^v^v^v^v^v^v^v^v^v^v
Borland 传奇
李维 著
电子工业出版社
内容简介
本书披露了 Borland 各个重要产品开发鲜为人知的内幕故事，第一次让读者了解了 Borland 顶尖技术天才的风采，并展示了在美国软件技术市场上波澜壮阔、激动人心的技术大战和产品大战，而且从资深技术人员的角度阐述了自己对于软件技术发展各阶段的深刻理解和对未来软件的发展趋势的思考。

图书在版编目(CIP)数据
Borland 传奇 / 李维著. －北京：电子工业出版社，2003.4

1SBN 7-5053-8658-1
Ⅰ.B...　Ⅱ.李...　Ⅲ.①软件工具-简介②程序语言-简介
Ⅳ.①TP311.56②TP312
中国版本图书馆 CIP 数据核字(2003)第 027159 号
责任编辑：孟迎霞
印　　刷：北京中科印刷有限公司
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：148x210 毫米　1/32　印张：12.88　字数：263 千字
版　　次：2003 年 4 月第 1 版 2003 年 4 月第 1 次印刷
印　　数：6000 册　定价：28．00 元
作者简介：
李维：乔治亚理工学院信息硕士，大陆《程序员》杂志、台湾《RUN!PC》之技术专栏作家，为著名实战 C++ Builder/Delphi 系列丛书的作者。目前已发表信息文章百余篇、技术书籍超过十多本。曾参与数个大型项目的开发，也经常受邀于各种技术研讨会和企业机构进行专业信息讲座。对 C/C++、Delphi、Java、分布式架构以及中间件技术都有着丰富的研究和实践经验，现为台湾 Borland 的资深技术顾问。

写在出版之前

Borland 是中国程序员最为景仰的公司之一，特别是对于资格老一点的程序员而言，他们学习编程的起步就是 Borland 的 Turbo Pascal、Turbo C++。

根据权威调查机构 2002 年的中国专业开发者调查报告显示，有超过一半的程序员使用 Delphi 作为主要的开发工具，Jbuilder 的市场占有率在 Java 开发工具中也名列榜首。

人们常常说，初恋情人最让人难以忘怀。更何况 Borland 的开发工具确属一流，所以很多人都具有所谓的"Borland 情结"，他们非常熟悉和热爱 Borland 的各种产品，但对于 Borland 公司的情况却几乎一无所知。

在开发工具领域，Borland 最主要的对手就是 Microsoft，坊间关于 Microsoft 的故事人们都已经了解得太多了，而对于 Borland 公司和她的重要人物，大家却知道得十分有限，其实 Borland 拥有非常杰出的技术大师，其企业发展也是跌宕起伏，有着十分精彩的故事。

Borland 以技术产品 Pascal 编译器起家，DOS 时代盛极一时，20 世纪 90 年代初曾经演出了一场小鱼吃大鱼的好戏，并购开发 DBASE 的当时第五大软件公司 Ashton-Tate，成为第三大的软件公司(前两位是 Microsoft、Lotus)，并且 Borland 的电子表格软件 Quatro Pro、数据库软件 Paradox 都在市场上表现不错，开发工具 Borland C++更是风光无限。遗憾的是历史上直接挑战微软的对手最终或者销声匿迹，或者被人收购，如 WordPerfect、Lotus、Netscape 等，Windows 时代 Borland 也遭受重创，在应用软件战场全线崩溃。不过在开发工具领域，Borland 凭借其卓越的技术实力，顽强地生存下来，即使其天才的架构设计师 Anders 在 1996 年就被微软挖走，公司决策也曾经多次失误，但 Borland 仍然找到了自己的发展之路。

Borland 的秘密在哪里？Borland 产品成功和失败的原因又有哪些？Borland 盛极而衰，又东山再起，这些年她经历了哪些重大转变?她又是如何引领技术的发展的？

李维先生的《Borland 传奇》一书，为我们披露了 Borland 各个重要产品开发鲜为人知的内幕故事，以 Borland 为例展示了在美国软件技术市场上波澜壮阔、激动人心的技术大战和产品大战。并且从一个资深技术人员的角度，阐述了自己对于软件技术发展

各阶段的深刻理解以及对未来软件的发展趋势的思考。
Borland 的发展史就是一部软件技术的发展史，一部软件产品，惨烈竞争的发展史。对于技术人员，对于管理人员，借鉴研究 Borland 的发展历程都是十分有意义的。

蒋涛
《程序员》杂志社社长
2002 年 10 月于中国北京

序

公元 2000 年，当笔者的第一篇有关 Borland 的回忆录在"Programmer 深度论坛(http://forum.vclxx.org)"发表之后，立刻得到了极为热烈的回响。在"Programmer 深度论坛"上，这篇回忆立刻成为当时点击率最高、阅读量最大的文章。随后笔者也意外地发现这篇文章被许多人在网上转载。很多读者不断地写信给我，希望能继续阅读随后的回忆文章。更有许多不认识我的大陆朋友也不断询问这篇文章的来源、作者。这么热烈的回馈，是我没有想到的。最早撰写和发表这篇回忆文章的原因之一是笔者已经离开了 Borland，但觉得 Borland 的许多故事非常精彩、非常值得回味，因此才动了写些小故事的念头。另外的一个原因，则是当时"Programmer 深度论坛"成立未久，为了吸引网友到"Programmer 深度论坛"共襄盛举，因此当时的创始坛主之一、也是我的好友李匡正先生希望我写一些吸引人的东西来帮帮忙。于是，这才促成了这篇回忆文章的诞生。

不久之后，大陆的《程序员》杂志也和笔者联络上了。《程序员》除了希望我能够为他们写一些 Borland 产品的技术文章之外，也对这些 Borland 的故事非常感兴趣，因此 2001 年的 7 月份在《程序员》杂志刊登了"Borland 的故事"一文。没有想到这篇已经在 Internet/Intranet 上广为流传的文章，刊登之后仍然引起了广泛的响应，许多读者都好评有加，强烈地希望《程序员》杂志能够继续刊登后续的文章。当然，《程序员》杂志也立刻和笔者联络，希望我能立刻撰写随后的故事。在此之后，台湾的《RUN!PC》也刊登了这篇文章，读者的反应也是如出一辙，热切地要求后续的内容。

我想，这篇文章之所以会引起巨大的反应，大概是因为许多软件人员都使用过 Borland 的开发工具，最明显的就是 Borland 的 Turbo 系列。相信 Turbo Pascal 是当初许多人学习数据结构(Data Structure)的必备工具，笔者就是其中之一。而 Turbo C、Turbo C++、Borland C/C++等更是许多人学习 C/C++语言的启蒙工具。不管这些人现在使用的是什么工具，即便已经不再使用 Borland 工具，但这些成长和学习的记忆仍然埋藏于每一个程序员的心底，而这篇回忆文章可能勾起了许多人早已消失许久的回忆，顿时之间这些曾经苦涩和欢乐的记忆又激荡在许多人的脑海里。当然这篇回忆文章也能让许多人了解到当年许许多多隐藏在背后的故事答案：为什么软件历史会这样发展？寻求答案的好奇心从不会因为时间的消逝而磨灭。当这篇回忆文章揭开了许多故事背后的答案之后，当然会引起读者更大的回响，因为这会让我们想要知道的更多。而更多封尘在我们内心深处的问题，会开始在脑中再次询问--接下来的发展会是什么样子？

在撰写第一篇回忆文章时，我的感觉比较轻松，因为那时我已经离开了 Borland，没有什么负担。但是"世事难料"，也许是和 Borland 的缘分未了，在 2001 年底，我又回到了 Borland 工作。再次成为 Borland 的员工之后，在写作时就有了顾忌，不知道结果会怎样，所以在这方面写作的进度使停顿了下来，甚至有很长一段时间没有任何的进展。不过后来仔细一想，这些只不过是一些历史的事迹以及笔者个人的想法和观察，如果 Borland 连这些都无法面对，那又要如何面对未来呢？于是我又开始断断续续地拾起了进度，慢慢地写了一些后续的内容，并且同步地发表于大陆《程序员》杂志和台湾的《RUN!PC》杂志。

集结这些回忆记录、发表我对未来软件趋势发展的观察和看法，是因为许多读者都希望能痛痛快快一次看完这些精彩的故事，而不是"望穿秋水"地等待我时断时续，因此不断地建议我集结出书。当时，我的第一个反应是这些故事内容的长度可能不够成为一本书，而且我的工作太忙，因此并没有把这个建议放在心上。不过在后来不断地撰写后续文章时，我才发现，以往的回忆不断地随着笔尖涌出。原本只打算写三四篇的计划，竟不可收拾地愈写愈多。而且随着叙述过往的回忆故事，我也开始思索一些更深的问题，并且开始观察软件趋势的发展以及 Borland 的演变等重要问题。我开始想，能否通过 Borland 的成长、衰退以及东山再起等发展史实，了解软件开发对于每一个软件人员的影响？如果换作我或读者来担任 Borland 的 CEO，那又会如何经营？如何面对挑战？如何帮 Borland 走出困境？在这些年的奋斗过程中，Borland 经历了哪些改变？Borland 要如何面对未来的竞争？软件开发的趋势对于 Borland 的影响又是如何？或许是因为笔者深爱着 Borland 吧，这些问题开始不断地在我的脑海中出现，不断地询问着自己。也许读者看完本书之后，也会开始遭遇和我一样的情形，开始思索这些复杂又有趣的问题。当然这些后续的发展是当初我所没有预想到的，但对于自己而言，这也是额外的收获。

事实上，我认为这些问题是提升自己最好的锻炼，因为可以培养我们对于事情的观察能力，去思考问题和背后的意义，并掌握未来的发展。当软件人员开始注意到这样的思考之后，往往也就预示着开始要更上层楼，这也是笔者这么多年和许多工程师工作时观察到的现象。只可惜事实证明，这些毕竟是只有少数人才会注意到的事情。七八年前，当我为鼎新计算机工作时，当时的副总要求我除了开发产品和技术之外，每个星期必须向他报告笔者对于未来市场、产品和技术趋势的意见和报告。也就从这个时期开始培养了笔者思索这些问题的习惯。数年后，当我开始为 Borland 工作，并成为产品经理之后，我又开始观察产品和技术趋势。这些训练都让我很自然地在撰写文章和书籍时开始观察过往的历史并且开始询问自己相关的问题。

因此，当《程序员》杂志的蒋涛先生也询及我出书的意愿之后，笔者开始认真地考虑这个可能。除了叙述 Borland 各项重要的战役故事、产品开发的幕后史以及精彩的故事之外，笔者也想写写这些年观察到的 Borland 的转变史，以及转变之后的影响。另外还想写出 Borland 未来面对的挑战、软件开发的趋势等有趣的讨论。当然，Borland 强韧的生命力仍然在继续不断创造她的历史，对于笔者和喜爱 Borland 的人来说，这些可都是可歌可泣的故事。

对于本书的架构，我也经历了一个从起初模糊的概念到最后清楚地表达自己想要描述的内容的过程。大致上，我将本书分为三个部分，这分别是：

- Borland 产品开发的精彩故事。在这些章节中，我将就记忆所知为读者娓娓道来许多或精彩、或动人、或令人惋惜的故事。它们包括了 Borland 本身的故事、C / C++ 开发工具、Delphi、Jbuilder、Visual dbase、IntraBuilder 和组件以及中间件的发展故事，让读者了解产品开发的艰辛史以及 Borland 如何在激烈的竞争中力求生存之道。
- Borland 的转变发展。许多人都知道 Borland，但是可能没有注意到 Borland 在将近 20 年来的变化。其实，每一个阶段的变化都是由于特定的人物和时势所造成。这部分笔者将为读者介绍这些微妙的变化并且叙说其背后的原因和故事。
- 软件趋势的发展以及未来的 Borland。如今的软件技术多得令人目不暇给，软件技术的发展也是以 10 倍的速度向前推进。在现在多元化的软件技术中，到底未来的主流技术或是平台将由谁胜出？软 / 硬件厂商之间的合纵连横又将对软件技术产生什么影响？Borland 在未来又将如何竞争？未来的 Borland 产品又将如何占有一席之地？这

都是笔者想要和读者一起分享并讨论的内容。
本书算是相当有趣的一个代表，因为她融合了精彩的故事、发人省思的竞争过程、最新的软件技术趋势以及对未来的思考。在阅读本书的过程中，读者可能会有不同的感受：可能会勾起许多人封尘已久的记忆；也可能会让读者感受快乐、惋惜、沮丧和振奋；也可能会让读者感觉焦虑，因为面对现在和未来多元化软件技术的发展，许多读者可能都不知所措。不过在阅读完本书之后，相信读者应该会对未来信息产业的发展有一定的掌握。
因此，这本书的内容可能会成为一些读者对自己职业生涯的纪念；成为一些读者学习信息历史的明镜；抑或成为大多数读者掌握现在和来来信息技术发展趋势的参考。对于笔者而言，此书的整理算是对自己以往工作和回忆的记录以及面对未来的准备。
我不是以写作为生，因此撰写时间非常有限。什么时候能够完成这本书?我并没有把握，只能尽力而已。在写作的过程中，我也历经了数个阶段：从一开始的兴奋莫名，到强求自己一定要在什么时候出书，再到现在随缘，做好了再说的阶段。
最后，我要谢谢所有鼓励过我的朋友，这毕竟是我第一本非纯技术类的书籍，心中难免心虚。自己所能做的便是尽量提供丰富的内容，至于最后的结果，当然交由各位读者来评判了。
李维
2002.10 于台北新店
目录

声 明
本书的内容主要基于我个人的回忆和看法，没有任何特别的偏见。许多历史资料来自我的记忆以及很多相关人的叙述和诉说，也许细节之处不是百分之百的正确，不过我可以保证这些内容的可信度。也许有一些历史事件确定的发生时间和顺序不一定与我的记忆完全一致，但相信大部分应该出入不大。当然，如果有读者知道更确切的史实，非常欢迎您的指正。
希望这些故事的经历能够一直陪伴我，陪伴所有对软件开发有兴趣的读者们一路走下去，成为大家共同奋斗的记忆，这也是我写作本书的最大心愿，谢谢。
^v^v^v^v^v^v^v^v^v^v
第一章 Borland 的诞生和发展
一直想写篇文章，讲述我个人在过去 10 多年来工作中经历的一些事情，以及这些日子中那些我心目中的伟大的工程师们对于信息界的贡献。如果读者和我的年龄差不多，那对于这些内容可能会更有兴趣，因为它们揭示了当时许多软件兴起和没落的过程以及原因。虽然这些事情距离我们很遥远，但我相信许多人仍然对于背后的故事感兴趣。即便没有经历过那段美好的回忆，那也可以把这些内容当成一个有趣的故事来读吧。不过我想，更重要的是让我们一起认识一些伟大的人物，我个人对于其中的许多人都非常佩服，也非常羡慕。甚至我常常在想，如果自己也有他们的环境，是不是也能够和他们一样这么有成就呢？这些人对于以往都有着重要的贡献，对未来也仍将有着重要的影响，因为他们都有一身不凡的技术。对于许多重要的人物，我都尽量收集了他们的照片，让各位也能够认识这些优秀的工程师、杰出的人物。
当然，如果各位能够从这些内容中学习到失败的教训以及成功的经验，那么本书就更有价值了。
Borland 的兴起
记得大学时，第一个在 PC 上使用的软件就是 SideKick。这个至今让我仍然无法忘记的软件，也曾让许多人津津乐道，而 Borland 当时也就是以 SideKick 成为全球知名的软件公司。不过 Borland 第一个奠立创业基础的软件，却是我大二用来交作业的 Turbo Pascal，而 Turbo Pascal 也是我听到的第一个关于 Borland 的有趣的故事。
当年 Philippe Kahn(Borland 的创始人)和 Anders Hejlsberg 到美国创业时，便由 Anders 以汇编语言撰写了 Turbo Pascal 的编译器，而 Philippe 则包办了 Turbo Pascal 其他的部分。在这两位仁兄开发完 Turbo Pascal 之后，穷得快连登广告的钱都没有了。Philippe 为了在 Byte 杂志(还记得这个著名的杂志吗？)刊登 Turbo Pascal 的广告，和 Anders 商量了一个方法，那就是直接约 Byte 杂志的人到当时 Borland 的办公室讨论刊登广告的事情。
当 Byte 的人到了 Borland 之后，Philippe、Anders 和公司的助理小姐故意忙着接电话，接受 Turbo Pascal 的订单，并且告诉 Byte 杂志的人等一下。过了一阵之后 Philippe 才进入房间向 Byte 的人道歉，说他们的 Turbo Pascal 受到市场的热烈欢迎，订单源源不断地到来，因此可能不需要在 Byte 杂志刊登广告了，接着 Philippe 向 Byte 的人展示 Turbo Pascal 这个产品。由于在当时的机器中 Turbo Pascal 能够在极少的 RAM 中常驻

执行，又提供闪电般的编译速度，这立刻让 Byte 杂志的人当场震惊。凭着专业知识和丰富的经验，Byte 的人立刻知道这将是一个革命性的软件，因此马上希望 Philippe 能够在 Byte 杂志刊登 Turbo Pascal 的广告，并且愿意以半价刊登。当然，Philippe 也立刻爽快地答应了，于是一个革命性的软件 Turbo Pascal 终于在 Byte 杂志刊登出来了。当时售价 49.99 美元的 Turbo Pascal 立刻为 Borland 带来了大量的财富，Turbo Pascal 也马上成为 PC 上除了基本的 Basic 之外最畅销的开发工具，由此正式揭开了 Borland 影响 PC 开发工具近 20 年的历史的序幕。

Turbo Pascal 是由 Anders Hejlsberg 亲自开发的，并且和 Philippe Kahn 谈好的条件是 Borland 每卖出一套 Turbo Pascal，Anders 便从中抽取一份版权费。由于当时软件的价格不算便宜，能够写编译器的人更是少之又少，所以编译器工程师通常都能够获得优厚的报酬。因此当时 Anders Hejlsberg 在完成了 Turbo Pascal、并且和 Philippe Kahn 谈好了合作条件之后，Anders 理所当然地认为一套 Turbo Pascal 会定价数百元美金，因为这不但是当时一般编译器的价格，而且 Turbo Pascal 还内含了一个开发环境和编辑器(Editor)，这是当时许多工具没有提供的。

没有想到极具商业头脑的 Philippe Kahn 了解到：如果把 Turbo Pascal 定价在数百美金，那么 Turbo Pascal 可能只会卖出数百到数千套，无法冲出大量的销售额。因此，Philippe Kahn

以极大的勇气，瞒着 Anders Hejlsberg 只把 Turbo Pascal 定价为 49.95 美金。这种价格在当时对于编译器和开发工具来说简直是不可思议的低价。当 Anders Hejlsberg 知道了 Philippe Kahn 的定价后，简直快气昏了。因为在这么低的价格下 Anders 的版税金一定少得可怜，因此当时 Anders 说他把最好的 Pascal 开发工具拿去让一个白痴销售。

没有想到的是，Philippe Kahn 的定价策略获得了极大的成功。Turbo Pascal 以极佳的品质和令人不可思议的低价格成为当时最具吸引力的 Pascal 开发工具。当然，在 Turbo Pascal 卖出了让人难以置信的成绩之后，Anders 便再也不提他把专业 Pascal 编译器让白痴去卖这件事了。

关键产品--SideKick

虽然 Turbo Pascal 快速地让 Borland 在当时全世界的程序员中成为最响亮的软件新星，但是真正让 Borland 打人一般计算机使用人群、快速成长为软件巨人的大功臣的，却是 Borland 早期最重要的产品--SideKick。

在 Turbo Pascal 之后，Borland 接着推出了 SideKick 这套软件。SideKick 可以说是随后著名的内存常驻软件(Terminate and Stay Resident-TSR)的始祖，也是 Borland 跨出开发工具领域、让几乎所有 PC 使用者认识 Borland 的关键软件。SideKick 在当时以许多丰富的小工具和记事功能让它成为每一个程序员爱不释手的工具。还记得当时我每天都会使用 SideKick 的 ASCII 对照表和计算器的功能，因为在汇编语言(Assembly)盛行的时期，查阅 ASCII 对照表和在 2 进制、10 进制以及 16 进制之间进行转换是每日必要的工作。

当然 SideKick 也很快成为了畅销软件，在全球狂卖数 10 万套，继续把 Borland 往顶尖的软件公司推进。

所谓的 TSE 代表 Terminate and Stay Resident。这个意思是说，这类软件在执行后会隐藏在内存的某个位置中，但是并没有出现在屏幕上。不过使用者通过一个快捷键就可以立刻调出这类软件让使用者使用，在使用完毕之后又可以按一个快捷键再度隐藏它。这样的软件运行方式在当时是一项全新的创举。

以我的眼光来看，SideKick 这个软件对于 Borland 来说是非常关键的作品，因为我将 SideKick 归类成"消费型软件"产品。所谓消费型软件，是指可以被所有计算机使用者

使用的软件，而不是只给程序员或是开发者使用的软件。凡是现今比较会赚钱或是规模比较大的软件公司大都属于开发"消费型软件"的公司。例如 Microsoft 除了有和 Borland 竞争得你死我活的开发工具之外，最重要的是 Microsoft 拥有两大"消费型软件"：Windows 操作系统和 Office。这两类软件才是 Microsoft 最赚钱的产品。Oracle 是另外一个很好的例子，数据库几乎是现在任何应用都需要使用的软件。同样，SideKick 就属于这一类型的软件，因为 SideKick 可以被所有的开发者使用来增加生产力，而不管开发者使用的是什么语言。因此当 Borland 推出 SideKick 之后，立刻在全世界狂卖，也成为继 Turbo Pascal 之后 Borland 最赚钱的产品。我认为在后来的数年之中 Borland 走得比较辛苦，便是因为 Borland 再也没有推出像 SideKick 一样属于"消费型软件"的重量级产品，而只有属于程序员和开发者小众市场的产品，这是非常可惜的事情。而"消费型软件"也是到现在我仍然认为 Borland 应该推出的产品。

由于 SideKick 的 TSR 技术是当时独一无二的，而且是如此的好用，这引起了当时许多人的好奇，并且成了所有软件厂商模仿的对象，我还记得稍后许多的计算机信息书籍都以如何学习 TSR 技术作为卖点。也是因为 SideKick 和 TSR 太成功了，因此 Borland 立刻进行了两个工作。第一当然是马上开发下一版的 SideKick，让 SideKick 继续执类似软件的牛耳，以防止其他软件公司推出类似的软件来分食 SideKick 打下的天下。

很快地，Borland 便推出了 SideKick 的后续版本，不但功能更多，而且 SideKick 从原本完全以开发者为中心的软件转变为适合所有计算机使用者使用的消费型软件。看看左图，从产品封面以"Desktop Organizer"为主题便可以了解到 SideKick 在当时的定位。果然，后续的 SideKick 又持续地大卖，这让 Philippe Kahn 非常振奋，也让他雄心大盛，开始想要通过 SideKick 的成功主导 PC 软件的标准，这当然就是 SideKick 一举成名的 TSR 技术。

在 Borland 通过 Turbo Pascal 和 SideKick 大获成功之后，也因 TSR 技术成为大多数开发者津津乐道的软件公司，许多软件公司都开始模仿 Borland 的 TSR 技术开发大量的 TSR 软件。不过当 TSR 技术大量被运作之后。最后却造成众多的 TSR 软件彼此冲突，无法正确地相互共存，这主要是因为许多 TSR 软件都使用了相同的快捷键来调出/关闭软件，或是隐藏在相同的内存位置。我还记得，当时同时使用几个 TSR 软件时，必须遵照一定的运行次序才可以正常使用。

为了解决这个扰人的问题，Borland 开始广邀软件公司，想要以 Borland 为首制定 TSR 的标准。如此一来，只要所有的软件厂商遵照 Borland 制定的标准，那么所有的 TSR 软件就可以彼此正确地运行在 PC 之中。当 Borland 公布了这个想法并且发表了初步的 TSR 标准规格之后，却立刻引起了 Microsoft 的紧张。因为当时 TSR 是如此的流行，Microsoft 害怕 TSR 技术由 Borland 主导之后会让 Borland 成为 PC 软件的霸主，进而严重影响 Microsoft 想主宰 PC 的计划。

因此在 Borland 开始正式制定 TSR 标准之际，Microsoft 便站出来反对 Borland 定义的 TSR 标准，并且声明 Microsoft 将在未来的 DOS 操作系统中加入对于 TSR 的支持，因此没有必要再额外制定 TSR 标准。当时的软件公司，包括 Borland 在内，都无法和 Microsoft 对抗。在操作系统厂商表明了反对立场之后，Borland 的这个构想很快便迫于形势而放弃了。关于 TSR 的争议应该算是 Borland 和 Microsoft 之间的第一场战争。虽然在没有引起太大的烽火之前便很快收场，不过也算是 Borland 和 M1crosoft 第一次真正的交手。也正是由于这次的相争，让 Microsoft 惊讶于 Borland 快速的兴起，并开始正视 Borland 这家在当时还算小的软件公司。

虽然在有关 TSR 的技术之争中 Microsoft 赢得了胜利，不过很奇怪的是，此后 TSR 软件反而开始慢慢地退烧。除了一些少数的公用程序软件仍然使用 TSR 之外，之后便没有

什么重量级的软件是使用 TSR 技术开发的，这算不算是另一桩 Microsoft 介入之后搞砸的技术呢？

最后再叙述一个从 Borland 老员工处听来的有趣故事。许多人一直想要知道：Borland 的总部在哪里？或是想要知道：为什么 Borland 会选择 Scott Valley 作为总部？事情的经过是这样的：

当年 Philippe Kahn 和 Anders Hejlsberg 到美国准备开始创业时，由于没有资金，Philippe Kahn 就在西餐厅打工，负责端盘子的工作，而 Anders Hejlsberg 则努力的在开发 Turbo Pascal。

当 Philippe Kahn 存了一笔小钱之后，两个人便开始了创业大计。首先他们必须找到一个公司的总部，可是要在哪里实现 Philippe Kahn 和 Anders Hejlsberg 心中的理想呢？虽然当时他们住在 L.A.附近，但是光凭 Philippe Kahn 存的一点小钱是绝不够在 L.A.大展鸿图的，因此 Philippe Kahn 和 Anders Hejlsberg 决定到比较偏远的地方试试。于是这两位仁兄便开着 Philippe Kahn 的破车往南出发了。听说当 Philippe Kahn 把车开到 Scott Valley 附近时刚好没有汽油了，眼看四周的环境觉得还不错，就决定在这个地方展开 Philippe Kahn 和 Anders Hejlsberg 的创业之梦。就是这个决定让原本默默无闻的 Scott Valley 在数年之后竟成为一个家喻户晓的高科技盛地。

和 Borland 的缘由

Turbo Pascal 是我在大二、大三撰写作业时的最爱，几乎所有的作业都是使用 Turbo Pascal 完成的。当然其时 Horowise 的 Data Structure 这门课也是使用 Turbo Pascal 过关的，因此从那个时候开始，我便非常喜欢 Borland 这家公司，慢慢地也开始对 Borland 有了特别的感情。

在我大二时，Microsoft 推出了 Microsoft Pascal，但是它和 Turbo Pascal 的确有一段差距，我使用了一次之后便把它丢到垃圾桶。稍后 Borland 也推出了 Turbo Basic 1.0。我记得这个编译器非常的棒，编译速度就和 Turbo Pascal 一样快，是一个非常有前途的产品。但是不知道为什么它只有 1.0，之后便和 Microsoft Pascal 一样消失了。后来听说是 Microsoft 和 Borland 互相交换条件，Microsoft 不进入 Pascal 的市场，而 Borland 则退出 Basic 的市场。至于是不是真的确有其事，我就不得而知了。

我在大二初次接触到了 C 语言，第一本阅读的书便是王兴隆先生写的 C 语言书籍，也从此开始和 C 语言结下了渊源。平生第一个使用的 C 编译器便是 Lattice C，不知道还有没有读者记得？当时使用两个 5 吋磁盘抽换以便编译 C 程序的情景，真是麻烦得不得了。稍后 Borland 终于推出了风行天下的 Turbo C 编译器，从此之后 Turbo C 便成了我不离身的工具，而 Borland 也通过 Turbo C 这第三项畅销产品迈向了世界前 10 名的项尖软件公司。

当完 2 年的兵之后，我在中研院首次使用了 C++语言。第一个使用的 C++编译器则是 Zortech C/C++，这家公司稍后被 Symantec 收购成为 Symantec C/C++的核心部门，这个故事稍后再说明。后来 Borland 也推出了它的第一个 C/C++编译器 Turbo C/C++1.0，但是和 Zortech C/C++比较之后，我还是觉得 Zortech C/C++比较好，因此就继续使用 Zortech C/C++。一直到 Borland 的 Turbo C/C++2.0 编译器推出之后，才逐渐成为 C/C++语言的王者，而我也像以往一样把 Zortech C/C++换成了 Turbo C/C++。

在我 1991 年到 Georgia Institute of Technology 念硕士时，终于使用自己的零用钱 49.99 美金购买了生平第一套正版软件 Turbo C/C++4.5，随后又购买了 Borland Pascal。在毕业前的一个 Quarter，Microsoft 推出了 Microsoft C/C++6.0 以及 MFC 1.0，由于 MFC 是第一个 C/C++的 Framework，因此也花了一些钱购买了一套 Microsoft C/C++以便学习 MFC。但是在收到 Microsoft C/C++之后，我却很失望，因为 Microsoft C/C++

6.0 仍然没有 Windows 图形集成开发环境，还是在 DOS 下的集成开发环境。而且以我的眼光来看，MFC 1.0 并不好用。Microsoft C/C++6.0 的 C/C++最佳化编译器在当时也是一个笑话，不但产生的程序代码效率不好，甚至会产生错误的程序代码。许多 IT 杂志也称 Microsoft C/C++6.0 是一个平庸的(Mediocre)产品。因此我就把它丢在一边再也没有使用。在 Microsoft C/C++6.0 推出之后不久，Borland 终于发布了 Borland C/C++3.0，而这套软件也开启了 Borland 雄霸 C/C++编译器长达五六年之久的序幕。Borland C/C++3.0 推出之后，由于拥有第一个在 Windows 下稳定的图形集成开发环境，而且它产生的最佳化程序代码也是 Microsoft C/C++6.0 望尘莫及的，因此，很快地几乎所有的 C/C++程序员都转而使用 Borland C/C++3.0。那个时候几乎所有的公用程序或是 Shareware 都是使用 Borland C/C++开发的，许多硬件厂商的驱动程序也是使用 Borland C/C++3.0 来撰写的。

1992 年我取得 Georgia Institute of Technology 的硕士学位之后，最想进入的公司便是 Borland 和 Microsoft，不过最后我还是决定回台湾工作。在此时 Borland 也逐渐进入了最巅峰的时期，因为 Borland 推出了 Borland C/C++3.1。

Borland 在 Borland C/C++3.0 获得空前的胜利之后，并没有松懈下来，因为 Borland 知道 Borland C/C++3.0 还缺一个最重要的胜利因子，那就是如同 Microsoft 的 MFC 一样的 C/C++ Framework，因为 Borland 也看出了 Framework 将会是未来 C/C++产品中最重要的一环。不过 Borland 此时来到了一个重要的十字路口，那就是到底要自己开发一个和 MFC 抗衡的 Framework，还是直接采用 Microsoft 的 MFC？如果要使用 MFC 的话，那么 Microsoft 会愿意授权给 Borland 吗？如果 Borland 要自己开发 Framework，那么势必要花上一些时间，但是 Borland 想趁 Borland C/C++3.0 如虹的气势再下一城，以便彻底击溃 Microsoft C/C++。因此，最后 Borland 决定向一家叫 White Water 的公司购买一套由这家公司开发的一个 Framework，这套 Framework 便是后来鼎鼎大名的 OWL 的源流。而 Borland 也因为向 White Water 购买了这套 Framework，因而也引进了一个日后非常重要的人物，那就是后来负责开发 Delphi 的一员大将--Zack Urlocker。

C/C++的光荣战役

Borland 购买了 White Water 的 C/C++ Framework 之后，便更名为 OWL(Object Windows Library)，并且很快地推出了以 OWL 1.0 为核心的 Borland C/C++3.1。由于 OWL 比当时的 MFC 1.0 封装得更为完整且好用，再加入 Resource Workshop 可视化能力，以及 Borland C/C++3.1 本身最强劲的编译器和集成开发环境，因此立刻风靡了全世界，其受欢迎的程度更是远远的超过了它的前一版本 Borland C/C++3.0。

Borland C/C++3.1 的畅销，立刻让 Borland 在 C/C++市场一举击溃 Microsoft C/C++，市场占有率超过了 50%，是全球第一的 C/C++产品，也把 Borland 推上了最高峰，成为全世界第三大的软件公司。

在当时，我所工作的开发小组也立刻改用 Borland C/C++ 3.1 来开发 Windows 下的 MRP 系统，而 Borland C/C++3.1 也是我使用过的 Borland 最稳定的 C/C++版本之一。由于那个时候一天到晚都使用 C/C++工作，因此就有了一些小心得。稍加整理后我便投稿到刚成立不久的《RUN!PC》杂志，也许是我的运气不错，《RUN!PC》很快发表了我的文章。就在这篇文章发表之后，台湾的 Borland 分公司注意到了我，开始和我联络，并且从此展开了我和 Borland 的互动。而 Borland C/C++3.1 也是第一套 Borland 免费送我的软件，当然代价就是希望我多写一些 Borland 产品的文章。

接着 Borland 又计划推出 Windows 版的 Borland Pascal。不过在 Borland 开发 Pascal For Windows 时，当时(现在也还是)最具盛名的 Charles Petzold(我看的第一本 Windows 程序设计的书就是这位仁兄写的，相信许多人也是看他的书一路学来的)就说除了

C/C++之外，Borland 不可能做出能够在 Windows 下执行的 Borland Pascal。不过很明显地，即使是 Windows API 的大师 Charles 也错了，Borland 不但做了出来，而且 Borland Pascal For Windows 还非常的畅销，当然 Borland Pascal For Windows 也是后来 Delphi 的根基。

当时的 Borland 可说是不可一世，不但产品大卖，而且日进斗金。Borland 在 Scott Valley 豪华的总部也是在那个时候由 Philippe Kahn 大手笔地花了一亿多美金搭建的(想想 10 年前的 60 多亿台币可以盖什么样的房子？)。不过也许是 Borland 太成功了，因此也开始让 Philippe Kahn 渐渐地养成了好大喜功、目中无人的态度，这也种下了 Borland 开始走向衰退的因子。

在 Borland 最强盛的时期，当然也就是 Microsoft 最痛恨 Borland 的时候，发生了一个著名的事件和一个著名的虚拟人物。由于当时 Microsoft 的开发工具一直打不过 Borland 的产品，因此在 Microsoft 的开发工具刊物上便出现了一个作者，不断地以文章嘲笑 Borland，这个作者的笔名是 Buck Forland。由于这位作者的文章内容以及他的笔名引起了当时 Borland 的不满以及大量 Borland 使用者的强烈抗议，稍后这位作者突然消失。因此有许多人推测这个作者应该是 Microsoft 的某位工程师，由于一直无法打败 Borland 的产品，恼羞成怒，因此才会以这个笔名来发泄。如果各位读者到现在还摸不着头脑，不知道为什么这个笔名会引起轩然大波，那么请试着把 Buck Foland 这两个英文字的第一个字母一对调就知道为什么了。现在各位是否会心一笑了？

■ 在 Borland C/C++3.1 大获成功之后，Borland 却开始松懈了，并且开始走下坡路。当然这有许多的原因，我所知的其中最重要的原因有数项：Philippe Kahn 和当时 Borland C/C++的产品经理闹翻了。这位 Borland C/C++的产品经理的名字是 Eugene Wang，Eugene 是一位非常聪明的越南人。他一手把 Borland C/C++带到了世界第一的地位，并且在 Borland C/C++3.1 成功之后有了更伟大的想法，那就是想在下一个 Borland C/C++版本中完整地以 OWL 封装所有的 Windows APl。因为 OWL 1.0 虽然比 MFC 1.0 来得优秀，但是 OWL 的隐忧就是尚未完整封装所有 Windows 的 APl。此外 Eugene 还计划以 OWL 为核心，开发一个类似今日 Borland C/C++ Builder 以可视化组件为开发方式的开发工具。请各位读者想一想，如果在当时 Borland 能够开发出这种 C/C++开发工具，那将会是一个多么可怕的产品，稍后 Microsoft 的 Visual C/C++1.0 只是能够在集成开发环境中自动产生 MFC 的程序代码就立刻轰动了 C/C++市场，造成了大量程序员转入 Microsoft 的阵营。而且，即使是目前的 Borland C/C++ Builder，使用的 Framework 仍然是以 Object Pascal 为核心的组件 Framework，而不是纯粹的 C／C++程序代码。如果当时 Eugene 能够做出他心中的下一版 Borland C/C++，那么我想，到现在 Borland C／C++可能还是市场中第一的 C/C++开发工具。

不过很不幸的是，Eugene 稍后和 Philippe Kahn 发生了激烈的争执。一气之下，Eugene 离开了 Borland。而 Philippe Kahn 则认为 Borland C/C++的地位已不可动摇，因此也没有想立刻开发下一版的 Borland C/C++。这样一拖竟然浪费了将近 2 年的时间，更大的麻烦是 Microsoft 可没有白白浪费这 2 年的时间。Microsoft Visual C/C++1.0 在 Borland C/C++3.1 发布两年之后推出，并且立刻获得市场好评。Visual C/C++不但在编译器方面能够和 Borland C/C++3.1 相抗衡，在集成开发环境方面更大幅领先了 Borland C/C++3.1，还能够自动产生 MFC 的程序代码，再也不是昔日的吴下阿蒙。直到此时，Philippe Kahn 才从梦中惊醒而急于开发下一代的 Borland C/C++4．0。但此时为时已晚，C/C++的开发工具已经发生了剧烈的变化，Borland 的 C/C++开发工具市场从此就开始逐渐地被 Microsoft 蚕食了。

Eugene 在离开 Borland 之后，立刻被 Symantec 所网罗，稍后 Eugene 也在非常短的时间

之内为 Symantec 开发出了著名的 Symantec C/C++。Symantec C/C++在当时被所有的技术刊物评比为拥有最棒的集成开发环境和最有创意的 C/C++开发工具，由此可见 Eugene 的功力。不过 Symantec C/C++稍后也终究不敌 Microsoft Visual C／C++，这个故事的原因在稍后"四大 C/C++ 编译器之争"的章节中再详细地说明。最后听说 Eugene 跑去做生意了，并且在前几年写了一本教导科技人员如何面试的书籍。一直很痛心 Borland 失去了这么一位优秀的人材。我常常想，如果当初 Eugene 没有离开 Borland，那么历史可能就不是现在的这样了，Sign!!!

■ Philippe Kahn 大手笔地花了 400 多 Million 美金买下了 Ashton-Tate 公司和 dbase。当时许多人都批评 Philippe Kahn 做了不值当的事情，因为 Ashton-Tate 不值这么多钱。但是由于当时 Borland 多的是现金，因此 Philippe Kahn 也不在意。不过 Borland 逐渐走向衰败的主因并不在此，而是在 Borland 买下了 dbase 之后，并没有立刻积极地开发 dbase For Windows，反而把 dbase 丢在一旁。Philippe Kahn 会如此做的原因便是当时 Borland 的另外一个和数据库有关的产品 Paradox 卖得也很好，因此 Philippe Kahn 并不急于开发 dbase For Windows。不过 Philippe Kahn 忘记了一件事情，那就是当时市场上拥有大量使用者数目的 dbase 程序员需要一个好的 Windows 版 dbase，但是 Philippe Kahn 购买了 dbase 却不提供 Windows 版的解决方案，因此当稍后 Microsoft 以极小的代价买下 Fox 这家公司，并且在数年之后推出 FoxBASE For Windows，吸引了大量原先的 dbase 程序员以及 Paradox 的程序员之后，Philippe Kahn 才警觉事情不对而匆匆忙忙地开发 dbase For Windows。但是当 dbase For Windows 推出之后，Microsoft 早已推出了两个 FoxBASE For Windows 的版本，占据了大部分的市场，dbase For Windows 其势已不可为了。

■ Microsoft 开始向 Borland 挖角。由于 Microsoft 在许多的开发工具战役中一直被 Borland 打得灰头土脸，更何况 Borland C/C++3.1 几乎抢占了大部分的市场，因此 Microsoft 便开始准备好好地对付 Borland。但是由于其时 Borland 在编译器的技术领域领先了 Microsoft 数年之久，Microsoft 无法在短时间之内赶上 Borland，所以 Microsoft 决定使用最有效的方法立刻追上 Borland 的技术，那就是直接从 Borland 挖角。结果，后来 Microsoft 的 Visual C/C++小组有 60％的成员是从 Borland 挖来的，这个举动不但立刻让 Borland 流失了大量的优秀技术人才，也在数年之后造成了 Borland 控告 Microsoft 的导火线。各位读者看到这里是否有什么感觉呢？不过我总觉得 Microsoft 并不是光明正大地击败 Borland，而是使用了不公平的竞争手段。

Philippe Kahn 在这段时间不但让 Borland C/C++被 Microsoft Visual C/C++反败为胜，也痛失了几乎所有 dbase 的市场，更浪费了大量的金钱，流失了大量的优秀人员。在这些重要的因素之下，Borland 已经不可避免地开始走下坡了。

我最后一次看到 Philippe Kahn，是在 1994 年末于亚特兰大(Atlanta)参加国际 Conference 时，还和他打了一声招呼。后来 Philippe Kahn 离开了 Borland，另外创立了 StarFish 这家公司，稍后 StarFish 也被 Motorola 并购。虽然 Borland 由于 Philippe Kahn 一些错误的决策而逐渐地从巅峰开始走下坡，但是 Philippe Kahn 也不愧为一个人物。因为 Philippe Kahn 能够和 Bill Gates 一直周旋数年之久，而同一时期的许多公司(例如 Lotus)都一一被 Microsoft 所击败，因此 Philippe Kahn 还是有一套的。此外 Philippe Kahn 也是唯一一个拥有工程师特性的 Borland CEO，Philippe Kahn 仍然重视技术产品和技术人员。但是 Borland 随后的 CEO 几乎都是 Marketing、Finance 或是 Sales 出身的人，这真让我怀念以往以产品和技术为优先的 CEO 了。

看完了上面这段令人伤心的历史，再让我们看看当 Borland 受到 Microsoft Visual C/C++的强大冲击之后，如何思索反击之道。在这段历史中出现了令我敬佩的第一个

Borland 技术工程师 Carl Quinn。
Carl Quinn 在 Microsoft Visual C/C++1．0 推出之后，立刻奉命开发一个能够和 MFC 相抗衡的全新 OWL，而 Carl Quinn 也是数年后 JBuilder 的 JBCL Framework 的灵魂开发人物。Carl Quinn 不但负责开发 OWL，也为 Borland 在组件 Framework 的技术领域做出了重要的贡献。由于 Carl Quinn 的投入，开启了 OWL 大战 MFC、Borland C/C++缠斗 Visual C/C++数年精彩好戏的序幕。
Carl Quinn 是我至今还记得并敬佩的人物，让我再一次的向他致敬，并且介绍他让大家认识。
火线全开
Borland 在开发工具市场和 Microsoft 激战之时，Microsoft 和 Lotus 也正在电子表格工具以及文字处理工具市场进行大战。这时 Borland 不思好好地集中资源开发新的开发工具和数据库工具(稍后本书会详细说明 Borland 在数据库市场的战役)，也不甘寂寞地投入了大量的资源进入这个惨烈的市场。也许是当时 Borland 太有钱了，或者是 Philippe Kahn 的脑袋出了问题，居然决定进入这个 Borland 陌生的市场，更何况在 Borland 投入时 Lotus 已现败象，Office 市场已经慢慢地被 Microsoft 所一步一步地掌握了。
Borland 进入 Office 市场的第一个产品是著名的 Quattro Pro 电子表格。虽然 Quattro Pro 是一个相当不错的产品，而且当时，由 Borland C/C++编译器所开发的 Quattro Pro 在执行效率上几乎是最好的，但是 Borland 没有想到使用电子表格的使用者是一般的办公室人员，这些人注重的是方便性和功能性，而不是执行速度，这和开发人员是不一样的。Borland 以开发者的心态来开发电子表格工具基本上是走错了方向。因此我记得在那段时间中，杂志评比 Microsoft 的 Excel、Lotus 的 1-2-3 和 Borland 的 Quattro Pro 时，在功能方面领先的都是 Excel 和 Lotus，在执行效率方面领先的则是 Excel 和 Quattro Pro。到了电子表格热战的末期，1-2-3 甚至比不上 Quattro Pro，因此 Lotus 败走电子表格市场已是不可避免的结果了。
不过 Borland 虽然赢了 1-2-3，但是和 Excel 仍然有一大段的距离，Microsoft 一统电子表格江山之势已不可动摇，因此最后 Borland 在损失了大量的资源之后，Quattro Pro 只能卖给 Novell。
除了 Quattro Pro 之外，Borland 也投入了很多的资源秘密地开发一个代号为 Spring 的文字处理程序(Word Processor)准备和 Microsoft 的 Word 以及 WordPerfect 竞争，这可能是许多人不知道的。但是这个产品最后仍然无法问市而胎死腹中，在文字处理市场 Borland 不但浪费了时间，更虚掷了大量的资源。
Philippe Kahn 在 Office 产品方面消耗了 Borland 大量的金钱和时间，却落得铩羽而归，更连累了开发工具市场以及最有可能成功的数据库产品市场。
另外一个和 Borland 无关的故事是关于 Microsoft Excel 是如何兴起的。话说当 Lotus 1-2-3 最盛的时期，Microsoft 一直在觊觎这个市场，但是苦于无法开发出一个能够和 1-2-3 相竞争的产品。有一次 Lotus 举办了一个 Lotus 1-2-3 的技术研讨会，由当时 Lotus 1-2-3 的首席工程师主讲。Microsoft 知道了这个技术研讨会之后，立刻派出了最好的程序设计师，现场询问 Lotus 是如何开发 1-2-3 的，并且趁机询问这位首席工程师如何克服 1-2-3 在许多技术方面的难点，而这些困难处正是 Microsoft 的工程师无法克服的。
当时，在现场中的 Lotus 首席工程师虽然知道这些人是 Microsoft 派来的，而且询问的问题正是 1-2-3 许多关键的技术点。但是这位首席工程师凭借着多年的开发经验，认为 Microsoft 不可能在短期之内追上 1-2-3，因此就没有多作保留地回答了许多重要的

问题。没有想 Microsoft 的这些程序员也是非常聪明的人才，一经指点之后，立刻畅然全通，在短短的 1、2 个版本之后不但马上追上了 1-2-3，许多功能方面更是青出于蓝，1-2-3 便逐渐失去优势了。我想这位 1-2-3 的首席工程师一定很后悔当时回答了关键的技术问题吧。

结论：千万不要小看 Microsoft，他们是非常精于模仿的。也永远不要小看你的竞争对手。

数据库市场的失误

Borland 全盛的时期，事实上也是开发数据库产品最好的机会。因为在当时 Borland 手握 DOS 最畅销的 Paradox，并购了 Ashton-Tate 而拥有世界大部分 dbase 的市场，又取得了 Ashton-Tate 从 HP 购买的真正关系数据库(RDBMS)--InterBase，可以说是当时全世界数据库工具实力最雄厚的厂商。

当时的 Oracle 和 Borland 比起来，简直是小巫见大巫，而 Sybase 更不知道在哪里。如果 Borland 能够好好地掌握这个机会，极力开发数据库产品，那么现在 Borland 就算不是世界第一的软件公司，也将是世界第二的软件厂商。可惜 Philippe Kahn 并没有看到这个从 80 年代末到 90 年代成长最快速的产品市场。说句笑话，如果当时 Philippe Kahn 的死对头 Bill Gates 早一点说出"Information At Your Finger-Tip"这句话，点醒 Philippe Kahn 数据库市场的重要性，那么 Borland 就可能是现在的 Oracle 了。

说到数据库市场，就不得不对 Microsoft 的眼光佩服，也不得不佩服 Microsoft 行销能力的强悍。当 Microsoft 以 FoxBASE For Windows 强占了 Windows 开发者的数据库工具市场之后，又了解到一般计算机使用者也需要使用简易好用的数据库管理工具，因此开发出了更简易的 Access。但是当时在类似的市场中，Borland 的 Paradox 占有开发者数据库大部分的江山，而一般使用者的数据库管理工具市场则由 Lotus 的 Approach 博得先机。

Microsoft 为了进入由 Lotus Approach 主宰的市场，采取了很多方法。我还记得在当时 Visual Basic 3 的软件包中 Microsoft 附了一张优惠卷，只要 800 新台币就可以买一套 Access。这简直就是流血大拍卖。不过它的目标很明显，就是击败当时卖 1 万多元的 Lotus Approach。果然，Microsoft 此招一出，Approach 便被 Access 打得落花流水，很快失去了市场，也很快地退出了市场。从此一般使用者的数据库管理工具市场便由 Access 所独占。

但是 Borland 并没有警觉到 Access 会继续往开发者市场进攻，因此仍然没有加紧在 Paradox 产品上的开发。Borland 总觉得 Paradox 的市场地位是无法轻易撼动的，而且 Access 的目标市场也不是 Paradox 的市场。

但是 Borland 忘记了 Microsoft 非常擅长模仿。在随后的 Access 版本中，Microsoft 不断地加入可程序设计的功能，因此也逐渐地吸引了一些 Paradox 入门使用者的市场。再加上 FoxPro For Windows 又持续地强攻开发者数据库市场，Paradox 终于在腹背受敌之下逐渐败下阵来。虽然在末期 Philippe Kahn 对 Paradox 投下重兵，希望能够挽回劣势。奈何时不我予，Paradox 在奋斗了 Paradox 6 和 Paradox 7 的 2 个版本之后，终究难逃失败的命运。

当时在看到 Microsoft 如何打击竞争对手时，我就和朋友开玩笑说，Microsoft 有天下无敌的三大绝招，那就是"打不过你就模仿你(这让我想起电影秘密客)。再打不过就和你比流血，看谁流得久(这让我想起吸血鬼)。最后如果再不行的话，那就挖光你的人(这让我想起电影 Other People's Money)"。Lotus 就在 Microsoft 的前两个绝招下倒地不起，而 Borland 还算是功力深厚，连中三大绝招，虽然不像 Lotus 和许多其他公司一样从此 Bye-Bye，但也是受伤极重的了。

ODBC 和 IDAPI 之争

当 Microsoft 逐渐地击败竞争对手、并且拥有了大部分 PC 数据库市场之后，便慢慢地了解到掌握标准的重要性。此外，Microsoft 为了统一各应用程序之间不同数据的存取，开始制定存取数据的统一标准--ODBC。Microsoft 更大的目的是为了准备和瞄准下一场的大战，那就是 PC 上的关系数据库产品的市场。

当然，Microsoft 要一统数据存取的江山，除了 Borland 不会同意之外，其时一心想从 Microsoft 扳回一城的 IBM 也不同意。而 Novell 更是害怕，因为 Novell 怕 Microsoft 成功之后，Netware 会消失得更快。于是 IBM、Novell 和 Borland 以及一些其他的小厂便聚集在一起，决定也制定一套存取数据的标准接口来和 Microsoft 对抗，这个制定的数据存取标准便是 IDAPI。这正式揭开了 ODBC 和 IDAPI 竞争的序幕。

不过 IBM、Novell 和 Borland 的结合很快就被证明是失败的，因为就像稍后说明的一样，IBM 在 PC 软件上的开发一直是三心二意，反反复复。因此当 IDAPI 1.0 的规格出来之后，IBM 这位老兄又失去了和 Microsoft 对抗的兴趣，于是退出了 IDAPI 联盟。至于 Novell 就更不用说了。Novell 对于和 Microsoft 竞争一向是"说说可以，真打不行"，一定要找到一群厂商才敢和 Microsoft 对抗。Novell 眼看 IBM 退出之后，也马上不战而降，很快地就也退出 IDAPI 联盟，这个现象和稍后 Novell 对于和 Borland 秘密合作的 Appware/AppBuilder 计划如出一辙，都是虎头蛇尾，草草收场。

在两个大同盟临阵脱逃之后，Philippe Kahn 仍然不畏惧 Microsoft 的竞争，还是以 IDAPI 1.0 的规格实现数据存取引擎，这就是我们现在使用的 BDE/IDAPI 和 SQL Links 的前身。当时 IDAPI 1.0 的功能规格比 ODBC 1.0 好得多了。我记得 Delphi 1.0 使用的 BDE/IDAPI 和 SQL Links 驱动程序也比当时慢得像乌龟的 ODBC 快得太多了。只可惜在 IBM 和 Novell 退出之后，其他的小厂也是一哄而散。因此 Borland 只能靠自己独自和 Microsoft 对抗。Borland 能够以少量的资源一直对抗到 Delphi 3 的 BDE/IDAPI 才逐渐地被 ODBC 追过，也算是非战之罪了，怪就只能怪 Borland 意志不坚的盟友们。

当然，由于 IBM 和 Novell 的行事作风如此，所以在稍后许多能够和 Microsoft 一较长短的机会也因为如此而消逝，最后自食恶果，逐渐失去了 PC 的软件市场，再也无力和 Microsoft 抗衡了。

^v^v^v^v^v^v^v^v^v

第二章　C／C++的圣战

"在惨烈的、大规模的 C/C++战役中，注定只有最强者才能生存下来!"

Borland C／C++的反击

当 Visual C++1.0 在 C/C++开发工具市场获得空前的成功之后，Borland 才从 Borland C/C++3.1 的胜利梦中惊醒，思考如何面对 Visual C++的猛烈攻势。事实上，Borland 如果脑袋清醒一点，好好看清当时 C/C++开发工具的市场，那么 Borland 应该会发现虽然 Visual C++经过两年多的整军经武，实力已经大胜以前。但是，Borland C/C++3.1 在许多方面仍然是可以和 Visual C++一争长短的。首先，当时 Visual C++的最佳化编译器仍然落后 Borland C/C++3.1；第二，MFC 仍然没有完整地封装 Windows API，而且 MFC 是以较低阶的方式封装 Windows API 的，面向对象做得并不好，也不是很容易使用。事实上以我的观点来看，正是因为 MFC 不好用，所以 Visual C++才需要在集成开发环境中提供以可视化方式产生 MFC 程序代码的功能。第三是 Visual C++当时并没有很好的封装数据结构的 Container Class，而 Borland C/C++却有非常好用的 BIDS 类别库。第四，也是最重要的，Borland C/C++3.1 仍然拥有绝大多数的市场，而且几乎所有的外围公用程序，Shareware 等都是使用 Borland C/C++3.1 开发的。因此，如果 Borland 不着急，好好地开发下一代的 C／C++开发工具，即使 Microsoft Visual　C++能够掠

夺一些市场占有率，但是如果下一代的 Borland C/C++能够像 Borland C/C++3.0 一样立刻拉开和 Visual C/C++的距离，那么 Borland 在 C/C++市场仍将拥有王者的地位。

可惜的是，也许是 Philippe Kahn 在和 Microsoft 的 FoxPro For Windows 一役中被吓着了，因此急于在 Visual C/C++1.0 之后立刻推出新的 Borland C/C++以扳回颜面。但是 Philippe Kahn 忘了，在这段时间之内 Borland 失去了许多的人才，Eugene Wang 也离开了。更重要的是在过去近 3 年的时间内，Borland 几乎没有持续地开发下一代的 Borland C/C++，短时间内怎么能够仓促地推出新产品呢？

可是 Philippe Kahn 管不了这么多了。他急忙找来了 Carl Quinn 等人后便要求立刻开发出下一代的 Borland C/C++，于是 Borland C/C++4.0 就在这鸭子赶上架的情况下匆忙地开发了。Borland 在开发 Borland C/C++4.0 时犯了许多的大忌。首先在这么短的时间内 Borland 决定全新升级集成开发环境；第二是把 OWL 完全重写；第三是大幅修改最佳化编译器；第四是整合当时棘手的 VBX，Borland 居然让 16 位和 32 位的 Windows 程序同时使用 16 位的、丑陋的 VBX。

上面所说的每一项都是大工程。Borland 早应该在 Borland C/C++3.1 之后便开始进行这些工作，现在要在短短的一年多时间内重新开发这么复杂的一个 C/C++开发工具，几乎是不可能的。但是在 Philippe Kahn 的强力要求下，这些 Borland 的工程师还是硬着头皮做了出来。

不过我必须很沉痛地说，当时在 Borland C/C++4.0 Beta 测试时，我便和台湾 Borland 的人说，如果 Borland 仓促推出 Borland C/C++4.0 的话，那么不但不会对 Visual C++产生任何的影响，反而是自杀的行为。因为臭虫实在太多了，整个集成开发环境的反应也很缓慢，它的最佳化编译器更是笑话，错误百出，真像当时恶名昭彰的 Microsoft C 4.0 一样。我还开玩笑地说，是不是因为 Microsoft 从 Borland 挖了大量的 Borland C/C++人才，因此好胜的 Philippe Kahn 也还以颜色，从 Microsoft 反挖 Microsoft C 的人，却不幸地挖到了 Microsoft C 4.0 的人。

但是，显然 Borland 并没有听到我或其他 Beta 测试人的心声。在 Visual C++1.0 推出后的 1 年多、推出 Borland C/C++3.1 之后的第 4 年，Borland 终于推出了新一代的 Borland C/C++ 4.0，这个肩负和 Visual C++1.0 对抗的新一代 C/C++开发工具。

在 Borland C/C++4.0 刚推出之际，Borland 确实为 4.0 做了极大的造势，我记得在当时所有重要的计算机杂志中，例如 Byte、PC Magazine、Dr. Dobb's 等，都有 4.0 整页的广告。这个广告的内容是以一个巨大的猫头鹰为主，再搭配蓝色底系的 Borland C/C++4.0，选用巨大的猫头鹰当然是因为 OWL 的原因，只可惜我现在找不到那幅广告的画面了。

当时 Borland C/C++4.0 使用了如下的广告用词：

Visual Is Only A Facial Facade

来讽刺 Visual C/C++只提供了产生 MFC 程序代码的基本精灵，而 Borland 除了提供相对应的 AppExpert 精灵(能够提供类似的功能，以产生使用者选择的 OWL 程序代码)之外，Borland C/C++4.0 的集成开发环境还提供了可视化的三面版窗口，能够让程序员完整地掌握整个项目的情形。

下图便是当初令人眼睛为之一亮的 AppExpert：

下图则是当时 Borland C/C++的注册商标，三面版窗口开发环境。看到此图又令我想起当初使用 C/C++撰写程序的日子，下方程序页面清楚地显示了我 1995 年在鼎新工作时写的智能型 Windows 排程系统，时间过得真快啊。

当时 Borland C/C++4.0 的三面版集成开发环境真正开创了一个新的局面，因为这个集成开发环境允许程序员知道每一个应用程序定义的窗口信息，并且能够立刻把它显示

在下方的程序代码窗口中，的确是非常的方便，也比当时 Visual C/C++的集成开发环境来得先进。再加上 Borland 较为先进的编译器技术和架构更好的 C/C++ Framework-OWL，照理说 Borland C/C++4.0 应该会获得极大的胜利，可为什么最后会以失败收场呢？

没错，在 Borland C/C++4.0 刚推出之际，订单的确如雪片般飞来，销售情形非常好。这毕竟是 Borland 在久违了数年之后的大作，许多 Borland 的用户都迫不及待地升级，当初我也是拼命地要求台湾 Borland 第一个给我 Borland C/C++4.0。但是在推出一段时间之后，市场的反应就急速地冷却下来，因为各种负面的批评不断涌现。这主要的原因当然是因为 Borland C/C++4.0 的品质实在不好，就像前面我在 Beta 测试时说的，由于 Borland 太急于推出 4.0，因此并没有在最后阶段修正许多的臭虫，又没有经过最后系统微调的工作，同时又过于大胆地加入太多先进的技术，造成了整个产品的不稳定，而犯下了大错。下面几点应该是造成当初 Borland C/C++4.0 惨遭滑铁卢的主要原因：

集成开发环境方面：臭虫太多，容易当掉而且反应速度缓慢

编译器方面：最佳化玩得过火，产生错误的编译程序代码

OWL 方面：采用全新的多重继承架构，虽然是正确的做法，却和 Borland C/C++3.1 中的 OWL 不兼容，造成许多程序员无法升级 C/C++项目

VBX 方面：大胆的采用在 16/32 位都能使用 VBX 的技术，造成一些 VBX 无法顺利地在 Borland

C/C++4.0 中使用

我想其中最可惜的就是 OWL 了。OWL 2．0 在各方面都有一流的表现，实在是 MFC 强劲的

竞争对手，获得了各方一致的肯定和称赞。无奈的是，由于 OWL 2.0 做了基本架构的改变，这虽然是为了解决当初 OWL l.x 使用了不标准的 C/C++编译器技术的问题，但是这造成了原来 Borland C/C++3.x 程序员极大的困扰，因为升级不易。对于新的 C/C++使用者来说，又因为 Borland C/C++4.0 本身不稳定的因素而却步，因此造成了 OWL 2.0 叫好不叫座的下场，真是可惜了 OWL 小组的努力。

还记得当时我的项目使用了 FarPoint 的 SpreadSheet VBX 组件，由于一直无法顺利地在 Borland C/C++4.0 中使用，并且会造成应用程序的当机，最后追踪执行程序代码却发现应该是 Borland C/C++4.0 的问题，因此最后只好在咒骂中放弃使用 Borland C/C++4.0，而回到 Borland C/C++3.1。当时想，对于我这个长期使用 Borland 产品的人都无法忍受 4.0 的品质，其他的程序员又怎能使用这个产品呢？我想这就是为什么后来 4.0 全面溃败的原因，因为 Borland 推出了根本不堪使用的产品。

我在 Borland 工作时，有一次在新加坡和现任 Borland 开发者关系部门副总裁的 David Intersimone 谈起这一段往事，David 也很感慨，他直呼"We screwed it up!(我们把事情搞砸了)"，"It's a mess(那实在是一团混乱)"。David 还说当时整个 Borland C/C++开发小组都很混乱，和以往 Borland C/C++3.0/3.1 的开发小组比起来实在是差太多了。除了因为一些重要的人物相继离开 Borland 以及 Microsoft 也挖走一大票人之外，与 Philippe Kahn 的直接介入，造成人事不和也有很大的原因。

在 Borland C/C++4.0 快速失利之后，Borland 也认识到问题的严重性，因此立刻着手开发 Borland C/C++ 4.0 的 Patch，当时是称为 Service Pack。但是在稍后的 4.01 版中并没有完全解决问题，一直到 4.02 才稍微解决一些严重的问题。无奈时不我予，拖的时间太长，市场已经起了巨大的变化。

Borland C/C++4.0 失败之后，立刻造成了严重的后果。首先是 Borland C/C++的市场

大量而且快速地流失，使得 Visual C / C++快速地成长。第二点是当初 Borland C/C++ 3.1 在公用程序市场打下的江山也拱手让人，原本许多使用 Borland C/C++3.0/3.1 撰写驱动程序的硬件厂商也开始转换到 Visual C/C++。而更严重的是，由于 4.0 的品质以及稍后 OLE 的关系，应用程序市场也开始大量地转为使用 Visual C/C++来编写应用程序。

此时，Borland 在三个主要的应用市场接连败退，C/C++的江山注定将易主，其颓势已不可挽回。

Borland C/C++、Visual C/C++、Watcom C/C++和 Symantec C/C++的缠斗

自 Borland C/C++4.0 一役大败之后，Borland 在 C/C++市场上建筑的巨大堡垒似乎再也不是牢不可破了。Visual C/C++固然在不断地接收 Borland C/C++失去的市场，这时在 C/C++市场上也开始出现另外两个坚强的对手，那就是 Symantec C/C++和 Watcom C/C++。

Symantec C/C++的发展史

Symantec C/C++和 Watcom C/C++这两个对手的来头都不小。先说 Symantec C/C++吧，它的 Think C/C++在 Macintosh 上便是非常有名的编译器，因此早在 C/C++领域便有深厚的基础。在 Symantec 并购了 PC 上第一个 C/C++编译器 Zortech C/C++之后，Symantec 进入 PC 的开发工具市场也是箭在弦上了，只可惜的是，其时 Symantec 还未找到一个在 PC 上有丰富经验的开发工具领导者。

也许是上天注定要引起稍后的 C/C++编译器大战吧，此时 Borland C/C++3.1 的幕后支柱 Eugene Wang 刚好和 Philippe Kahn 闹翻，离开了 Borland。Symantec 眼见机不可失，立刻重金招揽 Eugene Wang 到 Symantec，为 Symantec 推出第一个 Windows 上的 C/C++开发工具。1993 年左右，在 Eugene Wang 的掌舵之下，Symantec 推出了第一个 Symantec C/C++版本，立刻便获得了市场的好评。自此之后 Symantec C/C++军心大振，不断地继续改善，也逐渐获得了不小的 C/C++市场，俨然成为可以对抗 Borland C/C++、Visual C/C++的另一山头。当时 Symantec C/C++是以最华丽、先进的集成开发环境获得了市场的高度认同，在 C/C++编译器最佳化方面的表现也不输给其他的编译器。

当时我正为《RUN!PC》撰写有关 C/C++的文章，因此 Symantec 台湾分公司的人也和我联络过，并且送给我一套最高档的 Symantec C/C++版本，希望我除了为 Borland 写 C/C++的文章之外，也能够为 Symantec C/C++写一些东西。我还记得，在当时安装 Symantec C/C++之后，我的确被它的集成开发环境吸引得说不出话来，因为实在是太棒了。Borland C/C++和 Visual C/C++的集成开发环境同 Symantec C/C++的集成开发环境比较起来，立刻变成索然无味、平淡无奇了。即使到现在，我仍然必须竖起大拇指对 Symantec C/C++的集成开发环境说声"赞"。我想 Eugene Wang 在这么短的时间内把 Symantec C/C++打造得如此之好，除了证明他的不凡功力之外，也有向 Philippe Kahn 示威、证明 Philippe Kahn 让他离开 Borland 是错误决定的意思。我之所以如此说，是因为其时 Symantec C/C++最喜欢点名挑战的对象便是 Borland C/C++。

就我的感觉而言，Symantec C/C++就像是一个技艺精良、又装备华丽的 C/C++军团。

Watcom C/C++的发展史

非常有趣的是，Watcom C/C++走的路子和 Symantec C/C++几乎是完全相反的。当时出品 Watcom C/C++编译器的是一家加拿大的小公司，不过这家公司却对最佳化编译器有深入的研究。当时，Watcom C/C++是以在 DOS 下能够产生最好的最佳化程序代码闻名于世的，许多写游戏和 DOS Extender 的厂商都指名要使用 Watcom C/C++，因为不论是 Borland C/C++还是 Visual C/C++，它们产生的最佳化程序代码都比 Watcom C/C++的最佳化程序代码差上一截。再加上当时最有名的 DOS Extender 厂商 PharLap 公司也是

使用 Watcom C/C++，因此 Watcom C/C++在专业的 C/C++程序员以及系统程序员心中是第一品牌的 C/C++开发工具。

不知道还有多少读者记得 PharLap 这家公司，或是有没有读者记得 Andrew Schulman 这位伟大的软件技术人员。当时 Andrew Schulman 的 Undocumented Windows 一书红遍了半边天，也惹得 Microsoft 要告 Andrew Schulman。而 Andrew Schulman 便是 PharLap 公司的首席工程师，也是当时最著名的"The ANDREW SCHULMAN Programming Series"的总监。而 PharLap 公司是当时出版 DOS Extender 软件最成功的软件公司。

当时由 Matt Pietrek 撰写的 Windows Internals 也是轰动一时的巨著。谈到 Matt Pietrek，熟悉 Windows Programming 的读者应该很少有不知这位大师级人物的。Matt 长期在 Microsoft System Journal 撰写 Under The Hood 专栏，专门写一些深入系统的程序设计技术，在数年前便和 Andrew Schulman、David Maxey 成为 Windows System Programming 的三大巨头之一。Matt 也是著名的 Windows 除错工具 SoftIce、BoundsChecker 的主要研发工程师。Matt 本身是从 Borland 出道的，他初至 Borland 工作时便是在 Turbo Debugger 小组中研发除错工具。当时 Borland 的 Turbo Debugger 是 DOS 下最强的除错工具，即使是 Microsoft 也无法推出能够和 Turbo Debugger 抗衡的除错工具。Matt 在这个小组中吸收了大量的知识，并且快速成为这个领域的专家。后来 Turbo Debugger 小组的部分成员被 Microsoft 挖走，让 Microsoft 掌握了 Borland 的核心除错技术，以致后来也能够推出不错的除错工具。而 Matt 也出走到 NuMega 公司，成为开发 SoftIce、Bounds Checker 的关键人物。

写到这里还是不禁要佩服 Borland，因为当今许多名满天下的重量级软件工程师都是由 Borland 培养出来的。

Watcom C/C++在 DOS 市场站稳了脚跟之后，由于 Windows 已经逐渐成为市场的主流，DOS 势必将被逐渐淘汰出局，因此，Watcom C/C++如果要继续生存下去，也就-定要推出 Windows 平台的 C/C++开发工具。大约是在 1993、1994 年左右，Watcom 终于推出第一个 Windows 下的 C/C++开发工具。

不过，当时 Watcom C/C++在 Windows 推出的 C/C++开发工具实在是平淡无奇。其集成开发环境和另外三个对手比较起来简直像是远古的产品，-点特色都没有。不过 Watcom C/C++仍然是以它的最佳化编译器作为号召。因此当时发生了一个非常有趣的现象，那就是许多软件公司会同时买 Borland C/C++，或是 Visual C/C++，Symantec C/C++之一，再搭配一套 Watcom C/C++。在开发应用系统时使用其他三套开发工具之一，最后要出货时再使用 Watcom C/C++来编译以产生最佳的程序代码。

在 Watcom C/C++推出了 Windows 平台的开发工具之后，也吸引了-群使用者。虽然 Watcom C/C++的市场比起其他的三家来说是最小的，但是总算撑起了一片天，成为四大 C/C++开发工具之一。稍后 Watcom C/C++被 Sybase 并购，成为 Sybase Optima++的前身。

对我的感觉而言，Watcom C/C++就像是一个穿着朴素、但是却拥有最佳训练的白色 C/C++军团。

关键的时刻--MFC Or Not

在 Symantec C/C++和 Watcom C/C++逐渐站稳了脚跟之后，C/C++四大编译器决战的时刻也逐渐逼近了，一些其他出产 C/C++工具的软件公司早已自动退出了这个在当时竞争最为激烈的软件市场。在 1994 年末的决战之前，Symantec 和 Watcom 同时面对了一个非常严厉的考验，那就是 C/C++ Framework 的选择。

虽然 Symantec 和 Watcom 都以各自的特色占得了一定的市场，不过在当时对于一个 C/C++开发工具来说，最重要的功能之一就是 C/C++ Framework。因此 Symantec 和 Watcom 也都必须为使用者提供一套 C/C++ Framework。不过这对于 Symantec 和 Watcom 来说都

是一个难以解决的问题，因为当时的 C/C++ Framework 已由 Borland 的 OWL 和 Microsoft 的 MFC 所占领，虽然市场上也存在一些跨平台的 C/C++ Framework，例如 ZApp 和 Zinc 等，但是这些 C/C++ Framework 终究没有产生很大的影响。如果 Symantec 和 Watcom 要自己发展新的 C/C++ Framework，那他们还没有如此雄厚的资源，也无法在短时间之内完成。因此 Symantec 和 Watcom 必须决定，到底是要使用 Microsoft 的 MFC 还是使用 Borland 的 OWL 来作为他们开发工具的 C/C++ Framework。

1993 年初，Symantec 和 Watcom 分别和 Microsoft 签约授权使用 MFC 作为他们的开发工具的 C/C++ Framework，至此大局已定，在 C/C++ Framework 的市场已经形成三家夹击一家的形势。当时许多人便预测 Borland 会成为输家，因为市场已经成为一面倒的现象，MFC 看起来已经是胜券在握了。当时，Borland 的内部也展开了激烈的辩论，讨论是否也要授权使用 MFC 作为 C/C++的 Framework，停止继续开发 OWL。不过，后来 Borland 还是决定继续开发 OWL，而不使用 MFC，因为 Borland 的 C/C++技术小组认为 MFC 不论是

在架构上或是设计上都比不上 OWL。而且，由于当时 Visual C/C++对于 C/C++标准的支持不如 Borland C/C++，所以在 MFC 内部使用了大量的 Macro 以及不标准的语法，因此如果 Borland C/C++要使用 MFC，那么还需要修改 Borland 的 C/C++编译器来编译 MFC。对于这一点，我认为 Borland 是做了一个正确的决定。因为，如果当时 Borland 也授权使用 MFC，那么不但在气势上输了一截，而且，由于 MFC 的发展完全掌握在 Microsoft 的手里，采用 MFC 就等于脖子被掐在别人的手里，动弹不得。可惜的是 Symantec 和 Watcom 并没有看清这一点，以为有了和 Microsoft 一样的 Framework，就可以在其他地方和 Microsoft 以及 Borland 一决雌雄，Symantec 和 Watcom 却没有想到，就是这一点决定让自己在后来的决战中一败涂地，最终完全退出了 PC 的 C/C++开发工具市场。

随着 1994 年末的到来，C/C++开发工具的四大天王决战的日子也终于愈来愈近了。

OLE 的搅局

不知道是时运不济，还是 Microsoft 刻意如此，在 1994 年 Borland C/C++和 Visual C/C++决战的前夕，Microsoft 推出了 OLE(Object Linking And Embedding)技术。OLE 是 Microsoft 为了对抗 Apple 的文件技术以及 IBM OS2 的 Workplace 和文件技术应运而生的。OLE 可以让 Windows 平台的文件内嵌在不同的应用程序中，并且能够让文件在应用程序中被即地编辑(In Place Editing)。说实在的，Microsoft 的 OLE 和 Apple 以及 IBM 的技术比较起来实在是差多了，在稍后也被证明是失败的技术。不过，Microsoft 的 OLE 和 Apple/IBM 的文件技术都是失败的技术，都没有造成巨大的成功。虽然这些文件技术都没有成功，但是 OLE 却足以成为 Borland、Symantec 和 Watcom 失败的重要因素。

