

Agile 方法研究综述

钱乐秋^② 张敬周^{①②} 朱三元^{①②}

^① (上海计算机软件技术开发中心)

^② (复旦大学计算机与信息技术系)

(zjz@ssc.stn.sh.cn)

摘 要 本文综述了近来新出现的 Agile 方法, 对其应用范围、价值系统、核心实践给出了阐述, 并介绍了 XP、SCRUM、Crystal 等主要的 Agile 方法及其发展趋势, 在此基础上, 给出了 CMM 与 Agile 方法之间的比较分析, 最后提出了 Agile 方法待研究的问题。

关键词 Agile 方法, 轻载方法, CMM, XP

AN OVERVIEW OF AGILE METHODOLOGY

QIAN Le-Qiu^② ZHANG Jing-Zhou^{①②} ZHU San-Yuan^{①②}

^①(Shanghai Development Center of Computer Software Technology)

^②(Department of Computer and Information Technology, Fudan University)

Abstract Agile Methodology attempts a useful compromise between no process and too much process, i.e., “Just enough”. Given in this paper is an overview of agile methodology and its trends, including XP, SCRUM, Crystal, FDD, ASD, etc.. Also given are application scope, value system, core principles, and key practices of agile methodology. Based on the above survey and analysis, a comparison between CMM and agile methodology is presented, and several issues need further research are pointed out.

Key words Agile Methodology, Lightweight Methodology, CMM, XP

1. 引言

质量和生产率是软件工程的两个核心目标, 这两个目标相互作用相互影响, 在具体的软件开发实施中要作具体的权衡。随着技术的迅速发展和经济的全球化, 软件开发出现了新的特点, 即在需求和技术不断变化的情况下实现快节奏的软件开发, 这就对生产率提出了很高的要求。ISO-9000、CMM、SPICE 目前已被公认为软件质量保障方面的事实标准, 但由于其强调管理和控制, 追求项目的可预测性和过程状态的可视性, 在提高生产率方面并未予以足够的重视, 实施时一方面需要大量中间制品(过程文档)的制作, 给开发人员带来很大负担, 另一方面, 追求可预测性与实际需求的模糊和快速变化不相协调。在此情况下, 出现了一些新的开发方法。

新的方法主要有 Extreme Programming (简称 XP)、SCRUM、Crystal Methodologies、Feature Driven Development(简称 FDD)、Dynamic Systems Development Methodology(简称 DSDM)、Adaptive Software Development(简称 ASD)、Pragmatic Programming 等, 统称轻载(Lightweight)方法, 以区别于传统的开发方法(称重载方法, Heavyweight)。2001 年 2

月,新方法的一些创始人在美国犹他州成立了Agile 联盟,将轻载方法正式更名为 Agile 方法, Agile 有轻巧、机敏、活力的意思^[1]。

Agile 方法目前还没有一个明确的定义,其特点是对软件生产率的高度重视,主要适用于需求模糊或快速变化下的、小型项目组的开发。有人称, Agile 方法是在保证软件开发有成功产出的前提下,尽量减少开发过程中的活动和制品的方法,笼统的讲就是,“刚刚好”(Just enough),即开发中的活动及制品既不要太多也不要太少,在满足所需的软件质量要求的前提下,力求提高开发效率。

这些新方法提出后,引起了软件界的广泛重视, Tom Demarco 认为(Cutter Trends Report), “XP 对当今时代的作用可与 CMM 在八十年代和九十年代初的作用相媲美”。新方法在实践中也得到了巨大的成功,如 IONA 公司的 Obix 技术支持小组在采用了 XP 方法后,软件生产率提高了 67%^[7];软件生产率组织(SPG)的 Capers Jones 则称, SCRUM 方法可提高生产率 6 倍^[13]。

2. Agile 方法的核心理念及特点

任何软件开发方法都有一个相应的价值系统(Value system),方法通过价值系统对过程加以指导,方法只有在其应用周境(context)与价值系统相吻合时才能发挥真正效力,价值系统的基础是对世界的信仰和对软件开发特点的认识,可以说是核心理念。

Agile 方法的代表人之一 Martin Fowler 提出了 Agile 方法的核心理念^[2]:适应和以人为本^[2]。

2.1 Agile 方法基于适应而非预测。

人们不可能在软件开发之前准确地把握当时的需求及其之后的走势,这一点已由 Peter Wegner 用数学的方法给出了严格的证明^[27,28]。设想有一个虚拟的“理想软件”,它是对用户的最大满足,并能随时间的推移和形势变化而与现实保持即时同步,那么软件需求实质上是对“理想软件”的粗略的外部描述。理论上来说,软件开发应是一个跟踪过程,只能通过软件开发过程的自适应性来逼近“理想软件”,不可能完全实现。

根据自动控制理论,自适应系统是一个强反馈系统。在软件开发中,需求的获取和分析、软件设计、编码等实质上均为前馈环节,真正的反馈环节应该是用户对可运行软件的使用、使用中与“理想软件”的比较判断及判断后与开发人员的信息交流。另外,控制理论还要求,反馈和前馈这一回路的响应速度应大于被跟踪(或被适应)的系统的变化速度,这就要求软件开发有快速的产出能力。

基于适应性特点, Agile 方法通过快速、短迭代式的开发,不断产出和演化可运行软件,加强项目有关人员的信息交流等手段来提高反馈的速度、力度和准确性。在软件过程改进方面, Agile 方法通过对实际反馈信息的度量,根据用户满意度(时间和质量的权衡)进行微调,相当于缺陷驱动改进方式。

2.2 Agile 方法是以人为导向而非过程导向。

软件开发中,人的因素是第一位的。人是过程的主体,而人的工作承受力是有限的,任何超出人正常承受力的过程必然导致过程结果的偏离,这一点在注重过程改进的方法中往往被忽视或低估。 Agile 方法认为,软件开发中的绝大部分是需要创造力的设计工作,软件人员是创造性的工作者,有主观上做好工作的意愿,不同于大工业时代生产流水线上被动性的工人,因此,在管理理念上应注重领导和协作,而非命令和控制,充分发挥软件人员的能动性和创造力。软件开发应首先着眼于有用的可执行的软件,也就是首先考虑商务目标,而不

是为过程而过程。

Agile 方法特别强调软件开发中相关人员间信息交流，Alistir Cockburn 在对数十个项目案例调查分析后提出，“项目失败的原因最终都可追溯到信息没有及时准确地传递到应该接收它的人”，而人特别擅长面对面的交流，cockburn 认为面对面交流的成本要远远低于文档交流的成本，因此，Agile 方法的一个共同特点就是，努力营造诚信、开放的组织氛围，根据项目中信息流通的具体情况，按高内聚、松耦合的原则将项目组划分为若干小组（每个小组以不超过 10 人为宜，组员均在一个工作间内工作），通过小组内各种渠道的沟通，来减少中间制品的工作负担，提高应变能力。

