极限编程中的质量控制

Quality Control in Extreme Programming

袁 静 （陕西西安中国西安卫星测控中心 710043）

摘要：极限编程是近年来非常流行的敏捷软件开发方法，与其它敏捷软件开发方法一样，强调软件开发过程的自适应性和以人优先的价值观[1]，这与传统的重量级软件开发方法强调对开发过程的控制相反。那么，敏捷软件开发方法能否开发出高质量的软件产品呢？本文探讨在极限编程中如何进行软件质量控制，使其既能保持“敏捷”的特点，又能作为一种成熟的软件开发过程，为客户提供高质量的软件产品。

关键词：极限编程 便捷软件开发 质量控制 软件测试 同行评审

1 引言

近年来在国际上，软件开发方法却在向两极分化：一极是以传统瀑布方法为代表的“重量级”方法，另一极是以极限编程[1]、动态系统开发方法[2]以及 SCRUM 方法[3]为代表的“轻量级”或敏捷软件开发方法。敏捷软件开发方法主要希望在两个方面解决传统的重量级软件开发难以解决的问题：一是需求的变更；二是软件开发中软件设计与软件编程难以截然分开，软件编码通常也是软件设计的继续，而软件设计中的许多错误都要等到编程阶段才能发现出来，为此，敏捷软件开发方法都采用迭代式开发过程，强调软件开发过程的自适应性。由于其低成本、灵活高效的特点，敏捷软件开发方法深得中小型软件企业的青睐。极限编程因为对测试的极度重视成为近年来最为流行的敏捷软件开发方法。国际上成立了极限编程组织，每两年召开一次敏捷软件工程及极限编程国际会议，由工业界和学术界共同参与讨论解决极限编程及其它敏捷软件开发过程中的问题，分享成功的经验。

敏捷软件开发方法能否开发出高质量的软件产品，是大家最为关注的一个问题。本文详细研究了极限编程中的软件质量控制方法和手段，探讨敏捷软件开发方法保持“敏捷”的同时，又能作为一种成熟的软件开发过程，开发出高质量的软件产品的可行性。

2 极限编程

极限编程（Extreme? Programming）是 SMALLTALK 编译器的几个主要开发者通过许多项目实践总结的一套方法，适用于 10 人以下的小型项目。它既是一个轻量级的、灵巧的软件开发方法，同时它也是一个非常严谨和周密的方法。它的基础和价值观是交流、简洁、反馈和勇气，即，任何一个软件项目都可以通过开发人员之间的交流进行改善；从简单做起；通过各种反馈信息增加软件功能、提高软件质量；勇于实事求是。极限编程是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期；通过积极的交流、反馈以及其它一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。它以其 13 个核心实践为标志[1]：

（1）现场客户（On – site Customer）。要求至少有一名实际的客户代表在整个项目开发周期在现场负责确定需求、回答团队问题以及编写功能验收测试；

（2）计划博弈（Planning Game）。将软件开发过程划分成许多迭代周期，在每一个迭代周期前结合项目进展和技术情况，确定下一周期要开发与发布的系统范围；

（3）小型发布（Small Release）。强调在非常短的周期内以递增的方式发布新版本，从而可以很容易地估计每个迭代周期的进度，便于控制工作量和风险，同时，也可以及时处理用户的反馈；

（4）客户测试（Customer Tests）。在每次小型发

布之前,由客户编写测试用例,参加产品的合格性测试;

(5) 简单设计(Simple Design)。代码的设计尽可能的简单,只要满足当前功能的要求,不多也不少;

(6) 结对编程(Pair Programming)。由两个开发人员在同一台电脑上共同编写解决同一问题的代码,通常一个人负责写编码,而另一个负责保证代码的正确性与可读性;

(7) 测试驱动(Test – Driven Development)。强调"测试先行"(Test Before)。在编码开始之前,首先将测试写好,而后再进行编码,直至所有的测试都得以通过;

(8) 设计重整(Refactoring/ Design Improvement)。又称代码重构,在不改变系统行为的前提下,重新调整、优化系统的内部结构以减少复杂性、消除冗余、增加灵活性和提高性能;

(9) 持续集成(Continuous Integration)。提倡在一天中集成系统多次,而且随着需求的改变,要不断的进行回归测试;

(10) 代码集体拥有(Collective Code Ownership)。开发小组的每个成员都有更改代码的权利,所有的人对于全部代码负责;

(11) 代码规范(Coding Standard)。强调通过指定严格的代码规范来进行沟通,尽可能减少不必要的文档。;

