误将很容易被发现和修复，假如你新写的代码时5行而不是50行。言外之意是，你的编译器和回归测试套件的速度越快，那么更吸引人的是通过小的步骤进行开发。

4.14.2应用说明

采用测试驱动的开发，你需要通过编写一个单一的测试来做详细的设计，然后就是编写刚好够用的代码来通过这个测试。当你需要给你的系统添加新的功能时，您需要完成以下的步骤：

1) 迅速添加一个开发测试

你需要刚好够用的代码导致失败。举个例子，可以创建一个新的方法，即将被添加到一个类，只是抛出一个致命异常。

2) 运行测试

您通常会运行完整的测试套件，虽然速度的原因，您可能会决定只运行一个子集。我们的目标是确保新的测试其实是失败的。

3) 修改代码

我们的目标是只添加刚好够用的功能，使代码通过新的考验。

4) 再次运行你的测试套件

如果测试失败，你需要更新你的功能代码和重新测试。一旦通过测试，重新开始。

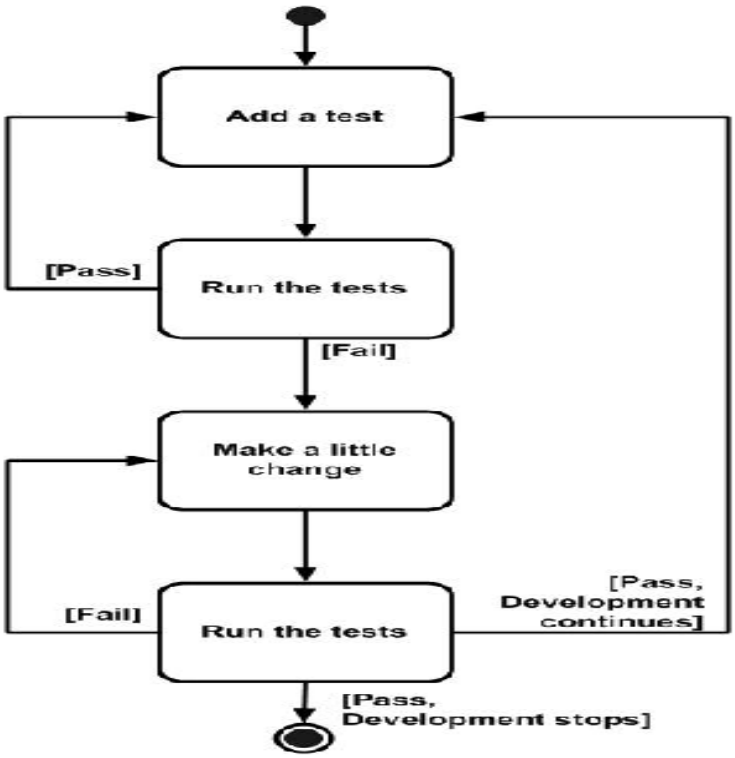


图4-5 测试套件

4.14.3案例说明

案例一：石化盈科某软件开发项目

◆ **项目基本信息**

- 团队规模：7人
- 环境：VSTS2010 & sql server

◆ **组织结构**

Scrum组织结构，1名Product Ower，1名Scrum Master及4名团队开发人员。另配置一名软件功能测试人员，来自测试部门。

◆ **面临的问题**

1) 开发人员中有一名为工作年限不长的刚毕业学生,对测试代码的设计能力把握不足；

2) 经常需要随时提供可运行程序；

3) 功能测试人员通常在系统集成之后才进入项目工作，把握需求的准确度不高。

◆ **解决方法**

1) 项目团队内部自发组织单元测试的学习与培训；

2) 在编写完单元测试代码后，编写程序代码，由单元测试代码驱动程序代码的编写；

3) 将单元测试发现的代码编写问题更新到代码规范中，在项目团队中共享；

4) 功能测试人员参与计划会议，充分理解每一项用户故事，对表述不当的地方尽早提出改进建议，在开发人员编写单元测试用例的同时根据用户故事开始编写功能测试用例；

5) 通过每日构建测试，保证服务器代码的稳定行和高质量。

◆ **主要收益**

1) 测试驱动开发，尽早发现程序中存在的缺陷，提升了研发效率；

2) 大大提高了代码质量；

3) 提高了测试效率；

4) 团队成员在学习过程中得到了能力提升，培养了良好的编码习惯。

案例二：EA公司B2B 电子商务平台开发项目

◆ **项目基本信息**

- B2B 电子商务平台开发
- 环境：Java，PHP + Drupal

◆ **组织结构**

标准Scrum组织架构，团队规模：7-15人不等。

◆ **面临的问题**

传统需求文档驱动开发无法良好的驱动Scrum模型。

◆ **解决方法**

使用ATDD：编写接收测试用例取代需求文档；不同层次的测试用例驱动TDD开发活动，接收测试自动化。

◆ **主要收益**

精炼、有序的需求文档，敏捷的相应需求变更，需求与测试驱动开发无缝衔接。