又一封公开信？

lextm

March 28, 2006

Powered by LaTeX.

1 写在前面

CnWizards 的稳定版本还停留在去年的十一月－－至少从首页上面的文字来看，很多用户有这样的感觉。那个十一月，也是我在 CnPack 的论坛上面贸然发布那封公开信的时候。

这里，我是不是在写公开信 2.0 呢？我感觉有些不是了。BDS 2006 的发布和 Borland 要分家等等一系列的新消息是这样的让人烦躁不安，不免要写一点东西一吐为快好了。

我自己做的项目 Code Beautifier Collection 本周一也终于发布了革命性的 WalkPace 版本。这篇文章的很大一部分将在 CnWizards 和 CBC 的比较中进行。

2 我和 Code Beautifier Collection

在公开信里面，我已经提到了 Code Beautifier Collection。从它的 1.0 版本，就是 JCFExpert，开始就是我做着好玩的一个小工具。公开信发表的时候，CBC 还是 2.2 版本的中期。

2.1 开始

在最初的几个版本里面，我做的工作主要是从 Sharp Builder Tools 0.9 的架构里面分离出来一个 CodeBeautifier 部件，随后加入呼叫 JCF 以格式化 Delphi 代码的功能。这就是 JCFExpert 和 CBC 2.0 的任务。

在 2.1 版本中，CBC 升级到了 SBT 3.1 的架构。同时，我开始考虑支持多个 BDS 版本的计划。不过计划太多，自己的能力又十分有限，正式发布的 2.1 版本出了许多的毛病，所以，我下定决心尽快地推出 2.2。2.2 版本的设计也十分顺利，到了正式发布的时候，基本上我计划中的功能全部做好了，而且主要功能方面没有致命的 bug 了。但是，由于当时缺少大量开源工具的使

用，NUnit，NANT，Inno Setup 等等，发布过程乱糟糟的，让我十分头痛。当然，结果是我不得不连续发布了 2.2 和 2.3 两个版本（这里的 2.3 实际上是 2.2 的修正版）。

发布 2.2 的日子也就是我完成公开信的时候。由于 CnWizards 和 CnBeautifiersWizard 不兼容了，我想给 CnWizards 加入一个使用功能的计划暂时停下了。而同时，从公开信的反映来看，由于生活工作的缘故许多重要的 CnPack 开发组成员不可能在短期内实现一个“理想”中的架构。况且，仅仅是 CnWizards 就有许多的功能，一个架构性的革命简直会是一场灾难。

2.2 重新上路

我也不打算等待了，而是稍作休整之后就带着我的 CBC 出发了。CBC 2.4 将要达到的目标第一个就是支持新发行的 Delphi 2006。我不知道 OTA for Win32 的情况如何。总之由于采用 OTA for .NET，CBC 升级几乎只是换了一个 Reference 那么简单。而且很明显 Borland 花了时间改正了一些 OTA 部分的 bug。第二个目标则是利用 BDS 2006 内置的十分顺手的 Together for C# 来做重构。

重构这个东西，我听说已经很久，但是几乎没有怎么做过——本科的课程设计需要的仅是一些小东西。至于设计模式，我则是买了一本书却从没有细细看过。可是，这一次 CBC 的开发让我有机会去体会种种方法论的巨大力量。

当然，另一个重要的事件是我开始了博客。最初的文章很少，甚至 CBC 的新闻也很少，不过现在情况大有改观。

CBC 路线图和 BigFace 开发代号的出现，让我感觉到一种“正规”的味道了。不过，事情也不是一帆风顺。这个版本最后直到我寒假归来才算做完，并且发布。BigFace 版本的巨大意义在于我开始尝试综合使用设计模式和数据结构来降低了代码之间的依赖性，为 WalkPace 版本中把整个项目分割为 Framework + LeXDK + Plus + Minus 的插件架构做好了准备。

2.3 WalkPace

WalkPace 是刚刚发布的 CBC 2.5 的开发代号。

由于逐步走上了正轨，所以 WalkPace 的开发十分的顺利。这一个版本之所以特殊，是因为我最终利用 .NET 平台的 Reflection 技术初步实现了我在公开信里面提到的插件架构。

3 WalkPace与CnWizards的架构比较

Table 1: 比较

架构特征	CnPack CnWizards	CBC 2 WalkPace
OTA	for Win32	for .NET
加入新特性	向原工程中加入新的单元	制作新的 Plus 工程
是否重新编译	整个 CnWizards 项目需要重新编译	不需编译 Framework 和 LeXDK。编译 Plus 工程，然后加入一个 .plus 文件就可以动态加载
保存数据方式	读写注册表	读写 XML 文件
新特性的发布	CnPack 的全部特性打包发布	LeXDK 及 Framework 打包发布，Plus 则分开独立发布（或者放在 Plus Pack 中）
开发组织	网络时代的集约开发	社区式开发

应该说，开发 CnBeautifiersWizard 的经历使我对于 GExperts 和 CnWizards 的架构是又喜欢又烦恼。

喜欢的是简单的扩展方式，仅仅继承于一个基类，然后实现/重载相关的方法就可以得到一个新的特性，而注册 IDE，注册图片，以及保存数据的问题，都由架构本身（特别是基类）代劳了。尤其是 CnWizards 的多语言支持，十分的别致，而且比较容易使用。

然而，到了发布的时候，又发现自己没有办法单独发布源代码给用户呀。而想要把新特性加入到 CnPack 里面，显然需要一个比较长的过程，或者很可能由于 CnPack 开发组已经有了类似的工具和计划，最后无缘“打包发布”的机