我记得当时 OLE 似乎成为了一个令人趋之若鹜的时髦功能。Word 的文件能够内嵌在 Excel 之中，而且使用者可以点选此 Word 文件，应用程序又立刻成为 Word 来编辑它，实在令人觉得非常的神奇。不过，在其时所有的软件厂商中，只有 Microsoft 的应用程序有如此的功能，其他的厂商例如 Lotus、WordPerfect 等都无法实现这种功能。

这明显地造成了不公平的竞争，因为 OLE 技术是由操作系统厂商 Microsoft 推出的，但是却让它的应用程序部门同步拥有这种技术，而其他的软件厂商都无法获得第一手的 OLE 技术来编写应用程序，这也是为什么当时其他的软件厂商如此火大的原因。

虽然后来其他的软件公司在取得了 OLE 的技术资料之后，也推出了具备 OLE 功能的应用程序，但毕竟是慢了 Microsoft 许久，市场也流失了许多。不过，我觉得很奇怪的是，在当时内建 OLE 功能的应用程序之中，几乎所有软件厂商推出的应用程序在激活数个应用程序而且使用 OLE 之后，就非常容易死机，只有 Microsoft 的应用程序不太会发生这种情形，因此许多人便认为 Microsoft 隐瞒了一些技术没有让其他的厂商知道。

由于 OLE 是如此的复杂，因此 Borland 无法立刻在 OWL 之中实现这种功能，于是就造成厂市场上负面的影响。至于 Symantec 和 Watcom 虽然授权使用 MFC，但是在其时它们授权使用的是 MFC 1.x 的版本，Microsoft 并没有把 OLE 实现在 MFC 1.x 中，而是实现在 MFC 2.0 之中。在 MFC 2.0 推出时，它最重要的功能就是 Microsoft 加入了 20000 多行支持 OLE 的程序代码，但是 MFC 2.0 仅限于 Visual C/C++使用，就是这关键的一点让其他三家竞争厂商吃了大亏。

对于 OLE 这个关键技术的影响，Borland 是深知在心的，因此计划在 Borland C/C++ 4.5 的 OWL2.5 中支持 OLE。当时 Borland 推出的对应解决方案便是 OCF(Object Component Framework)。

Borland 当初在设计 OCF 时有几个重大的目标，这些目标包括：第一、如何使 OLE 琐碎、复杂的接口单纯化。第二、如何使 OLE 在窗口环境下写程序的思考方式一致化--即使用"事件驱动"的方式来开发。第三、如何在微软占尽天时、地利(但未必人和)的情况下使 Borland 的产品具备 OLE 的功能。第四、如何让大多数 C/C++的程序员都能够享受 OLE 的功能而不局限于 OWL。由于上述的设计目标，从而造就了典雅而具有弹性的 OCF。由于 OCF 本身是一完整而独立的 Framework，因此它可适用于各种 C/C++ Framework 之中，包含了 OWL、MFC 以及 ZApp/Zinc 等 Framework。

不知道各位使用过 Borland C/C++的朋友们是否还依稀记得下图所示的 OCF 架构图之一，以及下面的 OCF 范例程序代码？这些可是我把 1994 年写的文章挖出来之后找到的，真是令我感慨，不禁回想起了当时的情景，在此也让各位回忆一下 OWL 和 OCF。对于不熟悉 OWL 和 OCF 的朋友，也可以从下图和程序代码中观察一下当时的技术以及设计的概念。我现在看这些图形架构，会发现基本上它们并没有落后现在太多，可见当时设计者的功力(当然又是 Carl Quinn 定义的佳作之一)。

程序 1 OWL 的 TOleWindow 支持 OLE 插入对象的成员函数

```
    //
    // Insert an OLE object into the view
    //
    void TOleWindow::CmEditInsertObject()
    {
001   PRECONDITION(OcView);
002   TOcInitInfo initInfo(OcView);
003   if (OcApp->Browse(initInfo)){
004      TRect rect;
005      GetInsertPosition(rect);
006      SetSelection(new TOcPart(*GetOcDoc(), initInfo, rect));
007      OcView->Rename();
008      InvalidatePart(invView);
         }
    }
```

程序 2 OWL 的 TOleWindow 支持左键双击的成员函数

```
    //
    // Handle left double-click message
    //
    void TOleWindow::EvLButtonDblClk(uint modKeys, TPoint& point)
    {
```

```
    PRECONDITION(GetOcDoc() && GetOcView());
    TOleClientDC dc(*this);
    dc.DPtoLP(&point);
    TOcPart* p = GetOcDoc()->GetParts().Locate(point);
    if (modKeys & MK_CONTROL) {
      if (p)
        p->Open(true); //Ctrl key forces open editing
    }
    else {
      SetSelection(p);
      If (p && p == GetOcView()->GetActivePart()){ //resync the active flag
        p->Activate(false);
    }
    GetOcView()->ActivatePart(p); //In-place activation
  }
```

虽然 Borland 及时地在 OWL2.5 中加入了 OLE 的支持，无奈 Microsoft 随后又在 OLE 中加入
了许多其他的功能。因此让 OCF 还是无法完整地支持 OLE 所有的功能，Borland 又无法
不延后 Borland C/C++的推出，因此直到 l994 年末，Borland 才终于推出了决战性的
Borland C/C++4.5 版本。

C/C++开发工具的最后圣战

"虽然已经过去了许久，但是我仍然忘不了那场最惨烈的战役！"

在 1994 年末、1995 初，Borland 痛定思痛，终于清除了 Borland C/C++4.0 中所有的问
题，也开发出了自 Borland C/C++3.1 以来最稳定、最快速的 Borland C/C++4.5，准备
和 Microsoft 决一死战。我记得当时许多有关 Borland C/C++和 Microsoft C/C++的书
籍都是使用十字军的封面。不同的是 Borland C/C++的系列丛书都是以蓝色为色系，
而 Microsoft 的则是以红色为色系，仿佛两大军团终将决战似的。

不过，这次的战役不仅仅是 Borland 的蓝军和 Microsoft 的红军相对抗。在 Symantec 的
华丽军团经过了整军经武，Watcom 的白色劲旅枕戈待旦，而且都从 Microsoft 授权使
用了 MFC 之后，蓝、红、花、白四大军团决战的日子终于来临。

首先，当 Symantec 和 Watcom 分别取得了 MFC 之后，Symantec 便推出了 C/C++ 7.x 的版本，
和 Watcom C/C++混战了起来。两个使用系出同门的 C/C++ Framework 产品战得不亦
乐乎，随后 Borland C/C++ 4.5 和 Visual C/C++的新版本也加入了这场最重要的决战。

但是，让 Symantec 和 Watcom C/C++大吃一惊的是 Microsoft 使用的 MFC 居然比他们使
用的 MFC 高出了一个版本(1.x 对 2.x)，而且新版本的 MFC 包含了完整的 OLE 支持能力。
而 Borland 虽然也有 OCF 这张王牌，但是仍然不敌新版 MFC 中的 OLE 能力。

由于当时几乎所有的应用程序都需要支持 OLE，但是却只有使用 Visual C/C++最新的
版本才能够开发完整 OLE 能力的应用程序，所以不管 OLE 到底有没有用，反正先加入再
说。因此市场上的形势很快就发生了巨大的变化，因为 OLE 的原因，几乎大部分的应
用程序开发者都选择使用 Visual C/C++，Symantec 和 Watcom 军团很快就败下阵来。

至于 Borland C/C++4.5，虽然它是一流的产品，如果没有 OLE 的因素，Visual C/C++
新版本真的并不比 Borland C/C++4.5 好：虽然 4.5 也有 OCF，但是在市场上只有 Borland
和 Novell、WordPerfect 选择使用 OCF。在和 Microsoft 的 Visual C/C++经过将近一年
的缠斗之后，其他大部分的厂商都选择了 Microsoft 的 MFC 2.x 版，真是形势比人强。

OCF 的架构真是个好东西，但却无法完整地支持 OLE，因为 OLE 的发展是掌握在 Microsoft 手中的，因此虽然 OCF 的架构良好，终究在功能上不及对手。Microsoft 结合操作系统、开发工具和应用程序的手段真是无往而不胜。击败 Lotus、Borland 是如此，歼灭 Netscape 亦是如此。

对于 Symantec 和 Watcom 来说，这场战役就如同"长平之战"秦军坑杀 40 多万赵军一样。杀得 Symantec 和 Watcom 全军覆没，大败而归。至此 Symantec 弃守 PC 的 C/C++开发工具市场，转而开始研发 Java 开发工具，进而在稍后推出了著名的 Visual Cafe。至于 Eugene Wang，则离开了 Symantec，也离开了 PC 开发工具的领域。

而 Watcom 则更为凄惨。整个公司在 DOS 的市场逐渐式微，而 Windows 平台的开发工具又大败而归，两头落空。不久之后，Watcom 便被新兴而起的 Sybase 并购，从此在竞争激烈的开发工具市场中消失了。

归纳 Symantec 和 Watcom 失败的原因，是因为 C/C++的 Framework MFC 掌握在 Microsoft 手中，在决战时刻 Microsoft 居然手握比 Symantec 和 Watcom 更新的 MFC 利器，而且在 Visual C/C++精进最佳化编译器技术并且改善集成开发环境之后，Symantec 和 Watcom 诉求的重点功能完全被 Microsoft 封死。因此在产品、技术、市场和气势上完全不如对手的情形下，自然只能任人宰割了。

对于 Borland，虽然没有像 Symantec 和 Watcom 那么溃不成军，但也再次败下阵来。虽然平心而论 Borland C/C++4.5 的确是一个非常好的产品，无论在 OWL、最佳化编译器、集成开发环境方面都有一流的表现。和 Borland C/C++4.0 比较起来简直犹如脱胎换骨一般，到现在 Borland C/C++4.5 仍然是我最喜欢的版本之一。但是无奈当初 Borland C/C++4.0 给人挥之不去的负面印象，以及无法完整支持当时如火如荼的 OLE 技术，因此还是在决战之中败了下来。好在蓝色的 Borland 大军毕竟是训练有素，虽然自此让 Microsoft 占据了超过 50％的市场，成为 C/C++开发工具的老大，但是 Borland 仍然掌握了超过 30％的市场，稍做喘息，并且支撑 Borland 在各重要战役失败之后维持公司的运作，等待 Delphi 的浴火重生，再重新出发。

经过这关键的一役之后，Microsoft 终于清除了大部分的对手。对于 Microsoft，程序语言开发工具的战争已经结束，这个市场注定将被 Microsoft 占据大部分的市场。在 Microsoft 手握操作系统、Office 软件和开发工具三大获利市场之后，Microsoft 也开始将矛头对准下两个竞争目标：关系数据库以及主从架构开发工具。在 Microsoft 正式进军这两个市场之后，当然也展开了连番的好戏，尤其是在主从架构开发工具方面又开启了 VB、PowerBuilder、Gupta/Centura 和 Delphi 的惊天动地大会战。另外一个意外开启的战争则是 Microsoft 在 1995 年和 Netscape 的浏览器大战。

在 C / C++最后一役之后，我认为开发工具的圣战已然结束，Borland 也正式开始走下坡路。更严重的是我认为自此之后 Borland 不但丧失了 C/C++的江山，也失去了对于 C/C++开发工具的创意，这是我感到最遗憾的地方，到现在为止我仍然认为 Borland 尚未重拾当初在 Borland C/C++3.0/3.1 时代独领 C/C++创意风骚的精神。也许，也许，要看看 C/C++ For Kylix 或是 C++Builder 的后继产品是否能够重新找回这个失去已久的精神，不再让大家失望了。

永不成气候的 C/C++开发工具：IBM VisualAge C/C++

IBM 在 C/C++开发工具市场扮演的角色一直令人啼笑皆非，因为在 C/C++编译器战争最激烈的时刻，IBM 这个全球信息大厂却一直是缺席的。一直到了 1995 年之后，C/C++编译器市场大势已定后才慢慢地加入战局，推出 VisualAge C/C++ 3.0，企图进攻此市场。但是此时市场早已由 Microsoft 的 Visual C/C++称雄。IBM 的 VisualAge 虽然能够以创新的可视化设计家定义对象之间的关系，但是在其他方面却乏善可陈，整个集成

开发环境也缓慢如蜗牛，需要非常高的硬件配置才能够顺利运行，和 Visual C/C++以及 Borland C/C++等工具比较起来就像是恐龙一般，因此几乎没有在市场上引起任何的反应。

在推出的 VisualAge C/C++3.0 并没有在 PC 的 C/C++开发工具市场获得任何的明显成果之后，IBM 又再次集中许多资源，开发下一代 3.5 版本，希望能够在此市场占有一定的比率。我知道 IBM 在 VisualAge 投注了大量的资源，因为从 Beta 版开始台湾的 IBM 便有人和我接触，希望我也在《RUN!PC》上为 VisualAge C/C++3.5 写些文章。因此在 1996 年的 6 月我写了一篇 C/C++编译器的比较文章，下面的资料便是数年前当时还是 Beta 版的 VisualAge 3.5 和其他编译器的比较结果(见下页)。

从图中的数据可以看到，其实 VisualAge C/C++3.5 的表现还不错，只是对于当时还在使用 AMD DX4-100/32M RAM 机器的我来说，实在是跑不动。后来台湾 IBM 负责 VisualAge 的产品经理请我吃饭，在此饭局中这位李经理同时请了贺元(后来成为资迅人的总裁)、薛晓岚(后来成为资迅人的副总裁)以及其他两位作者，希望大家在计算机杂志中继续为 VisualAge C/C++3.5 写写东西，一起 Promote 此产品。在这个饭局中我是第一次和贺元、薛晓岚见面，当时贺元在中文 PC Magazine 有一技术专栏。记得当我向这位李经理提起我的机器几乎无法跑得动 VisualAge C/C++3.5 时，他还立刻一口答应借我一台当时 IBM 最高档的 PC。同时每写一篇 VisualAge C/C++3.5 的文章，除了《RUN!PC》原本的稿费之外，IBM 会再付一字 2.5 元的稿费。乖乖，IBM 真是大手笔。我算算当时我的产能，写一篇文章就能够赚 2 到 3 万，又有免费的最高档机器可用，真是太好了。不过后来我还是觉得 IBM 在此市场可能不会深耕。在不愿意违背自己写作习惯和得罪 Borland 的顾虑下，最后还是没有答应。现在想想当时真是太笨了，放着好赚的稿费不赚，嘻。

IBM 的 C/C++开发工具之所以在市场无法成功，是因为 IBM 并不了解在此竞争激烈的市场中使用者到底要什么。另外一个原因则是 IBM 并不以 PC 上的开发工具软件为重要的事业。即使无法竞争和获利，对于 IBM 来说也没有什么影响，因为 IBM 主要是靠硬件和大型软件为主，不像 Borland 这可是生命之争。因此 IBM 只是兴起玩玩，随即放下。所以我觉得在 PC 平台使用 IBM 的工具是很危险的，因为 IBM 随时都可能会放弃这个市场。不知道现在 VisualAge C/C++到底下场如何?是不是还在 3.5 或是 4.0 版？IBM 已经数年没有任何的维护和改善了。

快速殒落的潜力之星：Sybase 的 C/C++RAD 工具 Optima++

1996 年左右，Sybase 并购了 Watcom 之后终于推出了石破天惊的 C/C++开发工具：Optima++。Optima++是当初结合了 Watcom 的最佳化编译器以及类似 Delphi 的组件拖曳开发环境的第一个 RAD C/C++开发工具。更棒的是 Optima++的组件架构(类似 Delphi 的 VCL)完全是以纯正的 C/C++程序代码撰写的。这可不得了，因为这代表 Optima++是一个融合了 Visual C/C++和 Delphi 两大王者开发工具为一身的超级赛亚人工具。

在我知道这个工具、并且尝试实际使用之后，极为震惊。因为对于我这个使用了 C/C++ 5、6 年的人来说，它比 Delphi 更具有吸引力。因此我立刻在《RUN!PC》上介绍了这个不可置信的工具。果然，Optima++很快开始风卷市场，虽然没有立刻占据很大的市场份额，但是已经造成了一股气势，开始为 Visual C/C++和 Delphi 带来压力。

我记得当时台湾 Sybase 办的产品发表会也吸引了数百人与会，不可一世。我的文章在《RUN!PC》6 上发表之后，台湾的 Sybase 立刻和我联络，由当时的余协理和我见面，也是希望我继续为 Optima++写文章，台湾 Sybase 也提供额外一字加 2 元稿费的待遇。但是我告诉余协理，Optima++1.0 虽然很棒，但是仍然有一些臭虫，而且和中文环境相冲突，无法处理中文，需要立刻解决这个问题才能够在台湾的市场成功。她答应我

立刻向总公司反应。我也老实地告诉她，在问题没有解决之前，我无法写一些不确实的东西。后来台湾 Borland 的总经理方先生也找我去询问有关 Optima++的事情，我告诉他 Optima++是好东西，但是中文有问题。如果中文问题能够解决，那么将对 Borland 和 Microsoft 的产品有很大的影响，当时我还不知道 Borland 由于 Optima++的影响，已经开始准备开发 C++Builder。

在 1996 年底左右吧，Optima++1.5 终于进入 Beta 的阶段。但是在我拿到 Beta 版时非常失望，因为中文的问题仍然没有解决。后来台湾 Sybase 又找我去，这次和我见面的是台湾 Sybase 总经理郭俊男先生，以及 Sybase 的新加坡技术总裁，不过我忘记这位先生的名字了。见了面之后，我立刻把 Optima++1.5 中文的问题以及许多的臭虫告诉他们，希望他们能够解决，如此 Optima++1.5 才能够在中文市场成功。可是出乎我意料之外的是，他们似乎并不着急这些问题，反而询问我是否有意愿为 Sybase 工作，做 PowerBuilder 的产品经理。

也许是因为我为 Delphi 写了太多的东西，让 PowerBuilder 在台湾受了很大的影响，因此他们希望我到 Sybase 工作，以打击 Delphi 并且 Promote PowerBuilder。当时他们提出的待遇条件实在是非常、非常的诱人，比我当时的薪水高出一倍左右(我当时在资策会工作)。不过由于我对 PowerBuilder 实在没有什么兴趣，因此我告诉他们，如果是做 Optima++的产品经理，那么我将会考虑并且接受。

没有想到，Sybase 的新加坡技术总裁告诉我 Optima++在 1.5 推出之后就可能会停止，因为 Sybase 要把资源移去为当时愈来愈红的 Java 研发一个新的 Java RAD 开发工具，那就是后来的 PowerJ。于是他询问我如果不愿意做 PowerBuilder 的产品经理，那么是不是愿意做 PowerJ 的产品经理？由于当时我已经知道 Borland 开始了 Open JBuilder 的研发，而我对 Open JBuilder 的兴趣远大于 PowerJ，因此没有答应 Sybase。果然，在 Optima++1.5 推出之后，不但中文的问题没有解决，Sybase 之后也没有继续对 Optima++研发下去。

Optima++一个如此有潜力的产品就这样消失了，真是令人遗憾。Optima++应该有很好的机会可以成功的。我相信，如果当时 Sybase 知道 C++Builder 后来的成果，可能就不会放弃 Optima++了，而 C/C++的 RAD 工具一直要到后来的 C++Builder 来完成这个梦。

C/C++的开发工具之争到此算是告一段落了，虽然后来 Borland 继续推出了 Borland C/C++5.0，但是品质仍然不够好，市场反应也不佳。后来 Borland 终于在 Borland C/C++5.02 之后宣布停止此条产品线的开发，Borland C/C++的光荣历史也就从此打住，真是令人不胜感叹，而 Visual C/C++从此在 C/C++开发工具市场中再也没有对手。不过没有竞争的市场的确会让人松懈，后来的 Visual C/C++进步的幅度愈来愈小，MFC 也数年没有什么大进步，不像当时和 Borland C/C++竞争时每一个版本都有大幅的改善。看来寡占的市场的确是不好的。

^v^v^v^v^v^v^v^v^v^v

第三章　传奇的开始--Delphi

"是惊世之作的 Delphi 让 Borland 重新站了起来，没有当初的 Delphi，就没有今日的 Borland！"

"是 Turbo Pascal 诞生了 Borland，但却是 Object Pascal 给予了 Borland 重生的机会！"

创造传奇故事的主角--Delphi

没有人会知道在两年后 Borland C/C++会遭遇到这么大的失败，也没有人会预料到 Borland 又会再次因为 Pascal 而东山再起。Borland 奋斗史精彩的地方就在于每当似乎要不支倒地之际，Borland 的 R&D 人员就会创造出一个明星级的产品来拯救 Borland。在其他和 Microsoft 对抗的软件公司纷纷消失的时候，Borland 却一次又一次地站了起

来。"打不死的勇者"这句话贴切地形容了 Borland 的韧性。Borland 靠 Pascal 起家，通过 C/C++绽放光芒，进而达到了巅峰的状态，随后又再次靠着 Pascal 浴火重生。Borland 这个从 C/C++跌倒，再通过明星工具 Delphi 重回战场的过程可以说是惊心动魄，其中更牵涉到了 Borland 两位创始人 Philippe Kahn 以及 Anders Hejlsberg 相继离开 Borland 的密闻，也激活了 Borland 逐渐转型的历史轮轴。对于 Borland 来说，这段发展史可以算是非常关键的里程碑，更重要的是，Delphi 的崛起也在软件工具业界产生了巨大的影响。Delphi 不但激活了 Windows 平台上 RAD 战争的序幕，开启了 Windows 平台主从架构的改变，同时也对组件技术做出了巨大的贡献。直到现在，Delphi 创造的组件技术仍然深深地影响了 JavaBeans 以及.NET 的组件思想和技术，这在稍后的内文中读者可以逐渐地了解。而故事的起源便在 1993 年左右……

Delphi 的发展起源

当 Borland 以 Turbo Pascal 获得了成功，并且令 Charles Petzold 等人跌破眼镜之后，到了 1992/1993 年的 Borland Pascal 7.x，Borland 似乎已经把传统的 Pascal 开发工具发展到了极限，再往下还能做什么呢？Borland Pascal 在销售了数百万套之后，程序语言的焦点已经从 Pascal 转移到了 C/C++，Borland Pascal 无法继续快速成长，进而转入了递减的状况，Borland 必须做些新的东西才能够延续这条产品线。

当时 Borland Pascal 产品的 Architect，即 Anders Hejlsberg，眼看 Microsoft Visual Basic 的成功，觉得当时 Visual Basic 是比较初级的开发工具，是一个学习 Windows 程序设计的好工具，但是尚无开发真正应用系统的能力。因此，Anders 和 Borland Pascal 的小组决定展开一个规模前所未有的项目计划，这个开发工具项目在一开始便设定了数个目标，希望能够达成并且超越 Visual Basic。这些初始的目标是：

- 延续 Borland Pascal 的传统，提供一个快速编译的开发环境
  - Borland/Turbo Pascal 的高明之处便是由 Anders 使用汇编语言撰写的 Pascal 编译器不但编译快速，而且能够产出极为有效率的机器码。当时的 Visual Basic 只是解译器(Interpreter)，无法产生真正的执行机器码，因此在这一方面 Borland 决定要远远地超过 VB，但是 Borland 的挑战是要开发出一个编译速度能够媲美解译器速度的新一代编译器。
  - Anders 另外一个重要的决定便是改善 Borland Pascal 程序语言，让这个新的开发工具程序语言具备面向对象的功能。这在当时是非常重要的决定，因为不但需要大幅修改编译器，也正式将 Borland Pascal 超越 Pascal 之父对 Pascal 定义的结构，让 Pascal 拥有现代语言最新的功能。虽然这个决定有很大的因素是因为 Borland 决定通过面向对象的方式建立新一代的 Framework 和组件架构，因此需要程序语言方面的支持。不过，这在当时整个信息界对于面向对象技术还很陌生的阶段，的确是一个很大胆的决策。这个程序语言的决策虽然可以吸引专业人士的激赏，不过也可能会让许多程序员无法跨越这个障碍。后来的发展也证明了这一点。
- 建立一个新的 Windows Framework 组件架构
  - 当时 VB 使用的组件是 VBX。不过 VBX 架构非常的复杂，只能使用在 16 位的环境，

并且在可视化拖曳设计方面又不是很方便。因此 Borland 希望在 OWL 之后建立一个全新的 Framework，这个 Framework 能够让程序员快速开发 Windows 应用程序，并且完整地封装 Windows 操作系统中的对象。此外，Borland 也希望定义一个标准的组件架构，让使用这个开发工具的程序员能够通过 Framework 和组件架构来开发各种组件，包括可视化和非可视化组件。这个 Framework 就是后来的 VCL(Visual Component Library)。在这方面，Borland 做得非常成功。如果各位读者有 VBX 的经验，就会知道当时，Microsoft

定义的 VBX 规格简直是一团混乱，根本像是拼凑出来的东西。在当时开发 VBX 组件痛苦不堪，后来 Microsoft 也彻底放弃了 VBX。

- 拖曳、可视化的开发环境
  - Borland 的想法是开发一个全新的集成开发环境，在这个开发环境中程序员可以使用可视化的方法拖曳 Framework 的组件来设计图形界面，再在其中的编辑器中使用面向对象程序语言来撰写应用程序。

这个开发工具项目的名称就是：Delphi！

Delphi 的核心成员

在 Delphi 决定开工之后，Philippe Kahn 还不放心动用太多的资源来开发这个产品，因为当时 Borland 正集中所有的资源，希望能够打赢 C/C++开发工具一役。因此 Philippe Kahn 一开始只答应拨给 Anders 四个开发人员，先进行产品雏型的开发工作。因此，Delphi 在当时被笑称为像 Apple 计算机一样，是在地下室开发的。

当时加入 Delphi 开发小组的当然就包含了 Anders，第二人是 Chuck Jazdzewski。其中 Anders 负责撰写新的 Object Pascal 编译器以及核心程序，而 Chuck 则负责设计 Delphi 使用的组件 Framework，即 VCL。在经过了 6 个月的初始雏型阶段之后，当 Anders 把开发的结果呈现给 Philippe Kahn 看时，Philippe 立刻被它所吸引。因为当时在 Borland 内部也希望为 Borland C/C++开发一个类似这样能够以可视化拖曳方式开发应用系统的 C/C++开发工具。没有想到在短短不到一年的时间内，Anders 已经从基本的构想开发出了雏型产品。于是 Philippe 马上批准了这个产品的开发计划，并且投入研发资源。许多后来举足轻重的人才便是从开发 Delphi 项目培养出来的。当时在这个项目中，各个重要的部分分别由下面的重要人员负责：

- Anders Hejlsberg：编译器，Object Pascal 程序语言，产品架构
- Chuck Jazdzewski：Framework，组件架构设计/实现
- Allen Bauer：集成开发环境的开发工具，Open Tools API
- Danny Thorpe：RTL (Run-Time Library)
- Zack Urlocker：产品开发方向，产品规划

有兴趣的读者可以打开下面的链接，这篇文章是由 Danny Thorpe(现在是 Borland .NET 的 Architect)撰写的，详细地说明了 Delphi 这个名称的由来以及开发的缘由。

http://community.borland.com/article/0,1410,20396,00.html

而批准 Delphi 的开发，则是 Philippe Kahn 在因为 Borland 营运不佳而辞去 Borland CEO 之前做出的最重要而且正确的决策。没有 Philippe Kahn 的同意，便不会有两三年后浴火重生的 Borland。

大规模的开发行动和 Philippe Kahn 的下台

在 Borland 如火如荼地进行 C/C++最后决战的同时，Delphi 也在快速的开发之中。1994 下半年，Delphi 1.0 几乎已经开发完毕，最后剩下的工作就是 Beta 测试的阶段。同年，Borland 决定为 Delphi 展开一项从未进行的尝试计划，因为 Borland 对于 Delphi 信心满满。这个计划就是为 Delphi 进行前所未有的大规模测试，以确保 Delphi 的品质，避免重蹈 Borland C/C++发生的覆辙。Borland 为 Delphi 发出了成千上万的测试版本，邀请了广大的程序员为 Delphi 进行长期的测试。这可是空前绝后的，因为自 Delphi 1.0 之后 Borland 再也没有任何的产品能够拥有这种气魄和规模。我记得在 1994 年底左右，收到了来自当时 Borland 台湾产品经理张书良先生寄来的神秘圣诞节礼物。当时打开包裹一看，是六七片磁盘，没有任何的文件和说明。张书良先生请我安装看看这个"东西"，并且请提供一点意见。

在安装了这些"磁盘"之后，映入眼帘的是一个陌生的软件。"这是什么啊？"这是我当

时的第一个想法。后来玩玩此软件，发现乖乖不得了。不但大部分的 Windows 对象都可以拉拉就产生程序代码，更绝的是编译应用程序的速度比使用 Borland C/C++的编译器快了数十倍，而且产生的是一个体积不大的原生 EXE 文件，执行速度更是媲美 C/C++的程序代码。这让我这个惯用 C/C++的程序员当场傻眼。

"这怎么可能?"在我发出呓语般的声音之后，旁边的同事也觉得怪怪的，于是一个一个地跑到我的计算机旁，看看我到底在做什么？其中当然包括了《Delphi 学习手册》的作者、也是笔者的好友李增坤先生。在大家玩了之后，每个人都急着拷贝我的 Delphi Beta 版以便回家继续玩。后来李增坤先生更是玩得出神入化，还能够让 Delphi 连接到当时相当封闭的 Informix 数据库(因为他们的开发小组是使用 Informix 的)，真是厉害。他是我所知的第一个 Delphi 好手。

"这绝对是一个 Super Star！"，当时我这样对张书良先生说。"真的?那么你可不可以在杂志上帮 Borland 写一些介绍它的文章？"张书良先生对我这么说。就是因为这段对话，让我开始和 Delphi 结下了不解之缘。至于我开始写 Delphi 书籍的缘由也是无心插柳造成的。在台湾 Borland 准备力推 Delphi 1.0 之际，张书良先生准备亲自下海，也亲自出面找到了旗标出版社合作出书，以推广 Delphi。后来由于张先生工作太忙，因此又找了我和李增坤先生帮忙。本来的约定是我和李增坤先生只负责一小部分，其他的部分都由张先生完成。没有想到，签约之后张书良先生完全没有时间投入，因此只好由我和李增坤先生完成《Delphi 1.0 学习手册》。由于我和李增坤先生以前没有写书的经验，投入撰写书籍的时间也不多，因此《Delphi 1.0 学习手册》是台湾所有有关 Delphi 1.0 书籍中最晚出的一本书，远远超过当时我们规划的时程。好在当时 Delphi 1.0 的气势简直如星火燎原般的炙手可热，因此这本书还是卖得不错的。

1995 年对于 Borland 来说是悲喜交加的一年。1995 年 1 月 11 日，Philippe Kahn 正式因为经营不善而辞去 Borland CEO 的职位，不过 Philippe Kahn 仍然是 Borland 董事会的成员之一。接任的 Gary Wetsel 的任务是大幅删减 Borland 的员工数，开始进行瘦身计划。因为当时 Borland 的员工数是为营收 500M 美金的 Borland 所打造的，但是在 1995 年 Borland 的营收已经下滑为不及 200M 美金的公司，而且一直在亏损之中，当时许多业界人士都认为 Borland 已经撑不过 1995 年。不过 1995 年 2 月 14 日的情人节似乎一夜之间改变了 Borland 的命运。

一炮而红的 Delphi 1.0

1995 年 2 月 14 日，是 Borland 永远会记得的日子，因为这一天是 Delphi 正式诞生的日子，也是 Borland 扭转命运的转折点。由于 Delphi 先前大规模的 Beta 测试计划已经在全球吸引了极大的兴趣和好评，信息业界也知道了 Borland 正准备推出一个跨时代的新开发工具产品。当然，更重要的是全信息界也都在静观，这个产品是否真的好到能够拯救 Borland 免于破产或是被并购的命运。决定生与死的日子终于在这一天揭晓。

1995 年 2 月 14 日，也就是 Borland 在全球发表 Delphi 1.0 当天，我在 Scott Valley 会见了当时的 Delphi 主舵手，产品经理 Lance Devin 先生。Lance 是一位非常亲切、有活力的人。Delphi 在他的主掌之下，立刻在全球吸引了所有的焦点，当时媒体甚至称 Delphi l.0 是 VBK(Visual Basic Killer)。

Delphi 1．0 发表之后，立刻造成了全球的狂卖。由于 Borland 并没有预料到 Delphi 的反应会如此的好，因此一时造成了 Delphi 的全球大缺货。Borland 从 Borland C/C++3.1 之后已经很久没有享受过这么美好的滋味了。

在台湾，由于早已预料到 Delphi 将会是一个成功的产品，因此，台湾几乎和美国同一时间发表了 Delphi 1.0。而且台湾 Borland 不惜血本，直接从美国空运了少数的 Delphi，而台湾能够取得的 Delphi 的数量也只是从美国抢破头才拿到的少量货。台湾

Borland 是在信义路的震旦行 2 楼会议室发布 Delphi 的。当天整个会议室几乎被塞爆了，因为有太多急于想一睹 Delphi 庐山真面目的软件人员。我还清楚地记得在发布会结束之后，会议室的门口排满了抢购 Delphi 的人潮。很快，所有的 Delphi 都被抢光了。记得当时李匡正先生没有抢到 Delphi 1.0，一直到 2 个多礼拜之后才取得。而我呢？很幸运的是在 Delphi 1.0 发表之前，张书良先生就已经送了一套正式的 Delphi 1.0 Client/Server 版让我玩。当然我也迫不及待地把 Delphi 介绍给我当时的老板，希望我们的软件包能够赶快使用 Delphi 来写 Windows 的版本，但是我的老板还是坚持使用 Visual Basic 来写。后来我就离开这家公司，找寻愿意使用 Delphi 开发的软件公司。

当时 Delphi 在台湾书市造成的旋风真可用"洛阳纸贵"来形容，任何和 Delphi 1.0 有关的书籍都立刻大卖，看得每一个出版社都眼红不已。我也还记得当时第一本 Delphi 1.0 的书是由波全出版社推出的。根据台湾最有名的天珑书局老板彭先生说，最热门的时候一天几乎可卖 500 本的数量。我想这一本 Delphi 书籍应该是台湾有史以来销量最好的 Delphi 书了，估计当时这本波全的书有数万本的销量。更夸张的是后来我居然在天珑书局看到由 2 本影印的合集 Delphi 书籍，由塑料套包起来，要价是"1500"块台币，居然也很快卖完，真是令人不可思议。这即使不是绝后，也绝对是空前的。

Delphi 1.0 的成功也许早在信心满满的 Anders 的预料之中，看看下面在 Delphi 1.0 中秘密内藏的 Easter Egg 中，Anders 笑得如此的灿烂似乎就已经预见了 Delphi 光明的未来。

Delphi 1.0 有多成功呢？根据非正式的统计，Delphi 1.0 当时在全球狂卖了 50 多万套，这实在是一个惊人的数字。读者如果没有什么概念的话，那么我可以举一些例子来比较一下。Borland 最成功的 Borland/Turbo C/C++系列卖到了 3.1 最巅峰的时候，全球的销量才超过 100 多万套，这可是累积了数年、数个版本后才达到的套数。而 Delphi 一个版本就到达了 C/C++几乎一半的销量，从这就可以知道当时 Delphi 有多成功了。

Delphi 1.0 的大卖，立刻拯救了财务困难的 Borland。Delphi 的收入不但让 Borland 立刻再投入更多的资源到 Delphi 开发小组，以准备下一个版本的开发，也让当时 Borland 内部的 Latte(就是后来的 JBuilder)小组获得了更多的研发资源，成就了数年后 JBuilder 再次接棒；把 Borland 推向另一个高峰。

再见了，Borland 创始人，Philippe Kahn

1995 下半年，Borland 发生了一件重大的事情，那就是 Philippe Kahn 正式被逐出他一手创建的 Borland。这真是令人震惊又难过的事情，相信许多关心 Borland 的读者都知道这件事情。但是为什么 Philippe Kahn 会被踢出 Borland 董事会、又离开 Borland 呢？这可是一个秘密。

事情都是从 Philippe Kahn 辞下 Borland 的 CEO 后开始发生的。在 Philippe Kahn 被逼下 CEO 之后，他觉得 Borland 的一些开发方向他并不是很认同，因此在外面又开了一家新的公司 StarFish，从 Borland 买走了 SideKick、DashBoard 等产品，并且开始研发移动和无线等方面的软件。

1995 年 Java 兴起之后，Philippe Kahn 觉得 Java 很有前途，并且希望结合 Java 以及移动和无线软件技术。其时 Borland 内部也在开始研发 Java 的产品，包含了代号是 Latte 的 Java 开发工具以及 Java 的 JIT 编译器等技术。而 Borland 没有预料到，由于 Java 的萌萌芽竟会造成 Philippe Kahn 和 Anders 的离开以及 Borland Visual dBase 小组的解体。

话说在 Borland 于 Java 方面逐渐有了成果之后，Philippe Kahn 的 StarFish 公司也开始步上轨道。1995 年，Philippe Kahn 眼看 Borland 内部 Java 的人才素质精良，于是就开始想挖一些好手到自己的 StarFish 公司。在 Philippe Kahn 的挖角动作愈来愈大之后，Borland 的董事会终于无法忍受 Philippe Kahn 这种挖 Borland 墙角的做法。于是，

Borland 的董事会成员一致投票决定，将 Philippe Kahn 逐出 Borland 的董事会和 Borland。这对于 Philippe Kahn 是一个极为重大的打击，Philippe Kahn 被迫离开了他一手创办和心爱的 Borland。即使后来 Philippe Kahn 的 StarFish 经营得不错，以致后来由 Motorola 以数千万美金并购了 StarFish，让 Philippe Kahn 大大地赚了一笔，但是他仍然无法释怀，也永远无法忘记 Borland 给他的成功、光荣、骄傲和屈辱。虽然 Philippe Kahn 一直想像苹果计算机的 Steve Jobs 一样有朝一日能够重返 Borland，但是，很显然 Philippe Kahn 没有 Steve Jobs 那样的运气，Philippe Kahn 一直无法完成这个愿望。

Anders 的计划以及 Zack 的想法

在 Delphi 1.0 大获成功、如日中天之后，雄心勃勃的 Anders 立刻开始了下一版 Delphi 的开发计划。此时 Delphi 研发小组的资源更多，因此可以做更多的东西。不过，在 1995 年 Delphi 1.0 推出之后，信息业界有了几项重要的改变，那就是随着 Microsoft Windows 95 的成功，企业使用 Windows 平台开发应用系统已经成为既定的趋势，再加上当时数据库市场的快速发展，因此许多企业开始在 Windows 平台寻找 Client/Server 的解决方案。正由于这些需求快速而大量的兴起，造成了当时 PowerBuilder 和 Gupta 这两个主从架构开发工具的盛行。当时，PowerBuilder 是 Window 平台下占有率超过 50％的主从架构开发工具，而 Gupta 则拥有超过 30％的市场。这真是可怕，因为光是两个工具就占据了 80％多的市场。由于当时主从架构几乎由这两个工具所寡占，因此，PowerBuilder 和 Gupta 的价格相当昂贵，我记得当时一套 PowerBuilder 要价 40 几万新台币，而 Gupta 也要 30 几万，真是令人无法相信。

在 Microsoft 方面，由于 Delphi 1.0 的成功，给了 VB 相当大的压力。因此 Microsoft 在 Delphi 1.0 推出之后立刻也推出 VB 4.0 正面迎战。VB 4.0 强调的重点是 VB 应用程序也可以编译成可执行文件，不过，由于 VB 4.0 的编译器品质尚不成熟，编译出来的效果并不好。再加上它的臭虫非常多，因此 VB4.0 算是一个相当不成功的产品。正由于这些因素，在当时也传出了 VB 双数版本品质不如奇数版好的传闻。不过，在当时由于 PowerBuilder 和 Gupta 的获利非常丰富，而 Microsoft 也看到了主从架构将会是未来数年重要的信息架构，因此 VB 4.0 开始，Microsoft 也开始逐渐为 VB 加入更多开发数据库以及主从架构的能力，并且搭配 Microsoft 的 ODBC 规格向主从架构市场进攻。

Anders 在 Delphi 1.0 成功之后，曾经接受媒体的访问，叙述他心中的 Delphi 2.0 想做的功能。当时 Anders 就说他希望为 Delphi 加入 Garbage Collection 的功能，因为 Object Pascal 在建立对象方面是使用 Heap-Based 的方式，因此为了减少 Delphi 程序员可能发生的错误并简化 Delphi 程序代码的撰写，他希望加入 Garbage Collection。现在的 Microsoft 的．NET 就内建了 Garbage Collection 的功能，而这个想法在 7 年前便已经存在于 Anders 的脑中了。

除了 Garbage Collection 之外，Anders 也想为 Delphi 加入更多 Stack-Based 的能力(是巧合吗？.NET 的 IL 也是 Stack-Based 的语言)，并且持续地改善 Delphi 的编译器，加入更多的编译器最佳化功能，让 Delphi 的程序代码执行速度能够超越 C/C++。

从 Anders 的想法中，读者应该可以感觉到 Anders 想做的都是属于比较语言、系统和低阶方面、影响层面较大的功能。但是，由于信息市场是逐渐走向主从架构，因此 Zack Urlocker 等人则希望 Delphi 2.0 能够在主从架构方面进行大幅的强化，再搭配 Borland 力倡的 BDE/IDAPI 技术，以便和 PowerBuilder/Gupta 等竞争，进入获利较为丰富的工具市场，这是第一次 Anders 和 Zack 意见分歧的时间点。

后来 Delphi 研发小组达成了共识，那就是下一个版本的 Delphi 将由 Anders 在编译器方面主导，为 Delphi 开发一个真正的 32 位编译器，而且具备最佳化的功能(因为

Delphi 1.0 是 16 位的开发工具)。但是 Delphi 2.0 也将大幅加入主从架构的功能，并且通过 BDE/IDAPI 提供连接各种 RDBMS 的驱动程序，再由 Chuck 改善 VCL 架构，提供更为强
劲的数据感知组件能力，让 Delphi 2.0 正式具备和 PowerBuilder/Gupta 竞争的本钱。这也埋下了日后 PowerBuilder/Gupta 这两个工具备受 VB 和 Delphi 强大的压力、终至快速衰退的命运。

由于 Delphi 2．0 开始确定走向主从架构和具备开发大型系统能力的方向，因此 Anders 没有多余的资源可以实现他的理想，再加上和 Zack 在产品发展方面日渐出现歧见，这些都是间接造成日后 Anders 离开 Borland 的因素。

Delphi 2.0，进入 32 位世界的开发工具

在 Anders 完成了 Delphi 32 位的编译器而且 BDE/IDAPI 小组也实现更多连接 RDBMS 的驱动程序之后，Delphi 2.0 便已经准备好出征了。Delphi 2.0 在推出之后果然也非常成功，Anders 亲手打造的 32 位编译器不但编译速度奇快，编译出的应用程序品质更是无话可说。在当时，Delphi 2.0 产生的执行程序代码屡获专业媒体和实验室的评比大奖，尤其在整数运算方面，更是比 VC++执行得还好。在一般应用程序方面，也和 VC++的程序代码不相上下。整体来说，只有在浮点数方面落后 VC++。这也是后来 Borland 编译器小组和 Anders 激活 Borland 下一代编译器项目的原因之一，目的是为 C++Builder 和 Delphi 开发一个共享的后端最佳化编译器。不过很可惜的是，Anders 稍后离开了 Borland，没有参与完成这个最佳化编译器，否则 Borland 的编译器应该会比现在更具威力。

Delphi 2.0 如同 Delphi l.0 一样大获成功，因为当时许多想在 Windows NT 下开发纯 32 位应用系统的程序员都愿意使用 Delphi 2.0，更不用说一些企业的开发者，在不愿意忍受 PowerBuilder/Gupta 等使用脚本语言执行应用程序的缓慢情形下，许多 PowerBuilder/Gupta 使用者都转而使用 Borland 的 Delphi，这是 Borland 继当初 Borland C/C++3.1 之后再次打入企业市场。Delphi 2.0 和 Delphi 1.0 的总销量也超过了一百万套。在全球的市场中，Delphi 在欧洲、亚洲和中南美洲都非常的成功，反而在北美的市场表现平平。由于 Delphi 表现得非常抢眼，给予了 VB 和 PowerBuilder 巨大的压力，因此 Microsoft 和 Sybase 在 Delphi 2.0 推出之后，也纷纷地准备推出 VB 5.0 和 PowerBuilder 6.0，正式揭开了 RAD 工具最后的生死战。

在 Delphi 2．0 顺利推出之后，很不幸的是 Lance 也离开了 Borland。接手 Delphi 产品经理的是 Ben Riga。要怎么说这位加拿大的仁兄呢？Ben Riga 也是很亲切的人，但是我当时向他建议的许多 Delphi 发展方向，这位仁兄都没有反映在 Delphi 的产品中。例如我在 3.0 时便强烈建议 Delphi 应该在 Web 方面投入更多的努力，在 Delphi 中实现类似当时 IntraBuilder 的能力，提供可视化的方式开发 Web 应用系统，也就是像现今的 Developer Express 的 ExpressWeb Objects 组件组，或是 IntraWeb 组件组。如果在当时的 Delphi 3 或是 Delphi 4 便能够有这种组件组，那么 Delphi 的 Web 能力将是 No.1，更不用说对于 Delphi 的销量会有多大的帮助。可惜笔者人微言轻，Ben 并没有把这个重要的功能放人 Delphi 的产品开发规格中。一直到现在，Delphi 6/7 才认真地考虑这个需要，虽然时间已经晚了，但是如果真的做得好的话，也比没有好。

RAD 殊死战

在 Delphi 和 VB 相继进入主从架构的世界之后，便和 PowerBuilder 有了更激烈的竞争。在 Delphi 2.0 出版之后，PowerBuilder 面临了更大的挑战。因为 VB 在易于使用和 Microsoft 的努力之下仍然呈现成长的趋势，但是 PowerBuilder 在许多场合却是直接和 Delphi 竞争。在 Delphi 2.0 推出之后，Borland 也很快查觉到许多 Delphi 的使用者是从

PowerBuilder 转来的，于是当时 Borland 内部便拟定了置换 PowerBuilder 计划，希望能够持续从 PowerBuilder 手中抢得更多的市场占有率。

Delphi 2.0 的纯 32 位编译器以及执行速度一直是许多人喜欢的，这也对 PowerBuilder 带来了压力，因为许多 PowerBuilder 的使用者，特别是软件公司对于 PowerBuilder 使用的语言 PowerScript 执行缓慢而不悦，但是却喜欢 PowerBuilder 的 DataWindow。因此在当时 Sybase 便宣告将为 PowerBuilder 开发一个编译器，以增加 PowerBuilder 应用程序的执行速度。不过 Sybase 并没有在 Delphi 2.0 之后的 PowerBuilder 5.0 实现出来，一直要到 PowerBuilder 6.0 才推出 PowerBuilder 编译器，不过问题仍然多多。

当时我曾经仔细地观察台湾参加 Delphi 发布会的使用者以及参加 PowerBuilder 的使用者，发现了一个很有趣的现象，那就是一开始参加 Delphi(1.0/2.0)的使用者几乎都是工程师，很少有 DB Center 的信息人员或是信息经理。而去 PowerBuilder 的使用者则甚少工程师，大部分都是穿着体面的 DB Center 的信息人员或是信息经理。不过这个情形在 Delphi 2.0 之后逐渐改变，因为当 Delphi 慢慢地被企业接受成为开发工具之后，也有愈来愈多的信息主管人员出现在 Delphi 的发布会中。

下图是我早在 1999 年便绘制的资料，从这个 Slide 中读者可以发现 Delphi、VB 和 PowerBuilder 的竞争是一个版本对应一个版本的。每当竞争对手推新的版本之后，另外的竞争对手都会立刻推出相对应的竞争版本。但是在 Delphi 2.0 之后，PowerBuilder 便开始逐渐无法同步跟上 Delphi 和 VB 的竞争脚步，差距也愈拉愈大。

基本上 Delphi、VB 和 PowerBuilder 的竞争在 Delphi 2.0 时期是最为激烈的，在 Delphi 3.0 和 VB 5.0 之后大势已定。PowerBuilder 已经定位成数据库开发工具，无法像 Delphi 和 VB 一样成为 Windows 平台中通用的开发工具。因此 PowerBuilder 的使用者群组便逐渐缩减到 DB Center 或是信息室的使用者。这个现象当时在台湾非常的明显，因为大多数的软件开发厂商、软件包厂商或是项目厂商逐渐地舍弃 PowerBuilder，不是选择使用 VB 就是选择使用 Delphi。在 Delphi 3.0 之后，Windows 平台的主从架构开发工具市场已经是 VB 第 1，Delphi 第 2，而 PowerBuilder 退到第 3，并且和 Delphi/VB 的差距愈来愈大。经过了 3 年的激斗，传统主从架构开发工具 PowerBuilder 和 Gupta 仍然不是 Microsoft 和 Borland 的对手，即使 PowerBuilder 曾经拥有超过世界 50％的占有率。当初 Sybase 也没有预料到在并购了 PowerBuilder 之后，这么快就损失了大片的江山。不过比起 Oracle 和 Informix 来说，Sybase 下的本钱还是比较划算的，因为同时竞争的 Oracle Developer 2000 早已只能送给 Oracle 数据库的使用者。而 Informix 的主从架构开发工具 Neon 更是凄惨，在出了一个版本之后，由于体积庞大，执行缓慢，又臭虫成堆，还没上场竞争，就被丢人垃圾桶了。这对当时所有投入大量资源开发主从架构开发工具的软件厂商来说，都是受伤不轻，当然除了 Borland 和 Microsoft 之外。

PowerBuilder 和 Gupta 数年辛辛苦苦建立起来的主从架构王国，只经过短短的 1、2 年就被 VB 和 Delphi 所鲸吞蚕食是有许多原因的。PowerBuilder 和 Gupta 着重在高价、专有的主从架构市场，却不知主从架构已经逐渐成为一般信息架构应用主流，大量的程序员需要的是价格合理、功能齐全的开发工具。PowerBuilder 和 Gupta 价格的奇高和功能的缺失，在 VB 和 Delphi 加强主从架构的功能之后便逐渐地浮现出来。此外 PowerBuilder 和 Gupta 当初风行的一个重要因素是提供了连接到各种 RDBMS 的驱动程序，以提供开发数据库和主从架构应用系统的能力。但是当 VB 和 Delphi 都分别提供了 ODBC 和

BDE/IDAPI 技术连接更多的数据库服务器之后，PowerBuilder 和 Gupta 的优势也就不再了。另外一个关键性的因素是 PowerBuilder 和 Gupta 一直无法成为大众化的开发工具。除了 DB Center 和企业之外，PowerBuilder 和 Gupta 无法被大量的 ISV、商业软件包厂商、

一般工程师、甚至是学生使用来开发各种不同的应用系统。因此 PowerBuilder 和 Gupta 只能在特定的软件族群中使用，限制了可能成长的潜力市场。

最后我认为最重要的原因是 Sybase 和 Gupta 在面对 Microsoft 和 Borland 这两个非常精于实现开发工具的厂商时过于轻心，反应的速度太过于缓慢，以致 PowerBuilder 和 Gupta 除了在数据库和主从架构功能之外，其他的功能都太过阳春。特别是在 Java 逐渐兴起之后，PowerBuilder 和 Gupta 最后跨平台的诉求也在瞬间瓦解，因此失去市场也是不可挽回的命运了。

Borland C++Builder 的诞生

在 Delphi l.0/2.0 大获成功并且对 Borland 产生了巨大的收益之后，来自 Borland C/C++使用者的需求，即要求 Borland 也推出一个 C/C++版本的呼声也愈来愈强烈，到了 Borland 无法忽视的地步。不过这对 Borland 来说，却是一个相当伤脑筋的事，因为 Delphi 是用 Object Pascal 写的，如何转成 C/C++？如果要开发 C/C++版的 Delphi，那么 Borland C/C++的产品线要如何处理？因此这也在 Borland 内部起了极大的争议，这又是另外一个故事了。不过不可否认的是，Delphi 1.0/2.0 的成功正是促成了 Borland C++Builder 面世的主要动力。

争执的开始

Delphi 2.0 的再次重大成功仿佛为 Delphi 研发小组注入了一支巨大的强心剂。Delphi 研发小组迫不及待地要开发 Delphi3.0，以便趁胜追击，进一步的拉近和 VB 的距离，并且再次给予 PowerBuilder/Gupta 重击，以便从 PowerBuilder/Gupta 取得更多的市场。

在 Delphi 2.0 推出之后，根据当时 Gartner Group 的调查，主从架构开发工具的第一名仍然是 PowerBuilder，第二名是 VB，而第三名则是 Delphi。Gupta 的市场占有率快速地下滑，已经无法进入前三名的占有率了。不久之后 Gupta 改名为 Centura，希望力挽狂澜，但是仍然无法改变市场的现实，并且逐渐淡出主从架构的开发工具市场。

这对于 VB 和 Delphi 来说都是重大的胜利，因为 VB 和 Delphi 在短短的一两年之内就几乎瓦解了 PowerBuilder/Gupta 在主从架构市场数年经营的优势。虽然 PowerBuilder 还暂时维持第一的占有率，但是已经从 50％几萎缩到了 40％，而且还在不断的下滑之中。PowerBuilder 在 VB 和 Delphi 的强烈进攻下岌岌可危，产品价格也在快速地滑落之中。

不过在 Delphi 3.0 的功能会议中，Anders 和 Zack 却再次发生了强烈的争执，因为这两个 Delphi 的灵魂人物对于 Delphi 3.0 的走向出现了不同的看法。其时由于 Microsoft 的 COM/DCOM 等技术逐渐成熟和受到愈来愈多的使用，因此 Anders 希望 Delphi 3 能够充分地支持开发 COM/DCOM 的技术，并且提供比 VB 更方便、比 VC 更强大的能力。为了实现

这个想法，Anders 力邀 Alain Lino Tadros 加入 Delphi 的研发小组。为什么 Anders 会找上 Alain 呢?这是因为 Alain 曾在 1996 年 BorCon 的研讨会上以惊人的技术写了 5000 多行的程序代码，把 VCL 组件直接变成 COM 对象。Anders 在 BorCon 上知道之后，立刻有一

个想法，那就是如何能够延伸 Alian 的技术，如果通过一种方法便能够把所有或是大部分的 VCL 组件变成 COM 对象以及 COM Container 的话，那将是多奇妙的事情。如此一来 Delphi 的研发小组也不需要为实现 COM/DCOM 的功能而大费周折了。而 Anders 的这个想法也促成了后来 Delphi 3 中 IVCLComObject 接口和技术的诞生,直接把 VCL 转换为 COM 对象的神奇功能。

除了邀请 Alian 之外，Anders 也计划直接修改 Delphi 的编译器，让编译器能够直接处理 COM 的参考计数值(Reference Count)，免除 Delphi 程序员的麻烦，并且提供有效的执行码。如果依照当时 Anders 的想法并且把 Delphi 3 依此实现出来的话，那么 Delphi

将会是一个比 Microsoft 工具更好的 COM 开发工具。
不过 Zack 却不认为 Delphi 3 应该把这么多的资源花在支持 COM/DCOM 上。虽然 Zack 同意 Delphi 3 必须对 COM/DCOM 有好的支持，但是他认为 Delphi 3 应该在 Delphi 2 的主从架构获得了胜利之后，继续往更高阶的方向努力，那就是分布式架构，以便把 Delphi 塑造成能够开发大型企业的开发工具，把 Delphi 打入获利丰富的企业市场。因此 Zack 和当时 Borland 开发工具部门的负责人 Paul Gross 激活了所谓的 Golden Gate 计划，希望提供 Delphi 强大的中介引擎功能，因此 Borland 并购了出版 Entera 中间件的公司，让 Delphi 能够通过 Entera 连接到后端大型的系统，希望大型企业能够使用 Delphi 开发客户端的应用程序。
由于 Golden Gate 计划得到了 Borland 高层的支持，因此 Delphi 3 的发展方向很快便朝向分布式技术前进。虽然 COM/DCOM 的支持也在产品的计划之中，但是并无法做到像 Anders 所想要的境界。由于 Anders 的理想无法被接受，因此在 Delphi 3 的发展中后期阶段，Anders 并没有介入太多 Delphi 3 的开发工作。
在 Delphi 3 正式发表之际，Delphi 的研发小组很巧妙地平衡了 COM/DCOM 和分布式技术的功能，也让 Delphi 3 成为 Windows 平台中第一个提供分布式开发的开发工具。而 Delphi 3 当时使用的 Marketing Slogan 以及产品诉求的重点也是"Component Foundry"，意谓 Delphi 3 可以轻易地建立 VCL 组件以及 COM/DCOM/ActiveX 组件。
Delphi 3 推出之后，Delphi 应该算是到达了巅峰的状态，销售数字也非常的亮丽，已经超过了 Borland C/C++3.1 的销量，而 Borland 也有了稳定的收入。但是人无远忧必有近虑，后来发生的数件大事几乎让 Delphi 重演当初 Borland C/C++4．0 的历史覆辙。
不知道读者看到这里有什么想法？我认为 COM/DCOM 和分布式技术都是重要的，后来的事实也证明分布式技术是 Delphi 的强项之一，而应用系统架构也逐渐走向分布式多层。不过没有像 Anders 的想法把 Delphi 塑造成无敌的 COM/DCOM 组件开发工具相信也是许多人的遗憾。
在 Anders 逐渐不介入 Delphi 3 的开发之后，随后发生的两件关键的事情便注定了 Anders Hejlsberg 继 Philippe Kahn 离开 Borland 的命运。
天才的损失和新英雄的接棒
在 Anders 和 Zack 对于 Delphi 的走向逐渐出现了歧见之后，Anders 便没有再主导 Delphi 3.0 的开发，反之 Zack 在 Delphi 开发小组中的角色却日益重要，后来几乎是 Delphi 3 和 Delphi 4 的主要领导人。为什么 Delphi 的 Architect Anders 会慢慢地淡出 Delphi 的核心呢？这和 Philippe Kahn 离开 Borland 也有重要的关系。
英雄落难
Philippe Kahn 和 Anders 共同创造了传奇的 Borland，两人之间有着浓厚的感情。在 Borland 工作时，对于 Anders 任何的想法和计划，Philippe Kahn 都是不遗余力地支持。也正是这个重要的支持力量，才有随后极为成功的 Borland Pascal 以及 Delphi 的问世。但是在 Philippe Kahn 离开 Borland 之后，Anders 再也没有了这股来自最亲密战友的强力支援。1997 年，Borland 新的 CEO Delbert Yocam 在掌握了大权之后，Borland 整个公司开始走向第二个重要的转变，Delbert 对于 Borland 产品的开发和趋势也有了不同于 Philippe Kahn 的看法。当 Java 在 1996 年逐渐快速发展之后，睿智的 Anders 也看到了 Java 成功的未来。因此在 Anders 不再积极参与 Delphi 2/3 的开发工作之后，他非常希望能够主导 Borland Java 开发工具的开发，期望能够像当初的 Delphi 1.0 一样，为 Borland 再次开发出全世界一级的 Java 开发工具。
不过，由于当时 Delphi 是 Borland 最重要的收入来源，高层仍然希望 Anders 继续在 Delphi 产品线上投入全力，因此当时的 Borland CEO Delbert Yocam 并没有批准 Anders

的请求。Borland 的下一个重要的开发工具 JBuilder，当时的产品开发名称为 Latte，仍然交由其他小组负责。依据我的推想，由于当时 Anders 对于 Java 已经有许多的想法，因此才会有后来的 VJ++以及 C#，这些产品和程序语言的许多特性想必已经在 Anders 的脑中存在了一段时间了。

Delbert 没有允许 Anders 带领 Latte 开发小组，但 Anders 仍然没有放弃他的新计划。也许是 Anders 注定和 Borland 的缘分已经到了尽头，这个时候正好 Microsoft 展开了有史以来对 Borland 最大的挖角行动。在 Anders 无法在 Borland 取得满意的支持之后，Microsoft 提供的优厚条件顿时对 Anders 产生了致命的吸引力，从而造成了 Borland 无法挽回的遗憾。

虽然 Anders 没有显赫的学历，无法获得 Turning Awards(即图灵奖，信息科学界最高荣誉的奖项，等同于诺贝尔奖)。但是我认为 Anders 的实力和贡献绝不输于任何一位 Turning Awards 的得奖人。Anders 是最好的信息实践型人物，在 2001 年，他终于获得了信息界最具权威的信息刊物 Dr. Dobbs' Journal 颁发的 Excellent Programming Awards，以表彰 Anders 为信息界做出的卓越贡献。我想 Anders 应该是许多本身没有高学历或不是计算机信息科系出身的优秀程序员最好的效仿对象。

Anders Hejlsberg 这位不世出的软件天才，是目前全世界最顶尖的软件技术人员之一。论实现技术，Anders 可能是目前的第一高手，因为他精通程序语言、编译器技术、开发工具、Framework 以及系统架构。我虽然知道许多软件界重要的人物和好手，但是尚不知有任何人能像 Anders 一样在这么多领域都能成为大家。下面是笔者整理出 Anders Hejlsberg 到目前为止重要的功绩、贡献以及获颁的重要大奖：

" 和 Philippe Kahn 共同创办 Borland

" 开发出 Turbo Pascal，当时首创的 In-Memory Compiler 震惊了全世界

" 开发出全世界最畅销的 Pascal 产品，Turbo Pascal(这是许多信息人员学习 Pascal 和 Data Structure 使用的经典产品)以及 Borland Pascal。Turbo/Borland Pascal 合计销售超过了数百万套。Dr. N. Wirth(Pascal 语言的创始人员)也应该向 Anders 致敬，表达 Anders 对于 Pascal 语言的贡献

" Anders 使用汇编语言撰写编译器，其功力无人能出其右。创造出了全世界速度最快、品质也是一流的 Pascal 编译器。在 Anders 离开了 Borland 之后，几乎没有人能够修改 Anders 的编译器

" 开发出影响深远的 Delphi 这个伟大的 RAD 工具

" 开发出 VJ++语言

" Microsoft .NET 的 Architect

" Microsoft 颁授 Microsoft Distinguish Engineer 大奖

" 发明 C#这个又将造成重大影响的语言

" 获颁 2001 年 Dr. Dobbs' Journal 的 Excellence In Programming 大奖

一个人一生能够做出几件让全世界都津津乐道的事业呢？Anders 却成就了无数 PC 界伟大的功绩，并且在程序语言、编译器、开发工具以及 Framework 方面都有重要的贡献。PC 软件界因为有了 Anders 而精彩、丰富了许多，也创造了许多令人惊叹的故事。更棒的是 Anders 现在仍然在继续贡献他惊人的天分，就让我们拭目以待，看看 Anders 还能创造什么功迹吧。不过，不管以后如何，相信 Anders 应该是大部分软件人员希望学习的目标。Anders 的功力也是大部分软件人员一生企望能够达到的境界。

在 2002 年 Borland Developers' Conference 中，Anders Hejlsberg 是排名第一的 Keynote Speaker，尚在 Java 的创始人 James Gosling 之前。根据现场同时聆听这两场 Keynote Speech 的听众报道，Anders 的 Keynote Speech 是非常精彩的，而 James 的

Keynote Speech 则相对的枯燥，许多人因此而提前离席。而且 Anders 在开始进行 Keynote Speech 之时，便获得了现场所有听众起立鼓掌致敬，看来，在大多数 Borland 开发工具使用者的心中，Anders Hejlsberg 是永远的巨星。

Microsoft 的挖角和 Anders 的离开

Anders 在不介入 Delphi 的开发、并且无法主导 Borland Java 开发工具开发的情况下充满了挫折感。没有了 Philippe Kahn 的强力支援，Anders 虽然是 Borland 最顶尖的技术人才，却也无法对抗 Borland 管理阶层的力量。当然这也是从 Philippe Kahn 离开了 Borland 之后、Bodand 开始转型有关，这在稍后 Borland 的转变一文中，我会作详细的说明。

虽然 Anders 在 Borland 遇到了挫折，但是对于 Microsoft 来说这却是千载难逢的好机会，在此时 Microsoft 展开了大规模的挖角行动，而且是明目张胆地进行，正是由于 Microsoft 如此大胆的行动，因此也造成了不久之后 Borland 对于 Microsoft 的法律控诉。这次的挖角行动中，Microsoft 同时锁定了数个 Borland 最杰出或是重要的人物，当然 Anders 是名列第一的挖角对象。时值 1996 年，Microsoft 终于展开了行动，使用的方式是最直接的攻击。Microsoft 直接派遣加长型的大轿车到 Borland 大门口找 Anders 吃饭，第一次 Microsoft 开出了年薪百万美元以上的条件。不过在 Borland 知道了这件事情之后，也很快进行了加码的动作，因此 Anders 并没有对 Microsoft 进行响应。

Microsoft 在苦等无应、按捺不住之下，很快就再次用大轿车找 Anders。这次 Microsoft 提出了两百万美元以上的条件，希望 Anders 能够首肯。对于这次的喊价，Borland 可有点为难了，因为两百万美元不是笔小数目，这已经比当时 Borland 许多副总裁的年薪还高。此外，如果 Borland 答应也加到两百万以上，那么是不是 Chuck 也要如此加码？其他的 Delphi R&D 小组要如何调整？这些都是非常棘手的问题。

不过 Borland 很快找到了解决的方案，那就是允许 Anders 从每一套卖出的 Delphi 版本中抽取一定数量的版权费。如此一来 Delphi 卖得愈好，Anders 便能取得愈多的回馈。不过就我的了解，Anders 注重的并不是金钱上的问题，因为在 Borland 创立的初期，由于 Turbo Pascal 的编译器都是 Anders 撰写的，因此当时 Anders 也是卖一套 Turbo Pascal 就可以抽取版税的。依照 Turbo/Borland Pascal 全世界销售数百万套来算，Anders 早就是富翁了。薪水多一点，少一点并无所谓，Anders 心中想的是自由发展的空间。在 Borland 提出了 Delphi 的随版抽税，再加上 Microsoft 并不知道 Anders 真正想要的东西，因此 Anders 仍然没有响应 Microsoft 提出的优厚条件。

不过，Anders 实在是太重要的人物，而且 Microsoft 在面对 Java 与日俱增的威胁下，非常渴望能够有像 Anders 这样的人才带领 Microsoft 开发下一代的开发工具，这当然也是由于 Microsoft 以前向 Borland 挖来的人都做出了不小的贡献所致。Microsoft 食髓知味，当然希望能够得到 Borland 的镇山之宝。在 Anders 两次不为所动之后，Microsoft 决定祭出最后的王牌，由 Bill Gates 亲自找 Anders 吃饭，进行最终的挖角行动。

不管读者喜不喜欢 Bill Gates，不可否认的是 Bill 也是一个天才。自古英雄惜英雄，在 Anders 和 Bill 相谈甚欢的情形下，Microsoft 开出了年薪三百万以上、数万股的 Microsoft 股票这个超高的条件，再以当时 Microsoft 高贵的股价来计算，真是一笔庞大的数字，也许对于搞软件技术的人来说，这已经是不可能的天文数字了。不过这些条件并不是打动 Anders 的主要原因，Bill 最后提出的条件是"答应给 Anders 一个小组和极多的资源，让 Anders 尽情地发挥"。这个条件可说是打到了 Anders 的心底，因为 Anders 正渴望有人能够支持他完成新的计划和想法。我想，在软件产业中大概也只有 Microsoft 能够拥有这种雄厚的资源可以挖角任何人吧。

在 Bill Gates 提了这样的条件之后，Borland 再也没有本钱能够和 Microsoft 进行比价了，只好眼睁睁地看着 Anders 离开 Borland 前往 Microsoft 再开创下一个人生的高峰。在 Anders 到了 Microsoft 之后，Bill Gates 果然重用 Anders，也立刻让 Anders 负责激活 Microsoft 的下一拨开发工具计划，当然这个计划也是 Microsoft 对抗 SUN/Java 的整体计划之一。Anders 到了 Microsoft 之后立刻展现了实力，让 Microsoft 的编译器技术又精进不少，最明显的例子就是 Microsoft 后期的 Java Virtual Machine 是 PC 上执行效率最好的，而且在 2、3 年后，VJ++编译出来的虚拟机械码的执行效率不但比任何的 Java 开发工具还快，在某些方面甚至比原生的 Windows 开发工具，例如 Delphi、VB、甚至是 VC++还有效率。这真是令人震撼，当然 Anders 为 VJ++打下的基础现在也展现在 NET 的 C#编译器以及.NET 的 JIT(Just In Time)编译器之上，.NET 的 JIT 在许多程序代码最佳化方面比 Delphi 还先进。因此在 2、3 年前当 VJ++即将推出之际，在 Borland 内部也引起了非常大的骚动，并且严阵以待，当然这又是另外一个故事了。

对于 Anders 来说，到了 Microsoft 之后不久又再次登上了生涯的另一个巅峰。因为当初 Anders 在 Borland 之时，就有如孙子兵法中叙述的"藏于九地之下"，虽然是不世出的天才，但是仅为少数的人所知，即使是使用 Borland 产品的人在当时可能也不知道 Anders 这号人物。因为 Anders 和 Borland 的作风很像，都是行事低调，不到最后绝不随意出手。但是 Anders 被挖角到 Microsoft 之后，由于 Microsoft 的企业文化向来是前进、积极的侵略性方式，因此 Anders 也就转变为"动于九天之上"，负责 Microsoft 开发工具大军的核心大将，不但广为人知，成为许多软件人员效法的对象，而且屡获大奖。他不但获得了信息软件业界的推崇，最后也终于获得了信息学术界的认可，可说是实至名归。

Anders 的离开对于 Borland 来说是一个很大的损失，不过对于 Delphi R&D 小组来说却是有好有坏，因为 Delphi 开发小组虽然失去了最重要的支柱，但是也给 Danny Thorpe 一个快速崛起的机会，在 1 年后，Danny 果然立刻成为了 Delphi/C++Builder/Kylix 中最杰出的人物之一(另外一个当然就是 Anders 的老搭挡 Chuck Jazdzewski 了)，Danny 也是我认为目前在 Borland RAD(注)部门中功力最厉害的人物。在稍后的内文中我会对 Danny 进行比较详细的说明。

注：Borland RAD 部门是指 Borland 的 Rapid Application Development 部门，RAD 负责的产品包含了 C++Builder、Delphi、Kylix 以及未来的.NET 以及 Mobile 的新产品。

巅峰之作和最后的胜利者

在 Zack 于 Delphi 3 开发的后期和 Paul Gross 逐渐取得了主控权之后，Delphi 的发展方向也偏向了 Zack 希望的 Multi-Tier 的技术以及由 Paul Gross 稍后主导的 Borland Golden Gate 计划。1997 年，Delphi 还是像 Delphi 1/2 一样以强劲的生命力，以一年一个大版本的速度准时推出了。当时 Delphi 3 也称为 Delphi 97。Delphi 97 和 Delphi l/2 一样有一个最重要的战略目标以及主要的革新技术，那就是 Multi-Tier 和 COM/DCOM 的支持。因此，当时 97 推出之际使用的行销术语是"Component Foundry"，意指 Delphi 97 能够开发任何的软件组件(事实上是指 VCL、COM/DCOM 以及 ActiveX 组件)，而且使用组件来开发 Multi-Tier 的应用系统。

Delphi 97 无论是在品质、功能或是市场和销售上都是非常成功的。也许是为了证明即使是没有了 Anders，Borland 的 Delphi R&D 小组仍然有实力开发出优秀的 Delphi 产品，因此 Delphi 3 一推出的品质就在高水平的地步，后来 Delphi 97 销售的结果也证明了它没有令人失望，Delphi 3 的推出让 Delphi 一举突破了 150 万套的销售大关，也几乎成为 Borland 有史以来最畅销的单一系列开发工具，我也认为 Delphi 3 是 Delphi 所有版本中最好的一个。

不过，Delphi 97 在即将推出之时也令一些人感到忧虑，因为在当时几乎没有任何开发工具强调 Multi-Tier 的开发，大多数的开发工具仍然着重在 Client/Server 的功能。Delphi 97 将许多新功能集中在 Multi-Tier 的开发的确是一个不小的冒险，因为在当时 Multi-Tier 的观念还非常的新颖，许多人对于 Multi-Tier 是什么？能够用来开发什么？为什么要使用 Multi-Tier 等等问题都不是很清楚。更重要的是，在那个时候 Multi-Tier 的基础工程都尚未完全准备好，使用者要如何使用 Multi-Tier 技术来开发应用系统、甚至是学习 Multi-Tier 的技术呢？

我所谓的"Multi-Tier 的基础工程都尚未完全准备好"，是指在那个时候 Microsoft 的 DCOM 技术都还没有推出。那么，Borland 该如何在 Windows 平台上提供 Multi-Tier 应用系统需要的分布式技术呢？说到这里，我们也不得不佩服当时 Borland 这些 Delphi R&D 小组的眼光，他们几乎已经精确地看到了未来的软件计算世界将会由网络和分布式技术主导，因此，即使是在操作系统平台尚未准备好的情形下，Borland 也决定快速向前出发。

另外一个驱使 Delphi 97 走向 Multi-Tier 的重要原因便是当时 Borland 已经准备走向组件技术的领域，准备在中间件(Middle-Tier Software)中成为一个关键的软件开发厂商，这就是 Paul Gross 和 Zack Urlocker 激活的 Golden Gate Strategy。而 Paul Gross 也就是由于 Golden Gate 计划扬名立万、进而成为当时 Borland 整个研发部门的副总裁(Vice President)，后来又成为 Microsoft 下一个挖角的对象。

