基于LabVIEW的文件管理要点

Courtney Lessard
LabVIEW产品经理

NATIONAL INSTRUMENTS

批量生产测试

结构健康监测

医疗电子

太空探索

大物理应用

航空电子应用

大型系统开发 基于LabVIEW

NATIONAL INSTRUMENTS

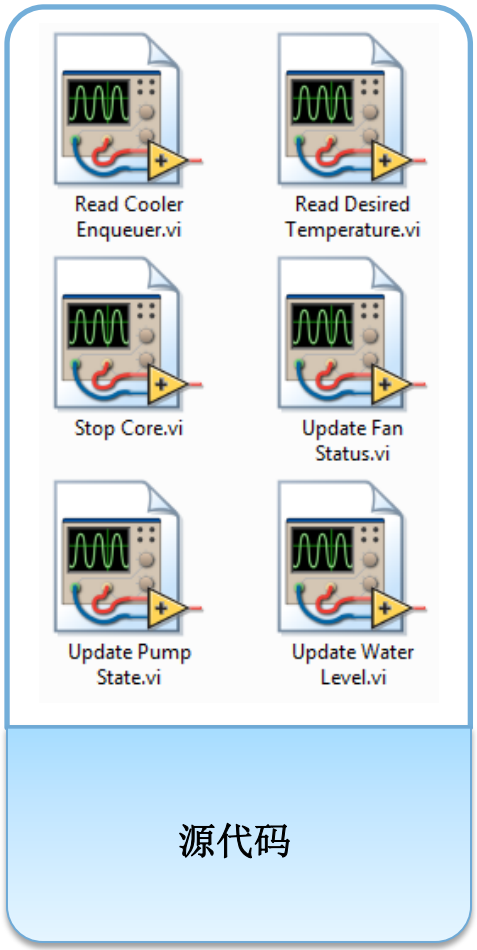
研讨会目标

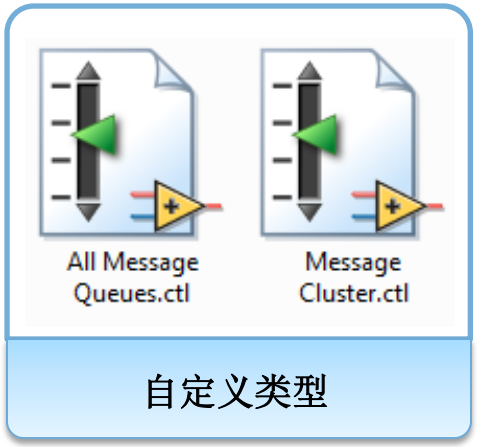
本研讨会侧重于以下方面的最佳实践：

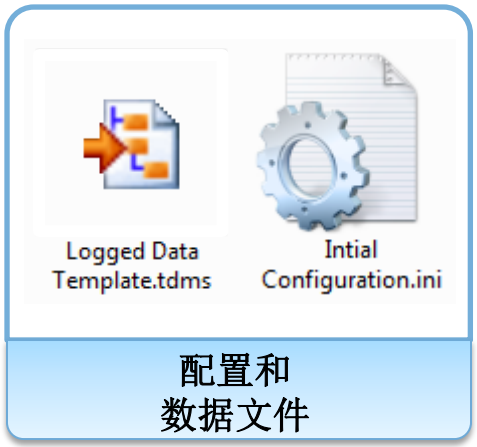
组织和管理LabVIEW应用程序
使用源代码控制管理代码库
构建和发布可复用程序库