2.3 Agile 方法中的价值系统和指导原则

Agile 联盟提出了“四个价值”、“十二个指导原则”^[1]。

Agile 方法的四个价值：

- (1) 较之于过程和工具，更注重人及其相互作用的价值。
- (2) 较之于无所不及的各类文档，更注重可运行的软件的价值。
- (3) 较之于合同谈判，更注重与客户合作的价值。
- (4) 较之于按计划行事，更注重响应需求变化的价值。

Agile 方法的指导原则：

- (1) 在快速不断地交付用户可运行软件的过程中，将使用户满意放在第一位。
- (2) 以积极的态度对待需求的变化（不管该变化出现在开发早期还是后期）。Agile 过程紧密围绕变化展开并利用变化来实现客户的竞争优势。
- (3) 以几周几个月为周期，尽快、不断地交付可运行的软件供用户使用。
- (4) 在项目过程中，业务人员和开发人员最好能一起工作。
- (5) 以积极向上的员工为中心建立项目组，给予他们所需的环境和支持，对他们的工作予以充分的信任。
- (6) 在项目中，最有用、最有效的信息沟通手段是面对面的交谈。
- (7) 项目进度度的首要依据是可运行的软件。
- (8) Agile 过程高度重视可持续开发。项目发起者、开发者和用户应能始终保持步调一致。
- (9) 应时刻关注技术上的精益求精和设计的合理，这样能提高软件的快速应变力。
- (10) 简单化（尽可能减少不必要工作的艺术）是基本原则。
- (11) 最好的框架结构、需求和设计产生于自组织的项目组。
- (12) 项目组要定期对其运作方面进行反思，提出改进意见，并相应进行细调。

此外，Agile 方法实施中一般采用面向对象技术（接口定义良好的其它开发技术也可），另外还强调在开发中要有足够的工具（如配置管理工具、建模工具等）支持。

2.4 Agile 方法的适用范围^[2]

Martin Fowler 认为，新方法不是到处可适用的。在以下情况下，适合采用 Agile 方法：

- l 需求不确定和易挥发（Volatile,意指今天的要求明天就不需要了）的场合。
- l 有责任感和积极向上的开发人员。
- l 用户容易沟通并能参与。

在以下情况下，不适合 Agile 方法：

- l 超过 50 人的项目组
- l 开发范围被限定（如由合同限定了价格）

3. Agile 具体方法介绍

Agile 方法是一组开发方法的统称，下面介绍几个典型的 Agile 开发方法。

3.1 XP (Extreme Programming) ^[3~9]

由 Kent Beck 提出，是 Agile 方法中最引人注目的一个，名称中 Extreme 的含义是将好的开发实践 (Practices) 运用到极致。XP 最初实践于 1997 年 Crysler 公司的 C3 项目 (人员薪金管理)，采用 Smalltalk 语言开发。适用于 10 人以下项目组、开发地点集中的场合，现已成为小组开发方法的一个典型，被业界广泛用于需求模糊和挥发性强的场合。XP 基于四个价值(values)提出了十二个核心实践 (Practices)。

3.1.1 XP 方法的四个价值:

- I **交流 (Communication)**。项目相关人员之间充分、多渠道 (最好面对面) 的沟通。
- I **简化 (Simplicity)**。在系统可运转的前提下，做最简洁的工作；在开发中不断优化设计，时刻保持代码简洁、无冗余。
- I **反馈 (Feedback)**。强调各种形式的反馈，如小交付、短迭代、先考虑测试再编码等。
- I **胆识 (Courage)**。面对压力做正确的判断并敢于付诸行动，如，敢于丢弃设计不良的代码，疲惫时立即休息等。

3.1.2 XP 方法的十二个核心实践:

- I **工作团队 (Whole Team)**。所有的小组成员应在同一个工作地点工作。成员中必须有一个用户代表 (On-site User)，由他/她来提出需求，确定开发优先级，把握开发的动向。通常还设一个“教练” (Coach) 角色，来指导 XP 方法的实施及与外部的沟通协调等。小组每个成员都应围绕用户代表，充分贡献自己的技能。
- I **计划 (Planning Game)**。包括两类：交付计划和迭代 (Iteration) 计划，前者是在项目开始时对软件交付日期的粗略估计，后者是在每个迭代开始时对本次迭代的工作安排。二者都可在执行时调整，只不过迭代计划要更为具体和准确。
- I **系统比拟 (Metaphor)**。系统比拟是待开发软件系统的一个形象化比喻，这个比喻必须是每个组员都熟悉的，它相当于一个粗略的软件体系结构 (Architecture)，使用户和开发人员建立一个一致的框架，从而指导开发的进行。
- I **小交付 (Small release)**。经常交付可运行的、具有商业价值的小软件，供用户代表评估或最终使用。这里的“小”意味着软件模块能尽快 (可在一个迭代内完成)、不断地交付。
- I **测试 (testing)**。XP 要求先开发测试用例 (可自动执行) 然后进行编码 (testing then coding)，而且测试要不断进行，保证代码测试要 100% 通过。分两种测试：单元测试和验收测试 (Acceptance test)。单元测试由开发人员负责，验收测试由用户代表负责。
- I **简洁设计 (Simple Design)**。有两个含义：一是设计只考虑当前定义的功能而不考虑以后需求的变化，二是指该设计是完成目前功能所需的最简洁的设计，要简单易懂、没有冗余。
- I **结对开发 (Pair Programming)**。开发人员两人结一对，在一台机器前进行开发，一人打字编程另一人查看其编码，配对的两人不固定的，建议每天轮换。

- I **设计改进 (Design Improvement)**。在整个开发过程中，应对程序结构进行持续不断的梳理 (Refactoring，在不影响程序的外部可见行为的情况下，按高内聚松耦合的原则对程序内部结构进行改进)，保持代码简洁、无冗余。
- I **持续集成 (Continuous Integration)**。保持项目组中所有开发的模块始终是组装完毕、集成良好且可执行的，一旦新的模块通过了单元测试，应立即将其组装、集成，形成新的可执行系统。
- I **代码共享 (Collective code Ownership)**。任何结对开发者在任何时候都可改进项目组中的任何代码。
- I **编码标准 (Coding Standard)**。对编程风格 (命名、注释、格式等) 要有一个大家认可的、共同遵守的标准，使整个程序如同一人编写。
- I **可持续步调 (Sustainable Pace)**。整个项目的开发节奏应该是可长期维持的，XP 提出每周 40 小时工作制，即不要加班，以使开发人员能长期保持工作的高效率。

3.1.3 XP 开发过程:

I 整体开发过程

如图 1 所示，用户代表提出**用户故事** (User stories，类似于用例，但更简单，不超过三句话即可，一般只涉及功能要求)，项目组据此进行讨论提出**系统比拟**，在此活动中有可能要进行体系结构的刺探 (**Spike**，意在试探解决有关技术难点，走通技术路线)。在**系统比拟**和**用户故事**的基础上，根据用户设定的优先级制订交付计划 (制订计划中可能需要对某个技术难点进行刺探)，然后开始多个迭代过程 (每个迭代一般为 1~3 周)，在迭代期内产生的新**用户故事**不在本迭代内解决，以保证开发不受干扰。经验收测试通过后交付使用。

