第5辑

荟萃全球软件智慧

Dr.Dobb's JOURNAL CHINA
www.ddjchina.com

软件研发

超越对象/ 轻型面向方面编程
项目管理/ 如何面对压力
编程艺术/ 耐心考验
算法蹊径/ Helix
嵌入式系统/ Spy

【专 题】

网络&安全

【软件春秋】
2002年
图灵奖背后的故事

in association with
Dr.Dobb's JOURNAL
C/C++ Users Journal
software development
windows developer
独家授权中文版

原版一览

Dr.Dobb's Journal

2003 年 12 月

【专题：数据库编程】

网络，ODBC 与 Perl

By Robert Kiesling

ODBC 解决了从各种数据源获取各种类型数据的问题，但是与网络连接的难题却没有解决。Unix 上的 ODBC 实现都只提供了对本机数据源的访问。本文中，作者通过一个 Perl API 和一个 P2P API 提出了一种解决方案。

应用程序级数据缓存

By Paul E. Boal

数据缓存可以极大地提高数据库的性能。本文中作者提出了一个应用程序级数据缓存库，可以将对 100 万个记录的查询时间从 4 小时减少到大约 4 秒。

Java NIO 与 iTunes 数据库

By Dimitriy Rogatkin

作者使用 J2SE 1.4 的新功能 Java NIO 拆开了 iTunesDB 数据库格式。后者可是目前很火的 Apple iPod MP3 播放器的核心。

【算法蹊径】

全文搜索与 Burrows-Wheeler 转换

By Kendall Willets

一种新型的全文搜索方法，可以在源文本中寻找任意字符序列，所需时间与序列长度成正比，所需空间比文本本身还要小。

【程序员工具箱】

Web 服务与 C++

By Peter Lacey

Systinet 公司的工程师阐述了如何使用该公司开发的 WASP Server for C++ 开发 C++ SOAP 服务和客户端。

【嵌入式系统】

代码效率与编译器引导的反馈

By Jackie Brenner&Markus Levy

【专栏】

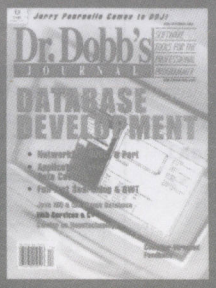
Michael Swaine 专栏

Ed Nisley 嵌入空间专栏

Jerry Pournelle 专栏

程序员书架

Verity Stob 专栏



Windows Developer Journal

2003 年 12 月

【专题：.NET 编程】

搜索健壮、安全的.NET IPC

By Shawn van Ness

在 VS.NET 中集成 DocExplorer

By Sushil Srivastava

处理多个 Win32 操作环境

By Matthew Wilson

编写自己的安装 / 卸载代码

By Alex Tilles

【专栏】

技术点滴

Windows 上 C++ “被零除” 行为考察

DBNull 形式的类型和相等性

AutoResource 模板

深入.NET

串行处理中的.NET 对象格式化

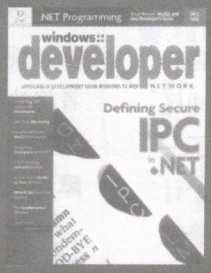
By Dino Esposito

Visual C++.NET 专家

数据封送

By Richard Grimes

书评



Software Development

2003 年 12 月

【设计中心】

灵活的工厂：构件软件时代是否已经来到？

by Clemens Szyperski & David G. Messerschmitt

【会议报道】

巅峰教导

【产品评论】

无线可不意味着无知 —— 安全产品、图书评论

By Rick Wayne

【书评】

苦尽甘来 —— 评《Bitter EJB》

By Sue Spielman

【职业发展中心】

完美的简历，系列之一

By Larry Runge

【项目管理论坛】

敏捷方法精要

By Robert C. Martin

【横切】

AOP 的关键

By Gregor Kiczales

【敏捷前沿】

正确的工具

By Scott W. Ambler

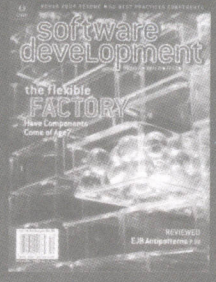
【编程艺术】

稳扎稳打

By Robert C. Martin

Warren Keuffel 专栏

公用出版（Utility Publishing）



C/C++ Users Journal

2003 年 9 月

【专题：Internet/ 网络编程】

用 C++ 模板构造静态分组

By Martin Casado

如何使用这项技术编写支持底层网络基本操作的通用例程

YAMI: 一种简单的 ORB 库

By Maciej Sobczak

YAMI 填补了原始对象间通信和复杂中间件技术之间的空白。其核心是一个由多个互相交换消息的对象组成的通信模型

Lambda 表达式与 C++

By Bjorn Karlsson

在许多函数型程序设计语言中，Lambda 表达式对于代码大小的精简是非常有用的，但是 C++ 中没有这种特性。Bjorn 使用 Boost Lambda 库解决了这一问题。

计算 Hough 变换

By Andrew Queisser

Hough 变换是一种在图像和数据点集中寻找几何形状的算法。本文中实现了线和圆的检测变换。

高效变量自动缓冲

Matthew Wilson

本文提出了一个简单的平台无关的 STLSoft 模板类 auto_buffer，实现了从封装了的数组中动态分配大小。

【专栏】

专栏: C++ Made Easier

Andrew Koenig&Barbara Moo

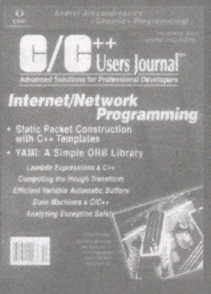
专栏:Generic<Programming>

Andrei Alexandrescu & David B. Held

Miro Samek 专栏：嵌入角

对话：流的重用

By Herb Sutter& Jim Hyslop



安全第一

2003年即将远去。对我们自己来说，这是应该载入历史的一年——《软件研发》横空出世，并且迅速成长。对我们这个行业而言，最热门的主题词又是什么呢？敏捷？服务？网格？无线？游戏？

我的选择是——安全。

现实生活中，安全与每个人都息息相关，我们许多人都会把“注意安全”常挂在嘴边。然而，在我们的软件研发行业里，长期以来对安全的重视远远不够。

这一点从《Dr. Dobb's Journal》上就能找到证据。创刊16年后的1992年，杂志才出现了“通信与加密”这样的专题，并第一次刊登了属于安全范畴的文章：Bruce Schneier撰写的《Untangling Public-Key Cryptography》（公钥密码学揭密）等。（推算起来当时Schneier大师不过而立之年，他的惊世之作《应用密码学》还没有出版。作为DDJ的中文版，我们第一次“网络与安全”专题有幸也能刊登他的最新成果，应该是一种天意吧。）

与此同时，计算机界一度还犯了把问题简单化的错误，将实现安全的注意力主要都放在了加密产品和其他安全软硬件产品上。似乎有了这些堡垒甲胄，我们就可以高枕无忧了。

在这种风气之下，安全的诸多重要方面：管理安全、物理安全、软件安全、数据安全等等被长期忽视了。其中尤其重要的是，整个计算机系统架构的设计中缺乏通盘的安全考虑，广大开发人员也缺乏编写安全代码的必要意识和素养（有多少开发人员能想到，自己指间飞泻而出的代码，会与千百客户的安全息息相关？高超如Microsoft，也只是到了2002年才开始在全公司范围普及安全编程的知识）。这种情况的直接恶果，就是直至今日，现有软件中仍然普遍存在大量漏洞，数量和严重程度都已达到非常惊人的地步：今年发布的主流操作系统、办公软件、即时通信软件……的新版本，有几个不是一经发布，就很快要打安全补丁？最近业内两大顶级厂商之间比较操作系统的论战中，居然用到了“你的安全补丁十几个，我的只有几个”这样的论据！而银行、保险、医疗、电力、电信等关键行业软件出问题的报道也同样不绝于耳。