组织和管理LabVIEW应用程序

定义LabVIEW应用程序





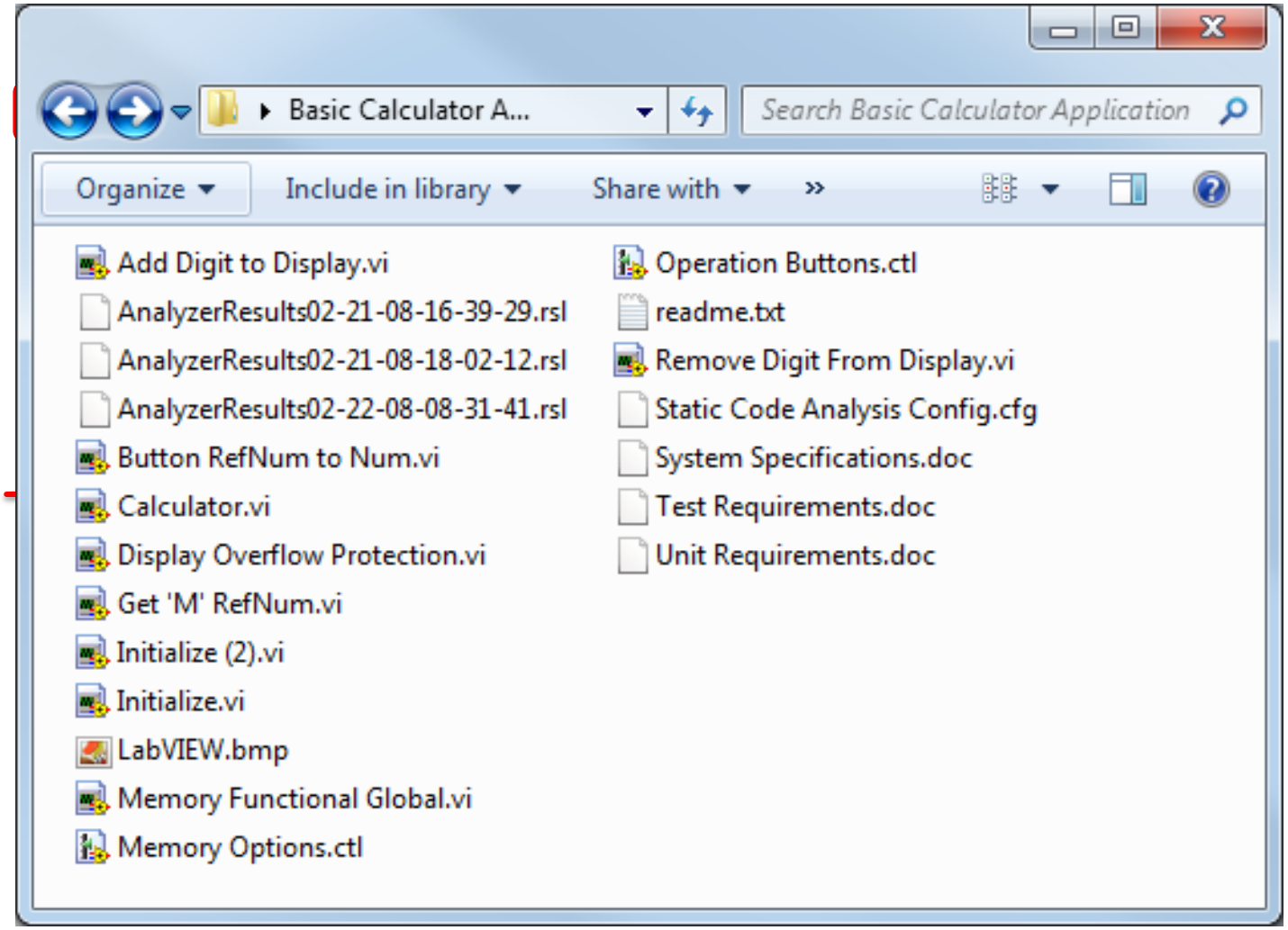


NATIONAL INSTRUMENTS

文件组织推荐

单个根目录

使用文件夹分组相关文件



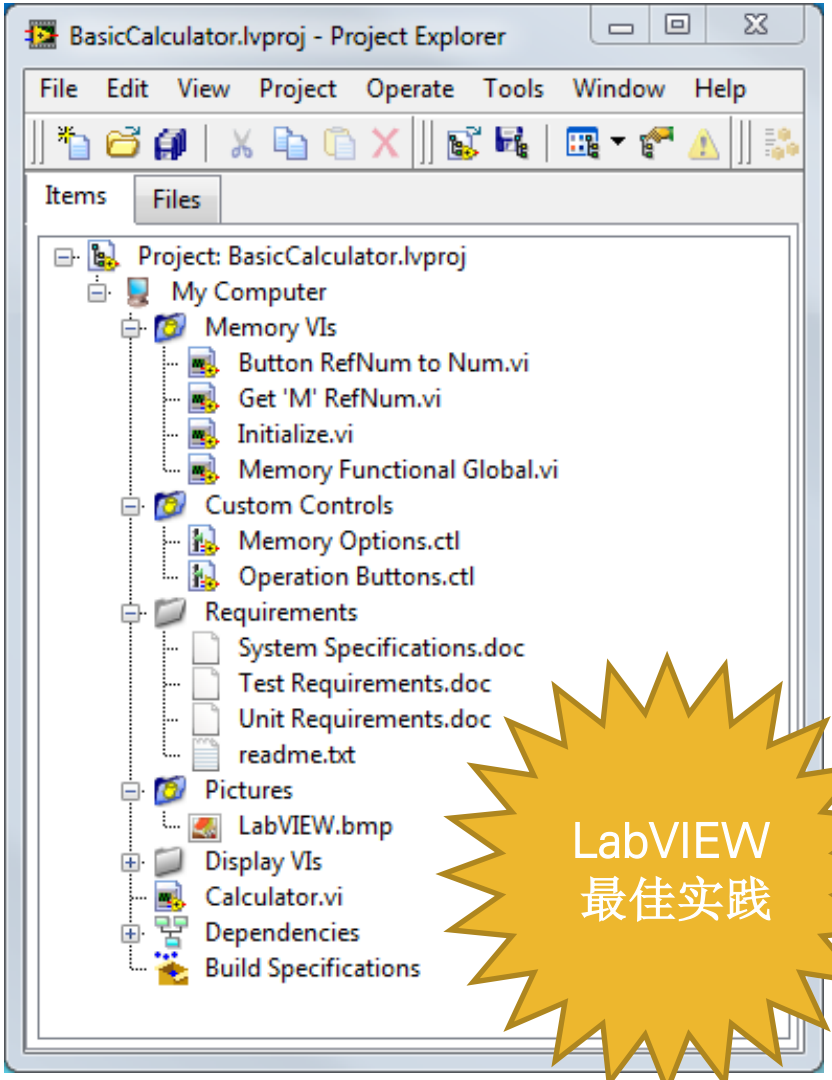
NATIONAL INSTRUMENTS

项目浏览器

提高开发者效率的工具：

- 简单的文件导航
- 移动文件时智能链接
- 集成应用程序生成器
- 部署代码至LabVIEW终端
- 访问源代码控件

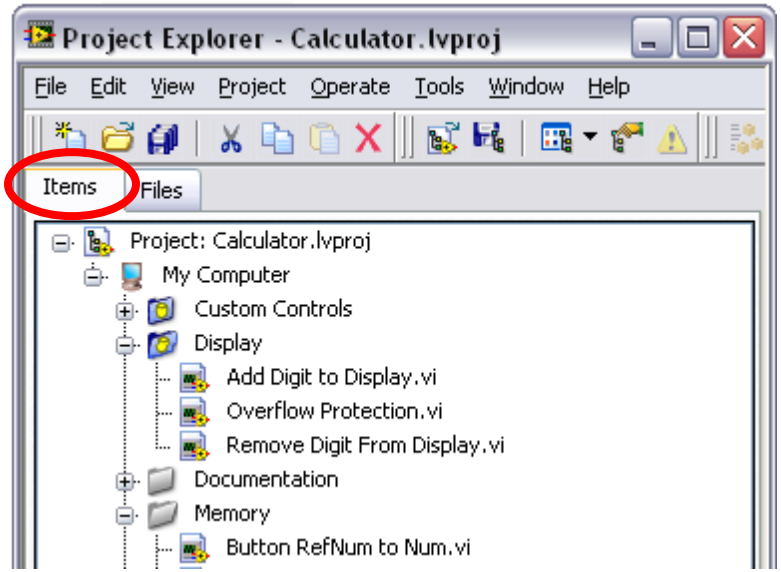
但是，项目浏览器
<u>无法</u>复制文件。



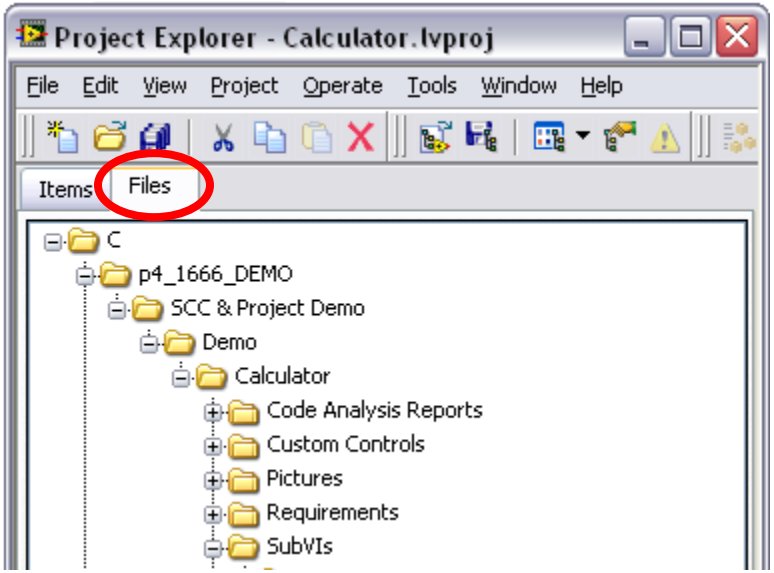
NATIONAL INSTRUMENTS

优化文件管理

