2002下·合订版

programmer

CSDN

程序员

增值合订本

适合开发者 项目经理 CTO&CIO 编程爱好者阅读收藏

专题应用 .NET/Java/Open Source/Borland开发技术

浓缩展现 2002年《程序员》杂志全年所有文章精粹

最新集锦 developerWorks中国网站权威文章/教程/工具

真挚提供 全年杂志电子版/源代码/编程手册

赠 Delphi 7/DB2 8.1/WSAD 5.0

增值版 上册

合订版 下册

贺岁版 CD1

程序员版 CD2

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

www.csdn.net/magazine

程序员 增值合订本

2002 下 · 合订版

《程序员》总　编：黄长著
《程序员》社　长：张悦校
《程序员》副 社 长：蒋　涛
《程序员》执行副总编：熊池舟
《程序员》副 总 编：唐　琦
《程序员》技术总监：曾登高
《程序员》美术总监：关峻峰
《程序员》责任编辑：孟迎霞　汤　韬　熊　节　行　舟
张　里　闫　辉　孔祥利　吴志民
程　琰　俞　波　武　超

電子工業出版社
Publishing House of Electronics Industry
北京.BEIJING

京海工商广临字 2002053 号

Programmer 程序员 增值合订本（2002下 · 合订版）

图书在版编目（CIP）数据

程序员增值合订本（2002下 · 合订版）/《程序员》杂志社编. —北京：电子工业出版社，2003.1

ISBN 7-5053-8300-0

Ⅰ.程… Ⅱ.程… Ⅲ.程序设计—文集 Ⅳ.TP311.1-53
中国版本图书馆 CIP 数据核字（2002）第 098484 号

责任编辑：郭立
印　刷：北京印刷三厂
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编100036
经　销：各地新华书店
开　本：850×1168　大1/16　印张：26　字数：800千字　附光盘2张
版　次：2003年1月第1版　2003年1月第1次印刷
印　数：80 000册　　定价：39.00元（上、下册+2CD）

读者订购或发现装订错误、缺损、破损，请与《程序员》读者服务部联系。
联系地址：北京市朝阳区北三环东路静安中心26层（100028）
电话：010-84480948/84540231-33转279
传真：010-84540263

程序员增值合订本

内容简介

《程序员增值合订本》全套共分增值版、合订版两本图书，贺岁版、程序员版两张光盘。

- 增 值 版 （上册）：专题应用.NET、Java、Open Source、Borland开发技术
- 合 订 版 （下册）：浓缩展现2002年《程序员》杂志全年所有文章精粹
- 贺 岁 版 （CD-1）：最新集锦 developerWorks中国网站权威文章、IBM最新开发工具
- 程序员版 （CD-2）：真挚提供 全年杂志电子版 源代码 编程手册
- 特 别 附 赠：Delphi 7 DB2 8.1 WSAD 5.0

上册为“增值版”图书。从当前软件开发人员的需要出发，针对2002年最热门和使用最广泛的技术，精心整理了这套增值版技术图书，大部分内容都是在国内首次发表。

本书内容主要分为如下四大版块：

.NET版块 2002年最令软件开发者重视的无疑是.NET技术，.NET书籍目前已不少，大部分.NET的使用者已经理解了.NET的基础知识，但真正应用的还不多。

.NET版块得到了微软中国公司技术专家的大力帮助，共分七大专题二十余篇文章，分别为蔡学镛C#讲座、FAQ（常见问题解答）、Web Services（MSDN专栏“At Your Services”中的精华文章）、Visual Studio .NET（Visual Studio .NET的高级特性介绍）、转向.NET的全面指南、Security和Case Study（如何用.NET技术开发真正的软件系统真实案例）专题。

Borland版块 Borland在国内最受欢迎的开发平台无疑是Delphi，内容共分为四大专题，即Delphi 7、Delphi高级应用、数据库开发以及IDE编程高级指南。全部文章都由资深专家撰写，具有很高的实用价值。

Java版块 Java现在已经真正成为主流的开发语言，该版块共分六个专题，即：深入浅出谈Java的蔡学镛专栏，Java手机开发的王森专栏；跨平台应用的典范JAX专题；详细论述Java垃圾收集、对象检查等JVM虚拟机底层内容的JVM专题；Persist JDO专题；以及Threading专题。

Open Source版块： 开发源代码运动的浪潮一浪高过一浪，它带给软件开发者的影响深远而又巨大的，这次特地推出的Open Source版块，内容主要分为三大专题：wxWindows（非常出色的跨平台GUI开发框架）、MemoryDB（内存数据库系统）以及Apache PHP。

下册为“合订版”

本书浓缩了2002年《程序员》杂志的精华，全书共收录12期杂志中数百篇精彩文章，内容覆盖人物&报道、管理、技术及服务四大版块，共分十五个专栏，分别为名人堂、创件创业、CTO论道、名家专栏、技术专题、Python专栏、名家访谈、MSDN专栏、技术讲座、开发心得、源码分析、开发工具、编程擂台、书评和好书推荐。此外，书后还附有全年12期杂志的目录索引。

全年12期杂志中最受读者欢迎的技术文章、最具指导性的管理、技术工具应用分析、最值得肯定的编程类图书评论，您都可以在这里找到。

CD-1为贺岁版

本光盘内容包括developerWorks中国网站2002文章集粹、IBM最新开发工具（强大的通用数据库DB2 8.1版、还有集成Java、Web和企业应用于一体的开发环境WebSphere Studio Application Developer 5.0版）。这两款工具标志着IBM在开发工具市场从大企业延伸到中小企业，并保持着大企业的高性能。

CD-2为程序员版

本光盘内容涵盖2002年程序员杂志的全部内容，另附送Borland Delphi 7完整试用版、用友华表Cell 5.0组件及Excel读写控件特别版等开发工具。此外，重点推出的“程序员编程手册”，包含了Java、VB、VC、Delphi、C++、SQL等开发工具的API、库函数、编码规范等，特别适合于开发者们学习、查阅和收藏。

希望读者能从这些精品文章和工具中得到真正有用的知识，学以致用，新年更上新台阶！

栏目	文章名	页码

《developerWorks 2003 预岁版》目录

- developerWorks中国网站2002文章集萃
- 教程
- 工具包
- WebSphere Studio Application Developer 5.0Beta版
- IBM软件全球专业认证

2002年《程序员增值合订本》配套光盘目录

- 2002全年12期《程序员》杂志电子版
- 《编程手册》包含Java、VB、VC、Delphi、C++等开发工具的API、库函数、编码规范等
- 2002全年《程序员》杂志文章配套源码
- 2002《程序员增值合订本》文章配套源码
- IBM DB2通用数据库 8.1Beta版
- 用友华表Cell 5.0组件及Excel读写控件特别版
- Delphi 7完整试用版
- 瑞星杀毒软件全功能查毒版

PHP 发明人 Rasmus Lerdorf

1995 年，Rasmus Lerdorf 开始研究一种新的语言，这种语言后来被命名为 PHP。当时他心里惟一的目标就是找出他正在浪费他的精力，作为一个独立软件承包人，Lerdorf 构建在雇主发送了许多包含 URL 介绍与浏览次数，为了给自己的网站建立访问日志，他为自己的页面创建了把标记嵌入在 HTML 代码中的一段 Perl CGI 脚本，来收集访问用户的信息，为了给雇主编报印象，他开始允许任何访问者查看他的日志。

因为日志是在个人主页中使用，他把这个日志代码称为“面向个人主页的 PHP 工具”。很快，就有客户向他询问如何得到这个工具，于是他打算把这个工具进行发布。“那真是这个软件的奇迹：你可以发布它并保有它。”Lerdorf 一语双关地回忆道。那时，开放源码运动还未出现，“现在，这叫做免费软件了。”在 1995 年晚期，Lerdorf 为共享软件的概念、bug 程序修补和代码改进建立起第一个 PHP 邮件列表。