随着网络的迅速发展，软件漏洞长期积累的问题开始总爆发，为害之烈，尤以今年为甚：以“冲击波”为代表，刚刚度过了20岁生日的病毒种类已近10万，垃圾邮件数倍于地球人口，还有如影随形的各类黑客、脚本小子……，它们的一点点动作，动辄就会造成十几亿、几十亿的损失！其中影响，相信大家都已经有切肤之痛。这一切追根究底起来，许多责任最终还是落在了开发人员身上：经典的缓冲区溢出漏洞，至今仍然在我们的代码中不时看到……

网络和软件的应用版图还在扩张：无线设备的普及使我们周围的天空也开始充满浓浓的电子气息，巴西通过电子投票选出了自己的新总统，而微软的操作系统已经进入自动取款机（均说最近还受到了Nachi病毒的感染），而且正在大举进攻汽车市场……可以预见，不远的将来，我们很少能有什么事情离得开软件，联网的软件。软件正在成为我们社会的基石，我们这个行业肩负的责任也随之越来越重。如果我们开发的产品，继续像现在这样动不动就门户大开，举手投降，不断酿成大祸，给客户给社会造成损失的话，必将被历史和市场淘汰。

怎么办？从我做起，从现在做起，关注安全问题，学习安全编程技术。本辑的专题文章正是从各种侧面介绍实际的安全技术和知识，力图给大家展示这一领域的丰富内涵。非常高兴能邀请到大家熟悉的潘爱民老师担任特邀编辑。

让我们共同谨记，安全永远是第一位的。

2003年12月

Dr.Dobb's JOURNAL CHINA www.ddjchina.com

软件研发

目 录

Dr.Dobb's JOURNAL CHINA www.ddjchina.com 软件研发

Feature: Networks & Security

Column

Management

Technology

执行主编：刘江
责任编辑：贾梅
编　　辑：吴怡 陈剑瓯
编辑信箱：editor@ddjchina.com
美术编辑：锡彬 禾余
排版制作：金浩
网　　站：周明涛

读者服务
电子邮件：reader@ddjchina.com
电　　话：88379061 88379062
网　　站：www.ddjchina.com

版权登记号 图字：01-2003-1691
图书在版编目（CIP）数据
Dr. Dobb's 软件研发．第5辑／（美）马丁（Martin.R.C.）等著；刘基诚等译．—北京：机械工业出版社，2003.12
ISBN 7-111-13664-0
Ⅰ.D… Ⅱ.①马… ②刘… Ⅲ.软件研发 Ⅳ.TP311.52
中国版本图书馆CIP数据核字（2003）第118818号

出版发行
机械工业出版社
（北京西城区百万庄大街22号 邮政编码 100037）
印刷：中国电影出版社印刷厂
版次：2003年12月第1版第1次印刷
889mm × 1194mm 1/16 · 7.5印张 16彩页
定价：18.00元

[编者按] 我们引进的《Dr. Dobb Journal》和《Software Development》都以读者水平一流而著称。这直接反映在杂志的读者来信中，与一般的杂志不同，来信中往往有丰富的信息量，有时不乏真知灼见。我们作为中文版，存在时间和内容上的差异，所以难于原汁原味地反映这些特色。本辑中我们选择了两个样本，借一斑而窥全豹，以飨读者。同时也希望我们的读者能够以这样的方式多多与我们、与大家互动。来信请寄：editor@ddjchina.com。来信的读者有机会获得我们的赠刊或其他奖品（比如优秀技术图书）。

基于 XML 的程序设计

亲爱的 DDJ，

我发现Gregory Wilson的文章“基于XML的编程系统”（《软件研发》第2辑）很有意思。和我有同感的读者，可能还会对JSML项目（由Wrox出版社的作者Erik Fuller和Michael Corning发起，请参考http://www.jsml.org/）感兴趣，这正是沿着Greg在本文中指出的方向向前迈出的一步。本质上他们通过一种XML结构让程序员能够使用标准的JScript编程语言构建对象模型，然后用XSLT将对象以标准JScript的形式输出。Greg文中提到了Lisp语言，这让我大觉心有戚戚焉，因为这种语言我曾经用了好多年。

Mike Morley
mikem@pandell.com

亲爱的 DDJ，

Gregory Wilson 的文章“基于 XML 的编程系统”（《软件研发》第 2 辑）用一个 JSP 实例开篇，其中用到了小脚本。这个例子其实可以通过 JSTL（JSP 标准标签库）直接用XML语法编写出来。如果使用XML语法的话，JSP页面可以是支持名字空间的XML文档。这样的话，该例子就变成了如下这个模样（可能有输入错误，请原谅）：

```
<html>
 <body>
  <table border="2">
  <c:forEach
    xmlns:c="http://java.sun.com/jstl/core"
    var="i" begin="1" end="3">
   <tr>
    <td>Number</td>
    <td>${i+1}</td>
   </tr>
  </c:forEach>
  </table>
 </body>
</html>
```

这样更有XSLT风味，当然这是一种命令式编程风格，而不是原来XSLT的递归模式匹配风格了。

Eduardo Pelegri-Llopart
Pelegri@sun.com

（Eduardo Pelegri-Llopart 何许人也？ Sun 公司杰出工程师，曾任 J2EE Web 层的架构师，负责 JSP，Servlet 等技术，目前是 Sun 公司 XML 和 Web 服务技术的总负责人。）

《软件研发》编辑部，

你们好：

我买到了《软件研发》的第3辑，感觉专题内容非常好！目前介绍软件模式的文章和书籍越来越多，但基本上都是对基础设计模式的介绍，如创建型模式、结构型模式和行为型模式。

其实，在企业应用这一层面上也存在着大量的模式，这些模式更加贴近企业应用的开发，但有关这方面的文章比较零散，不知是否有人整理出版这方面的书。《业务模式实例》这篇文章中所介绍的雇佣模式非常精彩、实用，还有许多资源和规则模式、目标模式以及过程模式是否会继续刊登？

非常希望你们能够出版一本有关业务模式方法的书籍，专门讲解资源和规则模式、目标模式以及过程模式的应用，最好有实例代码。这样学习起来也比较方便。

方智勇

DDJChina **回复：**《业务模式实例》一文节选自机械工业出版社即将出版的《UML 业务建模》（Business Modeling with UML）一书。此外，还有 Martin Fowler 著《分析模式》（中文版即将由机械工业出版社出版，影印版已经由中国电力出版社出版）讨论的也是相关内容。

《软件研发》编辑部，

你们好：

我认为《软件研发》的最大优势是拥有比较权威的外刊版权，而且内容比较偏向技术，不是一味的炒作，希望可以保持下去。我们需要的是思想和技术，不是什么崇拜，什么成功人士。

在这个浮躁的时代，希望贵刊能坚持住自己的风格，不断完善编辑水平，把最新最好最有前景的技术和思想带给我们中国人。不需要炒作与吹嘘，如果眼光没有错误，扎扎实实做出的好东西，就会成为精品。

祝您工作愉快。

Han Seven

DDJChina 回复：谢谢你的鼓励。国外的计算机技术杂志数以百计，而 DDJ 和 CUJ 等优秀杂志之所以能卓而不群，就是因为它们能不为流俗（hype，buzzword 等等）所动，坚持技术和实践为本。而 Software Development 又往往能做到冷静分析，高屋建瓴，破除迷雾。我们的原刊中很难看到哪篇文章完全是吹捧什么技术如何如何好，光动嘴不动手的。典型的 DDJ 风格的文章，往往是软件研发人员在一线实战中遇到问题、解决问题而有所心得，提升到理论高度，然后才著文立说的。

非常希望国内从事软件研发人员也能为我们、为大家撰写类似风格的文章。我们2004年的初步的编辑计划还在不断微调中，欢迎大家贡献宝贵意见。当然主要方向会长期保持稳定，欢迎投稿。来稿请寄 editor@ddjchina.com。

软件有多重要?