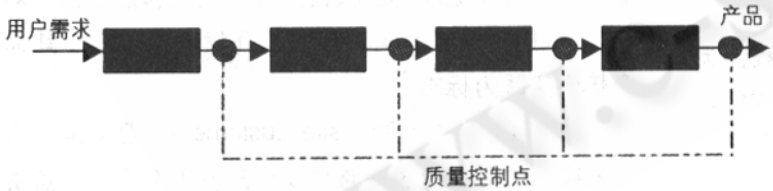

图 1 软件质量的粗控

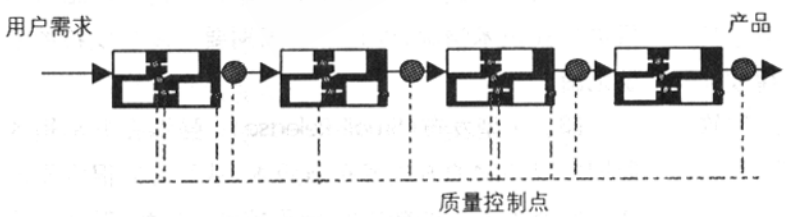

图 2 软件质量的精控

(12) 系统隐喻(System Metaphor)。通过隐喻来描述系统如何运作、新的功能以何种方式加入到系统;

(13) 适当的节奏(Sustainable Pace)。又称每周40小时工作制,极限编程强调一种轻松的工作环境,认为这有利于工作效率的提高。

这13个核心实践将极限编程组织成为一个和谐的过程,这种设计过程的结果是"纪律性"与"适应性"的高度统一,使得极限编程在敏捷软件方法中成为发展最好的一种方法。

3 极限编程中的软件质量控制

3.1 传统的软件质量控制过程

所谓软件质量,可以用软件中的错误密度表示。质量越高,错误密度越小,由于软件开发过程中随时可能引入软件错误,软件质量控制的目的就是要排除软件中的错误。传统上,评审和测试是进行软件质量控制的基本方法。评审是对软件的中间产品如软件需求规格说明、软件设计说明、软件测试设计进行错误检查和规范遵从性检查的主要手段,而测试主要是查找软件的最终产品,即软件运行代码中的错误。

软件开发过程的质量控制通常是与开发阶段的划分相关联的。在典型的瀑布开发过程中,软件开发过程通常划分为软件需求分析、软件设计、软件实现和软件测试等阶段。对软件质量进行粗控(如图1所示)时,把每一个软件开发阶段当作一个黑盒子,对每一个黑盒子的输入输出进行质量控制。该控制方法又可称基线控制方法:将软件开发过程划分为若干条基线,每一条基线意味着上一开发阶段的结束和下一开发阶段的开始,因此在每一条基线到来时需要评审或测试本阶段的工作产品,进行缺陷修正后,结束本阶段工作,而该阶段产品作为下一阶段工作的输入。

在有些项目中,需要对软件质量和软件过程作更加严格的控制,项目各开发阶段不再被当作黑盒子,而是一个透明的玻璃盒子,在各阶段内部过程中被插入若干控制点,使用多种方法如同行评审、走查、审计等方法查找软件错误,对软件质量进行监控。这种质量精控的过程如图2所示。

评审和测试是软件质量保证的基本技

术,在已定义的质量控制点使用软件质量保证技术是软件质量控制的过程化方法,如何保证这些方法的有效性,监督这些方法的有效执行则是质量管理的内容。

3.2 极限编程的质量控制

极限编程的实践中包含了严格的质量控制技术。

3.2.1 极限编程软件质量的外部控制过程

极限编程的整个软件开发过程是由若干小的迭代周期完成,每一周期实现部分用户需求,完成后进行一次小型发布,然后根据用户的反馈和进一步的要求,开始下一周期的开发。在每次小型发布之前,通过客户测试对该周期完成的产品进行合格性检查,而每当一个迭代周期完成后,需要确定下一周期要开发与发布的系统范围,以便开始下一次迭代过程。而下一周期开发范围的确定,是由现场客户参与确认的。极限编程软件质量的外部控制过程如图 3 所示。在此,现场客户的参与相当于从用户的角度对软件需求进行评审,而客户测试相当于传统意义的软件验收测试。可以认为,现场客户和客户测试两个核心实践是极限编程软件开发过程中每一个重要节点的质量保证手段。