为了配合 Golden Gate 计划并且成为分布式技术的领导厂商，Borland 由 Paul Gross 主导并购了 Boston 一家有名的顾问公司，这家顾问公司拥有一个颇为知名的中间件"Entera"。Entera 是一个以 RPC(Remote Procedure Call)通信协议为骨干的中间件，Borland 希望通过 Entera 快速进入中间件的市场。由于 Entera 的客户端能够执行在 Windows 的操作系统中，因此，Entera 在 Windows 平台上拥有基础的分布式技术支持能力，这刚好提供了 Delphi 97 需要的技术，所以 Delphi 的 R&D 小组立刻使用 VCL 封装了 Entcra 的底层 API，提供了一个称为 OleEnterpriseConnection 的 VCL 组件，让 Delphi 97 能够和以 RPC 为基础的中间件连接。

因此在 Delphi 97 中 Borland 提供了三种分布式连接组件，分别是使用 DCOM 技术的 DCOMConnection、使用 TCP/IP 技术的 SocketConnection 以及 OleEnterpriseConnection。在那个时候最成熟的技术应该是 OleEnterpriseConnection，而使用 DCOM 技术的 DCOMConnection 在 Microsoft 本身 DCOM 尚不成熟的背景之下，表现得并不好。至于 SocketConnection，我认为 Borland 一直没有很认真地地实现 SocketConnection，因此最好只可以用来作为学习的对象，尚未到达可以真正应用的水准。不过稍后随着时间和技术的进步，OleEnterpriseConnection 逐渐没落，而 DCOMConnection 则反而因为 Microsoft 的 DCOM 的慢慢改善而成为比较实际的应用对象。当然，这些又属于 Delphi 4 的开发故事了。

虽然 Delphi 97 采取了比较冒险的做法，但是由于其品质良好，又支持 COM/VCL 组件的开发，因此也很快征服了 Delphi 使用者的心而大获成功。当时许多 Delphi 的使用者仍然是以开发 Client/Server 应用系统为主，不过由于 Delphi 97 好用的 COM 技术以及应用 Web 的 WebBroker 技术，再加上 Multi-Tier 的功能，都让使用者大呼过瘾，把 Delphi 作为学习这些先进技术的好工具，因此 Delphi 97 在当时到达了巅峰的地步。而 Delphi/VB 和 PowerBuilder/Gupta 缠斗的情形也逐渐明朗。Delphi/VB 的销售量和市场占有率已经远超过 PowerBuilder/Gupta 了，PowerBuilder/Gupta 也注定将从通用开发工具以及主从架构开发工具市场中让出绝大的市场份额。

其时 PowerBuilder 仍然不肯服输，并且坚称 PowerBuilder 在总收入方面仍然是第一。

这是因为 PowerBuilder 在当时仍然非常的昂贵，比 Delphi/VB 至少贵了两三倍以上。但是在不断地快速流失市场的情况下 PowerBuilder 的售价很快就出现了大幅的下降，至此 Delphi 和 VB 终于获得了全面的胜利；Client/Server 开发工具的战争已止，但是接下又上演了另一出精彩的 Java 开发工具大战。

危机的开始

在 Delphi 3 再次获得了胜利之后，Delphi 4 决战的目标已经从 Delphi、VB、PowerBuilder 和 Gupta 混战的形势转变为和 VB 对战的局势。原本 Delphi 有继续进击的机会，因为在当时 Delphi 3 的功能的确强劲，而且许多功能都领先业界至少半年以上的时间，不过可惜的是，在 1996 年 Borland 担任 CEO 的 Delbert Yocam 先生正好在这个时候开始了改造 Borland 的计划，几乎造成了对 Delphi 不可挽回的错误，这要从 Zack Urlocker 被 Delbert Yocam 拔擢成 Borland Marketing 的副总裁开始说起。

1996 年 11 月，Borland 邀请了 Del Yocam 入主 Borland 成为新的 CEO，希望通过 Delbert Yocam 在管理方面丰富的经验改善 Borland 长久以来在公司管理方面的积弱不振。Delbert Yocam 原本是 Apple 计算机公司的副总裁，应该是拥有丰富的管理经验，不过不知道 Delbert 是不是在大型计算机公司呆惯了，因此花钱花得特别凶。Delbert 在一进入 Borland 之后立刻花了大钱更换总裁办公室里所有的家俱和装潢，其后到各地出差时又要求头等舱以及总统套房，花钱像流水一样。当时我便对这位新的 CEO 没有什么好印象。

Delbert 在进入 Borland 之后一直想改造 Borland。在 Delphi 3 大获成功之后，Delbert 便把重点集中在 Delphi 的身上。Delbert 认为 Delphi 非常成功，是 Borland 的摇钱树，因此一直想从这个产品线获得更多的收入。好大喜功的 Delbert 觉得 Delphi 的根基已经很稳固，因此可以更快速地从 Delphi 身上取得资源。这让 Delbert 在 1998 年下了一个严重的错误决策，几乎折损了 Delphi 的大好基业。

Borland 和 Microsoft 的法律大战

在 Microsoft 不断地挖角，甚至把 Anders 也挖走之后，Borland 已经到了"士可忍，孰不可忍"的地步。Microsoft 挖角的动作愈来愈像是恶意挖角，似乎是想把 Borland 搞垮。Borland 无法忍受的是 Microsoft 不但在操作系统和开发工具上进行不公平的竞争，现在甚至进行人事上的不公平竞争。Microsoft 不思在产品上和 Borland 一决雌雄，反而进行挖墙角的破坏，Borland 实在是忍无可忍了。

1997 年 5 月，Delbert Yocam 终于向美国法院提出了对于 Microsoft 的控告。当时 Borland 提出的控诉理由是"Microsoft 对于 Borland 进行 Brain Drain"的行动。Microsoft 在 30 个月内从 Borland 挖角了 34 名 Borland 关键的开发者，而且每一名被挖角的人到了 Microsoft 之后都担任和在 Borland 类似的角色，这摆明了就是要利用这些人在 Borland 的知识快速地帮助 Microsoft 提升和 Borland 的差距，并且了解 Borland 内部的产品和技术的开发状况。这种不公平的竞争应该是没有任何一家公司能够容忍的。

在 Borland 正式提出了诉讼之后，具备侵略性企业文化的 Microsoft 当然就立刻反击，和 Borland 对簿公堂。Borland 和 Microsoft 从开发工具战场一直斗争到法律战场，互不相让的情形让许多看客大呼过瘾。Borland 对于 Microsoft 来说似乎就是打不倒的勇者，不管情势再艰辛，仍然能对抗到底，因为 Borland 知道，对于 Microsoft 一旦让步，以后就永远没有机会了。当时，业界许多人都已经预测 Borland 迟早会采取行动，但是没有想到这么快就勇敢地响应。Borland 当时的行动引起了许多信息业界的尊敬和支持。

这场官司的进行很快就有了初步的迹象，所有的证据都对 Microsoft 不利。Microsoft 一看情势不对，又不想让 Borland 真的消失，以避免吃上在开发工具市场垄断的官司，

所以立刻表达愿意和 Borland 在庭外和解。正是由于这个原因，才有后来 1999 年时 Microsoft 对于 Borland 的投资，Microsoft 这项投资正是这次庭外和解的条件之一。
不过 Delbert 的运气并不好，虽然 Microsoft 愿意和 Borland 进行庭外和解，但真正和解的动作以及 Microsoft 对于 Borland 的赔偿事项却是发生在 Delbert 之后的 Borland 下一任 CEO，即 Dale Fuller 身上，Delbert 种下的稻穗却是由 Dale Fuller 来收割。
接二连三的错误决策
1998 年，Delbert Yocam 再次展现了好大喜功的本性，在没有充分地讨论以及共识之下，Delbert Yocam 决定把公司名称从众所周知的 Borland 改为 Inprise。Delbert 决定如此做有数个原因：
" 由于 Paul Gross 和 Zack Uplocker 的 Golden Gate 计划让 Borland 进入中间件市场，因为 Borland 在企业市场以往没有很强的知名度，许多人认为 Borland 只是开发工具厂商，因此 Delbert 为了解决这个问题决为 Borland 取一个新的名称
" 使用 Inprise 的意思是指 Borland 可以 Integrating The Enterprise。为提供企业整体解决方案的软件公司
" 基本上 Delbert Yocam 在进入 Borland 之后已经开始改变 Borland，悄悄地进行第二次 Borland 改造的行动。这就是以行销为主的 Borland，有别于以往 Philippe Kahn 所领导的以产品/技术为主的 Borland。为了代表 Delbert 的决心并且重新出发，Delbert 认为公司该使用一个新的名称
1998 年 4 月，在 Delbert 的主导之下，Borland 花费了数百万美元之后终于改名成为 Inprise 公司。不过 Delbert 原本是想在更改公司名称之后可以重新出发，但是没有想到，在 Borland 改名为 Inprise 之后，各种负面的效果却接踵而来。
首先，传统的 Borland 使用者都强烈反对 Inprise 这个名称，这些 Borland 使用者都喜欢原来的 Borland。第二个问题，是许多新的使用者都听过 Borland，但是在改名之后这些新的使用者找不到 Borland，以为 Borland 已经不见了，又从未听过 Inprise 这家公司。第三，则是竞争对手刻意放出的讯息，故意散布 Borland 已经被一家叫做 Inprise 的公司并购了，因此希望原先的 Bor]and 用户能够放弃使用 Borland 的产品。
Delbert 万万没有想到，在花了大钱更改公司名称之后，Inprise(Borland)却开始得疲于奔命地应付各种不利的后果。结果是赔了夫人又折兵，不但浪费了资源却无法在企业市场闯出名号，又折损了 Borland 的金字招牌。
另外一个有问题的决策是把 Zack Urlocker 拔擢成 Borland 行销部门的副总裁。由于 Zack 在 Delphi 3 出色的表现，以及和 Paul Gross 激活的 Golden Gate Strategy 让 Delbert 心动，并且 Delbert 认为通过 Golden Gate Strategy 可以让 Borland 打入企业市场，因此 Delbert 对 Paul Gross 和 Zack Urlocker 都印象深刻。不久之后，Delbert 分别提拔了 Paul Gross 为 Borland 开发工具部门的副总裁、Zack Urlocker 为行销部门的副总裁。
本质上 Zack 是一个非常好的开发者，对于产品也有很敏锐的感触，应该是非常理想的产品线管理人物，Borland 应该让 Zack 好好地呆在 R&D 部门，为 Borland 的产品线运筹帷幄，好好地开发 Borland 以后的产品。只可惜的是"水往低处流，人往高处走"，副总裁的位置放在面前，Zack 当然想升官发财(谁不是呢？)。不过 Zack 并没有想到自己的优缺点。他固然是出色的开发人员和产品开发人员，但是对于行销却是门外汉。在 Zack 做了行销副总裁之后，Borland 的 R&D 小组不但失去了一员大将，更糟糕的是 Zack 似乎也开始向 Delbert 学习，慢慢有了好大喜功的做事方式。
首先，Zack 扩充 Borland 行销人员到达了 100 多人的数目。可是，当时全世界 Borland 的员工才将近 1000 多人，行销的人员居然超过九分之一的比重，实在是太夸张了。不过，这么多人的部门在当时仍然没有做出什么好的行销工作，仍然被 Borland 的使用

者抱怨。我还记得当时我曾向行销部门要求所有开发工具的市场竞争资料，结果行销部门只说没有这种资料，当时我还很生气，这么庞大的部门居然连这么基本的竞争信息都没有。事实上，当时 Delbert Yocam 同意让 Zack 的行销部门招聘这么多人，除了是因为 Zack 很红之外，和 Delbert 想改善以往 Borland 做得很烂的 Marketing 工作也有很大的关系。Delbert 认为 Zack 在产品线方面表现得很出色，因此也希望通过 Zack 的能力来进行 Delbert 对于 Borland 行销方面的改善工作。

可惜的是 Zack 上任之后表现得不如预期，大家对于 Zack 的表现也是贬多于褒。很快，Zack 就失去了以往在 R&D 小组时的自信满满，开始逐渐消沉，最后终于离开了 Inprise (Borland)。Zack 从因为 Borland C/C++3.0 时的 OWL framework 快速窜起，在 Delphi 3 时达到生涯的高峰，一直到以行销副总裁之尊黯然离开 Borland，真是令人感慨。如果 Zack 能够清楚了解自己的优缺点，不要去接行销的工作，而继续在 R&D 部门发展，也许他会有更好的成果。从 Zack 的奋斗过程，我认为程序员也许应该想想自己未来的发展方向，好的技术人员一定是好的管理或是行销人才吗？

一开始我知道 Zack Urlocker 这号人物，是在数年前我还在 Georgia Institute of Technology 念书时从那时著名的"Windows Tech Journal (WTJ)"得知的。当时 Zack 在 WTJ 上一直有 Object Pascal 的专栏，写的内容都非常好，深深地吸引了我。因此当时每个月初都开车大约半小时到最近的计算机商店购买当期的 WTJ，目的就是为了阅读 Zack 的文章，在那个时候我就认为这个家伙真是厉害。

据我所知，Zack 在 1999 年离开了 Borland 之后，加入了 Active Software 以及数个其他的软件公司，大都是担任行销方面的高阶主管。Zack 在其后的数个职位上表现得不错，不晓得是不是因为在 Borland 时缴了大量学费而学习到的知识。不过不管如何，我仍然认为 Zack Urlocker 是一位值得尊敬的人物，因为他至少在一生中开发了 2 个重要而且成功的产品"Borland C/C++和 Delphi"，本身的技术水准也很高。相信 Zack 也将永远记得 Borland C/C++和 Delphi，这两个产品是 Zack 一生的成就和骄傲。再见了 Zack Urlocker，相信许多的 Borland C/C++和 Delphi 的使用者都会记得你的。

自巅峰而下--Delphi 4

中国人一直不喜欢"4"这个数字，认为它不吉利。难道说"4"对于 Borland 来说也是一个挥之不去的噩梦吗？当初的 Borland C/C++4．0 对 Borland 来说是永难忘怀的噩梦，到了 Delphi 4，难道 Borland 又要重蹈覆辙吗？

时值 1998 年，是下一个 Delphi 版本应该要推出的年份，也是 Delbert Yocam 进入 Borland 当 CEO 的第 2 年。对于美国企业的 CEO 来说，第 2 年是 CEO 向董事会显示经营绩效以及缴

出成绩单的时候了。Delbert 为了能够缴出靓丽的成绩单，因此在 1998 年决定必须拉高 Borland 的营收，以冲高 Borland 的股票价格。但是，当时的 Borland 才刚进入组件和中间件的市场，尚未在企业市场占稳脚跟，因此 Delbert 决定在 Borland 的开发工具产品线中动脑筋。Delbert 做的决定就是强迫规定从 1998 年起，在每一个 Quarter(也就是每 3 个月)，每一个 Borland 产品开发部门都必须推出一个新产品，让 Borland 每一个 Quarter 都有新产品可以销售，以便增加 Borland 的营收。

不过，DelbeN 这个决定却相当的糟糕，这让 Borland 每一个产品部门的主管都面临了强大的压力，因为即使是产品还没有准备好推出，但是时间一到，不管产品品质如何，都一定要出货。Delbert 这个错误的决定让 1998 年又成为 Borland 的噩梦年。

Delphi 1、2 和 3 的时间间隔都是 1 年多一点，展现了 Delphi 强劲的生命力。依照原本的计划，Delphi 4 也应该是在 1998 年推出的。但是 1998 年 Borland 在内部开始研发数项新的科技和产品，加上 Microsoft 不断的挖角行动，都让 Delphi 4 的研发工作受到

了延迟。依照当时 Delphi 4 的研发时程，Delphi 4 最早应该在 1998 年的第 4 季才能够推出。但是很不幸的是，为了达成 Delbert Yocam 的要求，Delphi 4 在 1998 年的第三季就必须出货。在接近 1998 年的第 3 季之时，虽然 Delphi 的 R&D 小组仍然无法完成 Delphi 4，并且极力抗拒出货，无奈在 CEO 强大的压力下，Delphi 4 仍然必须在第 3 季准时出货。

从我个人的角度来看，当时 Delphi 4 的产品品质应该只到 Beta 2 的阶段，离真正能出货尚有一段不小的距离。Delbert 强迫 Delphi 4 推出，不但打击了 Delphi R&D 小组的士气，如此乱搞产品线，以外行领导内行的结果只是让 Delphi 砸坏了自己的招牌。不过站在 Delbert 的角度则又不一样了，因为对 Delbert 来说，如果在 1998 还无法冲高 Borland 的营收，那么 Delbert 肯定是要下台的，因此 Delbert 只有孤注一掷了。

1998 年的第 3 季，Delphi 4 果然被强迫推出了。虽然 Delphi 4 的新功能仍然亮眼，但是品质不稳的恶名也很快地出现在使用者的抱怨之中。随后，使用者的不满愈来愈强烈，Borland 面对四面八方的反弹不得不快速地做出响应，立刻开始着手 Delphi 4 Patch 的开发，以期快速修正 Delphi 4 的臭虫。依我的看法，Delphi 4 一直要到 Patch 2 才应该是当初 Delphi 4 出货时的品质。由于 Delphi 4 的反应不佳，因此未能再次把 Delphi 的销售量拉上新高，Delphi 原本锐不可当的气势也为之一挫。对于 Borland 来说，Delphi 4 的销售并没有增加太多的收入，Delbert 打的如意算盘当然也落空了。Delphi 4 的失败也严重地影响了 C++Builder 的品质和销售，Delbert 恶搞产品的开发之后不但又让 Borland 开始赔钱，终于也自尝恶果，在 1999 年被 Borland 的董事会开除。不过。无论如何，Delbert 的决策已经对产品造成了巨大的伤害。

虽然 Delphi 4 的诞生过程充满了困难，命运也很坎坷，不若它的兄弟般好命。但是 Delphi 4 却意外地向全世界揭露了 Delphi 另外一个 Architect 的庐山真面目，那就是 Chuck Jazdzewski。在 Delphi 4 中，使用者只要开启 About 对话盒，并且按下 "Alt+chuck"，那么就可以看到下图的画面。这个简短的画面是 Delphi R&D 成员之一偷偷使用 V8 录下来并且放入 Delphi 4 中的，这也是第一次 Chuck 露脸于全世界。Chuck 事先并不知情，而在以往的 Delphi 1/2/3 中放的人物图片则一直是 Anders。这些隐藏的有趣图片以及 Delphi R&D 开发小组的名单在 Delphi 中称为"Eastern Eggs"。

直到 1999 年，在费城举行 Borland Conference 时，我才真正有机会见到 Chuck，并且和 Chuck 当面讨论一些事情。Delphi 4 的失利让 Delphi 从巅峰的态势往下沉沦，要等到 Danny Thorpe 继 Zack 之后掌握 Delphi 的研发大权后才能再次向前挺进。

Delbert 最后的挣扎

由于 Delbert 错误的决策先后让 Delphi 4 和 C++Builder 4 失利，Borland 每季又开始亏损了。显然，Delbert 自己也心知肚明，如果再没有任何建树的话，他很快就要下台了。为了做最后的挣扎，Delbert 决定和其他公司合作。1998 年正是 Java 快速兴起的年份，SUN 在 JavaWorkshop 失利之后，便一直想找一个好的 Java 开发工具。而当时 Borland 发表了 JBuilder 2，虽然 JBuilder 还不是 Java 市场上最受欢迎的 Java 开发工具，但是 JBuilder 是最快速成长以及最受好评的 Java 开发工具。SUN 看到了 JBuilder 的潜力，因此对于 JBuilder 拥有强烈的兴趣。

SUN 显示了对 JBuilder 的兴趣，无疑给了 Delbert Yocam 打了一针强心剂，几乎在绝望中的 Delbert 似乎看到了一丝曙光。Delbert 很快便和 SUN 接触，看看 SUN 能够提出什么条件。Delbert 的如意算盘是让 SUN 花大钱并购 Borland，如此一来，JBuilder 不就自然成了 SUN 的产品了吗？由于在那个时候 Borland 的股价已经跌到了 3 到 4 美金之间，而 SUN 的股价却高高在上，大概是在 80 多美金。因此，如果 Delbert 能够促成合并，那么 Delbert Yocam 便可以大捞一票，甚至在并购之后，Delbert 还有可能成为 SUN 的副总

裁，继续位居要津。
不过世事不能尽如入意。SUN 只对 JBuilder 有胃口，对 Borland 其他的产品却没有多大的兴趣，因为 Delphi/C++Builder 等都不属于 Java 系列的产品。而且 Delbert Yocam 又狮子大开口，希望 SUN 以每股 20 几美元的代价收购 Borland 的股票，当场吓得 SUN 退避三舍，这件事情后来也就不了了之了。当然，Delbert Yocam 很不甘心，因为促不成这宗合并案子，再加上 Borland 被 Delbert 搞得乌烟瘴气，下台的命运也就不可避免了。也许是"天将降大任于斯人也，必先苦其心志，空乏其身"吧，Borland 在 Philippe Kahn 离开之后，历经了数任 CEO，但是一直没有找到真正好的 CEO，能够适当地带领 Borland 走向光明。不过 Delbert Yocam 似乎是黎明前的黑暗，在 Delbert 不名誉地离开 Borland 之后，Borland 也即将看到未来之光。
Danny 的接棒和决心
Delphi 4 的仓促推出果然在市场上反应很差，销量上也一落千丈。原本寄望能够再次获得好成绩让 Delphi 的总销售量再次冲上新高，并且为 Borland 带来更多的营收。但这一切都很快地幻灭了，品质不好的产品仍然得面对市场严厉的考验。在 Delphi 4 遭受了前所未有的失败、接着 C++Builder 4 也铩羽而归之后，Borland 又开始走下坡路了。Borland 好不容易通过 Delphi 带来的希望却在错误的决策下被牺牲了。在 1999 年 4 月 Delbert 终于被 Borland 的董事会扫地出门，结束了在 Borland 的日子。我认为，Delbert 在 Borland 将近 3 年的时间里，对 Bodand 造成了许多的伤害，其好大喜功的管理方式对 Borland 的产品线更几乎造成了无法弥补的伤害，是我所认为的最糟糕的 Borland CEO。更离谱的是在 Borland 的董事会开除 Delbert 后，他居然还以合约未满为由，要求 Borland 支付额外的谴散费，大捞了一票，真是人心不古，工作做得如此差却还有脸提这种要求，在最后一刻仍然压榨 Borland。
在 Delphi 4 的伤害造成之后，Delphi R&D 小组要面对的是如何收拾残局，并且想办法解决造成的问题。在这个时候 Chuck 由于把精力放在 Borland 另外秘密的产品和技术的研发上，因此无法花太多的时间在 Delphi 5 的研发上。此时，在 Delphi 上一向表现良好的 Danny Thorpe 便逐渐挑起了 Delphi 的重负大任。
Danny 在 Delphi 2 之后便有大将之风，开始负责 Delphi 最低阶的编译器以及 RTL(Run-Time Library)的工作。Danny 是美国 San Diego 大学毕业的，主修就是编译器技术。
在 Delphi 4 之后，Danny 几乎成了 RAD 部门主要的 Architect，负责了 RAD 大部分产品的研发工作，甚至又成为 Microsoft 再次挖角的对象。
对于 Danny 来说，如何重塑 Delphi 5 让 Delphi 从失利中重新站起、找回昔日的光荣便是一个非常重要的工作。在 Delbert Yocam 于 1999 年离开 Borland 之后不久，现任的 Borland CEO Dale Fuller 先生便被 Borland 邀请加入成为 Borland 的代理 CEO，希望能够通过 Dale Fuller 的经验挽救沉沦中的 Borland。在 Dale 入主 Borland 之后，首先展开的工作除了整顿 Delbert 在位时形成的庞大无用的行销部门之外，在产品线方面则是看好 Linux 的未来，要求 Borland 的 RAD 部门必须开发出 Linux 下的 RAD 工具。
在 Danny 接掌了 Delphi 主要的开发责任之后，又和 Chuck 一起再次形成中坚的 RAD 精英份子。Chuck 主要负责新技术和新构想的实验作品，而 Danny 则是负责困难的编译技术以及 RTL。由于 Turbo/Borland Pascal 以及 Delphi 的最佳化编译器都是 Anders Hejlsberg 撰写的，因此当 Anders 离开 Borland 之后几乎没有人能够维护编译器程序代码。Anders 都是使用汇编语言(Assembly)撰写复杂的编译器程序代码，而且其品质是如此之好，不但连 Chuck Jazdzewski 都赞不绝口，更麻烦的是当时 Borland 几乎没有工程师敢随便更动这些程序代码。
因此在 Anders Hejlsberg 于 Delphi 2 离开了 Borland 之后，Borland 立刻采取了数项行

动希望能够解决这个"烫手山芋"。Borland 决定的第一件事情是从 Delphi 的编译器抽离大部分最佳化的工作。因为要在 Anders 的程序代码再继续加入最佳化程序代码是 Borland 当时没有把握的事情。另外，由于当时 Borland 已经决定开发 C++Builder，而 C++Builder 也需要一个最佳化的编译器，因此，Borland 认为如果能够提供一个共同的后端最佳化编译器，那么 Delphi 和 C++Builder 不仅都可以使用，还能够解决没有人敢修改 Delphi 编译器的问题。这个决定就是后来 Delphi 3 以及 C++Builder 2 推出之后 Borland 宣称的"Delphi 和 C++Builder 可使用共同的后端最佳化编译器"，这个工作当时是交由 Borland 的编译器小组 Lee 他们负责的。

不过共同的最佳化编译器只解决了一半的问题，对于 Object Pascal 语言本身的改善仍然需要能够修改 Anders 撰写的编译器，那么到底谁能够进行这个工作呢？答案当然就是另外一个软件天才--Danny Thorpe 了。Danny 在接手 Delphi 的开发大任之后，就开始为已经停止开发一段时间的 Object Pascal 语言本身进行演进的工作。此外，Danny 也开始为 Delphi 底层的 RTL 进行改造，并且为 Delphi 的编译器加入更多最佳化的功能。

Danny 之所以同时在 ObjectPascal 程序语言、Delphi RTL 以及 Delphi 编译器进行渐进的改善工作，是有许多因素影响的。首先，当然是为了接下 Anders 留下的工作，另外一个原因是在 Delphi 3 之后，必须再次对于 COM 的支持进行强化。最后，是为下在 Delphi 4 之后，准备把 Delphi 移植到 Linux 上。事实上，Borland 在 Delphi 的 R&D 小组中曾经

一度准备把 Delphi 和 C++Builder 移植到 SUN 的作业平台上，这是为了因应 Delbert 和 SUN 合并时进行的准备工作。甚至 Delphi 的 R&D 小组认为，既然要开发跨平台的 Delphi 和 C++Builder，那么不如把 Apple 的 Macintosh 操作系统也纳入考虑。Delphi 的 R&D 小组在当时甚至已经列出了开发 SUN 和 Macintosh 平台的时间表，但是稍后随着和 SUN 合并计划的破灭以及 Delbert 的下台，这个跨平台的 Delphi 计划也就暂停了。一直等到 Dale Fuller 上台强力要求开发 Linux 平台的 RAD 工具之后，Delphi 的 R&D 小组才再次激活跨平台的计划。

为了支持更好的 COM 开发能力，Danny 修改了 Delphi 的编译器，直接支持 COM 接口的参考计数值(Reference Count)的维护工作，以免除开发者繁杂的程序代码，提供了类似 Visual Basic 的能力。同时 Danny 也在 Object Pascal 程序语言本身中加入接口(Interface)的机制，让 Object Pascal 和 Java 一样对接口程序设计都提供 First Class 的支持。Danny 并且更进一步，巧妙地结合 COM 的接口以及 Object Pascal 程序语言的接口，让 Delphi 的程序员更方便地使用和处理 COM 接口。Danny 的这些努力，就是 Delphi 的使用者在 Delphi 3 之后逐渐在 Object Pascal 中看到的 Interface 机制。对于非常熟悉 Delphi 的读者来说，应该可以发现 Delphi 1/2 中 Object Pascal 变化的部分很少，但是从 Delphi 3 之后，每一新版的 Delphi 在 Object Pascal 程序语言本身都有进步，这些都是 Danny 所做的努力。

在 RTL 方面，Danny 更是投注了大量的心血，Danny 的第一步是去芜存菁。Delphi 经过了三、四年的发展，许多 RTL 中的程序代码不是过时，就是需要进行最佳化的调整。因此从 Delphi 4 开始，Danny 便逐渐整理和改善 Delphi 的 RTL，这方面的成果从 Delphi 5 之后便逐渐显露出来，Delphi 的 RTL 不但在效率方面有了进步，更提供了愈来愈多以前版本的 Delphi 所没有的功能。当然，Danny 在 Delphi RTL 方面最大的贡献是改善 RTL 成为跨平台的基础。Danny 维护后的 Delphi RTL 最后也成功地移植到了 Linux 平台上，并且克服了许多 Windows 以及 Linux 平台差异处的困难。当然，Danny Thorpe 和 Chuck Jazdzewski 是 Kylix 得以推出的最重要的功臣之一。为了解决 Kylix 在 Linux

平台上许多的技术问题，后来还引起了 Linux 开发者社群围攻 Danny Thorpe 的精彩大戏，最后导致 Danny Thorpe 不再管 Kylix 的开发而全力投入.NET 的阵营，这当然又是另外一个极为精彩的故事了。

对于 Danny 来说，只有一个最重要的目标，那就是再次擦亮 Delphi 的光芒，让 Delphi 4 的失败能够在下一个版本中一雪前耻，并且把 Delphi 开发成最棒的 RAD 开发工具。Danny 的决心也让 Delphi R&D 小组再次安定了军心，在历经了 Delbert 错误的决策、Microsoft 大幅的挖角、Delphi 4 的失利之后，Danny 带领了一些新的 Borland 工程师展开了艰苦的工作。

Danny 的杰出表现早已深获许多人的赞扬和肯定，也充分地展现了继 Anders Hejlsberg 之后，Borland 另外视为宝贝的天才的风采。现在，Danny 不但早已独当一面，更成为了 Borland .NET 的 Architect，负责综合整理 Borland 未来在.NET 上的开发工具。在 2002 年 Borland 的 Conference 上，Danny 正式由 Borland 的 CEO Dale Fuller 先生颁发 Borland President Awards 大奖，这是继 Chuck Jazdzewski、Blake Stone 之后，Borland 第 3 个获得最高殊荣的 R&D 人员。在 Danny 接受大奖之时，现场所有的 BorCon 参

加人员都起立热烈鼓掌，看来，即使在 Borland 没有颁发这奖项之前，Danny 早已被所有了解他的贡献的人所肯定和钦佩，这只是一个迟来的奖项而已。

我曾在 1999 年费城的 BorCon 见到了 Danny，并且在澳洲举行的 BorCon 和他有简短的对话。Danny 的身材不算高大，瘦瘦的，但是非常的温文尔雅。和 Danny 讲话是一件很舒服的事情，因为你可以问他许多技术的问题，只要 Danny 有时间，他会很乐意和你讨论的。

恭喜 Danny!Borland 又为 PC 软件界培养了一个天才和明星。我相信 Danny Thorpe 也将成为许多开发者学习的对象，当然也包括我在内。

和对 Anders Hejlsberg 一样，最后再让我整理一下 Danny Thorpe 对于 Borland 和产品线重要贡献和获得的殊荣，让读者也能对这位值得尊敬的软件开发人员致敬：

" 负责开发 Delphi RTL/编译器困难的工作
" 改善 Object Pascal 程序语言，加入现代语言元素--Interface
" 开发出 Kylix 并且解决 Linux 平台的臭虫
" 1999 年被 Borland 内部评选为全 Borland 最重要的 50 人之一，是 Borland 不可放弃的人才
" 2001 年荣升 Borland .NET Architect
" 负责开发 Borland .NET 下一代整合开发工具--Galileo
" 和 Chuck Jazdzewski 共同开发代号为 Charlotte 的下一代 Web Service 程序语言
" 2002 年于 BorCon 获 Borland President's Awards 大奖殊荣

对于我来说，Borland 孕育了无数的伟大软件工程师，当然有一些人我无缘认识，因此对于这些人，我只能说是"久仰大名"，例如 Windows 平台的系统和除错大师 Matt Pietrek。但是有一些人却是我认识、甚至有过对话的。这些人每一个都令我折服，也让我向往这些伟大软件工程师到达的境界，他们是：

" Borland C／C++、dBuilder 的 Framework 大将 Carl Quinn
" 不世出的软件天才 Anders Hejlsberg
" Borland 首席科学家 Chuck Jazdzewski
" Borland RAD 核心支柱 Danny Thorpe

当然，还有本书稍后会叙及的 Java 天才，Mr. Blake Stone。

重回基本的精致之作--Delphi 5

1999 年 8 月，距离 Delphi 4 推出将近一年之后，由 Danny 领军的 Delphi 5 终于准备推出了，这次没有 CEO 不合理的要求，Borland 又投入了适当的资源，再加上 Danny 和 Delphi R&D 小组的全心开发，Delphi 的开发步调又回到了以往的轨道。在 Danny Thorpe 的细心和坚持之下，Delphi 5 在推出时的完成度是非常高的。

注：Borland 的每一个产品在推出之前都会进行完成度测试和评估，这和其他的软件是一样的。每一个产品在完成度到达多少才推出是不一定的，一般来说在 85%以上就是不错的产品，低于 80％就推出的产品则等于是推出 Beta 版的软件，品质一定不好。

当 Delphi 5 推出到市场之后，其品质果然又受到了 Delphi 使用者的喜爱，销售数字也证明 Delphi 5 已经成功地扫除了 Delphi 4 时埋下的阴影。再加上当年的 JBuilder 3 又迈入了一个全新里程，打造成了一个完全由 Java 撰写的 Java 开发工具，因而大获市场肯定，进而正式为 Borland 在 Java 市场带来了空前的胜利和大量的收入。Borland 又开始产生盈余了，Delphi 和 JBuilder 从 1999 年开始也正式成为了 Borland 的两大摇钱树。

不过 Delphi 5 虽然成功，但是从销售数字来看，Delphi 的销售几乎已经到了顶峰，不易再高度成长并且带来更多的收入，这从其他 Windows 传统开发工具的情形也可以看得出来。所以对于 Borland 来说，应该要开始为 Delphi 准备下一代的功能和平台了，重新设计 Delphi 的所有功能和 GUI 接口，再次地演进 Delphi 的风貌。即使 Delphi 也和 Visual Basic、PowerBuilder 一样即将走入下一代的开发环境，但是 Borland 仍然有责任在世代交替之间提供 Delphi 使用者顺利的移植方案。可以很明显地看到，Delphi 6 提供了 Web Service 功能让以前和未来的应用程序能够相互整合，Delphi 7 则将提供 Microsoft .NET 平台开发的功能。Delphi 6 和 Delphi 7 即是为 Delphi 的开发者走向未来提供的垫脚石。

Delphi 5 应该是 Delphi 3 之后最好的一个 Delphi 版本，称为 Windows 平台下最好的 RAD 开发工具是当之无愧的。虽然 Borland RAD 小组持续的开发最好的 Windows 开发工具，但是在 Windows 平台上的开发模式却已经悄悄地进行了自 DOS 以来最大的改变，那就是 Microsoft 为了因应 Java 的攻势而展开的 .NET 计划。Windows 平台上的开发概念、开发工具和开发技术即将揭开新的序幕，Microsoft 也准备逐步淘汰原生的 Windows 开发工具。

PC 平台上的开发工具从 DOS、Windows 到未来的.NET，一共历经了数个阶段。在每一个阶段，开发工具的市场都有着群雄逐鹿、竞争惨烈的战况。开发工具市场几乎是所有软件中竞争最激烈的。从 DOS 时代 PC 的开发工具市场有着数 10 家软件厂商竞争，到了 Windows 平台只剩下 10 几家、最著名的也就不过五六家，再延续到未来.NET 的平台，举是轻重的开发工具开发商可能会只剩下 Microsoft 和 Borland。

上面的表格是我整理从 DOS、Windows 第 1 个阶段、Windows 第 2 个阶段以及未来.NET 平台下的开发工具竞争情势。从上表中，读者可以看到不同的阶段主宰流行的程序语言、以及最后在开发工具市场胜出的厂商。开发工具情势的演变是 Microsoft 和 Borland 仍然将主宰这个市场，其他的厂商只能占有极小的市场和影响力。Borland 也证明了只有她能够和 Microsoft 抗衡，也是在 Windows 平台下，除了 Microsoft 之外唯一的独立开发厂商。我相信 Microsoft 和 Borland 仍然将在.NET 平台的开发工具领域中缠斗下去，虽然 Microsoft 已经率先推出了 Microsoft Visual Studio，但是 2003 年第 2 季 Borland 也将推出全新的集成开发环境--代号为 Galileo 的产品作为响应，一场精彩的龙争虎斗又将在 2003 年展开序幕。

具有讽刺意味的是，数年前许多人质疑 Borland 是否能够活下去，许多人也因为担心 Borland 的未来而不敢使用 Delphi。但是现在证明，Borland 不但成长得愈来愈好，Delphi 还继续推出了新版本 Delphi 7，Delph 8 也在 Borland 的计划之列。Delphi 7 将

是 Windows 平台下最好的原生 Windows 开发工具，也是撑到最后一个的 RAD 开发工具，连 Microsoft 都没有做到。Delphi 7 不但延续了原生 Windows 应用系统开发的生命，也为未来.NET 平台的开发做了铺路和转移的准备。Delphi、VB、PowerBuidler 和 Gupta 经过了数年的大战之后，结果证明，Delphi 才是撑到最后的英雄。

^v^v^v^v^v^v^v^v^v^v

第四章　未完之传奇

"成功产品的背后有着更多不为人知的秘密!"

Chuck 的秘密计划

Chuck 像个藏镜人。虽然他始终是 Delphi 最重要的三个人物之一，但是却一直不愿意站在最前线面对大众，而宁愿躲在幕后进行令人惊讶的软件革新工程。Chuck 进行的许多开发和研究并不广为人知……

当 Anders 离开了 Borland 之后，Chuck 短暂地成为 Delphi 的总 Architect。不过，Chuck 负责 Delphi 的开发工作后不久，就把 Delphi Architect 以及重要的工作交由 Danny 来负责，因为 Danny 早已显示了大将之风，成为 Chuck 最为信任的软件专家。而 Chuck 呢？虽然他仍然负责 Delphi 中许多重要的工作，但是后来大部分的时间是花在新技术和新产品的秘密研究之中。对于一些 Delphi 例行性的工作，Chuck 并不会花费太多时间。

在 Delphi 3 的研发阶段，Chuck 的主要精力并不是在 Delphi 3 上，因为 Danny 和 Zack 负责得很好。Chuck 当时主要是进行两件重要的研发工作，即 Delphi 的 Java 编译器以及 Apollo 计划。

原来，在开发 Delphi 3 时，Anders 和 Chuck 都已经预知了 Java 将来必会成功，成为重要的语言和软件技术。因此 Anders 和 Chuck 都知道必须在 Java 方面进行一些因应之道，以未雨绸缪保持 Delphi 的竞争力。后来 Anders 离开了 Borland，而 Chuck 则选择了投入资源研究 Delphi 和 Java 的整合技术。当时 Chuck 的想法是为什么不能够开发一个类似 Java 的 JVM，直接把 Delphi 的程序代码转换为 Java 的 ByteCode，让 Delphi 的应用程序直接在 JVM 中执行呢?甚至，当时 Chuck 还想，为什么 Borland 自己不开发一个 Delphi JVM，让 Delphi 也可以执行在 Windows、Linux、Solaris 和 Mac OS 之中呢？

有了这个疯狂的想法之后，Chuck 立刻要求 Borland 的高层批准这个研究计划，让他能够有资源进行研究工作。由于当时正值 Anders 因为不满在 Borland 没有足够的研究资源而离开了 Borland，进入 Microsoft 一展心中的鸿图。因此，Borland 高层当然不愿得罪 Chuck，以免他也离开 Borland。由于当时 Delphi 3 的进度在掌握之中，而且 Delphi 为 Borland 带来了大量的资源，因此 Borland 高层批准了 Chuck 的这个不可思议的计划，让 Chuck 开始了研究之路。

Chuck 有了资源之后，就立刻投入研究的领域，在 Delphi 3 的开发末期也没有花太多的时间在 Delphi 上，反而加速地和 Borland 的编译器小组为 Delphi For Java 编译器进行研发工作。在 1997 年中左右，Chuck 有了初步的成果，已经能够把一些简单的 Object Pascal 程序直接编译成 Java 的 ByteCode、并且执行在 JVM 之中。这实在是一个不小的突破，因此在当年的 BorCon 1997 中，Borland 和 Chuck 正式对外公开了这个技术，立刻引起了 Delphi 使用者强烈的兴趣，因为这代表一旦这个编译器研发出来，那么 Object Pascal 便立刻成为一个像 Java 一样的跨平台程序语言，而且，如果 Borland 能够继续把 VCL 和 RTL 移植进来，那么，Windows 平台的 Delphi 程序员可以通过这个技术同时开发多个平台的应用程序，这实在是太美妙了。

当 BorCon 公开了这个技术之后，Borland 立刻面临了愈来愈多的 Delphi 使用者的询问以及要求 Borland 尽早推出这个技术的压力。当然，Chuck 以及 Delphi 研发小组也非常兴奋，因为这代表 Delphi 又将有新的市场以及新的成长动力，所以 Chuck 立刻要求

Borland 投入更多的资源，以加速研究这个 Delphi For Java 编译器以及相关的研究工作。

不过，此时却发生了两件事情，最终让 Chuck 放弃了继续开发 Delphi For Java 编译器的意图。首先，Chuck 和 Borland 的编译器开发小组发现 JVM 似乎和 Java 语言系结得太紧密，以致 JVM 的许多伪指令都和 Java 语言的架构系结在一起，无法轻易地由其他语言来提供类似 Java 语言的架构，除非修改这些语言架构来仿真 Java 语言的架构。这个原因造成了当 Chuck 想把 Object Pascal 一些复杂的数据类型和语言架构编译成 Java ByteCode 时发生了极大的困难。

第二个决定性的原因是，由于当时 JBuilder 已经表现得愈来愈好，Borland 希望投入更多的资源到 JBuilder 小组，而且不希望有其他的产品或是技术影响 JBuilder 的成长，因此，Borland 高层对于 Delphi For Java 编译技术的开发也没有很大的兴趣，再也没有批准更多的资源给 Chuck。最后，Chuck 的这个 Delphi For Java 编译计划便宣告终结了。这实在是件可惜的事情，不过，当时 Chuck 研究的东西并没有白费，因为现在 Delphi 小组也根据当时 Chuck 研究的成果来开发.NET 上的编译器，希望通过以前投入的资源和经验来开发更好的 Delphi For .NET 编译器。

另外一个 Chuck 在 Delphi 3 开发阶段秘密进行的研究计划则更为重要了。当时我更期望这个技术能够出现在市场之上，不过可惜的是，最后也由于 Borland 高层要求 Chuck 投入 Kylix 的研发工作而一直拖延到今日都还在软件实验室中，这就是属于 Data Component 技术的 Apollo 计划。

Apollo 项目的缘由要从 Delphi 2 开始说起。在 Delphi 2 开发时，Anders 一直想在 Delphi 中建立一个 Garbage Collection 的功能，而 Chuck 则希望继续扩充 VCL 的功能为 VCL 加入 Data Component 的能力。由于 VCL 使用的组件架构在连接数据时是使用数据感知组件(Data Aware)技术，但是许多真正使用面向对象技术的程序员反而对使用数据感知组件相当地反感，而且在大型面向对象项目中，数据感知组件也被证明是不适当的。因此 Chuck 为了赋予 VCL 开发大型面向对象项目的能力，决定加入 Data Component 技术。

所谓 Data Component 技术，是指 VCL 架构可以代表实际世界中的 domain 对象，这些 domain 对象可通过 VCL 的技术直接储存在数据库之中，或是从数据库中取出，类似 EJB 中的 OR Mapping(Object-Relational Mapping)技术。如此一来，Delphi 的程序员可以在 Delphi 中直接使用 VCL 组件来代表如员工和公司等实例(instance)，而且可以随时把员工和公司实例储存到数据库中，再从数据库中取出员工和公司成为对象，而不需要使用数据存取对象直接处理数据库中的数据。Chuck 早在五六年前就想在 Delphi 中实现目前 Bold 等公司提供的 Object Instance 技术。

没有想到，就在 Chuck 进行 Apollo 项目到了一半的时候，由于当时 Borland 的 CEO Dale Fuller 先生看好 Linux 的发展，因此下令所有 Delphi 小组的成员都必须投入到 Linux 开发工具的研发工作，全力为 Kylix 催生，于是连 Chuck 也被要求先暂缓所有的研究计划，投入 Kylix 的开发工作。其实，当时我就非常反对像 Chuck 这样的顶尖人才进入 Kylix 小组撰写程序代码，因为这实在是非常浪费的事情。Chuck 应该进行更为重要的研究计划，而不是只开发一般的工具而已。但是，当时 Borland 高层认为 Linux 将可带领 Borland 一飞冲天，因此仍然坚持所有的人力都必须投入。不过，市场就是变化得这么快，在 Chuck 和 Danny 都投入到 Kylix 的开发之后，虽然 Delphi 小组几乎以创记录的时程在 1 年半左右就在一个新的平台开发了一个新的产品线，但是在 Kylix 推出之后，Linux 平台的疯狂热潮却开始快速消退。所有投入 Linux 的厂商再也无法仅以沾上 Linux 的名称就可以让股票日创新高，市场终究是要回到基本点，只有真正获利的

公司才能够在市场成为赢家。
在 Chuck 被 Kylix 开发工作延误了近 2 年的时间后，Apollo 再也不像当初那么吸引人了，因为市场已经出现了类似的科技，例如 EJB 的 OR Mapping 技术和 Bold 等公司的产品。如果 Borland 当初能够让 Chuck 全力发展 Apollo 计划、并且在其他公司之前推出 Apollo 的成果，那么 Delphi 将可以在 OR Mapping 方面占有领导的地位，Borland 研究的 OR Mapping 技术说不定还可以被 SUN 授权使用，就像 Oracle 花了大钱从 WebGain 购买类似的技术一样。Anders 和 Chuck 这两位拥有一流技术和眼光的技术人物，或多或少地被许多平凡的管理人物糟蹋了好几次。
Chuck 本身是一位非常和蔼可亲的人物，我曾经多次和 Chuck 交谈，每次谈话时 Chuck 总是笑嘻嘻的，似乎没有事情可以让他感到忧虑。如果不知道 Chuck 的人和 Chuck 交谈，那么可能没有人会相信，这位看起来像是好好先生的人在软件方面有这么惊人的成就和高深的造诣，而 Chuck 一头接近红色的头发也让我第一次见到他时被吓了一跳。
当 Chuck 和 Danny 被征召开发 Kylix 时，其实也不是非常顺遂的。在 Kylix 激活之后，照例是由 Danny 负责 Linux 上编译器和 RTL 的研发工作，而 Chuck 则负责 VCL 和 CLX 方面的工
作。由于要在 Linux 上开发集成开发环境，必须先在 Danny 负责的底层 RTL 和编译器完成之后才能够开始设计。但是，Danny 在把 Delphi 的 RTL 和编译器移植到 Linux 的过程中发现了一些 Linux 的臭虫，因此，当时 Danny 在 Linux 的论坛上公布了他发现的臭虫，并且希望 Linux 的社群能够修改这些问题，如此一来 Borland 才能够继续研发 Kylix。
不过，也许是 Linux 的社群拥有排外的情绪，一直认为 Borland 不是正统的 Linux 软件厂商，因此对于 Danny 指出的 Linux 臭虫也嗤之以鼻，认为 Danny 什么都不懂就来说是 Linux 的臭虫。由于 Linux 论坛上的人非常的不友善，而且坚决不承认 Danny 提出的是臭虫，因此也惹得 Danny 非常不高兴，认为做软件的技术人员为何不能就事论事，明明有问题却死不承认。于是 Danny 便在 Linux 论坛上和这些人发动了笔战，愈吵愈轰动，最后演变成了两派人马互相批评。我在当时也想不通，为什么明明 Danny 已经指出了 Linux 有问题的地方，而这些也是搞软件的人却有如此的反应？这些人是不是太小心眼了呢？以 Danny 如此功力深厚的人反而被这些 Linux 的人说成是不懂软件开发真是笑掉人的大牙，这些人应该看看 Danny 做出了什么东西，看看他们能不能做得出来再说。
由于 Danny 无法在 Linux 论坛上得到结果和支持，因此一怒之下干脆自己来修改 Linux 的臭虫，好让 Kylix 能够继续开发下去，不再需要这些 Linux 社群的帮忙。这也是为什么在安装 Kylix 时，Kylix 不但会检查使用者 Linux 使用的版本，并且会安装 Patch 档案以修改 Linux 操作系统的问题。Danny 选择了安装额外的 Patch 档案的方式来解决 Linux 的臭虫，而不是直接修改 Linux 的核心，再由 Borland 分发 Linux Distribution。当时，在 Danny 解决了 Linux 执行时期函数库的一些臭虫之后，Kylix 才能够顺利地开发下去。
后来，在 Kylix 小组开发 Kylix 的集成开发环境时也发现了一些 XWindow 的臭虫，Danny 也是选择由 Borland 自己来修改加以解决，而不需要 Linux 社群的帮忙。
当然，由于 Danny 和 Linux 社群之间的大战也让 Danny 憋了一肚子气，在 Kylix 推出之后，就把随后相关的开发工作交给 Kylix 小组来负责，Danny 则专心到.NET 研发小组为 Borland 开发.NET 上的下一代开发工具了。Danny 离开 Linux 是 Linux 的损失，这些和 Danny 争吵的 Linux 程序员不知道他们在 Linux 上损失了一个天才型的软件人员。有时我想，一些庸才不就是不断地攻击天才吗？难怪古人说"不招人忌是庸才"了。看了 Danny 大战 Linux 论坛这一幕，我也只能在旁摇头叹息，不过我个人倒是很高兴 Danny 和 Chuck 全力开发 NET 产品，因为我一直想使用 Borland 的开发工具学习和开发.NET 应用程序呢。
目前，Chuck 在 Borland 进行的工作是在.NET 上研究先进的技术，包含了在 2002 年

BorCon 上 Chuck 公开展示的新语言--Charlotte。Charlotte 主要是提供 Web Service 的 First-Class 语言，是由 Chuck 定义 Charlotte 的语法、功能，并且实现 Charlotte 编译器的。我实在佩服像 Chuck 以及 Anders、Danny 这些人物，因为这些人几乎都可以独自开发和实现新的程序语言，其功力的确是一般软件人员难以想象的。

在 BorCon 上，Chuck 已经展示了 Charlotte 的语法以及初步的编译器，目前，在 Borland .NET 内部，Charlotte 使用了另外一个比较正式的名称，到了 2003 年或许我们就可以看到 Chuck 和 Danny 在 2002 年一整年努力的成果了。

回到未来

2002 年，Borland 推出了 Delphi 7。虽然此时 Microsoft 已经信誓旦旦地表明，.NET 才是 Windows 的未来，不过现在 Windows 应用程序的开发仍然是主流。但是未来呢？Delphi 的未来是什么呢？

Borland 已经对全世界宣布了 2003 年即将推出.NET 上的开发工具，首先支持的语言将会是 C#和 Object Pascal，而且在.NET 上，Delphi 已经成为 Object Pascal 的代名词，这意味着未来在.NET 上，Delphi 已经是一个语言名称了，Delphi 的使用者将使用 Delphi 语言在.NET 上开发新一代的.NET 应用系统。那么在 Windows 平台呢？Delphi 7 会是最后一个版本吗？

当然不，虽然根据各种信息调查的结果显示，从 2003 年开始，.NET 将进入起飞的阶段，但是原生 Windows 程序的开发仍然拥有三四年的需求。既然如此，那么一定还有许多的使用者仍然需要原生的 Windows 程序开发工具，Borland 不会放弃这些使用者和这么大的市场，因此 Borland 也一定会继续推出新的 Delphi 版本供使用者使用。

更何况，即使是对于想要开发.NET 的使用者来说，可能有极大部分的人也同时需要开发原生窗口应用程序。那么，为什么软件厂商不提供一个开发工具能够让使用者在同一个集成开发环境下同时开发原生窗口应用程序以及.NET 应用程序呢？这个需求就是 Delphi 的优势和机会。看看现在 Delphi 7 提供的功能，我们会很惊讶地发现，其实 Borland 已经偷偷地在进行一些革新的做法。

如果读者在 Delphi 7 的集成开发环境中安装了 Delphi for .NET command-line complier IDE integration，那么就可以如下图般在 Delphi 7 的集成开发环境中激活 Delphi For .NET 编译器，以便在 Delphi 7 中开始撰写.NET 应用程序。在 2002 年 11 月，Borland 又公开了 Beta 版本的 VCL.NET 供 Delphi 7 使用者下载，以便在.NET 中使用 VCL 组件。

不过，许多人会觉得光是拥有 Delphi For .NET 编译器以及 VCL.NET 并不够用。如果要开发.NET 的 WinForm 应用程序，那么 Delphi 7 目前并没有提供类似 Delphi 的 Form Designer，因此仍然非常不方便，Delphi 的使用者仍然需要一个解决方案。

让我们想想，虽然目前 Delphi For .NET 没有像 Delphi 的 Form Designer，但是如果我们能够使用 Delphi 本身的 Form Designer 作为.NET WinForm 的开发接口，然后，如果能够再通过一个工具把 Delphi 的 TForm 和 VCL 转换为.NET 的 WinForm 以及 VCL.NET 不就可

以了吗？如此一来，Delphi 的使用者几乎可以在不花费时间成本之下立刻在 Delphi 中开发.NET 可视化 WinForm 应用程序，这不是一举数得吗？没错，其实 Borland 也早就想到了，因此 Borland 也正想开发一个 Delphi 转换到.NET 程序的转换器让 Delphi 的程序员使用。这样 Delphi 的程序员就可以直接使用 Delphi 的 Form Designer 来设计.NET WinForm 的接口，最后再通过转换器自动地转换为.NET 的 WinForm 应用程序。

如果读者使用过 Delphi 7 的 Delphi For .NET 编译器，那在其中的文件以及 Delphi 的论坛中就可以看到"Morpheus"这个名称。其实，Morpheus 正是 Delphi For .NET 编译

器的研发计划的代号[在电影 The Matrix 中，Morpheus 是救世主(The One)基努李维尚未出现之前的领导者，Morpheus 的任务就是寻找救世主以拯救末世]。因此 Delphi 的研发小组很有创意地把 Delphi For .NET 编译器命名为 Morpheus，以代表 Delphi For NET 编译器是未来 Borland 推出纯.NET 开发工具之前的救世主，负责带领 Delphi 程序员走向未来.NET 的救赎之道。而 Morpheus 计划的任务就是为 Galileo 打下成功基础。虽然我们对 Delphi 8 可能提供的功能现在还不清楚，但通过使用者的需求以及市场的现况，可以推算出如下轮廓：

- 新的集成开发环境：这是为了让 Delphi 能够同时在集成开发环境中开发原生窗口应用程序、.NET 应用程序以及 Kylix 应用程序
- 新的 VCL 和 CLX：可以让 VCL 同时使用在原生窗口和.NET 之中。此外 Borland 也将再

  次修改 VCL/CLX 以增加 Framework 在三个平台的兼容性
- 新的 Delphi/Kylix 和 Delphi.NET 编译器：可以在 Object Pascal 语言上提供更为兼容的效果。这是因为在 Delphi For .NET 中，Borland 已经为 Object Pascal 加入了许多新的语言元素和功能，Borland 可能也将为 Windows 和 Linux 平台上的编译器加入这些功能
- 更多的辅助工具：帮助程序员同时开发三个不同的应用系统

当然 Delphi 8 还将有其他许多未知的功能，不过上面所列的几项应该是 Delphi 8 肯定具备的。如果 Delphi 8 真能提供上述功能，那我相信它将是使用率最高的窗口开发工具，因为除了程序员完全是开发.NET 应用程序之外，Delphi 8 可以提供最齐全的开发能力。

Delphi 8 将是 Delphi 最后的一个版本吗？这我也不知道，唯一可以用来判断的标准是 NET 多久能够完全占据窗口开发的应用。如果真有那一天，那也就是所有原生窗口退出市场主流的时候，就像数年前的 DOS 开发工具一样。届时也请读者把 Delphi 的最后一个版本保留起来，以作为我们一起经历过原生窗口开发的见证，同时，也作为这个曾经是最棒的原生窗口开发工具的纪念。

Delphi 风云榜

Delphi 的开发过程创建了许多记录，并且也造就了许多有名的人物。Delphi 创建的记录是许多开发工具无法企及的，而围绕在 Delphi 外围的杰出开发者也各领风骚，为 Delphi 的传奇添加了更多精彩的故事。这些 Delphi 记录和杰出的 Delphi 开发者故事值得读者们一一品味，特别是 Delphi 使用者们熟悉或听闻过的人们。他们虽然不像 Delphi 的灵魂人员 Anders、Chuck 或 Danny 那样，广为人知、受人尊敬，但对 Delphi 的发展，他们也具有不可磨灭的贡献，这里我们也来看看他们的"庐山真面目"。

Delphi 集成开发环境之父

相信每一个 Delphi/C++Builder 的使用者每日都花许多时间在 Delphi/C++Builder 的集成开发环境中，既然如此，那除了 Anders、Chuck 和 Danny 之外，大家一定要认识一下负责开发 Delphi/C++Builder 集成开发环境的主要领导人 Allen Bauer。

Allen Bauer 是 Borland 的资深工程师，已经在 Borland 工作了相当长的时间。Allen 除了从 Delphi 1 开始便负责集成开发环境的研发工作外，还不断地翻新集成开发环境、改善 Delphi/C++Builder 集成开发环境的公开标准：Open Tools API 的架构。我曾经在费城旅馆的电梯中和 Allen 交谈过，Allen 讲话非常轻声细语，给人一种翩翩君子的感觉。

Allen 的这张照片应该是最近的，因为相片中的 Allen 比 1999 年我在费城时看到的样子老了许多，看来最近几年 Allen 为 Delphi/C++Builder 的集成开发环境投入了不少的心

力。目前 Allen 正在为 Galileo 全力开发新的集成开发环境，据说 Allen 将在新的集成开发环境中加入许多更强劲的功能，2003 年我们继续关注 Allen 的下一个力作吧。

Borland RAD 工具的推广大使

我非常怀念 Charlie Calvert，因为在所有 Delphi R&D 小组中，我和 Charlie Calvert 有过最多共事的经验。Charlie Calvert 属于 Borland Developer Relationship 小组中的资深经理，主要工作是负责开发全世界 Borland RAD 工具并协调其与使用者之间的关系。Calvert 不但是著名的 Delphi/C++Builder Unleashed 书籍的作者，前段时间还撰写了 JBuilder 7 的书籍。

Charlie Calvert 本人是一位素食者，为人非常的热情和蔼。他在 Borland 工作的后期也参与了小部分 Delphi 和 C++Builder 研发的工作。Charlie Calvert 曾说当 Borland 不再开发全世界最好的工具时就是他离开 Borland 之际。两年前 Charlie Calvert 终于离开了，这让我非常难过，我认为他的离开是 Borland 的损失。我曾经问过 Charlie Calvert，为什么要离开 Borland？他回答说是因为不习惯当时 Borland 的转变(Delbert 乱搞开发工具的时期)而打算自己创业。不过令人高兴的是，在 Charlie Calvert 离开 Borland 之后，他仍然在从事 Borland 相关工具的训练工作，看来 Charlie Calvert 仍然对 Borland 的工具有着一份强烈的爱意。

Delphi 的强中手

除了 Delphi R&D 小组之外，我认为最强的 Delphi 高手应该是 Ray Lischner 了。Ray Lischner 博士从 Delphi 1 开始就积极参与 Delphi 的相关工作，稍后更撰写了名震 Delphi 圈的数本书籍，包括《Secrets Of Delphi 2》、《Hidden Paths Of Delphi 3》以及《Delphi In a Nutshell》等好书，其深厚的 Delphi 功力也是 Delphi R&D 小组所公认的。由于 Ray 的书籍一向令我折服，因此在 Delphi 3 时还特别要求台湾出版商引入 Hidden Paths Of Delphi 3，并且为 Hidden Paths Of Delphi 3 进行中文书籍的翻译工作。

除了撰写书籍外，每一个 Delphi 的新版本，Ray 都参加 Beta 测试。Ray 是一个非常直率的人，一旦遇到臭虫或是 Borland 没有做好的地方，他都会毫不留情地要求 Borland 更正或是批评 Borland 没有尽力。我曾经在 Borland 内部的 Delphi 论坛中看到 Ray 精彩绝伦地痛批 Borland 没有把产品做好。尤其在 Delphi 4 时更是不惧高层权势痛骂 Borland 乱搞 Delphi，看得我大呼过瘾。虽然我身为 Borland 的人，不敢骂 Borland 高层的人，但是心中所想是和 Ray 一样的，而由 Ray 这位具有身份地位的人口中骂出，实在令我觉得爽快，当然 Ray 如此做也是"爱之深，责之切"的缘故。因此直到现在，在参加 RAD 工具 Beta 测试时，我还是最喜欢看 Ray 的评论，因为 Ray 的评论不但有深度，更敢直言，通常是最有帮助的论坛讨论内容。

Delphi 双响炮

说起 Steve Teixeira 和 Xavier Pacheco 这两位仁兄，相信也是许多 Delphi 程序员耳熟能详的大人物，因为他们两位正是《Delphi Developer's Guide》这本好书的作者。Steve Teixeira 和 Xavier Pacheco 都曾是 Delphi R&D 小组的成员，不过这两位仁兄目前都离开了 Borland，各自创业去了，毕竟自己当老板更有赚头。说起 Steve Teixeira 和 Xavier Pacheco，令人好笑的是他们两人的身材实在是很强烈的对比，Steve Teixeira 年轻高大，而 Xavier Pacheco 则极为瘦小，因此当 Xavier Pacheco 站在 Steve Teixeira 旁边时，Xavier Pacheco 就像一个小孩一样。

我和 Steve Teixeira 比较熟悉，因为曾经和他在费城一起开过会，也有过交谈。Steve Teixeira 在 Delphi R&D 小组中负责比较低阶核心的除错功能，Steve Teixeira 让我想起当初 Matt Pietrek 在 Borland 的成长过程。由于 Steve Teixeira 和 Xavier

Pacheco 都是 Delphi R&D 小组的成员，因此他们自然能够取得最新、最深入的信息来写书。

Delphi COM 高手

说到使用 Delphi 来学习 COM 方面的知识时，就不能不提 Binn Ly 这位大名鼎鼎的人物，Binn Ly 也算是华人在 Delphi 圈中之光荣。话说在 Delphi 3 推出之时就以多层架构为重点功能，强调 Delphi 能够撰写多任务的中间件服务器。不过了解内部技术的 Delphi 程序员都知道，其实 Delphi 3 中的中间件服务器只能开发 Single Threaded 型态的 COM 服务器，还不能开发真正的 STA 和 MTA 型态的 COM 服务器，这也是为什么 Delphi 3 的 MIDAS 服务器在客户端使用者人数较多时执行效率就不甚理想的原因。

当时 Binn Ly 先生就指出了 Delphi 3 的问题，并且决定自己开发一个真正的 Delphi 封装 COM 技术的 framework，并且提供真正的 STA、Apartment 和 MTA 的解决方案。这在当时是一项非常惊人的工程，因为除了 Microsoft 的 ATL 之外连 Delphi 都还没有提供，而 Binn Ly 先生却决定完全使用 Object Pascal 来开发一个如此复杂的 framework。后来 Binn Ly 先生不但成功地开发出来，还在自己的网站上公开了 framework 的原始程序代码，而且还写了许多 COM 和 Delphi 方面深入的技术文章。虽然在 Delphi 4 之后 Borland 也终于慢慢地提供了比较完整的 COM 封装技术，但是 Binn Ly 先生在 Delphi 和 COM 方面权威的形象已经深入 Delphi 程序员的心中，他本人也成为日后 Borland 邀请在 BorCon 主讲 COM/COM+讲座的台柱之一。

VCL．NET Architect

Eddie Churchill 是在 Delphi 3 之后才加入 Delphi R&D 研发小组的，不过 Eddie 在 Delphi R&D 研发小组中却上升得很快。

Eddie 在加入 Delphi R&D 小组之后，便开始了 Delphi 中 Diagram Designer 的研发工作，原本 Eddie 的工作是准备在 Delphi 中开发出 UML 的功能，以便为 Delphi/C++ Builder 提供完整的 OOA/OOD 的功能。

不过由于后来 Borland 高层决定和 Rational 合作，而且 Eddie 小组需要极大的资源来开发，因此 Borland 的 RAD 部门便在 Diagram Designer 实现之后决定暂停这方面的研发工作。其实在 Delphi 3 时，当时的 Delphi 产品经理 Ben Riga 就想为 Delphi 加入 UML 的功能，只是一直无法拥有足够的资源来投入这方面的研发工作，一直到 Delphi 5 之后才逐渐在 Delphi/C++Builder 看到这方面的初步成果。

现在 Eddie Churchill 和 Danny 等人投入开发 Borland.NET 方面的产品，并且是 VCL.NET 的 Architect，负责把 VCL 移植到.NET 中，在 2002 年的 BorCon 中，Eddie Churchill 就说明了把 VCL 移植到.NET 之上的困难和挑战，目前在 Delphi 使用可下载的 Delphi For NET Preview 中已经可以看到 Eddie Churchill 在 VCL.NET 方面的初步成果了。

WebSnap 始祖

Jim Tierry 是在 Eddie Churchill 之后不久加入 Delphi R&D 研发小组的，他的第一个工作就是接手 Delphi 的 WebBroker 技术，以开始打造 Delphi/C++Builder 的下一代 Web 技术，初步的成果就是 Delphi 5 的 InternetExpress。

不过在 InternetExpress 推出之后，Jim 自己并不满意，因此立刻开始进行 InternetExpress 下一代的开发工作，那当然就是 WebSnap 了。由 Jim 带领的 WebSnap 开发小组终于在 Delphi 6 中推出 WebSnap 技术。不过也是由于资源的限制，因此在 Delphi 7 中，Borland 决定使用更为完整的 IntraWeb 作为 Delphi 在 Web 方面的主力。

目前 Jim Tierry 应该是在开发.NET 下 Web 方面的技术和产品，我也希望在 2003 年能够有机会和 Jim 见上一面，让他谈谈目前工作的方向。

Delphi Plug-In 第一把交椅

我在日常使用 Delphi 时，一个少不了的工具就是 Eagle Software 出品的 CodeRush，虽然有些人认为 CodeRush 有点不稳定，但我认为 CodeRush 是一个非常好用的工具，当然这也是为什么 CodeRush 几乎在每一年 Delphi Magazine 的年度产品评比中都是第一的 Delphi Plug-In 工具。

CodeRush 是由著名的 Delphi 高手 Mark Miller 先生开发的，Mark 不但是多个产品的作者，更是 BorCon 中经常受邀的主讲者。我曾经在许多地方和 Mark 有着交往，在 2000 年澳洲的 BorCon 时我还曾经告诉 Mark 当时的 CodeRush 在中文 Windows 2000 中有臭虫，因为当我使用 CodeRush 的快捷键时，CodeRush 会送出重复的字符。Mark 当场便拿出 NoteBook 查询 CodeRush 的程序代码，并且要求我也打开我的 NoteBook 展示 CodeRush 的臭虫给 Mark 看。在 Mark 了解了问题之后便询问我的旅馆房号，这才发现我们是住在同一个旅馆。没有想到的是，当晚 Mark 便打电话到我的房间请我过去再次追踪这个问题。后来 Mark 确定 CodeRush 没有问题，应该是中文 Windows 2000 在某些地方处理的方式和英文 Windows 稍有不同，这应该是 IME 方面的原因，不过 Mark 答应我将会提供 walk-around 的方式来克服这个问题。我回国后不久，就收到 Mark 寄来的一个 Patch 档案修正了这个问题，其工作效率真是令人惊讶。如果读者也在使用 CodeRush 这个产品，那可以经常到 CodeRush 的论坛看看 Mark 的发言，许多时候 Mark 的信件都是深夜二三点寄出的，Mark 工作的狂热的确令人吃惊，不过在 Mark 有了 Baby 之后这个现象就比较少发生了。

MIDAS/DataSnap 的掌舵手

Borland 在 Delphi 3 便推出了 MIDAS，一直开发到现在的 DataSnap。其实 Borland 一开始时就对 MIDAS 有非常大的野心，准备把 MIDAS 开发成中间件的标准技术。不过随着中间件从以 RPC 为主变化成 CORBA、COM/COM+以及 EJB，Borland 也很快地放弃了这个想法，

因为 Borland 不可能单独推出中间件技术来对抗业界的标准。

因此在 Delphi 4 之后，Borland 便开始转变 MIDAS 的角色，把 MIDAS 定位成一个通用的数据存取处理层技术，让 MIDAS 能够使用在 CORBA、COM/COM+之中提供方便有效率的封

装数据存取功能，以弥补这些组件技术在存取数据方面的不便--必须使用一笔一笔数据的方式来处理数据。

MIDAS 这样的转变是非常正确的，因为在 CORBA、COM/COM+中要存取数据实在是非常的

不便，经常必须在对象接口中提供特定的方法让客户端存取特定的数据，并且在移动数据时也非常的麻烦，而 MIDAS 却提供了完整的解决方案。正是由于 MIDAS 的方便好用，因此 Microsoft 也仿照在 ADO 中开始模仿 MIDAS 的功能，在 ADO.NET 中更是全面使用 MIDAS

的观念来处理数据的存取。Borland 也曾经把 MIDAS 移植到 Java 中成为 JMIDAS，可是由于 Java 世界非常强调标准，所有的标准都必须由 SUN 来制定，而 SUN 又使用 JDBC、Entity Bean 以及现在的 JDO 来处理数据，因此 JMIDAS 之后就停止了开发。

Delphi 3、Delphi 4 中的 MIDAS 都由不同的工程师负责。在 Delphi 4 中是由 Josh 负责的，在这期间 Josh 和 Dan Miser 合作提供 MIDAS 的技术和解决方案，当时的 Dan Miser 并不是 Borland 的员工，而只是 MIDAS 的爱好者。后来由于 Josh 离开了 Borland，Borland 就把 MIDAS 的维护和新版本开发工作 contract 给 Dan Miser，因此在 Delphi 6 之后的 DataSnap 是由 Dan Miser 开发的，而 Dan 也成为了 DataSnap 的头领。

由于 Dan Miser 非常喜欢 DataSnap，愿意为 Borland 工作，因此现在 Dan 已经到 Borland

应征，准备正式成为 Bodand 的员工，继续把 DataSnap 移植到.NET 上，并同时开发新的 DataSnap 功能。

Delphi Spirit

Marco Cantu 这位意大利的仁兄，应该是许多 Delphi 初学者都知道的人物，因为 Marco Cantu 的《Mastering Delphi》一直是 Delphi 书籍中的长青树，深受初学 Delphi 的人所喜好。

Marco Cantu 也经常是 BorCon 的讲座主讲人以及 Delphi 相关杂志的专栏作家，不过我一直无缘聆听 Marco Cantu 先生的讲座，因为每次我都选择了同时进行的其他讲座，真是令人遗憾。

元老重臣

最后一个我想介绍的人物可能许多人并不熟悉，但是这位仁兄却一直是 Borland 开发工具中的要角之一，那就是 Bruneau Babet 先生。Bruneau Babet 很少公开露面，除非在 BorCon 中读者才有可能见到他。

Bruneau Babet 从 Borland C/C++产品开始就是 Borland 的研发人员，在 Borland C/C++结束之后也转入 Delphi R&D 小组开发 Delphi，而 Bruneau Babet 工作的重点一直是以 COM 方面的技术为主。

Bruneau Babet 除了进行 Delphi 的研发之外，主要的时间是负责 C++Builder 的开发工作，因此他也从事 C++Builder 中融合 ATL 方面的工作，在 BorCon 中，Bruneau Babet 演讲的主题主要是 Borland 开发工具在 COM/COM+的技术支持。

在这里，我无法介绍完所有相关的重要人物，除了上述几位外，许多大家熟悉的人物，包括 Dr. Bob、John Kaster 等，我都没有介绍到。这是因为 Dr. Bob 和 John Kaster 是许多人非常熟悉的，他们的照片在网站上可以经常看到，因此就不再列为重点，这里是以读者不太熟悉的人物为主，让喜欢 Delphi/C++Builder 的读者能够看看这些杰出软件人员的长相、了解他们的个人资料，让许多"久闻其名"的人物真实地呈现在我们面前，作为我们学习的借鉴。

^v^v^v^v^v^v^v^v^v

第五章　逆转的奇迹--Borland JBuilder 的战斗发展史

"没有 JBuilder，Borland 就不可能拥有今日的荣景！"

Java 的快速兴起和成功是谁也没有预料到的，即便对于 SUN 自己似乎也是一个极大的意外，但是成功者一定是果断而且行动迅速的。当 SUN 察觉到 Java 的光明未来之后，便立刻开始大力推销 Java。SUN 的总裁 McNealy 先生数年来苦于没有直接和 Microsoft 对抗的机会，这下在 Java 的身上似乎找到了契机，当然更重要的是 SUN 接下来的一连串行动都被证明是正确而成功的。这些行动包括和各种厂商合作；与 Addison-Wesley 公司合作出版一系列畅销且成功的 Java 书籍；在各大媒体占据版面发表所有与 Java 相关的文章、专栏等；快速培养 Java 使用者的基础，吸引大众对于 Java 的兴趣。这完全是 Microsoft 一向无往不胜、攻无不克的手法，SUN 也发挥得淋漓尽致，并且"以彼之道还施彼身"。更重要的是 McNealy 立刻果断地投入大量的研发资源，不断地改善 Java，终于使 Java 从 1995 年开始展露锋芒，并且快速地成为业界焦点，自此展开了 PC 发展史上最大规模对抗 Microsoft 的争霸战，也改变了许多软件开发的习惯和方向。当然对于 Borland 来说，Java 的发展史也是一场惊涛骇浪的生死之战，是 Borland 从未经历过的大规模集团军混战。