软件已经成为我们社会各方面不可或缺的因素，各种新产品的开发都越来越离不开软件了，不信？日经BP社最近采访了索尼公司总裁出井伸之，他在分析公司目前处于困境的原因时说："尽管我们也预见到了从模拟到数字的技术变化趋势，但索尼本身的发展速度未能赶上世界的变化速度。电视机由超薄显示屏取代显像管，**电视机开始成为软件的集成。也就是说，要将微软的要素导入索尼，同时还得和原来一样继续生产硬件。**由于短期内还需要两线同时做战、进行双重开发投资，因此利润率下滑是必然的。"

J2EE 1.4 正式发布

虽然讲述J2EE 1.4的文章和图书已经满天飞，但由于Java规范制订过程JCP的特殊性，最后的一锤定音却是拖了又拖。11月中旬，J2EE 1.4在千呼万唤之下终于出世：151号Java规范请求（Java Specification Request）在Java社区过程（Java Community Process）执行委员会以16比0获得通过，相应的软件开发工具集也可以从http://java.sun.com/j2ee下载了。差不多与此同时，Sun宣布第一个遵守J2EE 1.4规范的应用服务器Java System Application Server 8免费提供。要知道，这个产品原来的价格可是每CPU 2000美元，看来Sun在面对.NET和JBoss等产品的竞争压力下，确实开始全面修改自己的策略了。

正如我们在第二辑"XML与Web服务"登载的专题文章"J2EE 1.4 Web服务"一文中详细说明的，J2EE 1.4的主要新特性是遵守了WS-I Basic Profile（WS-I基本功能集）1.0，并使自己成为目前最完整的Web服务平台。过去独立的JAX-RPC（Java API for XML-based Remote Procedure Calls）现在已成为J2EE 1.4中至关重要的一部分。其他新特性还包括：

J2EE Management 1.0 API（使用了JMX），定义了J2EE管理的信息模型，包括标准的管理EJB（MEJB）。

J2EE Deployment 1.1 API，提供了J2EE应用程序部署的标准编程接口。

安全方面主要改进是JavaACC（Java Authorization Contract for Containers，Java容器授权约定）API，将鉴别机制融入了J2EE容器中。

Web层的关键要素是JSP 2.0（英文版《Dr. Dobb's Journal》2003年7月有专文介绍）和Java Servlet 2.4。JSP中的改进较大，引入了一种新的表达式语言EL（Expression Language，语法为${表达式}，可以直接嵌入页面中，在服务器端计算）；引入了标签文件机制和简化了的标签扩展API，可以用JSP代码实现自定义标签。Servlet主要增加了对请求侦听器的支持，改进了过滤器。

EJB 2.1中加入了Web服务端点、计时器服务以及对EJB QL和消息驱动Bean的改进。

部署描述文件现在改用XML Schema定义，后者还可以用于验证开发人员自己的XML结构。

C++ for .NET 标准化?

.NET环境下，标准C++应该何去何从？我们的老朋友、英文版CUJ的特邀编辑Herb Sutter报道，与微软关系融洽的非盈利组织ECMA已经成立了一个任务组（TG5）负责开发一组标准的语言扩展集，在标准C++和CLI（微软CLR的标准化版本，支持垃圾收集、安全等等）之间创建绑定。此举将使C++的开发人员能够更容易地利用CLI平台。

神奇小子又显灵

因开发了DVD解密软件DeCSS而名声大噪的挪威人Jon Lech Johansen，最近又开发了一个Windows命令行程序，可以将QuickTime的流截获并输出到一个文件中。这个程序开放了源代码，从而使Apple公司围绕iTunes展开的音乐下载业务（日前Apple曾宣布该业务的市场份额已经超过了Napster）也开始受到版权问题的困扰。

SCO 诉讼战

Linux社区最近真是多事之秋。最大的两家发行商都出了新闻：Red Hat明确表示将不再发行面向桌面的Linux产品，SuSE则被Novell收购。当然，处于漩涡中心的，还要算SCO单枪匹马对整个Linux乃至自由软件社区发起的挑战。虽然诉讼成为现实有可能要到2005年，但整个事件到目前为止已经是一波三折，极具戏剧性：

今年3月份，出身Linux社区的SCO集团（前身是Linux发行商Caldera公司，在购买了原Unix厂商SCO后改为现名）从好公民突然变成了社区"公敌"，公开指责IBM将版权属于SCO的Unix代码用于Linux 2.4和2.6内核中，因此违反了双方的协议，要求赔偿，金额达10亿美元（6月又提高到30亿）。此外，更警告广大Linux厂商和商业用户，如果不缴纳版权使用费，也有可能吃到官司。

5月，微软第一个肯定了SCO的举动，与之签订了价值约1000万美元的Unix相关知识产权的授权协议，引起普遍关注。同月末，与SCO有传统密切关系的Novell公司却坚定地站到了Linux一边，它宣布1995年将Unix知识产权卖给SCO时，保留了版权和专利权，因此SCO完全是无理取闹。但SCO很快就找到了对自己有利的文件。

6月，德国程序员组织LinuxTag成为第一家进行反击的Linux社区成员，它以伤害竞争者、恐吓用户、损害Linux名誉对SCO提出诉讼。

8月，Red Hat公司提出诉讼，控告SCO对Linux的指责影响了Red Hat公司的运营。一周之内IBM也出乎许多媒体意料没有采取收购SCO的措施，提出了反诉。大戏于是上演。月底，SCO网站遭到拒绝服务攻击。

9月，HP也来凑热闹，宣布如果自己的用户受到诉讼影响，公司将给予赔偿。而德国的官司以SCO被罚款10800美元并删除网站上指责Linux的言词而告终。

10月，现金只剩下1100万美元、眼见几百万、几百万花出去的律师费都可能掏不出来的SCO，忽然得到一笔5000万美元的投资，来自有微软背景的投资公司BayStar。虽然该公司否认投资与微软有关，

但是各种“阴谋理论”已经流传开来。也难撸　勺皇俏⑷愁S谴怯钟写　运钠穑　骸CO的诉讼大棒下一个目标将是另一个让微软头痛的对手——Google。

进入11月，SCO与IBM的诉讼战愈演愈烈，传票满天飞。最不可思议的是，SCO把偶像级人物Richard Stallman、Linus Torvalds都牵扯进来，此外连带把Linus的前后两任雇主Transmeta公司和OSDL，还有自己的老伙伴Novell公司都告上了法庭。IBM也毫不手软，分两批将“帮凶”BaysStar、德意志银行、复兴风险投资和Yankee集团以及Sun公司(最近与SCO签订了Unix许可扩展协议，并许诺将购买SCO的股份)、Schwartz通信公司、代表SCO的一家公关公司、诺斯洛普——格鲁曼公司请入瓮中。

整个过程中，让我们这些关注技术的人失望的是，SCO始终没有拿出有说服力的被剽窃的代码证据，而已经透露出来的代码片断，要么是Unix的传统代码，甚至是当年Ken Thompson亲手所写，在《莱昂氏Unix源代码分析》和Bach《Unix操作系统设计》等经典著作中早已经记载，要么就是BSD代码。而开源社区以及SGI公司的比对研究都没有发现明显的支持证据。最近，SCO更是敢于冒天下之大不韪，开始攻击整个自由软件的基础——GPL，说GPL本身站不住脚，违反了美国的宪法、版权法和专利法云云。这让人更加怀疑其真正动机。

而大量法律层面的东西，不谈也罢。但是可不要以为SCO完全是在表演恶作剧，你知道这次代表SCO的律师是谁吗？David Boise！这位著名高科技诉讼律师，曾经在美国司法部诉微软一案中，代表政府痛击开源社区的“公敌”，并因此成为《时代》杂志2000年年度人物候选之一。此外还参加过Napster诉讼案。此公从今年年初就开始准备这场官司，如果没有足够的回旋余地，是不会轻易发难的。实际上，其中可以做文章的地方极多，且不说涉及源代码的官司本就技术性较强，很难认定和反驳，纷纷扰扰的Unix历史更是公案多多，就是GPL本身，也的确没有经受过法院的考验。当然，IBM从3月份成为被告，到8月份选择了反诉，这里头也肯定经过了充分的准备的。有意思的是，IBM选定的律师行正是David Boise单干之前，效力多年的老东家。