项视图



文件视图



如果使用文件视图来移动文件，那么LabVIEW将会识别这一更改，然后通知调用程序并自动保留链接。

NATIONAL INSTRUMENTS

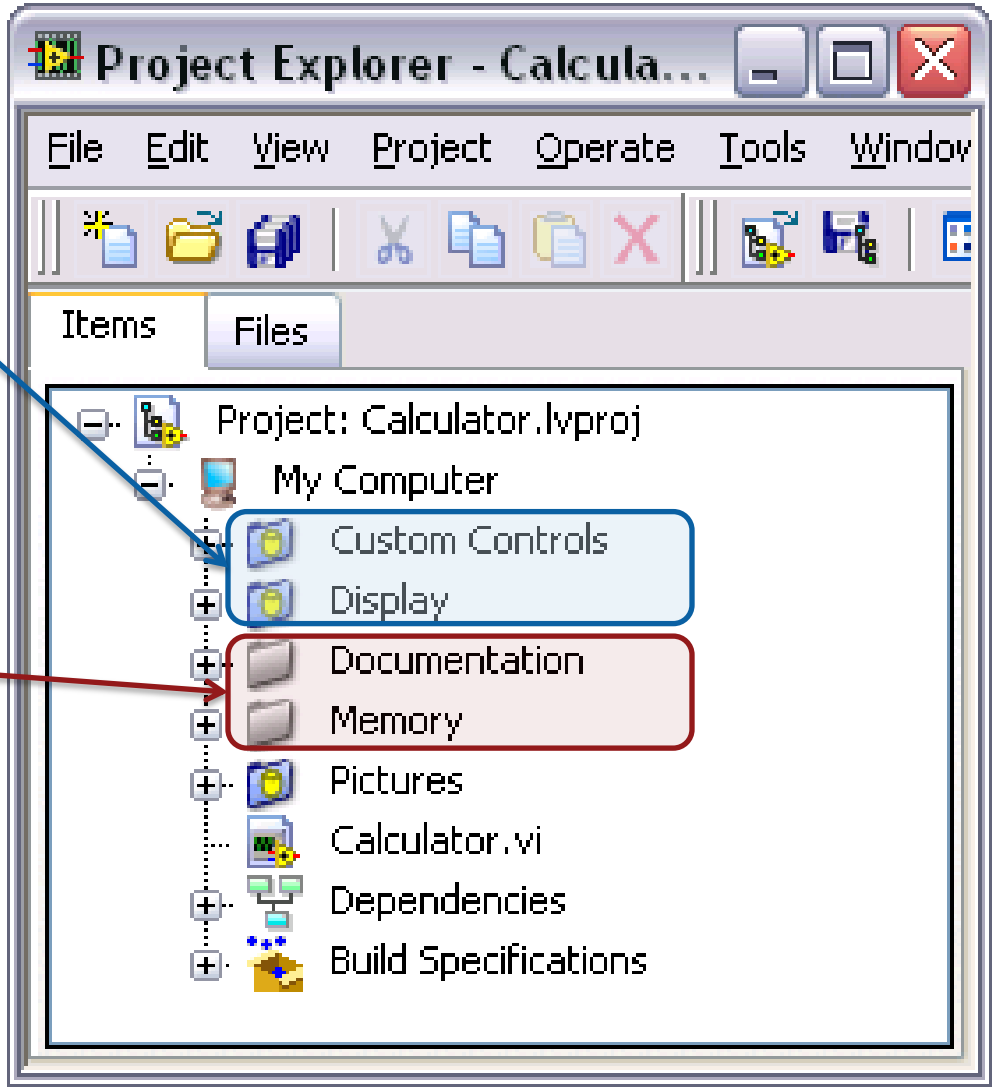
管理项目文件

自动填充文件夹

实时更新以反映磁盘文件内容的变化

虚拟文件夹

自定义文件的显示方式和位置

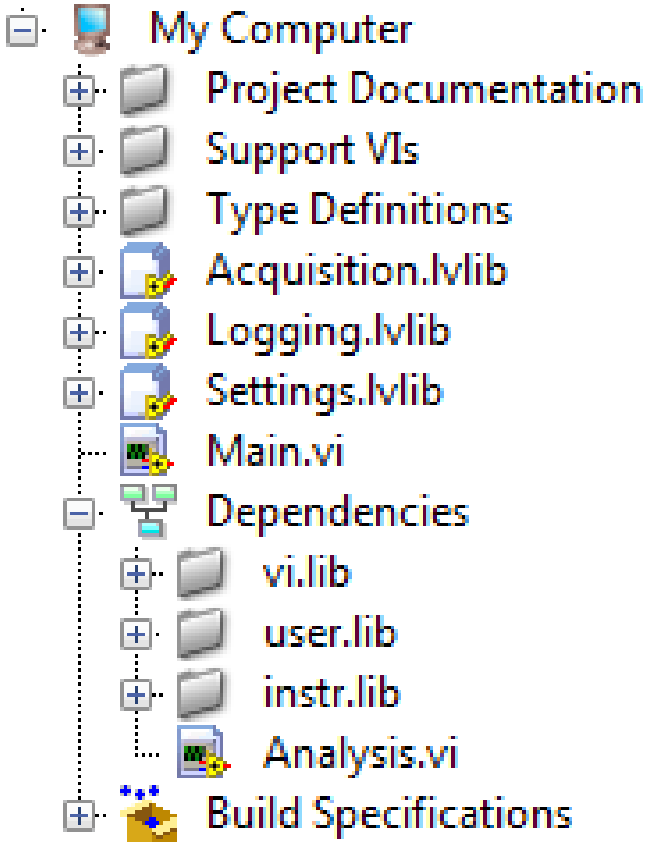


NATIONAL INSTRUMENTS

项目依赖关系

LabVIEW可自动识别项目中每个项所需的文件。

- 确保您使用的是正确版本的子VI
- 了解哪些文件应添加到项目中



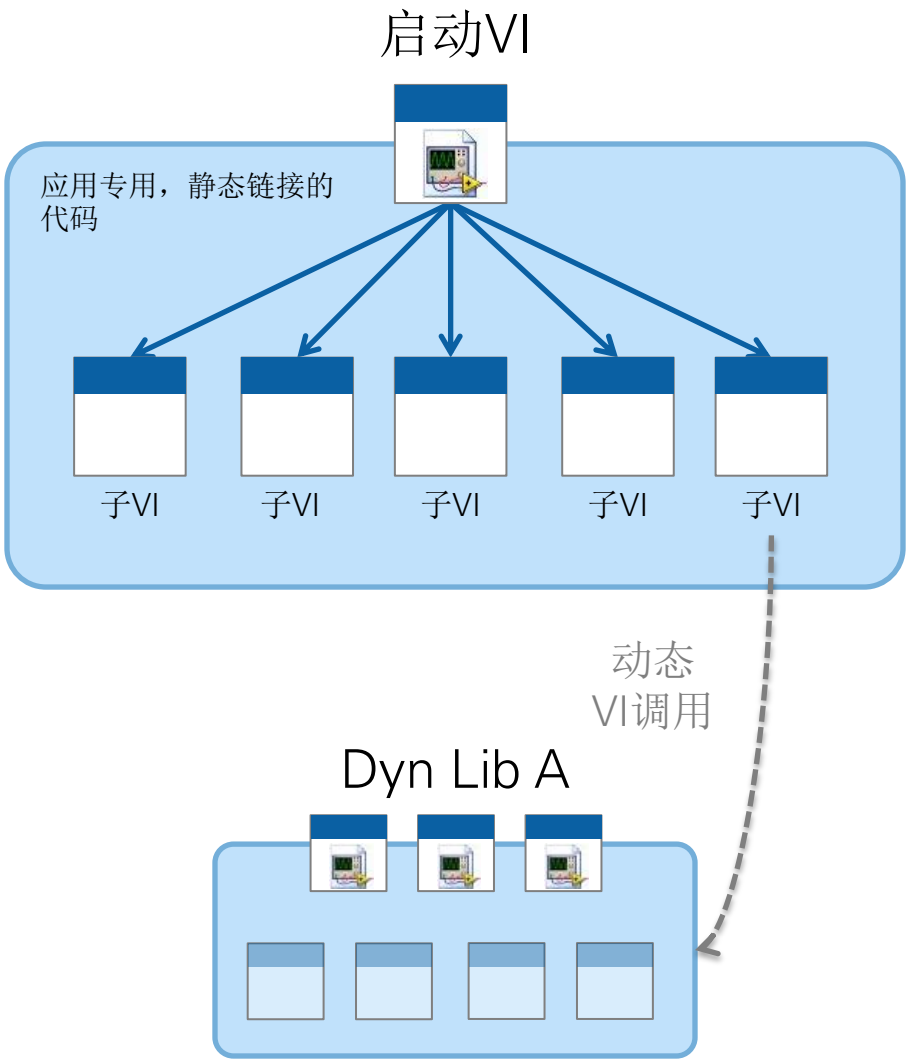
LabVIEW目录结构

vi.lib	包含内置VI库，这些VI在LabVIEW的函数选板上分组显示。