I 迭代过程

如图 2 所示。根据交付计划和**项目速率** (Project velocity，实际开发时间和估计时间的比率，如估计 1 天完成的工作实际为 2 天，项目速率为 2)，选择要优先完成的用户故事或待消除的 Bugs，将其分解为可在 1~2 天内完成的任务，制定本迭代计划，然后通过每天的**站立会议** (Stand up meeting，参加人员站着开会以缩短时间提高效率) 解决碰到的问题、调整迭代计划，会后是代码共享式的开发工作，开发人员要确保新功能 100% 通过单元测试，并立即组装形成新的可运行版本，由用户代表进行验收测试。

I 代码共享的编程

如图 3 所示。

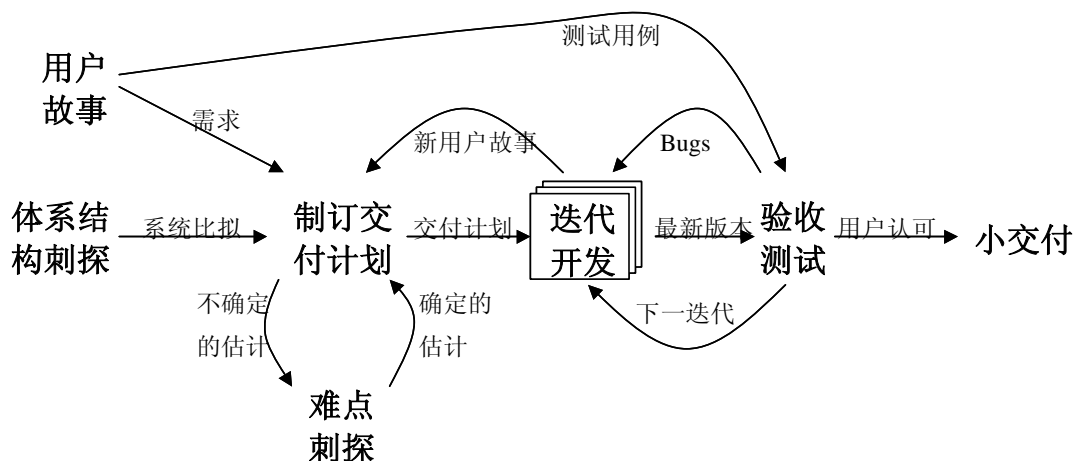


图 1 XP 项目开发过程

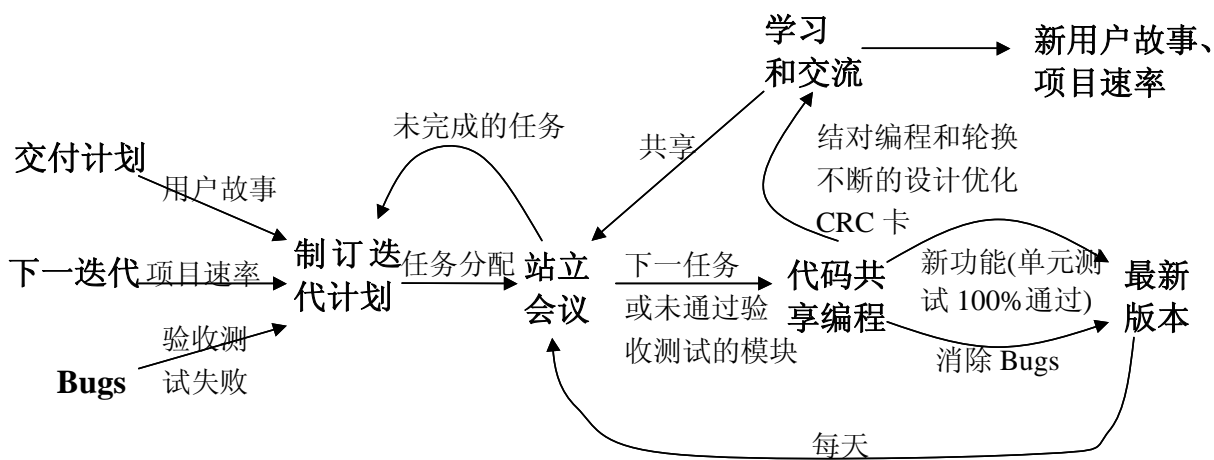


图 2 XP 迭代开发过程

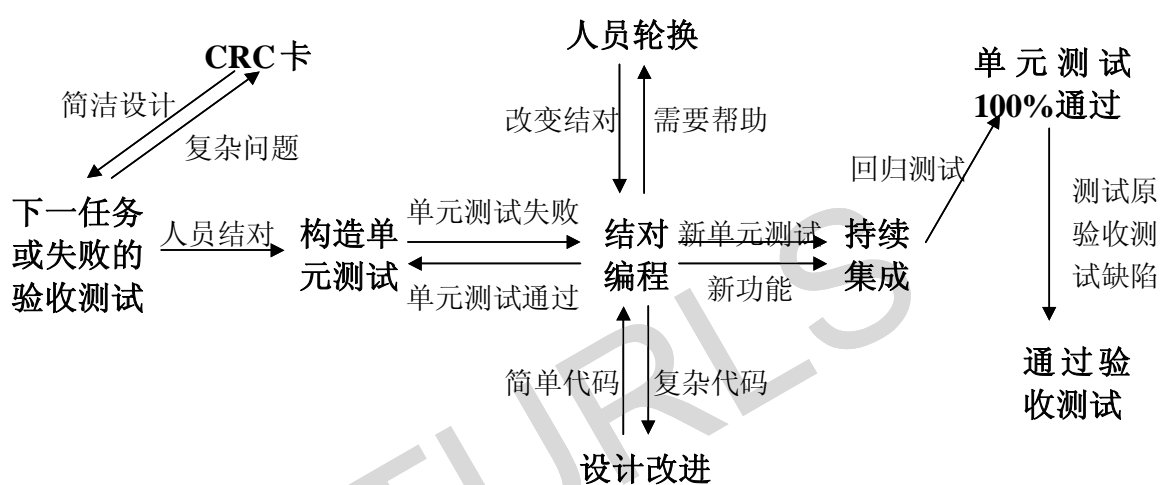


图 3 XP 的代码共享编程

I XP 的计划/反馈循环（如图 4 所示）

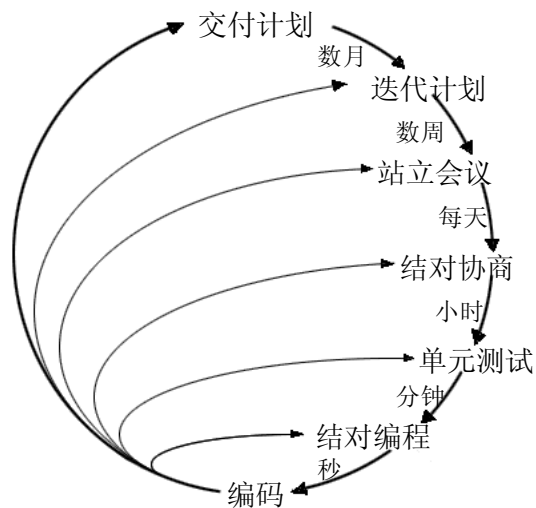


图 4 XP 的计划/反馈循环

3.1.4 XP 的实践和发展

XP 方法引起了世界各地广泛的注意，越来越多的项目组采用或正考虑采用 XP 方法。2001 年 7 月，在美国北卡罗来纳州举行了 XP 国际研讨会，会议上 SEI、IBM、SAS 等知名机构的专家作了专题演讲，表明了 XP 的活跃程度。