12月初的聆讯中，法官已经要求SCO首先拿出表明IBM侵犯了版权的源代码，让我们拭目以待。

Geronimo项目与JBoss

最近非微软圈子里真是不太平，这边Linux社区被SCO关于Linux盗用Unix源代码的诉讼搅得鸡犬不宁，那厢J2EE社区又因为JBoss指控Geronimo盗用其源代码而吵得一塌糊涂。看来开源运动发展到现在，虽然是一派欣欣向荣，但成长的烦恼也随之而来了。

PDC 2003

上月我们曾简单介绍了PDC 2003上的主要动向。对于国内目前占很大比重的Windows开发人员而言，此次会议中所透露的各种信息的重要性，绝对不亚于甚至超过了2000年奥兰多发布.NET的那一次PDC。从各种产品的命名上看，微软正在超越.NET。WinFX这个已经注册了商标的技术，不仅将取代我们的老朋友Win32 API，而且还会取代.NET框架。

除了编程模型的改变将再一次使开发人员“疲于奔命”以外，预计新一代操作系统会在界面和使用感觉（也就是文绉绉的“用户体验”）上让我们大吃一惊。类似于目前网页上众声喧哗、四处跳动闪烁的效果已经可以想像，更大胆的远景也许还在开发之中，比如三维、人性化的智能主体（还记得微软曾经推出的Bob吗？）、语音交互等等。也许几年后，我可以不用敲字而是口述，就能给大家写“软件近事”了。

由此可以想像，微软的产品还会继续开疆拓土，像语音识别、图像浏览和处理、网页制作、动画制作、小型办公信息系统开发等等功能，都会很自然地融入Windows或者Office当中。最近已经有消息说，微软正在开发一种绰号为“Flash Killer”的多媒体制作软件。由于Longhorn操作系统本身可能会将视频、动画等等作为内置对象类型，这种动向预示的威胁可不是闹着玩儿的——微软的大手也许会向图形图像工具等领域前进。

Java开发工具之春秋时代

与.NET平台开发工具基本上是微软和Borland两分天下不同，Java开发工具还处于春秋时代。扳着指头数一数：商业产品有Borland的JBuilder、IBM的Websphere Studio、Oracle的JDeveloper、BEA的WebLogic Workshop、Sun的Forte for Java……，开源产品有Eclipse、NetBeans等等。

现在越来越多的人认识到，Java阵营同微软的差距主要在开发者亲和性上。虽然后者一直由于其政策和形象背负骂名，但是大多数开发人员还是喜欢整日在微软的IDE中编写程序，还有更多的程序员和入门者使用Windows和.NET平台上的VB和Delphi，这种耳濡目染之下，久而久之，主要的IT人群对微软的技术和产品当然更加熟悉和了解，这对于整体市场的影响不言而喻。也难怪微软内部把VB作为自己成功的要素之一。在此背景下，各家Java大厂一方面加强工具开发力度，一方面纷纷开始合纵连横，可是看起来难度不小。为了实现Java开发人员达到1000万的目标，Sun公司下的本钱不小，Gosling被调到开发工具部门就是明证。近日代号“Rave”的全新IDE beta版已经可以下载，正式名称为“Java Studio Creator”，开发速度可谓惊人。Oracle等厂商则琢磨着想弄一个Java工具联盟组织，但是IBM和Borland又没有加入的意思。Eclipse项目虽然参与者挺全，但是独独缺了Sun。IBM一直和Sun商量将Eclipse和NetBeans合并，而且许诺只要Sun合作，Eclipse（意为“日蚀”）可以改名，但最终还是不欢而散。看来在开发工具上Java社区要赶上微软，还需时日。

（更多新闻和评论，请访问www.ddjchina.com）

【会议报道】

2003'软件研发最佳实践大会（下）

9月15~18日，《Software Development》主办的“软件研发最佳实践（SD Best Practices）”大会在美国波士顿召开。会议除主题发言之外，主要由各种技术报告会、讨论会、教程组成，涵盖了软件研发生命周期的所有阶段：“需求与分析”、“设计与架构”、“测试与质量”、“构建与部署”、“人、项目与团队”、“过程与方法”。上个月我们报道了Watts Humphrey和Ed Yourdon的主旨报告，还关注了会议上非常活跃的我们的新劲敌：巴西软件业的动态。

从会议各研讨班的主题来看，目前敏捷方法是大家关注的焦点中的焦点。其中敏捷方法相关的课程占据了一半左右，探讨和应用的深度和广度都超过了以往。我们曾在第一辑专题介绍过敏捷方法，所取材料很多介绍的是敏捷运动发展早期。那么两年多之后，情况又怎样了呢？本月，我们首先听听《Software Development》的编辑Laurie O'Connell是怎么报道本次会议上三位运动领导人共同参加的主旨发言的。

敏捷方法三杰：Martin Fowler，Robert Martin和Ken Schwaber

Robert Martin是著名的过程改进和面向对象技术咨询公司Object Mentor的创始人和CTO。他是《C++ Report》杂志的最后一任主编。所著《Agile Software Development: Principles, Patterns, and Practices》一书获得了2002年Jolt大奖。他是《Software Development》的顾问和专栏作家。

Martin Fowler是Thoughtworks公司的首席科学家，面向对象和软件方法学领域世界知名的专家，他撰写的《Analysis Patterns》、《UML Distilled》、《Refactoring》都已成为经典。最新作品《Patterns of Enterprise Application Architecture》又喜获2002年Jolt效率奖。Kent Beck曾开玩笑说：“Martin是我见过的头发最少而想法最多的人。”他是《Software Development》的顾问。

Ken Schwaber是Advanced Development Methods（ADM）公司的总裁，是一位资深的软件开发者、产品经理与专业顾问，经验丰富。在20世纪90年代早期，Schwaber发起了过程管理产品革命，并且与Jeff Sutherland一起开发了SCRUM开发过程的最初版本。

Agile 2.0

——三位业界名人来自一线的成功故事

在不太切题然而非常搞笑的主题曲中，三位敏捷运动的领导人，《敏捷宣言》（参见《软件研发》第一辑）作者，敏捷联盟成员Martin Fowler，Robert Martin和Ken Schwaber同时登台，开始了名为“敏捷运动新发展（Agile Update）”的主旨报告。

这场报告的主题称为Keynote（主旨）自然是实至名归——毕竟整个会议都带有浓浓的agile气息。满场的听众只能全部站着，这倒是符合敏捷方法开会的要求——站立会议（standing meeting）。切入正题之前，三位报告人首先用笑话引出了报告内容概述。其间几个人还在开发迭代的理想周期和索引卡的最佳大小几个问题上发生了争执，也不知道是真的，还是事先安排好的。

SCRUM方法创始人Ken Schwaber承认，在刚开始，SCRUM方法在各公司的推行确实面临管理层的很大阻力。但是，现在这一难关已经渡过，他的兴趣从如何生存开始转向展望充满希望的未来：“我们应该往哪里发展？怎样度量敏捷的程度？怎样更好地宣传推广？怎样让更多的人理解敏捷原则，而不是在没有了解基础的情况下就贸然采用？”

Schwaber重申了全面理解敏捷原则比仅仅采用几个实践要重要得多：“这仍然敏捷联盟的惟一工作：研究应该采取什么措施，防止敏捷方法仅仅成为另一种赚钱的方式。”