user.lib	保存用户创建的控件和VI的目录。 用户创建的控件和VI分别位于LabVIEW的用户控件选板和用户库选板。
instr.lib	包含所有已安装的仪器驱动。 这些驱动程序位于仪器I/O选板中。

磁盘和LabVIEW项目中的文件

演示

动态加载的文件



动态加载的VI并没有保存在内存中，除非调用VI加载了这些VI。

- 减少大型调用VI的加载时间
- 优化内存使用

不会在项目依赖关系目录中列出

跟踪动态加载的文件

动态加载的文件不会静态链接到项目的任何调用程序。 任何更改动态加载文件路径的操作都会阻止项目加载文件。

请确保动态加载的文件位于正确的位置：

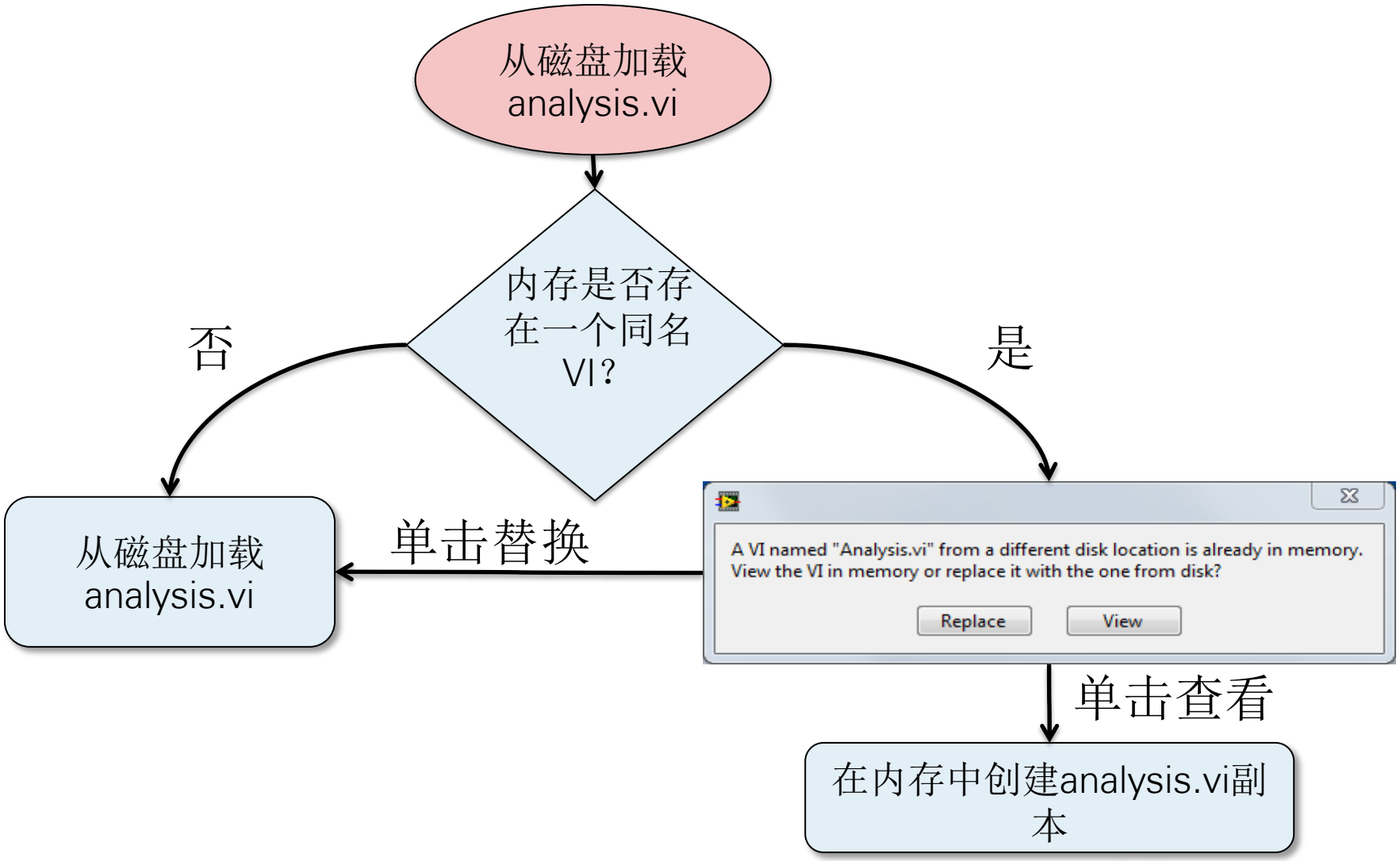
- 将文件保存在单独的文件夹内。
- 使用相对路径引用文件。
- 如要移动项目或发布一个应用程序，需将动态依赖关系所在的文件夹一并移动或发布。