对 XP 的批评主要有两个：一是 XP 中的文档问题（几乎没有文档），对此 Ron Jeffries 的解释是，XP 中的系统比拟、用户故事、和测试用例都是传统方法中的文档，另外，XP 紧紧围绕用户代表展开，他可决定文档制作的程度；二是 XP 只强调共享与合作，另一个推动社会进步的因素——竞争并没有体现，为此许多 XP 的实践者对该方法作了一定程度的变通。

3.2 SCRUM 方法^[10~16]

由 Ken Schwaber 和 Jeff Sutherland 提出，旨在寻求充分发挥面向对象和构件技术的开发方法，是对迭代式面向对象方法的改进，名称来自英式橄榄球（在比赛中每个队员都应时刻保持对场上全局的判断，然后通过集体行动，奋力实现同一目标——胜利）。SCRUM 方法最初实践于 Easel 公司（1993 年），现已被数十家公司数百个项目开发中应用，适用于需求难以预测的复杂商务应用产品的开发^[11]。SCRUM 提出的 SCRUM Meeting、Sprint、Backlog、SCRUM Master、SCRUM Team、Demo 等模式已被 PLOP 作为组织和过程模式（Organizational and Process Pattern）的标准^[12]。

SCRUM 将工业过程控制中的概念应用到软件开发中来，认为软件开发过程更多是**经验性过程**(Empirical Process)，而不是**确定性过程**(Defined Process)。确定性过程是可明确描述的、可预测的过程，因而可重复（Repeatable）执行并能产生预期的结果，并能通过科学理论对其最优化。经验性过程与之相反，应作为一个黑箱（Black box）来处理，通过对黑箱的输入输出不断进行度量，在此基础上，结合经验判断对黑箱进行调控，使其不越出设定的边界，从而产生满意的输出。SCRUM 方法将传统开发中的分析、设计、实施视为一个黑箱，认为应加强黑箱内部的混沌性，使项目组工作在混沌的边沿，充分发挥人的创造力。如将经验性过程按确定性过程来处理（如瀑布模型），必将使过程缺乏适应力。

3.2.1 SCRUM 方法的开发过程

包括三个过程：

(1) 计划和体系结构设计（确定性过程）

将 **Backlog**（急待完成的一系列任务，包括：未细化的产品功能要求、Bugs、缺陷、用户提出的改进、具竞争力的功能及技术升级等）按优先级排序形成 **Backlog** 列表，根据该表和风险评估制订产品交付基线。

建立系统体系结构（如为已有系统改进，则只作有限分析、调整），将 **Backlog** 项按高内聚低耦合的原则分解为一系列问题包(Packets,每个 Packet 是一组对象或构件的集合)，依据同样原则相应划分若干个开发小组（**SCRUM 小组**），分配各小组合适的 **Backlog** 项或问题包。建立开发运行环境。

(2) Sprint（经验性过程）

该过程由若干个迭代的冲刺(Sprint) 活动组成，直至风险评估认为产品可交付为止。一个 **Sprint** 是在限定时间段内（**Sprint** 周期，通常为 1~6 周，可在前一个 **Sprint** 结束时调整）的一系列开发活动（包括分析、设计、编码、测试等），每个 SCRUM 小组并行开发且必须步调一致（在一个 **Sprint** 结束后，均须完成所分配的 **Backlog** 项并有可执行的产出）。

每个 **Sprint** 包含以下活动：

- 1 开发。对分配的 **Backlog** 工作进行分析，将所需改动(changes)映射到各 packets，

打开 packets,进行领域分析,然后设计、开发、实施、测试、文档化这些改动。

- l **打包(Wrap)**。封装 packets,产生一个满足 **Backlog** 需求的可执行版本。
- l **评审(Review)**。所有的 SCRUM 小组一起开会,提交各自的工作并演示(**Demo**),然后提出和解决问题(**Issue**)及难点(**problem**),增加新的 **Backlog** 项;发布、审查或调整产品的标准规范;进行风险评估并提出合适的对策;确定下一个 **Sprint** 的工作内容和结束时间。
- l **调整(Adjust)**。根据评审会汇集的信息,对受影响的 Packets 进行适当调整和巩固。

(3) 交付和巩固(确定性过程)

一旦根据风险评估结果认为可交付产品时,即进入该阶段。该阶段的活动包括:组装,系统测试和回归测试(**Regression**),准备培训材料,完成最终文档。

SCRUM 过程认为一个产品的开发将一直持续下去,除非经风险评估后认为应停止。产品交付后的巩固活动类似于传统方法中的维护和改善,目的在于整理 **Sprint** 期压力下忽略的工作,为下一阶段的开发做准备,以便轻装上阵。

3.2.2 SCRUM 对过程的管理:

(1) SCRUM 的控制手段。

SCRUM 提出了八个控制项(**Controls**)用于开发过程的调控,其中风险控制是首要的手段。

- l **Backlog**。
- l 对象/构件。
- l Packets。
- l 变动(**Changes**)。实施一个 **Backlog** 项时,对相应 Packet 的改动。
- l 难点(**Problems**)。实施一个变动时所必须解决的技术难点。
- l 问题(**Issues**)。涉及到整个项目或在 **Backlog** 项分解到 Packet 之前须解决的问题。
- l 措施(**Solutions**)。对问题或难点的解决,通常会导致变动。
- l 风险(**Risks**)。影响项目成功的风险,应持续跟踪评估并相应做出调整。风险评估的结果将影响其他所有控制项。SCRUM 定义了六个概念性变量来用于风险评估:用户需求,时间压力,竞争,质量,远见(**vision**)和可用资源。

在 SCRUM 的各个阶段都使用这些控制项来评估和权衡,管理人员侧重于以此管理 **Backlog**,开发组用以处理变动和难点。所有人员一起来管理问题、风险和措施。