态度转变

虽然三个人在积极应对变化、平衡规范与灵活性等理念上是一致的，但细节上还是有明显区别。Schwaber和Fowler推崇的纯粹主义方法，“Bob大叔”就不太感冒，他的态度更接近折衷主义。Fowler说敏捷方法可以个别地使用，但是他推荐尤其对于XP，最好“一下子就跳进冷水中，而不要一次沾上一点。必须理解的是，[XP的]所有要素，虽然有些乍看上去并不自然，但都是相辅相成的，通过几个月的训练之后，就能很好地协作……逐渐和部分的采用会使你产生错觉。”

Schwaber对此表示同意，但是又加了一些条件：“你可以循序渐进地进行，但是理论上这还是一种变革。你可以进行计划，但是必须用自己的脑子决定怎么做；从看待工作和同事的方式上来说，这都是一种模式上的变革。”

Martin表达了更宽容的态度，他承认“XP本身是一种不可分割的整体”，但是他又说“人们已经成功地在一些公司中分别使用其中一些技术。”他力促与会者“像科学家那样冷静观察，大胆实践。”

近年来，敏捷阵营成长很快，迅速脱下了褴褛装，换上了漂亮的西装。Fowler这样描述已经进入主流的敏捷方法：“两三年前，我们主要的关注点是业务的发展，做销售的人都说：‘别那么强调敏捷和XP，我们不想对客户说这些。’而如今，他们嘴边挂着最多的变成了敏捷如何如何‘好啊，好啊’。这是一种明显的标志，表明使用敏捷技术和极限编程的兴趣在不断增加，越来越多的人开始认真考虑了。”

“应该把闲聊的、不太正式的、没人会写进正式交流文件中的内容也传达出来，这对于了解深层的情况至关重要。”

Martin Fowler

从怀疑到狂热

接下来，三位专家讲述了几个来自一线的真实故事，几家公司通过引入敏捷方法，极大地减少了产品的错误率，而且都经历了随着快速迭代带来的代码质量的提高，和管理层从最初的怀疑到最终变成狂热分子的过程。

“最训练有素和富有经验的那些XP小组都显著地降低了缺陷率，”Fowler提到。他承认已经出版的书里对成功案例介绍得远远不够。“我最近回到克莱斯勒公司，拜访了C3项目的老同事，对，就是创造了XP的那个传奇的失败项目。”（笑声）“这个小组从2001年开始为克莱斯勒的人事资源部门开发Web门户软件。2002年，他们的系统完成，整年中只发现了一个bug，不过是有些库中语义发生了变化。”台下一片惊叹声。

Fowler又举了一个企业级软件开发公司的例子，该公司200多个雇员全部转向了XP。“他们有大量遗留代码要处理，全面掌控很不容易。各个小组的改善幅度不太一样：有些非常积极，有些则不太明显，主要是因为那些项目中小组成员本身水平就非常高，而且没有遗留代码，但是在整个8个月的开发中，没有bug报告。”

Fowler认为这些例子说明了训练有素和富有经验的XP小组能够达到怎样的水平。而一家英国公司就没那么好了，这是一个“不成熟的问题小组”，有30～40个开发人员从事着几个小项目。“他们的效率很差，但通过引入XP实践，公司的水平有了很大提高，四个月以后，他们看上去已经像是一个顶级团队了。”

Fowler还讨论了他自己所在公司ThoughtWorks在大型分布式项目中应用敏捷方法的雄心勃勃的计划。“2001年，ThoughtWorks公司决定在印度班加罗尔开设一家开发公司，设置了一个三四十人的研发中心。我们的招聘标准是非常高的，在国内，我们一般每200个应聘者中可能才聘用一个人，而在班加罗尔这个数字接近于千里挑一。此外我们的开发方法都是相同的。”打破个人之间的天然隔阂是很大的挑战，他承认，然而“面对面的交流正是敏捷的价值之一”。

他继续说，过去几年中，公司有两个项目将很大一部分开发移到了印度，

另一个项目是在英国进行的，整个任务由英国和美国两地承担。“我们发现，到目前为止，在这样跨度的地域分布下，敏捷实践很多还是可以实施的，但是生产率上要付出一些代价。”而成功的要素是整个项目要共用一个代码库，而且有持续集成过程，这样多个小组才不会南辕北辙。

Fowler还高度赞扬了在各个分布的地点之间来回跑的关键角色——“大使”。虽然“来回穿梭永远比不上走到开发者的桌边有效”，但是他特别强调这种穿针引线的活动中“应该把闲聊的、不太正式的、没人会写进正式交流文件中的内容也传达出来，这对于了解深层次的情况至关重要。”

180度的大转弯

Martin接过了话筒，给大家讲述了自己指导的一个项目中发生的故事。在Martin进行指导之前，项目的甲方——一个公司的营销部门非常不高兴，他们抱怨说：“都6个月了，我们什么都没有看到，开发人员不知道都在干些什么。”在小组中建立了快速迭代的敏捷规则之后，很快，事情有了改观。“甲方的态度有了180度的大转弯，从不信任和失望变成了对下一次迭代无限期待。他们现在可以满怀信心地对客户承诺了——不是因为开发人员有许诺，而是因为每两周开发出来的东西看得见摸得着，速度都在那里摆着呢。”

这种速度仅仅是取得更重要的优势——质量的一种手段而已，Martin解释说。“他们的软件质量可以达到这种程度，销售人员只要到开发部门去，拿到这两周的版本，就可以到现场去演示了，不用担心有潜在的bug。”

Martin还举了一个银行开发小组的例子：“这个小组成员都非常年轻，充满活力，但是由于管理层制定的目标不切实际，他们陷入了最后期限的怪圈。最后项目超期了，大家都感到很歉疚，经理们也很不安——每个人都有挫折感。在我们介入以后，开始按两周迭代的节奏进行开发，整个情况大为改观。我们把直方图画在墙上，这样每个人都可以看到进度了。经理们也不再强行制定愚蠢的期限目标了，他们努力找出真正切实的最后期限，把特性都摆出来，重新排列优先次序。”

> “以前，我对自己编写的程序充满自信，但是对编写这些程序的方式就信心不足了。今天，我不仅对软件非常自豪，对开发的方式也同样骄傲。”
>
> *Robert Martin*

Martin讲述的另一个成功故事发生在分为三个小组的团队中。“在制定了XP过程之前，他们每两个人-日就会出现一个bug。现在，他们每20个人-日才会发现一个，缺陷率上差了一个数量级。每个版本发现的bug数按4倍的速度不断减少，而发布周期中没有发现的bug数减少速度是10倍。”

Martin说，如果经理们能看到明显的进度，他们就不会再设定不可能实现的期限，而是慢慢地开始考虑真正的管理措施了。

规范出自灵活

Martin深情地谈到自己对编写代码的热爱，这使许多听众产生了很强的共鸣。但是Martin也承认，多年来，他对“自己能够编写很好工作的代码，却不知道应该怎样编写”这一事实感到非常苦恼。他说：“以前，我对自己编写的程序充满自信，但是对编写这些程序的方式就信心不足了。我总是比较随意地、偶然地编写出一些模块，然后测试，再编写一些，尝试让它们工作，然后在接下来的两星期内痛苦地进行调试，与bug作斗争。”

在Martin看来，XP除了自己先天具有的一切灵活性之外，它更是一种对规范（discipline）的增进。“XP对我来说意义非常重大，”他的语气里充满了热情。“其中有些规范对我们编写程序是极为重要的。测试驱动的开发就是我作为一个开发人员所经历过的最具革命性的东西。它给了我一种规范，一种我能引以为傲的时时可以遵循的规范。现在我编写程序的时候，我不仅对软件非常自豪，对开发的方式也同样骄傲。代码优秀，编写代码的手艺也很优秀。我知道自己将怎样继续，因为我有一套每一次都能产生好结果的规则。”

有听众说敏捷方法中有一种宗教般的态度，对此Martin的回答显示了他特有的实用主义：“请把精力放在实实在在的证据上。敏捷方法和XP都是可度量的，当人们是根据证据采纳它们的话，就不能说它是一种宗教。我再一次声明，我反对对任何东西宗教般不加分辨地采用。大胆尝试，小心求证，自己做出决定。”

