XP 轻量型过程方法实践

Practice Using XP Lightweight Process Method

刘鹏远 （湖北经济学院计算机学院 湖北武汉 430205）

摘要：统一过程是目前主流的软件过程方法，它强调用例驱动、架构优先和迭代式开发，但它的缺点也很明显：文档庞大，对 UML 的要求，各种建模对开发速度的降低。

极限编程是新出现的最著名的敏捷轻量型过程方法。它是一组简单、具体而互相依赖实践结合在一起形成的一个敏捷过程。它更重视设计模式和编程实践，有着诸多与传统的过程方法迥然不同的特征。本文结合笔者主持的项目探讨了 XP 的核心实践，给出了一些对 XP 实践的深入理解，并介绍了 XP 实施中的一些辅助支持工具。

关键词：统一过程 敏捷过程 极限编程

1 引言

一个软件企业如果实施了一个不好的软件过程，其软件产品必然是质量低劣的，由此可见软件过程实施的重要性。RUP[1] 对大多数的中小型项目而言，其人为的制品过多文档过大、团队组织复杂、实施成本太高，因而显得不太必要，与此相反，2004 年初诞生的轻量型软件过程 XP[2] 能够帮助中小型企业和中小型项目的开发。本人借最近主持一省部级工程项目的机会，采用 XP 来进行软件过程管理，本文将给出实施项目时应用 XP 核心实践的理解和体会。

2 核心实践

XP 适合中小型组织用于开发需求快速变化的软件项目。图 1[3] 展示它由 4 个方面组成：价值、原则、行为与实践，本文不准备阐述 XP 理论，而重点关注其核心实践的应用体会。

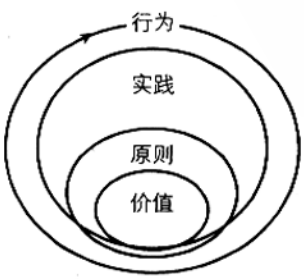

图 1 XP 组成

2.1 小型发布与持续集成

系统分析员应该指定提供客户业务价值的多次发布的周期。初次发布中的初几次迭代可能不具备客户价值[4]，例如展示用户接口的初步原型。本人在实际领导团队开发时，体会到尽早交付具有部分功能的系统和系统质量之间有很强的相关性；以逐渐增加功能的方式经常性地交付系统和最终质量之间有非常强的相关性，交付得越频繁，最终产品的质量越高。不断的小型发布意味着客户可以分步得到他的系统，这样可以在开发中一直导入客户的需求理解细化，而不是一直等待着最后的交付，后者很可能发生重大的需求变更风险。小型的发布需要系统组件的持续多次联编（集成），因此借助自动化工具来降低出错和提高效率，后面将给出几个强大的自动化工具实践。

2.2 简单设计与重构

设计必须尽可能地简单易被理解[5]，只应设计在本次迭代中要完成的用户故事而不考虑将来的那些故事，并寻找能够实现当前用户故事最简单的设计以及尽量不包含重复的代码。随着迭代周期的深入，简单可以在进程中逐步变得复杂，但是开始时应该尽量使得自然能够实现和尽可能抽象统一的方案。在多次迭代后，会逐渐发现一些重复的代码，为此还要经常使用重构技术来消除重复行为减少代码间耦合降低错误将来发生几率。

2.3 测试驱动

XP 依赖自动化测试工具来提供结果反馈，它强调测试先行设计[5]。正是由于测试先行，使得开发制品、测试制品以及交付产品间无缝隙。测试驱动非常有效而且意义生软，早已脱离仅仅是测试的范畴，本人将其归纳为以下三点：

(1) 测试驱动能够保证程序中的每一项功能都有测试来验证它的正确性,因为编写测试可以迫使我们使用不同的观察点,我们必须从程序调用者的角度去观察将要编写的程序。

(2) 为编写通过测试用例的代码(可测试代码),将激发开发人员解除程序中的耦合来使模块具有可测试性,这将迫使我们对程序结构做改善设计,提升了应用的扩展性。

(3) 测试可以作为一种交付产品时的产品说明书形式,比如如果想知道如何使用一个类或函数,可以直接对应观察其测试范例。测试驱动会迫使程序员即作者一开始就编写这种使用手册,无疑会减少后期产品交付时的文档编写工作量,并保证该使用手册的准确无误。

2.4 结对编程[6](pair programming)

XP 建议一台计算机由两人使用而共同完成任务。这是一个大胆而疯狂的设想,绝大多数项目经理会对此持否定态度,认为分工不清而且浪费人力工时。本人长期从事程序员工作,能感受到 XP 的确是更适合程序员的软件过程模型,极为推崇 XP 的这种观点,并将体会归纳为以下三点:

(1) 在软件企业内部,存在严格的模块分工和按能力划分小组,因而在整个项目开发周期内成员间普遍缺乏交流。如果能够使软件企业变成互相学习的课堂,会极大提高员工士气。结对可以极大促进知识在团队中的传播,鼓舞活跃气氛,让员工每天在汲取营养和激发灵感中快乐工作。当然,实际应用时,结对应因人制宜,结对分配得不好会很大影响项目进度和软件质量,特别是应针对员工的性格情商来做相应安排。本人的体会是,不对员工个性非常了解的项目经理也不是一个好项目经理,只有对各人的特性都很了解,多方面和他们谈心,才能做出最符合各自专长和性格爱好驱近或互补的结对安排。

(2) 互相结对有利于发现对方的错误。人的思维都有各自的特异性,而常见错误的发生一定程度上具备通用性。对某个员工,他可能对某个技术点十分了解,很少犯错误,写出的代码也极其优良美观,但可能在另一个技术点上他就比较模糊仅达到初学该项技术的掌握程度,因此需要有别的人来观察检查他的程序隐患。两个人可以在对方的编写中得到对自己的启发,更有助于发现对方的缺陷。

(3) 生活中的结对的例子并不鲜见,如会计与出纳、校对员与编辑。本人的理解是人天生对自己的创造会放松要求,认为是出色甚至完美的,这种心灵上的自我优越感会带来一些对代码错误和隐患的视而不见。只有结对,互相监督,才可能发现尽可能多的对方的错误。

2.5 编码标准化

编码的标准化不但大大提高程序的可读性,使一个人编写的程序具有"程序员世界通用"的特性,而且能降低隐藏错误。除此之外,标准化还可以为将来应用一些自动化 case 工具打好伏笔做好了应用准备。本人在采用 eclipse 平台的一些新插件做开发时发现,一些插件工具会试图理解代码而自动生成一些代码。如果程序员能够做到标准化,就会和这些工具配合得很好,反之则"受害"不轻。

3 实施周期

本人将应用 XP 实施项目过程管理的过程分为四个阶段,以下分别阐述。注意这与 XP 标准阐述有较多不同,但既然 XP 是来自于工程实践的软件过程,实用化的理解对领导实际项目开发实践显得更为适用。

3.1 需求分析调查

首先,项目经理应认真聆听记录客户对系统的需求陈述,这种陈述可能是非常详细的系统需求规格说明书,比如严格的欧美外包项目;也可能是非常抽象的只由 20 -30 词组成,这有可能是因为客户初期不能精确表达;更多情况可能是因为客户不明白需要什么,而只是为了为上项目而上项目;又或者是因为该客户不是最终的用户而是领导者岗位,缺乏对应用的现场了解。种种原因,导致该阶段客户对系统的需求陈述是非常梗概模糊和不准确的。此时 UML 图或任何可视化交互工具变得很重要。这里本人称为高级用户需求以及技术原型提供。开发组应该尽可能使用与业务域相关的语言来书写系统比喻,这种比喻是采用客户行业的语言来进行沟通,因此需要有深厚行业背景的供需双方人员都参与,达成一个简单的高层需求抽象。XP 的最大优点是可以不断设计持续发布和集成,每个迭代周期内都会有新的细化需求、设计和编码。因此,XP 该阶段显著有别于其它软件开发过程在调查阶段

花费大量时间来做需求的做法，它更重视在不断的发布和发布内的不断迭代中来细化需求。有了这个高层的需求抽象，就可以和客户一起探讨选择技术，完成一个用户故事列表或者用例。策略是只要客户能看懂和听懂各种表示和比喻，需求阶段的任务就完成了。大多数用户故事由 3 -4 句话组成，由客户编写，而此时开发人员估算实现时间。

(1) 编写用户故事。客户主要编写用户故事，随着项目的进展，客户会不断编写新的用户故事一直持续到项目完成。时间的估算需要修正参照，因此开发人员在初次发布开始时会原型化一两个用户故事来估算时间。对每个故事的初始估算是初略的，随项目进展在迭代级改进估算。有时还需对特定难点研究技术问题。原型化和研究难点 XP 中称为穿刺(spike)，穿刺辅助估算，帮助理解和选择典型的开发技术手段。

(2) 穿刺与估算。客户发现其需求，而开发人员发现可能的实现方法和估算，穿刺同时进行估算。首先需要知道某个具体目标的技术解决方案是否可行，然后估算完成这个任务所需时间——也就是完成任务穿刺花费的时间。

调查阶段相当于需求分析，但穿刺实际上也进行了编程实现工作(因此初步估算也是有根据的)，这些都不同于其它过程方法。调查阶段时间的长短取决于项目的性质与使用的技术，可以是几个星期或几个月。如果开发小组采用熟悉的工具和技术就可以减少用在穿刺上的时间。