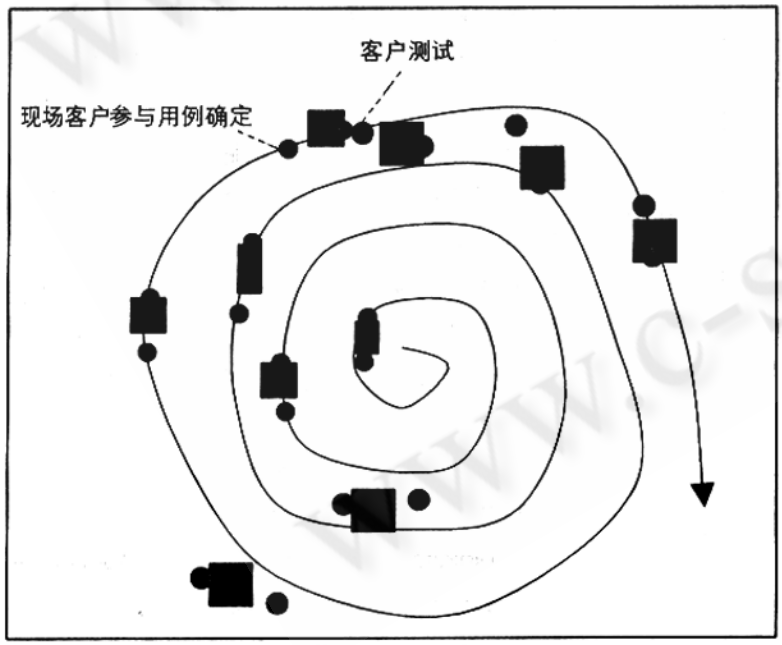

图 3　极限编程的质量外部控制过程

3.2.2 极限编程软件质量的内部控制过程

在每个迭代周期内部,软件开发过程仍然可以划分为软件设计、编码、单元测试等过程,但这些过程不再具有完全的线性关系,而是互相交错迭代(极限编程的每一迭代周期的实践过程流图如图 4 所示)。除此之外,极限编程不看重软件文档的作用,其观点是:编写规范、书写简洁、结构清晰的代码本身就是最好的软件文档。因此,在极限编程中,基于软件文档的软件评审不再适用,同时,烦琐的小组评审制也不适宜用于"敏捷"过程。尽管如此,极限编程却有更加严格的软件质量控制手段进行内部质量控制。事实上,它绝大多数核心实践的目的都是提高软件质量。

(1) 代码规范。代码的易分析性、易维护性是重要的软件质量因素,通过制定代码规范,极限编程让所有的程序员能够编写出风格统一、结构清晰、易于理解、易于维护的源代码,既是提高软件质量的重要途径,也是代码代替文档的重要前提。代码规范是质量控制的基础,而代码规范的依从性检查则是通过"结对编程"、"设计重整"、"代码集体拥有"等实践进行检查的。

(2) 结对编程。结对编程是一种典型的同行评审方式。该实践的宗旨是两个人的智慧比一个人强。由两个开发人员在同一台电脑上共同编写解决同一问题的代码,通常一个人坐在前端进行键盘操作,写代码,而另一个人坐在后面负责保证代码的正确性与可读性,相当于以个人评审的形式进行代码审查或静态测试。与此同时,另一个坐在旁边还可以考虑不同的设计思路和测试思路。

(3) 设计重整。又称代码重构,在不改变系统行为的前提下,重新调整、优化系统的内部结构以减少复杂性、消除冗余、增加灵活性和提高性能;当一段代码的设计不够完善、不够简洁、不够规范,对其进行设计重整,设计重整是同行评审或自我评审的结果。它导致更加完善的设计或编码。

(4) 测试驱动的软件开发过程。对测试的高度重视是极限编程的一个重要特点。极限编程中,测试划分为单元测试、持续集成和客户测试,这与传统的单元测试、集成测试和确认测试过程相对应。同时,极限编程强调"测试先行"的原则,即使对于单元测试,也要求在进行编码之前先写测试用例,既能在编写测试用例过程中从测试的角度对软件设计进行评审,又能保持测试设计的独立性。

通过一天中对代码的多次集成,不仅能够及时发

现软件的接口错误，还能及时得到现场客户的反馈意见，尽快对软件进行调整或修改。

通过客户测试，不仅可以验证软件产品对需求的满足度，还可以提高客户对软件产品的满意度。

综上所述，极限编程的核心实践提供了用于软件开发过程内部进行质量控制的一套精心的手段。在开发过程的每一个迭代周期完成后，通过客户测试完成系统合格性检查，而在迭代周期的内部，通过“结对编程”、“测试先行”、“单元测试”等实践，几乎在每一个设计、每一个模块编码完成后，都对其进行了错误排除，以期获得“干净”的软件模块；而每一次新的模块加入系统，通过“持续集成”，排除软件接口错误，又能得到“干净”的系统。