整个发言字里行间都洋溢着三位战友炽热的个性和敏捷方法实践者独有的特点。他们在各自的方法学领域显然都已经浸淫多年，深厚的经验让与会者获益匪浅。

热门话题：海外外包

也许你不相信，这次大会真正的头号热门话题（当然不是正式的主题）并不是敏捷方法，而是海外外包（offshore outsourcing）。这些年，整体市场的低迷，国际上相对廉价的知识工人的竞争，和企业降低IT成本的需求，美国的IT从业人员早就开始担心自己的饭碗了。有了纺织业和钢铁业惨痛的前车之鉴，这个话题并不轻松。对于我们这些事件的另一方，角度虽然不同，但是这个话题也同样值得特别关注。知己知彼，方能百战不

殆，不是吗？

在主旨报告中，Watts Humphrey 就谈到："我老是会碰到一些企业，他们得意地说：'我们已经把一切都外包了。'这很好，但是不能解决所有问题。现在通过外包，可以用一半的薪水雇到和美国同样优秀的程序员，这确实是事实。海外程序员正在威胁你们的工作，正在挣美国的钱。"

"我最近去了一次印度，和 Tamil Nadu 大学的校长会谈。你猜得到吗？他们有 60 万理工科学生，每年毕业两万五千个软件专业人员——和整个美国差不多。在中国，他们已经翻译了我的6本书，而且在中国你能以印度一半的薪水雇到同样的程序员。但是我们自己本土的人还是有独特优势的：业务还是在这里，我们真正了解要解决的问题。我们能够多大程度上专业、高质量地完成工作，将决定我们是否能保住自己的饭碗。"

由于大约10年前在《Decline and Fall of the American Programmer》一书中曾经预测美国程序员将会灭绝，另一位会议主旨报告人 Ed Yourdon 自然也无法回避这一敏感话题（虽然 1996 年他又在《Rise and Resurrection of the American Programmer》中修改了自己的观点）。近来海外外包对美国软件研发界的冲击，许多人这才感叹 Yourdon 10 年前的预言真的是有先见之明。

Yourdon 的回答当然离不开自己的主题"死亡之旅"。他承认，自从《死亡之旅》第一版出版以来，整个经济环境已经发生了很大变化，但是死亡之旅类型的项目并没有减少。"当我1964年开始工作的时候，就存在死亡之旅类型的项目了。由于90年代dot-com泡沫这种项目成了普遍现象，而dot-com泡沫的破灭，非但没有缓解，反而加剧了这种境况。"

"现在我们的工作机会开始流向印度，"他说。"10 年、11 年前我们就预见到这种情况了，现在它真的出现在地平线上。解决方案？我看很难有放之四海而皆准的办法，也许要根据具体行业、角色来定。"

那么，开发人员对自己的工作是否担心呢？他们认为这种趋势应该通过什么办法解决呢？个人、集体、还是政府行为？为此，《Software Development》主编 Alexandra Weber Morales 与许多与会者就此进行了交谈。此外，9 月 17 日还在波士顿市政厅召开了一次小型讨论会，有15个人参加，但是没有达成什么共识。与会者中大部分都是公司的骨干，收入丰厚，公司还希望给他们更多培训机会，看起来似乎无需担心。有一位刚刚离开了原来的公司，发现就业市场确实衰退得很厉害。还有一个是一位经理，正在考虑把遗留系统的维护外包给一家顶级的印度厂商，使美国团队专门从事更高级的项目。

显然有些人还是很担心的，但是仇外心理似乎不用担心，还不存在。相对而言，与由于廉价劳动力而不振的行业（有人提到了纺织和钢铁）比较接近的那些人，相比目前与客户关系还非常融洽的人，所表达的担心更多。一位开发人员（他的父亲正在经受美国纺织业衰落的打击）说，当一家大公司解雇 5000 名员工时，这并没有消除工作机会，其实损失的可能是客户。而且他们所雇佣的新的劳动力可能买不起他们制造的产品。

如果海外外包时问题的话，政府干预是否是解决之道呢？对此大多数与会者都不同意，有些人指出，其实本地政府正是热心于 IT 人力资源海外外包的大雇主。有一位提到，新泽西州就通过了一项法律禁止对因外包而引起的失业提起诉讼。类似地，国家性的再培训和工作促进项目都和发放津贴一样，被认为会扼杀个人的主动性，因此同意者寥寥。

许多人都同意，说到底，强调职业道德、创造性、沟通和价值是使美国本土IT工作更加安全最有希望的保障。一位开发者说，经济衰退已经使中西部的人大大提高了生产率，整个行业也没有原来那些趾高气扬的傲气了。但是，海外程序员，尤其是印度的开发者往往要投入得多，这可能是因为文化，也可能是因为一种经济现象（就和第一代美国人也显示出比后代更多的雄心道理一样）。一位开发者说，由于竞争压力，优秀的客户服务和强调客户价值，就越来越重要了。这时候，会议发言人和极限编程培训师Joshua Kerievsky插话了："你在公司内部有没有宣传这个观点？管理高层是否知道你们有多优秀呢？我们必须让他们知道我们所做的一切事物有所值的——也就是说，要证明外包是一种有风险的不良投资行为。"无论如何，要与目前流行的"认为IT是一种日用品，到哪里买都行"的看法做斗争，将是长期的，需要美国本土开发人员做足准备。

同时在波士顿另一会场还召开了也是由CMP主办的"嵌入系统会议"。会场上嵌入式开发界大腕云集。详细情况，我们将在下期报道。

（本文素材由乐之搜集并翻译）

2002年图灵奖背后的故事（上）

迫近年关，想必大家都有些要松口气、歇一歇的念头吧。这个当口儿，本来讲点轻松的话题最好，谁料主编大人非要搞个安全主题，而且要求编辑部全体必须配合。配合就配合吧，人说一涉及安全难免都要严肃一点，我偏不信这邪。本月的选点既喜兴儿，又切题，连主编都不得不夸我啦。谁叫今年的图灵奖恰好颁给了密码学家呢？

众位看官想必都已知道，今年早些时日，美国计算机学会（ACM）将2002年度的计算机界最高荣誉阿兰·图灵奖授予了Ronald L. Rivest，Adi Shamir和Leonard M. Adleman，以表彰三人在公钥密码学（public key cryptography）方面的贡献。这个消息，我们在《软件研发》第一辑曾经报道过。大半年过去，此事好像“至今已觉不新鲜”了，是不是？且慢，此间故事，其中充满曲折，小史我很快就要化腐朽为神奇，博大家雷鸣般的掌声（要求也别太高了，有人的就捧个人场吧）。

对于今年的图灵奖，其实许多人是有异议的。因为在公钥密码学的发展史上，RSA并非理论原创者，甚至也不是第一个实现者。对此评审委员会主席Robert E. Kahn博士也解释说：“我们有很多非常有资格的候选人，但是他们（指RSA）的成就很好地结合了理论贡献和广泛的实际应用，这一点给我们留下了深刻印象。”

在我看来，RSA的数学解决方案的确高深，非常人所能想到，但是Diffie、Hellman和Merkle三人提出的公钥加密思想本身却更加富于创造性，属于Andrew Koenig所说的“能够取得令人瞩目的进展，但是事后看起来又是那样的明显，以至于让人不明白为什么当初会没有想到”（参见《软件研发》第一辑《for语句新探》一文）的那一类可遇不可求的灵感，和Tim Berners-Lee关于超文本链接、Douglas Engelbart关于鼠标的奇思妙想均异曲而同工，可以相提并论。想想看，从传说中公元前凯撒大帝发明移位密码开始，两千多年来，人们加密的方式都是通信双方采用对称的密钥，始终没有解决密钥发布和共享中的安全问题。而这一难题到了1975年，却被一位30岁出头、读高中没有毕业、读研究生没有拿到学位的本科毕业生解开了。这难道不够神奇吗？