对于 Borland 来说，事情并没有那么顺利。1995 年，当 Java 开始起飞时，Borland 并没有预料到 Java 成长的速度会如此之快。Borland 一开始只是把 Java 当成 C/C++的延伸，因此只在 Borland C/C++5.0 中加入了支持 Java 的 P1ug-In。不过 Borland 很快就发现事

情并不是如此简单，因为除了 Java 的 Plug-In 反应并不好之外，也发现 Symantec 很快在 Java 开发工具找到了新舞台，而且发展得相当快速。在 Microsoft 对 Java 的态度未明之前，无疑 Symantec 占据了先机，Borland 这才警觉到自己的失策，Java 大会战一开始 Borland 就已经落后了。Borland 如何才能在下一场最重要的开发工具大战中进行反攻呢？

Java 开发工具初期的争战

当 Symantec 从 C/C++开发工具市场大撤退之后，Eugene Wang 不愧是相当高明的开发工具好手，立刻察觉到尽管在 C/C++市场遭遇失败，但利用原有力量却可以在即将茁壮成长的 Java 市场上扳回一城，因此立刻率领原先 Symantec C/C++的开发团队快速进入 Java 开发工具的领域。Eugene 很快以当初 Symantec C/C++的集成开发环境作为基础，开始开发 Java 开发工具，这就是后来著名的 Visual Café。

Symantec 几乎是第一个介入 Java 开发工具的软件公司，又利用了 Symantec C/C++的基础，因此在 1995 年，当 Java 获得愈来愈多人的注意之后，Symantec 也准备好了她的第一个 Java 开发工具--Visual Café。1996 年 10 月，Symantec 赶在 JDK 1.1 之前正式推出了 Visual Café。虽然当时许多人批评 Symantec 为什么不等到 JDK 1.1 之后再推出，以支持最新的 JDK 标准(因为当时的 JDK 1.0x 版本有许多的问题)，不过这些批评并没有妨碍 Visual Café的成功。

由于当时许多软件人员急于投入 Java 的学习行列，因此当 Symantec 推出了 Visual Café 之后，立刻在市场获得了极大的成功。特别是在 Java 学习市场和教育市场，Visual Café几乎是以席卷市场的姿势迅速占据了 Java 开发工具第一名的地位，成为炙手可热的产品，而 Symantec 公司也一扫在 C/C++开发工具被挫败的怨气，再次成为开发工具市场的领导厂商。

由于当时 Microsoft 对于 Java 采取敌视的态度，因此几乎不可能推出 Java 开发工具，而 Borland 也还正陷于 C/C++的苦战之中，尚未查觉到 Java 的潜力。至于另外一个死对头 Watcom 则已被 Sybase 并购，无法在开发工具市场再成气候。这对于 Symantec 来说简直是天赐良机，一个可以独打 Java 开发工具市场的绝佳机会。剩下唯一的威胁是 SUN 要推出的 Java 开发工具。但是 Symantec 已经抢得市场先机，而且已经成为领先者，只要好好的把握，就能够以逸待劳和 SUN 对战。在这个 Java 开发工具萌芽的阶段，Symantec 似乎是占了绝对的优势，不过很可惜的是接下来 Symantec 也接连犯了几个错误，逐渐失去了取得的优势。

首先是当 Visual Café推出之后，Eugene Wang 便离开了 Symantec 自己开公司做生意去了。这对于 Visual Café有着相当大的影响，因为 Symantec 靠着 Eugene Wang 的技术能力和眼光，才能够和 Microsoft、Borland 和 Watcom 在 C/C++开发工具市场对抗；Eugene Wang 又独具慧眼打造了第一个 Java 开发工具 Visual Café。Symantec 应该在 Visual Café获得初期的胜利之后再次借重 Eugene Wang 的功力继续攻城掠地，但是 Symantec 居然让 Eugene Wang 离开，立刻少了开发工具掌舵的大将。

第二个错误是 Symantec 当初为了尽快推出 Visual Café以抢占市场先机，因此集成开发环境是使用 C/C++语言撰写的。这造成了数项缺点，其一是由于使用了 C/C++来撰写可视化窗体设计家(Visual Form Designer)，因此程序员在设计时看到的可视化效果和真正 Java 程序执行时的效果是有一些差异的；其二是为了维护 Java 的控制组件程序代码和以 C/C++撰写的可视化窗体设计家保持同步的状态，在可视化窗体设计家产生的原始程序代码中内嵌了一些 Visual Café控制卷标(Control Tag)。这些控制卷标并不是 Java 的程序代码，只是为了可视化窗体设计家使用。如果程序员不小心修改或是删除了这些控制卷标，就会造成 Visual Café的可视化窗体设计家的失效。这是非

常严重的缺点，Symantec 应该在 Visual Café 1.0 之后立刻改善这些问题，然而 Symantec 却似乎一直无法有效地加以改善。当然，问题的根源在于 Visual Café的集成开发环境是使用 C/C++语言撰写的，要完全改善这个问题，Symantec 必须使用 Java 语言重新改写集成开发环境，这正是 Borland 后来采取的策略。不过使用 Java 撰写集成开发环境也是非常冒险的行动，Borland 后来因此付出了沉重的代价。Symantec 之所以没有如此做，大概也是因为当时的 Java 并没有成熟到可以如此做的地步。不过 SUN 显然不这么认为，其对 Java 信心百倍，不久就推出了 SUN 的 Java 开发工具，Java Workshop。当 Java 成功地掳获了开发者的心之后，对于 Java 开发工具的要求便与日俱增。虽然 Symantec 已经推出了 Visual Café，但是许多人仍然希望 Java 的正宗厂商 SUN 能够推出 Java 开发工具，让所有想要学习、使用 Java 的开发人员能够使用最标准的 Java 开发工具。当然，许多人也希望 SUN 能够使用 Java 撰写 Java 开发工具来向世人证明 Java 的能耐，让质疑 Java 能力的人以及 Microsoft 闭嘴。当然，SUN 在 Java 成功之后也信心满满地宣布了 SUN 的开发工具计划，以满足广大开发人员的需求。McNealy 多年来进攻 Microsoft 地盘的希望似乎即将出现光明的未来。

之后不久，在万方期待之下，SUN 终于推出了 Java 开发工具 SUN Workshop。在 Java Workshop 即将推出之前，Symantec 非常地紧张，因为这关系到 Symantec 是否能够在 Java 开发工具市场站稳脚跟。我记得在 Java Workshop 推出之际，所有的媒体、杂志都大幅报道 Java Workshop，SUN 也大力地宣传和促销 Java Workshop。从当时的气势来看，Java Workshop 颇有"千秋万世，一统江湖"的味道。我所认识的许多朋友不管是用买的、借的、下载等和--嗯--那个的……各种手段，都热切地想取得一套 Java Workshop 来玩玩。不过丑媳妇总要见公婆的，在许多人使用了 Java Workshop 之后，才发现它不但执行缓慢得像乌龟一样，而且问题多多，和一般的 PC 开发工具水准比起来简直是差了十万八千里。看来 Java Workshop 只适合使用在昂贵的 SUN 工作站计算机上，而在当时大多数人使用的 PC 上则根本跑不动，除非是拥有异于常人的耐性。

Java Workshop 雷声大雨点小，不久之后就人气溃散了。许多原来对 Java 有信心的人在用了 Java Workshop 之后，也开始质疑 Java 是否适合使用来开发复杂的应用程序？是不是只适合用来撰写 Applet？是不是只适合使用在 SUN 的工作站和计算机之上？虽然之后 SUN 仍然很努力地推出 Java Workshop 2.x 的版本，希望一洗 Java Workshop 1.x 的恶名，但是仍然无法挽回 Java 开发人员的信心。至于其他仍然对 Java 有兴趣的人则转而使用 Symantec 的 Visual Café，让 Visual Café进一步地扩大了市场占有率，也让 Symantec 吃了一颗定心丸。当然也有许多 Borland 的支持者开始强烈地期望和要求 Borland 能够推出最好的 Java 开发工具。

SUN 在 Java 开发工具市场大溃败之后，才了解到 PC 开发工具市场和 Solaris 开发工具市场不一样。在 Solaris 上 SUN 是一家独大，但是在 PC 市场上可是百家争鸣，竞争对手一个比一个强悍。SUN 不了解 PC 开发工具市场的特性，以为靠着 Java 正宗的招牌就可以通行无阻却是大错特错，并且在当时被 Microsoft 讥笑不懂得开发软件，这也是因为 SUN 经常讥笑 Microsoft 不懂得开发操作系统，看来在当时 SUN 也不必五十步笑百步。SUN 在 Java 开发工具市场弄得灰头土脸之后，不得不专心开发 Java 语言和 JDK 函数库，并且在 Java 语言更为成熟之后开始想要开发 Java 的组件技术，因此开启了稍后和 Borland 合作共同开发 Java Bean 的功能规格，再进而和 Borland 共同研发 JDK 的规格，最后更对 Borland 的 JBuilder 发生了强烈的兴趣，甚至想并购 Borland。当然这都是因为后来 Borland 展现在了 Java 方面高度的技术，让 SUN 从肯定到折服的原因所致。

Borland 的 Java 艰辛奋斗

"事情并没有这么顺利"，Borland 当时的 R&D 主管这么说，并且充满了焦虑。当 Borland

警觉到 Java 的潜力之后，Visual Café早已成功地上市，SUN 也准备推出 Java 的开发工具。当时 Borland 正逐渐从 C/C++市场失去王者的地位，财务上也开始出现经营赤字，整个公司正陷于一团混乱的情形中，似乎已经没有额外的资源可以投入 Java 的研发。在起步落后，又缺兵少粮的情形下，Borland 似乎即将失去进入 Java 市场的希望。
好在稍后的 Delphi 一炮而红，让 Borland 大赚了一票，也稳定了军心。Delphi 为 Borland 注入的资源也很快让 Borland 激活了 Java 研发小组。虽然 Borland 已经落后许多，但是 Borland 知道绝不可以失去这个市场，因为 Java 的市场没有 Microsoft 式的寡占，Borland 有希望在 Java 市场比 Borland C/C++、Delphi 等更成功。此外，Borland 更需要在 Delphi 这条产品线之外开拓其他的收入来源，否则只靠 Delphi 产品，公司仍然无法成长得更为茁壮，以和其他的软件公司竞争。
在 1994、1995 年间，Borland 正式成立了 Java 研究小组，开始研发 Java 的技术，准备开发 Java 开发工具。这个 Java 开发工具的内部研发名称便是 Latté。一开始 Latté小组的研发资源并不够多，因为当时的 Borland 是在风雨飘摇之中，无法注入足够的资源到 Latté小组。因此在 Latté开始开发的初期进展得并不顺利，进度很缓慢。一直到了 Borland 靠 Delphi 浴火重生之后 Latté小组才有了足够资源，研发的进度才开始加速。不过与竞争对手们比起来，Borland 在 Java 方面的确是相当落后的，几乎是跑在最后的参赛者。不过幸运的是 Java 开发工具之战似乎是一场漫长的马拉松比赛，除了一开始的表现之外，更重要的是比谁能够撑得比较久。事实上看 Borland 如何在 Java 竞赛场上反败为胜、一一打败强者，进而成为 Java 开发工具王者的过程是相当精彩的，而 JBuilder 小组使用的竞争策略更值得我们玩味和学习。
依我个人的眼光来看，在 Borland 开发 Java 开发工具的过程中经历了数个不同的阶段，每一个阶段都有着非常激烈的竞争，有着成功者和失败者。只是有的失败者仍然坚持竞争下去，有的却随风消散。JBuilder 最终能够成为王者，除了是因为愈挫愈勇、Borland 没有退出 Java 市场之外，还在于 Borland 在开发 JBuilder 3 时下了一个关键性的决定，以及在 JBuilder 3 之后每一个版本都有明确的目标，终于在 JBuilder 4 之后慢慢成为市场第一的领导者。当然这长达数年的争战过程是非常艰辛的，不过这段历程正是整个 Java 开发工具逐鹿中原的写照史。
第 1 阶段--Java JIT 编译器的战争
Borland 也许不是最晚开始研发 Java 技术的厂商，但是明显地落后于其他竞争对手则是不争的事实。Borland 在 Latté万事尚未具备的情形下，展开 Java 竞赛的第一步便是从 Borland 传统的拿手绝活开始，那就是从 Java JIT 编译器开始出发。不过由于 Borland 当时对于 Java 技术尚未拥有良好的掌握，因此一开始是和 Pascal 的祖师爷 Dr. Niklaus Worth 合作，由 Dr. Niklaus Worth 以及他的学生们为 Borland 研发 Java JIT 编译器，而 Borland 本身的 Latté小组则平行地开发 Latté其他的功能。由于当时 Java 已经逐渐在校园流行，而且吸引了许多学术研究的兴趣，Dr. Niklaus Worth 以及他的学生们很早便开始投入 Java 相关的研究。因此当 Borland 找上门之后，自然便一拍即合。Borland 缩短了开发时程，而 Dr. Niklaus Worth 研究小组则乐得有人赞助研发费用。
Dr. Niklaus Worth 研究小组的第一个作品就是在 1997 年初左右推出的 Java JIT 编译器，这个由 Dr. Niklaus Worth 研究小组研发的 JIT 编译器可以让编译后的 Java ByteCode 执行速度比当时 SUN 的 Java 编译器以及 Symantec 的 JIT 编译器快了数倍。Borland 宣布此 JIT 编译器之后立刻震惊了 Java 界，因为当时缓慢的 Java 执行速度是所有使用 Java 的人都希望能够立刻大幅改善的。而 Borland 推出的 Java JIT 编译器似乎给所有 Java 开发人员看到了未来的希望。虽然严格地说当时即使是使用 Borland 最新的 JIT 编译器编译 Java 程序，其执行速度仍然是很"龟速"的，但是对于使用 Java 来学习程序设计或

是撰写、执行一些小的 Applet 来说仍然是很好用的。因此当 Borland 一推出此 JIT 编译器之后，便立刻打响了 Borland 在 Java 界的知名度，所有 Java 开发厂商也开始视 Borland 为认真的竞争对手。否则以当时 Borland 的气势来看，除了 Delphi 之外，Borland 几乎已经一无所有了。

Borland 在 Java 的处女作 Java JIT 编译器一炮而红，立刻吸引了当时浏览器霸主 Netscape 的注意。由于当时 Netscape 大力支持 Java 以便和 Microsoft 竞争，因此非常需要有品质精良的 Java JIT 编译器内建在 Netscape 之中，以顺利且快速地执行 Java Applet，增加 Netscape 的竞争力和吸引力，突显与 Microsoft IE 的不同。不久之后 Netscape 便找上了 Borland，希望能够在 Netscape 中附带 Borland 的 Java JIT 编译器。

对于 Borland 来说，这又是一个千载难逢的机会。因为这不但证明了 Borland 在 Java 技术的努力成果，更重要的是 Netscape 在当时是不可一世的软件公司，全世界有数百万的使用者。这意味着一旦 Netscape 内建 Borland 的 Java JIT 编译器，Borland 在全世界将立刻拥有数百万的 Latté潜在使用者，对于 Borland 来说是好得不能再好的条件了。因此 Borland 立刻答应了 Netscape 的提议，让 Netscape 搭配 Borland 的 Java JIT 编译器。但是这一举动也立刻牵一发而动全身，进而导致了 Java JIT 编译器的大混战。

在 Netscape 和 Borland 达成了协议并且开始出货之后，却引起了 Symantec 的忧虑和不满。因为当时 Symantec 是 Java 开发工具的老大，而 Borland 连个 Java 开发工具都尚未推出，可是 Netscape 却跑去使用 Borland 的 Java JIT 编译器，这不是让全世界都知道 Borland 的实力并且让 Symantec 脸上无光吗？为了颜面以及避免失去 Java 开发工具的市场，很快 Symantec 便决定开始反击。Symantec 立刻也集中资源投入 Java JIT 编译器的研发，开发出比 Borland Java JIT 编译器更快的 Symantec JIT 编译器，并且准备开发一个直接把 Java ByteCode 编译成原生 Windows 程序代码的 Java 编译器。就在 Borland Java JIT 编译器风光不久之后，Symantec 也宣布了新的 Java JIT 编译器。Symantec 的 Java JIT 编译器比 Borland Java JIT 编译器更有效率，编译后的 Java ByteCode 执行效率比 Borland 的快了 2~3 倍。

在 Symantec Java JIT 编译器宣布之后，又轮到.Borland 脸上无光了。才刚和 Netscape 谈好合作条件，没有想到效率王位还没坐热就立刻被 Symantec 踢了下来，这如何向 Netscape 交待？因此 Borland 立刻进行改善 JIT 编译器的研发工作，力图再次超越 Symantec。果然 Borland 的努力没有白费，不久之后 Borland 的 JIT 编译器又打破了 Symantec JIT 编译器创下的效率纪录。自此 Borland 和 Symantec 便展开了 Java JIT 编译器的"竞速"比赛，不断地试图打败对方。也由于 Borland 和 Symantec 的 JIT 竞赛，当然更重要的原因是 Java 的执行速度在当时实在是太过缓慢，引起了 IBM、Microsoft 以及 SUN 在 Java 编译器方面的研究。

Symantec 在当时不愧是 Java 开发工具的王者，在和 Borland 几次的 JIT 编译器交手之后，便开始逐渐地占了上风。由 Dr. Niklaus Worth 研究小组研发的 Java JIT 编译器也逐渐不再是 Symantec 的对手。至此 Borland 决定收回 Java 编译器的技术，开始自行研发。Borland 发觉光是和 Symantec 在 Java JIT 编译器竞争没有多大用处，当务之急是赶快推出自己的 Java 开发工具。因此 Borland 开始退出和 Symantec 在 Java JIT 编译器的竞赛，以求全速催生 Latté。当然 Borland 退出 JIT 编译器的第一阶段战争之后的影响是不久之后 Netscape 便不再使用 Borland 的 Java JIT 编译器，改为使用 Symantec 的 Java JIT 编译器。至此 Symantec 终于获得了 JIT 编译器第一阶段的战争胜利，保住了 Java 开发工具第一厂商的颜面。但是 Symantec 真的获胜了吗？那可不能断言，因为 JIT 编译器战争才刚开始。

在 Symantec 的 Java JIT 编译器打败了 Borland 的 JIT 编译器之后，Symantec 便把脑筋动

到了 SUN 的身上，希望 SUN 也能够使用 Symantec 的 Java JIT 编译器，把 Symantec 推向 Java 核心技术的领导厂商宝座。不过 Symantec 的盘算显然是落空了，因为 SUN 已经决定收购一家专门研发 Java 编译器技术的软件公司，并且准备开发自己的 JIT 编译器，那就是后来的 SUN HotSpot 编译器技术。另外 Microsoft 和 IBM 也开始加入了 Java JIT 编译器的竞赛之列。IBM 为了和 SUN 争夺 Java 领导者的地位，不但自己研发 IBM 的 JDK，甚至
也研发 IBM 的 Java JIT 编译器。严格地说，当时 IBM 的 Java JIT 编译器品质比 SUN 提供的好多了，不但稳定而且执行速度比 SUN 的快了许多，让 SUN 也颜面无光，很不是滋味。甚至可以说 IBM 的 Java JIT 编译器品质不会比 Symantec 的 Java JIT 编译器差到哪里。
更麻烦的是 Microsoft 为了让 IE 能够和 Netscape 竞争，也可以执行 Applet，因此也开始研发精良的 Java JIT 编译器。特别是当 Microsoft 得到了 Anders Hejlsberg 之后，在编译器技术方面有了重大的突破。虽然 Microsoft 的 JIT 编译器一直不像其他厂商的 Java JIT 编译器那么符合标准，但是其品质却是相当的精良。在 Microsoft 不断地改善之下，依我当时的测试，经其编译后的 Java ByteCode 执行的速度是最快的，连 IBM 和 Symantec 的 JIT 编译器都不是对手。因此从我的观点来看，在这个 Java JIT 编译器的阶段，应该是 Microsoft 获了冠军。要不是 Microsoft 没有持续支持最新的 JDK 标准，又混杂了一些 Microsoft 自己的东西，到最后很可能使用最为广泛的 Java JIT 编译器反而就是 Microsoft 的 JIT 编译器。
至于 Symantec，在取得了 JIT 编译器表面上的优势之后，立刻又把重点放在了开发直接把 Java ByteCode 编译成原生应用程序的原生 Java 编译器。稍后 Symantec 成功地开发出了这种编译器，让 Borland 大为紧张，并且准备跟进。而 Symantec 也把这个原生 Java 编译器加入到 Visual Café中，成为一项吸引人的功能。不过很快地这个功能却引起了许多 Java 使用者的批评，因为他们认为这违反了 Java"Write Once，Run Everywhere"的精神，如此一来厂商必须为每一个不同的平台开发原生 Java 编译器，这会造成 Java 应用程序在不同的平台执行的反应不一致的现象，又陷入 C/C++语言开发的应用程序在不同的平台表现不一的相同问题。后来连 SUN 也不赞成这种做法，当然这是因为 SUN 想力推自己的 HotSpot 编译器技术。因此原生 Java 编译器在风行了一阵短暂的时间之后就不再吸引人注意了，而 Borland 原本为 JBuilder 开发原生 Java 编译器的计划也因此而打住。
Microsoft VJ++的威胁
1996 年，Anders Hejlsberg 来到 Microsoft 之后的第一个作品即将推出，那就是 Microsoft VJ++。VJ++的即将推出，对于许多软件公司而言都是一个很大的震撼。
对于 SUN 来说，这是 Microsoft 在 Java 领域的挑战。在 SUN 自己的 Java 开发工具不争气的窘境之下，又得面对擅长开发工具的 Microsoft，特别是由 Anders 领军开发的精品。对于其他的 Java 开发工具厂商来说，也是提心吊胆。Visual Café在 JBuilder、Visual Age For Java 陆续推出之后市场占有率已经慢慢地被瓜分，现在又得再次面对 Microsoft 的竞争，昔日 Symantec C/C++失败的阴影又缠上了心头。而 Microsoft 的死对头 IBM 更是在 VisualAge For C/C++、VisualAge For BASIC 连番失败之后，好不容易推出了 VisualAge For Java，准备在 Java 开发工具市场打一场好球赛，没有想到现在 Microsoft 又来搅局。
对于 Borland 来说，这个消息更是令人不安，因为 Borland 本身的 Java 开发工具仍然处于研发阶段，还没有推出，而且看样子将会是市场上最后一个推出的 Java 开发工具，落后主要竞争对手已经很多了。现在 Microsoft 居然更早一步推出 Java 开发工具，而且是由 Anders Hejlsberg 主持开发的。Borland 当然知道 Anders Hejlsberg 的实力，

自然不敢轻视 VJ++的影响力。更麻烦的是在 VJ++推出之前，Microsoft 一直对 VJ++保持模糊的态度，不愿意表明 VJ++是否是一个纯正 Java 开发工具。更让 Borland 惊讶的是，Borland 内部对于 VJ++ Beta 的测试表明 VJ++编译出来的程序代码在某些方面居然比 Delphi 等原生的 Windows 开发工具执行得还快速。这意味着 VJ++不但对于 Java 开发工具可能会有严重的影响，甚至对于一般的 Windows 开发工具都有可能造成威胁。不过 Borland 分析如果 VJ++真的开始对 Windows 开发工具产生威胁，那么 VB 将会是受到影响最大的开发工具。但 Borland 仍然感到忧心，因为 VJ++仍然可能对于 Delphi 和 C++ Builder 产生一定的影响，这是 Borland 不乐意见到的。当然这也加速了 Borland 研发 Latté的决心，因为已经不能再拖了。

记得当时我还和 Borland 在亚洲新加坡 R&D 总部的 Mr. Inn Nam Yong 谈过 VJ＋+的表现以及对于 VJ++可能产生影响的忧虑。Mr. Yong 也说 VJ++的表现令他们吃惊。看来 Anders Hejlsberg 在 VJ++的编译器技术上下了苦功，其表现早已超过了当时一般的 Java 编译器技术，的确是令人刮目相看，更麻烦的是从 VJ++的身上依稀可以看到 Delphi 的身影。Borland 的 R&D 已经了解了这个情形，Borland 的编译器小组也在研究相关问题的技术。由此可见当时 Borland 已经如临大敌，开始准备相关的技术，并且已经掌握了初期的状况。

Microsoft VJ++在 1996 年 11 月终于正式推出了，全世界也都屏息以待，准备看着 VJ++会产生多少的毁灭力量，而 SUN 更准备看看 Microsoft 是否会违反任何 SUN 和 Microsoft 之间的 Java 协议。当然 SUN 是担心 Microsoft 想破坏 Java 的开发。VJ++在一开始果然获得了一些回响，毕竟这是 Microsoft 推出的 Java 工具，使用 Microsoft 开发工具的软件人员当然会考虑 VJ＋+。同时 VJ++也吸引了一些想使用 Java 语言、但是仍打算呆在 Windows 平台的开发人员。

不过 VJ++推出之后也很快受到了所有 Java 开发工具以及支持 Java 平台厂商的全面围剿。他们害怕 Microsoft 对 Java 市场的入侵，会让其他厂商再次无法生存。之后连 SUN 也开始领军围攻 Microsoft，因为 SUN 除了害怕 Microsoft 会慢慢地主宰 Java 平台和标准之外，还发现 Microsoft 正在很有技巧地逐步破坏 Java 语言和标准，例如 VJ++便提供了许多非标准的 Java 用法并且很明显地把 VJ++绑死在 Windows 平台，破坏 Java 的"Write Once，Run Everywhere"的美梦。而且，Java 开发人员如果大量使用 VJ++，那么便再也离不开 Windows 平台。Microsoft 计划通过提供一流的"类 Java 开发工具"来限制开发人员的自由选择权的企图是昭然若揭了。

由于 SUN 的带头批判，想使用 Java 的开发人员和企业很快地发现 VJ++并不是标准的 Java 开发工具，因此对于 VJ++的热情很快消退了下来。而 VJ++对于 Java 以及 Windows 开发工具的威胁也很快地解除了。VJ++对于 Microsoft 来说很可能是自 DOS 版的 Microsoft Pascal 之后第 2 次在开发工具的大失败。不过依我的观点来看，VJ++在本质上是一个优秀的产品，不论是编译器、Framework 和集成开发环境都有高水平之作。VJ++之所以败阵下来实在是因为形势比人强，Java 平台也是第一次不是由 Microsoft 所主宰的市场。在 Java 联军的合攻之下，即使是软件巨人也得回避三分。

因为第一次在 Java 出击就弄得灰头土脸，并且 SUN 摆明了不会允许 Microsoft 在 Java 平台成气候，使得 Microsoft 下定了和 SUN 正面开战、在 Java 市场上全面开火的决心，进而造成了 SUN 控告 Microsoft 违反 Java 合约的规定的结果，而 Microsoft 稍后则干脆把 Java 支持从操作系统中移除。当然，这是 Microsoft 和 SUN 之间的 Java 平台之战，已超出本书讨论的范围，也许应该由 Microsoft 或是 SUN 的人来说明这整个过程。

虽然事后证明 VJ++在 Java 开发工具是失败了，但是 Anders Hejlsberg 在 VJ++中花费的心力却没有白费，因为 VJ++的编译器技术以及 Framework 和集成开发环境的技术都在

稍后融入 Microsoft .NET 计划的基础核心技术之中。例如 C#的语言和 Java 很相像，C#的编译器技术想必也借重了许多当初 VJ++优秀编译器的技术，因此 C#编译器的最佳化结果也在一些方面胜过了现在许多原生 Windows 开发工具的编译器水准。Anders Hejlsberg 的努力正激活了 Java 和.NET 的正面决战。

IBM VisualAge For Java 的推出

IBM 在 PC 开发工具市场的表现一直令人摇头，因为其"玩玩便跑"的作风总是让人无法放心使用它的开发工具。但是也许是 IBM 的招牌太大，再加上它会免费向购买 IBM 机器或是软件的客户奉送 IBM 开发工具，因此也总是有人会去使用 IBM 的开发工具。我个人在受了 IBM VisualAge For C/C++的教训之后，便对 IBM 的开发工具敬谢不敏了。

IBM 当然不会放弃 Java 这个潜力无穷的市场，因为对于 IBM 来说，Java 不光是语言和开发工具而已；更重要的是 Java 平台牵涉到 IBM 和 SUN 在庞大商机的硬件和大客户之间的竞争。IBM 不光是要支持 Java，更想从 SUN 手中取得 Java 的主控权，因此对于重要的 Java 开发工具市场，IBM 自是不会缺席。IBM 很快采用了许多当初在 VisualAge For C/C++中相当受欢迎的元素作为开发 VisualAge For Java 的基础。例如 VisualAge For C/C++的项目管理功能、组件设计家等等。事实上使用过 VisualAge For C/C++的读者会发现 VisualAge For Java 非常的具有亲切感。不但所有的按钮都是采用圆形造型，甚至连激活时缓慢的感觉、整个集成开发环境温温吞吞的表现也非常相似。由于采用了 VisualAge For C/C++的部分观念和程序代码，再加上 IBM 拥有最丰富的资源，因此 VisualAge For Java 进展得很快。

1997 年 9 月，IBM 终于推出了 VisualAge For Java，开始直接和 SUN、Symantec 竞争。在 IBM 推出了 VisualAge For Java 之后，Borland 注定成为最后一个推出重量级 Java 开发工具的厂商。不过 IBM 的竞争目标明显不是 Symantec 和 Borland 等纯粹以 Java 开发工具为目标的厂商，而是 SUN 和 Microsoft。IBM 在 Java 技术方面采取了数个平行的战略，希望能够在 Java 世界中取得龙头地位，因为这关系到 IBM 最大业务--硬件销售、服务提供以及 IBM 操作系统销售的收入。如果 IBM 能够在 Java 世界取得决定性的地位，那么就可以侵蚀 SUN 的市场，最不济的情形则是不希望客户因为想要使用 Java 技术而自然地想到 SUN。至于另外一个锁定目标 Microsoft，IBM 则是打算通过 Java 日益扩大的声势来打击或是抑制之。

因此 IBM 一方面和 SUN 签订 Java 合约，取得 Java 使用的合法授权，另一方面又投入大量的研发资源开发自己的 JDK 版本以方便移植到 IBM 其他的专属平台，而且做得比 SUN 的 JDK 还要稳定和有效率，随后让 SUN 和 IBM 之间一直有不和的争执。接着 IBM 推出了 Java 开发工具，再次和 SUN 的 Java Workshop 竞争。不过从特性上来看，VisualAge For Java 锁定的客户群应该是 IBM 的客户、大型企业客户以及其他直接竞争对手的客户，例如 SUN 的客户以及 HP 的客户。VisualAge For Java 需要比较强劲的机器来执行，此外一开始的版本就非常注重团队开发的支持，不像其他的 Java 开发工具一开始都是注重在方便、实用的功能，稍后才逐渐强化团队开发的功能，这些差异都是 IBM 想争抢较大型客户的证明。这也可以从后来许多专业媒体和杂志在进行 Java 开发工具评比时 VisualAge For Java 几乎都在团队开发功能方面获得了最高的评价得知。由于 VisualAge For Java 一开始锁定的客户群和 Visual Café以及 JBuilder 锁定的客户群不同，因此在 Java 开发工具的竞争初期并没有发生严重的竞争冲突。但是随着 Visual Café和 JBuilder 逐渐往上仰攻企业市场，而 VisualAge For Java 为了扩大市场而开始降价进入一般 Java 大众市场之后，稍后的 Java 开发工具恶战也不可避免了。

第 2 阶段--Java 集成开发环境的战争

Borland 在初期以开发 Java JIT 编译器练兵之后，已经逐渐对于 Java 的技术有了掌握。

在 Java Workshop、VJ++和 VisualAge For Java 陆续推出之后，Borland 知道再也不能够延迟 JBuilder 的推出时间了，否则就注定要退出 Java 开发工具的市场。因此 Borland 对 JBuilder 的研发小组下了最后的通牒，一定要在 1997 年推出 JBuilder。

JBuilder 小组在竞争的第一阶段掌握了 Java 的 JIT 技术之后，立刻兵分多路展开了整个 JBuilder 的开发工作。虽然 Java 是一种全新的语言以及革命性的平台，但是开发工具总不外乎编译器、集成开发环境、数据库存取能力、Framework 以及其他的工具和 Plug-In 等。当时的 Latté小组有许多成员是从以前的 Borland C/C++转进来的，另外的一些成员则包括了 Borland 原本的软件研究成员、Paradox 成员、Visual dbase 成员以及从 Borland 外部找进来的新工程师。

在 Borland 开发 JBuilder 之时，由于 Java 尚没有完整的组件架构，也没有数据感知组件标准以方便地开发 Java 数据库应用程序，更加没有完整的 Java 可视化组件，因此 Borland 决定先自行开发一套组件组以便让 JBuilder 拥有最好的组件开发能力。这刚好又是 Borland 擅长的技术，因为 Borland 要为 Java 开发一套 Java Framework，这就是 JBCL(JavaBeans Component Library)的由来，而 JBCL 的架构稍后也成为 SUN 制定 JavaBean 的基础技术。

当时负责 JBCL 架构的 Architect 是 Joe Nunoll 先生。这位帅哥原本属于 Paradox 小组，在 Borland 逐渐失利于桌上型数据库战场之后，便转到 Latté小组专门负责设计 Latté 的组件架构。

而 JBCL 的主要实现工程师则是当初设计和实现 Borland C/C++ Framework-OWL 的总工程师 Carl Quinn。Carl Quinn 在组件设计和 Framework 方面都有丰富的经验，OWL 就技术而言也算是精品。因此在 Borland C/C++产品线停止之后，Carl Quinn 由 C/C++转换到 Java 跑道是很自然的事情，毕竟 C/C++和 Java 是很类似的。Carl 拥有丰富的经验，由他来带领开发 Latté的组件 Framework 是再适合不过了。

由于 Carl 在 JBCL 的努力和成果，稍后又负责了 Borland 的 Java 组件模型 Baja 的开发。之后 Carl 凭借着对于 JBCL 和设计 Baja 的经验，在 SUN 采用 Baja 做为 JavaBean 的核心基础技术之后便自然地受邀于 JavaBean 的开发小组。由于 Borland 在 Java 组件方面卓越的表现，因此也开启了 SUN 和 Borland 逐渐密切的合作。Borland 虽然在 Java 方面投入的时程稍晚，但是却凭借着扎实的技术而慢慢迎头赶上。

Latté的 Framework 开发在 Joe Nunoll 和 Carl Quinn 的带领之下有了稳定的发展。事实上 JBCL 的表现一直是非常优秀的。当 Latté随后正式推出时，JBCL 也是让 Latté得以脱颖而出的重要功能之一。Joe Nunoll 和 Carl Quinn 的功劳可谓不小，而我钦佩的 Carl Quinn 又再次在 Java 方面证明了他坚实的技术和在 Framework 方面丰富的设计和实现经验。

Java Framework 虽然重要，但也只是整个完整开发工具的支柱之一。Latté要推出仍然需要编译器和集成开发环境的功能。在 Java 编译器方面，Borland 和结束委托 Dr. Niklaus Worth 研究小组开发 Java JIT 编译器之后，Latté开发小组便开始展开研究的工作。当时 JIT 编译器已经全面开战，而 Borland 在 Latté尚未推出、还没有足够资源的情形下如果再介入 JIT 的战争，那么不但胜算不大，而且可能会严重影响 Latté的推出时程。因此 Latté小组决定先专心开发一个编译品质良好，而且能够和 Latté完美搭配的 Java 编译器，而不再强求效率至上。

从现在的观点来看，当时 Latté小组的决定是非常正确的。因为：第一，当时 Borland 的确没有太多的子弹；第二，Latté的时程再也不能拖延；第三，也是最重要的是稍后 SUN 宣布了开发 Hotspot 编译器技术的计划，顿时之间所有 Java JIT 编译器的风头都被 SUN 抢走了。特别是在稍后 SUN 决定将 Hotspot 内建在 JDK 中之后，争夺 Java JIT 编译

器不但变得没有意义，而且对于 Java 开发工具而言也没有什么附加价值了。因此在 SUN 的 Hotspot 编译技术揭露之后，Symantec 很快就在 Java JIT 编译器市场上销声匿迹了。
Latté小组确定了 Java 编译器的策略之后，立刻便接手 Dr. Niklaus Worth 研究小组的后续开发工作，同时开发 Latté的编译器、Latté的 Plug-In 以整合 Java 编译器到 Latté的集成开发环境中，并且也进行 Java 编译器最佳化的研究工作。当时这个 Java 编译器开发小组由 Carl Fravel 等人带领，同时也包括了 Borland 的编译器小组。Carl Fravel 主要负责 Latté的编译器以及编译器的 Plug-In 软件，此外他也参与了 Latté 数据库方面的开发。当时 Java 在数据库存取方面仍然相当弱势，Borland 为了加强 Latté 这方面的功能，决定先借重 Delphi 在数据库方面的成绩，即通过 JDBC 标准接口封装 Delphi 已经有的各种数据库驱动程序，让 Latté能够立刻连接到最多的数据库，这就是后来 JBuilder 的 DataGayteway。
除了 Carl 之外(嗯，Latté开发小组有两个 Carl)，Sergio Cardoso 也是 Latté编译器最佳化的成员。Sergio 原本是 Borland C/C 十+的开发者之一，专门研究 C/C++最佳化的技术，他和 Carl Quinn 等人一样转移到 Java 的产品线。Sergio 和 Carl Fravel 合作，负责打造 Latté的核心引擎。在 Latté上市之后，事实上每一版的 JBuilder 的编译器都有进步。就现在来说，在 JBuilder 7/8 中编译 Java 应用程序相当的快速。这说明 JBuilder 和当初 Borland C/C++一样，都是在初期版本快速地强化引擎、不断地提升速度。
当时的 Latté并没有一个真正的总 Architect，只有不同技术领域的 Architect，例如 Joe Nunoll。因此 Latté当时主要的舵手应该是产品经理以及 Borland 的 Java 资深研究员了。而产品经理以及 Java 资深研究员再和所有的各领域 Architect 讨论 Latté的开发方向。因此在 Latté的初期开发阶段，其模式就像罗马时期的合议制。
当时的 Latte 产品经理是 Klaus Krull(K. K.)。这位仁兄长得又高又大，也是一位相当热心的人。K. K.原本是 Paradox 和 dbase 小组的成员，负责 Paradox/dBase 的产品线。在 Latté小组成立之后 K. K.立刻跳槽到 Java 产品线来。事实上许多 Paradox/dBase 的工程师也都希望转换到 Java 产品来，因此当时在 Borland 内部掀起了很大的风浪，也造成了许多人离职。这段故事在稍后的 dbase 相关章节中将详细说明。
K. K.加入了 Latté小组之后，便积极地带领 Latté小组往前冲。也许是因为 K.K.在 Paradox 和 dBase 时表现得并不好，因此想借着 Latté证明自己的实力。不过 K. K.人也许很好，可是管理方面似乎仍然不甚灵光，Latté初期的进度仍然稍嫌缓慢。在 IBM VisualAge For Java 于 1997 年推出之后，Borland 高层下令 K. K.绝对不可以再度延迟进度，一定也要在 1997 年推出第一个 Latté版本。好在当时 Delphi 3 大获全胜，而 K. K.又是当时 Delphi 产品经理 Ben Riga 的好哥们，因此由 Delphi 转入 Latté的资源也就源源不绝，让 Latté的进度慢慢地赶上了。
至于当时 Latté架构的主要人物并不是稍后众人皆知的 Blake Stone，因为 Blake Stone 是在 Latté推出之后才加入 Borland 的。Latté初期的架构负责人应该是 Steve Shaughnessy。
Steve 是 Borland 的资深研究人员，也是当初 Borland 最早投入 Java 技术研究领域的人物之一。不过资深研究人员的缺点之一是喜欢不断地想象以及研究软件技术，但是对于产品进度的掌握却不是他们的专长，也不是他们最关心的事情。这就是为什么 Latté一开始的开发进度非常缓慢的原因。直到最后 K. K.加入并且面临了 Borland 高层和市场巨大的压力之后才匆匆地集中所有的资源和时间想要追上进度。
当 Latté小组开发 JBuilder 的第一个版本时，就想学习使用 Delphi 成功的 Open Tools API 特性，为 JBuilder 定义完整而且极具弹性的开放式集成开发环境。不过由于当时

Delphi 的 Open Tools API 仍然没有大量的采用接口程序设计的架构，考虑到 Java 拥有定义良好的接口机制，无法完全采用 Delphi 的 Open Tools API 的设计，所以一开始 JBuilder 集成开发环境的 Add-ins 功能开发得很缓慢。当时负责 JBuilder 集成开发环境中 Add-ins 功能的工程师除了 Carl Fravel 之外，另外一位主要的工程师则是 Greg Cole。当 Carl Fravel 和 Greg Cole 了解到无法直接借用 Delphi 的 Open Tools API 来设计 JBuilder 的 Add-ins 架构之后，就决定开始研发 JBuilder 本身的集成开发环境开放架构，并且直接使用接口程序设计的机制来设计 JBuilder 的开放架构，这是和当时的 Delphi 不一样的地方。而一直要到 Danny Thorpe 为 Object Pascal 程序语言加入了接口机制之后，Delphi 的 Open Tools API 才展开了第 2 波的大改版，使用接口机制来重新设计，也就是后来 Delphi 著名的 OTA 架构(Open Tools Architecture)。

1997 年 11 月，Latté终于完成并且推出市场。正式的产品名称被定为 Open JBuilder，这是为了强调 Borland 的 Java 开发工具就像 Java 本身一样是使用开放的架构。

在 Open JBuilder 1.0 推出之后，Java 开发工具市场总算是竞争者齐聚一堂了，每一家厂商终于一一地使出了真本领来竞逐 Java 开发工具的市场龙头。Open JBuilder 1.0 推出之后不久，几乎所有的信息媒体以及 Java 的专业杂志都进行了 Java 开发工具的评比，想要比较所有 Java 开发工具的优/缺点，并且让 Java 的使用者了解当时市场的老大 Visual Café是否能够面对新兴势力的挑战，保住市场第一的地位。

当时大多数杂志评比的目标包括了 Symantec 的 Visual Café、SUN 的 Java Workshop、IBM 的 VisualAge For Java 以及 Borland 的 Open JBuilder 1.0。对于 Symantec 来说，第一次的 Java 开发工具大会战是处于以逸待劳的情势，而且 Visual Café也是 4 个 Java 开发工具中唯一完全使用 C/C++语言撰写的，因此面对当时 Java 编译器还不够好、JVM 品质也未若今日精良的情形下，Visual Café的执行速度占了非常明显的优势，在功能方面当然也胜出其他竞争对手一截。

Symantec 的 Visual Café唯一的缺点就是在 Java 程序代码中加入了开发工具特有的程序代码卷标，造成使用 Visual Café撰写的 Java 程序代码不易使用在其他开发工具中的后果。而且 Visual Café在 Render Java 图形使用者接口时仍然拥有不十分精确的问题。

当时对 SUN 的 Java Workshop 的评比是比较保守的。毕竟 Java Workshop 是 Java 正宗厂商 SUN 推出的产品。虽然它不论在功能、执行效率方面都比不上竞争对手，而且小问题一大堆，但是为了给 SUN 面子，媒体仍然没有给予太多的苛责。甚至有一些媒体还称赞 SUN 有勇气开发一个完全使用 Java 语言撰写的 Java 开发工具，向全世界证明 Java 是能够用来开发大型应用程序的。不过虽然媒体和杂志很给 SUN 面子，但 Java Workshop 终究逃不过市场的考验，从此慢慢地退出了 Java 开发工具的市场。

在当时的评比中 IBM 的 VisualAge For Java 虽然是执行最为缓慢的 Java 开发工具，但是在高阶功能方面的表现却是遥遥领先所有的竞争对手。VisualAge For Java 的团队开发功能、项目管理功能以及可视化设计家都大幅超越了其他的 Java 开发工具。不过 VisualAge For Java 使用了专属的格式，因此其程序代码不容易使用在其他的工具中，而且 VisualAge For Java 的项目在进入了 Repository 之后也发生过整个 Repository 毁损的情形，因此当时 VisualAge For Java 在易用性方面的分数是比不上其他竞争对手的。

对于 Borland 的 Open JBuilder 1.0 来说，这个最晚进入竞争市场的工具在第 1 次的集体评比中最后的结果不如人意。原本 Java 的使用者以及专业媒体对于 Borland 的产品有着高度的期待。因为以 Borland 一向精于开发工具，而且是最后才推出 Java 开发工具的情形来推断，大多数人都认为 Open JBuilder 应该是准备最充分的，但是评比之

后的结果却不是如此。
首先 Open JBuilder 并不是纯粹使用 Java 撰写的开发工具，而是混合了 Java 和 Delphi 的程序代码。不过最后的执行效率不但比不上 Visual Café，也不比纯粹的 Java Workshop 快上多少。此外 Open JBuilder 在功能方面比不上 Visual Café，在可视化设计家和高阶功能方面又不是 VisualAge For Java 的对手。在比上不足，比下只有险胜的情形下 Open JBuilder 让当时许多人大失所望，当然也包括了我在内。因此在大多数的评比中，Open JBuilder 只得到中等的评价。当然这样的结果也反映在 Open JBuilder 的市场表现之上。
Hotspot 编译技术是个笑话吗？
1997 年市场上逐渐出现愈来愈多的 Java 开发工具，愈来愈多的人开始尝试使用 Java，却也有愈来愈多的人抱怨 Java 的执行效率。当时的 PC 不像今日动不动就拥有 1GHz 的执行效率以及 512MB RAM 的内存，以当时的机器来执行 Java 是很痛苦的事情。还记得当时我还没有动力足够的机器来跑 Open JBuilder，每一次执行 Open JBuilder 时就觉得受不了。当时我还开玩笑地说，机器从执行 Open JBuilder 到进入 Open JBuilder 的集成开发环境这段时间，早就够我使用 Delphi 写完一支程序了。造成 Java 执行效率缓慢的主要原因当然是 Java 编译器以及 JVM 的品质不够精良了。
为了急于让信息界接受 Java 成为标准，SUN 必须想办法克服这个问题。虽然克服 Java 执行缓慢的现象是当时几乎所有支持 Java 软件厂商都想解决的事情，但 Java 的正宗厂商 SUN 是责无旁贷的。也是因为 Java 执行效率的缓慢，当时也兴起了许多小的软件厂商开发各种技术和编译器来改善或是解决 Java 的这个致命缺点。很快 SUN 找到了一家小软件公司，这家公司以开发出"Adaptive Compiling"技术来加快 JVM 执行效率、以及使用类似的技术来改善 Java 编译器的品质而闻名。SUN 在了解到这些杰出的技术之后便立刻决定购买这家公司，并且根据他们的技术来实现 SUN 的下一代 Java 编译器以及 JVM，这就是稍后 SUN HotSpot 技术的由来。
SUN 投入新的 Java 编译技术之后不久，就有了初步的结果。根据这个新的技术编译出来的 Java ByteCode 以及新的 JVM 的执行效率果然比以前进步了许多。这让 SUN 更有信心，便立刻向世界公告了这个新的技术，并且命名为 HotSpot。SUN 宣称最后推出的 Java 编译器和 JVM 将提供类似 C++的执行效率。
在 SUN 公布了 HotSpot 技术之后，立刻引起了全世界 Java 使用者的狂热。人们认为一旦 SUN 推出这个技术，Java 将可望克服最后一个缺点，从而一统天下。与此同时，这也引起了信息业界非常大的讨论和争议。特别是 C/C++社群的人认为这根本是不可能的，虽然"Adaptive Compiling"非常的有创意，但是要和已经存在数年的 C++最佳化编译器比起来，Java 的 ByteCode 是不可能超越 C++的。但是从 SUN 在其时公布的一些 HotSpot 编译数字来看，"Adaptive Compiling"是非常有希望的，因为它改善的幅度实在是很大。因此全球相关的人员都在屏息以待，准备看看 SUN 最后的成果。
在 SUN 第 1 次公布 HotSpot 的推出时程之后，果然让所有 Java 的使用者都引颈期盼，恨不得 SUN 立刻推出这个技术，解除大众执行 Java 的痛苦。不过随着时间不断的接近，SUN 在最后关头又宣布因为研发不及因此要延迟 HotSpot 推出的时程。软件研发的工作延后在软件界见怪不怪，当时也没有引起太多的争议，不但让 SUN 争取到了更多的时间，也顺利地延后了推出的时程。
不过在 SUN 宣布的第 2 次推出时程到期时，SUN 仍然无法推出 HotSpot 技术。很快的 SUN 不得不再次宣告要延后 HotSpot 的时程。就在这样 SUN 不断跳票的戏码重复上演的情形下，终于开始有人笑称 HotSpot 根本是一个骗局，SUN 根本无法推出接近 C++执行效率的 Java 编译技术。

到了 1999 年左右，SUN 自知再也无法推拖 HotSpot 推出的时程了。因此在当年 8 月，当时 HotSpot 研发小组的领导人(一位博士，但是我已经忘记了他的名字)在 BorCon 上进行 Keynote Speech，正式向参加 BorCon 的人介绍 HotSpot 技术并且在现场展示了 HotSpot 的研发成果。虽然一切看起来都很棒，但是当现场的听众直接询问到底 HotSpot 技术是否能够超越 C++的执行效率时，这位博士却没有正面回答，只解释说在一些应用中 HotSpot 的确可以提供超越 C++的执行效率。我听了之后心中大概就已经知道 HotSpot 最终的结果了。

果然在 HotSpot 被迫推出市场之后，大家很快地了解到 HotSpot 和 C++的执行效率相比终究是还有一段距离，根本无法超越 C++的表现。这造成了当初一些热切期待的 C/C++程序员回到 C/C++语言的市场，并没有转换到 Java 市场。这也是为什么后来 C/C++市场虽然受到了 Java 的影响，但是仍然有大量的使用者和市场，并没有像当时许多人预测的那样将会有大量的 C/C++程序员进入 Java 市场，终究还是因为 Java 无法完全取代 C/C++语言来完成一些工作。而 SUN 呢？为了转移大家对于 HotSpot 的失望而开始把研发重点转到 Internet/Intranet、EJB 组件模型和 Java Mobile 系统方面的研发。轰动一时的 HotSpot 热潮也逐渐淡去。

现在 SUN 再也不怎么提起 HotSpot 编译器了，只是在每一个新版本的 JDK 中不断持续的改善 HotSpot 的编译品质。想起当初 SUN 对 HotSpot 不可一世的吹嘘最是令人感叹。不过 HotSpot 也不是一无是处，的确是精进了许多 Java ByteCode 的产生品质以及 JVM 的执行效率，只是没有达到当初 SUN 夸口逼近或是超越 C 语言编译器品质的程度。在目前状况下，HotSpot 能够让 Java 的编译品质在伺服端的效率有着显著的提升，提供非常不错的执行效率。但是在客户端，尤其是牵涉到图形使用者接口 Render 方面的应用时，仍然是相当缓慢的。

就 Borland 本身使用 Java 的情形来说，Borland 使用 Java 开发的 VisiBroker For Java 的执行效率已经相当接近 VisiBroker For C/C++的执行效率。因此如果再搭配使用品质良好的 JVM，那么根据 Borland 内部的测试数据显示，VisiBroker For Java 甚至在一些特定的应用中超越了 VisiBroker For C/C++。

HotSpot 在纷纷扰扰的这么多年之后到底是不是一些人讥笑的"笑话科技"呢？不同的人到现在可能还是有不同的答案吧。

Borland 的困境和选择

Open JBuilder 虽然赶在 1997 年最末的一班车推出，但在市场上的反映并不如预期的好。当然这是有许多原因的。首先是 Open JBuilder 太晚推出，初期的 Java 市场早已被其他的 Java 开发工具，特别是 Visual Café所占领；第二是 Open JBuilder 急着推出市场，因此在和其他 Java 开发工具竞争时并没有什么特别突出的功能、明显的优势，竞争力当然不够；第三是 Open JBuilder 于一开始就混合地使用了 Delphi 和 Java 程序代码，因此 Open JBuilder 激活以及窗体设计家的反应都很缓慢，不像 Visual Café 那种以纯 C/C++程序代码撰写的 Java 开发工具反应迅速，从而给许多程序员造成了不良的印象。IBM 的 VisualAge For Java 虽然也很迟缓，但在高阶的团队开发方面却支持得很好，而且通常会使用团队开发功能的使用者大都是属于企业或是大型用户，因此使用的机器配备也很好，对于 VisualAge For Java 的缓慢反应也还能够接受。

Open JBuilder 的表现不如预期，这让 Borland 很着急，因为其无法承受失去 Java 开发工具市场的损失。因此在 Borland 的 Java 开发工具研发小组中开始有了一些讨论，那就是如何让 Open JBuilder 能够后来居上，取得胜利的果实。针对 Open JBuilder 的失败原因，Open JBuilder 的开发人员开始反思是否也应该像 Visual Café一样使用 Delphi 重新打造 Open JBuilder，让 Open JBuilder 的执行反应加快到使用者能够接受

的地步，因为在当时 Borland 实在已经无法再加快 Java 的执行速度。此外使用 Delphi 开发 Open JBuilder 的窗体设计家也可以避免许多 JDK 的臭虫，不会因 SUN 开发或是改善 JDK 的时程而影响到 Open JBuilder 的开发周期。

这个想法在 Open JBuilder 的内部引起了很大的争议。使用 Delphi 重写 Open JBuilder 的集成开发环境可以拥有许多短期的效益而且产品马上会有明显的改善，可以拥有和其他竞争对手一搏的本钱。不过反对的人则认为使用原生开发工具开发 Java 的工具是走回头路。这些人认为 Java 有朝一日一定会开发到成熟的阶段，到时 Open JBuilder 就会拥有最后的胜利，现在只是一时的挫折，没有必要灰心。

对于 Borland 来说，如何继续 Open JBuilder 是一个困难的抉择，因为当时 Borland 急需收入的挹注，而 Open JBuilder 的研发费用惊人，光靠 Delphi 力撑实在是很辛苦。不过如果再回到使用 Delphi 开发，那么可能又会失去未来的机会，这到底应该如何决定呢？

Java 天才的加入

这一切的答案在 Open JBuilder 的新产品架构领导人 Blake Stone 加入后才逐渐明朗。Blake Stone 原本是 DSW Systems Corporation 公司的技术主管，而 DSW 公司一向和 Borland 互动良好，许多 DSW 公司的人都曾在 Borland 的 Conference(BorCon)中负责技术讲座。Blake Stone 先生也在 1997 年的 BorCon 中负责了一个讲座。也许是 Blake Stone 和 Borland 在这次的 BorCon 中合作愉快，Borland 也很赏识 Blake Stone 的技术和才华，因此在 BorCon 结束之后不久，Borland 便和 Blake Stone 接触，看看 Blake 是否有意愿加入 Borland 的 Java 研发小组。也许是天意吧，在 Borland 失去了 Anders 这个天才之后，老天又给了 Borland 一个弥补软件天才的机会。

在 Borland 和 Blake 接触之后，Blake 不但对于 Java 未来的潜力看好，而且因为 Blake 也曾使用 Delphi，对于 Borland 研发开发工具的能力相当有信心。更凑巧的是由于 Open JBuilder 1.0 的不尽人意，因此此时刚好有一个 Open JBuilder 的 Architect 离职，让 Blake 立刻有了适当的职位。没有多久 Blake 便答应进入 Borland 作为 JBuilder 的 Architect，目的是带领 JBuilder 成为最成功的 Java 开发工具。由于 Blake 惊人的天分，因此很快就成为 JBuilder 的主要 Architect 以及技术的主领导者，JBuilder 未来开发的 Java 技术都由 Blake 负责研究和研发的工作。

Blake 进入 JBuilder 开发小组之后，面临的第一个挑战便是如何改造 Open JBuilder，让它执行得更为顺利，并且能够在竞争群中脱颖而出。当然 Blake 必须做的第一个抉择就是 Open JBuilder 到底该走向纯 Java 的开发工具或是改成原生的 Windows Java 开发工具。Blake 并没有迟疑多久，便决定把 JBuilder 带向纯 Java 的开发工具，使用 Java 语言本身来打造整个 JBuilder。Blake 做了如此的决定是有许多原因的。首先是 Blake 希望通过使用 Java 语言开发 JBuilder 本身来让 Borland 的工程师彻底掌握 Java 的技术，也希望通过这样的开发来证明 Java 的实用性。就像 Delphi 本身就是使用 Object Pascal 和 Delphi 研发、Borland 通过 Object Pascal 证明了 Delphi 的实用性和可靠性一样，Blake 也希望使用 JBuilder 来证明 Java 语言的可用性。

第 2 点是因为打造纯 Java 开发工具可以让 JBuilder 通过 Java 跨平台的特性把 JBuilder 推向其他所有支持 Java 的平台，让 Borland 能够穿透到以往无法进入的市场，这样可以让 JBuilder 的潜在市场和客户比竞争对手的更宽广、更多。

第 3 点因素则是 Blake 希望通过这个行动让 Borland 掌握 Java 的核心技术，最好能够和 SUN 有更密切的互动，让 Borland 能够在 Java 领域取得相关的领导地位。因为在和以往 Microsoft 交手的过程中，Borland 深深了解到如果无法在一个技术领域取得第 1 或是第 2 的地位，那么终将成为微不足道的角色，被市场淘汰出局。

Blake 在 JBuilder 研发方向制定的策略事后都被证明是正确的。后来 JBuilder 果然能够支持 Windows、Linux 和 Solaris 平台，成为当时架构最大、最复杂的 Java 应用程序。更重要的是 SUN 充分肯定了 Borland 在 Java 方面卓越的技术，进而采用 Borland 的 Baja 技术制定 Java Bean 规格并且邀请 Borland 共同参与开发 Java 的 JDK。Blake 在 JBuilder 早期设定了成功的趋势，奠定了 JBuilder 成功的基础。稍后 JBuilder 新的产品经理 Tony de la Lama 又成功地制订了 JBuilder 的市场研发脚步和竞争策略，终于让 JBuilder 在 3.5 版本之后一飞冲天，成为 Java 开发工具的翘首。

在 Blake 加入 JBuilder 开发团队并且决定了 JBuilder 走向之后，很快整个 JBuilder 的开发方向便朝着他决定的方向快速前进。Blake 也激活了 JBuilder 庞大的纯 Java 开发工具的计划。1998 年 JBuilder 研发小组在 Blake 的带领之下很快地交出了第 1 张成绩单，那就是 JBuilder 2 的推出。

JBuilder 2 的战略目标并不是成为完全的纯 Java 开发工具，而是为了快速跟上其他 Java 开发工具的功能，并且提升 Open JBuilder 1.0 为人诟病的缓慢执行速度以及问题多多的窗体设计家。

无疑 JBuilder　2 是非常成功的。我所谓的成功并不是指 JBuilder 在销售上的成功，而是指 Blake 为 JBuilder 2.0 设定的目标。因为 JBuilder 2.0 推出之后很明显的比 Open JBuilder 1.0 看起来成熟多了，而且在执行速度、包含的功能等方面都到达了合理的地步，也让 JBuilder 正式进入 Java 开发工具第一方阵的竞争群。在 Blake 的努力下，JBuilder 2.0 的实现程序代码已经进步到使用 25％Delphi 程序代码和 75％Java 程序代码，离纯 Java 开发工具已经愈来愈近了。Borland 也开始从 JBuilder 2.0 的身上看到了未来的曙光。也是上天注定，这个时候正是 Delphi 逼近于全盛的时期，需要 JBuilder 接棒才能够让 Borland 持续地成长。

JBuilder 2.0 的渐入佳境除了归功于 Blake Stone 之外，另外一个重要的原因便是此时 JBuilder 的产品经理也换成了颇具眼光的 Tony de la Lama。Tony 也像 K. K.一样是从 Visual dbase 小组转来的，而且在 K. K.离开了 Borland 之后接手成为 JBuilder 的舵手。本来没有人看好 Tony 的，没有想到 Tony 却是一个雄才大略的人物。

Tony 显然和 Blake 合作无间。在 JBuilder 一开始于产品和技术还相对处于劣势的时期，Tony 知道最重要的工作是先把产品做好，再求其他的策略。因此 JBuilder 在 2/3 版时主要是由 Blake Stone 操刀、Tony 为辅。稍后当 JBuilder 逐渐成为 Java 开发工具的重要角色之后，便由 Tony 主导在市场、产品定位和竞争策略方面运筹帷幄。

Blake 在研发 JBuilder 2.0 时，便设定了下面的目标，准备重新奠定 Open JBuilder 的竞争实力。

- 呈现精确的设计时期可视化效果。这方面胜出 Visual Café许多，也是 JBuilder 准备强攻 Visual Café弱点的策略
- 充分利用 Java 平台的特性
- 以 Java 来思考开发工具的开发。这一点非常重要，也是日后 JBuilder 胜出其他 Java 开发工具的重要因素
- 开始为 Java 设计组件架构。当时这项研发计划命名为 Baja，由 Carl Quinn 负责。Baja 也是日后 JavaBean 的前身
- 打算以纯 Java 撰写 JBuilder 并且为移植到其他平台做准备

在 JDK 1.1 推出了 JNI(Java Native Interface)之后，JBuilder 开发小组的工作就更为顺利了，因为他们可以通过 JNI 呼叫使用 Delphi 2 撰写的程序代码，呼叫 Borland 自行开发的原生 Virtual Machine，以快速地编译 Java 程序。此外由于 JBuilder 2.0 尚无法成为完全的 Java 开发工具，因此 Blake 在 Delphi 方面也进行了更好的最佳化调整，

以加速 JBuilder 激活和执行的速度。当然在 JBuilder 的窗体设计家方面，Blake 更是决定投入大量的资源，以求解决 Render 和 JDK 臭虫的问题。由于 Blake 的决定，Borland 发现并且要求 SUN 解决了当时 JDK 和 AWT/SWING 初期的许多臭虫。其中有一些是 SUN 来不
及或是不愿意立刻修改的，为了不延误 JBuilder 的开发，Borland 直接予以解决再提交给 SUN 作为参考。

当时 Blake 为了让 JBuilder 中 Delphi 和 Java 的程序代码有更好的整合和表现，甚至研发了许多低阶的技术来暂时强化 JBuilder 2.0 的执行效果，并加强 Java 和 Object Pascal 语言之间互动和交换数据的机制。嗯，看来 Blake 在数年前便想到并且解决了许多现今.NET 的技术问题呢。

Blake Stone 成功地带领 JBuilder 开发小组推出了 JBuilder 2.0，这只是 Blake 为 JBuilder 实施的第一阶段竞争步骤。当 JBuilder 2.0 获得了初步的成果之后，Blake 也正式激活了以纯 Java 打造 JBuilder 的计划。JBuilder 3.0 是 Blake 瞄准的第一个版本。在 JBuilder 2.0 之后的 JBuilder 研发工作中，Blake 已经拥有愈来愈多的资源，整个 JBuilder 开发小组的规模也开始和 Borland 的 RAD 部门不相上下了。

1999 年 8 月，在 Blake Stone 和 JBuilder 团队全力催生之下，几乎是完全由 Java 开发的 JBuilder 3.0 也在距离 JBuilder 2.0 推出一年之后正式推出了。如果说 JBuilder 2.0 是为了让 JBuilder 赶上其他 Java 开发工具的版本，那么 JBuilder 3.0 的定位无疑就是在 Borland 推出纯 Java 的 JBuilder 之前从落后于竞争对手到超越竞争对手的一个产品。

虽然 Borland 进入 Java 开发工具市场非常晚，但是在经过了 3 个版本的努力之后，终于在 JBuilder 3 毕其功于一役，进入了 Java 开发工具的领先群。此时当初第 1 名的 Java 开发工具 Visual Café正每况愈下，逐渐接近被 JBuilder 超越的命运了。

还记得在 1999 年的 BorCon 中，我曾经和 Blake Stone 有过短暂的交谈，明白了为什么许多人都说 Blake Stone 是一位天才型的软件人物。当时我去听其中一场讨论(如何调整 InterBase 执行效率的 Seminar)，没有想到坐下来之后才发现 Blake Stone 就坐在旁边。之后，我一直在暗中观察他。只见他在 Seminar 开始之后就拿出了 Notebook 专心写程序。我当时便想，Blake 参加这个 Seminar 大概只是消磨时间，主要是写写 JBuilder 的程序，并不是真的想听这个 Seminar 的内容。选择 InterBase 这个 Seminar 纯粹是因为人比较少，不会受到太多的打扰吧。知道了 Blake 的举动之后，我也一直想移动身体朝向 Blake，希望看看天才写的程序代码是什么样子？但是出乎我意料之外的是，当 Seminar 结束之后，主讲人开始接受询问问题，Blake 却不断地举手发问。

这令我大吃一惊，因为整场专心写程序的 Blake 看起来能够一心多用，不但脑袋可以想东西，手指可以打键盘，心思还能够倾听 Seminar 的内容，真令我佩服。Seminar 结束之后，我和 Blake 交谈了数句寒暄的话，恭喜他在 JBuilder 方面的成就。Blake 转身离开时，从身后看简直就像一位小姐。因为 Blake 身材纤细，又留了一头长发，不知情的人从身后看一定会认为这是一位美丽的小姐呢。

当 JBuilder 3 在市场上接受开发者的考验时，虽然 JBuilder 的市场占有率排名还不是第一，但是已经和主要竞争对手非常接近。从成长分析中也可以看到，JBuilder 是以快速的速度成长，而 Visual Café却不断地呈现下滑趋势。至于 VisualAge For Java，也几乎停滞不前。JBuilder 对于所有 Java 开发工具竞争对手的威胁与日俱增，Symantec 和 IBM 再也不敢轻视 JBuilder 的实力。事实上此时 JBuilder 的王者之姿也隐然若现了。

不过 Blake 并没有稍做停顿，给其他的 Java 竞争对手以喘息之机，而是很快蓄积力量，准备在产品面进行最后的一击。

2000 年 3 月 14 日，对于 JBuilder 来说是最重要的一个里程碑，因为这一天是 Borland 正式推出 JBuilder 3.5 的日子，也是 Borland 的 JBuilder 小组在经过了数年的努力之后终于拥有了一个纯 Java 打造的 Java 开发工具的日子。

JBuilder 3.5 不单是 Borland 第一个完全使用 Java 撰写的开发工具，也算是业界中第一个纯 Java 开发工具。虽然数年前的 Java Workshop 也是使用 Java 开发的，但是 Java Workshop 提供的功能远不如 JBuilder 3.5，此外 JBuilder 3.5 的复杂度和难度也比 Java Workshop 高太多了。

JBuilder 3.5 推出之后，立刻风靡了 Java 开发工具市场。大量的 Java 开发人员迫不及待地升级到 JBuilder 3.5，因为许多 Java 程序员都想使用一个纯正的 Java 开发工具。此时 JBuilder 在市场上的占有率排名已经达到第一位，而且正朝市场占有率 50％的目标迈进。依我的眼光来看，JBuilder 到了 3.5 版之后，在技术和产品方面的竞争已经告一段落。因为在这个阶段，JBuilder 本质方面的使命已经完成了。JBuilder 3.5 之后的竞争要点是在市场和产品策略上。如何让 JBuilder 超过 50％的占有率是 JBuilder 4 的任务。此时 JBuilder 的领导也由技术 Architect 转到产品经理 Tony de la Lama 身上。也是由于 Tony de la Lama 继 Blake Stone 之后表现得相当出色，很快 Tony de la Lama 就升为 Borland Java 事业部门的 Director，继 Blake Stone 之后再为 JBuilder 开辟一个光荣的战场。

JBuilder 从混合代码的 1.0 演变到 3.5 的纯 Java 开发工具，每一个版本核心都有精进。最后我列出 JBuilder 的演进史，让读者也能够了解 JBuilder 的变化。

	Delphi 程序代码	Java 程序代码
JBuilder 1.0	40	60
JBuilder 2.0	25	75
JBuilder 3.0	10	90
JBuilder 3.5	0	100

JBuilder 3.5 推出之后，Java 开发工具不可避免地开始进入了第 3 个大混战时期。

Blake Stone 的荣耀

1999 年 7 月，Borland 首次在公司内部设立首席科学家的荣誉职位，该职位颁给 Borland 最优秀和重要的软件人员。除了"首席科学家"光荣称号外，当然也有丰厚的物质奖赏。Blake Stone 当时就和 Chuck Jazdzewski 以及 Andreas Vogel 同时获得了 Borland"首席科学家"大奖。Blake 对于 JBuilder 的贡献也算是实至名归。在进入 Borland 短短的数年时间内，就以最年轻的软件人员身份获得此项大奖，这也证明了他惊人的实力。虽然 Blake 不像 Anders 那么锋芒毕露，也不像 Danny 一样著作丰富，但他本身在软件上的修行已属顶尖。下面列出他荣耀的历史：

- Borland 最年轻的首席科学家
- 成功带领 JBuilder 成为世界第一的 Java 开发工具
- 成功克服 Linux 上当初没有标准 JDK，让 JBuilder 能够在 Linux 上执行
- 世界 Java 专业论坛的主讲人
- 主导 Borland Java 开发工具和技术的关键人物

现在 Blake 愈来愈受到 Borland 的重视。除了原本的 JBuilder 产品之外，后来 Borland 并购 OptimizeIt、进一步强化 JBuilder 整体竞争力也是 Blake 的主张。Blake 已经慢慢从 JBuilder 产品线转而成为负责 Borland 大部分 Java 技术的关键人物。日前 Borland 又宣布 Blake 也将负责公司内新的生命周期管理软件的研发。看来 Borland 下一代 Java 产品也将由 Blake 贡献心力。继续加油吧，Blake！

第 3 阶段--大混战

"人人有希望，个个没把握"应该是对这个 Java 竞争时期最好的叙述了。在 JBuilder 逐渐成为 Java 开发工具的领导者、同时传统霸主 Visual Café也逐渐没落之时，对于每一个 Java 开发工具厂商而言，这似乎都是最后一搏的机会了。时间已经进入了公元 2000 年。虽然 Java 开发工具的市场独立于其他语言的工具市场，而且这个市场的规模也在不断地成长，但明显没有像当初 Gartner Group 等机构预测的那样有足够大的规模。更麻烦的是 Java 开发工具的价格显然在许多免费工具的竞争力之下，这些厂商的获利也不如预期。但是在每一家 Java 开发工具厂商都已经投入大量的资源情况之下，除了市场第 1、2 名以外，其他的开发工具厂商都势必无法拥有足够的获利。除非 Java 开发工具厂商有其他的利益或是因素而能够忍受不佳的获利甚至赔钱，否则迟早许多厂商都必须退出这个市场。

或许所有进入这个市场的厂商都没有料到，Java 开发工具会有如此激烈的竞争。当初这些厂商都认为 Java 是一个极具潜力而且没有"Microsoft"竞争的市场，因此一定比较好生存。但却没有想到因为没有"Microsoft"，反而所有其他厂商几乎都进入了这个市场，甚至一些名不经传的小公司都以为可以分一杯羹。想在这个市场成名立万就必须经过激烈的割喉竞争。

同样在 2000 年 11 月，Borland 向所有 Java 开发工具竞争对手挥出了致胜的一击，那就是 JBuilder 4 的推出。JBuilder 4 对于 Borland 来说是一个重要的 Java 里程碑，是 Borland 在 JBuilder 3.5 之后乘胜追击之作。在 JBuilder 3.5 以纯 Java 开发工具的号召获得了 Java 程序员的青睐之后，Borland 毫不放松到手的胜利，立刻以 JBuilder 4 一举冲破了 Java 开发工具 50％的市场占有率，正式成为 Java 开发工具的王者。

JBuilder 4 不但在销售上大获全胜，也让 Borland 在经过数年努力之后终于在 Java 开发工具重享当初 C/C++王者的荣耀。此外 JBuilder 4 产品本身也得到了专业 Java 媒体和杂志的高度评价，一致认为 Borland 的 JBuilder 是最好的 Java 开发工具。Borland 可以说是里外兼得。

在 Borland 以 JBuilder 4 横扫千军之际，唯一还能够与之抗衡的就只有 IBM 的 VisualAge For Java 了。而昔日的 Java 开发工具霸主 Visual Café已经沦落到第 3 名的地位，且其市场占有率和影响力还在快速的下降之中。Symantec 当初辛辛苦苦花了 3 年时间打造的 Java 王朝就在二三年之间很快幻化成泡沫，这也造成了 Symantec 完全退出开发工具市场、全心开发防毒和企业解决方案应用软件的结局。

因此在 2000 年 3 月，Symantec 把 Visual Café卖给了新成立的 WebGain 公司。Visual Café从此慢慢地从通用 Java 开发工具转变为专属的 Java 开发工具，其 Java 开发工具市场的地位也变得无足轻重了。

另外一个重要的现象是在 Borland 的 JBuilder 一轮猛攻并且成为 Java 开发工具的领导者之后，JBuilder 也慢慢地被 Java 开发人员视为正统的 Java 开发工具。因此当时许多在 Windows 平台使用 Java 的开发者都逐渐转至使用 JBuilder。而 Microsoft 的 VJ++由于不支持最新的 SUN Java 标准，又混合使用了许多 Microsoft 特定的功能，最后还和 SUN 发生了法律上的争执，也逐渐被视为非正统的 Java 开发工具，使用者逐渐流失。因此在 2000 年左右，当所有的 Java 开发工具都呈现成长趋势的时候，Microsoft 的 VJ++却出现了大幅的衰退，是唯一呈现负成长的"类 Java 开发工具"。此后不久，Microsoft 慢慢地退出 Java 市场，开始准备另起炉灶，再以 2 年后的.NET 进行反击。

JBuilder 4 算是 Borland 毕其功于一役的版本，因为 JBuilder 4 不但在产品功能方面大幅领先于其他的竞争对手，在市场占有率方面也终于达到了 Borland 一直想要达到的目标。到了这个时期，Blake 主导 JBuilder 的工作也算是圆满达成目标，在接下来 JBuilder 的开发过程中就开始由 Tony de la Lama 操盘，Blake 则转向主导 JBuilder 和