后来，Lerdorf 和多位多大学签订协议，为方便学生上网建立起一个拨号系统，该系统要求实现一个基于网站管理的接口，它能访问存储于一台 IBM 大型机上的大学图书馆的学生数据库，图书馆管理员将根据学生们上网账号上打出的预付款给予其相应的访问时间，并且访问时间的信息也将在数据库中随时更新。

当时还没有将用户界面接入网页和数据库的工具，Lerdorf 为嵌入式 HTML 文本建立了许多标记，用 C 封装器换下 Perl 代码。因为这些标记可以将数据录入表格，并将数据转换成符号变量方便其他系统输出，所以他将其称之为“表单注释器”。

在 PHP 工具包里集成表单注释器，这就是 1996 年 Lerdorf 开发出的 PHP 第二个版本，它被改名为 PHP—FI 后，Lerdorf 确发了开发商业产品的设想。但那时，他得到大量来自其他程序员的代码改进和补丁修正。

Lerdorf 最初编写接口代码用的是 MySQL，但其他程序员做的鲜外部具有与 Oracle 和 Sybase 数据库的关联性，他意识到他应该很好地保持代码开放，并继续得到所有这些免费的贡献，这样可以增加 PHP 的产品化和知名度。

Lerdorf 开发代码的逻辑是：“它必须对我有利。如果全世界都使用我写的代码，那么最终我将从中得到好处。”（他在 Linuxcare 的新工作证明了其预言的正确性）。随着 Apache 网站服务器的日益普及，开放源代码的观念赢得了赞扬。这是 Lerdorf 第一次正式开放源代码许可，他也意识到这比较适合于 PHP—FI 代码，“如果你没有出售它，你例不知公开它，”他说道。和所有的 PHP 开发者一样，Lerdorf 相信开放源代码是创建代码最明智的方式。“这是全新编码方式的开始”，他说，“这是当今发展的趋势。”

Lerdorf 认为，当今仍然私有化代码的唯一大型软件公司就是微软，其他诸如 Sun 和 Oracle 这样的公司都越来越开放其源代码了。他说：“这将会成为微软和开放源码之间的斗争。”他继续说道：“即使微软垄断为一个项目投入 50 名程序员，但开放源码却可以投入 5000 甚至 50000 名程序员。”

Lerdorf 继续潜心于 PHP，并一直是最核心的开发人员，虽然代码多次被修改，脚本引擎被重写得面目全非，为了调用 C 代码，Lerdorf 第一个在 HTML 文本中创建标记，这种观念已经演变成为创建动态网站应用程序的方式。另外，Lerdorf 还是 ActiveState 顾问团、GreatBridge 顾问团、Apache HTTPD Server Project 核心小组的成员，他还是世界性 PHP、Apache 和开放源代码项目等组织的发言人。

Rasmus Lerdorf 出生在格陵兰岛的 Godhavn，在丹麦渡过了大部分童年时光，后来，他一直和家人生活在加拿大。在作为高级开放源代码研究员加盟 Linuxcare 公司后，他和妻子 Christine 从多伦多搬到了现在的居住地——美国旧金山。（2002.01）

Thiuking iu Bruce Eckel

从 1986 年至今，Bruce Eckel 已经发表了超过 150 篇计算机技术文章，出版了 6 本书（其中 4 本是关于 C++ 的），并且在全世界做了数百次演讲。他是《Thinking in Java》、《Thinking in C++》、《C++ Inside & Out》、《Using C++》和《Thinking in Patterns》的作者，同时还是《Black Belt C++》文集的编辑。他的《Thinking in C++》一书在 1995 年被评为“最佳软件开发图书”，书自面世以来，已经被翻译成多种语言，Bruce 被称为“这个行业的明星”。

最近 Bruce Eckel 的《Thinking in Java（第二版）》又一次赢得了 JavaWorld 的“编辑首选图书奖”。欣喜之余，他谈起了自己的故事：

“在我七五岁那年，我到父亲的建筑工地上，看到他手那些又脏又累的活，那时我就暗下决心：决不让别人控制我的生活，决不让自己也去干这种脏活累活。所以，我选择了编程。”

“花两年的时候，我第一次接触到了计算机，那其实是一台 ASR 33 电传打字机，用磁鼓和穿孔纸带保存程序。它非常简陋，但却让我着迷。我和我的朋友们用它来编写简单的游戏。后来，在大学里我开始真正使用计算机，一开始，我只是为了研究应用物理而学习了一些编程技术，而到最后，我却拿到了计算机工程专业的博士学位。”

“毕业的时候，我比较精通计算机硬件，对编程则不那么重视，编程的工具也上要是汇编语言。我的第一份工作就是嵌入式系统开发。但在那时，我读了关于 Jack Purdum 的书，然后配深的被 C 语言吸引住了。对硬件的了解让我很快掌握了 C 语言的精髓，但是要说服人们使用高级语言来开发嵌入式系统实在太困难了：‘难道汇编不行了吗？它不是一直都很好用吗？’从那时起，我就开始努力让人们接受更具生产力的工具。C++、Java 乃至现在的 Python，莫不如此。”

“我喜欢用‘Thinking in ……’来给我的书命名。如果你想真正掌握一门外语，你就必须用它来思考，编程技术也是如此。如果你能做到‘Think in’某一种编程语言，那么它对你来说也就不再陌生了。现在我的书比较受欢迎，原因大概有四个：我一直按照自己的节奏来写书，这是第一个原因。第二个原因就是我经常主动讨论会，根据与会者的反馈不断修改我的书。第三个原因，也是最重要的原因：我在 Internet 上公开自己的书，这让我结识了许多非常专业的读者，得到许多非常精彩的反馈信息。第四个原因则是我自己编写的代码提取工具，它能帮助我检查书中代码的正确性。”

“我也不是任何时候都在工作的，实际上，我经常做些其他的事情。比如说，我在去年二月的时候到新西兰南岛骑车旅行了三天，非常好玩。现在我们又在筹划另一次旅行。我们打算每天的时候到欧洲去，在有空的时候，我经常从事户外运动，走步或者骑车旅行，甚至在开会期间，我都会带与会者在下午出去骑车，而且我还对艺术很感兴趣，包括摄影、雕塑和绘画等等。”《Thinking in C++》的封面就是我自己画的。”

Bruce Eckel 有一个用自己的名字注册的网站：http://www.bruceeckel.com。（2002.02）

Guido van Rossum

从本期开始，我们将陆续推出一系列关于Python的文章。在认识这种语言之前，请先来认识一下它的发明人：Guido van Rossum。

我的名字经常给美国人带来些麻烦。

发音：在荷兰语中，"Guido"中"G"的发音类似于英语"Jack"中的"ck"，也就是说，我的名字应该是"奎多"。但如果你是美国人，你也可以把它读成意大利语的"归多"。

拼写：我的姓有两个单词，我也希望别人都能明白这一点，在信用卡上也是这样写的。按照荷兰语的拼写规则，在写全名的时候，"van"是小写的："Guido van Rossum"，但是如果只提到我的姓，就应该大写："Van Rossum 先生"。

字母顺序：在美国，我的名字被分在字母"V"里面。但是在欧洲，我的名字就分在了"R"下面。而在我的一些朋友的电话本里，我的名字又到了"G"下面……

作为Python的创造者，我很愿意谈谈它的来源，以及在创造Python过程中的一点点个人看法。

到今天，我可以说：Python已经彻底改变了我的生活，它带我找到了新大陆。在工作的时候，我用Python来开发大型项目；有空的时候我就继续研究Python，或者回复与Python相关的电子邮件。我已经看到了Python的工作室、邮件列表、新闻组，现在又有了一本书。坦白地说，这已经让我非常满足了。不过，在我开始做白日梦之前，我们先来看看Python发展中的一些趣闻。