LabVIEW搜索顺序

两个相同名称的VI不能加载到内存中
磁盘上可存在多个相同名称的VI。

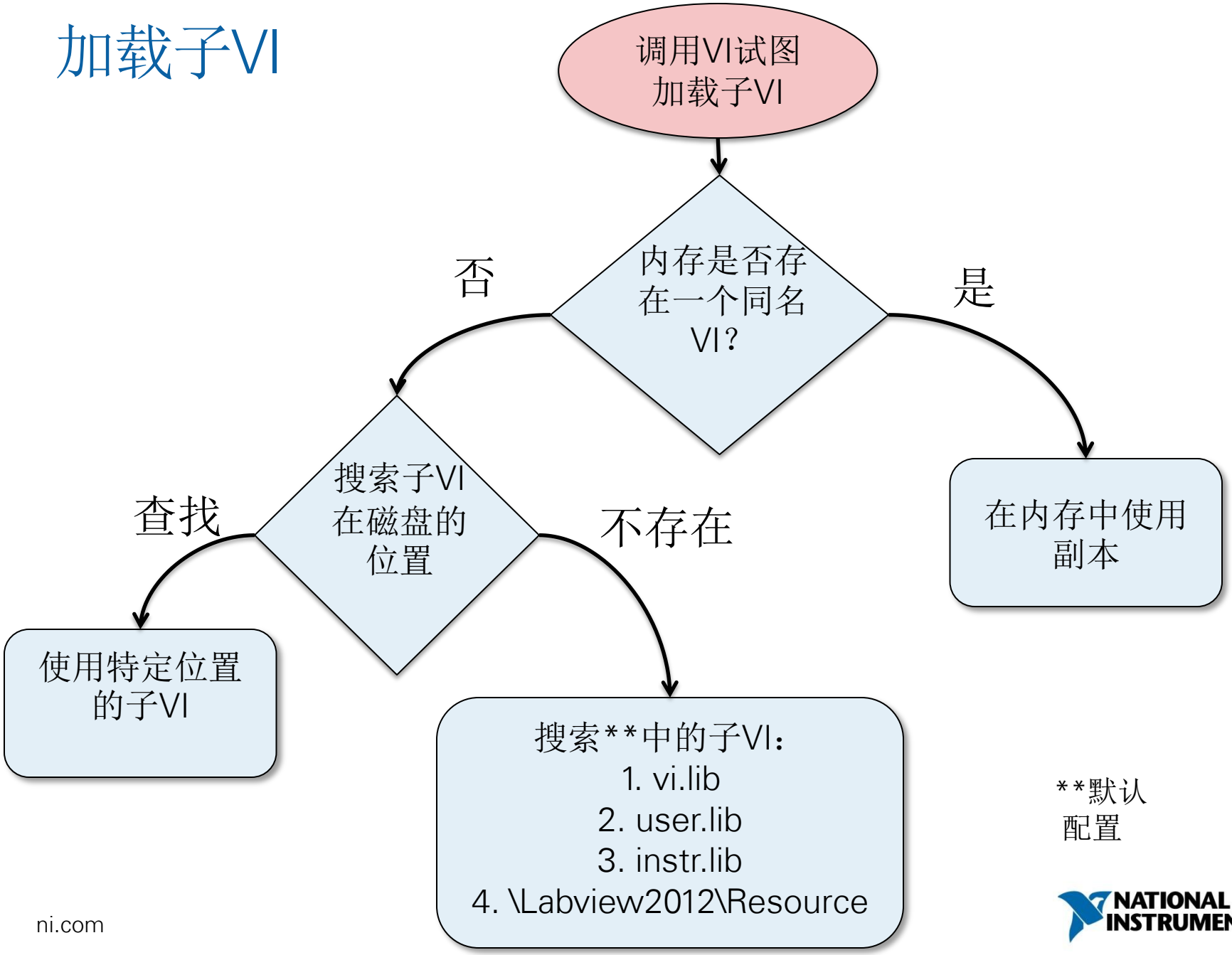
- 从磁盘
- 如果是某个调用VI加载子VI时该怎么办?