Java 先进技术的研发工作，并且稍后和 RAD 的 Chuck 一起，获得了 Borland 首席科学家的荣耀。

第 4 阶段--谁跟的速度最快

在 Java 开发工具进入大混战阶段之后，看起来虽然热闹、生气蓬勃，事实上却是惊险不已。各种 Java 开发工具多得令人眼花缭乱，每种 Java 工具也都有人使用。从要钱的到不要钱的，从完整集成开发环境到只提供最简单的 Java 编辑器，各种产品都有厂商提供，的确是一个百家争鸣的时代。

Borland 为了在这个阶段取得领先地位，以避免被淘汰出局，因此激活了 Borland 前所未有的开发模式，那就是在 JBuilder 的开发团队中同时有二组并行且彼此竞争合作的 JBuilder 小组。当其中一个小组开发目前的 JBuilder 版本时，另外一个小组便开始着手研究和开发下一个版本的 JBuilder。开发目前版本 JBuilder 的小组完成之后，便接着再使用另外一个小组已经开发出的成果持续开发下下版本的 JBuilder。这种作战方式从 JBuilder 3.5 之后便开始显现成果。JBuilder 不但及时地紧紧跟随着每一个 JDK 的版本，成为市场上最符合 JDK 要求的 Java 开发工具，也渐渐形成每半年推出一个新版本的速度，让 JBuilder 远远超越竞争对手，成为唯一的领先者。

JBuilder 成功的竞争策略，让 JBuilder 的最大竞争对手 IBM 的 Visual Age For Java 和 WebGain 愈显吃力，更不用说其他的 Java 小厂商了。在这个阶段 Java 开发工具开始了残酷的最后生存战。Java 开发工具市场注定将像当初 PC 平台上的 C/C++开发工具市场一样，只有少数的厂商才能存活下来。事实也证明从 2000 年之后不断有 Java 厂商被迫退出市场，甚至是面临关闭的命运。造成这种现象以及 Borland 能够脱颖而出是因为下面的原因：

■ Java 开发工具市场虽然有成长，但相对于其他种类的软件来说，毕竟是比较小的市场，因此开发工具厂商必须达到经济规模才能够获利，才能生存下去

■ Java 开发工具的市场并没有像 Gartner Group 在公元 2000 年预测的那样会有巨大的成长。事实上当时就是因为许多研究机构看好 Java 市场的成长潜力，因此才吸引许多软件厂商不断地进入。但是过多的软件厂商和不够大的市场注定会淘汰许多的竞争者

■ Borland 在残酷的开发工具竞争市场生存了将近 10 几年之久，因此当然知道竞争的激烈，也深黯竞争之道。

■ 最重要的一点，是 Java 开发工具市场对于 Borland 来说是生存之战。在 Borland 的 RAD 开发工具到达了饱和之后，Borland 必须靠新开辟的 Java 开发工具市场才能继续茁壮发展

在经历这一波的残酷淘汰赛后，每家厂商的心态昭然若揭。对于一些小厂商而言，无声无息的消失是很自然的事情，即使是大厂商也不见得好到那里。例如 IBM 在 Visual Age For Java 经营了 4 个版本之后，又采用了像当初对待 Visual Age For C/C++一样的手法。我早在数年前便已经预言 IBM 终会放弃 VisualAge For Java，如今果然被我言中。我之所以能预知结果，并不是能够洞察先机，而是从 IBM 处理 PC 软件的一贯模式推知的。PC 的软件经营金额对于 IBM 来说是九牛一毛，根本无足轻重，只是 IBM 竞争的手段之一，特别是在经营结果不甚理想之后自然是第一个开刀的对象。不过最惨的应该是 WebGain 了，曾几何时是市场第一的 Visual Café也将面临再次被卖掉的命运。这个比快的淘汰赛中，JBuilder 和它的孪生兄弟、Oracle 的 Jdeveloper 一起遥遥领先，而 IBM 则已经准备放弃 Visual Age For Java，而和 BEA WebLogic 搭配的 WebGain 也逐渐长路将尽了。

第 5 阶段--谁能走的最久

当 Borland 以 JBuilder 4 横扫千军之后，剩下的主要竞争对手就只剩下 IBM 的 VisualAge

For Java 了。不过 VisualAge For Java 和一般的 Java 开发工具定位不太一样，而且大多数的 VisualAge For Java 使用者都是 IBM 的客户。此外 VisualAge For Java 最强的功能是在团队开发方面，而 JBuilder 在这方面一直不算是做得很好。

为了和 VisualAge For Java 进行最后的决战，JBuilder 小组决定在 JBuilder 中大幅强化团队开发方面的功能，期望击溃 VisualAge For Java 最后的防线。2001 年 6 月，Borland 推出了 JBuilder 5，除了增加 JBuilder 对于愈来愈风行的各种 EJB 应用程序服务器的支持之外，还加入了可视化 EJB 设计家以及支持 CVS、Rational ClearCase 和 Visual SourceSafe 等团队开发和原始码管理的功能，开始投注更多的资源在 VisualAge For Java 强项的功能上。

JBuilder 5 推出之后，果然改变了许多 Java 专业媒体和杂志对于 JBuilder 在团队和大型企业开发能力的评价。此外由于 JBuilder 是所有 Java 开发工具中支持 EJB 应用程序服务器最齐全和最好的工具，因此从 JBuilder 5 开始，一些应用程序服务器厂商便逐渐地建议客户搭配 JBuilder 来开发 Web 和 EJB 应用系统。这个现象也开始微妙地影响了 JBuilder 的定位并且开始产生许多奇妙的效果。

首先，为了在市场上取得最大的占有率，Borland 一直坚持 JBuilder 必须支持所有市场上重要的 EJB 应用程序服务器，特别是对于市场的领导 EJB 应用程序服务器，例如 BEA 的 WebLogic 和 IBM 的 WebSphere，应该有最好的支持，以吸引这些 EJB 应用程序服务器的使用者来使用 JBuilder。但是由于 Borland 自己也有 EJB 应用程序服务器，而 Borland 的 EJB 应用程序服务器部门希望 JBuilder 能够先支持 Borland 自己的 EJB 应用程序服务器，再考虑其他的 EJB 应用程序服务器。因此当时 Borland 的 JBuilder 小组和 Borland Enterprise Server 小组之间有了一些小争吵。不过由于 JBuilder 是属于 Borland Java RAD 部门的产品，因此 JBuilder 小组仍然决定成为市场上支持 WebLogic 和 WebSphere 最好的 Java 开发工具。只是 JBuilder 小组也把 Borland Enterprise Server 当成是一线应用程序服务器，决定给予的支持就像给予 WebLogic 和 WebSphere 的支持一样。

第二个现象是一些 EJB 应用程序服务器开始和 JBuilder Bundle(捆绑)在一起销售，或是建议客户使用 JBuilder。因为在 Java 开发工具重新洗牌之后，JBuilder 已经是市场的领导者，其他的应用程序服务器厂商在可选择的 Java 开发工具愈来愈少或是原开发工具厂商退出 Java 市场之后，也不敢违抗市场的主导力量，当然纷纷搭乘 JBuilder 的便车了。

在 JBuilder 5 成功的在团队开发方面给予了 VisualAge For Java 极大的压力后，Borland 于同年的 11 月再次进逼，发表了 JBuilder 6，成为压垮 VisualAge For Java 的最后一根稻草。

JBuilder 6 不但继续强化团队开发能力，而且已经成为支持 EJB 的最好工具。另外 JBuilder 6 又开始整合 UML 和 Extreme Programming 方面的功能，比起 VisualAge For Java 已经先进了许多。而 VisualAge For Java 在开发脚步迟缓的情形下，早已跟不上 JBuilder 的健步如飞。更麻烦的是从 JBuilder 5 之后，JBuilder 成功地打入了企业市场，侵蚀了原本 IBM 的客户并且动摇了 Visual Age For Java 最后的大本营。VisualAge For Java 在功能和市场方面节节败退，已经到了穷途末路的地步了。

2001 年 12 月左右，IBM 终于宣布把 Visual Age For Java 开放给 Eclipse 计划，正式结束了 VisualAge For Java 五年来在 Java 开发工具市场的竞争。IBM 在久战不下，Visual Age For Java 又无法替 IBM 带来充分的利润之后，正好借 Open Source 的名义把 Visual Age For Java 拱手奉送。还可以利用 VisualAge For Java 最后的价值为 IBM 打打广告、做做形象。不过回头看看 IBM 在开发工具市场的记录，却是惨不忍睹，对于客户而言也没有什么保障。

当 JBuilder 6 成功地摧毁了 VisualAge For Java 的防线之后，连带在市场上排名第三的 Visual Café也无法再支撑下去，因为 Visual Café的拥有公司 WebGain 在失去了 BEA 的支持之后，已经没有能力再在 Java 开发工具市场竞争下去了。

第 6 阶段--胜利者的出线

2002 年 Borland 仍然以半年一个版本的速度又如期推出了 JBuilder 7。Borland 这种推出新产品的速度简直令人无法置信。JBuilder 的竞争对手们也早已一个一个的累倒在地上，纷纷出局了。JBuilder 很明显的已经成为了最后的胜利者。

同年 6 月，WebGain 在找不到后继的资金和投资者之后，决定把 Visual Café卖掉并且结束 WebGain 的运营。不久之后 TogetherSoft 以相当便宜的价格购买了 Visual Café，准备正式进军 Java 开发工具市场。而 TogetherSoft 加入 Java 开发工具战火也代表着新一波 Java 开发工具的竞争，这在稍后会继续说明。

Borland 的 JBuilder 在第 7 版虽然成为了王者，但是这并不代表已经打遍天下无敌手。因为在 Visual Café被 Case 和 UML 模型工具开发厂商购买之后，新一波的 Java 开发工具之争已经隐约成形了。此外当初由 Delbert Yocam License(授权)给 Oracle 的 JDeveloper 也表现得愈来愈好，开始给 JBuilder 造成不小的威胁。

第 7 阶段--Java 开发工具和 Case Tool 结合的趋势

Java 已经明显的成为大型企业开发应用系统的首选，许多名列 Fortune 1000 中的大企业也都开始采用 Java 的应用方案。在 Java 成功地穿透了企业市场之后，随着 Java 开发工具的流行，Java 使用者也开始要求开发工具必须结合 OOA/OOD 的功能，让 Java 使用者能够以面向对象的方式开发大型的系统。于是从 JBuilder 6 开始，Borland 便在 JBuilder 中逐渐加入 OO 和 UML 的功能，准备在 JBuilder 已经于 Java 开发工具金字塔中/底层成功的攻城掠地之后，再把 JBuilder 推向金字塔顶端。不过 JBuilder 的举动自然也引起了另外一群软件厂商的紧张，进而隐隐地激活了另一波的竞争，这些软件厂商就是开发 Case 以及 UML 工具的软件公司。

话说从 2000 年许多软件公司经历了高成长之后，从 2001 年起，这些公司为了继续维护成长以期获利，必须在既有的产品之外想办法再扩充产品线。JBuilder 就是一个很好的例子。JBuilder 只有不断地扩展它的功能面以及使用者群，才能够持续地在 Java 开发工具市场成长。对于像提供面向对象和 UML 工具的厂商来说(例如 Rational 和 TogetherSoft)，当金字塔顶端的使用者在大部分已经购买了这类工具之后，如何再让其他的使用者也购买这些工具便成为这类厂商首要的问题。就像在大部分的多金客户和老板们已经成为 Benz 的客户之后，如何让更多的人愿意购买 Benz 汽车，便是 Benz 需要想办法的一样。Benz 的策略是推出 C-Class 级的汽车，以年轻新颖化的设计为号召，企图打入年轻的使用者群，因为以前这一族群的使用者是 Benz 无法打入的。

Rational 和 TogetherSoft 为了扩展原有的面向对象和 UML 相关的产品线，开始想要跨入开发工具的市场。由于在所有的开发者群组中，Java 开发者是最接受面向对象和 UML 技术的群组，因此 Rational 和 TogetherSoft 自然以 Java 开发工具作为优先的市场。

Rational 采用的策略是自行开发"类开发工具"，那就是 Rational XDE。虽然 Rational 很小心地处理开发工具市场，但是仍然遭遇了开发工具厂商的抵抗。例如 Microsoft 原本在开发工具中内附 Rational 的简易 UML 工具，但是在 Microsoft 逐渐为 Visio 加入更好的 UML 支持之后，就放弃了采用 Rational 的产品。当然对于 Borland 来说，Rational 进入开发工具市场也有着非常微妙的影响，因为 Borland 和 Rational 也有合作的关系。现在 Rational 为了新增产品线而进入开发工具市场，令 Borland 和 Rational 同时处于既合作又竞争的地位。

对于 TogetherSoft 而言，是否进入开发工具市场更是难以决定的事情，因为 TogetherSoft

在 UML 工具方面和 Rational 竞争得非常激烈，分居此市场的第 1 和第 2 位，在 Rational 逐渐跨入开发工具、而且 Rational 和 TogetherSoft 的客户也开始对于结合开发工具和 UML 工具有了强烈的需求之后，TogetherSoft 也必须苦拟竞争对策，否则 TogetherSoft 和 Rational 的差距不但会被拉开，既有的客户群也会遭受 Java 开发工具的侵蚀。特别是在 Borland 于 JBuilder 6 开始加入低阶的 UML 功能(例如 Refactoring)之后，很明显 JBuilder 在某些场合便已经开始和 TogetherSoft 的产品竞争了。

正由于 Java 开发工具的厂商往上仰攻企业用户群，而 Case 和 UML 厂商又想向下开拓新的潜在客户，因此开发工具厂商和这些上端的 Modeling 软件厂商从以往互不相干、到合作打天下，再到目前阶段的合作竞争状态，看来这两类软件厂商彼此竞争开战或是合并的日子已经不远了。

在 2002 年 8 月，TogetherSoft 终于从 WebGain 购买了 Visual Café，正式准备进军开发工具市场。也许 TogetherSoft 的第一步是整合 Visual Café到 TogetherSoft 的产品线中，先提供 TogetherSoft 客户的需求。接下来势必强化 Visual Café的功能，加入 UML 的能力，让 TogetherSoft 正式和 Java 开发工具竞争。TogetherSoft 不但通过 Visual Café快速地进入 Java 开发工具市场，也让 Visual Café又神奇的再次延长了生命--Visual Café真可谓是 Java 开发工具界的"九命怪猫"了。

在 TogetherSoft 正式通过 Visual Café进入了 Java 开发工具市场之后，Borland、Microsoft、Rational 和 TogetherSoft 在开发工具市场的竞争也再次隐然而动。这个熟悉的影像就和数年前 C/C++市场的竞争一样，只是换了 2 个主角而已，看来虽然开发工具市场获利并不丰富，但却是一个具有制高点作用的重要市场，因此即使许多厂商都损兵折将，但是仍然不断的有新厂商投入。前仆后继，蔚为壮观。

不管 Java 开发工具未来的竞争形势如何，从这些厂商的动作来看，整合面向对象和 UML 的高阶功能似乎是不可避免的趋势。成功的开发工具必须适当地整合 Extreme Programming、UML 和传统的 RAD 集成开发环境，以向使用者提供最大的生产力。加入 UML 的功能是传统开发工具厂商要面对的挑战，提供 RAD 和 Extreme Programming 的能力则考验了 UML 厂商是否能够提供具备亲和力的开发工具。Java 开发工具目前的两种开发和竞争方向，的确是和数年前 C/C++开发工具的竞争有所不同。究竟谁能胜出，终究要通过使用者、市场和时间来考验了。

第 8 阶段--和.NET 的巅峰之战

从 1995 年到 2002 年，应该算是 Java 的成长和全盛期。这段时间内许多的 Java 厂商、开发工具和应用程序服务器不断的在竞争和厮杀，如同神鬼战士一般进行最后的生存战。为什么？我想这是因为必须剩下最强的竞争对手，才能与 2003 年起 Microsoft.NET 逐渐产生影响力时.NET 下的开发工具竞争吧。虽然 Bill Gates 日前在媒体上公开承认.NET 的流行速度没有 Microsoft 设想的快，但是从以往 Microsoft 的竞争模式来看，Microsoft 一向擅长后来居上，不可小看其决心和实力。更何况 Java 毕竟已经开发了将近七、八年，而.NET 不过是 1 年多左右，后面竞争的日子仍然很长。

在我日常的工作中，通过接触许多不同型态的客户，也可以察觉到传统 Windows 开发工具、Java 和.NET 势力的消长。Java 从 2000 年起开始有了明显的成长，特别是大型客户、使用多种不同平台的客户群更是快速地倾向使用 Java。由于这类客户在后端大都是使用 Mainframe 或是强力的 UNIX 机器，在采用 Java 之后，也自然需要引入 Java 的组件架构以及 Web Solution，故 JSP/Servlet 便大行其道，EJB 也开始逐渐风行并且快速成长，而造就了许多 EJB 著名厂商的壮大以及另外一个竞争激烈的 EJB 市场。由于大型企业偏向 Java 的解决方案，因此造成了许多大型企业、卫星软件公司也选择使用 Java。此外 Java 也在学校和研究界获得重视，许多大学和学术机构提供了大量的 Java 人才，

逐渐地形成了极为强大的 Java 凝聚使用群。这群使用者应该算是 Java 最有力的支持了。对于中小型信息客户来说，事情就没有这么顺利了。由于这类客户的资源并不像大型客户那样丰富，因此许多中小型客户一开始在 Java 风潮下也都尝试着使用 JBuilder 来开发，但是 Java 的高门槛立刻让这些客户退出了 Java 的世界，即使留下来的也仅限于使用 Java 开发 Web 方面的应用。

由于 Java 在客户端的图形使用者接口方面失败连连，从 Applet、AWT 到 Swing，Java 似乎一直无法为客户端提供堪用的解决方案，以致不断败退，造成了目前在客户端使用 Java 应用程序的应用系统仍然非常稀少，客户端仍然是 Windows 原生开发工具的天下。问题是虽然 Java 目前在中间及后端占了极大的优势，不过应用系统仍然需要客户使用客户端来呈现应用系统的数据以及图形使用者接口，因此 Java 如果无法在客户端取得一定的应用地位，那可能也将逐渐失去中间层的优势。在 Microsoft 的.NET 从 2003 年开始逐渐影响市场之后，Java 在客户端以及 Web 的应用都将会面对新的挑战。尤其是当 Microsoft 以.NET 虚拟执行环境提供更具安全和延展性的 Web 应用(指 ASP.NET，新的 IIS 服务器以及 ASP.NET 开始支持 Apache Web Server)之后，简易好用的 ASP.NET 技术和开发工具将进一步挑战 Java 在 Web 方面的应用。目前.NET 在中间的组件技术方面仍然落后 Java 阵营一大截，是 Microsoft 需要补强的地方，否则.NET 仍然不适合做为大型企业解决方案的应用(不过 Borland 却可提供非常良好的解决方案，那就是把 CORBA 移植到.NET 上，稍后的章节会说明这些有趣的开发)。但如果 Java 持续地在客户端、Web 应用以及移动消费端节节败退，那 Java 的应用也将局限在中后端的系统应用，情形就不怎么乐观了(就像 UNIX 一样)。

根据许多知名信息机构的调查，在未来的数年当中，Java 和.NET 的竞争将趋于白热化，而且最可能出现的是由这两大技术平分天下的状态。因此 Java 目前的开发以及声势似乎都遥遥领先，但是对擅长打持久战和逆转战的 Microsoft 仍然不可大意。更何况 Java 并不是完美的金刚之身，仍然有许多不甚令人满意的地方，还是需要加把劲才能够维持好的局面。

此外应用系统型态的开发趋势也深深地影响了开发工具的走向。开发工具必须满足客户的需要，当应用系统型态改变时，开发工具厂商必须提供适当的解决方案，让使用者能够开发应用程序。根据 Gartner Group 的调查，Java 和.NET 的应用将开始快速爬升，开始侵蚀原本的 Microsoft DNA 应用市场以及传统的 COBOL、4GL 和专属系统。数年前 Java 以 Web 应用开始风行，再进而成为企业解决方案。现在.NET 也遵循相同的路径进入市场。只是.NET 除了以 Web 进攻之外，还搭配了 Web Service 和一流的开发工具。Microsoft 有样学样的招式不但使用在开发产品方面，连和 SUN 技术平台的全面对战模式使用的策略也如出一辙，只是包装的更好而已。

技术和开发工具的竞争最终取决于使用者的需求以及提供的功能、服务、架构和完成度。不过由于现在软件技术的高度竞争，因此往往一方有了特定的技术，另一方很快的也会推出相对的技术来因应。这种竞争类似于 Java 开发工具在第 3 和第 4 阶段的竞争，只是希望 SUN 能够撑得久一点。我相信一旦有一方撑不下去，一定又会站出来以冠冕堂皇的词汇说明他们只是为客户提供最好的解决方案，而不是为了和另一方竞争。如果读者看到这一天，那就代表着即将分出胜负，出来说明的一方又将使用新的名词和技术寻找出路。无论如何，从下图 Gartner Group 公布的评估数据来看，虽然.NET 现在只是第 1 个版本，但是在许多方面已经不输给开发了数年之久的 Java。看来 Microsoft 具备在短短的 1、2 年之内达到 Java 开发了数年之久的功能和成就的能力，这也许就是寡占市场的好处。Java 要和.NET 竞争，SUN 还是需要加快油门，否则很快就会被 Microsoft 追上，竞争可是不等人的。

对于 Java 开发工具来说，似乎目前加入争战的厂商还彼此杀得不亦乐乎、难分难解。往好的方面想，这是为了找出最终的强者，再与 Microsoft 的.NET 以及.NET 下的开发工具竞争；往不好的方面想，也代表这些厂商还没有察觉到.NET 的威胁。不过从我的观察以及.NET 推出之后业界的反应来看，.NET 的确已经开始吸引一些客户从 Java 转向 NET，特别是中小型客户以及需要使用 Web 应用的客户，当然一些大中型的客户也有开始动摇的情形了。

对于 Java 开发工具的厂商来说，除了要和其他 Java 开发工具厂商竞争之外，或许也要开始面对.NET 开发工具的挑战。这些厂商必须加快把 Java 开发工具塑造成更容易使用、生产力更高的工具，否则面对精于打造开发工具的 Microsoft，小的 Java 开发工具厂商生存的时间就不长了。对于 Borland 而言，这却是一个机会。现在 JBuilder 已经执业界牛耳，Borland 又决定开发.NET 下的开发工具，如此一来 Borland 由于在两方都提供最好的解决方案，因此有机会进一步扩大 Borland 的客户群。

但无论如何，当.NET 到达了一定的规模之后，.NET 开发工具和 Java 开发工具的竞争是不可避免的。即使像 Borland 这样同时提供 Java、原生 Windows 开发工具和.NET 开发工具的软件厂商，也或多或少的都面临自己人的竞争。叶孤城的"天外飞仙"对西门吹雪的"一剑西来"，你赌谁赢？嗯，也许对决的结果会创造出新的混合体--像周星驰的"少林足球"，也不一定啊。嗯，有可能，有可能。

Java 需要面对和解决的问题

Java 在开发了七、八年之后，也逐渐进入成熟期。一旦产品进入这个时期，很多的压力就会出现，再加上.NET 逐渐产生的影响力，Java 开发工具以及 Java 的应用程序服务器产品线也开始面临了许多的变化，这些变化将影响 Java 未来的开发以及和.NET 对抗的趋势。依我的观察，目前许多 Java 厂商都逐渐陷入困境和挑战之中。因为.NET 和市场的压力愈来愈大，厂商高获利的时代结束，开始进入了"微利"的阶段，这会让许多 Java 厂商开始退出 Java 市场。Java 已经不再像数年前拥有横扫市场的绝对优势，Microsoft 的.NET 也逐渐在原本 Java 主导的市场形成气候。我认为目前 Java 正面对着下面最重要的挑战和威胁：

■ 开发工具价格往下降的威胁

任何产品都是一样，当产品开始进入成熟期之后，产品价格一定会开始下降，这是正常的现象。不过对于 Java 这种高投资的技术和开发工具来说，产品价格下降代表 Java 厂商会经营得更吃力，如果无利可图，那许多 Java 厂商将会退出这个市场。正由于竞争压力太大，许多 Java 厂商都希望尽快达到一定经济规模，以备 Java 开发工具价格快速下降后能够以量大来弥补，这也是这一波割喉竞争的原因。对于 Java 开发工具是这样，对于 EJB 应用程序服务器也是这样。Java 开发工具价格持续地探底将会注定只有排名前二或是前三的厂商才能够继续存活下去，其他的小厂商只能以非常便宜或是免费的角色存在于这个市场。但是如果这种现象持续下去的话，那 Java 开发工具的进步幅度和品质都有可能开始往下滑。

■ EJB 过度竞争的压力

EJB 是 Java 的组件架构。由于会使用 EJB 解决方案的企业都属于中、大型公司，而这些公司通常都属于财力雄厚的企业，因此愿意并且有能力花大笔的经费在建制 EJB 应用系统之上。这个市场获利丰厚而且具有主导应用系统架构的力量，吸引了世界一流大厂和许多著名的厂商开发和提供 EJB 应用程序服务器，当然也不乏许多小厂商想通过这个新的组件市场而功成名就，因此为数众多的软件厂商便在这个拥挤的市场中拼得你死我活了。举凡 IBM、SUN、HP 等世界级大厂都加入竞逐的行列，这些厂商原本就是在 UNIX 工作站和大型 Mainframe 的死敌，自然不愿意让其他的竞争对手有机会独大于

重要的 EJB 市场。经过了数年的争斗之后，IBM 和 BEA 已经很明显地居于领导的地位，Borland 和 SUN 等则处于第二领先群中。IBM 通过庞大的公司资源以及硬件的交互支持成为数一数二的 EJB 厂商，BEA 则是由于最早进入 EJB 市场并且通过高知名度的 Tuxedo 掩护成功地打入企业市场。至于 SUN，虽然是 Java 技术的领导厂商，但是推出的软件产品一向令人不敢恭维。继当初的 Java Workshop 失败之后，EJB 应用程序服务器 iPlanet 说实话一点也不好用，功能和执行效率也比不上竞争对手。要不是靠 SUN 的金字招牌，iPlanet 绝对无法在 EJB 应用程序服务器市场占有一席之地。

不过 EJB 应用程序服务器市场虽然逐渐分出胜负，但是在高度竞争以及许多 EJB 应用程序服务器厂商以免费作为诉求的同时，厂商不但必须投入极大的资源研发最新、最符合 JDK 和 EJB 规范的产品，还必须浴血奋战。这从观察 EJB 应用程序服务器的授权价格不断往下降就能看得出来，由于 EJB 授权价格快速地下滑，因此许多 EJB 厂商面临了困境。许多只占有极小市场的厂商开始无利可图，进而把价格压得更低甚至采用免费方式和大厂竞争，这造成了即便是市场领导者也无法避免这个风暴的局面。由于 IBM 主要标是硬件销售，再搭配 WebSphere，因此对于 EJB 应用程序服务器价格下滑仍然不感吃力。但是对于 BEA 和 SUN 来说，却是压力巨大，因为 BEA 的收入几乎就是靠 EJB 应用程

序服务器，而 SUN 则在投入了大量的资源研发 iPlanet 之后，不但在市场占有率上无法做大，又面临价格快速下滑，当然就吃不消了。

因此，在 2002 年 8 月，SUN 的 EJB 部门副总裁公开宣示，各 EJB 应用程序服务器厂商如果

再持续进行不计血本和免费大放送的劣质竞争，那么 EJB 应用程序服务器将提早出现大幅衰退的现象。当然 EJB 应用程序服务器价格下降是有利于使用者，但是这也将让所有的 EJB 应用程序服务器进行毁灭战，EJB 大厂必须通过坑杀小厂以取得更多的市场占有率来弥补。最后可能剩下不多的选择以及品质开始下降，对于企业级的使用者，这是福是祸呢？

■ 行动和消费端的竞争

SUN 一直想让 Java 主宰所有的软件市场，从大型企业的应用系统一直到消费端的行动解决方案，因此除了在 Java 语言以及企业的 J2EE 架构之外，也非常积极地力推 JINI 和 J2ME 等技术。如果 SUN 能够让 Java 同时在中/后端企业应用系统以及移动设备市场大获全胜，那么上下交攻，Microsoft 的领域势必被严重地压缩。不过 SUN 的如意算盘似乎是出了状况，Java OS 和 JINI 一直是雷声大雨点小，SUN 数年的心血似乎也一直无法让这两个技术成气候。反观 Microsoft，虽然在桌面型应用领域独占鳌头，但是在消费端的应用一开始是处于劣势的。Win CE 开始并不见声势，又被 Palm OS 压着打。但不屈不挠似乎是 Microsoft 反败为胜的惯例，Win CE 在 2、，3 年的努力之后已经到了和 Palm OS 平起平坐的地位，Pocket PC 又逐渐受市场欢迎，Microsoft 也介入传媒和移动电话的市场，并且以 X-BOX 进入家庭游戏市场，一再显示出 Microsoft 在消费端的电子行动设备方面已经悄悄地建立起了一片江山。在.NET 以相同的技术和 Framework 允许开发者同时开发企业以及消费端和行动解决方案之后，Microsoft 结合消费端和桌面型的优势将对 SUN 形成强大的竞争压力。如果 Java 无法在消费端和电子行动市场成长得如同其在企业端的绝对优势，那 Java 在消费端的力量将遭受严厉的挑战。

■ 语言的对抗

Java 通过简洁的语言风格和虚拟机器提供的安全执行环境获得了开发者的广大回响之后，其他语言当然也不会坐以待毙，新的语言势必出现，以求对抗 Java。当初 Java 能够成功地采纳 C/C++语言的好处并且开发出新的面向对象语言，其他的新语言也可以

学习 Java 成功的地方，融合更多现代的需求以求超越 Java。C#就是一个很好的例子，C#在许多方面都学习了 Java，却又加入了 Object Pascal 的优美特性，成功地塑造了新语言，分散了 Java 群组的使用者。Gartner Group 的调查就显示了 C#将同时侵蚀 C/C++和 Java 的使用群，更不用说许多原有的语言了。此外 VB 和 Object Pascal 也会或将要推陈出新，巩固原有的使用者群，并且在新的.NET 虚拟环境中吸引从 Java 转战而来的使用者。很显然，Java 现在的处境已经慢慢地失去了独尊的地位。

当然，Java 解决方案尚有许多其他的挑战和问题，但如何面对和解决上面重要的问题，是所有 Java 厂商以及 SUN 要尽快解决的。虽然现在没有人知道未来 Java 的趋势，不过唯一可以确定的就是 Microsoft 正在一步一步地逼近之中。

JBuilder 未来的开发

JBuilder 从 2.0 开始成功地执行了一个有效的竞争模式：先从产品功能面竞争，稍后再配合市场策略一举攻上王座。虽然 JBuilder 后半段以每半年一个版本的速度杀得对手措手不及，不过快速升级的方式也造成了一些后遗症，那就是由于改版速度太快，造成许多 JBuilder 客户以跳版本的方式来升级。例如 JBuilder 4 的客户更新到 JBuilder 6，JBuilder 5 的客户则升级到 JBuilder 7。

当然 JBuilder 快速升级方式和 Java 开发工具的竞争有关，但是很大一部分原因也是因为 Java 的 JDK 经常更新，迫使 Java 开发工具必须跟上 Java　JDK 的脚步，否则就有被淘汰出局的危险。因此为了兼顾 Java 快速开发和使用者的权益，JBuilder 在快速开发之后的下一个阶段也许应该考虑推出 1 年套餐版，让使用者在付费之后的 1 年内都可以免费升级吧。

Borland 的 JBuilder 还在快速的开发之中，精彩的故事也一定会持续发生，也许 1、2 年之后，再让我们回首，又将看到许多可歌可泣的故事，这就留待日后的书籍来叙述吧。当读者看到本书时，JBuilder 8 可能已经在市面上销售，而 JBuilder 9 可能也在开发的阶段，这就是 Java 充满活力、进步快速的特质。喜欢 Java 吗？那么就接受这个高度动感的世界吧！

^v^v^v^v^v^v^v^v^v^v

第六章　失去的王冠--Borland 数据库工具的战役

在 Borland 的产品线中，有两个产品是较少受到瞩目的，那就是 Borland 从 Ashton-Tate 并购来的 dBase 系列以及昙花一现的 IntraBuilder。对于 Borland 来说，dBase 和 IntraBuilder 是非常可惜的牺牲品。dBase 发展的黄金时机被 Philippe Kahn 白白浪费，IntraBuilder 带来的无限潜力硬生生地被 Delbert Yocam 糟蹋掉。Borland 最有机会的两个关键时刻分别被两任 CEO 蹉跎，不知到底是时也？命也？运也？

dBase 和 IntraBuilder 这两个产品，到底是如何在 Borland 中发展的呢？为什么最后 dBase 和 IntraBuilder 都会进入死胡同？让我们一起来探索其中的秘密。

IntraBuilder 的诞生

谁说"洞悉先机"一定是好事？当初哥白尼在几个世纪之前大胆地提出了天体运行论，同势力庞大的基督教对抗，因而被基督教视为邪说，一直到 3 个世纪之后才被罗马教皇承认。而哥白尼的一生都在承受着巨大的压力，Borland 的 IntraBuilder 几乎也面临了同样的命运。

1995 年，当浏览器的应用逐渐成为主宰力量之后，各种 Web 的应用也开始快速地发展起来。一开始 Web 应用是以面向文件为主，许多 Web 应用都使用纯文本编辑器来开发 HTML 网页。但是人们很快发现，这种方式非常不经济，因为 Web 应用虽然有很大一部分属于美工的需求，但是当 Web 进入人们的生活后，许多 Web 应用便开始需要结合数据处理而转向商业的应用。因此，很快 Web 的解决方案便开始从静态网页的应用进入到使用

程序来解决的阶段。但是，当时正值浏览器大战的阶段，Netscape 正和 Microsoft 的 Internet Explorer 拼斗得你死我活，而 Java 也开始兴起。此时浏览器并没有标准，连带着对 HTML、JavaScript 的支持也混乱无比。因此，虽然许多程序员都感觉需要一个 Web 开发工具帮助他们开发逐渐炙手可热的 Web 应用程序，信息业界也开始有强烈的需求，但是混乱的 Web 标准却让许多程序员不知所措。

不过，Borland 的 Visual dBase 小组却从中看到了极大的契机，因为在为 Visual dBase 未来的版本加入支持 Web 的功能时，Visual dBase 小组突然发现，既然 Web 的功能是许多程序员想要的，那么，为什么不直接提供一个可视化的 Web 开发工具，让需要开发 Web 应用程序的程序员能够拥有最好的工具，而不需要痛苦地使用纯文本编辑器来开发 Web 应用程序呢？

时值 1995 年，这的确是一个令人相当震撼的想法，因为它体现了未来需求的趋势。当 Visual dBase 小组提出这个想法之后，立刻在部门内获得了极大的回响。几经商议，Visual dBase 小组决定先开发一个可视化的 Web 开发工具来测试市场，而且他们决定就使用 Visual dBase 来开发这个新的产品。这实在是个大胆又令人惊讶的决定，因为当时不但没有类似的产品，而且决定使用 Visual dBase 而非 C/C++来开发新产品，更是不可思议，开发工具真的可以使用 Visual dBase 来开发吗？

当 Visual dBase 小组决定开发这个新的开发工具时，却面临了一些技术上的抉择，那就是使用什么语言作为这个新开发工具的核心？另外，该产品既然是一个 Web 开发工具，当然需要一个 Web Server 作为后端的驱动引擎。但是，当时的市场上只有 Netscape 和 O'Reilly 等少数厂商拥有 Web Server 引擎。因此，Borland 必须决定使用什么 Web Server。不过，这些问题很快就有了答案。

由于 Java 快速地兴起，Applet 也成为学习 Java 的入门知识，因此，JavaScript 很快就被众人视为开发 Web 应用程序的标准语言。于是 Visual dBase 小组决定使用 JavaScript 作为这个开发工具的核心语言，并且强化当时的 JavaScript 语言，以支持这个新的开发工具。另外，由于当时的 Web Server 大都不便宜，因此，Visual dBase 小组决定自行开发一个 Web Server 作为这个开发工具的内建 Web Server。最后，Visual dBase 小组定义这个开发工具必须拥有下面的功能：

- 可视化开发环境，允许程序员使用组件和拖曳的功能来设计 Web 应用程序
- 使用 JavaScript 作为核心语言
- 提供内建的 Web Server
- 结合 BDE/IDAPI 来连接各种数据库

这个开发工具便是 IntraBuilder--后来震撼一时的数据库 Web 开发工具先驱。

在 IntraBuilder 开始开发之后，Visual dBase 小组很快发现，虽然他们可以使用 Visual dBase 完成大部分的工作，但是，终究有一些功能是 Visual dBase 力所不逮的地方，因此，在 IntraBuilder 开发的后期，为了让它能够支持当时 Microsoft 刚推出的、同 Applet 相抗衡的 ActiveX 以及动态执行 Applet，Visual dBase 小组还是使用了部分的 C 程序代码来完成这些功能。

IntraBuilder 的震撼

1996 年 9 月，经过 1 年多的开发，IntraBuilder 终于推出在世人的面前。IntraBuilder 推出之后，全世界的专业媒体几乎都对 IntraBuilder 好评有加，而且都不能相信 Borland 能够如此快速且先知地推出数据库的 Web 开发工具。

全世界的好评如潮，因此，在 IntraBuilder 准备正式出货之前，Borland 也是信心满满。我记得，当时在拿到 IntraBuilder 的 Beta 版后，虽然我对于 Web 的开发仍然没有太多的经验，但是很快就了解了这个产品的潜力，因为 IntraBuilder 和当时其他的 Web

开发工具以及编辑器比较起来，简直是领先了数个世代之久，而且还能够用来作为学习 JavaScript 的工具和开发连接数据库的 Web 应用程序。这些功能在市面上几乎没有任何的竞争对手可以比拟。即便以今日的标准来看，IntraBuilder 提供的 Web 可视化设计能力仍属一流。因此，当时我就觉得这会是一个大卖的产品。

IntraBuilder 面对的困难

即便 Borland 非常有信心，专业媒体也一片看好，但是没有想到的是，在 IntraBuilder 推出之后，只带来了第一波销售热潮，随后的销售却很快冷却下来，造成了 IntraBuilder 叫好不叫座的情形。这实在是一件很奇怪的事情，因为 IntraBuilder 产品本身没有太大的问题，产品的方向也是正确的。但是为什么 IntraBuilder 在市场上就是无法拥有亮丽的表现呢？这个问题是 Borland 急于寻找答案的。记得当时在台湾发表 IntraBuilder 时，似乎也是回响热烈，但实际出席的人却不多。台湾 Borland 的产品经理还在会场询问我，为什么出席的反应这么不热烈，产品本身不是不错吗？

在 IntraBuilder 首次遭遇挫折之后，Borland 很快便找出了其中的重要问题所在。有些属于产品本身的小瑕疵，有些则是当时整个环境的问题。总结当时 IntraBuilder 1.0 失败的原因有：

- IntraBuilder 太过于先进，许多程序员不知如何使用
- IntraBuilder 不支持中文
- 浏览器对于 JavaScript 语言的支持程度混乱
- IntraBuilder 在 GUI 方面的 Render 尚有瑕疵

由于当时 Web 的程序应用还属于萌芽期，Internet/Intranet 程序应用仍然处于第一波面向文件的阶段，大多数的 Web 应用是使用 HTML 和一般编辑器来制作的。这个时期距离第二波程序员开始大量使用各种不同的 Web 语言来开发 Web 应用程序仍然有 1、2 年的时间差。

可惜的是，IntraBuilder 就是太早的察觉了这个趋势，因此当 IntraBuilder 推出之时，仍然是领先第二波 Web 应用的发展。从下面的图形，我们也可以看到 IntraBuilder 推出的时机的确是先于数年后其他 Web 开发工具的脚步。

正是由于 IntraBuilder 推出的时机太早，因此只能吸引站在前端的开发人员，大多数的开发人员对于这样革命性的产品，浑然不知其重要性，造成 IntraBuilder 一开始只能销售给 Borland 的少数客户以及其他领域顶尖者的结果。不过我认为这是一件好事，因为 IntraBuilder 先期的销售数量虽然没有达到 Borland 的预望，不过 IntraBuilder 一开始便攻入了最重要的客户群，占据了金字塔顶端客户的 mind-share。只要 IntraBuilder 能够再接再厉，等到 1、2 年后，当大多数的开发人员了解了 Web 开发工具的重要性以及实用性之后，IntraBuilder 将可快速收割成果。此外 IntraBuilder 的理念与技术领先于其他竞争对手数年之久，即使其他 Web 开发工具推出，IntraBuilder 也能够以逸待劳，痛击竞争对手。

在 Borland 分析了 IntraBuilder 遭遇挫折的因素后，很快便展开了相应的行动，因为 Visual dBase 小组对于 IntraBuilder 仍然非常有信心。在支持 DBCS 方面，由于 IntraBuilder 1.0 不支持 DBCS，因此造成了在许多亚洲国家和地区，包括中国台湾地区、日本和韩国以及中东无法销售的问题。这个影响当然是很大的，因为光是一个日本市场，几乎就可以销售数千万套。

另外一个扰人的问题，就是由 IntraBuilder 开发出来的 Web 应用程序在不同的浏览器中会发生网页内容和位置与在 IntraBuilder 中设计时不一致的情形。这个问题形成的原因很复杂，大都和当时不同的浏览器在 render 网页内容时的差异有关。当然，当时尚未有一致的标准，使得不同的浏览器支持的 HTML 版本和 JavaScript 版本不同。不过，

虽然这些问题不全是 Borland 的错误，但是，就如同当时一个 IntraBuilder 使用者在 Forum 中留下的一句话"It may not be Borland's error，but it definitely is a Borland's problem(不是 Borland 的错误，却是 Borland 的问题)"。

为了解决 IntraBuilder 面对的问题，Visual dBase 小组很快便开始进行了 IntraBuilder 第二个版本的开发工作，目的就是为了解决 IntraBuilder 客户所抱怨的问题，并且强化 IntraBuilder 在扩展性和执行效率方面的功能，以期让更多的客户愿意使用 IntraBuilder。1997 年 6 月，Visual dBase 小组手脚很快地推出了 IntraBuilder 1.5，进行第二次的出击。

IntraBuilder 1.5 几乎是一个成熟的 Web 开发工具，因为 IntraBuilder 1.5 可以支持 DBCS，并且大幅提高了 IntraBuilder 应用程序的执行效率。此外，Visual dBase 小组也特别使用 C 改写了 IntraBuilder 在 render 网页的功能，让 IntraBuilder 能够更精确地呈现 Web 网页的内容，并且大幅提升了在不同浏览器中的兼容性。

经过了这么多的改善之后，IntraBuilder 在全世界的销售果然有了起色，慢慢地向 Borland 为 IntraBuilder 设定的目标接近。Visual dBase 小组当然也是很高兴，因为这证明了他们的眼光是正确的。因此，Visual dBase 小组在 IntraBuilder 站稳了脚步之后，便开始进行 IntraBuilder 2.0 大改版的工作，希望通过 2.0 版本让 IntraBuilder 成为最成功的 Web 开发工具。

再接再厉，IntraBuilder 2.0 的开发

1997 年，Borland 已经准备好了新版的 IntraBuilder，并且在当年的 Borland Conference 中公开宣示了 IntraBuilder 2.0，也为未来的 IntraBuilder 3.0 提供了发展蓝图。新版本的 IntraBuilder 一切看起来是非常的顺利，而且 Visual dBase/IntraBuilder 小组也信心满满，准备为 IntraBuilder 再下一城。

当时的 Borland 正和最具影响力的 Netscape 以及 Microsoft 共同制定 JavaScript 的标准，并且准备提交到 ECMA。其时 IntraBuilder Architect Randy Solton 正忙于和 Netscape 以及 Microsoft 的人员定义 JavaScript 的最终标准，希望两大浏览器 Communicator 和 Explorer 能够在未来支持这个新的标准，以便让 IntraBuilder 的应用程序能够正确无误地执行在这两个浏览器中。不过，由于 Netscape 和 Microsoft 正处于最激烈的战火中，彼此各怀鬼胎、谁也不服谁，因此标准制定的流程进行得非常缓慢、不顺利，这也间接造成了开发 IntraBuilder 的难度。

在 Borland Conference 1997 中，当时 IntraBuilder 的 Director Michael Gardner 展示了 IntraBuilder 2.0 的新功能。

在 IntraBuilder 2.0 中，Borland 提供了一个内建的 HTML 可视化编辑器，以提供更为精确的网页编写功能(类似今日 Macromedia 提供的工具)。IntraBuilder 2.0 的 ActiveX 具有同时在客户端和伺服端执行的能力。这个功能非常 Cool，因为在当时，ActiveX 大都只能执行在客户端，而 IntraBuilder 2.0 却能够让 ActiveX 同时执行在客户端和伺服端，这可就稀奇了。另外，IntraBuilder 2.0 对于 JavaBean 的支持也将和 ActiveX 一样完全，这代表两种不同的组件技术在 IntraBuilder 中将会是相同的 First-Class 组件。这可是 Macromedia 在数年之后才能在 UltraDev 中实现的技术。

另外一个 IntraBuilder 2.0 最重要的功能就是提供了跨平台的能力。Borland 准备同时开发 Windows 和 UNIX 平台的 IntraBuilder，计划支持的 UNIX 平台包含了 Solaris、HP-UX、AIX 和 IRIX。这在当时可算是非常大的手笔，因为当时市场上不但没有类似的产品，更遑论是提供跨平台的 Web 开发工具。因此，我认为当时如果 Borland 能坚持下去，就将拥有绝佳的市场契机。

在 1997 年的 Borland Conference 中，除了 Michael Gardner 的讲座之外，IntraBuilder

的 Architect Randy Solton 也在 Borland Conference 主讲了两个讲座，深入地讨论了 IntraBuilder 2。0 的新功能和实现技术。
此外，当时 IntraBuilder 的产品经理 Klaus Krull(K.K.)也在现场同台演出，并且声明 IntraBuilder 2.0 的 Beta 版将提供给有兴趣的开发者测试。从所有的迹象来看，IntraBuilder 2.0 已经是蓄势待发了。
另外，当时 IntraBuilder 的 QA 工作，是由华人出身的 Ken Chan 所领军。其实从 Borland C/C++ 3.0 开始，华人在 Borland 的 R&D 以及 QA 部门中一直占有一定的比例，对于 Borland 产品开发有着不小的贡献。
不过，事情的发展很快就出乎所有人的意料，在 Borland Conference 1997 年举行过后不久，Borland 突然放弃了 IntraBuilder。这个消息传来，对于当时急切等待 IntraBuilder 2。0 的我来说，实在是晴天霹雳。为什么 Borland 会突然放弃 IntraBuilder，这是当时我一直想要了解的问题。我曾经询问过台湾 Borland 的好友，但是他们也不知道实际的原因。后来我曾经听到几种说法：其一是 Delbert Yocam 对于 IntraBuilder 没有兴趣，因此不愿意再投入资源开发下去；另外的传言则是 Borland 决定全力开发 JBuilder，因此把 IntraBuilder 的资源移到 JBuilder 去；还有的说法是 IntraBuilder 开发团队和 Delbert 处不来，因此集体离开 Borland。不过事情的实际答案仍然是一个谜，即使到了今日，我再次为 Borland 工作时，仍然无法获得确定的答案，实在令人遗憾。
我认为 IntraBuilder 是最为可惜的产品之一，因为早在 1996 年，当其他软件公司尚未察觉到 Web 需要一个良好的、能够和数据库整合在一起的开发工具时，Borland 居然就已掌握到软件时脉，而且推出了实际的产品，可说是一片大好，也是 Borland 少有能够走在别人前面的好时机。如果当时 Borland 好好地持续开发 IntraBuilder，我相信 IntraBuilder 一定会成为比今日 Macromedia 的 UltraDev 还好的产品，而且也将是我认为属于"消费型软件"的产品，Borland 将可在数年之后的公元 2000 年大展鸿图。只可惜 Delbert Yocam 似乎是脑筋坏了，不然就是没有眼光，居然在 IntraBuilder 2.0 几乎已经完成之前决定放弃。不但让 Borland 失去了在 Web 开发工具方面占有一席之地的机会，也失去了数年后让全世界疯狂的 Internet/Intranet 和 DotCOM 的黄金发展阶段，真是令我扼腕。甚至在 Delphi 3/4 时，我强烈建议在 Delphi 中开发类似的 IntraBuilder 功能的心愿也无法达成。我想，这应该是 Borland 在并购 Ashton-Tate 之后，另外的一个重大失策。
令人遗憾的结局
在 Delbert Yocam 决定放弃 IntraBuilder 之后，这个举动也几乎成为压垮骆驼的最后一根稻草，因为这对 Visual dBase 小组实在是一个非常大的打击。Visual dBase 小组已经看到 Visual dBase 的市场不断地下滑和萎缩，因此急需一个新的产品以增加收入并开拓未来的产品线。不过，在 Delbert 决定了 IntraBuilder 的命运之后，也代表了宣布 Visual dBase 小组终将结束的未来。正是由于 Delbert 的决定，引发了 1、2 年后 Visual dBase 小组所有人都急于跳下 Visual dBase 这个曾经一时的旗舰，转而纷纷希望加入 Java 这艘新的战舰，从而引发了稍后 Borland 内部的极大争议。
直到现在，我仍然非常喜欢和怀念 IntraBuilder。因为我在 CDC 服务时，便曾和一位同事共同把 IntraBuilder 引入 CDC 作为开发 Web 解决方案的开发工具。由于 CDC 使用 Delphi 作为主力开发工具，而 IntraBuilder 的开发模式又和 Delphi 很类似，因此对于 IntraBuilder 的接受程度很高。IntraBuilder 1.5 解决了中文问题和执行效率问题，当时我开发的 Pilot 系统可以执行得非常顺利，因此我决定在 Web 方面的工具使用 IntraBuilder。没有想到，后来 Borland 居然放弃 IntraBuilder，顿时之间所有的心

血都化为流水。身为 Borland 产品使用者的我，不能够接受 Borland 这种处理产品的方式，更何况 Borland 在当时也没有提供任何可取代 IntraBuilder 的产品。Borland 处理 IntraBuilder 的方式引起了当时许多 IntraBuilder 使用者的反弹，也让 Borland 几乎无法再涉足 Web 开发工具的市场。

命运坎坷的 dBase

回顾 dBase 产品的一生，实在令人不知说什么好。dBase 曾经主导了 PC 数据库技术的发展主流，在早期也几乎霸占了 PC 数据库市场。十几年前，当人们发现一台 PC 在执行了 dBase 之后，居然能够处理许多日常数据，立刻便为 dBase 不可思议的能力而着迷，进而创造了 dBase 不可一世的时代。

1980 年 8 月，George Tare 和 Hal Lashlee 两位先生创建了 Software Plus 软件公司。稍后，他们和 Microsoft 一样，从一个小软件公司购买了 Vulcan Data Base 软件，并且根据 Vulcan Data Base 开发出 dBase 产品的前身。很快，George Tare 和 Hal Lashlee 合作的软件便获得了许多使用者的好评，他们的软件逐渐在市场上受欢迎。不久之后，George Tate 和 Hal Lashlee 便成立了 Ashton Tate 公司，展开了 dBase 神话的时代。

在 Ashton-Tate 推出 dBase II 之后，正值 PC 开始快速成长的时期。由于当时的 dBase II 在 PC 上提供了合理的数据处理能力，因此很快便有了大量的使用者，dBase II 的影响力也开始渗入商业使用者领域，而 Ashton-Tate 这个招牌也逐渐成为广为人知的公司。

1984 年是 Ashton-Tate 一生最为重要的一年，因为在这年的 6 月，Ashton-Tate 推出了 dBase III。dBase III 在数据处理能力、运算速度方面都比 dBase II 大幅提升，正好符合当时 PC 愈来愈大量数据应用的需求。在 Ashton-Tate 推出 dBase III 之后，立刻在全球大卖，随后推出的 dBase III Plus 更是奠定了 Ashton Tare 在 PC 数据库方面至尊的地位。dBase III 和 dBase III Plus 的空前成功，使得 Ashton-Tate 营收大增，并且成为当时全球第 3 大的软件公司。和 Microsoft、Lotus 分别以 DOS、Lotus 1-2-3 和 dBase 各在操作系统、Spreadsheet 以及数据库市场鼎足而立。

当 Ashton-Tate 在数据库市场不可一世之时，Oracle 还是一个名不见经传的小公司。怎知 10 年风水轮流转，现在的 Oracle 已经成为数据库的霸主。为什么 Ashton Tate 这个曾经执 PC 数据库牛耳的公司后来会一蹶不振呢？这都要从 Ashton-Tate 的 dBase IV 说起。

急转直下的 dBase IV

当 Ashton-Tate 的 dBase III/Plus 成功之后，Ashton-Tate 的野心就更大了，急于和 Microsoft/Lotus 逐鹿天下。Ashton-Tate 决定投入大量的资源开发下一代的 dBase 软件，把处理数据的能力再次提高，并且提供更为复杂的功能。虽然 Ashton-Tate 的想法很好，要把 PC 数据库的竞争再次升高，提供更为高阶的应用，但却忽略了当时 PC 硬件设备是否能够跟上 Ashton-Tate 设定的标准的问题。

在 Ashton-Tate 开发 dBase IV 到中期之后，却发现当时 PC 的设备无法顺利执行 dBase IV，此时 Ashton-Tate 才发现事态严重。其实，在 Ashton-Tate 开发 dBase IV 之前，并没有评估硬件需求或是没有控制 dBase IV 的开发。无论如何，对于 Ashton-Tate 来说，dBase 几乎是唯一的软件，也是成功的支柱。结果，对于最重要的产品居然管理成这个样子，从这个迹象便可知当时的 Ashton-Tate 可能被 dBase III/Plus 的胜利冲昏了头，也开始夜郎自大起来。

Ashton-Tate 发觉了 dBase IV 的问题之后，立刻和一些软/硬件厂商合作，为当时只能使用 640K 内存的 PC 加入新的内存设备，以便执行需要海量存储器的 dBase IV。虽然后来的确弄出了一个解决方案，但却不为市场大众接受。dBase IV 在 Ashton-Tate 无法

解决内存需求问题之下仍然执意推出，结果市场一片负面评价。不但一般的 PC 内存不够无法顺利执行，再加上 dBase IV 的臭虫极多，立刻被市场所唾弃。原来的 dBase 使用者仍然继续使用 dBase III/Plus，不愿意升级到 dBase IV，这让 Ashton-Tate 面临血本无归的窘况。再加上 dBase 又得面对 FoxBase 和 Borland Paradox 愈来愈强劲的竞争，Ashton-Tate 无法推出让 dBase 使用者满意的下一代产品，只能眼看着市场不断流失。

1990 年初，Ashton-Tate 内部起了内讧，导致 dBase 的主要 Architect 以及许多工程师离开 Ashton-Tate，而 Ashton-Tate 在数年之间仍然无法解决 dBase IV 的问题也让人不可思议。此时，Ashton-Tate 在 Paradox 和 FoxBase 的鲸吞蚕食之下，几乎快变成一个空壳了。公司营收不断下滑，dBase 市场占有率不断下降，人员更是快速流失。到了 1991 年，Ashton-Tate 几乎已经撑不下去了。

在 Ashton-Tate 快速走下坡之际，却是 Borland 即将到达巅峰之时。1990 年，Philippe Kahn 从 Borland 本身的 Paradox 成长情形知道了 dBase 的状况，已经逐渐看出 Ashton-Tate 的颓势。于是 Philippe Kahn 知道，Borland 称霸 PC 数据库市场的时机即将到来，而且 Borland 必须比 Microsoft 先出手才能够赢得这个不容错过的好机会。

1991 年，在 Ashton-Tate 已经摇摇欲坠之际，Philippe Kahn 终于决定出手。因为 Philippe Kahn 知道，绝不能让 Microsoft 先叫牌。更绝的是，Philippe Kahn 一出手就是雷霆一击，条件好得让 Ashton-Tate 根本无法拒绝。1991 年 7 月，当 Philippe Kahn 以 440 Million 叫牌之后，Ashton-Tate 很快就全面投降，决定从此嫁入 Borland。

Ashton-Tate 的被并购和走入历史

1991 年，不可一世的 Philippe Kahn 以极大的霸气和本钱并购了当时已经快速走下坡的 Ashton-Tate。Philippe Kahn 购买 Ashton-Tate 的真正目的是为了与 Bill Gates 一争长短，以图成为 PC 软件界的霸主。另外一个目的则是为了彻底消灭 dBase，因为 Philippe Kahn 认为 Borland 自己的 Paradox 比当时问题多多的 dBase IV 好得太多了，一旦 dBase 退出市场，那么 Paradox 即可席卷 PC 数据库市场，让 Borland 同时成为 PC 开发工具以及 PC 数据库市场中的老大，进而同主掌 PC 操作系统的 Microsoft 和控制 PC 计算软件的 Lotus 分庭抗礼。

在 Borland 以不可思议的高价购买了 Ashton-Tate 之后，立刻引来了华尔街的批评，因为华尔街的分析师都认为 Philippe Kahn 出的价格太高，Ashton-Tate 在当时并不值 440 Million 多美金。不过对于 Philippe Kahn 来说，用 4 亿多美金来与 Microsoft/Lotus 一较长短的大企图相比，实在不算什么，因为 Borland 有的是钱。

在 1991 年 Borland 并购 Ashton-Tate 时，Borland 的市值其实比 Ashton-Tate 小。当时的 Ashton-Tate 是排名第 5 的软件公司，而 Borland 只排名到第 9。但是 Ashton-Tate 已经在走下坡，手头拮据。反观 Borland 则是蒸蒸日上，现金多多。Borland 并购 Ashton-Tate 一事，媒体都以"小鱼吃大鱼"为标题进行报道。当时 Ashton-Tate 的营业额大概是 250 多个 Million 美金，而 Borland 的营业额则大约是 230 多个 Million。因此，在 Borland 加上 Ashton-Tate 之后，立刻成为一个年营业额将近 500 Million 的软件公司，排名跃为当时全球第 3 大的软件公司，仅次于 Microsoft 和 Lotus。读者可以想想，现在的 Borland 年营业额才 200 多 Million，但是 10 年前却已经拥有 500 Million，可见当时 Borland 的盛世和规模之大。在 Borland 完成了 Ashton-Tate 的并购之后，Philippe Kahn 每日都得意洋洋，因为在 Philippe Kahn 看来，成为全世界第一大、击败 Microsoft 的日子已经不远了。

Ashton-Tate 被 Borland 并购之后，其光辉灿烂将近 10 年的时光也随之消逝。Ashton-Tate 原本很有机会成为今日的 Oracle，继续占据 PC 数据库市场龙头的地位，没有想到

Ashton-Tate 却把好好的一盘棋下到了死局，硬生生地把自己的命脉产品玩完。虽然 Ashton-Tate 是一个很好的负面教材，但人类似乎永远学不会历史，数年后的 Informix 也走上了和 Ashton-Tate 极为类似的道路。

不甘之作，dBase For Windows 5.0

在 Philippe Kahn 得意不久之后，Microsoft 也并购了 FoxBase 这家公司，并且快速地推出了 FoxPro 这套可以在 Windows 下执行的、与 dBase 兼容的软件。由于 Microsoft 掌握了原本 DOS 下 dBase 程序员急需一个 Windows 下的 dBase 开发工具的心态，因此当 FoxPro For Windows 推出之后，立刻吸引了许多原先 dBase III/dBase III Plus 的使用者。

虽然 Borland 在 Microsoft 推出 FoxPro For Windows 之后开始流失使用者，但是，由于其时 Paradox For DOS 的销售仍然良好，因此，Philippe Kahn 并没有放在心上，仍然认为最终 Paradox For Windows 可以击败 FoxPro For Windows。在这里，Philippe Kahn 显然犯了轻敌的错误。

在 Microsoft 连续推出两个版本的 FoxPro For Windows 之后，Borland 终于察觉原先 dBase 的使用者正处于快速地流失之中。虽然 Borland 已经推出了 Paradox For Windows，而且销售也在预期之中，但是很显然，Paradox For Windows 并不能阻止 dBase 客户的流失。Philippe Kahn 此时才开始着急。此外，Borland 也面临还在使用 dBase For DOS 使用者强大的压力，他们要求 Borland 推出 dBase For Windows。

其实，dBase For Windows 产品本身还是不错的，不过由于已经太晚加入 Windows 平台数据库战场，而且是在匆促上阵的情形下，本身的臭虫当然不少，再加上得面对轻装上阵的 FoxPro For Windows，dBase For Windows 几乎没有什么胜算。随后的结果果然如同许多人预期的一样，dBase For Windows 在推出之后不但无法憾动 FoxPro For Windows 的江山，反而引来原本期待的 dBase 使用者的绝望。dBase 的使用者在苦等 Windows 版的 dBase 数年之后，Borland 仍然无法提供一个高品质的产品。顿时之间，大量不满的 dBase 使用者都转向了 Microsoft 的 FoxPro For Windows，也造成了 dBase For Windows 不可挽回的败势。

在 dBase For Windows 失利之后，许多人都开始把矛头对向 Philippe Kahn，认为是 Philippe Kahn 的自大和轻敌搞死了 dBase For Windows 这条原本有机会的产品线。如果 Philippe Kahn 能够在并购 Ashton-Tate 之后好好地开发 dBase For Windows，并且在 Microsoft 的 FoxPro For Windows 之前推出，那么 Borland 将可让大部分 dBase For DOS 使用者转入 Windows 的市场。唉，如果时光能够倒转，如果 Borland 能够早一步推出 dBase For Windows，再进而开发出后来的关系数据库(Relational Database)产品，那么，Borland 现在可能仍然是前 3 大的软件公司。

最后的帝王--Visual dBase 7

很显然，Microsoft 以极小的代价购买了 FoxBase，并且用 FoxPro For Windows 抢走了 Philippe Kahn 花大钱购买来的 dBase 使用者，的确是等于狠狠地打了 Philippe Kahn 一巴掌，让 Philippe Kahn 知道，先出手并不代表会赢得最后的胜利。

这对于日日夜夜想打败 Microsoft 的 Philippe Kahn 来说，当然是无法忍受的耻辱，因此，Philippe Kahn 念念不忘的就是如何扳回一城。在 dBase For Windows 5.0 失利之后，Borland 决定再次重新出发，准备推出新版本的 dBase For Windows，来挑战 FoxPro For Windows。不过，市场情势的发展却出现了变化，PC 数据库市场已经开始走入关系数据库的时代，桌面型数据库的市场已经开始出现下滑的现象。

1997 年 12 月，Borland 推出最后一版的 dBase For Windows 7.0 来角逐市场。dBase For Windows 7.0 的品质和功能才是 Borland 早该在几年前推出的产品，如果 Borland 早几年推出 dBase For Windows 7.0，那么 Windows 下 dBase 的市场绝对会由 Borland 寡占，

FoxPro For Windows 将不是对手。只可惜时不我待，在 dBase For Windows 7.0 推出之际，Windows 下 dBase 的市场已经大势已定。虽然 dBase For Windows 7.0 的确是一个好产品，但是它再也无力改变市场了。此外，此时 PC 桌面型数据库的市场也逐渐萎缩，Microsoft 也准备走向关系数据库市场，Windows 下 dBase 的市场对于 Microsoft 来说，已经不那么重要了。

在 dBase For Windows 7.0 推出之后，Borland 事实上也察觉到了 PC 数据库市场的变化，准备以 InterBase 进入关系数据库的市场。至此，延续数年之久的 PC 桌面型数据库的战火也终于近乎停止状态了。

当 Borland 的 Visual dBase 小组发现整个数据库市场的变化之后，内部产生了相当大的矛盾，许多 Visual dBase 的工程师在不看好 dBase 产品的情形下纷纷决定转换跑道。"弃船"也许是当时最适当的形容词，几乎所有的 Visual dBase 的工程师都希望进入 Borland Java 开发小组，没有人愿意继续留在 Visual dBase 小组。因此，当时 Visual dBase 小组在 Borland 内部引起了不小的骚动。每一个人都想到 Java 小组，许多 Borland C/C++小组的工程师也都希望进入 Java 开发小组，但是 Java 小组并不能容纳这么多人。最后许多无法进入 Java 小组的工程师不是离开 Borland，就是随着 Borland 卖出 dBase 时跟随 Visual dBase 到了新的公司。

最后的晚餐

1998 年的 Borland Conference 应该是 Visual dBase 在 Borland 最后一次的盛会了。当时使用 Visual dBase 的使用者已经不多，因此在 BorCon 1998 年中 Visual dBase 的讲座也不多。不过对于一些 dBase 的忠诚使用者来说，Visual dBase 7.0 仍然是他们的最爱。因此在 BorCon 1998 年，当时在 Visual dBase 界中最著名的支持者 Alan Katz 负责了许多 Visual dBase 的讲座，也号召了许多 dBase 的使用者参加这次的 Borland Conference。

Alan Katz 的努力显然是希望 Borland 不要放弃 Visual dBase，能够继续开发 dBase 的产品。不过，这些努力仍然无法挽回市场的形势以及 Borland 的决心，BorCon 1998 也终于成为 Visual dBase 最后一次的舞台。

生命的延续--dBase 2000

1998 下半年，Borland 终于决定把 Visual dBase 卖掉，因为 Borland 已经不想在 Desktop 的数据库市场竞争了。在 Borland 决定卖掉 Visual dBase 的信息传出之后，立刻引起了许多 dBase 使用者的强烈反应。他们认为 Borland 不负责任，因为如果 Borland 随便把 Visual dBase 卖掉，那么 Visual dBase 便注定会从此消失。由于当时 dBase 使用者的压力太大，因此 Borland 不得不小心翼翼地处理这个烫手山芋。当时的 Borland 已经是放也不是，不放也不行了。

为了缓和 dBase 使用者的强烈不满，Borland 宣布会谨慎地选择购买 dBase 的公司，而不会随便把 dBase 卖出去。这个时候，Alan Katz 也知道了 Borland 的决定。出于对 Visual dBase 的热爱，Alan Katz 决定找寻资金来源把 dBase 的版权从 Borland 的手中买下来。很快，他找到了一些 dBase 的爱好者，每人拿出一定的资金来集资购买 Visual dBase 的版权。在 Alan Katz 取得了资金之后，便立刻和 Borland 联络，准备和 Borland 谈判。当然，在 Alan Katz 集资的过程中，Borland 也试着寻找对 Visual dBase 有兴趣的公司或是个人，不过这个过程并不顺利。

因此当 Alan Katz 和 Borland 接触之后，双方立刻有了交集，双方都有很高的意愿。不过，在深入的谈判之后，Borland 很快发现 Alan Katz 的资金和 Borland 想要求的版权费有很大的距离。原本 Borland 不太想再谈下去，不过，在 Alan Katz 和 Borland 接触的消息传出之后，却获得了许多 dBase 使用者的欢迎。Alan Katz 在 dBase 界的高知名

度以及多年来对于 dBase 的贡献都让 dBase 的使用者觉得，如果由 Alan Katz 收购 dBase，那么 dBase 仍然将有美好的未来。于是 dBase 的使用者开始向 Borland 施压，希望 Borland 能够成功地和 Alan Katz 谈好条件。

虽然 Borland 不太接受 Alan Katz 的条件，但是在遍寻不到适合的买主、再加上 dBase 使用者的强烈要求和 Borland 急于解决 Visual dBase 的情况下，Borland 终于在半买半送的条件下把 dBase 所有的原始程序以及版权卖给了 Alan Katz。在 Borland 决定出脱 dBase 给 Alan Katz 之后，Alan Katz 便立刻和朋友成立下 KSoft 软件公司，准备延续 Visual dBase 的产品生命。

1999 年 3 月 12 日，Borland 终于廉价售出了 1991 年花费数亿美金并购来的 dBase 产品，dBase 在 Borland 不受重视的日子也终于结束。Alan Katz 在购买了 dBase 的版权之后不久，便把公司更名为 dBase Inc.公司，1 年之后，也就是 2000 年的 12 月，dBase Inc. 推出了自己的新 dBase 产品，取名为 dBase 2000，至此 dBase 系列产品也终于正式延续了产品生命。dBase 在 80 年代诞生以来，持续生存了近 20 年的时间，可说是 PC 软件中生命力最为强韧的软件了。dBase 被 Philippe Kahn 的狂妄自大所牺牲，IntraBuilder 因 Delbert Yocam 目光如豆而夭折。虽然 dBase 因为再转卖给其他公司而得以延续生命，但是对于 Borland 来说，dBase 和 IntraBuilder 终究是在遗憾中结束了生命的产品。

Paradox

对于 Borland 来说，Paradox 一直是棵摇钱树，为 Borland 赚进了大把的钞票，同时也让 Borland 称霸 PC 数据库市场。不过 Paradox 并不是 Borland 自己研发的(嗯，写到这里，我才突然想到，Borland 数据库产品几乎都不是自己开发的，都是并购来的)，是从一家叫做 Ansa 的小公司购买来的。在 1985 年，为了进入 PC 数据库市场，Borland 看上了 Ansa 公司的 Paradox 产品，于是在 1985 年的秋季正式购买了 Paradox，成为 Borland 第一个数据库开发工具。

其实，在 Borland 所有的数据库产品中，Paradox 是最受 Borland 照顾的，这也许是因为 Paradox 是 Borland 的第一个数据库产品，也或许是因为 Philippe Kahn 对于 Paradox 情有独钟。但 Paradox 对于 Borland 也有着很大的影响，因为 Paradox 后来不但成为 Borland 的主力产品之一，存取 Paradox 的数据存取引擎也成为数年后 Delphi 的主要数据库和存取引擎，BDE/IDAPI 也是从 Paradox Engine 演化而来的。

Borland 取得了 Paradox 之后，很快就开发出了 Borland Paradox For DOS，正式进军 PC 的数据库市场。由于 Paradox 当时独特的 Query By Example(QBE)以及每一个版本都维持兼容的良好特性，很快就吸引了许多的使用者，Paradox 也成为当时 dBase 系列之外另外一个非常流行的数据库产品，在国外非常的盛行。而 Paradox 之所以没有在台湾/大陆等地流行起来，原因便是 Paradox 一直和 Double-Byte 的兼容有问题，无法正确地处理中文信息。

由于 Paradox 在 DOS 以及 Windows 初期的版本中表现得非常抢眼，因此 Philippe Kahn 一度想以 Paradox 称霸 PC 桌面型数据库市场，投入许多的资源研发 Paradox For Windows，并且不惜压制 Borland 自己的 dBase 产品来壮大 Paradox 在 Windows 市场的占有率。不过，很显然 Borland 的脑筋仍然没有转过来。在 dBase、Paradox 和 FoxPro 等 PC 数据库开发工具被使用了多年之后，已经开始慢慢地进入一般计算机使用者的市场来解决日常数据处理的工作，因此在 PC 桌面型数据库市场，已经开始需要一些比 dBase、Paradox 和 FoxPro 等更简易、好用的产品了。在这方面 Lotus 便掌握得比 Borland 好，因为 Lotus 开始开发适合一般计算机使用者的桌面型数据库工具，那就是后来的 Lotus Approach。

Paradox 的导向一直是以程序员为主，后来 Paradox 引以为傲的开发语言 Paradox Application Language(PAL)也以面向对象为宣传重点，强烈地吸引着想使用面向对

象技术开发数据库应用程序的程序员。正是因为 Paradox 从头到尾都是以程序员为导向，所以在 Paradox 到达了巅峰之后，仍无法吸引一般的计算机使用者，也无法进入这个新兴的市场--Paradox 对于这类计算机使用者而言实在是太困难了。因此，当 Lotus 的 Approach 步步为营(嗯，"Approach"这个名字还真符合当时的状况)，掌握了新起的 PC 桌面型数据库工具市场之后，Borland 等于同时失去了 dBase 市场给 FoxPro，又无法通过 Paradox 渗入新的数据库工具市场。

当 Borland 也察觉到 Paradox 的瓶颈以及新兴起的 PC 桌面型数据库市场之后，急于让 Paradox 进入 Lotus Approach 掌握的市场。因为 Borland 相信，以 Paradox 这么优秀的品质，绝对有机会同其他新的竞争对手一较长短。因此，Borland 开始了 Paradox For Windows 5.0 的研发工作，准备为 Paradox 加入许多简易的功能，以打开新市场的契机。

虽然 Borland 很努力地想要转变 Paradox 的产品形象并且打入新的市场，无奈 Paradox 的产品定位已经非常的固定，而且此时 Microsoft 也进入了 PC 桌面型数据库工具市场，并且以 Microsoft Access 屠杀和血洗 Lotus Approach，Paradox 当然再也没有机会在这个市场称雄了。不过也还好，Borland 晚了一步进入这个市场，才没有让 Paradox 和 Approach 一样被 Microsoft 的 Access 以极为不合理的手段所消灭。

1994 年 3 月，当时的 Novell 还想在 Office 产品线中和 Microsoft 一争长短，因此大手笔地并购了 WordPerfect 公司，并且从 Borland 买走了 Quattro Pro 以及 Paradox 的使用权。Novell 当时的想法是让这些 Office 产品和 Novell 的 Network OS 连接在一起，以便与 Microsoft 抗衡，挽救 Novell 日益下滑的局面。不过，当时我根本不看好 Novell 的这个举动。连开发商业应用程序为主的 Lotus 都不是 Microsoft 的对手，更何况从来不以商业应用程序为擅长的 Novell 呢？而且除了 NOS 之外，Novell 还开发出过什么知名的产品呢？因此，Novell 真正的目的恐怕并不是和 Microsoft 竞争，而是为了固守 Novell NOS 的地盘，防止被 Microsoft 进一步地侵蚀。从这个现象，我们可以知道 Novell 早在 1994 年便开始逐渐采取防守的策略，已经无力和 Microsoft 竞争了。

Paradox 的告别作

Paradox 和 Borland 的缘分似乎已经快到了尽头，虽然 Borland 试图在 Paradox For Windows 5.0 时改变 Paradox 的策略，转向一般计算机使用者，不过 Borland 的努力显然失败了，Paradox 的核心就不是为这个市场设计的。因此，在 Paradox For Windows 5.0 表现得不如人意之后，Borland 又决定把 Paradox 定位在专业的 P C 桌面型数据库工具市场，准备推出下一个 Paradox 重要的版本--Paradox For Windows 7.0。

1995 年 12 月，Borland 推出了几乎是品质最好的 Paradox，即 Paradox For Windows 7.0。严格地说，Paradox For Windows 7.0 是当时所有 PC 桌面型数据库开发工具中功能最强大、品质最稳定的工具，可以说是当时的王者。可惜时不我待，其时大部分的桌面数据库应用都被 Microsoft Access 抢走，一般 PC 使用者的人数远超过数据库程序员的数量，因此，Microsoft Access 的销售量是其他所有 PC 桌面型数据库开发工具的数倍之多，再加上关系数据库也快速地流行于 PC 的应用之中，PC 桌面型数据库开发工具在上/下夹攻之中，市场也逐渐地消失了。

对于 PC 桌面型数据库开发工具市场的不断萎缩、以及关系数据库市场的快速兴起，Borland 也了解到必须正视市场的变化。因此，Borland 开始着手从 Ashton-Tate 取得的 InterBase，准备进军关系数据库市场，同时卖出 Paradox 以集中资源开发 InterBase。此时，刚好 Corel 夹着 CorelDraw 以及绘图软件取得的雄厚资金从 Novell 手中又买下了 PerfectOffice 所有的软件。因此 Borland 也决定一次性把所有的 Paradox 版权卖给 Cord。1996 年 1 月底，Borland 正式和 Corel 签约，卖出最后 Borland 保留的 Paradox 权利给 Corel。

从此，Borland 不再拥有任何 Paradox 的权利，也不再继续开发 Paradox。这也就是为什么 Delphi/C++Builder 之中的 Paradox 数据库规范最高只到 Paradox 7，因为 Borland 再也没有权利开发新版本的 Paradox 以及 Paradox 引擎和数据库规范了。

Corel 在取得了 Paradox 之后，也持续地开发 Paradox For Windows 一直到 9.0 的版本，但对于市场已无任何举足轻重的影响，因为延续 10 几年的 PC 桌面型数据库市场已然退出市场的主流，不管是 dBase、FoxPro、Paradox 还是其他的类似产品，也注定被 Access 和关系数据库所逐渐取代。

后言

Borland 在 PC 桌面型数据库以及关系数据库方面的战役一直是问题连连。除了 Paradox 之外，Borland 接连错失了以 dBase 主掌天下的大好时机，也没有及早通过 InterBase 进入关系数据库市场。如果当时 Borland 能够在一开始从 Ashton-Tate 取得 InterBase 之后，立刻研发和进入关系数据库市场，那么以当时 Borland 的力量绝对可能成为关系数据库的霸主。因为在那个时候，Oracle 等公司还是非常小的，而 Microsoft 也没有关系数据库的产品，但是 Borland 手中却有 InterBase。无奈，Philippe Kahn 没有看到未来数据库市场可怕的成长潜力，任手中的宝贝闲置了好几年。等到其他的关系数据库厂商已经闯出了名号后，才发现原来自己家中早已有一个好东西，但是在落后别人已多的情形下想要追赶，却已不容易了。

Borland 在 PC 数据库市场上犯了过多的错误以及失去了许多宝贵的机会，否则很有可能主宰 PC 数据库市场、持续地和 Microsoft 竞争，并且站稳软件大厂的地位。回顾 Borland 在 PC 数据库市场的搅和，实在是令人不解而又令人叹息!既然有眼光并购潜力十足的各个数据库厂商，为何又放任大好的契机流失呢？这个问题的答案恐怕只有 Philippe Kahn 才知道了。

^v^v^v^v^v^v^v^v^v^v

第七章　中途岛之战--Borland 和组件技术

"没有中间件技术，我们就没有未来!"