根据对控制项特别是风险的不断度量评估和权衡,一方面,计划和进度(在每个 **Sprint** 结束时)不断相应调整,保证实现产品的商务目标;另一方面,对开发中的工作任务 **Backlog** 动态地进行优先级排序,开发组总是先开发优先级最高的 **Backlog** 项,这样就保证了资源的最合理使用。另外,SCRUM 强调度量(采用标准功能点度量方法)的重要性,通过对每个 **Sprint** 中生产率等的度量,计划和进度将越来越趋于准确。

(2) 项目组织。

项目组由全职开发人员及与该交付产品有关的市场人员、销售人员、用户等组成。设以下小组:

- l **项目管理组**。由产品经理领衔,包括总设计师,各 SCRUM 小组组长,市场、销售的高级职员及典型用户等。
- l 若干个 **SCRUM 小组**。各小组由组长(**SCRUM Master**)领衔。每个小组都是跨专业的(通常包括开发人员,文档人员,质量控制人员或用户代表等),通常为 3~7 人,以使小组内有充分的交流。小组的划分最好是功能导向的(按

所分配的问题包或 **Backlog**), 也可是系统层次导向 (按体系结构中的分层)。在项目组人数增大时, 可在管理组之上再设管理组 (**SCRUM of SCRUM**), 从而使 **SCRUM** 方法的应用到大项目中。

(3) **Sprint** 期间的调控。

在 **Sprint** 期间, 应使各 **SCRUM** 小组尽量避免外界的干扰 (不可将新的 **Backlog** 任务加进来, 组内产生的 **Backlog** 可放到整个项目的 **Backlog** 列表中, 也可在本次 **Sprint** 中解决), 使小组成员专心于目前的工作, 使他们工作在混沌的边沿。

为避免项目组在 **Sprint** 期间不陷入混乱, **SCRUM** 采取两个措施:

I **SCRUM** 会议 (**SCRUM Meeting**)。对小组行为进行监控和刺激。会议在 **Sprint** 期间每天在同一地点举行, 由 **SCRUM Master** 主持。会议上, **SCRUM Master** 对每个小组成员提三个问题:

- 1) 昨天的工作进展如何。
- 2) 有否遇到困难和障碍。
- 3) 今天的工作打算。

会后 **SCRUM Master** 集中精力排除障碍, 小组成员则进行当天的开发。

I **Sprint** 评审会议。评审后根据对每个人的工作成绩, 进行相应的激励。

3.2.3 **SCRUM** 方法的实践效果和发展方向:

SCRUM 在实践中大大提高了生产率 (据软件生产率组织的 **Capers Jones** 称可提高 6 倍), 在实施中有一个 “间断平衡” (**Punctuated equilibrium**) 现象 (类似于自然界中物种的进化, 在经过一段相对平衡的各自独立、并行的发展期后, 在交汇处发生变异), 即在经过紧张、并行的 **Sprint** 开发后, 在 **Sprint** 评审时, 软件产品产生较剧烈的变化。**SCRUM** 方法的最近动向是设法借鉴 **XP** 方法。

3.3 **Crystal** 方法系列^[17~21]

由 **Alistir Cockburn** 提出。与其它 **Agile** 方法的提出者不同, **Cockburn** 的研究基于对 **IBM** 公司近四十个项目 (这些项目有大有小, 方法有轻载和重载, 时间跨度为 1980~1999 年) 案例的调查。他认为不同的项目需采用不同的开发方法, 并随着开发的进行应不断细调 (**On-the-fly tuning**), 也就是连续不断的过程改进。据此他提出了一系列方法 (**Crystal Clear**、**Crystal yellow**、**Crystal Orange**、**Crystal Red** 等, 名称借自于自然界中的晶体, 有透明的、黄色的、橙色的等) 作为基本参照和一些原则来指导方法的调整。

Crystal 方法系列的核心理念:

软件开发可视为创造和交流相协调的过程, 其中人的因素是第一位的, 软件开发的第一目标是交付有用的、可运行的软件, 其次是保持开发工作的可延续性。

Crystal 方法系列强调的价值:

- I 以人和沟通为中心。工具、中间制品、过程均是服务于人的。
- I 高度宽容 (**High tolerance**)。应该认识到人与人之间文化的差异。

Crystal 方法系列的原则 (**principles**):

- I 通过经常产出可运行的代码、充分利用人与人之间的多种交流渠道, 项目组可减少中间制品 (如开发文档, 进度计划等) 的工作量。
- I 由于每个项目有不同的情况并且其状态在不断变化, 项目组内普遍接受的惯例和

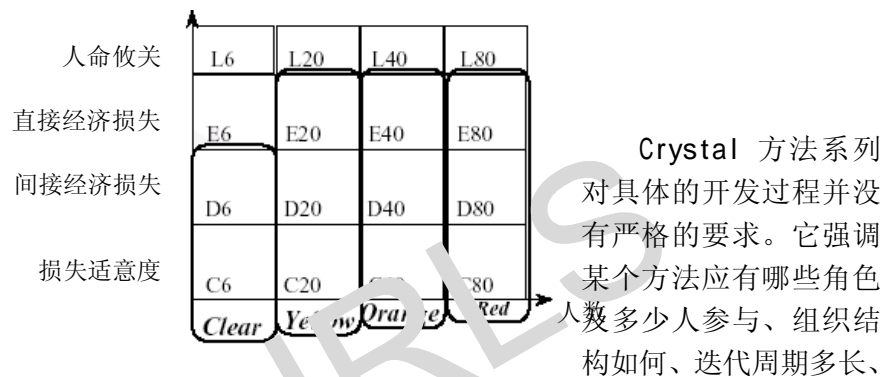
约定应随之塑造和演化。

- l 项目组人数越多，采用的方法应越重（需要更多的文档支持）。
- l 对软件可靠性要求越高，则应采用越严格的过程纪律。
- l 解决工作中的瓶颈则需工作的重叠，从而使非瓶颈性工作效率降低。

Crystal 方法系列的共同规则(rules):

- l 采用渐增迭代式开发，每个迭代不超过 4 个月。
- l 在每个迭代的开发和结束时，必须进行评审、反思，并相应调整。

Crystal 方法的显著特点是对人的高度重视，认为人是一个非线形的、能动的、行为不连贯的（极易受外部影响而中断当前行为）、擅长交流（特别是面对面交流）的构件（Component）^[21]，据此 Crystal 强调对方法的选择和调整要考虑两个因素，一是充分发挥考虑人的特长，二是满足待开发软件的可靠性要求。Crystal 方法的选择如下图所示，坐标横轴是项目组人数，纵轴为软件可靠性要求（软件缺陷可能导致的后果）。Crystal Clear 适用于项目组在 6 人以下，且该项目的疏漏不会造成直接经济损失的情况，依次类推 Crystal yellow 用于 20 人以下不造成人命损失的情况。