从磁盘加载VI



NATIONAL INSTRUMENTS

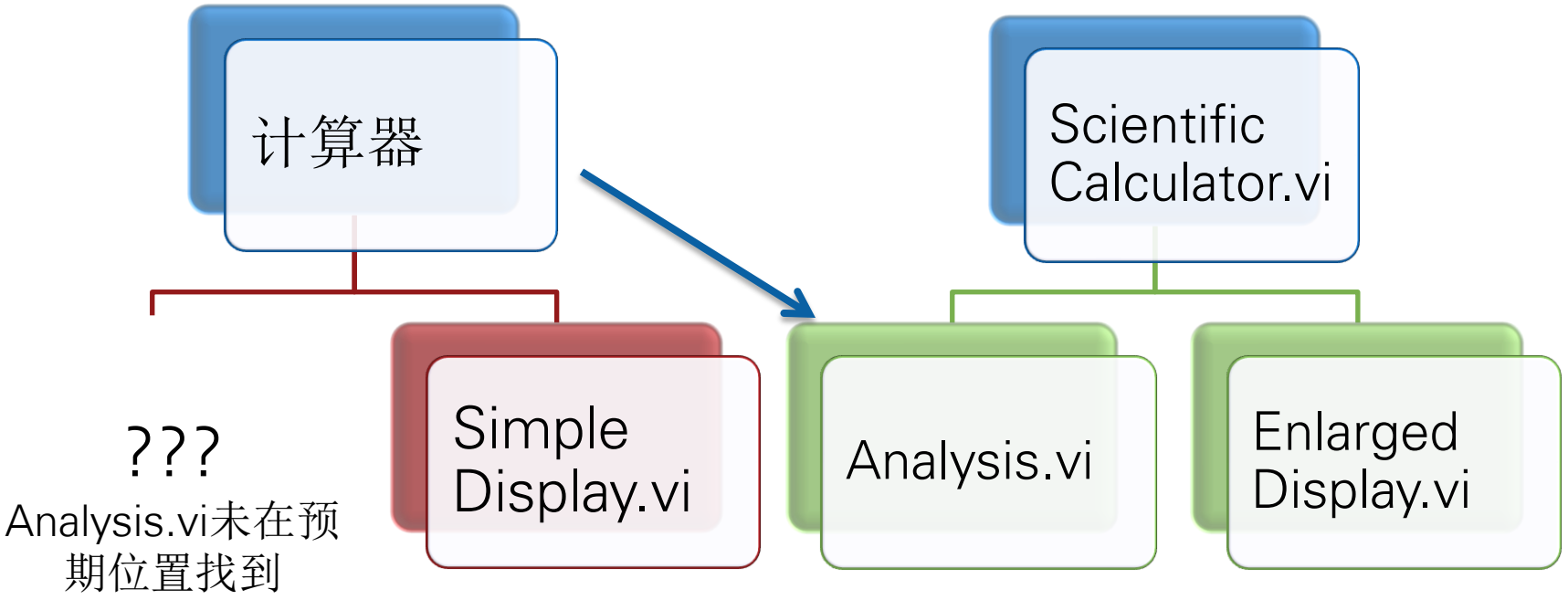
加载子VI



**默认
配置

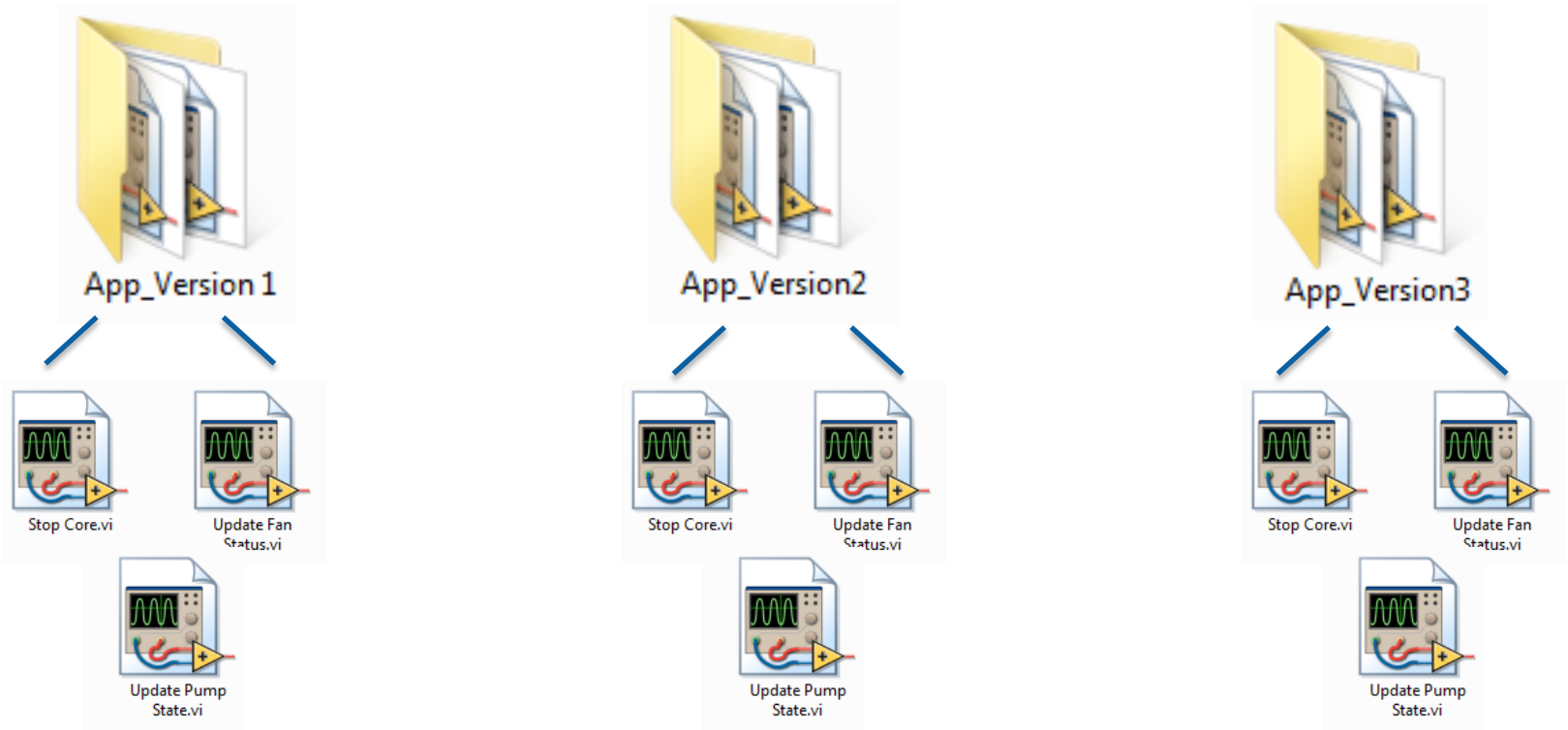
NATIONAL INSTRUMENTS

交叉链接定义



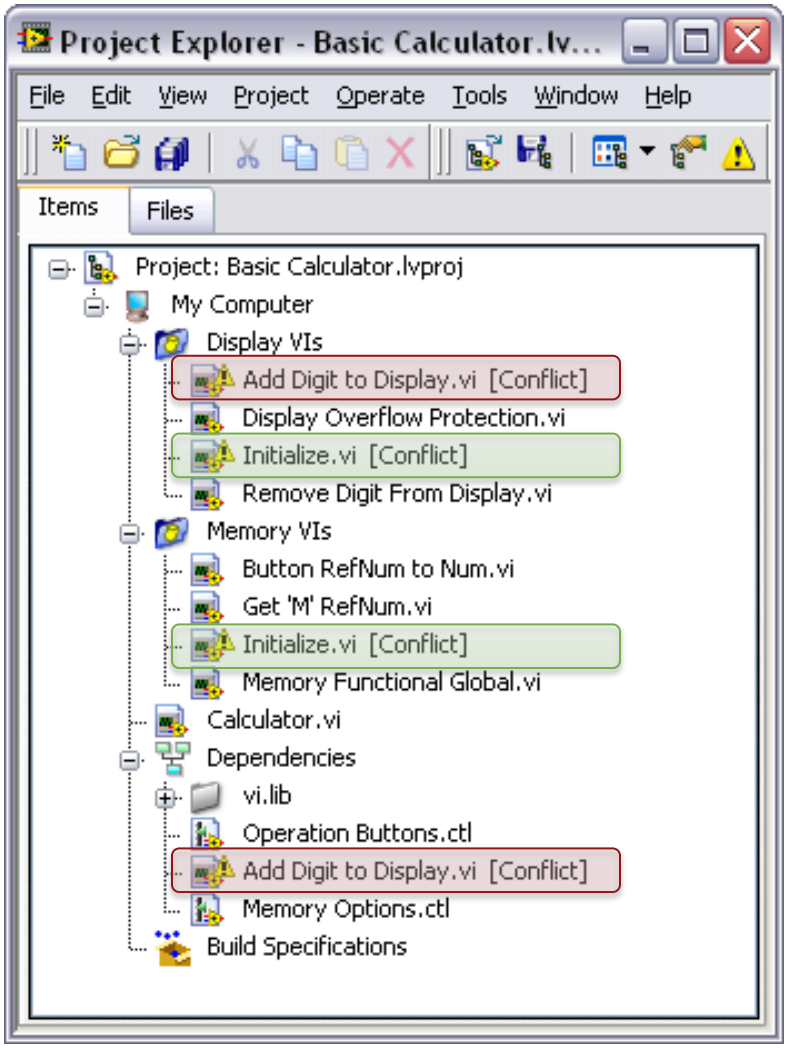
什么情况下会发生交叉链接？

如果您需要创建多个副本来备份工作目录，则特定VI将会在您的计算机上存在多个副本。



NATIONAL INSTRUMENTS

交叉链接通知



NATIONAL INSTRUMENTS

LabVIEW搜索顺序和交叉链接

演示

避免交叉链接

- ✓ 将所有文件添加至LabVIEW项目
- ✓ 考虑依赖关系
- ✓ 避免因创建多个备份导致代码重复
- ✓ 通过复用库在项目之间共享代码
- ✓ 确保VI名称的唯一性