Golden Gate Strategy

1996 年，Borland 察觉到软件技术将会开始朝中间件的方向发展。由于 Borland 一向只开发工具软件，因此如何面对这个软件趋势便成了重要的问题。当时 Borland 陷入一片混乱之中，新任 CEO Delbert Yocam 还尚未进入公司，软件和产品线的开发方向几乎都是由担任 Borland R&D Director 一职的 Paul Gross 负责。1996 年 7 月，Paul Gross 和 Delphi 的负责人 Zack 共同激活了一个崭新的计划，其目的是为了让 Borland 能够在未来的软件业界中保持高度的竞争力。

当时，Borland 已经开始想往企业市场发展。但是，Borland 缺乏企业市场需要的大型架构技术，那就是所谓的中间件(Middleware)。中间件通常都非常复杂，而且需要许多时间才能够开发完成，更何况中间件服务器都需要运行在许多不同的硬件平台上，例如 SUN、HP 和 IBM 等大型机器中，而那时 Borland 只是一个开发 PC 软件的公司，不但对大型机器的开发完全没有经验，而且也没有相关的硬件设备来支持开发。尽管如此，Borland 和 Paul 都知道，在未来中间件技术绝对是软件产品的主战场之一。如果 Borland 不趁早往这方面发展，那将永远没有机会成长为大型的软件公司。因此，Paul 和 Zack 知道，Borland 必须想办法克服所有困难，以取得中间件技术。当然，最快的方法就是并购拥有这方面技术和产品的公司。在寻寻觅觅了一段时间之后，Paul 终于找到了在 Boston 的一家软件顾问公司。虽然这家公司不大，但是却拥有使用 RPC(Remote Procedure Call)通讯协议技术的中间件市场的领导产品--Entera。

RPC 是一个存在了非常久的软件技术，发展得非常成熟。Entera 被许多如 HP 之类的大

型公司使用。于是 Paul 想通过并购这家软件顾问公司以取得 Entera，再通过 Entera 把 Borland 打入企业级市场。如果 Borland 能够整合 Entera 和 Borland 的开发工具，那么大型企业可能也会开始使用 Borland 的工具。这对于 Borland 来说，算是很好的机会，因为如此一来，Borland 不但可以取得中间件技术，更可以让开发工具进入以往 Borland 难以进入的市场。

就在 Paul 心动之际，又恰逢这家位于 Boston 的软件顾问公司经营发生了问题，也在寻找新的资金来源，因此和 Borland 可以说是一拍即合。不久之后，Borland 便宣布，Entera 正式成为 Borland 的产品之一。在 Paul 决定并购 Entera 之后，立刻激活了 Golden Gate Strategy(金门战略)，开始要求 Borland 的开发工具必须和 Entera 整合在一起。同时 Borland 也第一次开始大量购买大型的硬件，准备研发 Entera，至此 Borland 通过 Entera 正式进入了大型的以及基于 UNIX 平台的软件市场。

在 Borland 取得了中间件技术和产品之后，便很高兴地把 Golden Gate Strategy 呈现给世人，宣示 Borland 已经成为整合科技的领导厂商之一。当时 Paul 还特别拨了一笔预算，拍摄了一个宣传 Borland Golden Gate Strategy 的动画影片，其中使用的宣传语是"We don't want to own the world，we just want to make it work better(我们不想拥有世界，只想让它运作得更好)"。相信许多读者可能会记得这个影片。

Borland 在取得了 Entera 之后，算是进入了陌生的中间件市场。虽然 Borland 通过 Entera 企图打入企业市场，但严格地说，Entera 只是让 Borland 这个招牌被较多的企业知晓。而 Borland 在销售 Entera 方面表现得并不好，因为 Borland 一开始并不熟悉 RPC 技术，另外就是 Borland 当初的销售体制无法成功地销售企业级的软件，因为新的公司体制尚未建立起来。由于 Entera 在之后的表现不如人意，后来几乎只有 Entera 的旧客户才购买新版本，Entera 已经无法吸引新的客户了。这当然也是因为市场上的中间件技术主流已经慢慢转换为使用 CORBA 和 DCOM 技术。因此，不久之后，Borland 便把 Entera 的维护和开发新版本的工作交由 Borland 亚洲研发中心新加坡来处理了。

Borland 一直到取得了 CORBA 技术之后，才开始真正掌握中间件技术，并且逐渐打入企业市场。

并购 Visigenic，取得 CORBA 技术

Borland 在第一次的中间件尝试不甚成功之后，还是没有放弃想在中间件开发的决心。当企业市场的中间件主流技术转为使用 CORBA 之后，Borland 又看到了第 2 次机会。当 CORBA 技术逐渐被企业和 PC 界视为明星技术之后，提供 CORBA 相关产品的 IONA 和 Visigenic 便成为许多人眼中的潜力软件公司了。不过由于 IONA 已经早一步推出了 CORBA 产品并且也已经拥有了许多客户，因此 IONA 成长得非常快速。相反 Visigenic 是刚成立不久的小软件公司，而且还处于亏损状态。

Visigenic 是由 Roger Sippl 先生创立的。Roger Sippl 是信息业界非常有名的人，因为他也是 Informix 的创始人。像 Roger Sippl 这样的人，在美国被称为"创投冒险家"。这群人的特点就是拥有极为敏锐、先趋的眼光，不断找寻新的信息契机成立软件公司。一旦新公司有了一点成果之后，便立刻果断地卖出公司以求获利，而甚少长久经营一家公司。

Roger Sippl 创立了 Informix 公司，在有了一点经营成果之后，立刻把 Informix 卖掉，大赚一笔，此后又再试图建立其他的小公司寻找机会和买家。Visigenic 就是后来 Roger Sippl 看好 CORBA 技术之后成立的小公司。Borland 看到 CORBA 的潜力，但没有足够的本钱并购 IONA，因此看上了规模还小的 Visigenic。Borland 找上 Visigenic，表示愿意以数百万美金和股票分配权购买 Visigenic 公司，当然 Roger Sippl 立刻答应了。因为如此一来，Roger Sippl 不但可以让 Borland 负责 Visigenic 的负债，还能够赚进大把的

现金和 Borland 的股票，何乐而不为呢！

因此在 1997 年 11 月，Borland 正式购买了 Visigenic，而 Roger Sippl 也成为 Borland 当时的 CIO。当时我就知道，Roger Sippl 一定是暂时性的担任 Borland 的 CIO，目的就是帮助 Borland 顺利接收 Visigenic 的产品线。一旦 Borland 掌握了之后，Roger Sippl 一定会立刻离开 Borland，再次找寻新的机会。果然当 Visigenic 的产品在 Borland 稳定之后，Roger Sippl 也就离开去创立其他的软件小公司了。不过，对于 Borland 来说，这并没有损失，因为 Borland 要的本来就只是 Visigenic 的 CORBA 产品。1998 年，我还在 Borland 的总部 Scott Valley 聆听了已经快要谢顶的 Roger Sippl 的演讲，宣示 Borland 未来在中间件市场的光明前景。那也是我最后一次看到 Roger Sippl。Borland 取得了 CORBA 产品之后，果然没有再败坏家产，而是立刻投入资源开发 Borland 自己的 CORBA 产品线，那就是 VisiBroker。很快，Borland 就有了成果。当时 Netscape 正和 Microsoft 火拼到最高点，Netscape 为了增加 Navigator 在企业级市场的优势，决定在 Netscape 中内建 CORBA 客户端引擎。在 Netscape 评估了 IONA 和 Borland 的 CORBA 产品后，决定使

用 Borland 的 CORBA 引擎，因为当时 Borland 虽然没有像 IONA 一样拥有比较完整的 CORBA 产品和 CORBA 服务，但是 VisiBroker 却拥有体积小、执行速度快的优点，正好适合在客户端的浏览器中使用。

当 Netscape 找上 Borland 的时候，Borland 简直是喜上心头。当时的 Netscape 是如日中天的软件公司，全世界使用 Navigator 浏览器的人数超过数百万。如果 Netscape 决定使用 VisiBroker，那 Borland 不但得到了最大的客户，而且还可通过 Netscape 的名气立刻让全世界的人、包括企业级的使用者都知道 Borland 这家公司，了解 Borland 是有能力提供企业级的软件解决方案的。

在 Netscape 和 Borland 磋商之后，立刻就有了结果。Borland 答应以极低的价格授权 Netscape 在 Navigator 中使用 VisiBroker。虽然在这次的商业谈判中，Borland 几乎没有什么赚头，但是，Borland 却达成了极为重要的形势胜利。首先是通过 Netscape 的授权使用，Borland 的 VisiBroker 在 CORBA 市场的占有率立刻超越了 IONA；第二是 VisiBroker 通过 Navigator 打入了企业市场，让许多大型的企业开始对 VisiBroker 产生了兴趣，以致后来 VisiBroker 在金融和电信领域突破 IONA 的阵地，成为更受欢迎的 CORBA 引擎；第三点当然就是 Borland 可以通过 Netscape 这个成功的案例宣传 Borland 的 CORBA 产品，让 VisiBroker 再也不会矮上 IONA 的产品半截了。

在这次成功地出击 CORBA 市场之后，Borland 终于开始让 IONA 正视自己为最强劲的竞争对手了。这也开始了 Borland 和 IONA 之间无止境的 CORBA 大战，双方在各个 CORBA 应用

领域厮杀惨烈，一定要分出高下。

当然，Borland 在 CORBA 的成功也让 Patti 和 Zack 的 Golden Gate 计划显得比较圆满，而且在 Paul 和当时 Borland R&D Director Joe Bently 的要求下，Delphi 和 C++Builder 也都开始支持 CORBA 的功能。至此 Golden Gate 计划逐渐走向成型阶段，Borland 终于在中间件技术杀出了一条血路。

Paul Gross 的愤怒和 Golden Gate 的坠毁

但不幸的是，当 Delbert Yocam 在 1999 年 4 月被 Borland 董事会踢出门之后，担任 Borland R&D 副总裁的 Paul Gross 便一直认为 Borland 新的 CEO 职位应该非他莫属了，于是开始积极争取 CEO 一职。不过，Borland 的董事会却认为如果公司真的想进入企业市场，就必须拥有专业的销售团队，需要一个在销售领域非常有经验的人来担任 CEO。Paul Gross 出身于 R&D，对于销售没有太多的经验，无法为 Borland 建立起所需要的销售团队。因

此，Borland 董事会决定从外面寻找新的 CEO。
在得知了 Borland 董事会的决定之后，Paul Gross 非常愤怒。因为他认为 Golden Gate Strategy 是他策划的，Borland 能够进入企业市场，取得中间件技术，都是因为他的眼光和功劳。现在 Borland 董事会居然不考虑由他接任新的 CEO，Paul Gross 心中充满了怨言，对于 Borland 产品线的开发工作也就顿时失去了兴趣。
在 Borland 开始向外寻找新的 CEO 之际，Microsoft 也得知了这个信息，于是马上就和 Paul Gross 接触，想了解 Paul Gross 的动向。此时正是 Paul Gross 最不满的时候，因此和 Microsoft 相谈甚欢，很快便答应到 Microsoft 工作。由于 Paul 在 Borland 已经是 R&D 部门的副总裁，因此 Microsoft 答应给 Paul 的职位也是 Microsoft 开发工具部门的副总裁，负责 Microsoft 的 Visual Studio 以及 Internet 方面的产品研发工作。这又是一个 Microsoft 直接挖角 Borland 人才到自己公司担任类似工作的例子。Paul Gross 虽然是以副总裁的角色被挖走，Microsoft 给予 Paul 的待遇也很高，但和 Anders Hejlsberg 比起来还是差多了。这也说明美国对于技术专才的重视，高层管理人员的待遇不见得会比第一流的技术人员高。
注：从 Paul Gross 的事例中，似乎可以看到技术人员的宿命。技术人员最高的职位好像就是技术副总裁了，想做 CEO 是不太可能了的(除非是技术人员自己开的公司)。这是真的吗？
看来，Paul Gross 在 Gold Gate Strategy 中写错了那句名言。Paul Gross 是想拥有全世界的，所以才加入 Microsoft，不是吗，Paul？
随着 Delbert Yocam、Zack Urlocker 以及 Paul Gross 等人一一离开 Borland，Golden Gate Strategy 就再也没有人提起了。虽然 Borland 已经拥有了当初 Golden Gate Strategy 规划的所有软件技术，不过在 Java 快速兴起并且掌握了企业市场之后，软件的发展似乎已经移转到了 Java 应用程序服务器市场。这个软件趋势也造就了 BEA 的快速成长。等到 Borland 的下一任 CEO Dale Fuller 先生任职之后，他立刻投入大量的资源进入 Java 应用程序服务器的竞争市场。至此，Golden Gate Strategy 也就正式成为历史灰烬了。
到 EJB 的阵地吧！
虽然 Borland 在 CORBA 领域逐渐超越 IONA，并且开始成为市场的领导者，不过，Java 的日渐兴盛也开始让 CORBA 面临考验。而在 Windows 平台，CORBA 也必须面对 Microsoft 的
COM/COM+中间件技术的竞争。当 SUN 提出 Java 的 RMI 技术后，一些对 CORBA 并不了解的
人开始鼓吹 CORBA 已经是日薄西山了，他们认为 CORBA 这个已经存在许久且复杂的技术可能会被 Java 方面的 RMI 取代。不过，后来事实很快就证明 RMI 根本无法取代 CORBA。先不说 RMI 的功能和 CORBA 比起来简直是九牛一毛，更何况 RMI 的执行效率根本不是 CORBA
的对手，因此关于 RMI 是否能取代 CORBA 的争论很快就平息了下来。
不过，当 SUN 提出了 J2EE 架构准备力推 EJB 技术时，又有许多人不看好 CORBA 的命运，认为 CORBA 的长路将尽，Java 世界终将使用 EJB 而不是 CORBA。一开始，SUN 也认为 EJB
是最好的中间件技术，认为企业应该采用 EJB 作为中间件的引擎来开发企业应用系统。
不过当 SUN 真正试着把 EJB 打入企业市场时，才发现 CORBA 早已在企业市场根深蒂固，许多大型的应用系统都是使用 CORBA 技术开发的。如果 SUN 要把 EJB 推上企业中间件的主流，那么 EJB 一定要能够同 CORBA 兼容和沟通。因此后来 SUN 修改了 EJB 的规格，让

它和 CORBA 兼容，以顺利解决大型企业需要整合 CORBA 以及新的 EJB 技术的需求。Borland 很早就发现 CORBA 和 EJB 根本不冲突，反而有很高的兼容性，CORBA 的跨平台特
性又非常适合用来实现 EJB 的功能规格。因此在 1999 年，Borland 开始进行用 VisiBroker 作为实现 EJB 服务器引擎的研究工作。Borland 在中间件市场付出了庞大的成本，好不容易才在 CORBA 方面有了一点成果，如果又被 EJB 抢走中间件市场的主流，那么 Borland 岂不是损失惨重？而且 Borland 好不容易通过 Golden Gate 计划进入中间件市场，从 RPC、CORBA 一路走来，既然已经在中间件市场投下了重注，那就没有理由现在因为 EJB 而退出市场。更重要的是 Borland 看到了未来 EJB 市场的潜力，如果 SUN 真的能够把 J2EE 架构打人大型企业应用的核心，那么掌握 EJB 服务器的厂商将会拥有巨大的商机，因此 Borland 无论如何都不能放弃这个市场。

为了在 EJB 市场成为领导厂商之一，Borland 不惜巨资投入了许多的资源来研发 EJB 服务器的产品，并且以开发工具和 CORBA 产品赚得的资源源源不断地资助 EJB 开发小组，以期能够早日开花结果，让 Borland 在 EJB 的中间件技术市场再下一城，奠定 Borland 在这个领域技术至尊的地位。不过事情发展得却并不像 Borland 预期的那样简单。

几乎也在同一时期，另外两位重量级的厂商 BEA 和 IBM 也分别投下了巨大的资源进入 EJB 市场。BEA 选择的方式是快速并购 Tuxedo，并且以 Tuxedo 为引擎开发 EJB 服务器；而 IBM 则是占有最丰富的资源优势，以最大的团队规模投入 Java 和 EJB 市场。虽然 Borland 也投入了巨资，但是 Borland 的"巨资"和这两家比起来根本不够看。很快 BEA 和 IBM 就有了成果。当然，由于 UNIX 服务器战场的关系，当 BEA 和 IBM 推出 EJB 的服务器之后，立
刻引起 HP 等大厂商加入 EJB 来火拼。SUN 自然也不会置身事外。一场 EJB 的世界级大战展开了。

这么激烈的 EJB 战场，是当初 Borland 可能没有想到的。加入中间件技术大战的厂商每一个的规模都比 Borland 大上几十倍，甚至是数百倍。这种大规模的阵仗是 Borland 前所未见的。因此，当 Borland 开发出了 EJB 服务器引擎之后，才发现在大型中间件市场不是光比技术和产品。除了技术和产品之外，还要比公司招牌的知名度、专业服务、专业顾问咨询和专业的销售团队。此外，更需要极大的公司资源准备打一场长期的消耗战。

一直到进入了 EJB 市场，Borland 才真正了解到这个市场的对手是如何对战的。为了在这个割喉市场生存下去，Borland 公司策划了第三次的转变，以便为 Borland 建立专业的销售体系，准备以不同的手法开打 EJB 的战争。这在稍后 Borland 的演变一文中有详细的说明。

够强壮再玩下去吗?

到了 2002 年，中间件市场已经到了成熟而且是最后关头的时刻。EJB 厂商之间的竞争已经快见分晓，Borland 是否能够坐稳除了 BEA/IBM 之外的第三领导厂商地位，事关 Borland 能否在 EJB 服务器市场存活下去。又或是 Borland 与 BEA 成为联军，一起攻占其他 EJB 竞争对手的最后滩头堡呢？这个答案也许在 2003 年会揭晓。不过，不管如何，现在的 Borland 似乎又在中间件市场找到了另外一线生机，那就是.NET 平台下的中间件战争。这个故事将在稍后"EJB 对抗 CORBA？有趣的假设"一章中讨论。

^v^v^v^v^v^v^v^v^v^v

第八章　Borland 的成长和改变

许多人都用过 Borland 的产品，就像我和正在阅读本书的读者一样。有的人非常喜欢 Borland，有的人则只把 Borland 当成一个普通的软件公司。不可讳言，软件人员对于

自己使用的工具或语言，总有着一股喜爱和狂热。当然，在爱屋及乌的效应下，他们对于推出工具或语言的公司自然也有了类似的情结。

对于许多软件人员来说，"Borland"这个名字是非常可敬的。因为 Borland 敢于对抗 Microsoft，历经十几年仍然屹立不倒，而且还能不断地推出令人惊喜的产品。这令人印象深刻，使人打心底里佩服。当然，Borland 的产品品质有高有低。面对优秀产品，总会让人高兴不已，让人有少康中兴、打败 Microsoft、一吐心中闷气的愉悦；相反，对于并不令人满意的产品，又有一种恨铁不成钢、混杂着担心 Borland 未来的焦虑感。这种既期待又悸动、心情上下反复的感觉，是那些不喜欢 Borland 产品的人所不能体会的。

在十几年的发展中，Borland 展现了强劲的生命力。在每一个关键时刻，Borland 以不同的产品、技术和策略度过了一次又一次的挑战和难关。虽然 Borland 始终代表着除了 Microsoft 以外的最大独立软件开发厂商，可事实上，她从创立一直到现在，已经历经了三个重大的转变。这些转变不但影响了 Borland 产品的走向，也影响了 Borland 的组织架构以及未来的发展。不知喜爱或是关心 Borland 的读者是否注意到这些转变？在继续阅读本章的内容之前，不妨先回头想想您是否知道这些转变。

Philippe Kahn，产品和技术为主的 Borland

Borland 的创始人 Philippe Kahn 代表的是 Borland 第一时期。在 Philippe Kahn 主掌期间，Borland 从无到有、茁壮成长至全盛，成为全球瞩目的、世人最看好的软件公司之一，继而业绩快速下滑终至差点解体。Philippe Kahn 创造了 Borland 最辉煌的软件王国，也差点成为亡国之君。先不论 Philippe Kahn 的功过如何，总之，在他主控的时期，Borland 最注重产品和技术，甚至日后 Borland 的命运也似乎和 Philippe Kahn 的个性有着密切的关系。

恃才傲物的 Philippe Kahn 从来不给美国华尔街的分析师什么好脸色。1991~1993 年，是 Borland 最会赚钱的时代，也是 Borland 内部生产力最高峰的时期。当时 Borland 的股票价格高高在上，最高时期曾经每股单价超过了 100 多美元。在那个时候，Philippe Kahn 几乎是 Borland 的国王，拥有 Borland 大量的股票。功成名就的 Philippe Kahn 认为 Borland 的成功都是他的功劳、都应归结于他的领导，况且 Borland 的实力和产品非常坚强，不需要为那些华尔街的分析师进行什么公关工作，Borland 的股票仍然将是投资大众的最爱。因此，当华尔街的分析师希望 Philippe Kahn 能够向他们说明 Borland 未来的开发方向以及公司在管理和财务方面的信息时，Philippe Kahn 并不理会这些分析师。所以，在 Philippe Kahn 时代，Borland 和华尔街的关系并不好。

有一次，华尔街的分析师邀请了许多公司的 CEO 演讲，其中也包括 Philippe Kahn。没有想到，Philippe Kahn 的现场演讲却完全不给这些分析师面子，还嘲笑他们是笨蛋，因为 Borland 完全不需要任何的分析，Borland 就是最好的公司，投资 Borland 准没有错。Philippe Kahn 这个自大而且不给面子的行为终于惹火了许多华尔街的分析师。这不但使 Borland 和华尔街的关系更为紧张，而且，许多分析师也常常借机修理 Borland，宰杀 Borland 的股票。更糟糕的是，在 Borland 从 1994 年逐渐走下坡之后，许多分析师更是落井下石，看坏 Borland 的未来。因此，不时地有 Borland 将会关闭、被并购或是被拍卖的不实消息散出，让 Borland 的股票价格快速地往下落底。这都是 Philippe Kahn 种下的恶果。

Philippe Kahn 对于产品和技术的坚持造就了 Borland，但是太过坚持反而变成了刚愎自用，他完全听不进去其他的意见。通常来说，Philippe Kahn 一定会参加 Borland 许多的产品开发计划会议。对于好的或是他喜欢的产品，Philippe Kahn 坚定支持，即使产品可能没有市场，他也不去理会。或者是因为太过于坚持产品、一定要尽善尽美，

即使严重地延误了产品应该上市的时期，造成产品已经没有推出的价值，Philippe Kahn 也不管。因为 Borland 是他的，他有 100％的决定权。此外 Philippe Kahn 在许多会议中太过于强势，喜欢主导产品的研发，造成产品开发主管不知如何是好，因此惹火了许多有天分的研发人员，致使他们纷纷求去，Borland C/C++的 Eugene Wang 就是一个很好的例子。

Philippe Kahn 少年得志，以至于最后目中无人、狂妄自大，终致鞠躬下台。不过，Philippe Kahn 毕竟是所有 Borland CEO 中最重视产品开发的。自他之后，再也没有任何的 CEO 能像他一样，知道唯有产品和技术才是 Borland 的命脉。Philippe Kahn 对于 Borland 未来发展的规划也是以产品和技术为主轴。当时的 Borland 最重视软件开发人才、工程师和产品经理。但从 Philippe Kahn 下台之后，整个形势慢慢地有了改变。不过，总体来说，这个时期的 Borland，是我最喜欢的。

Delbert Yocam，强势 Marketing 为主的 Borland

Borland 第二个重要的时期应该是 Delbert Yocam 上台之后。在 Philippe Kahn 被赶出公司之后，Borland 历经了一段没有真正 CEO 的时期。在这个时期，担任 CEO 的人都只是暂时的，因为 Borland 董事会决定从业界寻找有实际大型企业管理的人物接手 Borland CEO 一职，以带领 Borland 走出低潮。事实上，从这个时期开始，Borland 已经逐渐走向一般大型软件公司的路线，而不再以充满软件开发者理想的公司自居，因为此时的 Borland 需要的是大量的收入和有效率的管理，以拯救 Borland 濒临破产的命运。Borland 放弃了 Philippe Kahn，建立起 Borland 理想而务实的生存之路。

寻寻觅觅了一段时间之后，Borland 终于找上了曾任 Apple 副总裁的 Delbert Yocam。由于 Delbert 拥有丰富的大型企业经验，并且对软件公司很有兴趣，因此和 Borland 董事会一拍即合，答应担任 CEO 以重振 Borland。事实上，在 Delbert 之前，Borland 的董事会也曾找了许多知名的美国大型企业经理人，希望他们担任 CEO。但由于当时 Borland 已经岌岌可危，许多专业经理人都不感兴趣，致使 Borland 一直无法找到理想中的 CEO。所以，Delbert 的应答立刻让 Borland 董事会精神大振，认为 Borland 终将起死回生，回复以往光荣的历史。不过始料未及的是，数年后的事实证明，找 Delbert 担任 CEO 是一个错误的决策。

不知读者是否发现问题：Borland 董事会中应该会有经验丰富的人选，为什么不直接从中寻找而要从外面找人呢？其实，原因也同样，那就是 Borland 董事会中有资格担任 CEO 的人也认为这个工作艰巨，不敢轻易尝试。另外一个问题则是，既然数年后证明当时找 Delbert Yocam 管理 Borland 是一个错误，那为什么这些经验丰富的董事会成员却无法预知当初就不应该找他担任 Borland 的 CEO 呢？可见，要成功地管理一个公司是不容易的，即使是经验丰富的专业人士，也不见得能够找到适当的管理人才。

Delbert Yocam 成为 Borland 的 CEO 之后，立刻开始了两项工作。第一是把他在 Apple 的工作团队带入 Borland。Delbert 逐一安插他的工作团队到 Borland 的每一个重要职位，以便完全掌握 Borland 并且执行 Delbert 的意志。Delbert 从 Apple 引进 Borland 的人物，大多属于管理阶层的经理人，只有 Rick LeFaivre 属于信息技术方面的人。当时，Rick LeFaivre 被任命为 Borland 的 CTO(Chief Technology Officer)，负责掌管 Borland 的信息技术和产品开发。

Delbert 在成功地建立了自己的管理阶层之后，第二步便是开始掌握 Borland 的产品。为了实现改造 Borland 的计划，Delbert 很快就提升 Zack 为 Borland Marketing 副总裁，开始试图改善 Borland 一向积弱不振的 Marketing 表现。

刚开始接手时，Delbert 是有心好好地管理 Borland。因此，在 Delbert 管理的早期，Borland 还有盈余。除了 Delbert 个人比较奢侈的享受外，并没有太大的错误。我也曾

经在 Borland 的总部听过 Delbert 对员工讲话，那是因为当时的那一季 Borland 表现得很好，Delbert 很高兴地鼓励大家继续努力。演讲之后，公司还举行了一个小 Party，提供有大量的食物和饮料。嗯，当时我还大饱了一顿口福。

不过，随着时间的过去，由于 Delbert 过度干涉产品的开发，Borland 开始走向亏损，公司股价也始终没有出色的表现。更麻烦的是，Delbert 砸下巨资让 Zack 领导的 Marketing 部门不但没有太大的表现，反而招致更多的抱怨，让 Delbert 管理下的 Borland 烽火连天、麻烦处处。Borland 董事会开始质疑 Delbert 的管理能力。在后期并购工作失败、而且 Delbert 带入的管理团队看到大势不妙、纷纷跳船求去之后，Borland 董事会也终于决定换掉 Delbert，另觅新的 CEO，自此 Delbert 的时代宣告终结。

基本上，我认为 Delbert 想要改善 Borland 在 Marketing 方面的弱势是正确的步骤，因为 Borland 一直拥有好的产品和技术，但在 Marketing 方面却一直混乱一团。当时，Borland 的收入主力仍然是 Box-Moving 的产品。这就是说，Borland 主要是以销售盒装的开发工具为主，尚未进入以组件和中间件为主的时期。因此，强化 Borland 在市场的知名度以及领导地位，再配合好的产品，Borland 是大有可为的。但是，Delbert 过于心急地干预产品开发时程和方向，这不但没有改善 Borland Marketing 的弱势，还把产品搞坏了，实在是赔了夫人又折兵，也让 Borland 元气大伤。

Dale Fuller，高效率 Sale Force 的 Borland

Delbert 离开 Borland 之后，Borland 的 CEO 真空了一段时间，因为董事会还在搜寻适当的人选，希望他能够带领 Borland 走出困境。不过在 Borland 找寻 CEO 的同时，软件业界进步和竞争的脚步并没有停顿下来。在开发工具和软件产业中，除了 Box-Moving 的产品之外，组件和中间件的力量愈来愈强。BEA 便是在这段时间通过购买了 Tuxedo 之后快速地在 EJB 应用程序服务器的市场中成长。而 Borland 在并购 Visigenic、取得了 CORBA 的技术和产品，而且也推出 EJB 应用程序服务器之后，Borland 传统的销售 Box-Moving 方式却不适合用来销售中间件。因为 Box-Moving 的产品可以靠 Channel 卖出，但是中间件产品却需要搭配教育培训、专业咨询服务来一起销售。另外，中间件产品需要较长的销售期间，更需要专门的销售人员来服务客户，这些都是当时 Borland 没有建立起的公司制度。因此，即使 Borland 拥有最好的组件和中间件技术，却因为不知道如何销售而任凭好时机消逝。此外，在 Windows 平台上的开发工具也已经逐渐达到了饱和。虽然 Windows 平台上的一些开发工具厂商慢慢地淡出，使 Borland 和 Microsoft 能够逐步蚕食其他厂商留下来的市场，但无疑这已经所剩无多。因此，Borland 需要新的市场、新的刺激，才能够持续成长。

Borland 下一任 CEO 面对的挑战是很艰辛的，因为 Borland 需要再次改变，以适应销售组件和中间件的产品。另外，新的 CEO 必须开拓新的市场，让 Borland 的开发工具产品能够拥有成长的空间。最后，新的 CEO 必须能够带领 Borland 走出亏损，并且想办法拉高 Borland 的股价。Borland 的董事会在接连找了数个失败的 CEO 之后，此次对于新的 CEO 特别谨慎，希望新的 CEO 不要再把 Borland 搞砸了。

Borland 董事会找上 Dale Fuller，是因为当时正值 Internet/Intranet 热潮兴起之时。Dale Fuller 成功地创办了自己的公司、并且在经营了一段时间之后又漂亮地把公司以高价卖出。因此，Borland 董事会对于 Dale Fuller 高明的手腕和眼光深感兴趣，开始接触 Dale Fuller，看看 Dale 是否有意愿接手 Borland。在卖掉了自己的公司之后，Dale 此时也正需要另外一个一展身手的舞台，所以，当 Borland 的董事会和他接触之后，Dale 开始注意 Borland 这家软件公司，并在思考了一段时间之后决定接任 Borland CEO。当然，我不知道 Borland 董事会和 Dale 之间的约定，不过，在 Dale 开始接任 Borland CEO 之时便有谣传，说 Dale Fuller 是希望以高明的手腕看看能否为 Borland 找到归属。

因为，Dale Fuller 选择了大量的 Borland 股票权而不要求高薪，希望用股票选择权大赚一票。

和 Corel 的合并案

Dale 上任之后不久，便逐渐接近公元 2000 年全球 Internet/Intranet 疯潮以及股市狂飙的时期。当时所有和电子商务以及 Linux、CORBA 有关的软件公司的股票都狂飙不已，例如 RedHat、BroadVision 和 IOAN 等。但是奇怪的是，虽然 Borland 也拥有 CORBA 的技术和产品，可是 Borland 的股票却从来没有狂飙过，真是令人不解。Dale Fuller 眼看 Linux 概念股都开始狂飙，立刻决定投入 Linux 市场，责令 Borland 的 RAD 部门必须立刻开发出 Linux 上的开发工具(这就是 Kylix 的源起)，希望能够通过这个举动吸引全球看好 Linux 的热钱大买 Borland 的股票。

此外，Dale Fuller 和当时 Corel 的 CEO Michael Cowpland 经人介绍之后，两人一见如故。当时 Corel 决定开发 Corel Linux，而 Borland 也决定开发 Linux 上的开发工具，更重要的是两人都看好 Linux 的热潮，而且当时 Corel 的股票已经开始上涨到 30 多美金。因此，Dale 和 Michael 在试探了彼此合作的意愿之后，立刻一拍即合。公元 2000 年的 2 月 7 日，Corel 和 Borland 宣布了两家公司准备合并的消息。

在 Dale Fuller 和 Michael Cowpland 宣布合并的消息之后，原本以为靠着 Linux 概念和两家公司(一个开发 Linux 操作系统，一个负责 Linux 开发工具)的完美组合，肯定会带动两家公司股票价格的上涨。

没有想到，消息公布之后，却不被业界看好。因为当时许多业界的专家都已经预测 Corel 撑不了多久，原因是 Corel 的产品的市场占有率快速地下降，花下大把银子推出的 Corel Office 根本无法动摇 Microsoft Office 的地位。此外，Corel 的现金也即将用完、情形不妙。

果然，合并消息发布之后不久，Corel 的股价很快地往下跌，而且居然跌到比 Borland 还低。至此，Borland 和 Corel 的合并应该算是破局了，因为 Borland 不可能和比自己价值还惨的公司合并。于是在公元 2000 年 5 月 17 日，Dale Fuller 终于叫停了这桩合并案。Dale 提出的理由有三：

- Corel 2000 年第 1 季的收入锐减
- Corel 宣布接下来的 2 个 Quarter 的情形也是一样
- Corel 宣布在 3 个月之后现金将会用完

当时 Borland 虽然亏损，但是仍然握有大量的现金，而 Corel 却是现金即将用完。如果不停止这个合并案，那么 Corel 也很快会把 Borland 的现金耗尽，到那个时候 Borland 便万劫不复了。因此 Dale 毅然坚决地停止这桩合并交易，并且付给 Corel 一笔金钱作为取消合并的赔偿。

在此事件之后，Michael Cowpland 也从 Corel 辞去了 CEO 一职。而 Dale Fuller 在遭遇了这个挫折并且无法找到适当的合并公司之后，决定开始改变自己的角色，准备再次转变 Borland，让 Borland 成为适合当前竞争的软件公司。

开始打造成销售的 Borland

Dale Fuller 在抛开了被合并或是并购其他公司的想法之后，第一件事情便是开始回头检视 Borland 的状况和产品线，看看为什么 Borland 会陷入亏损的地步、以及如何让 Borland 重振雄风。其实，当时 Borland 虽然处于亏损，但拥有的现金仍然相当充裕，有稳定而大量的使用者群组，并没有亏损的理由。因此，Dale 首先决定，稳定 Borland 开发团队的军心，弥补 Delbert Yocam 对于 Borland 产品带来的伤害，并且大幅整理 Borland 的管理阶层，尝试为 Borland 建立一个更有效率的专业管理团队。此外，Dale 大刀阔斧地进行了下列工作：

- 稳定开发工具产品线，回归产品开发的本质
- 扩充新的产品线
- 建立专业销售团队

他首先将 Borland 的产品线划分为三个主要部分，分别是负责 Windows/Linux RAD 开发工具的 RAD 部门；负责 Java 开发工具的 Java 部门；以及负责组件和中间件的 Enterprise 部门。每一个部门都建立了明确的管理组织架构，力求改善 Delbert 时混乱的产品管理。

第二项工作便是扩充新的产品线。由于 Microsoft 在 Windows 平台上占尽了优势，虽然 Borland 仍可和 Microsoft 一搏，但 Dale 认为，Borland 如果要持续成长，必须拥有更多的收入。因此，Dale 决定在 Linux 上增加新的产品，除了原先 Linux 上的开发工具之外，也要开发下一代产品。

Dale 发现 Borland 拥有 CORBA 和 EJB 的技术和产品，但是却没有这方面专业的销售团队，使用的仍然是传统的销售方法，造成了这些产品叫好不叫座的状况，迟迟无法打开市场。因此，Dale 执行的第三个计划，便是为 Borland 建立专业的销售团队，使用适当的方式销售 CORBA 和 EJB 产品。于是，Dale 开始在 Borland 的销售区域中建立专业的管理和销售团队，并且开始改变 Borland 的一些公司政策，以便配合新的销售策略。

Dale 的努力果然逐渐显现出成效来。Borland 从 2000 年开始便逐渐转亏为盈，而且营业额不断增加，进而创造了 Borland 在 2001/2002 年连续获利和营收上升的表现，在全世界一片不景气的情形中呈现出亮丽的成果。Borland 的年营收额从当时 Delbert 的 180 几 Million 美金成长到现在超过 220 几 Million 美金，开始愈来愈接近 10 几年前 Borland 全盛时期的营业额了。

Dale 改造 Borland 的行动也影响了 Borland 的日常工作方式。例如，在我以前为 Borland 工作时，除了负责产品和技术之外，也兼做一些 Presale 的工作，因为那时 Borland 没有正式的 Sales 人员，所以技术人员有时也必须客串一下 Sales。但是在 Dale 引入了专业的销售人员之后，有许多工作现在都是由专门的 Sales 人员负责，技术人员则配合 Sales 进行工作。现在的 Borland 编制中有了以前没有的 Sales 团队。

Dale 的改造除了增加了 Borland 的营收之外，也改变了 Borland 的组织结构。Sales 人员和 Sales 部门现在是 Borland 强势的力量，和以往 Philippe Kahn 时以技术人员为主、以及 Delbert Yocam 时以 Marketing 为主完全不一样。现在 Borland 的技术人员和 Marketing 人员必须配合 Sales 来工作，所有员工的表现也以 Sales 绩效为衡量的根据。虽然我非常怀念 Borland 以往的时光，但是我也知道这个转变是不可避免的，因为这是让 Borland 更有竞争力的必要步骤。只是我认为，太过强势的 Sales 在长期来说仍然不太好。Dale 应该在 Borland 逐渐成为有效率的销售公司之后，再回头好好加强在技术和研发方面的力度，让 Borland 能够再开发出几个划时代的伟大产品。

在 Dale Fuller 成功地打造富有成效的销售企业之后，Borland 不但每季都赚钱，更重要的是，在 2001/2002 年全世界不景气中居然还不断地成长。同时，Borland 开始在欧洲和亚洲快速地成长，不断带动 Borland 营收创造新高。这也让许多专家跌破了眼镜，创下软件公司在不景气的气候中还持续成长的范例。Borland 的股票价格虽然没有狂飙，但是已经比许多当时有名的软件公司的股价还高。Dale 成功地完成了 Borland 第三次的转变。Borland 除了技术和产品之外，同时也成为拥有高效率管理和销售的公司，的确是和以前不一样了。现在 Borland 手中不但拥有大量的现金可以并购对 Borland 有帮助的小公司，更有许多知名的软件大公司现在都和 Borland 合作。例如，Sybase 和 Borland 合作推出了 JBuilder-Sybase Edition、JBuilder-WebLogic Edition。从 2000 年到 2002 年，Borland 的确打了一场漂亮的胜战。

第四波的演变

Borland 在历经了三波的改变之后，已经逐渐成了一个比较典型的美国软件公司。虽然不再有 Philippe Kahn 时代以开发者为主、带有理想主义色彩的软件公司形象，但是在本质上比以前好多了。特别是经 Dale Fuller 大力整顿之后，Borland 的营运效率比 Delbert 时代好上数倍。不过，Borland 虽然比以前更有效率、更有竞争力，但是却得面对比以前更为险恶的生存环境和竞争。

首先是软件平台和技术的改朝换代。虽然 Borland 努力在 Java 方面有了一定的成就，但是又面临了 Microsoft .NET 的推出。其实，Borland 一开始也为了是否要跟随 Microsoft 进入.NET 新平台而犹豫不决，因为 Borland 当时想要趁 Linux 的热潮改走跨平台的方向，而不是在 Windows 平台辛苦地和 Microsoft 竞争，这也是为什么 Borland 在.NET 方面进入得比较缓慢的原因。不过，随着 Linux 在 2000/2001 年从爆炸性成长逐渐回归成正常的发展之后，Borland 很快发现，光靠 Java 和 Linux 市场将无法获得足够的利润。另外，在所有知名的信息研究机构的分析都指出.NET 在未来将和 Java 一起占有相当大比重的市场之后，Borland 才知道是不可以失去.NET 市场的。于是，Borland 这时匆匆地投入了.NET 产品的研发。

第二则是随着软件利润的趋于下降，Borland 必须想办法维持公司的成长，开辟新的产品线。从 2000 年开始，Borland 推出并且扩充了 Kylix 产品，研发了新的.NET 产品，并且有数个内部不公开的技术和产品在进行中。这些行动让 Borland 投注了许多的资源，是否能够成功，是对 Borland 开发产品的功力以及公司是否能够在市场和策略方面合理搭配的考验。在现在这个时代推产品，已经不像数年前只要品质够好就可以，还需要其他许多方面的配合。

第三则是随着软件技术愈来愈多，应用也愈来愈多元化。这使得产品的潜在客户群被分散。如何凝聚使用者成了开发工具厂商重要的工作。Borland 除了需要巩固原有的使用者外，也急需开发新的潜在客户才能够达到经济客户规模。因此，Borland 必须在使用者族群方面建立良好的支持社群，并且强化和使用者互动的关系。

第四，Borland 除了需要持续增加效率之外，回归基本面、好好地着重产品开发并且建立坚强的开发者社群是最重要的工作。

软件高科技公司的命运

如果读者是在信息产业经历了一段时间，那么可以观察到目前著名软件公司在经营上风格的异同。每一家公司在成立之初，创办者都有特定的理想或是技术，软件公司也一样。不过随着公司的成长和发展，软件公司的风格却逐渐地出现差异。我说的风格当然不是指每一家的企业文化，这当然会有不同，我指的是软件公司的创始人目前是否仍然是这家公司的主要负责人。

美国企业喜欢并购，许多公司的创始人也喜欢在公司有了一点成就之后立刻卖出公司，以股票换钞票，许多的美国公司董事会也喜欢动不动就换 CEO。这造成许多公司的 CEO 只注重炒短线的工作，只想把公司股票价格炒高，然后卖了公司就闪人。如此做法会让软件公司的产品和营运快速地走下坡路。Borland 就历经过这种 CEO。但是请读者观察，只要是公司创办人仍然还在掌舵或是担任重要职位的软件公司，对于产品就会坚持，不会随意以炒股票来恶搞。例如以 Microsoft 来说，虽然这是一家饱受争议的公司，但是她的创始人 Bill Gates 仍然掌握 Microsoft，因此 Microsoft 仍然在源源不断地开发产品和提出新的构想。不管这些产品是好是坏，我们都不可否认，Bill Gates 仍然在努力地实现他的理想，即使一般大众都认为这是狼子野心。这让 Microsoft 持续地保有活力和竞争力。

再比如，Oracle 的创始人 Larry Ellison 仍然执掌 Oracle。虽然 Larry Ellison 也是属

于好大喜功的人，但是我们不可否认，在 Larry 的领导下，Oracle 仍然振奋向前。这些现象可以从其他的、正在力争上游的软件公司看出，例如 Developer Express、Atozed software 以及目前正和 Microsoft 在多媒体市场打得不亦乐乎的 RealPlayer 等，这些公司的特点便是对于产品的努力和坚持。

反观一些走下坡或是关闭的公司，其中许多都是由于公司的创始人离开而导致。例如以前著名的数据库厂商 Informix，就是由于创始人 Roger Sippl 卖掉 Informix 的股票另起炉灶。后继者对于公司没有热忱，最后因为经营不善而终于成为历史。Informix 如此，Netscape 又何尝不是呢？Netscape 被 American Online 并购之后，快速地走下坡路。前一阵子我看到的新闻消息指出，目前 Netscape 在全世界的占有率只剩下 3%。这个曾经占有超过 90%市场并给予 Microsoft 强烈威胁的浏览器巨人，也在创始人离开之后消声匿迹，逐渐被人淡忘，真是令人不胜唏嘘。

Philippe Kahn 的 Borland 是只会做产品的公司，Delbert Yocam 时的 Borland 只会短线操作，以致让 Borland 快速走下坡路。现在，Borland 在 Dale Fuller 的主掌之下好不容易地回稳，并且在 2001/2002 年全世界不景气之中仍然呈现稳健的成长，算是相当难能可贵的。Dale Fuller 虽然不是 Borland 的创始人，但是似乎对于经营 Borland 真的发生了兴趣，愿意管理 Borland，也愿意投入心力规划 Borland 的产品线，是相当认真的 CEO。Borland 在经过了多年之后终于找到了一位不错的 CEO，希望 Dale Fuller 能够以公司创始人般的热忱，带领 Borland 再创另外一个高峰。

当然，公司创始人是否仍然是软件公司的重要人物不是判断一间软件公司唯一的指针。如果软件公司能够适当地吸引专业的管理人才来改善公司的体制是很好的事情，因为这样的公司有可能结合创新的理想，对产品的坚持以及有效率的专业管理，这是对公司以及使用者最好的保障。没有了创始人的存在，软件公司总是少了一股对于理想的坚持，不是吗？

Borland Conference

Borland Conference 对于 Borland 的使用者来说是非常重要的事件。就像回教徒一生一定要去麦加朝圣一样，Borland Conference 也是许多 Borland 开发者想要参加的重要技术研讨会。Borland Conference 已经有了相当久的历史，在美国也算是评价很高的技术研讨会，每年几乎都会被专业媒体评选为最值得参加的 Conference 之一。

我一直都非常喜欢参加 Borland Conference，因为这绝对是超值的盛会。Borland Conference 最近几年都维持一个惯例，那就是一年在美国东海岸举行，下一年在美国西海岸揭幕。我曾很幸运地参加过 3 次 Borland Conference，分别是 1995、1998 和 1999，每一次参加都收获满满。参加 Borland Conference 除了可以聆听许多精彩的 Seminar 之外，也可以遇见许多信息界重要的人物，例如我就在 Borland Conference 中见过 Bruce Eckcle、Martin Fowler 和 Peter Codd。当然，也可以和 Delphi、C++Builder 和 JBuilder 的开发团队见面而且和他们交谈、互动。通过和这些 R&D 研发小组见面，我也认识了许多 Borland R&D 中重要的人物，Chuck、Danny 和 Blake 等。也只有在 Borland Conference 中才有机会、有比较多的时间和他们交谈，否则就必须飞到 Borland 总部 Scott Valley 才能见到他们了。

早期的 BorCon 是比较偏向开发工具的 Seminar，每一个开发工具各自提供多场的 Seminar 让参加者选择。因此，在 BorCon 上很容易就可以从参加者的人数和反映来了解 Borland 每一个开发工具产品在市场上的表现。C/C++的参加者在较早的 BorCon 中占据了大部分的 Seminar，每一场 C/C++ Seminar 都是挤得水泄不通。但是在 1995 年 Borland 推出了 Delphi 之后，每一年参加 BorCon 的人中 Delphi 的使用者占据了愈来愈多的比例，而 C/C++的参加人数反而快速地下降。一直到了最近几年，Delphi 的人数还是 BorCon 中

占有最大比例的群组。但是，在 BorCon 的参加者中，Java 的开发者也占有愈来愈高的比例，由此可见开发工具和语言使用的趋势。

在 Borland 陆续取得了组件和中间件技术、推出 CORBA、EJB 产品之后，最近数年的 BorCon 更是呈现了多元化的发展。BorCon 中的 Seminar 愈来愈多，同时也有许多的 Seminar 是围绕着 COM+、CORBA、EJB 以及软件设计、开发和管理的主题。在早期参加 BorCon 时，我大都选择参加开发工具和技术的 Seminar。但是到了后期，却非常喜欢参加软件设计和架构的 Seminar，因为在这些 Seminar 中可以听到非常多的、宝贵的知识和经验，这是很难在其他地方能听到的。

我喜欢收集 BorCon 的 T-Shirt，不但免费、好穿，更有纪念价值。现在我仍然拥有一件 1995 年的、紫色的 BorCon T-Shirt，夏天时穿在身上好不神气。每一届 BorCon 都有专门设计的 Logo 和代表本次 BorCon 的 T-Shirt。下面是我收集的一些 BorCon Logo 和参加 BorCon 的小花絮，与读者一起分享。

1998 年，我参加了在丹佛举办的 BorCon。不过在丹佛实在有些无聊，除了白天的 Seminar 之外，晚上不知道做些什么。丹佛的 BorCon 结束之后，Borland 又另外举行了内部工程师训练，因此我总共在丹佛呆了一个多星期，都快发霉了。好在是和朋友一起参加的，彼此间还有个伴。另外也巧遇了几个从台湾去的人，居然其中还有相识的，感觉特别温暖。

费城的 BorCon 是非常令人难忘的。我和李匡正(Tom Lee)在赴会途中被整惨了。先是从台湾坐飞机到 Newark 机场，因为空中交通拥塞，飞费城的飞机在延滞了 4 个多钟头之后才取消，我只好和 Tom Lee 从 Newark 坐巴士，这样又花了 4 个钟头才到费城。当时香港和新加坡的同事还在苦等我和 Tom Lee，很久未到，以为我们出事了，吓出了一身冷汗。

但在费城参加 BorCon 时却很愉快。因为一行中有高官相随，所以吃了费城最好的牛排，也抽了点时间游览费城市区。但是从费城回台湾又是噩梦一场。因为从费城到 LA 的班机延误，所以当我和 Tom Lee 到了 LA 机场之后，回台湾的班机已经出发了，而且随后的一班飞机也没有空位。害得我和 Tom 在 LA 机场空等了 12 个多钟头，才搭上回台湾的飞机。更惨的是行李却已经直接 Check-in，而我的长袖衣物都在 check-in 的行李中，害得我在 LA 机场差点被冻僵。

我很想参加 2002 的这次 BorCon，因为很想看看 Anaheim 市的模样。当然，这也是因为 Anaheim 有一个 Anaheim 天使队，Tom Lee 很喜欢这支球队。不过，由于当时我正忙于工作，无法分身参加，实在遗憾。2003 年将是 BorCon 20 周年的庆典。听说这次 Borland 将要隆重举办，提供最丰富的 BorCon。我也想参加不可错过的 BorCon 2003，拿几件 T-Shirt 作为永远的纪念。想想，一个技术 Conference 能够举办 20 年，可真是不容易，不知道参加的时候，是否能够有机会在异乡巧遇我的读者呢？

并购、自保和集团战

商场真是诡谲多变，没有人知道会发生什么事情。但是，2002 年底，数个令人震惊的软件公司并购案可能会大大地影响 2003 年软件公司的竞争，也有可能影响到我们每一个人的饭碗。

200l/2002 是 Borland 绝处逢生、营运蒸蒸日上的时段。Borland 靠着在 Java 开发工具市场所向披靡的机会连续 11 季产生盈余，并且在大多数软件公司赔钱的情形中快速地累积了大量的现金。虽然就公司营运的角度来看，拥有大量未使用的现金不见得是好事情。但是，在这段不景气的时间中，"Cash is King"，拥有现金就是拥有力量。在蓄积了强大的能量，并且思索了如何在 Java 和.NET 平台的竞争下找出一个康庄大道后，Borland 终于在 2002 年底出手了。

首先，Borland 确定了在.NET 上的开发之路，誓言成为除 Microsoft 之外最好的.NET 开发工具和企业解决方案提供厂商。此外，Borland 也决定在 Java 战线上往上再提升一个层次，拉大和对手的竞争距离，以确保 Borland 在 Java 平台的优势。

由于未来软件的竞争势必是 Java 和.NET 两大平台主宰，因此，如何在两大平台保持竞争力，是每一个软件厂商都必须严肃面对的。坦白地说，别说是 Borland，现在已经没有任何公司能够再建立一个新的平台同 SUN 的 Java 和 Microsoft 的.NET 竞争了。所有的软件厂商都必须在这两个平台下提供最好的工具、技术和解决方案，软件厂商如何同手握平台主宰力量的 SUN/Microsoft 竞争，这便是每一个软件厂商能否生存下去的重要话题。

对于 Borland 来说，既然无法取得系统平台的优势，而单靠工具以"点"为单位作战的时代也已经过去，如何进行"面"的全面备战便是 Borland 必须思考的战略问题。还好，Borland 还是有一些聪明的人，想到了既然如果无法争取到系统平台的优势，那么为什么不在系统平台上提供一个"中立的应用平台"呢？这真是一个好想法，并且也值得一试。

所谓"应用平台"，是指 Borland 要在 Java 和.NET 上提供全方位的软件解决方案。这也就是说，Borland 除了以往的开发工具、中间件和 J2EE 之外，将提供完整的软件供应链。在 2002 年的产品线中，要想实现软件供应链，Borland 仍然有很大的缺陷。例如 Borland 几乎只有开发工具和后端的中间件，缺少了效率调整、整合测试、Modeling 工具、自动测试等软件和工具，更不用说许多在中间件方面的服务，例如 Messaging 服务、Transaction 服务等。这些工具都是小规模的软件，因此 Borland 面临了是需要自己开发还是通过其他途径来取得所缺少的软件的抉择。

其实，Borland 面临的选择已经不多而且很明显了。如果 Borland 决定自己开发从未进入过的软件市场，那不仅需要花费长期的开发时间，而且在开发出来之后，又必须面对市场已经存在的竞争对手，胜算不大且缓不济急，等到 Borland 开发了新的软件之后可能已经来不及。因此，Borland 决定通过现金并购的方式来取得这方面软件，所以展开了稍后的一系列 Borland 大并购行动，继而也震惊了软件界，引发了另一波软件公司并购的热潮。

并购行动的展开

Borland 出击的第 1 步，便是根据 Blake Stone 的建议，并购 VMGEAR 公司的 OptimizeIt 产品线。由于 JBuilder 已经是 Java 开发工具市场的老大，许多程序员经常会询问为什么 Borland 没有提供 Java 效率调整方面的工具，也都非常希望 Borland 能够为 Java 程序员提供这样的工具。当时，市场上 Java 效率调整工具主要是由 Sitraka 的 JProbe 和 VMGEAR 的 OptimizeIt 在进行激烈的竞争。在 Blake 看好 OptimizeIt 的情形下，Borland 很快就决定并购 VMGEAR，因为 OptimizeIt 不但可以让 JBuilder 的战斗力更为坚强，还能够帮助 Borland 填补欲形成的软件供应链的缺陷。

2002 年 1 月 22 日，Borland 以现金快速收购了 VMGEAR，很快成为 Java 效率调整工具的领

导者，OptimizeIt 也很快整合到 JBuilder 中并且扩充功能，增加了 OptimizeIt Suite 这个新的产品。Borland 并购 VMGEAR 并且在很短的时间内便推出新的 OptimizeIt 产品，可见这次是玩真的，而不像以前往往在并购了新的公司和软件之后便放在那里不闻不问。

在购得了 VMGEAR 之后不久，同年的 10 月 9 日，Borland 以更大的动作收购了闻名的团队开发以及软件管理公司 Starbase。这个收购行动当时出乎许多人和软件公司的意料。因为在 2000 年，Starbase 公司的市值还超过 Borland，其股票的价格也远远高出 Borland

的股票价格。没有想到两年之后，Borland 居然有能力并购 Starbase。由此可见两年来 Borland 和许多软件公司势力的消长。Borland 购得 Starbase 公司之后，意欲提供软件应用平台的计划也隐约可见了。

在 Borland 当时的计划中，软件应用平台应该包含需求分析工具、分析和设计工具、开发工具、测试工具以及执行应用软件的服务器工具。Borland 购得了 Starbase 之后，就拥有了 Starbase 的团队开发工具以及软件需求分析和管理工具，再加上从 VMGEAR 取得的效率调整工具、Borland 自己的开发工具以及 CORBA/J2EE 服务器，距离 Borland 想要形成的软件应用平台仍然缺少重量级的 Modeling 工具。

其实，从 Borland 开发工具的演进过程中，已经可以发现若干 Borland 欲往 Modeling 方向发展的蛛丝马迹。先从 JBuilder 开始说起，在 JBuilder 6 中，Borland 已经开始想为 JBuilder 加入 Modeling 的功能，随后出现在 JBuilder 中的 Refactoring 和 Class Diagram 的功能就是一个证明。

虽然急着往 Modeling 的方向开发，但是 Borland 也知道自己开发这方面的软件将是旷日费时的。因此，Borland 决定使用并购的方式快速取得这方面的技术和产品。放眼望去，Modeling 工具市场几乎只剩下两强相争的局面，那就是 Rational 和 TogetherSoft。由于 Rational 的市值远比 Borland 大了许多，虽然 Borland 已经和 Rational 有商业的合作，但是 Borland 仍然没有可能并购 Rational。既然如此，那剩下的答案就非常清楚了。

在 Borland 完成了 VMGEAR 和 Starbase 的并购之后，Borland 的软件应用平台几乎已经完备，剩下的缺角就是设计和分析工具了。

虽然 TogetherSoft 并不像 Rational 一样是 Modeling 工具市场的第一，也不像 Rational 拥有比较完整的产品线，但是 TogetherSoft 的 Modeling 工具在 Java/C/C++市场享有盛名，其软件功能和图形使用界面远远超过了 Rational 的软件。就软件品质来说，TogetherSoft 已经超越 Rational 甚多。只是 TogetherSoft 没有 Rational 的三位知名大师而已。

在 Borland 决定和 TogetherSoft 合作之后，TogetherSoft 也非常欣然地接受了 Borland 的提议，因为 TogetherSoft 正想往开发工具市场发展，以补足 TogetherSoft 没有适当开发工具来结合其 Modeling 工具的遗憾，这也是为什么 TogetherSoft 在早一步从 WebGain 购买 Visual Café权利的原因。现在，Borland 拥有最佳的开发工具，再结合 TogetherSoft 的 Modeling 工具，两家公司有机会共创双赢、打败 Rational 而成为盟主。于是 TogetherSoft 很快就答应了 Borland 的建议，同意和 Borland 合并。世事真是难料，没有想到当初 Visual Café和 JBuilder 恶斗数年之后，竟然由 Borland 取得了 Visual Café，也让 Visual Café最后在 Borland 的手中成为历史。

正是因为 Borland 并购 TogetherSoft，成了压垮骆驼的最后一根稻草，所以在消息宣布之后，立刻引起了许多软件公司的震惊和不安，这也激活了软件界的并购风暴。我当时的预测就是 Rational 将首当其冲。

并购的涟漪效果

Borland 宣布并购 TogetherSoft，立刻引起了软件界的轩然大波。由于 Borland 和 TogetherSoft 的合作，对于 Rational 产生了巨大的影响，而 Borland 又和 Rational 有软件合作，因此为了缓和对 Rational 的冲击，Borland 便下令公司内的人必须对这件事情封口。当时我就认为 Rational 也不是傻瓜，他们一定会了解事情的严重性，即使 Borland 不提，Rational 也一定会有动作。果然在不久之后，Rational 便通知 Borland，结束和 Borland 在 Java Enterprise Studio 以及 Windows Enterprise Studio 的合作，反将了 Borland 一军。当然这也正式代表 Borland 终将进入 Modeling 市场的大战。

Rational 和 Borland 的战事尚未真正开火，就被"螳螂捕蝉，黄雀在后"的 IBM 盯上了，IBM 开始正式向 Rational 下手。为什么 IBM 会找上 Rational 呢？这要从 IBM 和 BEA 之间愈
演愈烈的 EJB 服务器战争说起。
由于 IBM 的 WebSphere 和 BEA 的 WebLogic 已经进入最终的市场第一位争夺战，两方人马都是无所不用其极地想要干掉对方。不过，由于 EJB 服务器的市场已经进入成熟的阶段，现在光靠 EJB 服务器核心已经无法作为胜出的筹码了。这也是为什么 WebSphere 和 WebLogic 都开始加入其他的辅助功能，例如 Portal 服务、管理服务等，以求能够压过对手。不过，当大部分的软件服务都被加入之后，剩下来的当然就是从整合开发工具以及分析/设计软件上动脑筋了。
这也是为什么当初 BEA 会和 WebGain 合作而 IBM 也不愿意放弃 VisualAge For Java 的原因。即使 Visual Café和 VisualAge For Java 已经无法在 Java 开发工具市场成为第一位的工具，但由于使用 EJB 技术的企业愈来愈多，也有愈来愈多的企业要求结合 Java 开发工具和 Modeling 工具以便开发大型的 Java 以及 J2EE 应用系统。IBM 很显然也注意到了这个市场和需求，于是在 Borland 并购了 TogetherSoft、Rational 感觉到巨大压力的时候，立刻找上 Rational。由于 IBM 出手阔绰，再加上 Rational 自知要独自面对 Borland 和 TogetherSoft 的联军没有多大胜算，因此也很快答应了 IBM 的条件，正式由 IBM 接收了 Rational。
在 IBM 确定并购 Rational 之后，这股软件公司之间重量级的并购潮不但没有结束，反而更为暗潮汹涌，因为 IBM 并购了 Rational 之后，开始对 BEA 和 Microsoft 产生更大的影响。IBM 在取得了 Modeling 工具之后就在 EJB 服务器中取得了整合的优势，对于 BEA 将有更强大的攻击力。而 BEA 在原本已经逐渐落居下风的 EJB 战役中如果还面临 IBM 整合 Modeling 的攻势，那么情势必将更为恶劣。因此，当时我认为 BEA 将是 IBM 这桩并购案最大的受害者。果然之后不久，许多专业媒体都评论了 BEA 不利的局势，预言 BEA 最强的支持者将会是 Borland，甚至许多人也传出 BEA 将并购 Borland 的消息。对于这个传言，BEA 的响应似乎是正面的，因为现在 BEA 已经和 Borland 有所合作，而 BEA 和 Borland 的产品线也非常互补，没有什么严重的冲突。不过，我个人还是希望 Borland 能够维护独立的软件公司。
另外一个受影响的软件厂商则是 Microsoft。Microsoft 在以前早就和 Rational 有合作，不过 Microsoft 还是那个调调，在自己没有 Modeling 工具之前希望和 Rational 合作，但是一旦有了类似的工具之后(即 Visio)，就停止了和 Rational 的合作，这种做法类似当初 Microsoft 和 Sybase 的合作关系。不过，在 IBM 取得了 Rational 之后情势又不同了，因为现在 IBM 拥有了非常完整的产品线，IBM 可使用这条产品线，以强大又完整的 J2EE 架构正面攻击 Microsoft 的.NET。因此，后来也传出了 Microsoft 有意并购 Borland 以取得 Modeling 工具的风声。如此一来，Microsoft 不但可以在．NET 上提供全世界最好的开发工具、Modeling 工具，又能够取得.NET 需要的组件模型，即 CORBA.NET，以对抗 EJB，可以说是一举数得。不过在开发工具方面，Microsoft 和 Borland 有严重的重复，又可能会引起独占市场的疑虑，因此我个人认为，这是不太可能的结合，真的要说，那么双 B(Borland 和 BEA)的组合反而比较可能。
不管未来的发展如何，应该发生在 2003 年的大战终于在 2002 年末正式提前开打，Borland 也即将进入另外一个新的转变。
^v^v^v^v^v^v^v^v^v^v
第九章　软件技术和平台的大竞赛
2002 年的 2 月，Microsoft 终于推出了.NET，也击败了许多爱看 Vaporware 好戏的人。

.NET 的出现，代表了窗口平台的软件开发将进入一个新的领域，对于窗口平台上开发工具厂商也有深远的影响，因为.NET 是有史以来变动最大的窗口平台。第一次，Microsoft 把窗口变成一个虚拟执行环境，通过 SOAP/Web Service 技术，把窗口和各种行动以及电子装置整合在一起，提供了下一代的整合虚拟数字世界。这个影响是深远的，它不但冲击了操作系统，影响了下一代窗口操作系统的发展方向，也改变了开发工具在这个虚拟执行环境中的角色。开发工具厂商必须重新定义、定位开发工具在.NET 中扮演的角色以及未来的发展趋势。

Windows3.0 和 3.1 曾为窗口平台带来了最辉煌的时代，造就了 C/C++四大天王(Borland、Symantec、Watcom 和 Microsoft)、C/S 双雄(PowerBuilder 和 Gupta)、COBOL 两大家(RM COBOL 和 Acu COBOL)以及无数充满活力的开发工具厂商。图形用户界面的盛行也让各种 Framework 充斥于市。随着 C/C++语言的流行，其他语言很快便退居 2 线。MFC 的出现让 Symantec 和 Watcom 退出市场，VB 和 Delphi 的快速成长则让 C/S 双雄饮恨不及。开发工具市场在 Windows 98 之后有了快速而巨大的变化。最后，除了 Microsoft 和 Borland 等少数厂商之外，大部分的开发工具厂商都逐渐退出了这个竞争最激烈、门槛最高的市场。随着.NET 的推出，Microsoft 又把竞争门槛再度拉高。这次 Microsoft 瞄准的是企业信息市场以及 Java 平台，程序语言和开发工具的竞争不再是 Microsoft 关心的重点，Microsoft 的重点是如何在窗口平台提供类似 Java 已经发展将近 10 年的计算环境。不过，这个目标却苦了开发工具厂商，因为他们必须面对新的虚拟执行平台、新的编译技术和新的 Framework。更糟糕的是，开发工具厂商必须在.NET 中找到一条新的生存之道，由于.NET 包含了：

- 一个虚拟执行环境--Common Language Runtime(CLR)
- 一个庞大且完善的 Framework--.NET Framework

因此开发工具厂商必须在这两者以及两者交错产生的软件元素中找到新的技术、新的应用和新的利基，才能够持续在.NET 中生存。更麻烦的是，Microsoft 已经提供了一个开发工具范例--Visual Studio.NET，它比当初 SUN 的失败作品 Java Workshop 好得太多。这不但证明了 Microsoft 是一个比 SUN 更精于开发工具的厂商，而且，其他的开发工具厂商要想凸显其产品更在 Visual Studio.NET 之上，也将是一件更艰苦的工作。也许.NET 的出现会加速淘汰更多的开发工具厂商，让.NET 平台上的开发工具厂商更纯化，最后只剩下最具实力的少数厂商。目前各家开发工具厂商如何适应.NET 的冲击？它们会提出什么新的软件技术同 Microsoft 以及其他厂商竞争？谁会是最后的胜利者？这都将是非常有趣的事情，仔细观察并分析这些问题，或许从中也能学到许多的宝贵经验。

.NET 和 Java 的发展过程提供了许多耐人寻味的东西。有趣的是，.NET 和 Java 虽然在现在以及未来的发展有许多类似的表现，但是这两个平台的骨子里却有一些重要的差异。其中最明显的，就是 JVM 和 CLR 分别如何执行最终的应用程序以及单一语言对语言中立的考验。除此之外，Java 和.NET 对于中间件组件技术的对抗也是最激烈的一环，因为中间件技术将是未来主宰系统架构的主要因素，SUN 和 Microsoft 都希望自己力推的平台成为新的应用标准。

Java 和.NET 的竞争，虽然从虚拟执行环境、程序语言、Framework 一直延续到最新的软件技术--SOAP/Web Service 和数据存取技术，但是组件模型仍然是其中最重要的，因为它代表的目标市场--企业信息领域，才是这两家必争之地。Java 和.NET 的组件模型是程序语言设计之奇、Design Pattern 之美、数据存取架构之广以及设计构想之深的结晶。组件模型不但是 SUN 和 Microsoft 市场的关键，也代表了两家领导厂商的软件技术实力以及系统架构的思想逻辑。因此在讨论 Java 和.NET 竞争时，充分了解 J2EE 以

及.NET 在组件模型方面的发展是很重要的。通过了解这两个阵营在组件技术的竞争，我们也可以很容易掌握未来 Java 和.NET 的发展趋势。因此随后的文章将从 Microsoft 和 SUN 发展组件模型的历史和趋势开始讨论，让读者了解 Java 和.NET 中位居关键地位的技术演进，以及组件模型如何影响 Java 和.NET 的未来走向。在本文后半部分，我们将讨论.NET 对于窗口平台中开发工具厂商的影响，以及未来开发工具的适应和发展趋势。

Microsoft 的 COM 组件模型

Microsoft 的 COM 组件模型一直在很稳定的发展中。舍弃繁杂的 OLE 细节之后，COM 才真正地奠定了 Windows 组件模型的核心，开始可以提供制作企业逻辑对象的能力。DCOM 开始提供远程访问和分布式计算以及对象回收的机制，让 COM 组件模型能够提供企业级计算的能力。不过在 DCOM 的时代，客户端仍然是通过 Proxy/Stub 直接和 COM 对象互动，还未到达像 EJB 组件模型那样由虚拟服务器控管以提供系统服务等功能。但是，Microsoft 很快在 MTS 1.0 中正式加入了这个功能，至此，COM 组件模型能够顺利地加入企业核心服务，例如 Object Pooling、Role-Based 安全权限和交易管理等功能。严格地说，在 MTS 出来之后，COM 组件模型才有资格成为关键性系统的核心组件模型。也因为 MTS，才有后来的 Microsoft DNA 架构。在 Windows 2000 中，MTS 正式成熟演进到 COM+1.0，除了把 MTS 调整得和操作系统更契合之外，最重要的进步是大幅提升了执行效率，因此，Microsoft 的 TCPP 数据大都是以 COM+加 VC++撰写的。

在不久前推出的 Windows XP 中，COM+又进步到 1.5 版。在 COM+1.5 版中，Microsoft 对 COM+进行了许多改善，其中最重要的便是再次提升了 COM+的执行效率，让它比 COM+1.0 更快。此外，延展性也是 COM+1.5 的调校重点。Microsoft 为 COM+1.5 加入了 Partitioning 功能，企图让 COM+的 Application 能够在不同的 Container 服务器(DllHost.exe)中执行，提供对象并联的架构，以增加 COM+应用系统的延展性。不过，从 COM+1.5 目前实现的程度来看，这应该是初步的规划，未来应该还有很大的进步空间。

此外，COM+1.5 也加入了 Application Pooling 的机制，让程序员可以控制 COM+ Container 服务器执行的数目。当 Container 服务器的执行数达到右图中集区大小的数目之后，Windows 操作系统便会重复使用已经存在的 Container 服务器，而不会允许客户端继续建立新的 Container 服务器，如此一来，就不会让客户端启动太多的 Container 服务器以拖垮 Windows 操作系统。

而应用程序回收(Application Recycling)功能则是 Microsoft 为了克服 COM+内存泄漏(Memory Leak)问题加入的。在 COM+1.5 中，程序员可以指定 COM+的 Application 在一定时间、或是在 COM+的 Application 的内存使用到达一定的数量、或是被调用了一定的次数、或是被启动了一定的次数之后就重新启动 COM+的 Application。这样，可以让 Container 服务器结束运行，而操作系统则可以回收因为 COM+对象泄漏的内存。在 COM+尚未像 EJB 或是.NET 一样以虚拟执行环境进行 Garbage Collection 之前，这倒不失为一个好方法，因为 Windows 2000 和 XP 在进程安全控制方面有了大幅的精进。

