关于 Borland C++BuilderX 的一些问题的回答

左轻侯

2003.10.20

在 Borland 发布 C++BuilderX 前后的这一段时间里，我在 CSDN 上陆续看到了不少关于 C++BuilderX（以下简称 CBX）的讨论。作为 Borland 中国公司负责快速开发工具和开发者社群关系的工程师，我认为我有必要对大家的一些疑问和猜测进行回答。

需要说明的是，我并没有得到 Borland 公司的正式授权来做这个回答，因此下列文字，不一定等同于 Borland 公司的官方观点；我只是把从我这个角度看到的一些东西，尽量和大家进行沟通。如有错误或不当之处，还请不吝指正。

CBX 诞生的背景

就我所见，网上对 CBX 的问题主要集中在这几个方面：Borland 公司出于什么动机开发 CBX，它与 C++Builder 的关系怎样？CBX 的 framework 详情如何，与 VCL 的关系怎样？Borland 公司对 CBX 的定位和长期计划是什么？

要回答这些问题，首先必须从开发 CBX 的背景说起。

如大家所知，Borland 公司在 2003 年发布了一系列的新产品，其中在 Windows/.NET 平台的开发工具就有三个：Delphi.NET、C#Builder 与 CBX。这些新产品的推出，表示着 Borland 公司在 Windows 平台下的开发工具阵营发生了大的变革和分化。而这些变化的发生，又根源于操作系统本身的一次深刻的革命，即从 Win32 平台到.NET 平台的全面过渡。

根据我的所见所闻，国内有不少技术人员仍然对这一转变没有足够的认识。简言之，Microsoft 公司已经下定决心，将操作系统从 Win32 全面转向.NET；逐渐地，Microsoft 不会再提供新的 Win32 API，未来操作系统的新功能将全部以.NET SDK 的方式提供；从企业级服务器，到桌面系统，再到嵌入式系统，.NET 将是 Microsoft 公司解决方案中的统一的编程模型。虽然这个转变缓慢、艰难、痛苦，但这是历史潮流的方向。无论你是否喜欢，除非你放弃在 Microsoft 平台下的开发工作，否则就只能适应它。

这一转变给 Windows 平台的开发工具厂商带来了很大的挑战，Borland 公司也不例外。目前的形势，就和当年从 DOS 过渡到 Windows 的情况差不多，整个操作系统与编程模型与发生了彻底的变革。Borland 的 Delphi 和 C++Builder 都深深植根于 Win32 平台，面对这一转变，必须作出积极的反应，才能求得生存和发展。Borland 公司确实也做出了这种反应。

正象大家看到的那样，Delphi 的方向是和操作系统一起转向.NET 平台，其结果就是 Delphi.NET 的诞生，它将和全新的 C#Builder 一起，成为 Borland 公司在.NET 平台下的主力开发工具。（但 Delphi 仍然保留了在 Win32 下进行开发的能力，因为 Win32 平台下的应用程序在未来相当长的一段时间内会继续存在。）

至于 C++Builder 的方向，在 Borland 公司内部进行过讨论，结果大家应该已经猜到了，那就是 C++Builder 并没有向.NET 靠拢，而是坚持走原生开发工具的道路，并且扩展到了多个平台、多个编译器。原来 C++Builder 和 Delphi 一起，由同一个 RAD 开发小组进行开发，这也是为什么 C++Builder 和 Delphi 总是交替升级版本的原因之一。Borland 在做出这个决定以后，成立了新的 C++/Mobile 开发小组，C++开发工具从此和 Delphi 分道扬镳。

Borland 为什么做出这一决定，有多种原因。首先，原生 C/C++开发仍然是最重要的市场之一。在 Microsoft 以外的平台上，无论是服务端的 Unix/Linux，还是越来越热门的嵌入式系统，C/C++仍然是挑大梁的语言。即使在 Microsoft 平台下，往.NET 的过渡完成以后，

在某些底层的开发中 C/C++仍然具有不可替代的地位。回顾 Borland C++的光辉历史，展望未来的战略布局，Borland 都不能够放弃这一市场。其次，C++语言有其特殊性，不适合向.NET 移植。熟悉.NET 平台的朋友可能知道，Delphi 的许多特性与 C#语言相当类似，从对象引用模型，到单根继承结构，以及 PME（property, method, event）模型，无不如此（毕竟 C#和 Delphi 同出自于 Anders 之手）。对 Delphi 进行扩展和调整，以符合.NET CLS（Common Language Specification），并不会损害 Delphi 的风格，反而有利于一个更强大的 Delphi。而 C++则具有完全不同的开发思想，如果要符合 CLS，必须对 C++进行大幅度的修改，结果是 C++不再成其为 C++了。Visual Studio .NET 中提供的 Managed C++，在 C++社群中恶评如潮，就是一个例子。