如 Crystal Clear, 用在 6 人左右的项目组（无下层小组），迭代周期应为 2-3 个月，角色应有项目资助者、高级设计开发员、设计开发员、用户，开发制品有：交付序列、进度安排、有注解的用例、设计纲要、通用对象模型、可执行代码、移植代码、测试用例、用户手册。

3.4 DSDM 方法^[26]

起源于英国，是一个工业联盟。由 16 家公司发起（1994 年）并组成，最初的目的是开发一种更好的面向领域的快速应用开发（RAD）方法，现在会员已超出英国（美国、印度、德国、法国等），应用范围也不再限于 IT 行业。由于有良好的组织基础，它在技术支持、培训认证、应用推广、研究改进等方面都远比其他 Agile 方法要完善的多。DSDM 适用于时间要求紧的项目。

DSDM 的基本观点是，任何事情都不可能一次性的圆满完成，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，力争需求的最大化满足（传统方法一般是需求固定，时间和资源可变）。具体做法是采用时间框（TimeBox，指项目计划进度中一个固定的时间段，有固定的完成时间，并被赋予一按优先级排序的用户需求集合，在结束时要求有可见的产出。时间框可以嵌套，即一个时间框可由一系列子时间框组成）技术和 MoSCoW（Must do, Should do, Could do, Won't do）优先级排序方法，先提出项目的整体时间框，然后根据需求的细化提出一系列子时间框，依次类推直至子时间框被分解为 2-6 周的时间长度，在每个时间框内按 MoSCoW 优先级原则进行开发，这样就保证了在限定时间内完成最有用的功能需求。另外，DSDM 通过工作间

(Workshop, 即将跨部门跨专业的人员组成小组, 使他们在—个房间内工作, 并合理安排布局以确保充分沟通) 方法加强信息沟通, 提高开发速度。

DSDM 方法的基本原则:

- I 积极主动的用户参与。
- I 赋予 DSDM 项目组充分的决策权。
- I 强调经常性的产品提交 (注重结果而非过程)。
- I 以适合商业目标作为工作确认的首要衡量标准。
- I 迭代和增量式的开发方法 (不同于传统的瀑布型模型)。
- I 开发中所有变化是可追溯的 (reversible)。
- I 基于软件需求提出的开发计划作为宏观进度控制的基线。
- I 测试活动贯彻于软件开发周期的各个阶段。
- I 强调所有项目相关人员的合作。

3.4.1 DSDM 的开发过程

包括 5 个阶段, 其中前两个阶段在时间上有顺序关系, 是后续工作的基础和前提。后三个阶段在实施中视情况可交叉重叠或并行。

(1) 可行性研究 (Feasibility Study)

确定 DSDM 方法是否适合于该项目的开发, 检查项目实施的技术和管理条件是否满足。该过程一般在几周内完成。

(2) 商务研究 (Business Study)

研究整个项目活动的范围, 为进一步的开发提供业务和技术基础。根据功能和非功能需求设定开发进度基线, 提出系统体系结构和可维护性目标。一般在一个月内完成。

(3) 功能模型迭代

该迭代主要侧重于系统商业方面的细化, 构造高级 (high-level) 的功能和技术方面的需求。与设计与构造迭代类似, 主要有四个活动:

- I 标识要开发的功能需求。
- I 讨论决定如何做和何时完成。
- I 产品开发。
- I 评审。

采用演化的原型法辅助需求分析, 目标是通过本迭代和设计构造迭代, 形成经过测试的系统。所有 DSDM 中的原型都是被用来继续演化的, 因此应充分考虑其健壮性。

(4) 设计与构造迭代

该迭代主要侧重于用户可用的、高标准的、经过测试的系统。通过对功能模型迭代中的原型进行增量开发, 形成可用的系统。

(5) 实施

将设计与构造迭代中形成的软件系统从开发环境转移到运行环境, 进行必要的用户的培训和系统移交, 整理项目评审文档, 检查所有的需求情况, 总结项目的产出, 对项目的整体完成程度进行评审。由于采用渐增式开发, 评审可能有几种结果:

- I 所有需求都完成并交付, 无需后续开发。
- I 开发中发现了新的功能区域。开发转至商务研究阶段。
- I 由于时间限制, 一些非重要功能在开发中忽略。将其提交功能模型迭代阶段。
- I 非功能性需求未能满足。开发转至设计与构造阶段去调整。

3.5 FDD (Feature Driven Development) ^[22, 23]

由 Jeff de Luca 和 Peter Coad 提出, 是一个模型驱动、短迭代的开发方法, 适用于变化周期短的业务应用开发。所谓的特征点 (Feature) 是一些用户眼中有用的小功能项, 一个特征点能在两周或更短的时间内被实施, 且产生可见的、能运行的代码。

FDD 提出了三个概念用于系统建模: 主特征点集、特征点集、特征点。

I 特征点的表示模板:

<action> the <result><by|for|of|to>a(n)<object>(对某个对象执行某个动作得到某个结果)。例如, Calculate the total of a sale(计算某个销售总额)。

I 特征点集由一系列业务相关的特征点组成, 表示模板为:

<action><-ing>a(n)<object>(某个对象某个行为)。如: Making a product sale to a customer(销售产品给某客户)

I 主特征点集, 由一系列业务相关的特征点集组成, 表示模板为:

<object> management(某对象的管理)

很明显, FDD 的建模方法充分利用了英语自然语言的特点和面向对象技术。

FDD 将开发过程分为五个过程, 每个过程提供了一个简洁的过程指南使项目参与者能迅速进入自己的角色。每个过程指南 (不超过两页篇幅) 采用 ETVX (Entry Criteria, 入口准则; Task, 任务; Verification, 评审确认; eXit Criteria, 出口准则) 方法描述, 并明确了哪些角色参与哪些子任务, 哪些子任务是可选的、哪些是必须的。

3.5.1 FDD 的开发过程:

(1) 总体模型开发 (Develop an Overall Model)

成立建模小组, 首先由领域代表 (Domain member) 对该系统领域作要领式、框架式的概述, 然后在总设计师 (chief Architect) 的指导下, 各主程序员 (Chief Programmer) 分组进行细化, 再汇总调整形成总体模型, 识别出系统中的对象 (对象的行为和属性) 及对象之间的联系, 形成非正式的特征点列表。

(2) 构造特征点列表 (Build a Feature List)

在上一阶段的基础上, 将对象的行为转换成特征点, 形成按主特征点集、特征点集、特征点三层组织形式的特征点列表, 并将之按商务目标进行优先级排队, 将复杂的特征点分解成可两周内实施的小块。这一阶段是形成具体的、正式的、经排序的特征点列表。

(3) 按特征点计划 (Plan by Feature)

根据前面形成的正式特征点列表, 确定各特征点的开发顺序, 然后将特征点集分配给各主程序员, 主程序员再将类 (Class) 分配给各具体的开发者 (称为 Class Owner, 类拥有者, 只有他才有权修改该类代码)。项目经理、主程序员、类拥有者经协商后制订各特征点和项目的完成日期。