史前史

Whitfield Diffie也许是上帝派来创造公钥密码学的人。10岁的时候，Diffie的小学老师曾经花了一天时间给孩子们讲解密码系统，让小Diffie深深地着了迷。他求爸爸到纽约城市学院把所有相关的图书都借了回来。 Diffie甚至尝试读Helen Gaines的经典名著《Cryptanalysis》，但是没有读懂。Diffie是跟着感觉走的人，他对密码学的热情没有持续太久，到了十几岁的时候，他的兴趣已经转到了军事上。有一阵子他甚至觉得密码学很俗，因为似乎什么人都开始对它感兴趣，而Diffie喜欢搞的是深奥的、别人不太懂的东西。

上大学时，他先进了加州大学伯克利分校，但是由于受到了当时左翼学生运动的影响，1964年又转到MIT，在那里获得了数学学士学位。

毕业后，Diffie来到Mitre公司，从事一个符号数学操纵系统的开发。他是数学系毕业的，对计算机本来一窍不通，这份工作将他培养为一名系统程序员，他开发了一个编译器，并且接触到程序正确性证明的概念。在此期间（1965年），Diffie听到了一种传言，说美国的国家安全局把自己办公楼里的所有电话都加了密。他信以为真，开始了思考：所有电话都加密，岂不是要有一个很大的多人共享的密钥中心？密码学的一个重要出发点就是不要相信其他人，只能信任加密系统。如果有这样一个密钥发布中心的话，人人不都得信任它吗？他继而想到，如果要保护整个北美的电话系统，又该怎么办呢？问题可以归结为两人通信的情形：两个人互不认识，但是需要交流秘密信息，按传统的方法他们必须事先交换其他人不知道的密钥。这里，条件和要

求是互悖的：如果能够事先交换密钥的话，何不直接交换秘密信息呢？而且在大型网络和组织中，特别是电子交易环境中，要求互不认识的人事先交换密钥，也是不现实的。这就是所谓的密钥发布问题，也是公钥密码学要解决的头号问题。

一个偶然的机会，他有幸遇到了John McCarthy（人工智能之父，Lisp语言的创造者，1999年图灵奖得主）。当时，McCarthy恐怕是惟一一个与Diffie对程序验证的重要性有同样认识的人。1972年，他来到斯坦福大学的人工智能实验室，在McCarthy手下从事程序证明和验证方面的研究。

但斯坦福对于Diffie并非福地，他的研究方法与众不同，因此日子并不好过。McCarthy是一位形式方法学家，精通形式逻辑。而Diffie对编程方法学更感兴趣。就在此时，改变Diffie命运的机会悄悄来到了。当时为Diffie的研究项目提供资助的是ARPA信息处理技术处，其负责人Larry Roberts希望对年轻的ARPANet（Internet的前身）进行加密保护，而在大型网络中众多密码的管理和发布就成了突出问题。他决定资助密码学方面的研究。当然，这个项目Roberts为什么会找到研究方向并不相关的McCarthy，就只有他们自己知道了。

Diffie根据自己当年在MIT对Multics中应用加密系统进行评估的经验，开始了思考：加密系统应该是内置于网络中的，运行速度应该很快，而且不会给用户带来不便。当时对一个文件加密的时间，一般要百倍于文件复制时间，因此根本没有人愿意在传输文件之前进行加密。所以Diffie决定从算法开始研究，主要关注其复杂性而非密码强度。他逐渐发现自己碰到了“big thing”（大家伙），对一项研究充满如此的激情，对他来说还从来没有过呢。他决心好好学习一下密码学，于是从1972年秋天开始研读David Kahn的书[1]，但是进展很慢。直到1973年春，Diffie在密码学方面还什么事儿都没干呢，更别提什么成果了。John McCarthy再也受不了他了，两个人终于闹翻，Diffie只好走人。

Whitfield Diffie现为Sun公司的副总裁，首席安全官，院士（Fellow）。1965年在MIT获得了数学学士学位。1991年加入Sun公司之前，曾在北方电信担任安全系统研究经理，设计了公司用于X.25分组网安全系统的密钥管理机制。整个90年代他都在研究密码学的公共政策，曾多次在美国国会作证。

Whitfield Diffie小档案

失去了工作的Diffie陷入了人生的最低谷，他开着一辆破车，开始周游全国，当然同时也继续思考密码学的问题，和每一个可能的人讨论。但是，当时密码学很大程度上还属于机密科学，感兴趣的人不多，做研究的人就更少了。真是老天注定，在颠沛流离之中他居然在新泽西遇到了一位志同道合的女子，同是天涯沦落人，两个人当时都是穷困潦倒，惺惺相惜，当然容易一见倾心，于是相约结伴游历大好河山去了。这位女子后来就成了Diffie的夫人。

公钥密码学的诞生

1974年两人来到了Diffie的母校MIT，在那里度过了整个夏天。一天，他们去拜访位于纽约Yorktown Heights的IBM研究中心，那里离MIT只有两个小时的车程。当时IBM的研究人员正在紧锣密鼓地设计一年后成为美国国家标准的DES加密标准，因此这里云集了大批此道中的高手，使Diffie大有如鱼得水的感觉。他遇到了一个曾与Horst Feistel共事的人，了解到Feistel对广泛应用于军事上的敌我辨别问题（公钥密码学另一个要解决的重要问题，也就是数学签名问题）的解决方案——质询—响应机制。在一场研讨会上，他遇到了自己的一位老熟人Bill Mann，在讲解单向函数（one-way function）的时候，Mann产生了误解，结果却阴差阳错地导出了陷门单向函数（trap one-way function）的概念——需要额外信息（即公钥）才能求出逆运算的函数。在MIT学术氛围熏陶下成长起来的Diffie马上意识到这一概念的重要性，他在笔记本上记录了下来。大家对此进行了讨论，都觉得这样的函数似乎很难构造——Diffie后来回忆说，其实从某种意义上讲，公钥密码学在此刻已经诞生了，只不过当事人自己都没有理解罢了。

Diffie还想找其他技术人员好好聊一聊，但是项目的技术负责人Allen Konheim拦住了他，对这位不速之客Konheim可以说是守口如瓶。然而枯谈中他无意地透露了一条信息，就是这条信息最终改变了历史：“原来曾经在这里做研究员的Martin Hellman几个月前曾经来过这里，现在回斯坦福去了。”Diffie笑了：“Martin Hellman？我认识他，他是斯坦福电机工程系1969届毕业的博士。”“那你去找他吧，三个臭皮匠，顶得上一个诸葛亮啊。”临别时，Konheim这样说。

Diffie很快回到了斯坦福，立即给Martin Hellman打电话。两人相见，晤谈甚欢，一直到深夜。彼此都大为感慨，深为自己找到了如此默契的合作伙伴而庆幸。甚至连两个人的妻子都找到了共同语言：狗。要知道，在此之前，Hellman自己也在寂寞中前行了很长一段时间，已出版的资料非常少，他的许多同事也对此都不感冒，认为这种研究天生就应该列为机密，出不了什么成果。从此，两人开始了长达5年的合作。Hellman为Diffie找到了一些资金，名义上他是比自己还年长一岁的Diffie的研究生导师，当然一直跟着感觉走的Diffie最终并没有拿到任何斯坦福的学位。凑巧的是，1975年初John McCarthy要去日本，因此找Diffie住到家里看房子

并照顾他年幼的女儿——要知道，Diffie夫妇此前一直住在车里，这下可是从地下来到了天上：McCarthy的家中居然有当时罕见的高速网络连接和很棒的工作站，而且位置与Hellman家只有200米——“真是天助我也！”在良好的环境下，两个人的合作研究进展很顺。