在这种情况下，C++Builder 不是转向.NET 而是另辟蹊径，不能不说是明智之举，甚至可能是唯一的选择。

兼容性和 VCL

CBX 的向下兼容性，特别是与 VCL 的兼容性，在网上是备受瞩目的问题。

这个问题可以明确地回答如下：在 CBX 中，没有 VCL。CBX 内置的 framework，是一个全新的、100%由 C++开发的、跨平台的 framework，其中的 GUI 部分，基于第三方的跨平台的 wxWindows 组件库（GUI 及相应的 RAD 功能在 CBX 1.0 中还没有，将在 2.0 中提供）。

由于历史的原因，C++Builder 的 framework 使用了 Delphi 的 VCL。这使得 C++Builder 的程序员往往发现自己处于一种相当尴尬的境地：他不得不与另外一种语言写成的 framework 打交道。而且，为了和 VCL 兼容，C++Builder 中增加了象__property 这样的非标准语法，从而使 C++Builder 看上去象一个 Delphi 的复制品。因此，网上有“ BCB 的高手必定是 Delphi 的高手 ”的说法。从 C++Builder 诞生的那一天开始，用 C++重写整个 framework 的呼声就没有停止过。

CBX 的诞生，使得这个愿望终于实现了，而且这个 framework 甚至超过了我们的期望：它不但完全使用标准 C++写成，而且支持跨平台和交叉编译（Cross Compilation，即在一个平台下编译生成另一个平台下的可执行代码），同时也对某些专业领域，例如嵌入式开发，提供了专门的支持。另外，它还能够方便地挂接 ACE、Loki、Boost 等第三方的 C++库。

但是，这种改变也有不利的一面，那就是 CBX 不再向下兼容 VCL。这是许多技术人员表示不满的原因，因为原来的 C++Builder 已经与 VCL 密不可分，CBX 抛弃 VCL，将会导致 C++Builder 的程序员必须重新学习新的 framework，并且给原有代码的升级带来很大的麻烦。许多质疑都指向同一个问题：为何 Borland 没有实现 VCL 向 C++的平滑过渡？

这个问题确实是一个很大的问题，可以追溯到当初 Borland 在开发 C++Builder 的时候，为什么没有用 C++开发自己的 framework，而是不惜修改 C++，让它使用由 Object Pascal 编写的 VCL；以及为什么在 C++Builder 的 6 个版本的升级过程中，一直没有用 C++重写 VCL；甚至涉及到 Delphi 的设计思想与 C++和 Object Pascal 的区别。

一般认为，Borland 当初是出于时间和资金的压力，才出此下策，让 C++Builder 嫁接到基于 Object Pascal 的 VCL 上。我曾经也持这种观点，但是后来在逐步深入学习 Delphi 和 C++的过程中，才慢慢发现，这一举动可能有更重要的原因，那就是由于 C++和 Object Pascal 的差异，将 VCL 移植到 C++有着几乎不可克服的困难。

以所谓“ 虚拟构造函数 ”为例，在 C++中，构造函数不能够是虚拟的。C++之父 Bjarne Stroustrup 曾针对此问题进行过讨论：“ 虚拟调用是一种能够在给定信息不完全(given partial information)的情况下工作的机制。特别地，虚拟允许我们调用某个函数，对于这个函数，仅仅知道它的接口，而不知道具体的对象类型。但是要建立一个对象，你必须拥有完全的信

息。特别地，你需要知道要建立的对象的具体类型。因此，对构造函数的调用不可能是虚拟的。”但是在 Object Pascal 中，构造器（constructor）却可以是虚拟的。原因是，Object Pascal 的对象模型与 C++不同。在 Delphi 中，有所谓的“类类型”（TClass），这是一个描述类的相关信息的类，类似于 C++中的 type_info 类，但比后者强大得多。更重要的是，Object Pascal 会根据类类型来决定创建的对象，通过这种机制，它避免了“给定信息不完全”的问题，从而轻易地实现了虚拟构造器。