(4) 按特征点设计 (Design by Feature)

本阶段开始了为期两周左右的迭代过程, 一个迭代包括按特征点设计和按特征点实施两个活动。各主程序员将本次迭代需完成的特征点进行分析, 标识出涉及到的对象和类并与相关的类拥有者协商, 成立本次迭代的特征点组 (Feature Team)。特征点组讨论并画出具体的类时序图 (或协作图), 各类拥有者写出类和方法的概要, 并在主程序员的指导下进行设计评审。

(5) 按特征点实施 (Build By Feature)

各类拥有者完成类的编程、单元测试, 特征点组一起进行代码的评审, 提交配制管理, 在主程序员确认后, 将这些类代码打包形成可执行的制品。

3.5.2 FDD 对项目的管理:

(1) 项目组织。

采用主程序员方式,每次迭代都按特征点、以主程序员为中心形成临时的特征点组,成员为该特征点所涉及的类拥有者。一个主程序员可以同时带 2~3 个特征点组,一个类拥有者可同时参加多个特征点组。主程序员指导类拥有者,总设计师指导主程序员,从而加强了设计和实施的一致性。

(2) 计划和进度控制。

I FDD 提供了开发中五个过程的时间比例及每个迭代中各个活动所占比例,供计划和实施时参考。

总体模型	构造特征点列表	按特征点计划	按特征点设计和实施
14% (10%Initial, 4%ongoing)	5%(4%initial 1%ongoing)	4%(2%initial 2%ongoing)	77%

领域遍历	设计	设计评审	编码/测试	代码评审	提交
1%	40%	3%	45%	10%	1%

I 进度跟踪和汇报。每周由软件交付经理 (Release Manager) 召集各主程序员开会,由各主程序员介绍特征点的进展情况, Release Manager 进行汇总,用不同的色彩标示其进展状态,向高级经理和用户汇报,并张榜公布各主程序员、类拥有者的进展情况。每月以特征点集为单位向高层管理汇报工作进展。

I 如需加快开发进度,可通过增加主程序员来加强工作并行程度。

3.6 ASD 方法^[24, 25]

由 Jim Highsmith 提出,基于复杂自适应系统理论 (Complex Adaptive System),旨在通过提高组织的自适应力以应对 Internet 时代下极度变化、难以预测的快速软件开发要求,最近正与 Crystal 方法相借鉴和融合。

CAS 理论提出了三个主要概念: **Agents**、**环境 (environments)**、**浮现 (emergence)**。在一定环境中的 **Agents** 相互竞争和合作,导致系统产出的**浮现 (结果)**。对应于 CAS 理论,ASD 方法视开发组织为**环境**,组织中的成员为 **Agents**,产品为**浮现**。开发组织的首要目标是快速响应变化,即提高适应力,而适应力只能孕育,不能通过命令和控制来获得,Jim Highsmith 提出了领导-协作(leadership-collaboration)模式来提高组织的自适应力。ASD 提出了两个关键策略来建立自适应和协作的环境,一是管理人员要关注结果而非过程;二是管理人员要提供工具和技巧 (techniques) 来培养自组织能力。

关注结果要求迭代的开发方法,ASD 将每个迭代分为三个阶段: **思索 (speculation)**、**协作 (collaboration)**、**学习 (leaning)**。对产品的**思索**就是讨论和定义本次迭代要达到的目标 (ASD 认为通常计划过程实质上是思索过程);基于思索阶段定义的产品特征 (feature) 和持续的外部输入,项目组展开**协作** (开发活动);**学习**阶段就是通过思考对结果进行评审,并为下一迭代做准备。如下图所示。

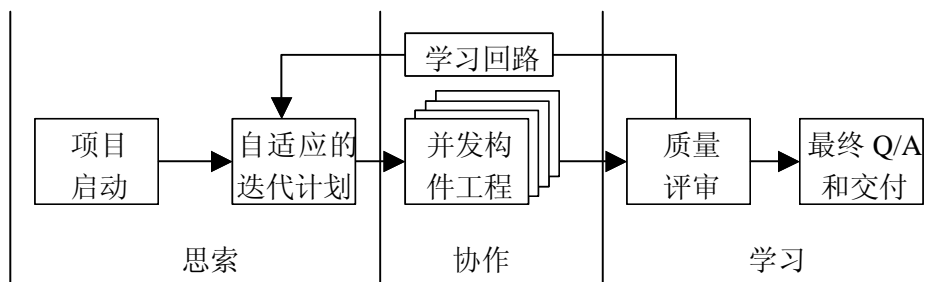


图 ASD 的开发周期

每个迭代有以下特点：

- l 基于项目洞察的使命驱动（Mission-driven）
- l 结果驱动（Result-driven）或构件驱动（每个迭代要开发出预定的构件）。
- l 每个迭代对应于一个限定的时间框，所有时间框之和为整个项目开发时间。
- l 风险驱动（risk-driven）
- l 容忍变化（change-tolerant，视变化为竞争优势和机会）

可以看出，ASD 是更加柔性化的开发方法。

3.7 其它方法^[31~33, 36, 40]

Open Source(公开源码)是一种完全基于 Internet 的开发方法，其典型的成功案例是 Linux。**Open Source** 项目一般设数个或十几个 Maintainer（源码库维护员），由 Maintainer 发布开发任务并对源码库进行修改。Eric Raymond 在 *The Cathedral and the Bazaar* 一书中对该方法进行了总结并提出了 19 条经验。

其它的 Agile 方法还有，Lean Development, Pragmatic Programming 等。

Rational RUP 与 Agile 方法的结合也是最近较热门的话题，Craig Larman 对 RUP 按 Agile 思想进行了剪裁，Robert Martin 将 RUP 与 XP 结合提出了 dX 过程（dX process）。

4. Agile 方法与 CMM 的比较

CMM 专家 Mark Paulk 认为^[9, 37]，CMM 更注重管理问题（组织过程的有效性和过程的系统化改进），Agile 更注重技术和效率；CMM 提供了一个高度抽象的框架，有广泛的适用范围，Agile 适用于小组织和需求不定、有用户紧密参与的情况（在高可靠性要求和大型项目组，或虚拟项目组中不宜采用）；CMM 与 Agile 方法虽有一些公共的特点，但 Agile 方法只是满足或部分满足了 CMM2 到 3 级中部分 KPA（关键过程区域）的要求。同时他也指出，Agile 方法提出了在某些周境下非常有效的实践，与 CMM 方法有一定的互补性，CMM 着重于“应该做什么”，Agile 则侧重于“如何做”。

XP 专家 Ron Jeffries 认为^[6]，XP 方法从某种意义上来说是 CMM2 到 5 级的一个垂直切片（满足了 CMM2 到 5 级中的部分 KPA 目标要求），若将之应用于整个组织则还须更多的度量工作，但他同时指出，XP 方法中更多的度量不是不可以做，而是要根据投入回报分析决定是否有必要。