会。CnBeautifiersWizard 是一个例子，但是我想还可能有其他的例子。

看了《预构（Prefactoring）》一书的书评，加上自己的一些体会，我认为 GExperts 一直没有实现一个插件架构，还是因为 Erik Berry 太忙了吧，是一个遗憾。而 CnPack 暂时没有做到，也是因为它一开始太像 GExperts 了，想要在现在这个阶段转到一个合适的插件架构上来也是十分繁杂的任务。大家都没有精力来做这个，况且，从架构的迁移中，我们不可能看到什么好处。毕竟，最终用户不在乎你架构的好坏，他们最大的目的是你尽快推出改进后的新版本，不断出他们需要的新功能。即使是对于维护项目的人员，也是希望尽可能的复用现有的代码。

所以，我做了 WalkPace，但是主要目标其实只是试试看自己可不可以做一个插件架构来玩玩。很幸运我成功了。仅此而已。当然，顺道我也发现了 LWT（LeXtudio Wrapping Technology）－－一个帮助 CBC 很容易的吸收其他 OTA 项目的小技巧（类似“北溟神功”或者“吸星大法”）。C#Builder Goodies 和 AddMany 都是很不错的例子，收录在 CBC Plus Pack I 里面。

4 社区式的开发

虽然使用了网络式的开发，我感觉 CnPack 开发组的结构还是十分类似普通的集约式开发的－－因为工程需要集约式的编译。这自然是由于现在的架构的约束。

那么 CBC 的开发呢？我觉得从 WalkPace 开始，可以进入一种社区式开发的状况。

4.1 我的作用

首先，我的作用不是十分重要了。曾经我想过自己继续的做 LeXDK，就像 Linus 控制着 Linux 内核一样。但是，由于 LeXDK 1.0 的内容实在少得可怜，而我不知道日后自己有没有这样大段的时间可以花在 CBC 上面，因此在临近发布 WalkPace 的时候，我又引入了 Minus 的概念。这一概念，进一步削弱了我的作用，也真正的确立了社区式开发的基础。

4.1.1 Minus

Minus 实际上可以看成是 LeXDK 基础库的一部分，可以实现对于 LeXDK 的扩展。例如，我把一些 BeWise 中被 Framework 和 Code Beautifiers Plus 中间用到的类库都放在了 BeWise.SharpBuilderTools.Minus 里面。

4.1.2 Core

原来计划的 LeXDK 1.0 最后被作为 Core 部分发布，仅仅包含一些关键的数据结构和基础类库。这样的话，将来也就没有必要非要发布一个很大的 LeXDK Core 2.0，仅仅添加 Minus 的数量然后打一个包发布就会很好。

如此一来我的主要工作就是定期打包发布 LeXDK 了。

4.2 你的参与

如果你想要参与 CBC 的开发，也就十分的简单。

4.2.1 改进/除错

如果你做了这样的工作，可以如同过去一样发邮件告知我。我也会在专门的地方——手册中，官方站点上——写上你的大名。

4.2.2 Plus/Minus

如果你做了新的特性，或者对于 LeXDK 做了新的扩展，我希望你可以把你编译后的 Plus 和 Minus 文件发给我（是否有源代码并不重要）。当然，你最好附上一个简单的说明，让我知道你使用的 LeXDK 版本和特性的使用说明。

我可以在官方站点上发布你的工作成果，让更多的人知道你的成果。自然，在定期发布的 Plus Pack 里面，也会打包发布最 COOL 的特性。

4.3 松散的联邦

其实这样看起来，其实 CBC 的开发组就是一个十分松散的联邦。任何人不需要事先加入什么“开发组”，或者取得什么“权限”，只要有热情和一定的开发能力就可以参与。

这里不需要一个统一的 CVS 服务器，不需要一个统一的计划方案，甚至不需要一个头头——我可以发布我的 CBC，而其实任何人都可以，只要按照 GPL 的约定来发布我所写的代码就可以了。

当然，由其他人士开发的 Plus 和 Minus 可以采用灵活的授权协议，而不局限于 GPL 协议。我想，出现共享/商业类型的 Plus 和 Minus 对于用户来说也是增加了不错的选择。

5 写在最后

其实 WalkPace 的架构绝对不是什么极好的东西，离完美就差得更远。至少问题在下面几个方面还存在：

- 我仅仅有机会使用了少数几个设计模式，而且仍然感觉核心代码不够简练和高效——虽然功能层面上面已经基本成形。
- 由于采用了 .NET C# 开发，架构基于 BDS OTA for .NET，所以，似乎只能用 .NET 平台的技术和语言来做 Plus 和 Minus——我对于 Platform Invoke 了解不多。我写的 Plus 绝大多数都是用的 C# 开发——因为我正在学习这一语言，但是如果你分析 AddMany Plus 的源码，你会发现用 Delphi for .NET 绝对是可行的。使用 Delphi for .NET 的一大好处就是可以快速的利用一部分原来用作 Win32 平台的代码。我想 CnWizards 的一部分代码可能通过 Delphi for .NET 的编译。
- 由于 WalkPace 包含的仅仅是很少的一部分基础库，所以，LeXDK 还有很多必要的功能没有实现，比如多国语言支持，Plus 管理器。但是，柔弱不表示无用。通过一定时间的努力，我想 CBC 还可以做出很多很大的东西。

期待着 CnPack 和 CBC 将来的成长。也希望中国的开源力量得到持续的增长。