一切都是从ABC开始的，那是我在80年代早期参与创造的一种有趣的教学语言。它的优雅和强大令人难以置信，而且完全面向非专业的程序员。尽管它优雅、强大而且完全自由，但是ABC还是从来没有在Unix C的世界中流行过。我推测，原因可能是ABC的扩展太困难了。它是一个"单片机"，一个封闭的系统，只有最基本的输入／输出操作：从控制台读一个字符串、写一个字符串到控制台。我决定，绝不在Python中重复这个错误。

用Python写的程序有着相当好的可读性，这不是偶然的。作为一种面向对象的语言，Python鼓励程序员写出可复用的代码。但是，就算我们把文档写得再完美，没有可读性的代码也是很难复用的。Python的许多特性共同作用，使得Python的代码具有很高的可读性。

可读性的增强通常伴随着可变性的减弱。在Python中，对应每个常见的问题都有一种很简单、很显而易见的解决方法，这就减少了程序员面临的选择，提高了下一个程序员读懂这些代码的可能性。当然，Python的可读性还应该归功于那些最保守、最传统的符号格式。绝大多数的符号都是一目了然的，没有那些希奇古怪的符号——比如"@&$"什么的。

我承认Python并不是运行得最快的脚本语言，但是它很稳定，而且可以节约程序员编码和调试的时间。当然，这方面的节约才是真正的节约。尽管很难做准确的度量，但是我们认为Python的确帮助程序员节约了大量的开发和调试时间。

不管是好是坏，Python都应该算是我的个人作品。但是它的成功完全要归功于广大的用户。他们在网上传播Python，他们在各种环境下用Python开发软件，他们在邮件列表里讨论Python的问题，他们把各种想法都反馈给我，正是由于有了这些用户，Python才能不断发展。

Guido van Rossum和他的Python刚刚赢得了自由软件基金会的年度大奖。他最推崇的两句话是"笨拙并以此为傲"和"在互联网上没有谁知道你是一条狗"。他本人的主页是http://www.python.org/~guido，如果你有什么话想告诉他，也可以写信到guido@python.org去。(2002.03)

软件之母——Ada Byron

多年前，美国军方要给一种计算机语言取个名字，大家提了许多动听的名字都觉得不太中意，后来有人提议，将这种计算机语言命名为：Ada。没有人提出异议。为什么大家都同意用这个名字呢？这还得从100多年前说起。

1815年12月10日，英国，一个女孩降生了。她的母亲，一个有着杰出数学天赋的女人，人称"平行四边形公主"；父亲，一个狂热的充满幻想的诗人——拜伦。可是女孩从生下来就从没见过父亲。这位狂热的、浪漫的诗人结婚不久便离开了英国，再也没有回来。母亲给她起了个动听的名字：阿达（Ada）。那位诗人尽管离开了英国，当听说自己有了一个女儿时非常高兴，可也非常懊悔，但他始终没有回去看女儿一眼，只能用诗来表达自己对女儿的思念和歉疚，他的一首诗的名字就叫《阿达》。

1834年11月，阿达在一次宴会上遇到了一位对其一生产生重要影响的人——查尔斯·巴贝奇。此时的巴贝奇正到处游说他的计算机设想。当时的人们很少理会他，以为他是在"痴人说梦"。同样，他也将自己的设想全盘托出，讲给阿达听。此时的阿达只有18岁，但她听完他的设想并看了他的文稿后，彻底领会了他的设想，并深深地为之陶醉。凭着她深厚的科学功底和丰富的想像力，她认为这是一个伟大的设想，世界将因之而改变。

共同的追求，使两人成了忘年交。阿达的母亲曾试图阻止阿达与巴贝奇的交往，认为巴贝奇不过是个江湖骗子，不会给阿达好影响。这点阻力对于秉承父亲性格的阿达，算不了什么，她完全投入到了计算机的研制中去了，负责为巴贝奇设想中的计算机编写软件。

1841年，巴贝奇在意大利都灵向人们详细地介绍他的设想，希望能引起大家的重视，但无人喝彩。他用法语出版的论文也不受人欢迎，但阿达执意要将其翻译成英文。翻译结束后阿达将文稿给巴贝奇看，巴贝奇发现：阿达不仅在论文中加入她特有的想像，而且补充了许多阿达独到的见解。阿达特别强调存储程序和数据的重要性，而这与今天的计算机技术不谋而合，并且拟订了一份设计图，这份设计图被公认为世界上第一个计算机程序。阿达在文中对计算机应用前景的展望，连巴贝奇自己都从来没有想到过。如：阿达认为，计算机应该发展成一个可用符号来表示任何事物的装置，这不正是今天的编程语言吗？她还预见到计算机可以用在纺织机械上，用卡片存储复杂的花样，可以用来绘图、演奏音乐。这些预言表明阿达是现代人工智能技术的拓荒者。

阿达对论文的修改，使巴贝奇深感惊奇和鼓舞，他对别人称赞说："阿达是个充满想像力和洞察力的女孩"，"她是个数字女神"。她将诗歌的激情融入了论文之中。

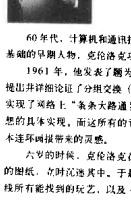
经过阿达翻译后的文稿其内容增加到原来的一倍，论文实际上成了两人合作的产物，但遗憾的阿达在著名的时候，只将自己的名字简单地署为：A.A.L。

在后来的一系列论文中，阿达在计算机软件领域做出了许多开创性的贡献：如变量、递归、程序算法的提出等。

阿达后来与威廉伯爵结婚，婚后生有三个孩子，但为了研制计算机，她将孩子们都放到母亲那里抚养，这对于一个女性，在当时是不被人理解的，好在丈夫非常支持她的研究工作，这使她深感欣慰。此时的巴贝奇已是一贫如洗，阿达也付出了许多。长期的研究耗费了大量的心血，身体状况也一天不如一天，疾病时时纠缠着她，但为了那个美丽的幻想成真，她孜以继日地工作。她的座右铭是：工作是我的报酬。

1852年，阿达因病症去世，同她的父亲一样，年仅36岁，没有等到计算机的诞生。如果她再多活一年，就会看到在福典，由乔治和爱娜华根据巴贝奇的方案制造出的一台差分机，这不能不说是人类的一个遗憾：第一位软件工程师，却没有看到自己的设想结出的果实。

阿达、巴贝奇两人对计算机事业的贡献就好像火对于人类，他们是钻磁取火的人，是盗得火种的普罗米修斯，让我们记住巴贝奇，也记住这位杰出的女性——阿达（Ada），用她的名字给一种计算机语言命名，以能告慰我们对她的纪念和怀念，却还远不能表达她为计算机技术作出的重要贡献。（2002.04）



构筑互联网络血肉之躯的克伦洛克

60年代，计算机和通讯技术基础研究如火如荼。而作为最定Internet基础的早期人物，克伦洛克功不可没。

1961年，他发表了《大型通讯网中的信息传送》的论文，首次提出并详细论证了分组交换（Packet-Switching）的理论设想，这一设想实现了网络上"条条大路通罗马"的信息传递。后来，他毛持了这一设想的具体实现，而这所有的贡献，他认为要完全归功于六岁时读到的一本连环画提带来的灵感。

六岁的时候，克伦洛克在一本连环画报里找到了制作晶体管收音机的图纸，立时沉迷其中。于是，他便用父亲的剃刀、手纸、铅笔头、电线所有能找到的玩艺，以及一个不知从哪里弄到的公共电话亭的话筒，开始了他的研制工作。但可爱电容却无论如何不能自己做，于是他瞒着母亲带他到离家很远的电子器材商场。六岁的他毫不怯场，走上前去对售货员说："我要买一个可变电容。"售货员连听的看着眼前这个还没有柜台高、满脸稚气的孩子，禁不住逗他："你要什么型号的？"这一问可把他难住了：小小年纪只是凭着一股好奇，哪知道什么型号啊？他将自己的设想描述给售货员，而后者竟然听懂了，并立刻卖给他一个可变电容。最终，他的设计成功了！优美的音乐从话筒中传出，一个小小电子工程师诞生了！