不管怎么说，由 CMM 和 Agile 的产生背景有助我们对两者的理解，CMM（借鉴于 Crosby 的 MMM, Manufacturing Maturity Model）最初的提出是为了项目投标评估，这就使 CMM 强调过程的外部可观测性（带有外部包装的味道），实施时追求企业商业利润——这一最终目标显得不够突出和直接；Agile 是软件开发者在开发中被“逼”出来的，强调过程结果（可运行软件）的可观测性，更注重内在的生产率和商业价值（如 Mark paulk 所说，Agile 方法是为解决迫切的生存问题）。CMM 为企业过程改进指明了明确的方向，便于过程改进的具体实施，Agile 则讲究根据具体情况、商业目标要“刚刚好”，注重过程的持续不断的细调，实施时不易把握。相对于 CMM 来说，Agile 更注重个人及其技艺，适合于软件开发中对创造力要求高的场合。

5. 结束语

我国已正式加入 WTO，政治经济体制正发生着剧烈而深刻的变革，“推进国民经济和社会信息化，是覆盖现代化建设全局的战略举措”，对软件开发来说这意味着需求的迅速变化，

另一方面，我国 95% 以上的企业均不超过 500 人（美国的一些专家认为在 500 人以下的均属小型软件企业）。因此研究并实践 Agile 方法对我国的软件企业尤为重要，我们不仅要面对迅速变化的需求，还要直接应对国际范围的竞争。

Agile 方法也面临一些亟需解决的问题。一，对过程的管理要“刚刚好”（just enough），即不要太多也不要太少，那么这个“刚刚好”的尺度在实践中不易把握，需要进一步的理论研究；二，目前这些新方法只限于小型开发组织，如 XP 最好用于 10 人以下，Crystal 40 人以下，Agile 方法能否应用于大型组织？三，由于目前的大多数软件开发组织采用的是传统方法，如何在 CMM 的框架下实施 Agile 方法？可否通过 Agile 方法的实施来帮助整个企业提高能力成熟度？四，Agile 方法特别强调人的重要性，那么如何考虑文化的差异，特别是中国的文化？等等。

References:

- [1]Agile Manifesto, <http://www.agilealliance.org/>
- [2]Martin Fowler, *The new methodology*, Development Magazine, December 2000.
Available on <http://www.martinfowler.com/articles/newMethodology.html>
- [3]Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, MA, 1999.
- [4]Beck, K. *Embracing Change with Extreme Programming*. IEEE Computer, 32, 10 (October 1999) 70-77.
- [5]XP websites:
<http://www.xprogramming.com>
<http://www.extremeprogramming.org/>
- [6]Chaelynne M. Wolak, *Extreme Programming (XP) Uncovered*, DISS 725 Spring 2001
- [7]Charles Poole, Jan Willen Huisman, *Extreme Maintenance*, 2001 XP Universe Conference
- [8]James Grenning, *Using XP in a Big Process Company—a Report From the Field*, 2001 XP Universe Conference
- [9]Mark C. Paulk, *Extreme Programming from a CMM Perspective*, Paper for XP Universe, Raleigh, NC, 23-25 July 2001
- [10]Ken Schwaber, *SCRUM Development Process*, (OOPSLA'95 Workshop on Business Object Design and Implementation)
- [11]Jeff Sutherland, *Inventing and Reinventing SCRUM in Five Companies*, 21 Septmeber 2001, Cutter IT Journal
- [12]Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, Jeff Sutherland, SCRUM: An extension pattern language for hyperproductive software development (PLoP: The 1998 Pattern Languages of Programs Conference)
- [13]Schwaber, Ken. *Controlled Chaos: Living on the Edge.* American Programmer, April 1996.
- [14]K.Schwaber's ,SCRUM web page: <http://www.controlchaos.com/>
- [15]J.Sutherland, SCRUM web page
:<http://www.jeffsutherland.org/scrum/index.html>
<http://www.tiac.net/users/jsuth/scrum/index.html>

- [16]Patterns Home Page , <http://hillside.net/patterns/>
- [17]Crystal website, <http://www.crystallmethodologies.org/>
- [18]Alistair Cockburn, *Agile Software Development* , Addison-Wesley Pub Co, 2001
- [19]Cockburn, A., *Just-in-time methodology construction*, presented at Extreme Programming and Flexible Processes, 2000, online at <http://members.aol.com/humansandt/papers/jitmethy/jitmethy.htm>.
- [20]Cockburn, A., "Selecting a project's methodology," IEEE Software, July/August 2000, pp. 64-71 <http://members.aol.com/humansandt/papers/methyperproject/methyperproject.htm>.
- [21]Cockburn, A., " Characterizing People as Non-Linear, First-Order Components in Software Development", 4th International Multiconference on Systemics, Cybernetics, and Informatics, Orlando, FL, July, 2000. Online as Humans and Technology Technical Report, TR 99.05, at <http://members.aol.com/humansandt/papers/nonlinear/nonlinear.htm>.
- [22] FDD Website
<http://www.togethercommunity.com/>
<http://www.nebulon.com/>
- [23]Coad, Lefebvre, De Luca., *Java Modeling in Color: Enterprise Components and Process*. Prentice Hall ,1999
- [24]James A. Highsmith III. *Adaptive Software Development*. Dorset House, 2000.
- [25]ASD website: <http://www.adaptivesd.com/>
- [26]DSDM website: <http://www.dsdm.org/>
- [27]Wegner,p.(1995). "Interactive Foundations of Object-Based Programming." IEEE Computer 28(10):70-72
- [28]Wegner,P.(1997). "Why Interaction Is More Powerful Than Algorithms." Communications of the ACM 40(5):80-91
- [29]Ziv,H. and D. Richardson(1997). *The Uncertainty Principle in Software Engineering*.ICSE97,19th International Conference on Software Engineering, Boston,MA,IEEE
- [30]Agile Modeling website: <http://www.agilemodeling.com/>
- [31]Open Source website: <http://www.opensource.org/>
- [32] Eric Raymond, *The Cathedral and the Bazaar*,
available on: <http://tuxedo.org/~esr/writings/cathedral-bazaar/>
- [33] Karl Franz Fogel,Karl Fogel, *Open Source Development with CVS*, Coriolis Group, 11/1999 .
- [34] Fowler,M.(2001). "Is Design Dead?" *software Development* 9(4)
- [35]Steve McConnell, *rapid development (First Edition)*
- [36]Rational RUP
Website: <http://www.rational.com/>
- [37]CMM Website: <http://www.sei.cmu.edu/>
- [38]SPICE Related Website: <http://www.sqi.gu.edu.au/spice/>
- [39] Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design* , Prentice Hall, ISBN: 0137488807 05/1998
- [40]Pragmatic Programming ,Related Website:<http://www.pragmaticprogrammer.com/>
- [41]朱三元、钱乐秋、宿为民. 软件工程技术概论. 科学出版社, 2001