第30卷 第24期 计 算 机 工 程 2004年12月
Vol.30 №24 Computer Engineering December 2004

·开发研究与设计技术· 文章编号：1000—3428(2004)24—0189—03 文献标识码：A 中图分类号：TP311

敏捷开发：极限编程在管理信息系统开发中的实践探讨

邓靖颖，黄 穗
(暨南大学计算机科学系，广州 510632)

摘 要：极限编程是敏捷的和基于实践的软件开发方法学。通过介绍极限编程的特点及其在一个管理信息系统项目实际开发中的成功实践，探讨研究了极限编程对于中小型需求易变的软件开发项目应用的优势和不足。

关键词：敏捷开发；极限编程；管理信息系统

Agile Development: Practices for eXtreme Programming in Developing Management Information System

DENG Jingying, HUANG Sui
(Dept. of Computer Science, Jinan University, Guangzhou 510632)

【Abstract】 eXtreme Programming(XP) is a software development methodology that is agile and based on practice. This paper introduces the features of XP and how to practices XP in developing management information system, also discusses and studies the advantages and disadvantages when applying XP in small or middle sized projects with inconstant requirements.

【Key words】 Agile software development; eXtreme Programming(XP); MIS

目前国内很多中小型软件项目时间紧迫，需求经常发生变化，使用传统软件开发方法会导致开发资源浪费、质量低下。敏捷开发方法为有效解决这种状况提供了良好的解决方案。极限编程XP(eXtreme Programming)是敏捷开发的代表，是一个混乱而有序的、基于实践的方法，它通过非常短的迭代周期来应对需求的变化。广州某外贸公司准备开发一套外贸管理信息系统(FTMIS)，面临两个问题：(1)时间约束比较紧，在了解全部需求之前，项目就要被迫启动；(2)开发人员不足，团队只有3人，实际编码人员只有2人。基于这些客观原因，我们利用极限编程XP，与一名客户代表一起进行系统开发，最后在最短的时间内获得了客户最满意的结果。

1 XP方法特点

XP价值观如下：

(1)个人和交互高于过程和工具：软件是由人组成的团队开发的，要进行开发必须是程序员、测试员、项目经理、建模人员以及客户有效地工作在一起。

(2)工作软件高于详尽的文档：软件开发的主要目标是创造软件，而不是文档；相对于描述软件内部工作的复杂技术图或者对软件用法的抽象描述，用户会更容易理解所制造的软件。

(3)与客户协作高于合同谈判：只有客户能告诉你他们想要什么。成功的开发人员应该与客户紧密合作，花精力发现客户的需要，并在这个过程中引导客户。

(4)对变更及时做出反应高于遵循计划：随着系统工作的进展，项目关系人对问题领域的和正在构建的系统的理解会发生变化，业务环境在变化，技术随着时间也在变化，变化是软件开发的现实，一个软件过程必须反映的现实。项目计划必须具有可塑性，在形势改变时计划必须有变动的余地，否则计划会很快变得与实际情况不相关。

XP最需要优先考虑的是通过尽早地和不断地提交有价值的软件使客户满意，其运作项目的方法是将项目分成多次迭代，每一次的迭代交付一个通过质量检验、可投入使用、包含了一些新实现的用户故事(User stories)的软件。一个XP项目的状态变迁如图1所示。

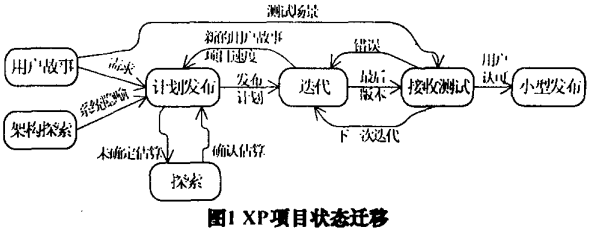

图1 XP项目状态迁移

图1中“用户故事”同时产生“需求”和“测试场景”。从需求定义开始，XP省略了常规的系统和架构的设计步骤，在进行初步“架构探索”后，就从简短的“计划发布”直接进入编码的迭代循环。“测试场景”则用来进行功能测试。编码和设计是同时进行的，而且特别强调测试的重要性，提倡测试驱动。测试驱动的编码方式实际是一个循环：写对应新功能的测试->运行测试发现错误->编写代码->运行测试成功->写对应新功能的测试。最后测试完成得到“用户认可”后进行“小型发布”。