总结

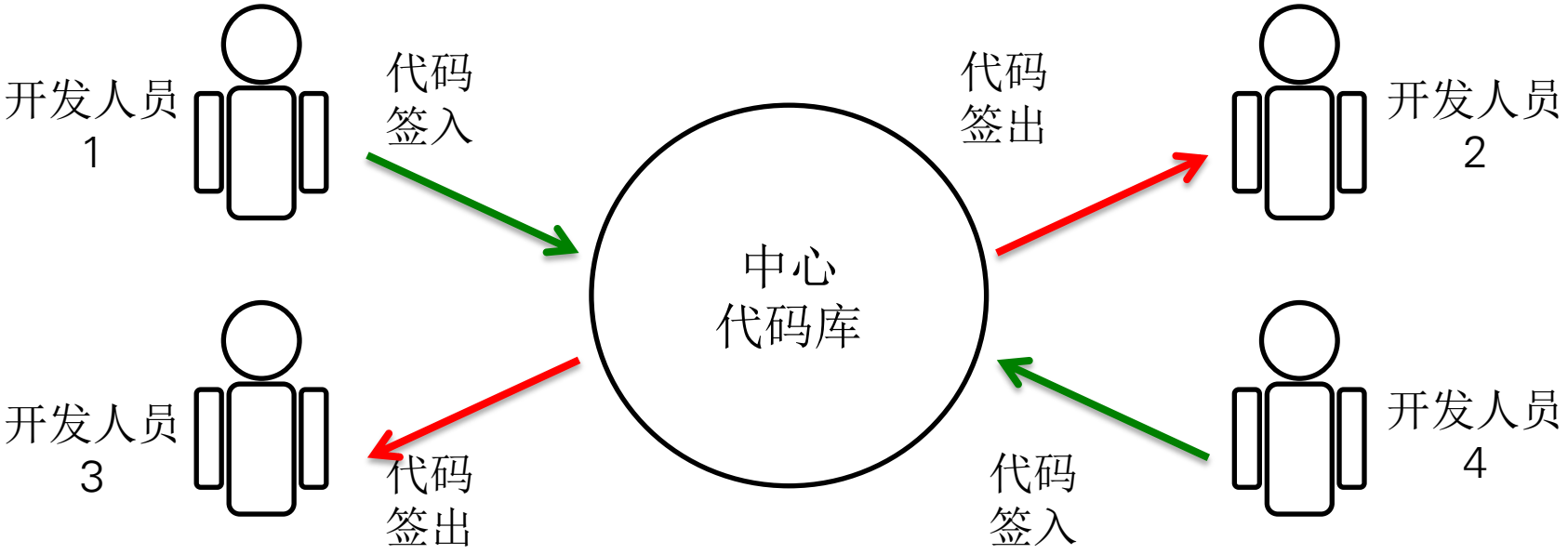
项目浏览器可用于组织应用程序和预防开发陷阱。

- 使用自动填充和虚拟文件夹来自定义文件的组织方式
- 通过文件视图可在移动文件时保留链接
- 动态分组已链接的资源
- 考虑依赖关系

使用源代码控制管理代码库

源代码控制

源代码控制用于在开发过程中跟踪、保存和管理与应用程序相关的所有文件。



NATIONAL INSTRUMENTS

为什么使用源代码控制？

通过允许多个开发人员共同在一个控制环境中进行开发来提高生产力

- 避免由于覆盖而导致的代码丢失

在整个开发周期中有效地管理文件

- 代码修订历史记录可帮助开发人员快速追踪漏洞和文件更改

使用合并和比较工具加速开发

源代码控制对于任何现代软件开发项目来说都是一个推荐操作，无论项目的复杂度或开发团队规模如何

哪些工具可用？

推荐配置

Perforce
Subversion

更多选择

Microsoft Visual Source Safe
Microsoft Team Foundation Server
Rational ClearCase
PCVS (Serena) Version Manager
MKS Source Integrity
Seapine Surround SCM
Borland StarTeam
Telelogic Synergy

哪些文件应该放置在源代码控制之下？

- ✓ VI
- ✓ 文档
  - 使用源代码控制跟踪修订并记录到需求文档中
- ✓ 配置文件
- ✓ 类型定义

如果是*.lvproj文件，会怎么样？

是否也需要将*.lvproj文件放置在SCC下？

LabVIEW *.lvproj文件是一个XML文件，包含：

- 项目包含的文件的链接
- 项目设置
- “虚拟项”，比如程序生成规范

所有开发人员必须采用*.lvproj文件的最新版本，以确保获得所有最新的依赖关系和资源

是否需要将*.lvproj文件放置到SCC下？

重命名或添加项目中的文件时，*.lvproj文件会随之变化并需要从源代码控制中签出，这样会影响使用该项目的所有开发人员。

```
<Item Name="Acquisition.lvlib" Type="Library" URL="../Acquisition/Acquisition.lvlib"/>
<Item Name="Logging.lvlib" Type="Library" URL="../Logging/Logging.lvlib"/>
<Item Name="Settings.lvlib" Type="Library" URL="../Settings/Settings.lvlib"/>
<Item Name="My Analysis.vi" Type="VI" URL="../Analysis/My Analysis.vi"/>
<Item Name="Main.vi" Type="VI" URL="../Main.vi"/>
```