在麻省理工学院，他发现绝大多数同学都致力于一些相对容易的研究工作，于是他选择了既富挑战性、又能满足他对电子技术痴迷的，醉为人知的计算机网络研究，并最终取得了突出成就。在研究中他确立了许多重要的基本原理，特别是包交换技术的提出与完善，奠定了今天国际互连网络的理论基础，为其后的研究确立了正确的方向。他本人也成为计算机网络研究的权威，被后人誉为Internet之父。克伦洛克的许多重要贡献都记载在麻省理工学院1964年出版的《通讯网络》一书中。这已成为该院今天引以为豪的事迹。

ARPANET的详细规划完成于1968年，1969年1月BBN（Bolt, Beranek and Newman）公司赢得了完成这一计划的合同。此时距克伦洛克提出这一理论已经过去了将近十年。ARPA决定由他领导一个小组，将加利福尼亚大学洛杉矶分校作为连入ARPANET的第一个节点。一个月之后，能们又在斯坦福大学建立了另外一个节点，这是第一个HOST-HOST（主机到主机）的通讯系统，在上月初，他们进行了从一个节点肇录到另一个节点的试验。

这一天，众多关心这一计划的人们亲临现场等待他们的成功。各大计算机协会人员、美国政府官员、美国军方代表、大学院校代表、记者、ARPA的代表……

克伦洛克和他的助手们输入："LOGIN"，两边的程序就都同时戴上耳机，随时联系他们计算机的通讯情况。在洛杉矶分校那端，他们输入了一个"L"，然后问对方收到什么，对方回答："GOT THE L（收到了L）。"然后这边输入一个"O"，对方回答"收到了O。"通讯成功了！双方都怀抑不住兴奋的心情，狂呼起来，这是智慧与毅力的结晶。洛杉矶分校那边又输入一个"G"，可这次，系统崩溃了，他们又进行了第二次试验，结果好极了！

这些创业者们相信：他们的网络系统将战胜所有的单机系统。很快四个节点便全部完工，1970年又有了10个节点，再后来，网络发展的迅猛让每个人都不再怀疑它的威力，特别是80年代商业运作的介入，其对人类的影响更是一发而不可收。

ARPANET的缔造者克伦洛克在构筑了网络通讯的"血肉之躯"后，又提出了当时更超前的设想：让移动用户使用网络共享信息。他开始了构筑"无形"网络的研究。时至今日，"移动"二字快成时尚。

今天，当我们享用这快捷便利、无所不在的国际互联网时，请不要忘记领导缔造了第一个节点的克伦洛克，六岁的小小电子工程师，Internet之父。（2002.05）

阿兰·图灵

阿兰·图灵16岁开始研究爱因斯坦的相对论。

1931年，他进入剑桥大学研究量子力学、概率论和逻辑学，这逐渐学是由剑桥大学的怀特海和罗素创立的数理逻辑。德国大数学家大卫·希尔伯特（D. Hilbert）在此基础上，于1928年提出著名的"希尔伯特纲领"，认为《数学原理》所定义的系统既是一致的，也是完备的。换言之，任何系统的完备和一致性，可以由系统本身证明证明。1931年，"希尔伯特纲领"被奥地利逻辑学家库尔特·哥德尔（K. Godel）驳出一个大窟窿，他认为没有一种公理系统可以导出数论中所有的真实命题。除非这种系统本身就有悖论，这就是著名的"哥德尔定理"。它不仅推翻"希尔伯特纲领"，还直逼《数学原理》，说它本身就是不一致的，真是"城门失火，殃及池鱼"。一下子，整个逻辑和数学的天空中阴云四布。

在这种背景下，天才的图灵在数理逻辑大本营的剑桥大学提出一个设想：能否有这样一台机器，通过某种一般的机械步骤，在原则上能一

个接一个地解决所有数学问题。1936 年图灵发表一篇著名的论文《论数字计算在判决难题中的应用》。他提出了一种十分简单但运算能力极强的理想计算装置，用它来计算所有能想象得到的可计算函数。它由一个控制器和一根假设两端无界的工作带组成。工作带起着存储器的作用，它被划分为大小相同的方格，每一格上可书写一个给定字母表上的符号。控制器可以在带上左右移动、控制带有一个读写头，读写头可以读出控制器该时格子上的符号，也能改写和抹去这一符号。这一装置以后一种理想的计算模型，或者说是一种理想中的计算机。这就是电脑史上与“冯·诺依曼机器”齐名的“图灵机”。

在这篇开创性的论文中，图灵给“可计算性”下了一个严格的数学定义，并提出著名的“图灵机”(Turing Machine)的设想。“图灵机”不是一种具体的机器，而是一种思想模型，它由三部分组成：一条带子，一读写头和一个控制装置，能计算出任何给定的计算，也能执行任何可能的任务。图灵的这一思想实际上奠定了现代计算机的基础。电脑事实上就是用相应的程序来完成任何设定好的任务。同年，图灵赴美，在普林斯顿高等研究院进修，遇见了冯·诺依曼，后者对他的论文由与赞赏。冯·诺依曼后来一再强调，他的“存储程序”的思想主要来自图灵。正如飞机的真正成功得力于空气动力学一样，计算机由模拟计算向数字计算的飞跃中，图灵的理论起了至关重要的作用。

1947 年图灵写了一份内部报告，提出了“自动程序”的概念，图灵的这份报告后来收入爱丁堡大学编的《机器智能》论文集中，至今人们犹能的是，图灵早在 1945 年就提出“仿真系统”的思想，并有一份详细的报告，想建造一台没有固定指令系统的电脑，它能够模拟其他不同指令系统的电脑功能。但这份报告直到 1972 年才公布。这说明图灵在“战结束后就开始了后来被称为“人工智能”领域的探索。也开始关注人的神经网络和电脑计算之间的关联。1950 年，图灵发表了里程碑式的论文《电脑能思考吗》，第一次提出“机器思维”的概念，图灵通条反驳了机器不能思维的论调，作出了乐观的回答，图灵提出一假想：一个人在不知情的条件下，通过一种特殊的方式和一台机器进行问答。如果在相当长时间内，他分辨不出与他交流的对象是人还是机器，那么，这台机器就可以认为是能思维的。这就是著名的“图灵测试”(Turing Testing)，当时全世界只有几台电脑，它们肯定无法通过这一测试，但图灵预言，在本世纪末，一定会有电脑通过“图灵测试”，电脑能做我们想象不到的事情，时至今日，图灵的人才预言终于在 IBM 的“深蓝”上得到印证实现。

在 42 年的短暂生涯中，图灵在量子力学、数理逻辑、生物学、化学方面都有深入的研究。在晚年还开创了一门新学科——非线性力学。当然他最为的成就还是在电脑和人工智能方面。是这一领域开天辟地的人物。缘此，后来人们把计算机领域的最高奖以他的名字命名。“图灵奖”又叫“图灵大奖”，是世界电脑界的诺贝尔奖，迄今已约获得人 Richard W. Hamming，结构化程序设计的创始人 Edsger W. Dijkstra，排版工具 TEX 的创造者 Donald E. Knuth，数据库管理系统理论的缔造者 Edgar F. Codd，MODULA和PASCAL等编程语言的创造者 Niklaus Wirth 等等计算机科学界的大师都曾经获得过这项荣誉。