此外，COM+1.5 的组件服务程序也允许程序员直接管理和设定旧的 COM/DCOM 组件，不需要再使用 DCOMCNFG.exe 程序。1.5 的组件服务程序整合了所有型态的 COM 组件，是相当不错的功能。

从 COM+1.5 的发展来看，其他的许多功能都属于小进步，未来 COM+的发展将会很小，进而 COM+会真正地转变成 Windows 的核心服务之一。未来 Windows 的组件模型应该是由.NET 的组件架构来接棒，因为 Microsoft 仍然需要一个提供虚拟执行环境的组件架构，以提供良好的 Garbage Collection 和 Partitioning 的功能，进一步和 EJB 竞争企业系统的延展性，而这个征兆可以从稍后讨论的.NET 发展看得出来。

SUN 的 EJB 组件模型

经过了将近 2 年的时间后，SUN 终于在最近推出了新一版的 EJB 2.0 功能规格，很快也被 BEA 和 Borland 的 BES 实现出来。SUN 在 EJB 2.0 中提出了许多先进而复杂的功能，目的是为了大幅强化 EJB 作为企业核心组件架构的本钱，以便在企业系统中和 Microsoft 下一代的.NET 组件竞争。

从 SUN 定义 EJB 的规范开始，就展现了其和 COM 不一样的观念。EJB 一开始就非常重视 Design Pattern 和组件种类，例如 Session Bean 和 Entity Bean 各自负责不同的角色，再借助于 Java 的 Garbage Collection，提供了成为企业信息系统组件必要的基础。但是，在 EJB 1.x 中，所有的 Bean Instance 之间的调用都是使用 Remote Interface 方式，因此在许多的应用方面付出了较大的开销，导致一些情况下执行效率不佳。在 EJB 1.x 中发展出了许多的 Design Pattern 来改善这种现象，例如鼓励使用 Coarse-Grained 对象，以减少网络 Round-Trip 和 Bean 之间的调用次数。

在 EJB 2.0 中，SUN 终于为改善这个问题而提出了 Local Interface。所谓 Local Interface，是指当 Bean Instance 在同一个 EJB Container 中时，EJB Container 可以使用 Local Interface 调用来代替 Remote Interface 调用，这样可以增加 3 倍以上的对象启动效率。另外，SUN 加入 Local Interface 功能的重要原因是为了支持 EJB 2.0 中大幅强化的 CMP(Container Managed Persistence)功能。

简单地说，EJB 2.0 中的 CMP 允许程序员使用 EJB Query Language 来定义 Bean 之间的关系。只要程序员使用 EJB QL 定义了一个 Bean 和另外一个 Bean 的关系(Relationship)，那客户端便可以在存取了主要 Bean 之后，再通过 Bean 定义的 Finder 或是 Selector 方法取得所有的从属 Bean。如果读者不太了解这个意思，可以参考下面的示意图。在客户端建立主 CMP 之后，可以通过主 CMP 的 Finder 或是 Selector 方法取得所有的从属 Bean，此时，主 CMP 便可以通过 EJB QL 向数据源查询所有相关的数据，再由 EJB Container 根据查询的数据自动建立从属 Bean、并且和主 CMP 建立关联。如此一来，2.0 的 CMP 可以免除程序员自行撰写存取数据和建立 CMP 的程序代码的麻烦，并且允许 EJB Container 使用最佳化的方式从数据源存取数据和建立 CMP。为了增加这个过程的执行效率，EJB 2.0 的功能规范要求这种 Bean 必须支持 Local Interface，当然，这也代表着这些会建立关系的 Bean 必须执行在一个相同的 EJB Container 中。EJB 2.0 的 CMP 增加了功能和效率，但是也增加了 Bean 之间相互依靠的关系，这会影响 EJB 程序员在设计系统时的布局。此外，EJB 2.0 的 CMP 虽然提供了这么强大的功能，但这也是不同厂商实现的 EJB 服务器执行效率有差距的地方。不同 EJB 服务器使用的实现方法的不同，会大大影响执行效率。这就是为什么 SUN 定义了 ECperf 标准来检验和评比 EJB 服务器的真正执行效率的原因，这稍后会谈到。

因此，在 EJB 2.0 功能规格中，SUN 定义了数个机制来增加 EJB 服务器的执行效率，这在 EJB 的架构已经很完整的情形下是很自然的下一步。当然，除了上述的功能外，EJB 2.0 功能规格也增加了 MDB(Message Driven-Bean)，MDB 可以让程序员使用异步的方式传送信息，事实上把原本 JMS 的功能加入到 EJB 的功能规格中，是为了对抗 Microsoft 把 MSMQ 整合进 COM+。请看 EJB 进化的示意图，我认为 EJB 2.0 最重要的进步便是执行效

率、OR Mapping 以及 EJB 的查询语言。EJB 的查询语言开启了未来和 JDO(Java Data Object)的整合，目的是和 Microsoft 在.NET 中定义的 OPath 和数据对象相互竞争，这在稍后也会再说明。至于 OR Mapping，则是一个非常复杂的机制。它规范了 Bean Instance 和数据源之间的关系，这个标准可能会让许多小的 EJB 服务器厂商退出 EJB 市场，或是无法完整地支持 EJB 2.0 的功能规格。

再看看 Local Interface 的意义。在 EJB 1.x 中，客户端调用 Bean Instance 时，在 Container 中 Bean 和 Bean 之间都是使用 Remote Interface 的方式进行调用，如下图所示：
事实上，图中显示的启动模式是非常浪费资源的，因为图中的 Bean 都属于同一个 Container 之中，为什么要使用缓慢的 Remote 调用模式呢？因此在 EJB 2.0 中定义了 Local Interface。如下图，现在只有在跨越网络或是跨越不同的 Container 时才需要使用 Remote 调用模式，这大大地改善了使用的资源和调用效率。
更好的是在 EJB 2.0 中，一个 Bean 可以同时定义 Remote Interface 和 Local Interface。如此一来，Bean 的使用者和组合者(Assembler)可以更有弹性地分发和部署 EJB Bean。
在 EJB 2.0 中，Bean 只要从 EJBLocalObject 继承下来，就可以拥有 Local Interface 的功能。例如程序员可以用下面的程序代码来提供 Local Interface 的功能。本质上，实现和定义 Local Interface 的方式和原本的 Remote Interface 非常类似，因此 EJB 的程序员可以很自然地学会这个新的 EJB 功能。
public interface YourobjectClass extends EJBLocalObject
public interface YourobjectClass extends EJBLocalHome
了解了 EJB 2.0 增加的功能之后，现在就可以回到前面朋友询问我的问题了，为什么在 EJB 中没有看到任何像 COM 一样的线程模型之类呢？事实上这很简单，因为 EJB 是一个标准功能规格，并不包含如何实现的细节，在一般的 EJB 书籍中当然看不到类似的东西。而且，COM 之所以有这么复杂的各种线程模型，是因为 COM 发展的包袱以及历史的因素所造成的。不过，这并不代表在 EJB 中没有线程模型的问题，因为 EJB 厂商如何实现 EJB 功能规格会深深地影响 EJB 服务器的效率。因此，线程模型反而是 EJB 程序员应该知道的东西，只是依据不同的厂商而有不同的结果，不像 COM 功能规格是由 Microsoft 定义的，也是由 Microsoft 实现的，因此会有一致的执行行为。
EJB 的线程模型应该是使用 Object Per Client 的模型。这个意思是说，EJB Container 会为每一个请求的客户端建立一个独立的 Bean 服务。因此，如果 EJB 厂商没有特别进行最佳化的工作，那 EJB 使用的模型应该是类似 COM 中的 STA，也就是说，一次只有一个 Worker 线程在 Bean Instance 中执行。下图就显示了这个架构，对每一个客户端就启动一对 Worker Thread/Bean Instance。
上图叙述的是正常的情形，那如果让两个客户端同时存取一个 Bean Instance 时，会发生什么情况呢？下图就显示了这个架构。在这个情形中，如果有两个客户端要同时存取 Bean Instance，那 EJB Container 如何控制呢？在一般的 EJB 书籍中，似乎也没有看到和同步处理有关的范例，难道说，可以不进行任何的处理就让两个客户端同时存取吗？这当然不会，因为此时 EJB Container 就会进行管理，以 STA 的模式控制同步存取，因此客户端的存取必须依序(排队)来调用 Bean Instance。
这个情形也可以直接从 Bean 的实现程序代码中看出，例如下面的程序代码是 EJB 的标准范例 Entity Bean 的实现程序代码。请注意，在这个 Bean 类别中定义了数个 private 变量，并且在 Bean 的方法中直接存取和处理这些 private 变量，完全不需要考虑任何的同步处理机制，这就是因为 EJB Container 一般就是使用 Object Per Client 的模型以及类似 COM 的 STA 的线程控制模型。
这只是一般的 EJB Container 可能会使用的模型，但有一些 EJB 服务器提供了最佳化的机制，可能会提供更为有效率的方式。下面的表格列出了 COM/COM+和 EJB 在线程模型方面的比较：
因为不同的应用程序服务器厂商实现而不同
读者必须注意的是，上表并不代表 COM+是比较好的，只能说 COM+提供了较多的选择，可以让有经验的程序员调整执行效率。但是，相对地也让情形复杂了许多，而且 COM+

的 MTA 线程模型也不容易实现。
正由于 EJB 功能规格会因为不同的 EJB 厂商实现而有不同，因此，除了前面提到的 EJB 2.0 中 CMP 和 OR Mapping 会影响 EJB 服务器的执行效率之外，如果再结合线程模型和对象建立的技术，那下面列出的问题是影响执行效率的重要因素：
●如何实现和控制 Worker Thread。事实上这就是 EJB Server 中 Thread Pooling 的机制
●如何实现和控制 EJB Bean Instance。这就是 EJB Server 中 Object Pooling 的机制
为了让 EJB 服务器有公平的效率比较基础，SUN 定义了 ECperf 标准让使用者能够用来评量各家 EJB 服务器的执行效率，以避免各说各话的情形。从这一点也可以看出，SUN 现在开始注重 EJB 服务器的执行效率因素了。
为什么我说线程模型会因为不同的 EJB 服务器而有不同呢？现在让我们以实例来看看 EJB 服务器的行为。下图是我使用 4 个 Delphi 建立的客户端应用程序，并且使用 SIDL 技术来调用 Borland Application Server-BAS 中的一个 Stateless Session Bean 的结果。
从图中可以明显地看到，即使是在有 4 个客户端的情形中，BAS 仍然使用了 MTA 模式，只建立一个 Stateless Session Bean，并且让 4 个 Worker 线程同时存取，因此执行效率非常高，使用的内存资源也非常少。
而下图则使用 4 个 Delphi 客户端应用程序调用 Stateless COM+对象(使用 Both 线程模型)，从图中可以看到，COM+使用 Object Per Client 的模式，建立了 4 个 COM+对象服务 4 个客户端，虽然执行效率也非常高，但是使用的资源稍比 BAS 多。
接下来，再让我们讨论一下未来 Microsoft 的组件模型以及 SUN 的组件模型的演进趋势。
Data Access Technology
在未来，Microsoft 和 SUN 的组件模型大概都会强调数据存取的技术，因为从前面讨论的 EJB 2.0 CMP 的内容中我们可以知道，现在 SUN 已经在为对象和数据之间建立连接的技术了，而未来的 JDO 技术将进一步紧密结合数据对象的观念，让程序员面对的所有东西都是对象，不再有数据和对象不一样的观念和使用方式。
不过别以为 Microsoft 只会呆在原地，在 PDC 2002 中 Microsoft 已经宣示了未来 ADO.NET 的发展方向。ADO.NET 未来将会结合数据和组件的观念，让.NET 的程序员以对象的观念来代表数据，就像 EJB 中的 CMP/BMP 一样。如此一来，.NET 的程序员可以像 EJB 一样声明代表数据源中数据的数据类型，并且使用以 XML 格式封装的数据对映叙述器(Data Descriptor)来连接数据对象和数据源之中的数据。如此一来，.NET 的组件模型也提升到和 EJB 2.0 加上未来 JDO 一样的层次。
例如程序员可以定义如下的数据类型：

```
public abstract class Customer {
   public abstract string Name{get; set;}
   [Link(Account)] public abstract IList Accounts {get;}
}
public abstract class Account {
   public abstract float Amount {get; set;}
   public void CalculateTotal() {
      // business logic
   }
}
```

并且定义上述 Customer 和 Account 之间的连接关系，这和 EJB2.0 中新的 CMP 功能一样，然后再定义如下的对象/数据对映器，把对象和数据源连接起来，请特别注意下面 relationship 的部分：

最后，程序员可以使用如下的形式通过数据对象存取数据，并且在数据对象之间自动形成关联的关系。这非常有威力，和 EJB/JDO 不相上下。事实上，ADO.NET 和 EJB/JDO 实现的观念和想法非常类似，这是巧合还是模仿呢？基本上可以说，这两大阵营都有互相参考对方技术的地方。

下图就是未来 ADO.NET 的数据对象架构，程序员只需要修改 Schema Mapper 就可以连接到不同的数据源，例如 MS SQL Server 或是 Oracle 等。

除了 ADO.NET 的数据对象外，Microsoft 也开始定义类似于 EJB QL 的对象查询语言，目前暂时称为 OPath。当然，我们可以进一步地讨论更为深入的组件技术问题，不过由于篇幅的限制，就让我们以后在专门讨论技术的书籍中继续说明好了。

下图很清楚地说明了 Microsoft 和 SUN 组件模型的发展趋势。从图中，我们几乎可以知道这两者非常类似，发展的方向也趋于一致。未来比较的因素可能是执行效率、延展性、能够执行的平台以及开发工具的支持程度和使用的方便性吧。

综合上述内容，从最近 Microsoft 的 COM+/.NET 的推出、SUN 的 EJB 2.0 功能规范的完成、以及中间件厂商实现的 EJB 应用程序服务器来看，Microsoft 似乎也已经开始采用类似 Java 的虚拟执行环境以及 EJB 的模型来重新塑造.NET 的组件模型了。COM+将逐渐退居幕后提供系统核心服务，甚至会慢慢地消失于未来.NET 的执行平台之中。不过由于.NET 的进入门槛不低，而且目前仍然有大量的原生 Windows 开发人员以及 Windows 应用程序，因此，这个从 COM 组件模型完整转换到.NET 的过程可能仍然需要数年之久，而 COM 在现在开始的数年内仍然是 Windows 平台上最重要的中间件技术。

据 Gartner Group 的调查和估计，在 2003 到 2004 年使用 EJB 技术开发的 Java 应用系统将占整个 Java 平台的 40％左右，这表示 EJB 技术已经获得了大型企业和专业软件厂商的认可，是企业级的组件模型。EJB 2.0 必须开始增加执行效率，故此加入了 Local Interface。此外延展性也成为 EJB 应用程序服务器的发展重点，因为 EJB 应用程序服务器势必将承载更多的存取，以担负起企业的关键应用。因此，EJB 厂商开始在 EJB 服务器中切割虚拟伺服环境，并且在每一个虚拟伺服环境中执行不同的软件。例如一个虚拟伺服环境负责执行 JSP/Servlet Container，而另外的虚拟伺服环境则执行 EJB Container 等，如下图所示。这样做的好处是不但每一个 Container 更安全，而且应用程序服务器的延展性将更为优秀，因为在多 CPU 的机器中可以分配专门的 CPU 给不同的 Container，并且在一个 EJB 服务器中可以同时执行多个 EJB Container。

这里有一个很有趣的区别，那就是由于 Microsoft 掌握了操作系统，Microsoft 可以尽量地把.NET 的虚拟执行环境移往操作系统的核心，提供更为良好的执行效率；但是由于提供 EJB 的厂商没有这项优势，因此必须以更好的实现方式来开发 EJB 应用程序服务器，这也是为什么 SUN 以 ECperf 这个标准来评定各家 EJB 应用程序服务器的执行效率的原因。但是从目前 EJB 服务器的实现观念和技术看，仍然是领先于 Microsoft 的.NET。

不过不要小看 Microsoft，虽然.NET 在 2002 年的第一季才推出，但是 Microsoft 已经在开发.NET 的第 2 个版本了，.NET 的发展步伐是很快速的。

中间件技术将会继续不断地发展下去，各种新的组件观念和实现技术也将持续地出现。组件模型技术和中间件已逐渐取代早期的程序语言和数据库服务器，成为现在信息架构的主导力量，Microsoft 和 SUN 都希望成为这个领域的领导者。不过谢谢信息市场的竞争力量，让这两家大厂都无法消灭对方，反而由于竞争的力量造成了组件模型不断地创新，使信息人员能够持续地使用新的、更好的、更成熟的中间件技术，来实现日趋复杂的信息系统，虽然这个学习的过程很辛苦，但这也是信息行业让人感觉到有趣味的地方，因为你不会总觉得工作是一成不变的。

只是现在 Web Service 的兴起让组件模型的界限开始显得模糊了，而 Web Service 也是

Microsoft.NET 和下一版 Java JDK 强调的重点功能。看起来，Web Service 技术将会开始把组件模型逐渐地转换为面向组件服务，让组件模型的决胜点从面向功能逐渐转向面向服务。以后哪一个组件模型能够提供企业级的服务模型，将会是决定系统使用的架构的关键点，而这个现象已经可以从一些中间件厂商最近的动作中隐约的看出。

.NET 对于开发工具厂商的影响

.NET 的推出，对于所有开发工具厂商而言都是一大挑战，这除了牵涉到技术层面之外，还包含了复杂的产品定位的问题。相对于当初 Windows 3.0/3.1 推出时各个开发工具厂商百家争鸣的盛况比起来，如今的.NET 平台就显得逊色了许多。当然这主要的原因在于.NET 中语言不再是重点，再加上语言可以内嵌在 Microsoft 的 Visual Studio.NET 中，这顿时让许多的开发工具厂商失去了定位以及竞争优势。如果开发工具厂商只是做一个语言的 Plug-In 到 Visual Studio.NET 中，那将很难生存下去。

对于像 Borland 的 Delphi、C++Builder 以及 Sybase 的 PowerBuilder 而言，如何在新的.NET 环境中保持竞争优势是很重要的问题。因为在.NET 中，应用程序执行环境、Common Language Runtime(CLR)以及.NET Framework 都是由 Microsoft 所掌握，其他工具厂商如何在 Microsoft 一手控制的环境中营造出竞争优势呢？另外在.NET 中，开发工具厂商必须把应用程序编译成 Common Intermediate Language(CIL)的格式，再由 JIT 编译器编译成原生机械码执行，如下图所示。

因此，如果开发工具厂商要在.NET 环境中继续提供竞争产品，那至少必须在下面的三个领域中找到答案，并且做出实际的解决方案：

- 编译器的竞争--如何把程序语言最正确且有效率地编译成 CIL
- .NET Framework 的竞争--如何在.NET Framework 上进行加值的工作，并且定位产品竞争力
- 开发工具本身功能集(Feature Set)的竞争

从编译器角度来说，由于.NET 的 CLR 内建的 Virtual Execution System(VES)支持一般的程序语言功能，同时又提供了丰富的对象模型支持能力，以提供面向对象语言对映到 CLR 的能力，因此.NET 可以说是 OOP-Friendly 的执行环境，这非常有助于面向对象程序语言在.NET 中实现，例如对 C/C++、Object Pascal 等真正的 OOP 来说是个好消息，而 Microsoft 的新语言 C#就是一个好的 OOP 实现范例。但是对于使用脚本语言作为骨架的开发工具(例如 PowerBuilder)来说，可能就需要花上许多的功夫重新规范，以便能够适当地使用 CLR 的特性。当然除了程序语言之外，如何开发出一个有效率的 CLR 编译器更是开发工具厂商需要费心的地方。

在 Framework 方面，Microsoft 的.NET Framework 摆明了要和 SUN 的 J2EE/J2SE/JEME 等竞争，而且花了许多的资源打造.NET Framework，力求能够提供给程序员最好的开发功能。但是，对于开发工具厂商来说则是有喜有忧。一方面，Microsoft 虽然提供了良好的.NET Framework，可以减少开发工具厂商需要花费的成本；但另一方面，开发 Framework 的权力掌握在 Microsoft 手中，特别是 Microsoft 也有 Visual Studio.NET 作为竞争产品，因此如何定位便成了重要的问题。就我的看法，如果开发工具厂商无法在.NET Framework 上进行增值的工作，那最后仍然难逃被淘汰的命运。

即使开发工具厂商能够克服前面讨论的两个问题，最后仍然要回到产品本身的竞争力上来。没有集成开发环境、组件架构、调试环境和高生产力，仍然无法和 Visual Studio .NET 竞争。开发工具厂商不但要像以往一样提供一个集成开发环境，甚至还必须做得比 Visual Studio.NET 更好、更具创意。这也不是一件容易的事情，因为这必须有突破性的想法。例如，其中的一种可能就是再把.NET 的通用性延伸，除了像.NET 不把语言的差异作为重点之外，也不把 CIL 产生的结果作为差异。由于 CIL 是一组标准的中介

信息，开发工具厂商可以继续把 CIL 转化为.NET、原生窗口应用程序、Linux 应用程序，甚至是移动设备上的程序代码，如下图所示。
如此一来，这种开发工具将更为广泛和实用，也是开发工具极好的竞争优势，特别是现在仍然有许多的软件厂商需要继续开发小而快的原生窗口应用程序。
Microsoft .NET 的出现不单对于 Microsoft 本身有重大的意义，对于窗口平台上所有开发工具厂商和 SUN 都有巨大的影响。开发工具厂商正面对着从 Windows 推出以来最严格的考验，这是一场生与死的竞争。对于 SUN 来说，.NET 代表的是 Microsoft 正式全面地向 Java 平台挑战，时间将决定 JVM 和 CLR 的胜负，而 Java 单一语言的通用性也将面临.NET 语言中立的考验。至于传统的窗口程序设计人员而言，也许正如"魔戒传奇"中的哈比人一样，明知前途坎坷，仍然必须选择走向严寒的雪山或是诡谲的地道，因为目的只有一个：在新一波的软件技术和平台中找到一条生存之路。
^v^v^v^v^v^v^v^v^v^v
第十章　令人焦虑的时代
"通向未来之路在哪里？"
时间进入 2000 年之后，许多事情变得似乎都不确定了，世界经济的走向和信息技术的趋势变得更令人困惑。在经过了 Internet/Intranet、Linux 和 Open Source 的洗礼之后，目前信息技术的发展似乎已经趋向多元化的状态。虽然许多的信息系统仍然在使用我们早已熟悉的技术(例如 Web 和主从架构)，但各种新的信息技术也在层出不穷地出现(例如 SOAP 和 Web Service)，再加上.NET 和 Java 两大平台之间逐渐升温的战火，让许多软件开发人员眼花缭乱，继而心生疑惑--"自己未来的前途到底在哪里？"
其实，信息人员产生这样的疑惑是很正常的。因为信息技术的发展到达了前所未有的阶段，不但各种程序语言之间掀起了混战，操作系统平台、虚拟执行平台、开发工具、组件模型等都兴起了热战。而虚拟执行平台让跨平台模糊了以往壁垒分明的开发领域，程序语言的多样化稀释了原本由数种语言瓜分天下的态势，而 Web 和多层架构又逐渐瓦解了传统的信息架构。这些信息技术的多元化发展，不但让传统的开发人员面临难以抉择的命运，虚拟平台、程序语言和信息架构等众多的组合变量，也让开发人员顿然之间感觉负担沉重，担心自己已经跟不上信息发展的快速脚步，那软件开发人员的未来到底在哪里呢？
信息技术多元化的发展
和许多人的工作一样，也许你还在使用 Delphi/C++Builder 开发主从架构或是 Web 或者一般的应用系统，又或许是使用 JBuilder 开发 Java 应用系统。总之，你可能已经熟知目前所使用的技术，并且大量地应用在日常的应用系统中。但是，身为软件开发人员，必须了解软件趋势的发展，必须随时注意新的软件技术，因为唯有不断地增加自己的附加价值，才能够在这个竞争激烈、演进快速的产业中生存下去。
其实从整个软件技术发展的趋势中，细心的软件人员已经能够看出未来的方向。在软件开发的过程中，每一个时代都有主导的软件技术在影响着当时的产品以及软件公司的兴衰。当然，能够掌握软件趋势的人或是公司也都获得了成功。从下图中我们可以看到，在不同年代中不同的信息技术掌握了当时的主宰力量。60/70 年代是由数据处理和程序语言独领风骚，到了 80 年代便由数据库当家作主，90 年代各种组件和中间件又主导了系统架构。
但是从 60/70/80/90 年代的软件技术来看，每一个时代都是由一个点的信息技术来主导。不过在 Internet/Intranet 时代，面向对象和 Modeling 等技术对于信息系统的影响愈来愈大，信息技术的演进逐渐从点形成了面，上图就显示了在 2001 年之后主要的软件力量来自平台的整合和竞争、以及全方位的软件技术。其实，作为软件人员，我

们也可以从自己的信息生涯中咀嚼出这个趋势，问题只是在于有没有花时间进行自我思考。

数年前的软件人员可能只需要了解程序语言即可，例如只需要会 C/C++就可以找到工作。那时数据库也几乎属于专门的技术领域，当时的 DBA 只要会管理数据库就行，因为还有许多专门写 SQL 的程序员。但随着时间的推移，软件人员开始需要同时会程序语言、SQL 以及管理数据库。接着又需要了解组件技术、Web 技术、终至面向对象和 Modeling 等技术。为什么对软件人员的要求会愈来愈高？这是因为整个软件的发展趋势正在走向信息技术整合的道路。

未来的软件趋势是走向软件和系统整合，这代表着软件人员必须知道得更多，掌握更多的技术才能够顺利迎接未来的挑战。唯有掌握每一个独立的软件技术，软件人员才可能有能力拥有系统整合的能力。从许多的观察、分析和统计中，我们可以抽离出下面这些最重要的软件技术或是软件特质。这些技术需求和软件特质是未来成功的软件人员都必须具备的，唯此才能够持续地在竞争愈来愈激烈的软件业中保持高度的竞争力。

- 了解多种程序语言
- 熟悉更多的系统架构
- 面向对象和 UML 模型技巧成为软件人员的基本要求
- 快速学习和开发的能力
- 精致化的开发能力

对于上述的技术和特质，许多读者会认为这本来就是正常的事项，为什么还需要在这里再次提出？这是因为其中许多的事项由于时空因素的关系，不是有了新的意义，就是有了更大的压力。在本章和第 13 章中我会分别做详细的说明。

软件人员在发展本身技能并且了解信息技术发展的趋势之时，当然也需要了解目前各种软件平台和软件领域发展的状况，以便规划本身的发展方向。目前，如果我们以平台作为分类的标准，便可以概略分成 UNIX/Mainframe、Windows、Java 以及.NET 四大平台。由于未来趋势的演进，在这些不同平台中的软件人员也会有不同的境遇和发展。不过总体来说，UNIX/Mainframe 和 Windows 平台的前景都属于逐渐下滑的趋势，其原因就在于这些平台已经处于成熟、饱和或是即将由新的平台所取代。

如果仔细观察 Java 平台，可以发现它已经开始进入爆炸成长期。事实确实如此。Java 在历经了数年的奋斗之后，的确开始在全世界开花结果。Java 除了在美国和欧洲快速占据市场之外，在亚洲也开始快速崛起。例如 Java 在台湾的表现一年比一年好，不但使用 Java 的人数增长了许多，Java 的开发工具(例如 JBuilder)也一年比一年卖得好。JBuilder 已经隐然有和 Delphi/C++Builder 分庭抗礼的趋势。也由于 Java 的势力日盛，因此 Java 软件人员的身价也水涨船高。

更有趣的是.NET 的发展。虽然.NET 在 2002 年才正式推出，但是许多的分析和预测都显示了.NET 的发展将不会像 Java 一样需要花上 6/7 年才达到一定的高度，.NET 的脚步将快上许多。从上图中我们也可以看到.NET 平台的趋势已经处于温和上升的状态。根据 Microsoft 最新公布的信息，到目前为止，全世界已经有 4 千万台的 PC 安装了.NET 的虚拟执行环境。估计在 2003 年，Microsoft 推出下一代的操作系统 Microsoft .NET Server 之后，将有为数更多的 PC 安装.NET 虚拟执行环境。当然这也代表了.NET 的时代可能会比我们想像中更早到来，这同样预示着.NET 软件人员的需求会开始浮现。

根据 Gartner Group 的调查显示，以后的信息势力会由 Java/.NET 平分市场，最有可能的结果是 Java 将会称霸中/后端以及 UNIX/Linux/Mainframe 市场，而.NET 则可能控制客户端、Microsoft 的行动消费端，并且逐渐朝向中间件攻城略地。未来更有可能通

过 Intel/AMD 高阶 CPU 的计算能力以及 Microsoft 的.NET Server 而在原本由 UNIX 控制的低/中/高阶工作站市场取得一定的优势。

下面的分析图更是显示了四大平台之间势力消长的情形。.NET 将很快取代 Microsoft 原本的 DNA 架构而成为 Windows 平台下的企业系统核心技术和架构。我认为这个现象是合理的，但是更有趣的问题是.NET 何时将穿透 Mainframe 和 Java 平台呢？

看完了平台之间的竞争后，再让我们看看 2002 年应用程序开发种类的趋势。

应用系统分布趋势

每一位读者实际开发的应用系统种类可能都有不同。有的读者可能是开发 MIS 应用系统的，有的可能在开发 Web 解决方案，也有读者是在开发分布式应用系统或是低阶的系统软件、嵌入(Embedded)式软件，或者驱动程序系统。不管读者主攻哪一种信息系统，了解整个信息产业的开发分布状况都会是很有帮助的，因为这些信息有助于信息人员准备和规划自己的职业生涯，了解整个信息产业的走势。

下图显示了 2002 年信息人员开发的应用系统种类统计结果。从图中可以看出，主从架构仍然是第 1 名，如果结合数据库的开发，一共拥有 24.8％的占有率。同时我们也可以看到，Web 的开发几乎已经超过主从架构，估计到了 2003 年，Web 开发将超越主从架构，成为最流行的应用系统开发种类。

在 Java 和.NET 企业平台愈来愈有影响力之际，未来最重要的系统种类是什么？其实答案已经相当明显了，那绝对不会是低阶的系统种类，而是多层架构、Web/Web Service 系统、组件系统等应用。各位读者可看到了未来所需求的人才？

组件架构使用趋势

Java 的 EJB 和 Microsoft 的 COM+都想在企业市场竞逐，成为企业对象应用系统的核心组件架构。但是 EJB 和 COM+却选择了两个不同的发展方向。对于 EJB，SUN 只是定义出其标准功能规格，再由各个 EJB 厂商根据标准 EJB 规范来开发 EJB 服务器。而 COM+却是由 Microsoft 定义标准规范并且自己实现。正由于 EJB 和 COM+采取不同的策略，因此 EJB 获得了较多厂商的支持，推出了许多不同的 EJB 服务器，但是 COM+却凭借着内建于 Microsoft 的操作系统而拥有较多的使用者。

目前 EJB 的功能规格已经发展到 2.x 版本，Microsoft 也准备推出 COM+1.5 版。SUN 和 Microsoft 推广了许多年的 EJB 和 COM+，到底这两个组件架构在业界被使用的状况如何？两者是不是雷声大雨点小呢？

根据去年调查的结果显示，EJB 的确已经开始在企业生根，已经有 19.3％的企业在使用 EJB 技术；另外有 15.3％的企业准备开始使用 EJB。这个趋势和 Gartner Group 对于 EJB 的预测非常符合，估计到了 2004 年，有 40％左右的企业会使用 EJB。从这个现象来看，EJB 实在可以说是已经成功了，其实用性也获得了证明。台湾 EJB 的实用性正日渐普及，许多的大型企业、ISV 都已经开始使用 EJB 作为关键企业应用系统的核心技术，EJB 的人才需要也日渐升高。

由于 COM+是内建在 Microsoft 的 Windows 的操作系统中，理所当然地其使用率应该是不低。只是 COM+的前身 COM/MTS 等由于其延展性受人质疑，因此使用 COM/MTS 作为企业核

心技术的并不多，大多数是使用 COM 作为客户端的可重复使用软件组件。但是 COM+推出之后，由于其执行效率和延展性都大幅精进，再加上 Microsoft 在 TPCC 上显示的惊人效率也都是使用 COM+组件，因此 COM+逐渐打入企业市场，成为 Windows 平台核心的组件架构，被许多企业的应用系统所采用。

根据 2002 年调查结果显示，COM+使用率已经超过了 34.5％，同时还有 13.9％的企业有兴趣采用。Microsoft 在努力推广 COM 技术数年之后终于有了一定的成果。不过 Microsoft

现在面临的挑战是.NET 对于 COM+的影响，由于 Microsoft 的平台正从原生窗口转换到.NET 的阶段，许多人也开始困惑起来，.NET 中的组件架构仍然是 COM+？还是有其他的组件架构来代替？如果.NET 仍然要使用 COM+作为组件架构，那么纯.NET 的开发工具又无法开发原生的 COM+组件，如此一来在.NET 中 COM+不就又不是 First Class 的组件架构了吗？如果纯粹使用.NET 的组件，那么纯.NET 组件的延展性尚未经过实际的验证，也不提供 Two-Phase Commit 和分布式 Commit 的能力，又如何能用来作为企业应用系统的核心组件呢？看来 Microsoft 在这方面还有很长的一段路要走。

CORBA 呢？在 EJB/COM+的强攻之下，CORBA 似乎失去了原有的光彩，但是不可否认的是，

CORBA 仍然是架构/服务最完整、经过最长时间验证的组件模型。根据 2002 年的调查结果显示，CORBA 仍然有相当数量的使用率，而且未来也仍然保有稳定的使用率，可见 CORBA 仍然受到相当的欢迎。

随着 Microsoft .NET 的出现和成长，CORBA 似乎反而可以在.NET 平台有更大的潜力，相关的讨论请参考下一章"EJB 对抗 CORBA？有趣的假设"。

信息技术到了现在的阶段可以说是"平台逐渐整合，但是程序语言则呈现百家争鸣的情形"。再加上 UML 等 Modeling 的技术和软件愈来愈贴近软件人员，数据库和 SQL 技术更已经成为软件人员最基本的技能，因此软件人员本身也自然朝向身通 18 般武艺的阶段。只是好的软件人员耍起来虎虎生风，而一般的软件人员在愈学愈多的情形下则一无精通，到最后反而是害了自己。

回到前面的问题，目前的信息技术的确是朝着多元化的方向发展。太多的技术、产品、架构、程序语言、数据库、客户端种类等技术，造成了许多软件人员的困惑和焦虑，这是很自然的现象。但是另一方面，平台在收敛，系统开发正走向整合阶段，对软件人员的要求也走向整合。因此，除了焦虑之外，软件人员在了解了软件趋势之后，更应该提出疑问：

你准备好了吗？

快速的开发周期

上大学时，要学习系统分析和设计(System Analysis and Design)课程，正好看到一个统计，说程序员一天只写一行有效的程序代码，当时我简直乐坏了。心想一定要进入信息行业，又有高薪又有地位，更重要的是工作如此轻松，一天写一行代码，闭着眼睛都可以做到。没有想到，在真正进入了信息行业之后，情形却大为不同。不但工作繁重，每日更需撰写成百上千行的程序代码，这哪里是一天写一行的日子？不禁心生感叹，自己似乎入错了行。

所有的读者都可以感觉到，现在系统开发的时程要求得愈来愈快。我记得 5/6 年前，系统和项目开发的速度还是 1 年到 1 年半左右，到了 3/4 年前已经缩短成 10 个月左右，在 Internet/Intranet 快速兴起之后，许多系统和项目甚至缩短到了 3 个月就要推出的迫人时限。信息机构的调查显示，系统和项目的开发时程将持续地缩短，到了 2012 年居然只有一天的时程，这种超高标准的要求可能会吓坏许多的软件人员。不管这份调查的数据是否 100％正确，但是从这份信息调查趋势以及我本身的经验来看，愈来愈短的开发时程却是不争的事实。

面对愈来愈短的开发时程，软件人员应该如何适应？这是许多人面临的困扰。其实许多的程序语言、开发工具、软件工程都强调帮助软件人员提高生产力，以适应快速开发的要求，只是随着时程的要求愈来愈高，许多传统的程序语言和开发工具都已经呈现力不从心的现象。既然如此，那解决问题的方案是什么呢？

数年前，业界对于软件开发的速度要求愈来愈快，这造成了软件品质的下降。许多软

件在完善率尚未达到一定的水准下便仓促推出，造成了更多的问题，因此软件业曾经被许多人讥笑为"不可靠行业"。为了改善这个现象，许多软件工程技术相继出现，以求提高软件品质。其中最为突出的是以面向对象的各种软件工程，强调软件 IC、可重复使用的软件组件为特质，以加快软件开发的速度并且提高软件品质。面向对象的大旗造就了 3 位面向对象大师，也让 UML 等模型工程独领风骚，进而让 Rational 公司成为近年来最重要的软件公司之一。不过几年下来，UML 对于软件开发的贡献一直受到争议，许多软件人员对于 UML 过于复杂的流程和过多的 UML 图形产生怀疑，因此前一阵子又兴起了 Extreme Programming 的热潮，强调轻便、动态、快速开发的特性。其实这个现象正反映了软件业界对于开发"速度"的要求已经超越了过份强调软件工程流程的重要性。现在的信息业界需要的是"灵活的"开发速度。

右边的图片明确显示了从 2002 年下半年起，许多企业和软件公司对于软件开发特性的要求，其中列为最重要的特质就是"灵活性"，接着仍然是强调速度的 Extreme Programming 的特质。其实，这些对于软件开发的要求也正回应了上一张图形显示的对于开发速度的要求。

我认为 UML/RUP 和 Extreme Programming/Agile Development 之间并没有冲突，反而是相辅相成的。对于大型、复杂的系统开发，UML/RUP 仍然是目前为止最好的方法之一，只是需要适当的修正，不要过份强调所有的形式流程。而 Extreme Programming/Agile Development 则特别适合中/小型系统，需要快速反应时间的开发要求。

既然快速开发已经成为现在最重要的软件开发要素之一，那传统软件人员应该如何适应呢？其实这个原理也不难了解，答案就隐含在上图中 Developer-Centric 包含的意义。想想看，现在的许多企业是如何增加效率的呢？答案之一就是组织扁平化，尽量减少不必要的重复浪费。软件开发也是一样，软件工具应该尽量以开发人员为中心，让开发人员使用较少的循环能够完成更多的开发阶段，并且提供更可靠的软件。例如，新一代的开发工具允许开发人员在一个开发环境中，同分析/设计人员以 UML 的模型来沟通，并且能够在一个开发环境中撰写程序代码，以可视化方式检视程序代码的关系和可靠度，能够在相同的开发环境中进行单元测试、压力测试、负载测试，并且和测试人员合作。另外，在这个开发环境中也能够和开发团队共同进行项目管理和版本管理的工作。如此一来，开发人员可以在较短的时程内提供更高的生产力，缩减开发循环。当然，这意味着以后软件人员必须知道得更多、要求也更高，但是，这的确可以让开发人员更有效率，并且增加整个团队开发的生产力。

面对要求愈来愈高的动态开发时程，身为专业软件人员的你：

可准备好了吗？

程序语言的战争

从程序语言的发展史来看，数年前似乎就有统一的趋势。在商业应用上 COBOL 几乎是事实标准，在科学计算上 Fortran 有不可取代的地位，而 C/C++则几乎垄断了大部分其他的应用。但是随着 Internet/Intranet 以及 RAD 工具的兴起，这个由数个主流语言掌握的局面很快被打破了。特别是在各种脚本语言(Scripting Language)和 Java 在 Internet 应用上逐渐取得了优势之后，各种新的程序语言纷至沓来，让人眼花缭乱，不知如何选择。程序语言战场在寂静了数年之后却突然陷入了前所未有的热战之中。

其实，各种程序语言的不断出现，反映的就是以往的程序语言已经不足使用或是不适合使用在特定的应用中，因此才需要新的程序语言来解决问题。新的程序语言代表了信息应用的多样化，而以往由 COBOL、Fortran、C/C++主掌的情势也被快速地打破。如今的软件开发人员可能必须同时熟悉数种程序语言，并且在不同的应用中选择使用最适当的程序语言。

以 Web 应用系统的开发为例，看看目前这个最流行、最重要的应用系统是由什么程序语言开发的。在台湾地区，不可否认的是 ASP 绝对是最多人使用的解决方案，这是由于台湾大部分的 Web 应用都属于中、小型系统，而且 Web 应用系统分布的区域并不算广大，因此 ASP 就足够满足大多数的需求，软件人员选择的标准是好用、开发迅速的程序语言。

但是，对于美国或是大陆这种幅员广大、并且拥有许多大型系统的应用而言，很可能和台湾地区的情况大不相同。那么，到底世界上的软件人员是使用什么程序语言来开发 Web 应用系统呢？对于这个问题的答案，我很有兴趣，因为可以拓宽自己的视野。下图显示的是信息机构对美国软件人员调查的结果，从图中可以发现使用最高的居然是 XML，而不是 ASP 或 JSP。不过 XML、ASP 和 JSP 这 3 者加起来占据了 3 分之一强，可见这

3 种程序语言是 Web 应用开发的主流。另外值得注意的是，虽然 PHP 最近声势不断地增长、并且拥有 Open Source 的优势，但是居然只占有 4%的使用率。PHP 的成长率很惊人，这在稍后也会说明，不过 PHP 要想在 Web 开发领域占有显著的地位，仍然需要多多努力。

除了 Web 的开发之外，如果不注重开发工具的区别而纯以程序语言的角度来看，下图显示的就是目前针对程序语言所做的调查统计结果，Java、Visual Basic、C/C++和 C#是目前 4 大主流程序语言。不过下图中的 C#数字是指 C#在 2003 年的表现，而不是在 2002 年的情形。

图中的调查结果，显然和我国台湾地区以及大陆的情形有些出入。在台湾，Java 正快速爬升，C/C++的影响力则持续下滑，而最大使用的程序语言应该就是 Visual Basic 和 Object Pascal 了。台湾地区的情形和大陆稍有不同，目前在大陆 Java 也是快速地兴起，不过尚未形成最有影响力的语言之地位，反而是 C/C++仍然为大陆目前最有影响力的程序语言，Object Pascal、Java 和 Visual Basic 则紧迫在后。根据 CSDN 网站统计的结果，目前各程序语言讨论的文章数分别如表所示。

从右边图形和数字，我们可以了解在大陆使用 C/C++语言的人数实在是众多，而 Java 已经超过了 Visual Basic。不过未来的发展情形则更令人好奇，因为在全世界的 C/C++都开始逐渐走下坡之际，C/C++在大陆的市场会不会也发生类似的现象呢？Visual Basic 和 PowerBuilder 会不会持续下滑？Java 何时会成为霸主？C#又何时会迎头赶上？这些问题都非常值得观察，因为这不但影响了各程序语言之间势力的消长，也代表着软件工作市场的需求和变化。

有趣的是，在 Windows 和.NET 平台上，程序语言正以前所未有的生气蓬勃发展，而且不同的程序语言间互相激荡、相互影响，再产生新的程序语言或是在原有的程序语言中加入新的机制以符合新的需求。但是在 Java 平台，却几乎只有 Java 语言，没有其他的选择。那么读者是喜欢单一程序语言的单纯，还是喜欢多种语言产生的灿烂火花呢？我个人是比较喜欢多种语言的，因为我认为，我们可以通过欣赏不同程序语言的设计精神和思想来吸收更多的知识和技术，通过不同程序语言彼此竞争的结果，可以嗅出未来程序语言的发展方向，这大概也是因为我从大学时就特别喜欢程序语言和编译器课程的结果吧。

知己知彼，百战百胜

就让我们以 EJB 服务器市场这个实际的竞争范例来看看信息产业是如何竞争和演化的吧。三四年前，当 EJB 市场开始成长时，许多软件厂商相继投入这个潜力十足的市场。当时，所有 EJB 厂商竞争的基准大概都是符合 EJB 规范、使用最新 JDK 标准以及执行效率最良好的 EJB 服务器。当时竞争的 EJB 服务器可以用满山满谷来形容，为什么呢？因

为在这个阶段纯粹是以技术为决胜点，这个门槛并不高，拥有技术的软件人才多得是。因此，这个阶段中有眼光和智慧的厂商便了解到，光凭 EJB 服务器引擎是无法作为胜出的要件的，必须想办法增加竞争门槛以阻绝竞争对手持续逼近，或是增加"软件服务"以提供竞争对手没有的功能。

到了 EJB 服务器竞争的第 2 阶段，逐渐胜出的 EJB 厂商便开始为 EJB 服务器加入各种服务，其中一类是以增加商业应用服务为主，例如 Portal Service、ERP Service 等。而另外一类则是以技术服务为主，例如分布式交易服务、安全服务等。由于这些软件服务的加入，在 EJB 服务器竞争的第 2 阶段便逐渐产生了 EJB 服务器的领先群，许多其他的 EJB 服务器厂商在眼看竞争无望之下自然地退出了市场，HP 就是一个例子。

现在 EJB 服务器市场又进入了最后厮杀的阶段，因为在领先群中的厂商大都提供了类似的服务。那么接下来要如何竞争呢？IBM 很快就看到了关键点，那就是为 WebSphere 加入整合开发平台的能力。通过并购 Rational 取得 Modeling 工具和技术，IBM 可结合原有的 Java 开发工具为 WebSphere 形成一个关键开发平台，同时吸引企业使用者以及开发人员为 WebSphere 形成的企业平台开发更多的客制化服务，以便为 WebSphere 形成势力强大的专属力量。这是 WebSphere 最大竞争对手 WebLogic 目前没有的优势。

从 EJB 服务器竞争的过程来看，致胜的关键便是软件人员是否能够看到别人看不到的优势、并提供额外的服务。对于软件人员的发展，道理也是同样。大部分的软件人员都只注重在特定的开发工具或是程序语言中精进，但似乎都是随着别人的脚步而前进，少数软件精英除了在信息技术领域有高人一等的修为外，真正胜出的却是能够为自己增加附加价值、提供创造力的能力。就像 JBuilder 的 Chief Scientist--Blake Stone，为什么他能够在年纪轻轻的情形下成为领导者？许多资历、年纪都比 Blake Stone 多的研发人员反而只能当工程师呢？很简单，因为 Blake Stone 更早地看到了 JBuilder 的发展趋势和竞争策略，掌握了 Java 开发工具发展的动脉，能够为 JBuilder 开发团队提供更多的价值，再加上坚强的技术实力，终于成为家喻户晓的 Java 天才。

我一直认为任何软件技术终将被大众所熟知和掌握，就像数年前的主从架构一样，因此光凭借技术并不能让人领先多久。反而是通过平日不断培养的眼光和趋势、再加上专业的技术才能够让软件人员保持在领先群中，并且掌握软件趋势的动脉。拥有这种特质的软件人员能够不时的提供服务，不时地为团队提供持续的创新力，自然也就能够百战百胜了。

结论

人类对于新事物的害怕，通常都是由于对新事物的无知所引起。同样，软件人员对于未来的困惑和焦虑则来自不知如何是好，这也是对于未来趋势发展的无知所引起。本节中，我们通过分析、观察和统计了解了以往、现在和未来软件趋势的发展，当事实的走向豁然开朗之后，困惑和焦虑不再重要，软件人员反而应该反躬自省地自问：面对未来，我们准备好了没有？

不管 Java 和.NET 两大平台之间的战争如何，不管哪一个软件组件或是程序语言会获得最后的胜利，世界对于软件系统快速开发的要求不会因为那一方胜出而改变，软件人员必须准备好面对这个趋势。想想，JBuilder 以每半年一个新版本出现，Java JDK 和 .NET 也几乎以 1 年半之间推出两个版本的速度在彼此争战之中，连传统的开发工具现在也几乎以一年一个版本的速度出现。更夸张的是，Internet/Intranet 的技术现在几乎每 3 个月就有一番新的面貌，开发人员应该如何自处呢？

机会可能会降临在每一个人的身上，但是只有时时准备好的人才能够把握住机会，不让它从手缝中溜走。

^v^v^v^v^v^v^v^v^v^v

第十一章　　EJB 对抗 CORBA？有趣的假设

"组件模型的两大巨头终将对决？"

什么是.NET？我们可以从各种技术角度探讨.NET，NET 的技术书籍也可以撰写成几十本、甚至是上百本。但是 Microsoft 提倡.NET，最重要的目的是提供一个足以和 Java 平台对抗的"企业平台"(Enterprise Platform)。Microsoft 希望企业能够使用.NET 作为企业应用系统的核心平台，根据这个企业核心平台再开发各种应用系统，连接新式的移动设备(Mobile Device)，形成企业应用系统需要的完整信息供应链，提供类似目前 Java 拥有的企业业务。Microsoft 希望通过.NET 打入企业市场的企图是不言而喻的。

为什么 Microsoft 急于打入企业市场呢？主要因为企业市场是获利最为丰富的市场，也是 Microsoft 长久以来最想进入的市场。另外，Microsoft 不希望 Java 独占企业市场、进而产生对 Microsoft 的威胁，这是第二个关键因素。其次更重要的原因是，Microsoft 在客户端几乎已达饱和，继续成长的空间有限，需要另外能够持续成长的市场，而企业市场、消费端市场、通讯市场以及游戏市场就是 Microsoft 注重的目标了。

Microsoft 要想打入企业市场，需要面对的是表现愈来愈好的 J2EE 架构。目前 J2EE 已经被愈来愈多的大型企业用作企业核心的信息技术。根据 Gartner Group 的调查，EJB 的架构将逐年增加，到了 2003 年将会拥有超过 40%的使用率。这代表 EJB 将成为企业应用系统的核心组件架构，更多的企业系统将使用 J2EE 来建制。

这些现象在大型的信息业界反映得非常真实和明显，例如台湾的台积电(TSMC)、联电(UMC)和友达光电等都已经在使用 Java 和 J2EE 作为新一代信息系统开发的核心技术。

再看看目前 EJB 服务器的两大领导厂商产品--BEA 的 WebLogic 以及 IBM 的 WebSphere。WebLogic 和 WebSphere 除了已经成为许多企业的核心 J2EE 服务器、提供企业开发应用系统的基石之外，还慢慢形成了企业应用系统的解决方案中心，并从其中衍生出许多新的软件需求和应用。例如许多的 Portal 系统、ERP、CRM 或 Web Service 应用都围绕在这两个 J2EE 服务器的外部进行增值的应用。这种现象已经开始形成非常巨大的力量，让 EJB 服务器的竞争从 EJB 核心服务器演变到 EJB 解决方案的地步，宛如一个黑洞在不断地吸引新的应用和力量进入、使用这个架构。这种软件群聚的效应正是企业级信息技术应该形成的现象，因为唯有如此，才能够让影响力不断扩大。当初 Microsoft 的 Windows 就是这样，吸引了全世界程序员为 Windows 开发应用软件，以 Microsoft Windows 为应用的核心，这才造就了 Windows 成为全世界客户端操作系统的霸主。

由此可见，一个核心软件技术对于信息应用的重要性。当初 Microsoft 在 Windows 上力推 COM/COM+组件模型，希望它们成为 Windows 上的核心组件技术，提供企业在 Windows 平台上开发企业应用系统的解决方案。其实以效率来说，COM+的确是不错的。根据许多的测试以及 TPCC 上公布的结果来看，COM+的执行效率几乎是最好的。而前段时间 The ServerSide 上公布的 EJB 对 COM+的评比更是闹得满城风雨。Java 业界仍然不肯承认 COM+比 EJB 来得有效率，但是以目前的数据来看，在 Windows 上 EJB 的确远远比不上 COM+。在 Microsoft 多年的推展之后，COM/COM+的确也有了不错的使用结果。根据 2002 年北美关于 COM+的信息调查显示，使用 COM+技术的人占了 34.3%，而且还有 9%的人回答将会采用 COM+。虽然 COM+在执行效率方面表现很好，但不可否认的是 COM+只能在 Windows

平台使用，而且在 Internet 应用中使用不便，延展性也不若 EJB，这都是 COM+致命的缺点。

.NET 核心组件技术

既然 Microsoft 希望.NET 成为企业应用的平台，那.NET 需要提供什么呢？除了开发工

具之外，.NET 最重要的便是提供一个类似 J2EE 架构的软件解决方案，以吸引软件人员为.NET 开发企业级的应用系统，帮助.NET 打入企业市场和 Java 平台竞争。

那现在的.NET 中有什么类似 J2EE 架构的技术呢？从下图我们可以清楚地看到，Microsoft 在.NET 中正逐渐建立起同 SUN Java 平台竞争的技术核心。Microsoft 和 SUN 的虚拟竞争平台现在几乎非常类似，不过，目前的 Java 在组件技术的领域远远超过了 Microsoft 提供的解决方案。

Microsoft 在.NET 中的组件技术是以.NET 组件配合 COM+为主。COM+在.NET 中转为操作系统的核心服务，提供事务管理(Transaction Management)的功能，而.NET 组件则可以同时扮演.NET 中的可视化组件(Visual Component)、数据感知组件(Data-Aware Component)以及中间件(Middleware Component)。然而使用.NET 组件作为.NET 平台中间件有许多问题。首先是.NET 组件必须依靠 COM+组件提供分布式事务管理能力；另外，.NET 组件目前也没有像 EJB 一样的 Fail-Over 和 Load-Balancing 能力，这在企业级的应用中是明显不足的。

此外，.NET 组件必须和 COM+一起搭配使用，这意味着程序员必须开发 COM+组件，而.NET 的原生工具无法开发 COM+组件，程序员还是必须使用原生的 Windows 开发工具。此外，一旦使用了 COM+，代表此.NET 应用系统无法移植到其他的平台。如果 Microsoft 把.NET 移植到 Linux 或是 FreeBSD 上，那使用 COM+组件的.NET 应用系统将无法移植到这

些平台之中。

那么，Microsoft 是不是需要一个新的、能在.NET 平台使用的组件架构呢？就我的眼光来看，如果 Microsoft 希望把.NET 平台定位在企业系统同 J2EE 竞争，那的确是需要这种技术，但这对于 Microsoft 是有一点困难的。首先，Microsoft 正忙于在一二年内达到 Java 平台花了七八年的成果，正忙于开发.NET 本身、.NET 开发工具、.NET 下的数据库，并且把所有的 Microsoft Server 应用程序移植到.NET 之下，可能一时无法投入太多的资源来开发一个全新的企业级的组件架构。

另外，再看看 Microsoft 建立组件架构的历史就可以了解到，在这方面 Microsoft 的确不太在行。期望 Microsoft 在短时间内在.NET 平台定义类似 EJB 的企业组件架构的规格并且将其实现出来，似乎是不太容易的事情。

组件技术	结果
16 位 VBX	失败
32 位 VBX	失败
OLE	失败
COM	尚可
ActiveX	失败
MTS	失败
COM+	不错
.NET 组件	不适合使用在企业级应用中

既然新创.NET 下的企业组件架构不是短时间内可以完成的，那是不是代表着.NET 在这方面已经无法和 J2EE 竞争而提早出局呢？其实并不一定，因为如果无法快速开发一个新的组件架构，那为什么不使用已经存在且已经验证是适合企业应用的组件架构呢？

让我们想想.NET 的特性和需求是什么，就可以推知什么组件架构最适合在.NET 平台下成为企业级的组件架构了。

什么组件架构已经使用过很长的一段时间，且经过了市场的验证呢？

→ CORBA、EJB 和 COM+

什么组件架构可以跨平台？

→ CORBA 和 EJB

什么组件架构允许多种语言开发和使用？

→ CORBA

从上面的问题中，我们已经看到答案是呼之欲出了。更关键的是上面的第 3 个问题。

由于.NET 是一个语言中立的平台，程序员可以使用任何语言来开发.NET 应用程序，因此在.NET 平台的组件架构必须能够让各种不同的程序语言来开发和使用，而不像 EJB 一样只能使用 Java 语言。对比一下，CORBA 正好符合语言中立的要求。此外 CORBA 是一个跨平台的组件架构，可以随着.NET 移植到各种平台。因此从各方面来看，CORBA 实在非常适合使用在.NET 之中并成为.NET 的核心组件架构。

CORBA 和 EJB

CORBA 已经在企业应用系统使用了很长时间，是一个架构成熟、经过了市场严格考验的组件技术。在 CORBA 之后的许多组件架构其实也都学习了 CORBA。例如 J2EE 中的 EJB 规格可以使用 CORBA 来实现，在 EJB 2.0 之后也要求 EJB 服务器必须和 CORBA 兼容，由此

可见 CORBA 组件架构的重要性和成熟性。目前，CORBA 仍然是一个在持续发展中的组件规格，OMG 也持续为 CORBA 定义更为完整的核心和服务规格。

CORBA 使用的 Stub/Proxy 以及组件服务架构深深地影响了 COM+和 EJB 的实现架构，以致 COM+和 EJB 组件架构和 CORBA 都有许多神似的地方。

由于 CORBA 几乎是其他组件架构的始祖，因此 CORBA 绝对有能力、有份量和 EJB 分庭抗

礼。如果.NET 能够在其平台上以 CORBA 作为核心组件架构，那么.NET 就可以提供同 J2EE 一样好的企业应用架构。虽然许多人会质疑在 PC 上执行企业应用系统会不会力量不够？但是，随着 64 位的 CPU(Intel 和 AMD)即将推出，Microsoft 也准备推出 64 位的操作系统。在.NET 虚拟执行环境的保护下，如果再加上安全强固的 CORBA，那么.NET 的确可以提供相当有竞争力的企业执行平台，与 J2EE 在中/低阶的企业应用中竞争。如此一来，Java/J2EE 就不再拥有企业应用中的绝对优势了。

既然 CORBA 对于 Microsoft .NET 平台的企业运算有这么大的助力，那么 CORBA 组件架构在.NET 平台上将以什么样的架构来提供服务呢？

CORBA For .NET

CORBA 要如何在.NET 平台上提供企业级的服务呢？由于 CORBA 使用 IIOP 通讯协议作为调

用 CORBA 服务的接口，因此传统的 CORBA 引擎(例如 Borland 的 VisiBroker)，应该会为 CORBA 伺服端和客户端产生可连接的 DLL 或是函数库，这些 DLL 和函数库就负责让客户端通过 IIOP 调用伺服端的 CORBA 服务器。因此 CORBA 即使是执行在.NET 平台中，也是使

用 IIOP 作为调用的通讯协议，如右图所示。

不过，在.NET 下 Microsoft 是使用.NET Remoting 机制作为远程和分布式沟通的通讯协议。那么，.NET 的 CORBA 开发工具要如何结合.NET 的 Remoting 机制、并且允许.NET 下各种不同的语言通过 IIOP 调用远程的 CORBA 服务呢？

其实，这同在 Windows 下提供原生窗口开发工具开发 CORBA 应用系统几乎是一样的，只是在.NET 下 CORBA 引擎必须进行一些额外的工作以让.NET 的开发工具和应用程序能够使用 CORBA 技术。

首先，.NET 下的 CORBA 开发工具必须能够提供.NET 下主流程序语言的支持，这代表

CORBA
For .NET 必须为每一种程序语言产生客户端的 CORBA.NET Stub 和伺服端的 CORBA.NET Proxy。其次，为了和.NET Remoting 机制结合在一起并提供 IIOP 的能力，CORBA For .NET 必须产生一个 Adapter 和.NET RemotJng 作为沟通的机制，.NET Remoting 通过这个 Adapter 再调用最底层的 CORBA For .NET 引擎，CORBA For .NET 引擎再把.NET Remoting
的调用惯例转换为 IIOP 通讯协议。经由 Adapter 和 CORBA For .NET 引擎的转换之后，就可以调用远程的 CORBA 服务和 CORBA 服务器，甚至通过 RMI Over IIOP 调用到远程的 EJB 服务器，如下所示：
上面说明的架构还是比较详细的。由于在.NET 下各种程序语言都会被.NET 的编译器转换为.NET 的 IL，因此 CORBA For .NET 工具可以产生一个 IL 型态的客户端 CORBA.NET Stub。
如此一来，不管程序员使用的程序语言是什么，都可以保证能够通过这个 CORBA.NET Stub 来调用远程的 CORBA 服务。这个架构其实比以前 Windows 下的 CORBA 开发工具更容
易，因为在 Windows 下，CORBA 厂商还必须为不同的程序语言产生不同程序语言的客户端 CORBA Stubs。现在.NET 下 CORBA 厂商反而因为.NET 几的特性而更轻松和一致了。
一旦 CORBA 厂商能够在.NET 下实现 CORBA For .NET，那么不但可以让.NET 的程序员开发真正企业级的.NET 应用系统，更由于 CORBA 和 EJB 是兼容的，因此.NET 下的 CORBA 服
务器也能够通过 RMI Over IIOP 调用和整合 EJB 服务器，提供.NET 和 J2EE 无缝整合的能力，并且允许使用者的应用系统能够从.NET 平台顺利地移转到 J2EE 平台，如下所示。
更何况，现在许多的 EJB 服务器还能够像管理 EJB 对象一样地管理 CORBA 对象，这意味着执行在 Mainframe 或是 UNIX/Linux 的 EJB 服务器能够管理和整合执行在.NET 中的 CORBA
对象或是 CORBA 服务。.NET 有了 CORBA 这个解决方案之后，终于有了进入可提供企业级
应用程序架构的能力了。
巨人终将对决？
CORBA 的复杂度使其一直未能被广大的程序员所接受，现在的 CORBA 本身已经非常安全强固，而且经过了十几年企业市场的考验(即使是 EJB 也不过在企业市场才经历了 2 个版本的洗礼)，CORBA 的开发厂商应该已经清楚，CORBA 不被大多数开发人员接受的原因并不是 CORBA 不够好，而是 CORBA 太复杂。因此这些开发厂商应该舍弃 CORBA 中鲜为
人用的服务，先提供一个完整的 CORBA 引擎以及企业应用最重要的两个服务--事务管理服务(Transaction Service)和安全服务(Security Service)。更进一步的是，如
果 CORBA 厂商能够在客户端提供.NET 的组件来封装比较复杂的 CORBA 调用和存取机制，并且结合数据存取的能力，那么，.NET 下的程序员将能够以非常快的速度学习和使用 CORBA 组件架构。如此一来，CORBA 将有机会在.NET 平台中大展身手，有机会成为.NET 平台中最有潜力的组件架构，也将有机会让 CORBA 一吐闷气，让世人了解 CORBA 的价值。
"EJB 和 CORBA 两大巨人终将对决"？不，更正确的说法应该是 J2EE/EJB 终于找到了可敬的对手--.NET/CORBA，而 CORBA 和 EJB 也将进入"既竞争又合作"的时代，当然前提是有厂商推出 CORBA For .NET 的软件产品。
^v^v^v^v^v^v^v^v^v^v

第十二章　回到 C/C++的王国

"让我们重返荣耀之都吧！"

当年 Windows 平台 C/C++开发工具四大天王一战，在 Microsoft 取得了市场的主导力量之后，C/C++开发工具的市场和竞争反而缓慢了下来，Windows 上 C/C++开发工具的进步也开始牛步化。VC++一连两三个版本的进度幅度并不大，除了稍后推出的 ATL 还有新意和技术革新，VC++编译器除了在 C/C++语言上更趋近于标准之外，MFC 本身几乎已经没有什么大的进步了。在 Watcom 和 Symantec 退出市场之后，VC++也顺利地接受了 Watcom 和 Symantec 的市场。而 Borland C/C++虽然也损失了大量的市场，并且失去了 C/C++的王座，但是在数年后，Borland 推出 C/C++Builder，以 C/C++ RAD 工具、以及更符合 ANSI C/C++标准和 VC++进行市场的区隔，也慢慢地收复了一些失地。虽然 Borland C/C++工具系列已经无法像以前一样是市场第一的 C/C++开发工具，但是 Borland 在 Windows 的 C/C++开发工具市场仍然占有 30％强的市场份额。

C/C++开发工具在 C/C++ Framework 一战之后，开发的重点却似乎模糊了起来。由于 VC++没有强劲的竞争，因此整个的发展速度缓慢下来。不过 C/C++技术在 C/C++语言、函数库(Library)和通用 Framework 方面却快速地如雨后春笋般兴起。特别是在 C/C++语言的标准化更为完善、以及 Template 的功能被 C++ Standards Committee 接受而且被广泛地由 C/C++编译器支持之后，各种支持和使用 Template 的 Framework、C/C++函数库也快速地占据了 C/C++开发者的心灵，成为有力的程序技巧之一。在 Java 日益兴盛、开始威胁 C/C++的市场时，反而激发了 C/C++语言前所未有的高度发展。不过，目前 C/C++开发工具以及 C/C++编译器是否跟上了 C/C++这么快速的发展脚步呢？在本章继续讨论之前，也许应该让我们先看看目前 C/C++市场的现况。

日不落帝国

曾几何时，C/C++是征服全世界的语言之一。在数年前 C/C++语言全盛的时期，我记得几乎所有的应用系统都选择使用 C/C++来编写，如从系统程序、公用程序、软件包到项目开发，因此也造就了 C/C++开发工具横扫软件销售市场的现象。但是随着 RAD 工具和 Java 的逐渐受欢迎，让 C/C++开始从许多的市场撤退。特别是当 Java 兴起之后便快速取代了以往 C/C++在跨平台语言的主导角色，让 C/C++语言在这个市场受到 Java 最大的威胁。不过，C/C++仍然在许多方面的应用不可否认地具有绝对的优势，特别是在需要高度执行效率的应用系统中，例如驱动程序和低阶的系统程序等。那么 C/C++目前的市场到底有多少？有没有像两、三年前许多信息机构预测的那样，Java 将会大幅抢走 C/C++的市场、并且吸引大量的 C/C++程序员呢？让我们以实际的数据来看看目前的状态。

右图是全世界专业信息机构对于 C/C++开发工具市场规模和使用状况的调查结果。从这个结果图形中我们可以得知几个非常重要的 C/C++信息：

首先请读者注意的是，就整体来说 C/C++开发工具的市场的确是处于小幅的下降趋势之中，根据 Gartner Group 的调查，C/C++市场是以 5％的幅度下降，而根据 Evans Data Survey 的调查，C/C++市场则是以 3％的幅度下降。不过稍后我们会说明，C/C++开发工具是在哪些平台和应用中产生变化。

另外一个值得注意的地方，是 C/C++语言主要是用于三个应用领域之中，分别是客户端、伺服端和维护现有的应用程序。从图中我们也可以发现 C/C++语言被使用的转变状态，在工业应用方面，C/C++开发工具仍然有很大的成长，这当然是因为 C/C++语言被广泛用于驱动程序的开发，例如显示卡驱动程序、网卡驱动程序等。此外 C/C++语言也被用于移动设备的开发，例如 Nokia 为了和 Microsoft 的 Smart Phone 对抗而推出的 Symbian 手机系统。当然，在操作系统、系统程序和低阶核心应用方面 C/C++语言仍

然有着不可取代的地位。
但是，C/C++在其他方面的应用的确是在下降之中，特别是在企业的应用系统方面。例如目前在大型项目、软件包、MIS 和企业内部的应用系统中，使用 C/C++语言的比例的确在下降。其中主要的原因是 C/C++语言本身的难度较高，因此生产力也不如其他语言和开发工具。加上较易使用的 RAD 工具和 Java 出现之后，C/C++语言在这些领域的影响力是大不如前的。这个现象也非常契合台湾地区目前的状况，在前几年 C/C++兴盛的阶段，几乎大部分的软件包厂商和 SI 以及系统厂商的确都是以 C/C++开发工具为第一选择。不过由于 C/C++需要的人力素质较高，而且生产力无法大幅提高，因此在目前软件包和项目的开发大多都由 Delphi、VB、PowerBuilder 以及 Java 所瓜分。
至于 C/C++开发工具使用的操作系统分配状况，则可以由右面的调查结果来说明。从图中我们可以发现，UNIX/Linux 操作系统平台仍然是占了最大的使用平台，这当然是由于 UNIX/Lmux 本身就是使用 C/C++语言开发的。而且在 UNIX/Linux 平台我们可以发现，C/C++开发工具的规模仍然在成长，可达成 10％左右的年成长幅度。由此可知，虽然 Java 现在已经入侵 UNIX/Linux 平台，但是对于 C/C++的影响仍然不太显著。
C/C++开发工具第二个最大的平台就是 Windows 平台了，虽然现在 Windows 平台是开发工具百花齐放的状态，但是不可否认的是，C/C++仍然是 Windows 最重要的数个语言之一，因为 122 Million 到 137 Million 的市场规模是相当大的。而 Windows 平台的 C/C++开发工具的成长虽然在为数众多的开发工具瓜分之下，仍可达到 12％的成长率。这代表 C/C++语言即使是在 Java 强力竞争之下仍然拥有一定的成长量。由于 Windows 平台下的 C/C++和 Java 开发工具是处于同时成长的情形，因此，这可能表示在 Windows 平台下许多的程序员应该是同时使用了 C/C++和 Java 开发工具。
至于其他平台的 C/C++开发工具则呈现下降的趋势，而且是处于快速下降的情形，这也可以解释为什么 Java 在 Mainframe 和 OS/400 等大型专属平台成长快速的情形。由此可见，在这些专属市场中 C/C++语言的确是受到 Java 很大的影响。
除了 C/C++语言本身之外，再让我们观察一下目前主流语言应用的现况，通过观察不同语言之间势力消长的情况，我们也可以了解其他语言对于 C/C++语言的影响。右图即显示了信息机构对于目前几个主流语言之间成长和下降的预估。
从图中我们可以看到，几乎所有的传统语言例如 VB、C/C++和 COBOL 等都呈现下滑的趋势，相同的现象当然也在第 2 级的主流语言例如 Object Pascal 和 PowerBuilder 等中看到，但是新一代的虚拟语言却呈现了对比的情形而大幅上升和成长，表示使用这些新语言的程序员人口正在快速的兴起之中，例如 SUN 的 Java 和 Microsoft 的 C#，而 Java 快速兴起也可以解释为什么 Borland 的 JBuilder 现在已经是 Borland 最大收入来源的开发工具。
看完了 C/C++整体市场的趋势之后，C/C++语言目前在程序员人口中使用的情形到底是如何呢？下图是 2002 年针对美国程序员调查的结果，从这个结果中我们已经可以看到，在所有调查的人数中使用 C/C++的程序员占了 45.6％的比率，但是只使用 C/C++单一语言的比率只有 3％，可见，现在大部分的 C/C++程序员应该已经开始同时使用两种以上的语言。
而第二幅图则是针对美国程序员对于未来计划使用 C/C++语言的调查结果，从图中可以证明前面图形和分析的结果，C/C++语言的确是以 3％到 5％的速度在衰退之中，也有愈来愈多的 C/C++程序员开始使用多种语言来进行开发的工作，当然 C/C++程序员选择的最多语言就是 Java 和 C#了。
在"令人焦虑的时代"一章中我们已经讨论了 Java 语言目前使用的状况以及未来的发展。从其中我们了解了 Java 虽然快速地兴盛，但是也看到了 Java 似乎已经在美国进入成熟

期，开始出现稳定的状态并且有小幅的衰退。既然 C/C++和 Java 这两个拥有共同基因的语言都处于稳定或是小幅衰退的情况，那么流失的程序员到底到哪里去了呢？当然答案很明显，这些流失的程序员是转到拥有相同基因的 C#语言阵营了。

虽然 Microsoft 的 Visual Studio.NET 是在 2002 年的 2 月才正式推出，但是 C#的编译器和相关的工具早已在 Beta 阶段便为许多程序员所使用，因此在 2002 年便已经吸引了一些程序员使用，而这些第 1 波使用 C#的程序师大都是从 C/C++和 Java 语言转换跑道而来的。右图是 C#语言在 2002 年使用的状况调查，C#在不到 1 年的时间便吸引了美国 14.6%的程序员人口使用是相当惊人的表现。

那么未来呢？C#还能够稳健地成长吗？因为唯有稳健成长的语言才能够有机会成为主流的语言。右图便是对于 2003 年 C#语言使用状况的评估，从这些数据我们可以看到，C#语言果然将以稳健的脚步成长，每年以将近 10%的速度发展，而 C#如果持续地照这样的速度发展下去，那么 C#将在 4 年之内达成 Java 花了七八年才达成的现状。当然，C#这种成长趋势也暗示了 Microsoft 的.NET 将在不久的时间内对于 Java 平台产生重大的影响。

对于 C/C++、Java 和 C#这三个拥有类似基因的语言，如果我们把它们的发展放在一起比较的话，会发现目前 C/C++和 Java 语言正处于激烈竞争的状态。但是 C/C++和 Java 千万不可忽视 C#这个后起之秀，C#正以旺盛的企图快速地向两位老大哥挑战之中，以竞逐在程序员心中主流的地位。

从上面所有的分析中，我们可以知道使用 C/C++语言的人数虽然的确是在下降之中，但是幅度并不大，这代表 C/C++语言有着非常稳定的支持力量，这当然也是因为在许多的应用中 C/C 十十语言拥有不可取代的优势，更何况 C/C++开发工具的市场仍然拥有将近 600 Million 美金的规模。这实在是一个非常大的数字，以 Borland 来比较的话，Borland 全年所有的软件营收不过是 240 Million 左右，可见 C/C++市场的潜在力量，对于 Borland 来说这是绝对不可放弃的开发工具市场。

相对于欧洲的发展模型和美国非常接近，另外一个全世界最大的程序员市场--中国大陆，并没有在这次的调查中显示出开发工具的使用状态，也许未来应该有全球软件语言的调查评估。不过从各种迹象显示，大陆的市场目前是以 C/C++和 Ddphi 分占程序员使用的大宗，而 Java 则在快速的成长之中。这和台湾地区有一点不同，那就是在台湾地区是以 VB、Delphi 和 C/C++为主要的语言力量，而 Java 则是几乎进入成熟的阶段，开始和 VB、Ddphi 以及 C/C++分庭抗礼。因此对于 Borland 来说，不管是在中国大陆和台湾地区，C/C++开发工具都是很重要的，所以 Bodand 的 RAD 部门宣称中国大陆的市场是 Borland RAD 部门最后的圣地，因为在中国大陆 Borland 的 C++Builder、Delphi、Kylix 和未来的 C/C++开发工具以及.NET 的开发工具都拥有全世界最大成长潜力的机会。