3.2 制定发布计划

XP 放弃详细的需求分析，而通过快速迭代给客户提供直截了当真正的结果。发布时间表内划分多个迭代。该阶段客户决定用户故事的业务价值；开发人员向客户提示技术风险，估量预算和计划开发小组的进度，并和客户一起决定本次发布中包含的用户故事。

发布是一次较大的交付而迭代是一次较小的交付(本人认为迭代应以周为单位，应尽可能的短)。开发人员通过度量以前迭代中所完成工作量为本次迭代设定预算，在不超过预算前提下客户为本次迭代选择任意数量用户故事。一旦迭代开始，客户不可修改本次迭代中已选择的用户故事的定义和优先级别，但可以任意改变或重新安排项目中的其它用户故事。一次发布要经历几个月时间其内包含多次迭代，同样开发人员通过度量以前发布中所完成工作量为本次发布设定预算，在不超过预算前提下客户为本次发布选择任意数量用户故事，并可以随时改变计划的内容。注意该过程有客户参与，因而可以让客户全程参与项目的进程。

3.3 迭代内的任务确定

实际开发工作在迭代期间完成，迭代之初将召开规划会议，开发人员探讨将本次迭代中用户故事确定为多个编程任务，任务通常为 1 -3 个理想编程日。最初 1、2 次迭代可能侧重于建立基线系统所需架构级的组件(如建立数据库)而没有客户价值，而以后迭代将完成用户故事而提供不断增加的客户价值。用速度度量工作量，使用上次迭代完成故事数定义今次迭代应有速度。每日小组召开一次例会(standing meeting)以迅速发现问题和获悉进度。在迭代进行一半时间时召开一次中点会议，当未完成一半任务时，将设法重新分配任务和职责。若开发人员认为无法实现该分派，由客户从迭代计划中去除一个任务或故事(反之当然可以增加故事)。

开发人员编写测试，添加到测试框架中，然后以结对编程形式开始一个任务。

3.4 部署与维护

XP 是持续发布和不断增量部署系统的，因此在一次发布后而不是等到最后一次发布时才开始部署。这也是 XP 与其它软件过程的不同之处。产品通过核实和验证后开始部署到客户生产环境。专家组织最终用户开始验收测试，客户需提供一个筹备环境(staging environment)仿真实际生产环境，须建立独立筹备环境避免部署与维护阶段的混乱。部署要指定数据迁移和系统修补策略，这将影响开发小组的速度，小组应记录和度量新的生产量。部署后就是持续的维护与改进。

4 自动化工具

XP 强调持续发布与集成，以及测试驱动的开发理念，如果没有自动化工具的支持，发布与集成测试的工作量将无法忍受。本节给出重要的集成部署工具 Ant 和测试工具 JUnit 的使用方法。

4.1 JUnit

JUnit 可以应用于单元测试，也即对某个类模块的

测试。结合 TestSuite，它还可以完成对一系列测试类绑在一起的集成测试。首先需要下载安装好 JUnit（之前有 JDK），设安装在 C：\JUnit。set CLASSPATH =% CLASSPATH% ；C：\JUnit\JUnit. jar。然后按照以下方法安排测试。

（1）从 junit. framework. TestCase 派生一个单元测试用例类，将初始化分配资源的数据放在 setUp 方法中，将回收资源放入 tearDown 方法。然后定义返回类型都是 void，名称以 test 打头的各种测试方法。如果需要大规模的集成测试，就定义另外一个无须继承其它类的类，它只要定义一个 Testsuite 方法，其内将各单元测试类加入到它的 Suite 类的成员中。

（2）调用测试用例，命令行方式：java. junit. textui. TestRunner 测试用例类全名。

注意，先编写测试用例，然后从外部行为的角度编写实现功能的类代码，测试的功能就是调用了待实现类的 Public 方法，应尽量使两者方法名对应。类之间有依赖，因此，在最初的测试编写时，解耦合既是为了能测试也是为了应用程序的扩展性，对依赖的类使用 interface 达到目的。

4.2 Ant

Ant 是强大的集成工具，适用于快速和频繁的集成交付。从命令行运行 ant，它会读取解析其目录下的 build. xml 来执行集成构建过程，build. xml 的结构如下：

```
<project name = "test" default = "all" basedir = "." >
//project 标记是构建文件的开始
  <property name = "compiledir" value = "compile" />
//一个 property 标记内一个变量方便后面引用，可定义多个 property
  <target name = "all" depends = "all" >
//先读取 project 的 default 指示的 target，然后读取该 target 的 depends，
//按 depends 中从左至右顺序执行 target 中的任务，注意，depends 可能嵌套，因此可能不按一个 depends//描述的左右顺序执行。一个 target 只执行一次，不管出现在几个 depends 中。
  <javac srcdir = " ${compiledir} " /> //javac 是 java 编译任务标记，有许多内置的任务标记，
  ……
//如 junit，sql，vsscheckout，vsscheckin，csc（C#项目编译），exec 扩展任务等。
</target >
… //可有多个 target，从 project 的 default 开始分析执行顺序。
</project >
```