2 XP在外贸管理信息系统开发中的实践

2.1 发布计划

很多项目都会在前期消耗过多的时间，只有等所有的需求完成之后才开始设计工作，采用XP方法，只需进行短时间的需求分析就可以开始设计工作。在经过一周多一点的需求分析和探索之后，FTMIS的几个管理模块就基本确定了，如图2所示，各个模块之间的数据交换一般建立在共享数据库上。

作者简介：邓靖颖(1978－)，男，硕士生，主研方向为网络与数据库应用、软件工程；黄 穗，副教授

收稿日期：2003-10-15 **E-mail**：iamwing@21cn.com

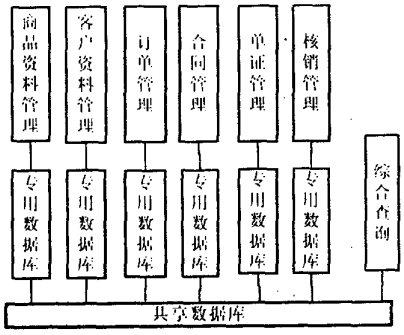

图2 FTMIS的组成

XP要求结合业务和技术情况，快速确定下一次发布的范围，即小型发布。一个版本的发布周期在1～3个月合适。客户代表确定系统的核心内容。甲方外贸公司主要做的是出口产品，所以该客户代表制定了第一次发布需要实现的内容(商品管理、客户管理、报价单管理、订单管理、出口合同管理、业务报表)，并将这些模块细化为各个用户故事(User stories)，加上程序员对这些故事的估算，定下第一个发布计划的时间是6周。客户对每个发布版本的选择在技术上可能不是最有效的，但他确保每个版本都给企业带来最大的收益，商业价值重于技术效率。在进行第一个版本的开发过程中，客户继续计划了第2个版本和第3个版本要实现的内容，整个开发周期的发布计划如表1所示。

表1 版本发布计划

版本发布计划	版本1	版本2	版本3
实现时间	第1～6周	第7～9周	每10～12周
主要实现内容	商品管理、客户管理、订单管理、报价单管理、出口合同管理	单证管理、核销管理、成本预核算管理、收付款登记管理、仓单管理……	其他合同管理、代理协议管理、快件管理、综合统计、用户备忘管理……

小型发布的形式可以让系统最快地投入生产，用户的意见也可以迅速地得到反馈，及时在系统中得到解决。

2.2 迭代计划

系统的一个版本被分解为几个一周的迭代，以便尽快地从客户代表那里得到意见反馈。迭代长度在项目开始时已选定，并在以后保持不变。客户代表为用户故事定下优先级后，程序员将这些故事分解为任务，每个任务的时间上一般不超过一二天。用户故事根据迭代的时间适当进行合并或拆分或移到下一迭代周期。FTMIS的第一个版本的迭代计划如表2所示。

表2 版本1迭代周期

迭代计划周期								
迭代1		迭代2			…	迭代7		
获得迭代2要实现的用户故事	实现迭代1用户故事	交付迭代1	获得迭代3要实现的用户故事	实现迭代2用户故事	…	交付迭代6	准备下一版本迭代计划和实现故事	实现迭代7用户故事

XP提倡模块间的松藕合，使系统因变更所受的影响最少。通常开发MIS的经验都是先做用户界面，再进行数据库设计，最后出报表。由于XP强调现场客户的参与，在本项目中该客户代表从用户角度考虑，在制定用户故事时定下了报表故事的优先级最高的要求。根据客户这一需求，我们先设计数据库，用实验数据或客户提供的业务数据加以填充，然后出报表。这样当报表模块开发出来并通过客户验收后，数据库结构就非常确定了。这时再进行界面的开发速度就提高很多。现场客户的参与保证了开发出来的系统具备用户最需要的功能。

2.3 开发

XP方法的一个重要开发原则就是结对编程。结对编程是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，通常一个人负责写代码，另一个负责保证代码的正确性和可读性。结对编程优点有：重要的设计决策至少由两个人决定，不易出现大的失误；至少有两个人熟悉系统的每一部分，保证了开发的延续，同时减少了文档编写的需要；两人可以很好地交流工具的使用，学习新的技巧。但并不是任何时候结对编程都是高效的。比如在调试改错的时候，有一个人坐在旁边并不会让思维敏捷一点。还有在做重复工作的时候也不需要，因为在此之前的结对编程已经解决了问题。