同时，这种机制也导致了 Object Pascal 和 C++中的 RTTI 的巨大差异：标准 C++中的 RTTI 只能得到类名称（编译器可以提供额外的 RTTI 支持，但很有限），而 Object Pascal 中的 RTTI 不但可以得到类名称、类继承关系、对象实例大小，还可以获得类的属性、方法、事件等信息。Delphi 中的这种 RTTI 类似于 Java/C#中的 Reflection，或者也许可以反过来说，Java/C#中的 Reflection 受到了 Object Pascal 的影响。Object Pascal 之所以能做到这一点，又离不开它的单根继承结构：由于 Object Pascal 中所有的类都继承自 TObject，因此可以轻易地添加 RTTI 信息。在这一点上，C++和它又有着本质的区别。

在 VCL 的源码中，大量地使用了类类型和虚拟构造的技术来构建庞大的继承树，而强大的 RTTI 则更是整个 Delphi 组件模型的基础。如前所述，由于 Object Pascal 和 C++在语言上的差异，要用 C++依样画葫芦地重写一遍 VCL，根本就是不可能做到的事情。说到底，这已经是业界两种不同的编程思想之间的差异，不是简单地折衷一下就可以调和的。

让我们把这件事放到更大的历史背景中来看。Anders 开发 Delphi 的时候，正是 C++借助 OOP 思想快速成长，而 Pascal 语言日趋没落的时候。Anders 不但果断地突破了标准 Pascal 的限制，将 OOP 全面引入到新的 Object Pascal 中，而且针对 C++在实践中暴露出来的一些问题，有意识地进行了弥补和改进，以构建新一代的 framework。这一行动的结晶就是 VCL 的诞生。事实上，Object Pascal/VCL 的许多经验，后来也被 Java/C#所汲取，用于构建 J2EE/.NET 中的组件架构。从这个过程，我们可以看到编程思想与编程语言的演化。

当然，这并不是说 C++就已经没落过时。事实上 C++也一直在不断的成长中，上述的问题，可能有别的解决途径。拿“虚拟构造函数”来说，Bjarne 本人就指出，可以通过工厂模式（factory pattern）来间接实现。用 C++构建一个功能上类似 VCL 的 framework，是完全可以做到的，但是实现后的产物，肯定与 Object Pascal 写成的 VCL 大相径庭。其实这已经与 C++Builder 程序员们的初衷相去甚远了。我认为，这才是 Borland 权衡利弊，最后不惜损害 C++的兼容性，把 C++Builder 嫁接到 VCL 上的主要原因。

考虑到这些因素以后，VCL 没有在 CBX 中延续其生命就是很容易理解的了。另外一个重要原因是，VCL 是在 Windows 平台上发展起来的，因此很多地方都只考虑到跟 Windows 平台的集成。Borland 既然已经决定另起炉灶，打造一个完全跨平台的 C++开发工具，自然会趁此机会，摆脱与 VCL 扯不断理还乱的关系，用全新的、100%C++的 framework 取而代之。

IDE

CBX 将使用 JBuilder 的 IDE，这一消息现在已经得到了证实。为什么 C++开发工具却要使用一个用 Java 写成的 IDE？这也是引起许多朋友质疑的地方。

CBX 选用 JBuilder 的 IDE，至少部分是由于统一 IDE 的需要。众所周知，Borland 作为一个平台中立的厂商，在很多平台上都有自己的开发工具，随着这些开发工具的发展与分化，统一 IDE 已经是不可避免的趋势。据称，Borland 以后将只维护两个基本的 IDE，即基于.NET 的 Galileo 和基于 Java 的 PrimeTime。C#Builder 和 Delphi.NET 基于前者，JBuilder 则基于后者。CBX 作为一个以跨平台为重要特征的开发工具，选择后者作为 IDE 是顺理成章的事情。

事实上，当我们把眼界放开阔后，就会很容易地发现，统一的 IDE 已经成为业界的一股潮流。Microsoft 在 Visual Studio.NET 中提供了统一的 IDE，同时支持 VC#、VB.NET 和 VC++.NET 等多种语言。IBM 主持下的 Java IDE 项目 Eclipse，宣称以后将以 CDT（C/C++ Development Toolkit，用于 Eclipse 的 C/C++开发工具箱）为重要的发展方向。SUN 的 NetBeans 则早已将 C/C++纳入支持的范围。可以说，用运行于 Java/.NET 这样的受管制代码（Managed Code）之上的 IDE 来作为 C/C++的开发环境，并非 Borland 的首创，反而是顺应潮流之举。

另外，以我的个人意见，JBuilder 的 IDE 确实要比 Delphi/C++Builder 强上很多。虽然 JBuilder 背有速度缓慢的恶名，但是在硬件条件能够保证的情况下，它的 CodeInsight、Code Template 以及 Refactoring 等功能，比 Delphi/C++Builder 明显领先，更不用说对 UML 的支持以及和 Together Edition for C++的集成。因此，我认为 CBX 借用 JBuilder 成熟强大的 IDE，是一件好事。