图灵的才气纵横天得经，横跨剑桥和普林斯顿两校，上联罗素、怀特海，维特根斯坦、哥德尔，下接冯·诺依曼、是他用电脑、人工智能在现代数理逻辑和现实世界间搭起了一座不朽的桥梁。(2002.06)

关系数据库之父——埃德加·考特

在数据库技术发展的历史上，1970 年是发生伟大转折的一年。这一年的 6 月，IBM 圣约瑟研究实验室的高级研究员埃德加·考特(Edgar Frank Codd)在 Communications of ACM 上发表了《大型共享数据库数据的关系模型》一文。ACM 后来在 1983 年把这篇论文列为从 1958 年以来的 25 年中最具里程碑意义的 25 篇论文之一，因为它首次明确而清晰地为数据库系统提出了一种崭新的模型，即关系模型。

“关系”(relation)是数学中的一个基本概念，由集合中的任意元素所组成的若干有序同构关系，用以反映客观事物间的一定关系，如数之间的大小关系、人之间的亲属关系、商品流通中的购销关系等等。在自然界和社会中，关系无处不在：在计算机科学中，关系的概念也具有十分重要的意义。计算机的逻辑设计、编译程序设计、算法分析与程序结构、信息检索等，都应用了关系的概念。而用关系的概念来建立数据模型，用以描述、设计与操纵数据库，考特是第一人。

由于关系模型既简单、又有坚实的数学基础，所以一经提出，立即引起学术界和产业界的广泛重视，从理论与实践两方面对数据库技术产生了强烈的冲击。在关系模型提出之后，以前的基于层次模型和网状模型的数据库产品很快走向衰败以至消亡。一大批商品化关系数据库系统很快被开发出来并迅速占领了市场。其交替速度之快、除旧布新之迅疾是软件史上所罕见的。基于 70 年代后期到 80 年代初期这一十分引人注目的现象，1981 年的图灵奖很自然地授予了这位“关系数据库之父”。在授受图灵奖时，他被了题为“关系数据库：提高生产率的实际基础”的演说。(刊于 1982 年 2 月的 Communications of ACM 第 109 至第 117 页，或见《ACM 图灵奖演说集》第 391 至第 410 页。)

考特原是英国人，1923 年 8 月 19 日生于英格兰中部的港口城市波特特兰。第二次世界大战爆发以后，年轻的考特应征入伍在皇家空军服役。1942 年 1945 年期间任机长，参与了许多重大空战。为反法西斯战争立下了汗马功劳。二战结束以后，考特上牛津大学学习数学，于 1948 年取得学士学位以后到美国谋求发展，他先后在美国和加拿大工作，参加了 IBM 第一台科学计算机 701 以及第一台大型晶体管计算机 STRETCH 的逻辑设计。主持了第一个有多道程序设计能力的操作系统的开发。他自觉缺件知识缺乏，于是在 60 年代初，到密歇根大学进修计算机与通信专业(当时他已年近 40)，并于 1963 年获得硕士学位，1965 年取得博士学位。这使他的理论基础更加扎实，专业知识更加丰富。加上他在此之前十几年实践经验的积累，终于在 1970 年迸发出智慧的闪光，为数据库技术开辟了一个新时代。

由于数据库是计算机各种应用的基础，所以关系模型的提出不仅为数据库技术的发展奠定了基础，同时也成为促进计算机普及应用的极大推动力。考特特提出关系模型以后，IBM 投巨资开展关系数据库管理系统的研究，其“System R”项目的研究成果极大地推动了关系数据库技术的发展。在此基础上推出的 DB2 和 SQL 等产品成为 IBM 的主流产品。System R 本身作为原型并未向世，但鉴于其影响，ACM 还是把 1988 年的“软件系统奖”授予了 System R 开发小组(获奖的 6 个人中就包括 1998 年图灵奖得主 J. Gray)。这一年的软件系统奖还破例同时授给两个软件，另一个得奖软件也是关系数据库管理系统，即著名的 INGRES。

1970 年以后，考特继续致力于完善与发展关系理论。1972 年，他提出了关系代数和关系演算的概念，定义了关系的并、交、投影、选择、连接等各种基本运算，为日后成为标准的结构化查询语言(SQL)奠定了基础。

考特还创办了一个研究所(关系研究所)和一家公司(Codd & Associations)，他本人是美国国内和国外许多企业的数据库技术顾问。1990 年，他编写出版了专著《数据库管理的关系模型：第二版》，全面总结了他几十年的理论探索和实践经验。(2002.07)

UML之父 Grady Booch

《UML用户指南》、《UML参考手册》、《统一软件开发过程》……

学过面向对象软件工程的人都知道这几本书意味的是“权威”、计人热兴趣的是，在这几本书的封面上都赫然印着一个名字：Grady Booch。除了这三本书之外，他还撰写了被称为“OO圣经”的Object-Oriented Analysis and Design with Applications，凭借上述四本皇皇巨著作，Grady Booch给自己换来了“OO教父”的美誉。

1977年，Grady Booch毕业于美国空军军官学校。他并没有去做一名翱翔太际的“做气雄鹰”，而是在加州大学Santa Barbara分校获得了软件工程硕士学位。此后的15年中，他致力于绝世面向对象软件工程领域研究探索，在OO软件开发技术上提出了许多突破性见解，例如迭代式软件开发过程（iterative software development process）及软件体系结构（software architecture）等等。迄今为止，除了已出版的四本书，他已在各学术性刊物上发表了70多篇有关OO技术与软件工程方面的文章。

但是，Grady Booch取得的最大成就是在软件开发方法和面向对象建模语言上。最早的面向对象建模语言出现于70年代中期，从1989年到1994年间，其数量从不到十种陡增到五十多种。但是OO方法的使用者不了解不同建模语言的优缺点及彼此之间的差异，因而很难根据实际用特点选择合适的建模语言，于是爆发了一场“方法大战”，1990年，报新方法出现了，其中最引人注目的方法是Booch、OOSE和OMT-2等，这三种方法分别由Grady Booch、Ivar Jacobson和James Rumbaugh三人创造。比较老的C++程序员也许还记得这样一本书：Designing Object Oriented C++ Applications Using Booch Method（Robert Martin著，Addison-Wesley 1992年出版），在这本十年前的著作中，我们已可以看到当时已经相当成熟的Booch方法对于软件开发的指导意义，在Booch方法中已经有了“静态建模”和“动态建模”的概念，并且有了一系列完整的图形化工具，包括类图、顺序图等。

面对众多的建模语言，由于用户没有能力区别不同语言之间的差别，他们很难找到一种适合应用环境的语言，极大地妨碍了用户之间的交流。因此对于学术界来说既有必要精心比较不同建模语言的优缺点，总结面向对象技术应用的实践，组织联合设计小组，根据应用需求取其精华、去其糟粕，求同存异，统一建模语言。1994年10月，Grady Booch和Jim Rumbaugh开始致力于这一工作。他们首先将Booch 93和OMT-2统一起来，并于1995年10月发布了第一个公开版本，称之为统一方法UM 0.8（United Method）。1995年秋，OOSE的创始人Ivar Jacobson加盟到这一工作。经过Booch、Rumbaugh和Jacobson三人的共同努力，他们在1996年6月和10月分别发布了两个新的版本，即UML 0.9和UML 0.91，并将UM重新命名为UML（统一建模语言，Unified Modeling Language）。1996年，他们又倡议成立了UML成员协会，以完善、加强和促进UML的定义工作。当时的成员有DEC、HP、I-Logix、Itellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI以及Unisys。这一机构提对UML 1.0（1997年1月）及UML 1.1（1997年11月17日）的定义和发布起了重要的促进作用。

从1980年起，Grady Booch就是Rational公司的首席科学家，此外，他还是美国计算机协会（ACM）、电子及电气工程师学会（IEEE）和美国科技发展协会（AAAS）等组织的成员。{2002.08}