XP主张在实现功能代码之前，先编写单元测试。由于系统需求的不断变化，程序的测试也要频繁地进行，为了降低开发成本，测试的自动化变得非常重要。系统采用Delphi进行开发，所以我们将适于Delphi的测试框架DUNIT集成到开发环境中。DUNIT使用简单，如我们在开发中为编写一个加密解密的类TCipher的测试单元TestCipher，首先在单元TestCipher中引用测试框架单元TestFramework，新建测试框架的主类TtestCase的子类TCipherTestCase，并重载TtestCase类的Setup和TearDown过程，这两个过程分别在测试之前和之后被调用。接下来在过程TCipherTestCase.TestEncryption中就可以编写测试代码，主要代码及注解如下：

```
{测试TCipher的加密解密正确性的子程序}
procedure TCipherTestCase.TestEncryption;
var
  Src, Rlt: string;
begin
  Src := RandomString(32);      //随机产生32个字节的字符串
  Cipher1.EncodeString(Src, Rlt); //将Src加密得到Rlt字符串
  Cipher1.DecodeString(Rlt, Rlt); //对Rlt解密
  Check (Src = Res, '解密错误！'); //解密后的Rlt与Src不等的
//话，证明加密方法有错!
  end;
```

然后在单元最后的Initialization语句后加入

```
TestFramework.RegisterTest('TCipherTestCase.Suite'); //对本测试
//类进行注册
```

即可在程序初始化时对本测试类进行注册，之后在工程源文件引用单元TestCipher，并将APPLICATION.RUN;改为GUITestRunner.RunRegisteredTests;就可运行已注册的类了。

编写非GUI类的自动化测试单元所花费的时间不多，所带来的好处显而易见。但对有些GUI类的测试编写就比较复杂，费时费力，所以在实际运用中应只选择一定的合适的类来编写单元测试，避免得不偿失。

除了程序员编写的单元测试外，在每个迭代完成时还应由客户代表进行功能测试。实际用户和开发人员看问题的角度是有所不同的，尤其在需求不完善的情况下，每周的迭代完成后的用户测试使得意见反馈迅速而高效，减少了错误积累带来的项目失败风险，返工重构的工作量也不至太多。

整个FTMIS开发周期经过18个周的迭代，到最后第3个版本发布时，系统已经稳定地为企业服务多时。同时因为开

发过程比较严谨地遵守代码规范，很大程度减少了后期开发人员所不愿意写的技术文档。在迭代周期，用户故事制订得比较合适也使开发人员始终工作在松紧有度的环境中。

3 结束语

XP方法在FTMIS项目中的成功实施证明了以代码设计为中心的自底向上过程的合理性和有效性。国内诸多中小型项目开发时都是时间紧迫，需求分析过于简单，XP比之重量级开发过程，显得更有优势，同时也不难实施。但对于XP的一些原则实践，不能生搬硬套，应该灵活进行一些修剪补充，使项目获得最大的成功。

参考文献

1 Beck K. 解析极限编程：拥抱变化[M]. 北京:人民邮电出版社，2002

2 Ambler S. 敏捷建模：极限编程和统一开发过程的有效实践[M]. 北京：机械工业出版社，2002-04

3 http://www.extremeprogramming.org

4 张 恂. XP的价值和局限[J/OL].非程序员,2002,(15)

5 Newkirk J,Martin R C.极限编程实践[M]. 北京：人民邮电出版社，2002

☆☆

(上接第18页)

Step 2 for k=0 to $\sqrt{N}$ -1 par-do将∏(k $\sqrt{N}$ ：(k+1) $\sqrt{N}$ -1)中的元素循环向右轮转k mod w位, (k为各阵列的编号);

Step 3 COLUMNSORT /*列排序*/。}

PROCEDURE UNBLOCK1

{Step1 for k=0 to $\sqrt{N}$ -1 par-do将∏(k $\sqrt{N}$ ：(k+1) $\sqrt{N}$ -1)中的元素循环向右轮转k · $\sqrt[4]{N}$ mod $\sqrt{N}$ 位, (k为各阵列的编号);

Step 2 COLUMNSORT /*列排序*/。}

PROCEDURE SHEAR1

{Step 1 for k=0 to $\sqrt{N}$ -1 par-do将k为偶数的阵列∏(k $\sqrt{N}$ ：(k+1) $\sqrt{N}$ -1)中各元素按升序排序，行号k为奇数的阵列∏(k $\sqrt{N}$ ：(k+1) $\sqrt{N}$ -1)中的各元素按降序排序;

Step 2 COLUMNSORT /*列排序*/。}

PROCEDURE PARSORT4

{Step 1 对阵列的每个行块调用BALANCE($\sqrt[4]{N}$ ， $\sqrt{N}$);

Step 2 对整个阵列调用UNBLOCK;