由上可知 Ant 强大构建集成能力通过其任务标记发挥。对于没有内置任务标记支持的，如 VB. NET 项目编译，可以使用 exec 扩展任务标记。下面是 Ant 运行前的设置：

（1）下载安装 Ant（之前要求有 JDK），设安装在 C：\Ant。

（2）set JAVA_HOME = JDK 所在目录；set PATH =% PATH% ；C：\Ant；。

（3）对需要支持的其它任务，如 VSS，要将 VSS 命令行 ss. exe 路径加到 CLASSPATH，并创建 SSDIR 环境变量指向 VSS 根目录；对 C#项目编译，将 C#编译器命令行 csc. exe 路径加到 CLASSPATH 中。

XP 还有其它众多可以使用的自动化支持工具，它代表着过程管理的新趋势：从越来越多的文档转向使用简单、清晰的工具，表 2 中列出了常使用的一些自动化工具。

表 1　XP 重要的辅助支持工具

Wiki Openwiki	共享式记事本，可任意搜索，适合用户故事编写与任务或任何交流，后者为 ASP 版本。
CVS VSS	源代码管理工具。前者支持各种操作系统平台，后者集成（VS）中。
JUnit NUnit	Java 开源工具，用于编写测试。与 Ant 配合自动化集成测试和构建。后者为微软提供的. NET 下类似免费工具。
Ant NAnt	Java 的 Jakarta 开源工具，其内任务支持 JUnit 与源代码管理工具（如 VSS）发挥强大的测试、集成功能。也可通过 exec 任务支持非 Java 项目开发（如. NET）。后者为. NET 下类似的免费工具。
CruiseControl	自动化监视工具监视源代码仓库，有新代码检入（check in）时触发构建。需要与 Ant、Web 服务器和某种 servlet 引擎（如 Tomcat）配合使用。

5　小结

XP 直接关注编程实践的管理而对文档和模型之

（下转第 109 页）

(1) ActiveX 控件

控件定义了两个方法：SetStatusText 和 UserTool，用于接收 MapInfo 的回调。

为了转发事件，控件还定义了两个 ActiveX 事件：ReceiveSetStatusText 和 ReceiveUserTool，其触发放在对应的方法中进行。

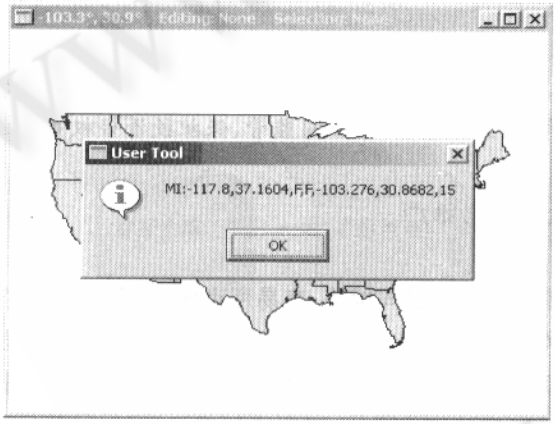

图 3

(2) Qt 用户程序

用户程序是一个单窗口程序，主窗口类为 Main – Window，在该类的构造函数中完成 MapInfo 的集成与回调。MainWindow 定义了两个槽：slotSetStatusText 和 slotUserTool，用于接收 ActiveX 控件转发的事件。

运行例程，定位地图文件 STATES. TAB，我们将看到地图被集成在窗体中，同时标题栏显示当前鼠标指针的地理座标。如果在地图窗口进行拖 – 放操作，将弹出一个对话框显示相关信息，如图 3 所示。

本文例程均用 VC + + 2003 编写，并在 Qt4. 2. 2、MapInfo8. 5 上调试通过。为方便读者，例程和参考文献 1 已放在互联网上，下载地址：

例 程 http://oldsong. wa203. bootchina. com/000/download/qt – integrate – mapinfo – sample. rar

参 考 文 献 1http://oldsong. wa203. bootchina. com/000/download/callback – pdf. rar

参考文献

1 祝晓鹰等，用 VB、VC 实现 MapInfo 的回调[J]，计算机系统应用，2004，7.

2 Jasmin Blanchette, Mark Summerfield. C + + GUI Programming with Qt 4 [M]. 2006.

3 王晓武、陈宗敏、杜兴国，MapBasic 程序设计[M]，北京：电子工业出版社，2000.