怀念 Dijkstra

2002年8月6日，在与癌魔多年的斗争之后，计算机科学及工程界的泰斗Edsger Wybe Dijkstra教授在荷兰Nuenen的家中与世长辞，享年72岁。

1930年，Dijkstra出生在荷兰鹿特丹，他的父亲是一位化学家，母亲是数学家。他在美顿大学获得了数学和理论物理的硕士学位，又在阿姆斯特丹大学获得了计算机科学博士学位。1952年至1962年间，他在阿姆斯特丹数学中心做程序员，1962年至1984年在艾恩德霍芬科技大学担任数学教授，1984年至1999年，他在美国奥斯汀的德克萨斯大学担任计算机科学教授，1999年退休，任德克萨斯大学名誉教授。

Dijkstra提出了一个著名的论断：算法逻辑是而且必定是一切有用的计算机程序结构的基础。此外，他的著名观点还包括：将操作系统作为明确同步的顺序进程来构造、计算机程序开发的正规化、有效控制不确定性的智力投资等。他发明的最短路径算法有着极高的效率，他还设计并实现了第一个Algol 60语言编译器。他还是废除“GOTO语句”的领导者。

Dijkstra是一位高产的作家。在http://www.cs.utexas.edu/users/EWD这个网址，你可以找到他全部超过3，000篇的作品，他经常与数百位朋友、同事通信——不是用电子邮件，而是传统的信件。无论是写学术论文还是写信，他都坚持使用钢笔，而不用计算机。

Dijkstra成广为人知的则是他的智慧、雄辩而一针见血的话语。他说：“计算机能不能思考？这个问题就好象‘潜水艇能不能游泳’一样。”

当年轻的科学家问他如何选择研究课题时，他回答：“只做你能做的事。”在获获图灵奖之后的演讲中，他说：“作为一种工具，计算机使我们的文化起了深远的改变，但也仅此而已；作为对人类智力的一种挑战，计算机就起的野蛮大波在人类历史上是史无前例的。”

Dijkstra提出了许多新概念、新术语，大大丰富了计算机科学的语言。他提出的概念包括结构化程序设计、问题分解、同步、死锁、“哲学家晚餐”、最弱前提……以及著名的用于控制计算机进程同步的“信号量”。牛津英语词典收录了他在计算机科学领域中使用的“vector”和“stack”这两个词。

在他的科学生涯中，Dijkstra一直坚持学院派的高风亮节，不让自己的研究受到来自商业、管理、政治等等因素的影响。简单、优美而有说服力，还就是他的风格。在程序设计和数学领域中，他对“优雅”的一贯坚持激励了成千上万的人，他把自己的工作评价为“最高标准”，并以此鼓励他的朋友做到同样的高度。另外，他还义无反顾地担当了苏格拉底的角色——一城邦的牛虻。他一次又一次地指出时髦观点中存在的错误，像牛虻一样刺激同家这匹“骏马”不断前进。和苏格拉底一样，他最重要的遗产就是那些未完的研讨、归纳了一半的理论和没有完成的探索。所有与他讨论、共事过的人，尤其是那些参加过他在艾恩德霍芬和奥斯汀组织的读书小组（著名的“星期二下午俱乐部”）的人，都从他身上受益匪浅。

Dijkstra是1972年图灵奖的得主——图灵奖常常被称为“计算机科学界的诺贝尔奖”。他是荷兰皇家科学院的院士，美国国家科学院的院

士、同时还是荷兰皇家科学院院士、同时还是不列颠计算机协会的名誉会员。他还在 1974 年获得过美国信息处理学会联合会（AFIPS）颁发的哈里·古德奖，在 1982 年获得电气电子工程师协会（IEEE）颁发的计算机先驱奖，在 1989 年获得美国计算机学会（ACM）为计算机科学教育作出突出贡献者而颁发的 SIGCSE 奖。雅典经济大学曾在 2001 年授予他名誉博士头衔。2002 年，日本的 C&C 基金会对 Dijkstra 的评价是："他对软件基础理论、算法理论结构化程序设计和信号机制进行了开创性的研究，为计算机科学奠定了坚实的基础。"

值此巨星陨落之际，让我们借用曼多给苏格拉底的悼词："我们毫无疑问地宣称，在我们所知的与他同一时代的所有人中，他是最聪明、最公正、最优秀的。" Edsger Wybe Dijkstra 引导了并且将继续引导这个星球上所有的程序员，他的贡献和影响将与世长存。(2002.09)

C++"第一神仙眷侣"

1998 年 6 月 22 日，在相识相知 20 年后，Barbara Moo 和 Andrew Koenig 在新泽西的莫里森镇喜结连理。"C++ 之父" Bjarne Stroustrup 等人参加了他们的婚礼。Andrew Koenig 和 Barbara Moo 堪称 C++ 研究领域的"第一神仙眷侣"。他们两人不光有着多年的 C++ 开发、研究和教学经验，而且还亲身参与了 C++ 的演化和变革，是对 C++ 的变化和发展起到重要影响的人。不管是技术讲座，还是著书立说，Andrew Koenig 和 Barbara Moo 都是如影随行，夫唱妇随（虽然这不太符合西方人的习惯）。

Andrew Koenig 是 AT&T 大规模程序研发部（前贝尔实验室）成员。他从 1986 年开始从事 C 语言的研究，1977 年离开哥伦比亚大学加入贝尔实验室。他编写了一些早期的类库，并在 1988 年组织召开了第一个真正意义上的 C++ 会议。在 ISO/ANSI C++ 委员会成立的 1989 年，他就加入了该委员会，并一直担任项目编辑。到目前为止，他已经发表了 C++ 方面的 100 多篇论文，他曾在哥伦比亚大学和普林斯顿大学执教，为 Usenix 协会、斯坦福大学、波士顿大学、朗讯技术学院、SGIS 会议提供技术指导，还曾应邀到世界各地演讲。最近，Andrew Koenig 开始对 Python 语言产生兴趣，在今年的第十届国际 Python 会议上作主题演讲。

Barbara Moo 现任 AT&T 网络体系结构部门负责人。在 1983 年加入贝尔实验室不久，她开始从事 Fortran 77 编译器的研究工作，这是第一个用 C++ 编写的商业产品。她负责 AT&T 的 C++ 编译器项目直到 AT&T 转让出软件开发业务。她还负责指导 SIGS 会议、朗讯技术学院和 Stanford 大学。

在技术研究领域和开发业务中，他们值得罗列的成绩还远远不止这些。STL 做为"近 10 年来 C++ 最大成果"被纳入标准库，作为 C++ 标准委员会的项目编辑的 Andrew 功不可没，他早期为 C++ 编写的一些库，被标准 C++ 采用，他在 JOOP、C++ Report、C++ Journal 等著名杂志发表了百余篇论文，其中的《Function objects, templates, and inheritance》和《Function Adaptors》等被认为是 C++ 发展史上举足轻重的文献。Barbara Moo 更是"巾帼不让须眉"：她负责领导贝尔实验室的 Foundation 项目时，管理了包括 Stanley Lippman 和 Martin Carroll 等在内的开发团队；她负责 cfront 项目时，开发出了最早的 C++ 编译器。全世界能够做到这些的，恐怕也只有 Andrew 和 Barbara。

提到 Andrew Koenig 和 Barbara Moo，人们肯定不会忘记他们的 3 本著作：《Rumination On C++》、《C Traps and Pitfalls》和《Accelerated C++》。这三本书自出版以来就受到国外诸多大师的推崇，Bjarne Stroustrup（《The C++ Programming Language》的作者）承认除了自己的著作以外，《Rumination On C++》是他最乐意推荐阅读的技术图书。他在接受专业媒体专访的时候说："……我乐意推荐强调编程效率的图书，但这类图书中一直没有让我感到非常满意的。Koenig 和 Moo 的《Ruminations on C++》将会为许多人提供有用的帮助。对于 C++ 是什么样的以及能够做些什么，他们的先见之明在这本书里随处可见。"《C Traps and Pitfalls》历经了十几年的岁月，依然好评如潮，屡次重印，成为很多 C 专家的案头必备。至于《Accelerated C++》，ACCU 的主席 Francis Glassborow 称是自己所见的最好的"introductory"的书籍，他更是毫不吝啬地夸赞道，Andy 是世界上最出色的几位 C++ 专家之一。