C++的未来

让我们暂时抛开 CBX，来看看 C++语言本身的一些东西。

毫无疑问，C++仍然是最主要的工业语言标准之一，特别是近几年来，C++语言出现了蓬勃的发展，各种新技术和新概念层出不穷，世界范围内的 C++社群也是蒸蒸日上。但是，勿庸讳言的是，C++的地位确实受到了来自 Java/C#的有力挑战。在应用领域、特别是在高端的应用领域中，Java 正在逐渐取代 C++成为主流。

导致这种情况的原因是多样的，但最主要的原因有两个。一个是 C++的标准推出太晚，直到 1998 年 ISO C++标准才正式推出，在此之前，各种风格的 C++版本把时间浪费在内耗上，将大片的市场拱手让给了 Java。另一个更重要的原因是，虽然 ISO C++标准的制定统一了 C++的语言，但是却没有统一 C++的 framework。虽然 C++标榜自己是平台无关的语言（它的确也是），但是对于同一个问题，在不同的平台下有各种不同的解决方案。C++自己的标准库只是一个语言的 framework，而不是一个应用的 framework：在 I/O，多线程，Socket，GUI，数据对象模型等等常见的问题上，开发者们不得不要么自己封装特定平台的 API，要么寻找难以保证质量的第三方类库。没有统一 framework 的 C++，就象没有 VCL 的 Delphi，没有 JFC 的 Java，没有.NET framework 的 C#。因此毫不奇怪，C++在应用领域无法与 Java/C#抗衡，而逐渐退守到底层编程。

去年 10 月，C++之父 Bjarne Stroustrup 博士访华的时候，我曾经当面向他提出过这个问题：为什么 C++没有一个统一的、跨平台的、面向应用的 framework？C++的标准库是否有向这个方面努力的计划？Stroustrup 博士严肃地回答了这个问题。他说，与 Delphi、Java、C#不同的是，C++是一种厂商中立的语言，它的标准掌握在 ISO C++委员会手中。这固然保证了 C++的纯洁性，但也导致了没有厂商投入大量的资源来开发一个统一的 framework——即使有厂商这么做了，也不一定能够保证它的权威性。最后，他建议我可以考虑利用一些优秀的第三方库。

由于这个问题涉及到很多技术之外的因素，甚至涉及到 C++本身的目标和定位，看来没有也不会有更好的解决办法了。谁知没过多长时间，事情突然出现了转机。Borland 的 C++BuilderX，不正是在试图扮演一种这样的角色吗？

从背景来看，Borland 作为老牌的开发工具厂商，在 C++编译器和 framework 的开发方面的技术和经验是无可置疑的。而 Borland 往日最大的竞争对手 Microsoft 已经明显地将重心转移到.NET 和 C#，不会再大幅度地发展原生的 Visual C++，更不会去开发面向应用的 C++ framework。因此，Borland 等于接收了 Microsoft 转型后留下的真空。同时，Borland 也是业界具有领导地位的软件公司中，唯一一家完全平台中立的公司，这和 C++的宗旨是相称

的；由它来完成一个真正跨平台的 C++ framework，真是再合适不过了。或者反过来说，除了 Borland，业界很可能不会再有一家厂商会来承担这一工作。

从目前看到的产品来说，CBX 确实在各方面都达到了目标。统一的、跨平台的、符合 ISO C++标准的 framework（包括用于 RAD 的 GUI 解决方案）；跨平台的 IDE；支持多种编译器；支持交叉编译；跨平台的数据库访问引擎（dbExpress）；对移动设备开发的解决方案；对 Boost、ACE、Loki 等第三方 C++库的开放式支持；再加上与 Together Edition for C++BuilderX 的集成，引入了令人激动的 MDA 开发方式；即使是 Stroustrup 博士，恐怕也难想象比这更好的 C++了。

第一次见到 CBX 的演示的时候，确实很给人震撼的感觉。我曾经和包括公司同事在内的一些朋友热烈地讨论过，一旦 CBX 成功，会产生什么样的后果？在一个统一的、跨平台的 framework 的影响力之下，C++是否会重返应用领域？Java/C#的市场是否会因此受到冲击？不仅是 Borland C++这个令人难忘的名字，而且是整个 C++世界，是否都会契此机会东山再起，重振声威？一切都有待市场的检验。