与此同时，一位加州大学伯克利分校的硕士研究生Ralph Merkle也在独立研究同样的问题，而且颇有心得。1974年秋天，他选修了著名计算机安全专家Lance Hoffman教授的课，这位教授要求每个学生都要交一篇论文，还要尽早提交开题报告。Merkle是一个喜欢挑战的人。他开始选择自己的论文题目了：传统的计算机安全主要通过权限和信道安全保护来实现，那么如果在一个开放的信道比如大型共享网络中，终端和连接都无法保密，两个用户能不能安全地进行通信呢？他本来是想证明不可能，却很快发现这是无法证明的，于是转而开始琢磨如何实现的问题了。几天后（多么让人难以置信！），他找到解决的思路（请注意公钥加密的第一个解决方案就这样诞生了），提交了自己的开题报告，Merkle提交的就是后来成为公钥密码学经典文献之一“Secure communication over insecure channels（保护不安全信道上的通信）”的雏形。在文中Merkle提出的思想可以简单地叙述成这样：如果通信的发送方A生成K个谜语（每个谜语生成需要花费1个单位工作量），然后通过开放的网络发送给接收方B，B从中随机抽取一个，解出来（花费K个单位工作量），那么双方就都以K个单位的工作量拥有了共同的信息——这个谜语。而对于第三方侦听者E而言，虽然他能够收到所有信息，但是需要花费K^2个单位的工作量才能。如果K的数目足够大，秘密信息就是很难破解的。当然，为了实用，论文中也给出了更复杂和详细的公式与方案。多年后，Diffie在评价Merkle的贡献时说：“Merkle的谜语系统是最早的实际解决方案，在某些信道上甚至仍具有实用价值，而且其安全性的论据极为清晰。直到现在我还是对此感到吃惊，这真是一个神奇的发现。”

遗憾的是报告教授没有看懂，他要求Merkle重写。不幸的是，虽然多次返工、退回，报告最后仍然没有通过——教授还是看不懂，Merkle只好放弃了这个学分。按Diffie的说法，Hoffman就这样错过了用“Hoffman-Merkle”冠名公钥密码学而名垂青史的机会。Merkle的思路本质上是设计一种公开密钥发布系统，由于需要耗费大量带宽（需要K足够大），实用性有问题，但思路是革命性的。在当时他的创新思想不仅Hoffman教授不懂，当时几乎没有其他人理解。他的论文从几页扩展到40多页，仍然被权威学术杂志《Communications of ACM》多次拒绝，审稿人（都是当时安全界的权威）的退稿理由五花八门，有的是典型的学阀口吻：“这篇论文不属于目前主流的密码学思路”，有的说与Hoffman的口气如出一辙：“思路不清楚，不可理解。”最气人的理由居然是：“没有引用任何参考文献”！Merkle的传奇之处，恰恰是他没有密码学背景，没有做过密码学研究，此前没有发表过论文，学术圈子里没有人认识他。而且，他的革命性思想是前无古人的，当然没有什么文献可以引用！不过他的论文确实写得不好，连Diffie也觉得难读。最终这篇论文是公钥密码学已经开始成为显学的1978年4月才得以发表的，这时候离Merkle最初的提交时间已经3年多了。

1974~1975年间，Diffie和Hellman每周都举办研讨班，讨论相关问题和解决方案。在一次讨论班上，有一个来自伯克利的学者讲解了Merkle的研究情况。Diffie当时认为这种做法是不可行的，但是却由此开始了更深层次的思考。Diffie后来承认，Merkle的研究对于最终发现公钥加密至关重要。

1975年5月的一天，Diffie终于灵光闪现，他捅破了那一层窗户纸！他想，一直想解决的两大问题：加密和签名，也许能够通过一种机制同时解决。只要能找到一个合适的带陷门的单向算法，就能为通信双方都生成两套密钥，一套用于加密，一套用于解密。加密的那一套可以是共享、公开的，解密的一套保密，但是由于单向算法的性质，从公开的秘钥中很难逆向破解私钥。而数字签名也能通过这种方式由甲方用公钥加密，乙方用私钥解密。这样不就既解决了加密问题，又解决了数字签名问题吗？

【下接第30页】

Ralph C. Merkle 和 Martin E. Hellman 小档案

Ralph C. Merkle 目前是佐治亚理工学院的计算学院教授。在加州大学伯克利分校获得学士和硕士，在斯坦福大学获得博士。曾在施乐PARC担任研究员长达12年。除了计算机安全以外，他近年的兴趣转向了纳米技术，曾获1998年费曼纳米理论奖。

Martin E. Hellman 是斯坦福大学电机工程系的荣休教授。1966年获得了纽约大学学士学位，1967年和1969年斯坦福大学硕士和博士学位（请注意他三年就拿下了博士学位！牛人就是牛人，不服不行），专业都是电机工程。1969~1971年在MIT任助理教授。此后返回斯坦福大学任教，直至1996年退休（51岁就退休了，也真够舒服的）。

专题导言

■ 特邀编辑 潘爱民

“Internet的美妙之处在于你和每个人都能互相连接，Internet的可怕之处在于每个人都能和你互相连接。”Internet正在日渐普及，但是安全问题也越来越突出了，近几年来发生的一系列网络安全事件足以让每一个连接到Internet的用户提心吊胆。

1 计算机网络安全的范围

10年前，大多数计算机都没有联网，那时候，如果拥有一台计算机，绝大多数情况下是可以放心地使用的，你不用太担心有人会在你工作的时候搞乱你的机器。

今天，如果机器没有联网，那么你还可以用这种方式来工作，但事实上，大多数机器都有了联网的机会，在网络上你根本不知道后台会发生什么事情，即使计算机专家也无法抵挡得住来自外界的强有力攻击。

计算机网络的安全范围由近至远可以分成多个层次：终端系统的安全、局域网络的安全、周边网络的安全和主干网络的安全。如图1所示。

从安全设防强度和成本投入的角度来看，离用户越远的地方，安全意识越好，相应的投入越大。网络安全是一个全范围工程，不是远处一道屏障就能万事大吉的。为了使终端用户能够正常地工作，仅仅依赖ISP提供的安全设施和安全服务是不够的，用户必须借助一些安全工具来及时发现问题，并且与ISP配合解决问题。在周边网络和骨干网络上通常有专职的安全管理员，他们利用各种检测工具以及专门的管理工具来实时监视网络，一旦发生安全事件，他们会以最快的速度响应。而且，他们能够通过各种途径获取到当前的安全状况，比如，最新的病毒信息、最新的攻击特征等。然而，终端系统和局域网络缺乏这样的专业人员来维护，用户需要工具的提示或者外界的提醒。

在现阶段，还没有一种一劳永逸的方案来保护终端系统或者局域网络。计算机系统不同于电话机，电话机可以整天接在电话网上而不用担心被人利用或者受到攻击，但是，计算机一旦联网就有可能成为攻击目标。所以，用户需要有安全意识，了解潜在的敌人会如何闯入。

图1：网络安全的范围

2 协议安全性

如果没有一种共同的语言，任何两方是无法进行通信的，在Internet上，TCP/IP协议就是这样的语言。由于TCP/IP是一个庞大的协议族，并且它在设计之初没有足够地考虑安全性，所以，针对TCP/IP协议出现了大量的攻击，下面是一些典型的协议缺陷：

- **IP地址伪造。**Internet上的通信主要是以端到端的方式进行的，但是中间需要经过很多路由器。路由器在路由一个数据包的时候，它检查包中的目标地址，从而决定下一跳的出接口。这就导致了源端可以假冒别人的地址给目标端发送数据包。攻击者在从事恶意活动的时候，往往会使用伪造的IP地址以便隐藏自己的IP地址。另外，TCP协议中的会话也有可能被恶意劫持。
- **数据传输的保密和可靠性问题。**数据包的传送是以明文形式进行投递的，TCP/IP协议本身并没有考虑任何保密措施。而且，协议只考虑了传输可靠性，没有考虑数据包被恶意篡改的可能。众多网络接口的混杂模式(promiscuous mode)也加剧了网络数据包的不安全性。
- **应用层协议的缺陷。**在TCP/IP基础上有各种应用层协议，最常见的有