这么多年来，Andrew Koenig 和 Barbara Moo 不像 Bjarne Stroustrup、Stanley Lippman 等人那么受到国内技术界的关注。但是，这丝毫不影响他们对 C 和 C++ 的卓越贡献以及他们大师级的技术水准。(2002.10)

面向对象的"吹鼓手"：

Ivar Jacobson

"面向对象"的建模语言出现在 20 世纪 70 年代中期至 80 年代后期之间，那时方法论的学者们面对面向对象编程语言的新风格和日趋复杂的应用，开始用分析与设计的新方法进行实践。在 1989 到 1994 年间，"面向对象"的方法从不足 10 种增加到 50 种以上。面对这么多的方法，很多用户很难找到一种完全满足他们要求的建模语言，于是就加剧了所谓的方法战争。通过从实践中学习，新一代方法开始出现，其中一些优势明显的方法脱颖而出，最著名的有 Grady Booch 的 Booch 方法、Ivar Jacobson 的 OOSE（"面向对象"的软件工程）方法和 James Rumbaugh 的 OMT（对象建模技术）方法。这些方法中的每一种方法都是完整的，但是每一种方法又都被认为各有优缺点。简单来说，Booch 方法在项目的设计和构造阶段的表达力特别强；OOSE 对以"用例"作为一种途径来驱动需求捕获、分析和高层设计提供了极好的支持，而 OMT-2 对于分析和数据密集型信息系统非常有用。

到 20 世纪 90 年代中期，Grady Booch（Rational 软件公司）、Ivar Jacobson（Objectory 公司）和 James Rumbaugh（通用电气公司）各取所长，进而推动三种方法的统一进程。1994 年 10 月，Rumbaugh 加入 Booch 所在的 Rational 公司，从而正式开始了 UML 的统一工作。最初，他们的计划是着重于联合 Booch 和 OMT 方法。"统一方法"（当时的名称）

0.8 版本（草案）在 1995 年 10 月发布，大约就在那个时候，Jacobson 也加入了 Rational 公司，于是 UML 项目的范围又做了扩充，把 OOSE 也结合进来。经过努力，他们在 1996 年 6 月发布了 UML 0.9 版本。在 1996 年全年，他们在软件工程界征求和收集反馈意见。在此期间，有很多软件组织聚集起来把 UML 作为商业策略来考虑，他们与几个愿意致力于定义一个强大而完善的 UML 的组织合作成立了 UML 伙伴组织。从此，UML 成为了建模语言的事实标准。

现在，Ivar Jacobson 博士是 Rational 公司的首席科学家和副总裁。早年，他在斯德哥尔摩皇家技术学院获得了工科博士学位，并曾经作为访问学者在麻省理工学院进修。毕业之后，他在爱立信（Ericsson）工作了一段时间，后在瑞典创办了 Objectory AB 公司。该公司于 1995 年被 Rational 收购之后，他也加入了 Rational 的阵营。

Ivar Jacobson 对软件开发的其他很多领域作出过重大的贡献。他在爱立信公司时提出了“将组件用作软件开发中的‘积木’”的构思；他还发明了雕件图和协作图，用于对组件之间的交流建模；他还首先将状态转换图应用在组件建模上，除了 UML 之外，他也是结构化建模语言（SDL）最初的设计者，这是一种在电信界通用的建模语言；是他发明了“用例”的概念，以详细说明软件系统的功能性需求；他还发明了“用例驱动”的开发方法，通过“用例”来驱动用户界面设计、软件设计和测试；他开发了一种基于“业务用例”和“业务对象”的业务建模技术；他还发明了基于组件的 Objectory 过程，是 RUP 过程的前身。

Jacobson 博士著有五本关于“面向对象”软件工程、软件复用和软件开发类书籍，其中二本已经有了中文译本：《UML 参考手册》、《UML 用户指南》和《统一软件开发过程》。这五本书都是这个领域中具有权威性的著作。

在工作之余，Ivar Jacobson 喜欢滑雪，徒步旅行和网球。他也很喜欢跳舞，尤其喜欢来自拉丁美洲的伦巴萨舞。对古典音乐、爵士乐和流行音乐他很有兴趣。他有四个孩子（两男两女），他们都已长大成人，都继承了父亲的职业，在 IT／电信行业中工作。

在他的大女儿 Agneta 的公司主页上，有一个名叫“Cyber Ivar”的虚拟人物，能够与用户对话，并解答一些常见的技术问题。在 http://www.jaczone.com，你可以看到这位长相与 Ivar Jacobson 酷似的 Cyber Ivar，开始“他”聊天。（2002.11）

认识 C++ 之父

认识 C++ 之父，要从他的名字开始。

C++ 之父的名字是 Bjarne Stroustrup。呃……我猜你正在默念这两个单词，并且我猜你多半已经念错了。按照 Stroustrup 博士自己的说法，对于非北欧语系的人来说，这个名字的确很难读。你应该“先用挪威语读几遍，在瑞典语嘉上一块马铃薯，然后再来读”。好于我的中国人，读成“比扬尼·斯卓斯朱夫”是比较接近的。

然后，你应该如何称呼他呢？你可以叫他“Stroustrup 博士”或者“Stroustrup 教授”。如果你同他的话，他会让你直接叫他“Bjarne”，因为他对于这些繁文缛节没有兴趣。不过，请尽量不要用“BS”这个缩写来称呼他，不论是在口头上还是在书面上。这个缩写在美国俚语中是一个脏字，意思跟汉语的“SB”差不太多。

Bjarne 出生于丹麦的第二大城市奥尔胡斯，那是日德兰半岛东岸一个美丽的城市，共有 250,000 人口。他于 1975 年获得了奥尔胡斯大学计算机系授予的硕士学位——强少有人能在六年之内获得这个学位，而 Bjarne 正是这少数人中的一个。此后，Bjarne 在剑桥大学的计算实验室里研究分布式系统的设计，并于 1979 年获得了剑桥大学授予的博士学位。他是长吉尔学院的一员。在那里他和他的夫人玛利安度过了很多快乐的日子，他们的女儿也出生在那里。

1979 年，Bjarne Stroustrup 进入了贝尔实验室计算机科学研究中心，并[illegible]到了位于美国东海岸的新泽西。自从 AT&T 贝尔实验室一直在 AT&T 实验室研究部门工作。直到最近转投德[illegible]学系之前，他一直担任大规模程序设计研究部的带[illegible]p 设计并实现了 C++ 语言。在过去的十年中，C++ [illegible]面向对象程序设计语言，在许多主流项目中得[illegible]Stroustrup 在面向对象程序设计和泛型程序[illegible]究的领域包括一般系统的编程、交换[illegible]户界面、嵌入式系统及科学计算等。

“计算语言”是同类书籍中流传最广的一部，被翻译成 14 种语言。另一本《C++ 的设计与演化》则将 C++ 的设计思想、灵感、问题、限制等琐碎道来，并辟了程序设计语言类书籍的新天地。Stroustrup 还发表了 60 多篇论文，在 ANSI/ISO C++ 标准的创建过程中，他也扮演了非常重要的角色。

1993 年，由于在 C++ 领域作出的巨大贡献，Bjarne Stroustrup 荣获了 ACM 的 Grace Murray Hopper 奖，并成为 ACM 的成员。他也是 IEEE 的高级成员。此外，他还一直为开源研究委员会效力。在 1990 年，他被《财富》杂志评为“美国最优秀的 12 名年轻科学家”之一；1995 年，他被 BYTE 杂志评为“过去 20 年中对计算机产业造成最深远影响的人物”之一。