蓬勃发展的新兴 C/C++力量

其实不管是什么程序语言，在面对竞争日益激烈的情势中，程序语言的开发厂商和爱好者莫不卯足全力地捍卫和鼓吹其支持的程序语言，对于 C/C++的发展厂商和爱好者来说也是一样的情形。更有趣的是，虽然使用 C/C++语言最大的平台是 UNIX/Linux，但是 Windows 上的 C/C++开发工具反而是竞争得最为激烈、进步幅度也是最大的平台。对于 Borland 来说，在 Windows 平台上是市场排名第 2 的 C/C++开发工具厂商，而且 C++Builder 这条产品线对于 Borland 来说，占据了开发工具第 3 位的收入来源，对于 Borland 有着重要的贡献，Borland 不但不可能放弃，反而更要想办法增加市场规模。在 C++Builder 推出并且从 Microsoft 抢回了部分的市场份额之后，Borland 计划推出更新、更强劲的 C/C++开发工具。Borland 也在 BorCon 2002 中透露了一些有关未来 C/C++开发工具的计划。不过在我们讨论 C/C++开发工具的未来之前，先让我们看看目前在 C/C++

技术方面重要的发展。

首先在 C/C++编译器方面 Windows 平台上厂商的表现实在是差强人意，不管是 Borland 或是 Microsoft 都没有完全实现出符合 ANSI C/C++标准的 C/C++编译器，这和数年前四大 C/C++编译器厂商彼此竞争激烈、快速进步的情况来说实在是令人不满意，这也可见失去竞争的市场其进步缓慢的现状。不过 Borland 已经宣称在发展下一代最佳化的 C/C++编译器，不但能够产生更好的最佳化 C/C++编译机器码，而且也将符合 ANSI C/C++标准。相对于 Borland 在 C/C++方面的大动作，Microsoft 反而显得比较沉寂，除了把 VC++移植到.NET 上的 VC.NET 之外似乎没有什么大的改善。当然，Borland 是不是能够真地推出宣称的 C/C++编译技术还要看在 2003 年的表现。另外，在 C/C++连接器(Linker)方面 Borland 也宣称将要搭配新一代的 C/C++编译器推出新一代的 C/C++连接器，提供更聪明、更紧密的最终机器码。

除了编译器、连接器和 C/C++开发工具之外，另外一股发展快速的 C/C++势力便是各种 C/C++的开放函数库和 Framework 了。许多的 C/C++函数库和 Framework 由于品质良好而且采用开放源码的设计，因此也快速被许多的 C/C++程序员使用而盛行于 C/C++程序员的领域中，除了早为大多数 C/C++程序员广泛使用而享大名的 STL 之外，其中最著名的当属 ACE、Boost 和 Loki 这三个 C/C++函数库和 Framework 了。

C/C++的王牌 Framework--ACE

ACE 是一个使用面向对象方式设计的 C/C++Framework，主要是提供开发通讯应用软件使用的核心同步处理(concurrency)和分布式设计模式(design patterns)的功能。ACE 提供了 C++的封装类别(wrapper)和组件，让程序员在许多 UNIX 操作系统、Win32 平台和实时操作系统(Realtime Operation System)平台开发高效率的系统服务和应用程序。ACE Framework 提供了将近 150000 行的程序代码以及 450 个左右的类。

ACE 为了分隔 Framework 的复杂度，采用了层次的架构来设计，下图就是 ACE Framework 的设计架构图。在 ACE Framework 的低阶层次中封装了 OS 的 Adapter 以及 C++的封装类别，以增加 ACE Framework 在不同平台之间的移植性。而在 ACE Framework 的高阶层次中，则提供了延伸低阶 C++封装类别的能力，以提供可重复使用的分布式组件以及分布式计算中间件。由此可知，ACEFramework 的目的是提供一个跨平台的中间件 Framework，以便让 C/C++的程序员在各种平台中开发高效率的分布式计算应用系统。

由于 ACE Framework 的流行以及广泛被使用，因此已经被许多 C/C++程序员视为主流的 C/C++Framework。目前也有许多的应用程序使用 ACE Framework 成功的开发出高品质的分布式软件。例如下图的 ACE ORB 便是使用 ACE Framework 实现重要的 CORBA 规格的

实时 ORB 引擎：TAO。TAO 由于使用了 ACE Framework，因此也属于一个免费的 ORB 引擎，

从遵照 OMG 规格的 CORBA 都能够使用 ACE Framework 来实现这一点，就可以了解 ACE Framework 的实用性。读者可以在 www.cs.wustl.edu/~schmidt/TAO.html 找到 TAO 的数据。

另外一个使用 ACE Framework 实现的著名软件就是 JAWS 了。JAWS 是一个高效率的 Adaptive Web Server，下图是 JAWS 提供的复杂，强大的功能。读者也可以在 www.cs.wustl.edu/~jxh/research/找到 JAWS 的数据。

由于目前 ACE Framework 被使用得愈来愈广泛，所以许多 C/C++编译器也开始支持 ACE Framework。因此新一代的 C/C++开发工具必须能够支持 ACE Framework，最好还能够提供整合 ACE Framework 的功能，直接在 C/C++开发工具内部支持 ACE Framework。

Template 和 Design Pattern 的极美结合：Loki

Loki 是一个愈来愈流行的 C/C++类函数库，它是由 Andrei Alexandrescu 先生开发的，而 Andrei 也是"Modern C++ Design"一书的作者。事实上，Loki 就是因为"Modern C++ Design"一书的介绍才逐渐被许多 C/C++程序员使用。

Loki 是结合了 Design Pattern、Generic Programming 和 C++语言集成的 C++函数库，充分展示了 C++语言的优美和威力，并且提供了 C++语言使用新的应用。由于 Loki 的优美和盛行，因此现在许多 C/C++编译器和开发工具都以支持 Loki 为重要的功能之一。

最新的 C/C++标准函数库 Boost

Boost 是除了 ACE 和 Loki 外另一个快速崛起的 C/C++标准函数库。目前 Boost 已经被 C/C++ Standard's Committee 提议成为 C/C++标准的核心函数库，由此可见 Boost 的重要性。目前 Boost 同样被许多 C/C++编译器支持。未来的 C/C++开发工具应该在核心部分就会支持 Boost。未来的 C/C++开发工具最应该采用的开放架构应该是在核心部分支持 Boost 和 Loki，并且以开放的 Adapter 来整合 ACE Framework。

著名的 C/C++函数库和 Framework 的开发厂商 Rogue Wave

数年前使用 C/C++开发工具的程序员可能都知道 Rogue Wave 这家软件厂商，因为 Rogue Wave 就是以提供各种专业的 C/C++函数库和 Framework 著名的。在数年前 Borland 和许多的 C/C++开发工具厂商也都向 Rogue Wave 授权使用 Rogue Wave 的 C/C++函数库。我记得，数年前在使用 C/C++语言时最喜欢使用的函数库也是 Rogue Wave 出品的产品。当年在 C/C++User's Journal、C/C++Report 等著名的杂志中，Rogue Wave 的产品也是经常可见的。不过随着 C/C++的盛况不再，Rogue Wave 的声势似乎也不如前了，许多当时 Rogue Wave 著名的 C/C++函数库也随着消失，在前一阵子甚至传出 Borland 可能并购 Rogue Wave 的传言。

但是随着 C/C++语言最近的重振声威，Rogue Wave 似乎也开始有了比较积极的动作，也推出了许多新的 C/C++函数库和 Framework，有兴趣的读者可到 Rogue Wave 的网站上看看。

不过，Rogue Wave 的发展史也见证了 C/C+4-语言使用的演变。以前 Rogue Wave 是以提供高品质的 C/C++函数库著名，例如 Rogue Wave 曾推出过封装各种数据类型运算方法的 C/C++函数库，但是在 STL 等开放 C/C++函数库流行之后，Rogue Wave 的产品自然走入了历史。另外，Rogue Wave 也曾推出过封装 ODBC 的 C/C++类函数库，以提供 C/C++程序员在各种平台使用 ODBC 存取关系数据库的能力，但是随着 0DBC 成为历史，Rogue Wave 这样的产品自然也开始消失了。

因此，如何为一个已经流行超过 10 年的语言不断注入新的创意、技术和应用，是每一个 C/C++开发厂商都必须面对的事情。

C/C++开发工具的未来

那么 C/C++开发工具的未来是什么？难道在四大 C/C++编译器厂商大战之后 C/C++开发工具的市场便没有创新了吗？除了 Microsoft 的 VC.NET 和 Borland 的 C++Builder 之外，Windows C/C++开发工具市场就此沉寂了吗？

当然不，在前面我们看到了 C/C++函数库和 Framework 的蓬勃发展，相较于目前 C/C++开发工具厂商来说是有活力得多了。因此，未来的 C/C++开发工具必须能够跟上最新的 C/C++标准以及各种颇具威力的 C/C++ Framework。未来的 C/C++开发工具除了本身提供的编译器、集成开发环境和 Framework 之外，必须采用新的架构设计以提供 C/C++程序员整合 Third-Party 或是 Open Source 的 C/C++ Framework，而无需 C/C++程序员辛苦地自己修改这些 C/C++ Framework 才能够使用。另外，未来的 C/C++开发工具必须提供类似 Java 的高移植性，让 C/C++程序员能够在各种平台开发各种 C/C++应用系统。除了一般的应用程序之外，在移动设备、低阶系统程序等都必须能够胜任，而不像现在

的 Windows C/C++开发工具一样，各在不同的应用中占有优势。
目前，Microsoft 的 VC++在窗口平台上的 C/C++开发工具发展方向已经非常明显，那就是维持原生窗口 C/C++开发工具的现状并且往 VC.NET 发展。Borland 呢？除了 Borland C++Builder 6.0 之外，未来 Borland 的 C/C++开发工具将提供什么新的发展呢？
在前一阵子 Borland 已经宣布了未来仍将投入大量的资源研发新一代的 C/C++开发工具，将采用下图的架构提供给程序员最具整合威力的 C/C++开发工具。
从上图中的架构，我们已经可以预知未来的 Borland C/C++开发工具将允许程序员高度整合最流行的 C/C++ Framework，例如前面讨论的 ACE、Boost 和 Loki 等。这是非常重要的，因为未来的 Borland C/C++开发工具将提供跨平台/移动设备的能力，而这些 C/C++ Framework 也大都提供跨平台的功能。如果 Borland 能够提供完整的整合能力，那么这代表未来的 Borland C/C++开发工具不管在什么平台，都能够提供最完整和强劲的功能。
如果 Borland 真能推出这种新一代的 C/C++开发工具，那么这将是 Borland 从当初 Borland C/C++3.0 以来最具创意的产品，也是最值得程序员期待的 C/C++工具。Borland 是不是能够遵照承诺推出呢？也许答案在 2003 年便会揭晓了。
--
□[1;36m 昨天□[0;37m 是□[1;32m 今天□[0;37m 的□[1;33m 历史□[0;37m，□[1;32m 今天□[0;37m 将成为□[1;34m 明天□[0;37m 的□[1;33m 历史□[0;37m。 □[m
每当我们回顾□[1;33m 历史□[0;37m 的时候，总会发现两样东西 □[m
□[4m□[1;31m 嫣红的血□[0;37m，和□[4m□[1m 晶莹的泪□[m
□[m□[1;34m※ 来源:·飘渺水云间 freecity.cnzju.net·[FROM: pcq]□[m
^v^v^v^v^v^v^v^v^v^v
第十三章 软件科技的发展和 Borland 的未来
"Into The Future？"
在前面的章节中，本书讨论了许多现象和问题。除了 Borland 本身的发展故事之外，也讨论了一些科技的现状和未来的发展。在 Java 和.NET 平台的竞争以及许多科学技术的发展下，Borland 的未来到底会如何呢？Borland 又要如何适应才能够持续在信息界竞争、生存下去，进而茁壮成为更大的信息公司呢？在本章中我将提出一些个人的看法。
除了软件公司的发展之外，我也观察到了一些信息技术的走向。这些信息的发展在未来也都将牵动着开发人员的走向。除了在第 10 章中讨论的事项之外，我也认为更精致化的程序开发能力、面向对象和 Modeling 的平民化、Web Service 的发展以及.NET 平台的普及化都将在 2003 年开始对于开发人员产生愈来愈深的影响。其中，Web Service 和.NET 是开发人员无法控制的信息发展潮流。开发人员唯有在了解了它们的趋势之后，及早准备以适应未来的趋势。
而精致化的程序开发能力、面向对象和 Modeling 技术的平民化，则是属于比较贴近开发人员的发展，也是开发人员能够掌握和进一步控制的因素，是软件人员必须了解未来继续从事软件开发工作时必须克服和掌控的技术趋势。
到底这些因素的影响事项是什么呢？为什么它们对于软件人员在未来有很大的影响呢？这些也是本章讨论的重点。
不都是整理和抽丝剥茧吗？
我在从事信息工作的生涯中使用过数种不同的程序语言、数据库、组件模型以及 Framework。面对许多新的技术不断地出现，开发人员似乎陷入了永远学不完新东西的梦魇。不过，如果开发人员仔细回味许多技术的本质，却会发现这些技术其实只是把

我们已经了解的东西再以更细致化的方式加以运用，关键在于开发人员是否注意到了这些本质和趋势而已。

例如，目前在 C++中流行得火热的 Template、Policy-Based template，在 Java、Object Pascal 和 C#中当红的接口程序设计，以及各种组件模型和 Web Service 中的服务接口等，如果我们仔细地咀嚼，会发现许多的东西正是发挥程序员原本就拥有的整理和抽丝剥茧精神，再加以发挥的东西。这怎么说呢？让我们以数个例子来说明读者就容易了解了。

首先让我们想想为什么会出现数据库这类的产品？很简单，因为由于数据愈来愈多，数据种类也愈来愈繁杂，因此造成了我们需要一种软件产品能够整理这些数据让它们更容易的被我们处理和使用，因此才有了数据库的想法和产品。

在每一个程序员学习撰写程序代码时，也会发现随着撰写的程序代码愈来愈多，许多的程序代码不断重复出现和被使用，因此很自然的程序员开始使用例程(routine)/子程序(subroutine)或是过程(procedure)、函数(function)等机制帮助我们进行程序代码整理和抽丝剥茧的工作。

这些数据和程序代码整理的工作几乎是每一个程序员的求生本能，只是有的程序员只做基本的整理工作，而更聪明的开发人员则对于整理的工作有不同的看法，进而促使了许多延伸软件技术的出现，也开始对软件开发产生了重大的影响。例如，对于原本杂乱的程序代码以数据和程序代码分离的看法而逐渐产生了面向对象的技术，以分离例程/子程序和数据类型为看法的应用则产生了类似 C/C++中的 template 技术，而以函数面对服务的看法，认为开发人员应该面向服务的开发模式则造成了接口程序设计(Interface Programming)的应用热潮。虽然现在这些从程序代码延伸出的技术都独领风骚，在软件开发界中产生了重大的影响和开发模式的改变，但是，如果我们追根究底来观察，这些技术不都是从对程序代码和数据的分析、整理和抽丝剥茧之后，以更精致的方式来处理和开发软件吗？

因此，本着相同的想法和精神，聪明的开发人员开始脱离单一程序语言的架构而进入了开发出可重复使用的软件组件模型，让不同的程序语言都能够在统一的组件模型中达成团队开发的功效。这个更聪明的整理和抽丝剥茧的想法造就了 CORBA、COM/COM+和 EJB 等组件模型的驱动力。

除了脱离程序语言之外思考的开发人员外，另外有一些开发人员则再次回头检视本身和他人的程序代码，并且努力搜寻优良和成功程序代码的基因，因此发现了这些优良和成功的程序代码似乎都有着类似的模式和架构，再经过进一步的分析之后终于产生了 Design Pattern，这成为目前最重要的软件开发模式和技巧之一。在这之后，这些聪明的开发人员了解到如果能够成功运用 Design Pattern，并且把程序设计转变成以服务为目标的方式，将更能够简化、标准化和结合 Design Pattern 的运用，并且隐藏复杂的实现技巧，这就进而产生了 Service Interface Programming 的观念和技巧。

由此可见，只要开发人员能够发挥细心整理和抽丝剥茧的能力，那么即使无法创造出伟大的新软件工程或是软件技术，但是仍然能够帮助我们增加生产力和软件品质。因此，对于开发人员来说重要的不是无止境地学习层出不穷的各种新技术，而是到底有没有了解这些技术之后代表的观念、思想，以及学习最重要的对于软件开发整理和抽丝剥茧的能力。在我的工作生涯中，一直认为技术终究是会被大多数的人学会的，但是在辛辛苦苦地努力这么多年后，到底我们的思想、眼光和抽丝剥茧的能力是否有所精进呢？如果没有，那么我们永远就像被蒙着眼睛，只能尾随着他人告诉的技术前进，永远找不到自己的方向。

现在，再让我们以一个 C++的例子来证明只要开发人员能够看透程序语言和技术背后

代表的真实意义，那么即使是在已经被众人熟知的技术中，仍然能够创造出新的技术和含义。在 Andrei Alexandrescu 先生所著的"Modem C++Design"一书中，我们再次看到了聪明的开发人员对于程序语言的了解和对于程序代码撰写整理以及抽丝剥茧的惊人能力。Andrei 的想法不算复杂，但是却巧妙地运用了对于 C++ template 深刻的了解而创造出了自己的精彩之作。其实，全书呈现的思维之妙，让读者可以从一开始的小范例就看出如何运用已经广为人知的技巧之后呈现出的不同风貌。

例如，Andrei 想法是以 Policy-Based 想法为主，以各种不同的准则来提供服务函数，那么通过 C++ template 的能力，让开发人员能够根据自己的需求来选择需要的 Policy 和数据类型，结合于 C++的 template，可以提供开发人员前所未有的自由度，并且开启了以往函数库开发人员无法想象的挥洒空间。例如，下面的程序代码中提供了三个不同的类别，这三个类别都可以建立指定类别 T 的对象实例(Object Instance)，但是，这三个类别各自使用了不同的方式来建立 T 的对象实例。在这里提供了建立 T 类别的对象实例的准则 Create()方法，但是却允许开发人员自由地根据自己的需求选择要使用那一种方式来建立对象实例。

由于上面的三个类别提供了相同的 Policy(其实，从 Service Programming 的角度来看，可以说它们都提供了相同的服务)，因此，开发人员可以再自行定义一个 consumer 类别，并且结合 C++的 template 功能，让这三个服务类别成为客制化数据类型，再通过 template 的能力，自由地被开发人员选择使用。例如在下面的程序代码中，WidgetManager 类别通过 template 功能可以在编译时期动态决定使用那一个 Policy 类别作为父代类别，而自动拥有建立 T 类别的对象实例的能力。

最后，我们可以再次使用 template 能力在编译时期由开发人员代入欲建立的 T 类别的实体类别定义，通过 template 功能结合 Policy 服务和各种不同的数据类型。例如，下面的程序代码即指定了使用 OpNewCreator 这个 Policy 服务类别，以传统的 new 操作数来建立 Widget 类别的对象实例，并且定义成新的客制化类型 MywidgetMgr：

typedef WidgetManager<OpNewCreator<Widget>>MywidgetMgr;

在这个范例中，我们看到了 Andrei 真正了解了程序语言的机制，并且经过他的思考和抽丝剥茧之后，开创出了以 Policy 为主的 template class library。Andrei 的这番思考的确为 C++语言开创了新的应用和视野，这正是发挥开发人员聪颖的整理和抽丝剥茧能力的另外一个好典范。

不过，C++的 template 功能却只局限于 C++程序语言本身，这是因为 template 是 C++语言本身的特性，只有 C++编译器提供了强劲支持。所以，C++的 template 无法在程序语言之外和其他的程序语言合作提供类似组件模型的能力，因为其他的程序语言并不了解 template，也不支持 template，这也是为什么 Microsoft 会以 COM 来提供不同程序语言之间的整合，EJB 则更单纯地只限定使用 Java 的原因。

其实在上面讨论的 C++ template 中，仍然可以通过混合编译时期和执行时期的功能来提供 C++在组件模型和其他程序语言或是技术结合的能力，同时又能够使用 C++本身强劲的语言机制。例如，我们可以在外部使用 XML 作为组态文件，以指定我们想要使用的 Creator 以及想要建立的对象。例如下面的 XML 内容即指明了和前面相同的 Creator：OpNewCreator，以及要建立的对象：Widget：

而 C++可输出一个纯粹的服务接口，类似 COM 的接口以便和其他组件模型或是程序语言整合：

最后，在 CPPCreator 的实体衍生类别中可以通过分析 XML 组态文件的内容来决定建立何种的 Manager：

上述的机制可以让 C/C++语言提升至组件模型和其他的技术整合的层面，又能够仍然

使用本身强大的 template、Policy-Based template 或是 template 函数库。当然，这里我并不是以讨论 C/C++程序语言的技巧为主，不过，上面的程序代码仍然可以进一步使用 dynamic dispatch 来改善，成为品质更好的程序代码。

其实，这些想法和实现机制仍然是在使用整理和抽丝剥茧程序代码的方式来解决问题，只是以更细致的想法重新给予程序语言或是工具新的意义并且运用在日常的开发生活之中，有时候只要脑筋稍为转个弯就能够看到新的应用。

现在，除了在程序语言层面运用各种整理和抽丝剥茧的技术来增进我们开发的速度和品质之外，许多人已经开始运用相同的想法在建立企业应用系统了。例如，现在许多人已经了解 Design Pattern 除了在程序语言方面有实质的帮助之外，在企业应用系统的设计方面更有极大的应用价值。而且许多人已经开始整合这方面的 Design Pattern，例如 Martin Fowler 最新著的"Patterns Of Enterprise Application Architecture "一书中便分析和整理了他观察和使用 Design Pattern 在设计和发展企业应用系统的心得。在这本书中，Martin Fowler 也清楚地说明了他只是发挥了整理和抽丝剥茧的原则提供给开发企业应用系统的开发人员参考，许多的 Design Pattern 并不是他发明的。可见，现在许多的开发人员只是更精炼地观察和整理多年的开发经验，以萃取出更佳的 Coding 和开发的技巧以及开发惯例。

而 Design Pattern 运用在企业应用系统中的功用是能够帮助开发人员更了解整个系统的架构，并且更容易掌握如何分门别类企业应用系统不同层次之间如何的切割和分发，能够营造出体质更为健全的复杂企业应用系统。

目前，这股重新整理和抽丝剥茧的风气也已经蔓延到各种信息开发领域，从程序语言、组件模型一直到大型应用系统的设计和开发。我认为，下一步将继续进入整个开发流程的领域之中。当软件厂商提供了完整的开发流程工具之后，就开始会有人研究如何在开发流程中再度应用 Design Pattern 等技术。

因此在未来，开发人员必须了解 Patterns，并且在开发的过程中时时注意软件开发的趋势和使用惯例，不断吸收更多的技巧，以更精致的思想和方式来开发软件，如此一来才能够脱颖而出，在软件开发的生涯中出人头地。

Web Service Works

SOAP 和 Web Service 从去年开始快速兴起，并开始占据信息整合应用的市场。虽然许多人提出对于 SOAP 和 Web Service 执行效率和安全性的质疑，但是，SOAP 和 Web Service 的穿透力、整合力却无庸置疑是极具吸引力的。因此，目前 Web Service 的各种规格除了蓬勃发展之外，Web Service 的应用也的确开始出现在我们的四周。不过，Web Service 到底应用在哪些方面呢？SOAP 和 Web Service 目前在信息业界使用的情形如何？相信这些都是许多人关心的问题，也是许多人想要知道的答案。

最近，我被邀请到一家信息机构交流信息技术的心得。主持人告诉我他们现在拥有一个分布区域极为广大的信息系统。每一个区域使用的硬件、操作系统、数据库和开发工具都不同。而且，目前这些系统之间并没有专线连接在一起。现在他们想要整合这些系统，而且希望能够在机构中心向不同的区域查询货物数据并且在机构中心整合查询到的信息。

这位主持人询问我有没有什么方法可以完成这个信息架构。在详细地讨论之后，我了解到机构中心从各个区域查询的信息都是属于小量数据的查询。由于在每一个不同的区域都有自己的数据库，因此可以通过每一个区域的数据库服务器从大量的数据中撷取查询数据，再把查询到的结果传回机构中心进行简单的整合工作。

对于这个信息架构，我想最简单的方法就是在每一个区域的服务器上实现一个 CORBA 服务器，再由 CORBA 服务器对外提供查询接口。由于 CORBA 拥有跨平台、数据库和开发

语言中立的特点，因此非常适合使用来作为原有专属系统提供对外的标准服务接口。
有了 CORBA 服务器作为服务接口之后，我们可以继续把 CORBA 服务转换为标准的 Web Service，再由机构中心使用 SOAP，即可轻易地使用标准机制穿透并且整合原本的异质系统。
使用 Web Service 的原因是由于在这个应用中只会有少量的资料查询，因此 Web Service 绝对可以胜任，而 Web Service 提供的穿透力和整合力是其他技术难以相比的。对于安全的需求，可以使用 HTTPS 加上 CORBA 的安全服务即可提供一定的安全可靠性。
原本看起来困难的事情一下子就被 Web Service 和 CORBA 联手解决了。这正是一个非常好的 Web Service 应用范例。
那么在 2002 年，Web Service 在信息业界应用的情形到底是如何呢？到底有没有信息系统在使用 SOAP 和 Web Service 技术呢？其实，我们从各种开发工具都支持 Web Service 的应用来看，一定是有人已经在使用 Web Service 了，否则没有必要几乎所有的开发工具都争先恐后地加入对于 SOAP 和 Web Service 的支持。
下图是 2002 年信息界对于使用 Web Service 的最后调查结果，从数字中我们可以看到，没有使用 Web Service 的比率是 43.2%，但是超过 50%的调查显示 Web Service 已经或多或少的被应用在信息系统之中了。而这些统计数据也代表了 Web Service 被实际应用的证明。
另外一份对于 Web Service 应用的调查结果如下页所显示。我们可以看到在 2003 年中 Web Service 将有更大的使用比率，可见 Web Service 的应用将会快速地提升。
如果我们把两份统计结果以趋势图同时呈现的话，会发现 Web Service 应用的成长比率几乎不会输给一般的开发工具或是程序语言的成长比率。
在 2003 年 Web Service 除了将愈来愈普及之外，新的 Web Service 规格也将慢慢完善并且开始被软件厂商实现。除此之外，也开始有信息厂商对 Web Service 的缺点加以改善推出变形的解决方案。不过千变万变，不变的是在现在信息多元化的时代正显示了我们的确需要 Web Service 代表的穿透力和整合力。
许多人当初说 Web Service 是不实际的技术，从目前的各种迹象和统计数字来看这些人似乎是错了。Web Service 的简单化不代表无用，其缓慢也不代表不可用。我们只需要在适当的地方使用适当的技术，Web Service 就是一个很好的例子。毕竟当初 Don Box 在定义 SOAP 时最原始的想法本就是"简单(Simple)"，不是吗？
面向对象技术的平民化
"你们是用什么方法来开发系统的？"，"你们使用 UML 吗？你们在使用面向对象方式开发应用系统时使用所有的 UML 图形吗？"，"你们遵循 RUP 来发展软件吗？"，这些问题是我在和一些信息界的朋友聊天时经常询问的问题，因为我也非常想了解 UML/RUP 和 Modeling 在业界使用的情形。
UML 和 Modeling 的需求在三位 OO 大师多年的提倡并且成立 Rational 公司开始大卖 Rose 后，照理说 UML 和 Modeling 在信息业界应该是被广泛地使用，不是吗？但是情形似乎并不是如此。
在我知道的许多案例中，许多公司或是信息机构在购买了 Rose 之后，要么被供奉起来成为一种先进/时髦的象征，不然就是被使用来作为画图的工具。即使是真地使用 UML 和 Modeling 的公司也大都只是使用 Rose 画画 Use Case、Class Diagram 和 Object Diagram，再继续深入得几乎没有。为什么会如此呢？UML 已经被证明是非常好的理论、开发方式和沟通语言，Rose 也推出了这么多年，为什么 UML 的普及率仍然非常低呢？为什么许多购买了 Rose 的公司和机构也没有完全使用 Rose 的功能呢？这其中一定有一些问题存在。但是，这是什么问题呢？

就我个人的经验来说，在许多的项目开发之中大概都只有使用到 Use Case、Class Diagram 和 Object Diagram，最多画画 Sequence Diagram，接着就是结合组件模型、开发工具和数据库开始进入开发的阶段，比较注重 CBD 的开发模型，鲜少使用到其他的 UML 图形，因此可以说是偏向结合 UML 和 Extreme Programming，以项目时程为最重要的依归，并不强调完全遵照 UML 和 RUP。因此，我也非常想要和其他的朋友交流，了解其他人使用 UML/RUP 的情形，或者其他人是如何使用 OO 技术开发项目的。

我个人也是从事信息工作的一员。虽然没有什么显著的贡献，但是我对于 UML 和 Rose 始终有一份怀疑。当然，这份怀疑并不是指 UML 和 Rose 没有用，相反，UML 的确对于软件工程有着卓越的贡献。不过我认为 UML 和 Rose 之中的许多东西过于繁琐，要实际应用在项目发展之上，除非项目没有时程和资源的限制，就像 Rumbaugh 自己在 GE 时从事的实验计划，拥有许多的资源和宽阔的时程，否则，怎么可能有时间和资源把所有的 UML 图形都画出来呢？至少就我个人的项目生涯来说是从来都不可能的，因为在我个人的信念中项目开发最重要的准则是"On-Time Delivery Of A Working And Decent System"，不是 UML，不是 RUP，更不是任何其他时髦软件技术。

另外，我一直认为 Rose 实在算不上好的软件，每一次我使用 Rose 就有种回到 Windows 3.1 时代的感觉。此外，Rose 在绘制 UML 图形上始终有一些小问题，从版本 1 开始到现在都没有改善。因此我也曾经开玩笑地说，"Rose 是全世界一流的 OO 分析师配合三流的程序员开发出来的产品"。因此我个人对于 UML/RUP 一直有着一份怀疑，只是人微言轻，不敢轻易表示对于 UML/RUP 的质疑。

不过，在 Extreme Programming 对于 UML/RUP 开发模式提出类似的质疑和逐渐分庭抗礼之后，我也在 Internet/Intranet 上看到愈来愈多对于 UML/RUP 的批评以及许多人公开讨论使用 UML/RUP 失败的原因和检讨。此后，我总算如释重负，因为这些都证明了不单是我个人有疑问，许多人都有相同或是类似的问题。我认为这些批评和质疑对于 UML/RUP 是一件好事，因为这可以让软件界再次审视 UML/RUP 不足之处，找出问题的所在并且加以改善，才能够让 UML/RUP 持续地对软件界和软件工程做出贡献。正由于 Extreme Programming 对于 UML/RUP 的挑战，反而可以让我们更清楚地了解什么方法才是最适合的。我个人认为，对于小/中的系统或是项目而言，简易的 UML 和 Extreme Programming 是比较适当的，而对于大型的企业应用系统(Enterprise Application System)而言，UML 和 RUP 被证明是有效的。

一直到 2003 年初听了 TogetherSoft 的首席科学家(Chief Scientist)Todd Olsen 的演讲后，才正式确定了我的想法没有错。连 Todd Olsen 都认为 UML/RUP 太过于学术化，对于学术的研究没有问题，但是在实际的应用中则稍显繁杂。开发人员应该在开发工具的辅助下进行适当的修整以找出最"适当"的方式来进行项目或是系统、工具的开发。连 Todd Olsen 这种经验丰富、软件开发实力又惊人的大师级人物都这么说，我也就放下心来了。看来属于实战型的开发人员，依照武侠书的讲法，应该掌握的是"最具杀人威力的剑法，而不是华丽的招式"。当时我在聆听了 Todd Olsen 大胆的说法之后不禁大呼过瘾，隐藏在心中多年的质疑终于一挥而去。

虽然过于拘泥于 UML/RUP 的开发模型不算是好的方式(或许这对于学术研究是正确的)，但是也没有人完全否认 UML/RUP 对于软件开发的贡献。事实上 UML 是很重要的，因为它可以让开发人员使用同一种语言沟通，也可以很有效地使用 Use Case 让企业人员和开发人员沟通。但是为什么在 Rational 推广 Rose 这么多年来普及的成效仍然有限呢？我个人认为有如下的原因：

- 价格昂贵，难以普及
- 只锁定金字塔顶端的开发人员

■ 过于强调完整的 UML/RUP 开发模式
■ 没有和开发工具整合在一起，以打入一般的开发人员群体

由于 Rose 采取高价的策略，因此虽然为 Rational 赚进了大把的钞票，但是也让 UML/RUP 和 Rose 的普及率难以大幅地扩展。想想 10 年前的 Client/Server 技术也是在 PowerBuilder、Gupta 等采取高价措施而难以快速普及，一直要到 VB 和 Delphi 等大众化开发工具提供了 Client/Server 功能之后，才让 Client/Server 快速为大多数的软件人员视 Client/Server 技术为理所当然的基本技巧。在 PowerBuilder/Gupta 错失了占据 Client/Server 的庞大市场之后，再也无法成为 Client/Server 的领导厂商。

同样地，如果 Rational 一直为 Rose 采取高价，只锁定高阶开发人员市场的策略，那么 Rose 很可能会在其他的竞争对手推出更好的 UML 产品之后快速流失市场，事实上这正是 Rational 在 2002 年面临的困境，因为 Rose 不但遭受许多 UML 小厂商的竞争，更被其最大的竞争对手 TogetherSoft 打得难以招架，要不是 Rational 还有 3 位 OO 大师的名号在力撑，Rose 早就被 TogetherJ 或是 TogetherSoft 的 Control Center 打得落花流水了。从最近 4 年 TogetherSoft 可怕的成长速度可以看出来，Rational 早已被 TogetherSoft 逼得寝食难安了。

在 Borland 并购了 TogetherSoft 之后，Rational 将面对更为艰难的状况。一旦 Borland 成功地在其的开发工具中整合 TogetherSoft 的产品，那么将可能会像当初的 Client/Server 技术一样，可通过提供更平易近人的 UML 工具而快速让 UML 为大多数的开发人员接受而使用。再加上如果 Borland 以合理的价格提供 UML 和开发工具，那么将可以让 UML 打入金字塔中/低阶的开发市场，快速鲸吞 Rose 的市场。到时 Rose 在 UML/Modeling 产品本身不如 TogetherSoft 之下，再加上 Borland 开发工具的强力支援，Rational 的大势不妙。因此这是为什么当 Borland 宣布并购 TogetherSoft 之后受伤最深的公司就是 Rational，而 Rational 也立刻声明中断和 Borland 的合作的原因。最后在 Rational 眼看后势欠佳，再加上 IBM 提出动人的并购条件之后便立刻接受了 IBM 的提议。

根据 2002 年的信息调查显示，大多数的开发人员已经视 Modeling 工具为相当重要的工具。

而且目前使用 Modeling 工具的开发人员也相对地满意于 Modeling 工具提供的功能。因此，如果 Modeling 工具能够再和开发工具紧密地结合，那么 Modeling 未来的发展将更为快速。

目前在所有和开发相关的工具中，Modeling 和设计工具已经占据了相当重要的地位。根据 2002 年调查的结果显示，设计和 Modeling 工具已经分别占据了所有开发相关工具的第 2 和第 8 名，而且还呈现持续上升的状态。

由此可见，开发人员已经愈来愈重视设计和 Modeling 工具。在 Borland 并购 TogetherSoft 之后，我认为 Borland 会以较为合理的价格提供整合 Modeling 和开发工具的软件包，快速把 UML 技术打入一般开发人员的市场，并且将会正式触发使用 UML 和 Modeling 功能成为开发人员的核心基本技巧，就像数年前 Client/Server 技术对于开发人员一样。

因此，我们可以发现下面图形呈现的还是去年以前开发人员拥有的技术状况。开发人员的核心技术只需要拥有程序语言、数据结构和 Algorithm 即可。

但是从 2003 年开始，一旦 Borland 或是 IBM 推出整合 Modeling 工具和开发工具的新一代软件之后，面向对象、Modeling 和 Design Pattern 等技术将被压缩到开发人员的核心基本技术之中。这代表未来的开发人员必须熟知面向对象、Modeling 和 Design Pattern 等技术，再也无法逃避学习这些重要的软件技巧。

因此，我们可以说信息公司的合并不但影响这些软件公司之间的竞争，也会对开发人员产生影响。在面向对象、Modeling 和 Design Pattern 等技术成为开发人员的核心技

能之后，当然可以增加软件开发的速度和可靠度，这对于整个软件产业而言是正面的结果。对于像 Borland 或是 IBM 等公司，由于让 UML 和 Modeling 技巧和产品成为一般开发人员必须拥有的技巧和使用的产品之后，也可以通过更大量的市场来弥补产品价格下滑的损失。一旦 UML/Modeling 技术大众化和产品平价化之后，软件公司反而可以拥有更多的收入。

面向对象和 Modeling 平价化之后便会开始进入开发人员的生活之中，也会开始影响我们开发软件的方式和流程。这两者会像从前的其他技术，例如 Client/Server、数据存取和 Web 等，慢慢成为几乎每一个开发人员必备的技术。然而不同的是，面向对象和 Modeling 对于我们的思考模式却有更大的影响和改变，因此造成的影响也将远比从前的技术更为深远。因为除了面向对象和 Modeling 的思想和开发流程之外，伴随着它们而来的将是更多的软件工程和软件技术。

不过对于开发人员来说这实在是一条辛苦的不归路，学习的道路不但没有尽头，沿途还充满了艰辛。软件开发工作真是个辛苦的行业，不是吗？不过反过来想，软件开发生涯也将是充实、满载而归的路途，不是吗？

准备迎接.NET 时代的来临

2002 年是.NET 初临的时代。虽然.NET 的初鸣并没有给人太亮眼的表现，但是同七八年前 Java 初次展现于舞台上时相比较，.NET 的表现并不逊色，甚至比当年的 Java 表现得更好。

在 2003 年，Microsoft 更准备推出新版的.NET、.NET Server 以及新的.NET 开发工具，而且大部份的调查也都指出.NET 将开始在 2003 年起飞。面对 Microsoft 一连串强势的动作，我们其实可以预见.NET 也即将更为活跃和更有影响力。其实我也很想了解.NET 在 2002 年到底表现得如何？除了市场上许多的.NET 书籍、文章之外，我也在业界和许多朋友讨论以及询问 2002 年.NET 在信息界使用的状况。我得到的结果都是在评估.NET 之中，实际使用.NET 开发的产品还不多，但是 ASP.NET 被使用的情形则是令人惊讶的快速。我已经发现许多的网站开始使用 ASP.NET 来开发，可见.NET 和当初的 Java 一样，是从 Internet/Intranet 开始入侵日常的信息应用。

最近一份.NET 的调查报告终于把.NET 在 2002 年使用的粗略状况呈现出来，显示出.NET 的确已经有了初步的成果，虽然也许没有达到 Microsoft 的期望，但是也有不错的成绩。下图是 ASP.NET 在这份调查中的结果，读者可以看到，已经使用和准备使用 ASP.NET 的比率已经超过 50%，而且已经有 18.9%的软件人员在使用 ASP.NET 开发 Web 应用系统，这对于推出才 1 年的技术来说是非常惊人的，也可以说 ASP.NET 是非常成功的。而下一份图形则显示出开发人员对于 Microsoft .NET Server 的使用情形。从数字结果可以看到.NET Server 的接受度也非常高，也有超过 50%的接受度。可见在 Microsoft 推出下一代的.NET 操作系统之后市场反应也会非常的正面。

从上面的两个统计结果来看，.NET 已经比我们想象的更快地进入实际的应用领域中。在 2003 年看来准备在 Microsoft Windows 平台讨生活的开发人员的确是开始需要学习.NET 了，因为很可能从 2004 年开始我们便将看到 Windows 平台又将进入世代交替的现象。

Borland 的未来

Borland 正站在十字路口上，面对未来的方向。数年前 Borland 错失了开发消费型软件的契机，以致无法持续成长为更为强大的软件公司。看着 Borland 从 2000 年开始一连串的发展以及在 2002 年完成的并购动作，我们可以看到 Borland 已经选择了另外一条道路，那就是全力往企业市场前进。

企业市场一向是获利丰富的市场区块，这也是为什么 Microsoft 在称霸了客户端市场

之后急于切入的市场。不过企业市场也是更为危险的市场，因为在这个市场中的竞争对手不但更大、更强壮，而且竞争的规模也远超过一般的软件市场。这也是为什么在这个市场区块中的竞争公司几乎都是数一数二的公司，例如 IBM、SUN、HP 和 Microsoft 等。Borland 如何同这些资源丰富的厂商竞争，对于 Borland 的管理阶层将是非常严酷的考验。

不过，这似乎是不得不走的路，因为 Borland 传统的开发工具市场虽然在持续成长，但是传统开发工具的价格却不断下滑。例如当年 Borland C/C++3.1 的价格是一套 799 美金，现在的 C++Builder Professional 的功能比当年的 Borland C/C++3.1 多了数倍的功能，但是价格现在却只有 399 美金。这是许多信息产品相同的命运。Borland 必须想办法扩充其他的市场，否则，只能像许多的开发工具厂商一样等着成为历史。既然数年前 Borland 错失了像 Symantec 成功地打入消费型市场的机会，因此，进入企业市场似乎是 Borland 无可避免的道路。

问题是 Borland 如何在企业市场以小搏大、对抗世界一级的信息大厂呢？原本这样的情势实在不怎么看好，没有想到在 2000 到 2002 年，世界历经了全球的不景气，许多信息厂、甚至包括许多一级的信息大厂例如 HP、Rational、IONA 等都面临了前所未有的严酷考验，不是元气大伤，就是被并购消失于历史之中。反而 Borland 通过公司有史以来最聪明的并购，不但成功地取得了关键技术和产品，更重要的是，Borland 在顿时之间取得了一个绝佳的地位和制高点，拥有其他厂商所没有的完整产品线。这让 Borland 进可攻，有机会成为一流的软件大厂，重回世界一级的软件信息公司。也可让 Borland 退可守，成为小而美的独立软件公司，继续下一个 10 年的经营，甚至能够以最好的价值和其他的软件公司合并成为更大的软件信息公司。这也是为什么最近一直有传言称 Microsoft、BEA、IBM 和 Oracle 都在重新审视 Borland 的价值，并且重新评估 Borland 这位突然之间实力大增的竞争对手。

当然世事有得便有失。Borland 在极短的时间之内取得许多的公司、技术、产品和企业文化，如何整合这些对于 Borland 来说也是一个挑战。在下一章中，我也会提出一些 Borland 面临的问题的个人看法。不过从 Borland 的走向来看，不管如何，势必需要面对和克服这些挑战和问题才能够持续在软件产业中竞争下去。我个人认为，Borland 必须赶快在下面的事项中取得领先的地位才能够拥有高度的竞争力，并且顺利地面对其他的竞争对手。

提供全方位的开发工具

既然单靠开发工具已经无法在现在的软件竞争中取得一定的优势，那么 Borland 必须发挥技术优势，并且整合所有的产品以便在 Java 和.NET 平台提供全方位的开发工具。目前，Borland 已经形成了完整的软件应用供应链。我预见 Borland 除了会在不久的将来提供整合性的开发工具产品之外，应该会持续在测试工具领域取得关键技术和产品，以便形成更强劲的产品线。Borland 最近推出的 OptimizeIt ServerTrace 便是向这个方向努力的一个很好的例子。

为什么？因为在日后开发工具日趋整合之后，整体的测试工具便显得重要了，因为在整合的开发工具中牵涉到的技术或是组件模型将会非常复杂，而传统的测试工具已经无法处理这些复杂的应用。这是为什么目前在测试工具市场能够同时存在多种用途的测试产品，而且由于测试工具市场快速地成长，因此，目前这个市场的利润相对的比开发工具好得多，例如出品 LoadRunner 的 Mercury 公司便非常成功而且成长得非常迅速。因此当大型软件公司开始注意到这类产品的重要性以及相对的价值之后，势必想要进入这个市场。而 Borland 在拥有了分析/设计、Modeling、开发工具、基础测试工具和组件模型以及分发平台之后，补强在测试领域的工具便很自然地成为下一步了。

一旦在测试市场拥有强劲的技术和产品，Borland 的实力将更为强大，同时可通过全面、整合性的技术和产品而保持合理的利润，以提供持续成长的推动力。

提升开发工具的价值

当开发工具的价格不断下滑之后，Borland 面对核心产品市场的趋势应该如何处理呢？这并不难解决。既然开发工具已经逐渐成为大众化的产品，高价的入门开发工具时代已经不可能再回来，那么 Borland 可利用产品区隔来增加这类产品线的收入，而这正是 Borland 在最近几年采用的策略。所谓开发工具产品区隔，是指在原有的产品线中提供更为高阶的产品，以吸引金字塔顶端的软件人员。例如 Delphi 原本提供三个不同的产品线：Personal、Professional 和 Enterprise。到了 Delphi 6 之后开始加入 Architect 和 Studio 的版本，以便增加产品的附加价值，吸引资深的 Delphi 开发人员使用这些高阶产品，JBuilder 终究也一样会采用这种策略。否则，世界上将仅仅 Microsoft 能够以不惜血本地大量抛售开发工具以便维护其 Windows 平台的利润，而任何软件公司都需要一定的利润来持续经营的。

进入 Run-Time 市场

请读者现在想想，在软件产业中除了像 Microsoft 是靠大量的消费型软件大赚其钱之外，其他最赚钱的软件公司是靠什么种类的软件在赚钱呢？答案便是靠所谓的 Run-Time 软件赚钱。什么是 Run-Time 软件呢？让我指出几个例子读者马上就了解了。例如 Oracle 最赚钱的产品 Oracle 数据库就是属于 Run-Time 软件，IBM 和 BEA 的 EJB 服务器也是属于 Run-Time 软件。Run-Time 软件拥有大量使用、分发授权的特性，因此拥有相当良好的获利水准，甚至能够和消费型软件并驾齐驱，一种以大量取胜，一种以价值取胜。

Run-Time 软件一直是 Borland 想要拥有和进入的市场。从当初并购 Visigenic 和 Entera 开始，Borland 就有这种想法。只是当初 Borland 的产品尚不够齐全，管理阶层也没有经验，因此一直不知如何进入这种市场。现在 Borland 产品线已经相当齐全，管理阶层也有相当大的雄心，因此，这是为什么 Borland 一直坚持持续开发 CORBA 和 EJB 服务器的原因。如果 Borland 能够在 Run-Time 软件市场成功，又能够通过 InterBase 数据库进入嵌入式数据库市场，再通过整合开发工具大军横扫市场，那么，我可预见 Borland 将可快速重返全世界前 10 大的软件公司，恢复昔日的光彩和雄风。

结论

2003 年对于 Borland 是重要的一年，因为 Borland 在整合了许多的公司之后如何在 2003 年推出新一代的产品，对于 Borland 的管理阶层以及 R&D 都是一项考验。此外，随着.NET 的脚步不断逼近，Borland 也必须尽快推出.NET 之下的产品。许多人对于 Borland 如何在.NET 下继续在开发工具市场竞争充满了期望、疑问和困惑。不过，随着 Borland 取得了 Modeling、分析、测试和分发技术/产品之后，Borland 的确可以在.NET 下推出 Microsoft 无法提供的开发工具和解决方案。我个人也非常期待 Borland 能够尽早推出.NET 下完整的产品线。如此一来，Borland 不单是以开发工具和 Visual Studio.NET 或是其他.NET 开发工具厂商竞争，还将提供更为全面的低、中、高产品线来竞争。这正可突显 Borland 的不同。而且 Borland 将可再把竞争门槛往上提升，象征新一代 Borland 的竞争力。Borland 从 80 年代率先推出 In-Memory 开发工具，90 年代推出可视化集成开发环境开发工具，到了 2000 年之后，如果又能在.NET 下率先推出新世代的整合性开发工具，那么即代表 Borland 又将再次改写开发工具的意义，持续下一个 10 年的竞争领先地位。Borland 是否能像凤凰一样浴火重生、并且再次展翅飞翔呢？

其实，我认为 2003 年对于 BEA 和 IBM 来说也将是分出胜负的一年。如果 BEA 无法在 2003 年增加 WebLogic 的竞争力，那么 IBM 的 WebSphere 将在开发工具和 Modeling 工具的助阵下逐渐向 WebLogic 攻城略地，蚕食 WebLogic 的市场占有率。随着 HP 淡出 EJB 服务器市

场，SUN 的 iPlanet 无法在市场上取得优势，看来 IBM 在 Java 阵营的影响力将会愈来愈大。这对于 SUN 是一项警讯，因为现在 SUN 除了还控制 Java 语言、JDK 和 EJB 的规范之外，
在 Java 开发工具、EJB 服务器市场节节败退。现在，甚至在 Web Service 规范、Web 开发规范等方面也都沦陷于 IBM/Microsoft 和 Apache，SUN 在 Java 各方面的势力真是日渐式微。
而目前看来处于真空的.NET 市场也即将出现变化。一旦.NET 势力兴起，必将冲击 Java 的市场。那么对于许多 Java 厂商和 EJB 厂商会发生什么影响呢？.NET 对于企业信息应用会产生多大的影响力？.NET 何时将对 Java 产生穿透力？这些影响和变化也都将从 2003 年开始发酵。
我认为 2003 到 2004 年对于信息界来说将会发生数件重大的事件，信息势力也将会被重新划分而产生新的主控力量和领导厂商。我们就等待着精彩好戏的上演吧，2003/2004 这两年绝对不会是寂寞的年份。
^v^v^v^v^v^v^v^v^v^v
第十四章　　传奇的篇章仍将继续！
对于 Borland 来说，2001/2002 年实在是很奇怪的时期。因为在这段时间里，Borland 无畏于全世界经济的不景气，仍然持续地获利和稳定地成长。和许多 1T 公司的不断亏损和裁员相比，Borland 的表现实在是非常的亮丽。究其原因，这主要与 Borland 慢慢建立起来的、有史以来最有效率的销售团队有非常大的关系。假如 Borland 没有这支专业的销售团队，那肯定也会和其它 IT 公司一样的局面：亏损和裁员。
也许是天佑 Borland 吧，在 Borland 引入专业的销售团队之后，原本积弱不振的 Marketing 也开始逐渐上轨道了。不过我认为 Borland 仍然有隐忧，那就是本身产品线拉得太长、投入.NET 的时间太晚、而业界的 Java 市场即将进入成熟期。
过长的产品线？
如果花点时间研究 Borland 的产品，那读者可能会发现 Borland 实在拥有太多的产品线。从以前的 4 个开发工具 Delphi、C++Builder、JBuilder、Kylix，到数据库，再到 CORBA、EJB 服务器等，Borland 拥有的产品超过了 10 个以上。面对这么多的产品线，仅研发费用一项就已很高，如何能够照顾好这么多的产品？这是好的现象吗？
看看许多大型的软件公司，其赚钱的产品可能只有三、四个。例如 Norton 在退出开发工具市场之后，凭借着专心开发企业应用软件而一再成长，目前的规模早已超过 Borland 许多。Norton 靠 Norton Anti-Virus 产品打入消费市场，让 Norton Anti-Virus 成为广受欢迎的"消费型软件"之后便一飞冲天。再通过 Norton Anti-Virus 衍生出其他相关的软件，进而扩大公司营业额，成为大型的软件公司。又例如 Oracle 仅靠数据库就成为第 2 大的软件公司，再衍生出 Oracle Finance、Oracle ERP 等，这些企业软件也都是以 Oracle 数据库为核心发展出来的。成功的软件公司大都拥有一两个"消费型软件"来支撑公司的发展，但是 Borland 呢？
Borland 目前提供了十几个产品，并且不同产品线的收入都差不多，没有一两个核心赚钱的产品，自然也就没有其他衍生的产品出现，因此 Borland 必须通过不断地推出新版本的工具和产品来维持收入，这实在是很辛苦的事情。因为这代表 Borland 需要不断地投入巨额的研发才能够推出产品，而且只能赚到少许的利润，这样当然成长得很辛苦了。
其实这就是我在前面章节所说的，Borland 在 SideKick 之后就一直没有办法开发出"消费型软件"的问题。当然软件公司可以采取不同的发展策略，例如软件公司也可以走向企业市场，因为企业市场的利润一向比 PC 市场来得大。这正是 Borland 目前想做的，

因此 Borland 才会并购相关的软件公司以壮大其进入企业市场的本钱。不过我仍然认为 Borland 在开发工具市场拥有大量的使用者，除了进入企业市场的选择之外，Borland 还是应该在 PC 市场推出"消费型软件"。如此一来上下夹攻，不但可以精简产品线，更可以获取更多的利润以支撑更精良的产品研发。

现在的 Borland 应该做的是重新规划产品线，去芜存菁，好好地发展产品，让每一个产品都有独特的定位以及合理的投资回报。这样不但可以集中研发资源，也可以提供给使用者品质更好的产品。好在目前 Borland 似乎已经注意到这个现象，开始收敛过多的产品线，清楚地为每一个产品定位市场。虽然 Borland 到了现在才了解了产品线的问题，但是"亡羊补牢，犹为未晚"，希望 Borland 在 2003 年以及未来的日子中能够恰当地分配研发资源，好好地推出精炼的产品，而不要一味地求多。

进入.NET 的时机

1999 年底，在和一位供职于 Microsoft 的好友吃饭时，他很好奇地问我为什么 Borland 到了当时还不加入 Microsoft .NET 的支持开发厂商之中，以便开始发展.NET 的产品。听了之后，我也觉得奇怪，为什么 Borland 在支持.NET 方面进入得这么晚呢？

这是有原因的，当时 Borland 内部对是否要支持.NET 争论不休。许多人认为，在 Windows 平台中 Borland 已经和 Microsoft 竞争了这么久，非常清楚在 Microsoft 主控的平台中要和 Microsoft 竞争不但非常辛苦而且不易获利，因此这些人便反对继续支持 Microsoft 的平台，他们认为 Borland 应该在其他平台找寻出路。

而另一派的人则认为，Microsoft 虽然利用垄断进行不公平的竞争，可 Microsoft 的平台仍然是最大的平台。虽然和 Microsoft 竞争相当吃力，但是 Borland 已经这样生存了十几年，已经了解如何和 Microsoft 竞争。如果放弃 Microsoft 的平台，那不但一点都吃不到，而且其他平台的占有率太小，即使 Borland 有办法在其他平台占有大量的市场，但是相加相乘之后，说不定还是比在 Microsoft 平台上的占有率小。

就在这两派争论不休之际，刚好 2000 年 Linux 开始吸引众人的目光，因此当时的新 CEO Dale 便下令全面转进 Linux，准备放弃.NET 平台。就是这个决定，让 Borland 在.NET 方面晚了将近 2 年才投入。

Borland 在.NET 平台进入的时机是晚了一点，但这是好是坏目前并不清楚。有人认为，晚投入.NET 是件好事，因为 Borland 正好先让 Microsoft 推广.NET，到了 2003 年.NET 逐渐起飞之后，Borland 刚好推出.NET 下的开发工具和产品。但是也有人认为，Borland 太晚投入.NET 已经造成了影响，因为在 2002 年已经有许多程序员希望学习.NET，需要 NET 开发工具来使用，而 Borland 却错过了这一波的需求。

Borland 在.NET 是不是投入太晚呢？以我的看法，确实是晚了一点，因为 2002 年我也在为苦无 Borland 的.NET 开发工具使用而伤脑筋，我想这个感觉应该是许多 Borland 程序员都感同身受的。但是 Borland 晚一点投入.NET 有没有什么严重的影响呢？我想时机应该还不是最重要的问题，让我们回想以往 Borland 在 Windows 上推出开发工具的时间，同样也是晚了 Microsoft 一到两年，但这是没有办法的事情，因为系统的主控权是在 Microsoft 手中。不过在以往的记录中，Borland 仍然能够力挽狂澜，凭借好的工具来争取人心。因此我认为，既然已经晚了一些时机，那么 Borland 就应该更好地发展.NET 下的开发工具和产品，以品质和功能来吸引程序员，这才是最重要的。

Java 市场即将进入成熟期

不光是 Borland，对于许多软件公司而言，目前都是非常辛苦的时期。2002 年 10 月份左右，SUN 的 iPlanet 副总裁发表了"如果 EJB 服务器厂商再不停止流血竞争，那么 EJB 服务器市场将进入微利时代"的言论。当然，这位副总裁的目的是希望 EJB 厂商不要再降价竞争或是提供一大堆的 Open Source EJB 服务器，以避免扼杀 SUN 好不容易经营起

来的企业市场。不过，iPlanet 副总裁显然忘记了 SUN 在 EJB 服务器市场上并没有太大的占有率，而 EJB 服务器的价格之所以会不断地下滑，是因为 IBM 和 BEA 的 EJB 服务器占
有率已经达到经济规模，因此，他们可以使用降价的方式来追杀其他的 EJB 服务器厂商以持续增加市场占有率。而 EJB 服务器的竞争从强调 EJB 核心功能、附加服务，到现在的以价格来竞争，代表的就是 EJB 服务器市场已经进入成熟的市场，无法获取高额的利润，因此只能以量来弥补价差。
除了 EJB 服务器市场之外，Java 开发工具的竞争大概也已经分出了高下。无法继续参与竞争的 Java 开发工具将走向 Open Source 的不归路，Java 开发工具的胜利者也必须面对价格下滑的局势。因此，市场领导者必须再次在市场寻求更大的占有率，这也将引发 Java 开发工具第 2 次的争夺战。看看目前的胜利者，我们就可以大概知道下一波的竞争态势。Borland 在取得了 TogetherSoft 之后，势必会在 JBuilder 中整合 TogetherJ 的功能，把 JBuilder 再往上提升，形成一个同时提供 UML 和 Extreme Programming 的全能 Java 开发工具。而 IBM 取得了 Rational 之后，也一定会把 Rational 整合到 WSAD 之中。不过这还不是最可怕的，如果 IBM 决定采用把 WSAD 和 Rational 半买半送的策略来攻击 Java 开发工具市场、以掩护 WebSphere 击倒最后的敌手 WebLogic，那么不但 Java 开发工具将有一番浴血战，EJB 服务器市场也将进入烽火连天的场面。对于 Oracle 的 JDeveloper 来说，势将面对更为强劲的对手--JBuilder 和 WSAD。虽然 Oracle 也开始在 JDeveloper 中加入 UML 分析和设计的功能，但是不管 Oracle 怎么做，大概都无法和 TogetherJ 及 Rational 竞争。Oracle 的 JDeveloper 最后只能像以前的 Oracle Developer 2000 一样送给 Oracle 的用户，或是 Oracle 决定展开价格战，以极低的价格来响应 JBuilder 和 WSAD。如果 Oracle 采取这个策略，那么受伤的应该是 JBuilder，因为 Oracle 绝对不会打击到资源更为雄厚的 IBM。
因此，在 Java 市场即将进入成熟期之际，Borland 必须再次提升 JBuilder 的价值才能够持续拥有成长的空间、并且推出新的 Java 工具以刺激市场需求。否则，在 JBuilder 之后，Borland 又有什么工具或产品能够像当初 JBuilder 接下 Delphi 的棒子时那样持续地往前冲刺呢？
软件产业即将进入各拥山头的竞争模式？
在即将完成本书时，我知晓了一个令人遗憾的消息，那就是在 Delphi Third-Party 业中享有盛名的 TurboPower 公司宣布退出零售市场，并且把许多的产品开放成 Open Source。这实在令人惊讶，因为 TurboPower 公司出品的软件组件和软件产品都非常精良，是许多 Delphi/C++Builder 程序员都在使用的，现在居然连 TurboPower 公司都无法继续经营下去，那这代表着什么呢？
其实，就在 TurboPower 宣布退出市场之后，另外一家著名的软件公司 DeveloperExpress 也很快地发表了 DeveloperExpress 目前经营良好、未来将持续开发 VCL/CLX 和.NET 下的软件组件的消息。DeveloperExpress 的这个动作是希望使用者不要受到 TurboPower 负面消息的影响，并且想进一步吸引 TurboPower 的使用者能够转而使用 DeveloperExpress 公司的产品。为什么一个极富盛名的软件公司都无法继续生存下去呢？让我们看看其他 Third-Party 软件公司，也许就可以一叶知秋了。
目前，所有的开发工具都围绕着许多的 Third-Party 软件厂商提供各种不同功能的软件组件，这就是所谓的软件组件市场。软件组件市场的泛滥应该起源于数年前的 Microsoft VBX，由于当时 Visual Basic 能够开发的功能有限，因此 Microsoft 定义了 VBX 规格，让其他软件厂商能够据此开发 VBX 组件以弥补 VB 功能的不足，这造就了 VBX 市场，许多小软件公司便凭借着创意和实力开发出各种 VBX 让 VB 用户使用。由于大多

数的 VBX 都是使用 C/C++开发的，因此稍后 VBX 也成为 C/C++开发工具使用的软件组件。Delphi 成功地成为第 2 大 RAD 工具，这为 Third-Party 市场带来了巨大的商机。特别是 Delphi 的 VCL Framework 采用开放源码的方式，让 Third-Party 厂商得以快速而且安全地为 Delphi 开发软件组件，因此各种 Third-Party 开发的 VCL 组件很快地出现在市场上。Delphi Third-Party 软件组件的市场几乎是 VB 的好几倍，而 VCL 组件也在 Delphi 3 到 Delphi 5 之间成为全球最为兴盛的软件组件市场。

Java 在定义了 JavaBean 规格之后也带动了 Third-Party 的 JavaBean 组件市场。但是由于 Java 在客户端应用市场已经输给了 Microsoft，因此 Java 开始退居中介端、伺服端和 Web 方面的应用，这造成了以客户端功能和 GUI 为主的 Third-Party JavaBean 组件无法拥有经济规模的市场量，也因此一直无法像 VCL 组件市场一样的蓬勃发展。

Microsoft 的.NET 推出之后，虽然.NET 组件架构吸引了 Third-Party 软件厂商为.NET 开发软件组件，但由于.NET 目前在萌芽阶段，.NET 的软件组件需求暂时没有太大的市场，这让一些已经提供.NET 软件组件的厂商苦无成长的机会。更何况如果.NET 也像 Java 一样以中介端、伺服端和 Web 方面的应用为主，那.NET 组件市场可能也会像 Java 的 JavaBean 市场一样，无法大幅地成长。

Third-Party 组件市场的萎缩代表着软件竞争的日趋激烈，唯有提供多样化产品或是拥有优势的软件公司才能够生存。目前许多 VCL 组件厂商已经停止再开发新版本的 VCL 组件，这可以想像这个市场已经逐渐走入后成熟期。Third Party 组件市场唯有在新一代的开发工具的开放架构下才可能有回春的一日。

2002 年 12 月底，InfoWeek 报道了 IBM 并购 Rational 的消息，并且分析说以后将是软件大厂相互竞争的时代，以往为数众多的中型软件厂商将会日趋减少。同时，InfoWeek 也指出，以后开发工具的软件市场将形成 3 大软件公司竞争的局面，这三家软件公司分别是 IBM、Microsoft 和 Borland。从这个报道我们可以知道，在 2002 年一连串的并购之后，关键的开发工具市场已经发生了质变，Borland 通过一连串的策略并购后已经被软件界提升到能够和 IBM、Microsoft 相提并论的地位。这个现象就如我的好友李匡正先生说的，软件行业虽然只发展了数千年，但是软件行业的发展，终会像发展了上百年的汽车工业一样，只剩下少数的大厂，而且这些软件大厂将从开发工具、应用程序、服务器一直到分析、整合软件一应俱全。

软件行业真得会逐渐进入各拥山头的竞争模式吗？能够再次和 IBM、Microsoft 并驾齐驱，这是 Borland 一直梦想的事情，也是 Borland 在低迷了十几年之后另外一次重返软件大厂之列的大好良机。问题是在此之前，又会有哪一家目前台面上的软件公司会倒下、成为另外一个牺牲品呢？Borland 能够把握这次契机，重造数年前营收规模 500 Million 美金、或是营业额更大的软件公司吗？

但是传奇的故事仍将继续

在我撰写本书时，时间已进入到 2003 年。对于许多软件公司来说，2003 年都将是严峻而且即将见分晓的时代，因为.NET 在 2003 将开始起飞，开始对 Java 产生更多的影响，Java 和.NET 这两个软件平台之争将牵动更多软件公司未来的发展。对于 Borland 来说，努力形成的软件供应链是否能够在两大平台下杀出一条血路？Borland RAD 部门几乎倾全力打造的.NET 产品 Galileo 是否仍然能够掳获开发者的人心？如何在 Microsoft 主导的.NET 平台和 Visual Studio.NET 竞争？Borland 在庆祝成立 20 年之际是否仍然能够在未来拥有一席之地？这都将牵动着 Borland 未来的命运。

我并不能预测未来。不管 Borland 在 2003 年推出的产品是否能够成功地带领 Borland 走向下一个繁荣的 10 年，Borland 以小搏大，分别在 Java 和.NET 平台和世界一级软件大厂竞争的勇气已经值得佩服了。在许多比 Borland 更大的软件厂商纷纷在这场激烈的

竞争中败下阵来之际，Borland 却靠着坚强的技术能力和创意十足的产品屹立不倒。
在过去十几年中，Borland 和 Microsoft 之间高潮不断的竞争好戏又将在 2003 年再次上演。只是过去竞争者众，如今却只剩下 Borland 和 Microsoft。2003 年巅峰之战的结果将在以后的岁月中揭晓，不管是谁胜出，Borland 已经在撰写.NET 的传奇故事。也许数年之后，还会有人把 Borland 发展.NET 产品的内部秘密以及在.NET 平台奋斗的故事再次揭露于世吧。

^v^v^v^v^v^v^v^v^v^v

附录：　Borland 大事记

1983.5.2　Philippe Kahn 和 Anders Hejlsberg 在美国 Scott Valley 共同成立 Borland International 公司。
同年 11 月，发布 Turbo Pascal，Borland 一举成名。

1984　发布内存常驻工具软件 SideKick，成功打入消费软件市场。

1985　发布 Borland 第一个，也是最后一个 Basic 开发工具产品：Turbo Basic。
从 Ansa 公司购得 Paradox。

1986　发布 Turbo Prolog。

1987　发布 Turbo C 1.0，提供 C 语言开发集成环境工具。
Turbo Pascal 4.0 也在这一年推出。

1989　在购入 Ansa 公司(1987 年)后，推出 Paradox 3.0。

1990　在 Turbo C 基础上推出 C++开发工具 Turbo C/C++。
该产品也被称为 Borland C/C++。

1991　购入 Ashton-Tate 公司，获得 dBase。
发布电子表格软件 Quattro Pro。该软件生不逢时，在与 MS Excel、Lotusl-2-3 残酷竞争之后，最后败给 Excel，被 Novell 收购。

1992　发布 Borland C/C++3.0。这是第一个基于 Windows 的 C++集成编程环境软件。
在 Borland C/C++3.1 中加入了 OWL 作为核心。
兼并 Ashton-Tate 公司，推出 dBase 1.0。
同时也取得真正的 RDBMS——InterBase。

1993　匆匆推出旨在与 Visual C++对抗的 Borland C++4.0。该版本尽管有不少创新，但最终被证明是失败的。
发布 DOS 版本的 dBase-IV 2.0，并被证明是可靠的数据库产品。

1994　发布 dBase for Windows 5.0。虽然承袭 dbase 名号，但其核心却是 WordTech 公司的 Aragon for Windows。Borland 在 Comdex 上了解到 Aragon for Windows 后，通过并购获取了这项技术。而真正的 dBase 只留下调试器于 dBase 5.0 中。
在面临 C/C++战场三面夹击的情况下，推出以 OCF 技术支持 OLE 的 Borland C++4.5。此役之后，Microsoft 占据 C/C++市场半壁江山，而 Borland 的市场占有率却滑落到 30%，开始走下坡路。

1995　Philippe Kahn 因经营不善辞去 CEO 一职，但继续留任董事会成员。CEO 由 Gary Wetsel 接任。而 Philippe Kahn 则由于产品理念分歧的缘故，自己开办 Starfish Software 公司，致力发展 SideKick 等应用软件。后 Starfish 在无线通讯等领域颇有建树，并最终被 Motorola 以数千万美金的高价收购。
同年情人节发布 Delphi 1.0。该产品一炮而红，成为扭转 Borland 命运的转折点，也成为众多 Delphi 开发者的"初恋情人"。

1995　发布品质最好的 Paradox For Windows 7.0。一年后，Paradox 被卖给 Corel

公司。
同年 11 月，由于无法忍受 Philippe Kahn 对 Borland 的一再挖角，董事会决定将其逐出公司。

1996 发布以 32 位编译器为核心，并且大幅支持 C/S 编程的 Delphi 2.0。
发布 IntraBuilder 1.0，是业界第一个数据库 Web 工具。但由于太过先进等原因，叫好不叫座。一年之后，Borland 宣布放弃 IntraBuilder 开发。
继 Philippe Kahn 之后，Anders Hejlsberg 也离开了 Borland。
Delbert Yocam 随即成为 Borland CEO。
购入中间件 Entera 技术，准备进军大型的基于 UNIX 平台的软件市场。

1997 发布 Delphi 3.0。该版本较好地平衡了 COM/DCOM 支持和分布式多层架构，并成为全球热卖的产品。
发布 C++Builder 1.0。尽管 Borland 并没有作太多的市场推销活动，但该工具推出之后仍广受好评，被誉为"C++开发者天堂"。C++开发者终于可以和 Delphi 开发者一样，通过 RAD 的方式进行编程。
Borland 委托 Dr. Niklaus Worth 研究小组开发出效率优良的 Java JIT 编译器，随后发布 Borland 第一个 Java 工具：Open JBuilder 1.0，但市场反应不如预期。
并购 Visigenic Software 公司，取得 CORBA 技术，并很快据此开发出 VisiBroker。通过与 Netscape 的合作，成功地向大众展示该技术。
发布 dBase 7.0。产品虽好，奈何时势不再。

1998 宣布公司更名为 Inprise，希望籍此表达 Integrating the Enterprise 的公司发展目标理念。改名行动以及"打造行销导向 Borland"的计划最终都一败涂地。
发布匆匆研发的 Delphi 4.0，在市场遭到惨败。Delbert Yocam 的好大喜功再次让 Borland 付出沉重代价。
JBuilder 2.0 发布，Borland 的 Java 开发工具渐入佳境。

1999 在和 Borland 就"Brain Drain"事宜展开诉讼并发现局势不利之后，Microsoft 提议庭外和解并投资 Borland。
Delbert Yocam 被解雇，Dale Fuller 任 Borland CEO。
发布 Delphi 5，一扫 Delphi 4 带来的耻辱。
JBuilder 3.0 发布。一年后的 JBuilder 3.5 纯以 Java 打造而成，毕其功于一役，充分体现了 Borland 的实力。
出售 dBase 予 Ksoft(后更名为 dBase Inc.)。

2000 发布 JBuilder 4.0，是继 JBuilder 3.5 的乘胜追击之作。推出之后很快就成为市场的霸主。
和 Corel 的并购案失败。

2001 发布 JBuilder 5.0，大幅改变人们对 JBuilder"不适用于团队开发"的印象。
同年底发布的 JBuilder 6.0，整合 UML 和 Extreme Programming，更是支持 EJB 的最好开发工具。

2002 发布 JBuilder 7.0，最终奠定在 Java 开发工具市场唯我独尊的地位。
并购 VMGEAF,公司，获取 OptimizeIt，并将其整合到 JBuilder 产品线。
同年 10 月，并购 Starbase 公司，准备提供软件应用平台。
随即，对 TogetherSoft 的并购案，给业界带来极大震动。
发布 Delphi 7，被认为是 Windows 平台原生开发工具向.NET 平台开发工具

过渡的一代产品。

随着 Delphi 6、7 的发布，Kylix(Delphi for Linux)也不断更新，品质也在持续的精进中。

(整理/韩磊)

写在最后

1983 年 5 月 2 日 Borland International 正式成立

1983 年 11 月 20 日 Borland 正式推出第 1 个代表产品--Turbo Pascal 1.0

2003 年 2 月 5 日《Borland 传奇》一书正式完成。

谨以本书向一个成立 20 年的传奇软件公司致敬，因为她为许多人留下了一生美好的回忆

李维

北京市朝阳区北三环东路 8 号
静安中心 26 层 2662
《程序员》杂志社
读者服务

尊敬的读者：

感谢你购买 P&C 出版的计算机图书，为了更好的为读者提供服务，请详细填写回执卡(背面)各栏，邮寄或传真回本公司，我们将根据您的宝贵意见不断追求进步，并不定期提供最新出版信息。

我们的联系方式：

地址：北京市朝阳区北三环东路 8 号静安中心 26 层 2662
邮编：100028
客服信箱：book@csdn.net
网址：http://www.csdn.net
电话：010-51661202-279
传真：010-84540263

(本书如有破损、缺页、倒装请寄回更换)

【全书完】

【致谢】

感谢 taopian 提供的源书，他的首印纪念版已经被我糟蹋的不成样子了，390 页的书已经被翻成了新华字典那么厚，实在不好意思，看来是要再买一本来还他了。大家想买书也可联系他。当然还要感谢偶的狗狗 mm 这些天的支持和协助^_^

本来是想做电子书的，后来觉得还是只把文字介绍给大家，原书所附的大量图表和照片都没有扫描，这中间包括了很多大牛的照片和统计数据，建议有兴趣的朋友一定买一本看看。从 2003 年 4 月到现在本书已经印刷了六次，作为软件公司的经典案例和技术发展的亲身见证，深受读者好评。

本电子版使用汉王尚书六号中文 OCR 软件及 Microtek SM3830 扫描仪制作。

【再次版权申明】

pcq@freecity 谨识

2004 年 01 月 11 日临晨 5 时 32 分