Step 3 对阵列的每个列块调用BALANCE($\sqrt{N}$ ， $\sqrt[4]{N}$);

Step 4 对整个阵列调用UNBLOCK;

Step 5 For 0≤k≤2 do SHEAR;

Step 6 对每个子阵列中的各元素进行升序排列 。}

该算法的复杂性分析如下：在COLUMNSORT过程中，step 1和step 3可以在O(1)时间完成，根据PARSORT2的分析，step 2使用 $\sqrt{N}$ 个处理器对 $\sqrt{N}$ 个元素进行排序，在最好情况下O(log $\sqrt{N}$)时间，最坏情况下O($\sqrt{N}$)时间完成；在BALANCE1中，循环轮转可以采用多到多广播操作完成，其时间复杂度为O(1)，因而，BALANCE1的时间复杂度由COLUMNSORT决定,在最好情况下为O(log $\sqrt{N}$)时间，最坏情况下O($\sqrt{N}$)时间。同理，算法UNBLOCK1和SHEAR1的时间复杂度也由COLUMNSORT决定,在最好情况下为O(log $\sqrt{N}$)时间，最坏情况下O($\sqrt{N}$)时间，从而算法PARSORT4的时间复杂度在最好情况下为O(log $\sqrt{N}$)时间,最坏情况下O($\sqrt{N}$)时间。因而下列结论成立：在LARPBS模型中，使用N个处理器对一个具有 N 个元素的序列进行排序在最好情况下为O(log $\sqrt{N}$)时间，最坏情况下O($\sqrt{N}$)时间。在最坏情况下该算法的成本为O($N^{3/2}$)，比算法PARSORT1和算法PARSORT2的成本更优。

3 结语

基于流水光总线可重构线性阵列系统LARPBS是一种高效的新型并行计算模型，它将为光连接并行计算机系统结构的进一步研究与开发提供重要的基础，也将在很多领域得到广泛的应用。在本文中，我们介绍了Y.Pan提出的两种基于流水光总线阵列LARPBS模型上的快速并行排序算法，它们的成本在最坏情况下都为O(N^2)。笔者对这些算法进行了改进，提出了一种LARPBS模型上的可扩展的快速并行排序算法，对N个元素进行排序，使用p(1≤p≤N)个处理器，在最好情况下以O(NlogN/p)时间,最坏情况下以O(N^2/p)时间完成排序。另外还提出了一种LARPBS模型上改进的快速并行排序算法，该算法对N个元素进行排序使用N个处理器在最好情况下以O(log $\sqrt{N}$)时间、最坏情况下以O($\sqrt{N}$)时间完成排序。该算法的成本在最坏情况下为O($N^{3/2}$)，比Y.Pan提出的两种并行排序算法的成本更优。到目前，人们虽然已经初步研究并设计了一些问题的LARPBS并行算法，如矩阵运算、排序、选择、归并和图论问题等，但还有许多应用问题的并行算法及其可扩展性有待人们进一步研究和改进。同时，通过对LARPBS模型及其并行算法的进一步深入研究，有助于推动我国光互联技术和并行处理技术的进步，对光连接的并行计算机的实用化、开发新一代高性能计算机具有十分重要的意义。

参考文献

1 Pan Y, Li K. Linear Array with a Reconfigurable Pipelined Bus System ——Concepts and Applications.Information Sciences ,1998,106(3/4) 237-258

2 Pavel S, Akl S G. Computing the Hough Transformation on Arrays with Reconfigurable Optical Buses.In Parallel Computing Using Optical Interconnections.Li K,Pan Y,Zheng S Q(eds.)Kluwer Academic Publishers,Boston,USA,Hardbound, ISBN 0-7923-8296-X, 1998-10:205-226

3 Qiao C, Mei Y. On Efficient Embedding of Binary Trees in Reconfigurable Arrays with Spanning Optical Buses. Int'l Journal on Parallel and Distributed Systems and Networks, 1999,2(1):40-48

4 Pan Y. Basic Data Movement Operations on the LARPBS Model.In Parallel Computing Using Optical Interconnections. Li K,Pan Y,Zheng S Q, Eds., Kluwer Academic Publishers, Boston, USA, Hardbound, ISBN 0-7923-8296-X, 1998-10:227-247

5 Maberg J M , Gafni E. Sorting in Constant Number of Row and Column Phases on a Mesh.Algorithmica, 1988,3(4):561-572

6 Hamdi M,Tong J,Kin C W. Fast Sorting Algorithms on Reconfigurable Array of Processors with Optical Buses.International Conference on Parallel and Distributed Systems,1996:183-186