以上就是在网上、书上和其他地方可以看到的 Bjarne Stroustrup。今年 10 月，Bjarne 到中国一游，笔者有幸见到了他，由此认识了不太一样的 C++ 之父。

Bjarne对人很有礼貌，但绝不上谄媚。就连Bruce Eckel、Scott Meyers这样的当红技术作家，Bjarne 也只把他们当作后生晚学，并不以为意。他言谈幽默，但绝不上和蔼，在交谈间，笔者问他“对Design by Contract有什么看法”，他答我：“那是一个含糊程序员应该做的。我从《上世纪》70 年代就已经开始这样做了。”嘻得程序天下数再提一个问题。他似乎也并不介意让一个素昧平生的交谈者脸红——老师让学生脸红有什么可介意的呢？

我所认识的 Bjarne Stroustrup 就是这样的一个人，没有什么商业味道，浑身上下散发着学院派的清新与笑傲不朽。在北大讲座的时候，他一会儿走到墙边擦擦作业上，一会又盘腿坐在折叠椅上，甚至还直接把一只脚踩在椅子上。他也不太掩饰自己的想法，每当有人提出提“刺激”的问题时，他的脸上就会写满失望的表情，其至还会长叶短叹——据我的观察，每当有人要他象 C++ 和其他语言做比较时，每当有人问到 Java、C# 时，他的失望表情尤其明显。如果你有机会当面向 Bjarne 提问，记住一定不要再提这类问题。（2002.12）

项目模型及业务流程分析

撰文 / 九点

网络技术的应用所产生的电子流程工作方式就不能如法套搬传统的工作流程，也不是对传统工作流程的简单复制，而需要对传统的工作流程进行合理的优化、改进和重组。

关键词：User Story　脚本发布　迭代　交叉调板

前言

随着技术的不断发展和用户对网站功能需求的不断提高，静态HTML文件已经远远不能满足网站项目的设计。与前几年网站设计由一两名网页设计师自由的创作相比，网站项目的设计和开发越来越像一个软件工程，也越来越复杂。网站项目的设计和开发进入了需要遵循流程和分工的时代，建立规范的、有效的、健壮的开发机制，才能适应用户不断变化的需要，达到长期的计划目标。

网站项目管理（WPM）的原文为Web-based Project Management，即Web应用程序为主要表现方式的架构来进行的项目设计处管理。这样的架构包含了浏览器、网络和Web服务器等关键主体，主要体现在网站设计、以浏览器为客户端的Web应用程序开发（例如信息类网站、网上商店、理机邮局、客户关系管理）等项目管理中。

按照笔者的经验，网站项目管理可以分为以下六个阶段进行控制：（1）需求分析及变更管理。（2）项目模型及业务流程分析。（3）系统分析及软件建模。（4）界面设计、交互设计及程序开发。（5）系统测试和文档编写。（6）客户培训、技术支持和售后服务。

相应地，"网站项目管理是如何完成的"系列也分为六篇。笔者将网站项目管理与软件工程的统一过程管理（RUP）进行参照比较，并结合实际工作经验，力求将网站工程管理的角色、分工、流程进行完整的阐述，使网站项目管理逐渐走向规范化。

本章包括以下内容：

一、编写项目模型文档，使所有人都一目了然

二、业务流程分析及进行流程设计

三、界面工程师设计用户界面原型

四、以用户为中心的设计思考

五、制作设计计划书

六、总结

一、编写项目模型文档，使所有人都一目了然

为什么要制作项目模型文档？

通常用户提出的需求是戏剧的，不完整的，甚至是不正确的，而且更细致的需求经常是在项目开发进行中才被挖掘发现的。这对于开发人员来说是个极其困扰的问题。那么，在进行需求分析后制作项目模型文档，能在项目进入开发前，双方对即将要开始完成的项目的结果有个共同的认识，并提早暴露可能出现的需求变更，那么将大大提高开发的效率和质量。

缺乏经验的项目人员往往在接受任务后迫不及待地进行系统分析和开发，而不愿多花一点时间在同客户反复推敲项目需求和模型，开发过程中想当然地实现为客户做很多返工，费了九牛二虎之力却不讨好，可想而知，在不知道终点在哪里的马拉松比赛中，你会获胜？！

因此在确认了客户的初步需求以后，业务人员应该进行项目模型的设计描述。

首先，我们要定义一下词汇表，并非每个客户或者项目小组或贵都能够明白"用户"、"角色"、"用例"之间的差别，也不见得都能很好地理解"通道"、"窗台"、"后台"到底是什么含义，为了让项目模型文档使每个简易者正确地理解，定义词汇表是非常需要的，尤其是面对传统行业和改进行信息化设计的用户。

模型描述采用最自然的语言进行描述，这份文档是对需求分析报告的进一步描述，使得客户代表、项目经理、开发人员对即将展开的项目通过项目模型的描述产生基本直观的印象，并针对关键的问题进行讨论并达成统一认识，比如功能要求、性能指标、运行环境、投资规模等等。

通过制定项目计划书将项目的目标、实现方式及费用工期确定下来，作为将来项目开发、测试和验收的标准。

二、业务流程分析员进行流程设计

业务流程分析员的人员应该善于简化工作，担任北角色的人员中必须要有具备广博的专业领域知识，并且具有良好的沟通技巧。

业务分析人员重点需要协助客户详细需求进行归纳分析，查找出所有的业务主角，确定业务主角后，每个主角的相关活动及流程应清晰地制定出来，最终设计出逻辑规范、用户界面示意图。比如一个电子商店系统，除了系统管理员、业务经理、业务员、物流配送员、客户服务人员等角色以外，可能还存在许外部协作单位的不同角色，比如供应商、分销商、广告客户、还有购买用户、甚至再细分为普通消费用户、VIP消费用户、集团消费用户等等，每一类角色参与系统活动时的入口和流程都有所不同，通过逻辑图和示意图，业务流程分析员将系统的机构简要明确地进行描述。

在进行业务流程设计，需要注意以下事项：

- 调查用户间结环境和配置，使架构设计师能够制定合理可行的系统架构；
- 调查用户偏好和技能水平，这将直影响到项目开发的深度和用户界面的设计；

"虽然开发人员和管理人员很容易认为他们了解用户需要，但实际情况常常不是这样。人们往往关注于用户应该如何执行任务，而不是用户偏好如何执行。多数情况下，偏好问题不仅仅是简单地认为已掌握了用户需要，尽管这本身就很值得研究。偏好还要由经验、能力和使用环境决定。"

- 预测并制定系统的性能指标，为测试人员编写测试计划提供依据。

许多项目设计中比较重视功能的实现，测试阶段看似满足了客户的需求，但一旦投入使用的时候，便会发现性能上面临着一个个瓶颈，客户由于对专业知识的了解程度有限，也往往忽略了这方面要求，因此为了避免日后陷入纠纷，事先预测并制定性能指标是非常重要的。

三、界面工程师创建用户界面原型

为了在实际系统开发前，创建用户界面模型是非常重要的、开发原型的成本远远低于实际开发的成本，在项目初期，创建完整的用户界面脚本和演试系统的所有功能和可用性，并能够使客户代表参与讨论及修改，可以大大提高项目的成功几率。

创建正确可行的原型以后，系统分析，设计及代码的编写都必须遵照原型进行，确保构建的系统是正确的，测试人员和客户也能够在开发过程中即时地参与检查，可以有效地保障了项目的质量。

根据业务流程分析员构建典的流程分析逻辑图及示意图，界面设计工程师开始设计制作用户界面原型，目前这个阶段，对于界面设计人员