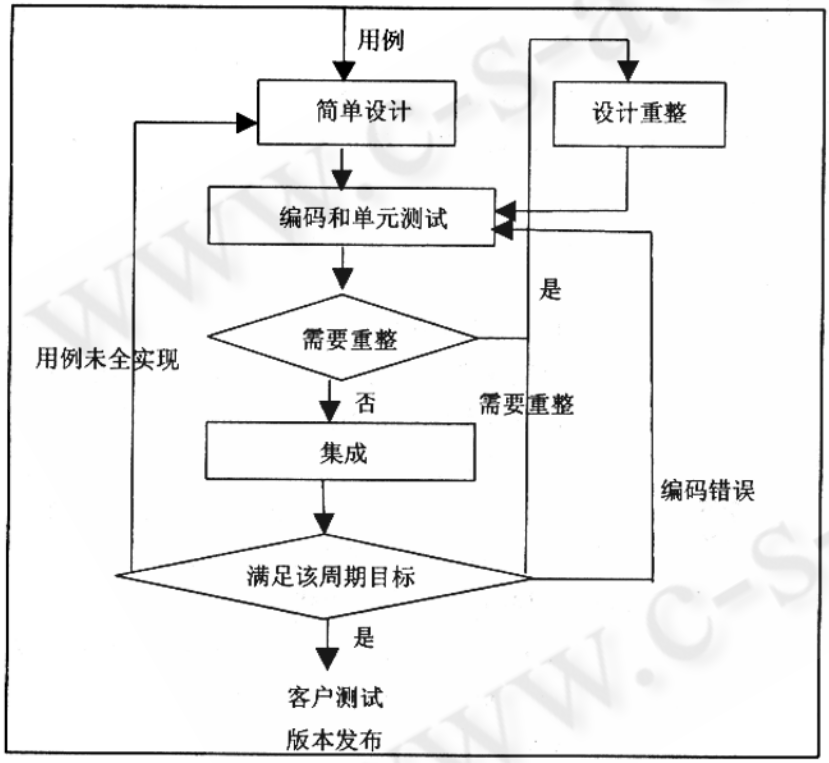

图 4　极限编程每一迭代周期内部的实践过程流图

3.2.3 “轻量级”软件质量管理

尽管极限编程有如此众多的核心实践支持软件质量的控制，可是这些核心实践能被不折不扣地执行吗？结对编程能象传统的同行评审那样有效吗？软件的测试设计足够发现软件错误吗？软件的测试是否 100% 覆盖了软件需求或设计？如何保证这些质量控制技术的有效性，是质量管理的问题。

极限编程中使用的是一种富有弹性的、轻量级质量管理过程。

首先，极限编程虽然没有强调专职的质量保证组织，但极限编程中的开发人员是“一专多能”，既要进行系统分析、软件设计，还要进行软件编码、软件测试以及代码审计，他们部分承担了软件质量保证工作。

其次，在软件质量管理中两个角色起到重要作用：一是教练（Coach），教练是一个软件开发和软件过程的行家，他的职责是密切关注项目开发过程，随时发现问题，解决问题，调整项目组的软件开发方向，以期达到最佳结果。教练是极限编程中举足轻重的人物，是项目的指挥。另一个角色是跟踪员（Tracker）[1]，跟踪员的职责是在不影响开发人员的情况下收集项目的过程数据如计划的执行情况、软件的缺陷数、软件错误的改正日志等等，他收集的数据是项目分析和管理的依据。

4　结论

极限编程通过软件的单元测试和集成以及客户测试，排除软件错误和对需求的偏离；通过代码规范、设计重整和结对编程，确保软件代码的可读性，从而确保软件的最终产品——软件代码的质量。通过项目跟踪人员的对软件质量进行跟踪和统计，通过教练校正软件开发人员未发现的问题和错误，实施轻量级的质量管理过程。通过一系列质量控制方法和措施，极限编程可以生产出高质量的软件产品，而其精干的人员配置和灵活的管理手段，又使它具备敏捷高效的特点。

参考文献

1 Beck Kent. Extreme Programming Explained. Embrace Change. Addison – Wesley, Reading, MA, 1999.

2 Jennify Stapleton. Dynamic Systems Development Method. Addison – Wesley, Reading, MA, 1997.

3 Linda Rising and Norman S. Janoff. The Scrum Software Development Process for Small Team. IEEE Software, 2000 17(4): 2 – 8.

4 Mark C. Paulk. Extreme Programming from a CMM Perspective. IEEE Software, 2001, 18(6): 19 – 26.