.lvlib文件仅代表Project File by Library名称，只要库的名称保持不变，无需修改.lvproj文件也可修改库的内容。

每个VI都是由项目中的名称表示。这意味着只要项目添加或删除VI，.lvproj文件就会自动修改。

使用源代码控制管理项目文件的最佳实践

在开发初始阶段确定应用框架。

- 为之后要编写的所有代码段创建占位符，避免改变项目文件
- 使用.lvlib文件来避免修改项目文件

如果需要进行更改，可让某个开发员签出项目文件并进行编辑

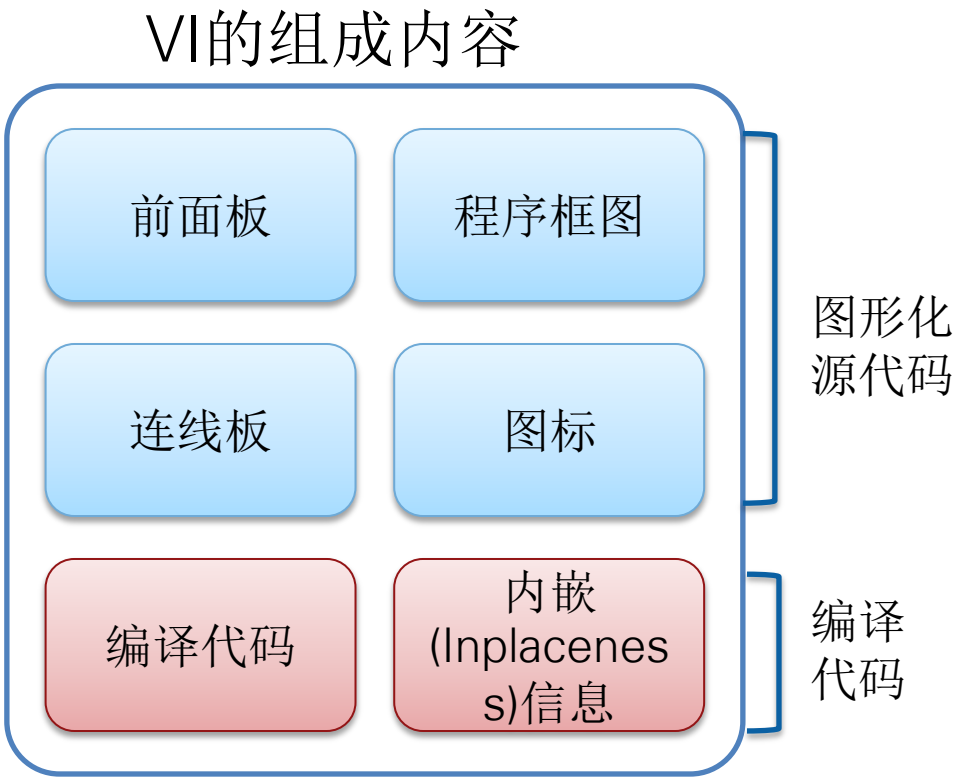
- 确保所有其他开发人员立即获得最新版本的项目文件

使用LabVIEW配置SVN源代码控制

演示

将VI保存于SCC时的考量因素

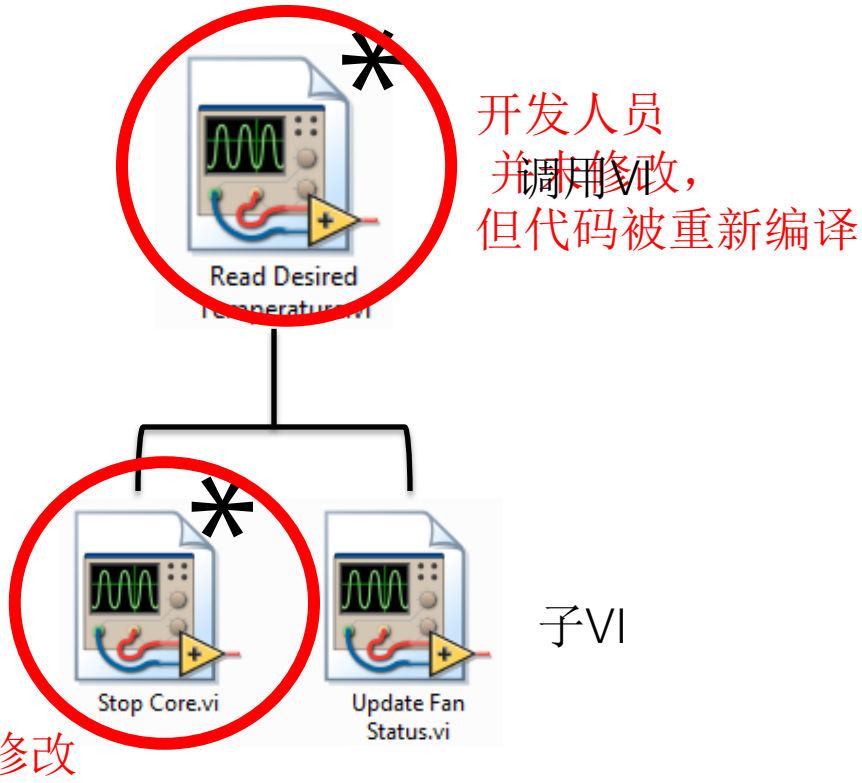
编辑VI时，LabVIEW会重新编译VI代码。 **LabVIEW也可能会重新编译该VI的调用程序来优化代码。**



将VI保存于SCC时的考量因素

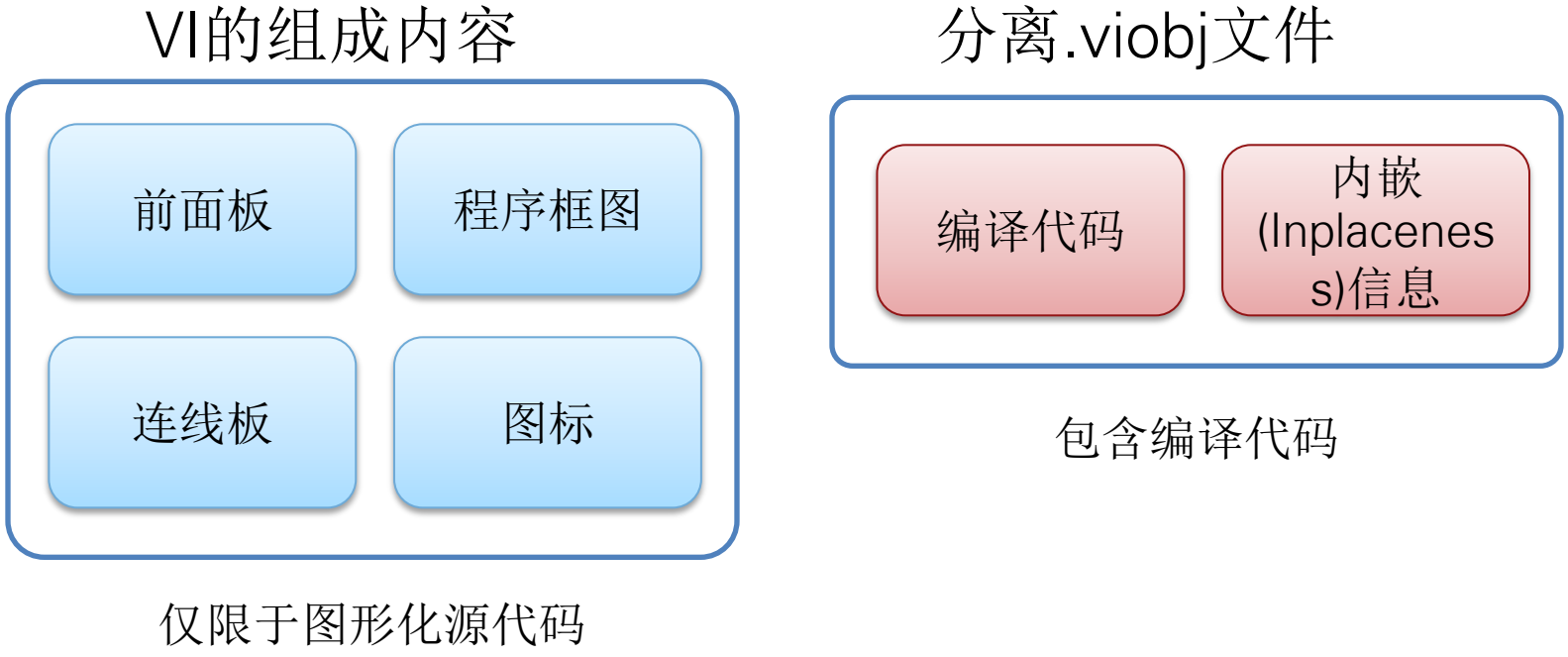
编辑VI时，LabVIEW会重新编译VI代码。 **LabVIEW也可能会重新编译该VI的调用程序来优化代码。**

调用编程人员**未**修改过的VI可能意味着这些VI之前进行过修改，因此需要重新提交至SCC。



将VI保存于SCC时的考量因素

无需重新保存和重新提交文件至源代码控制，除非开发人员对图形化源代码进行修改



确定合适将编译代码从VI中独立出来

分开编译代码可：

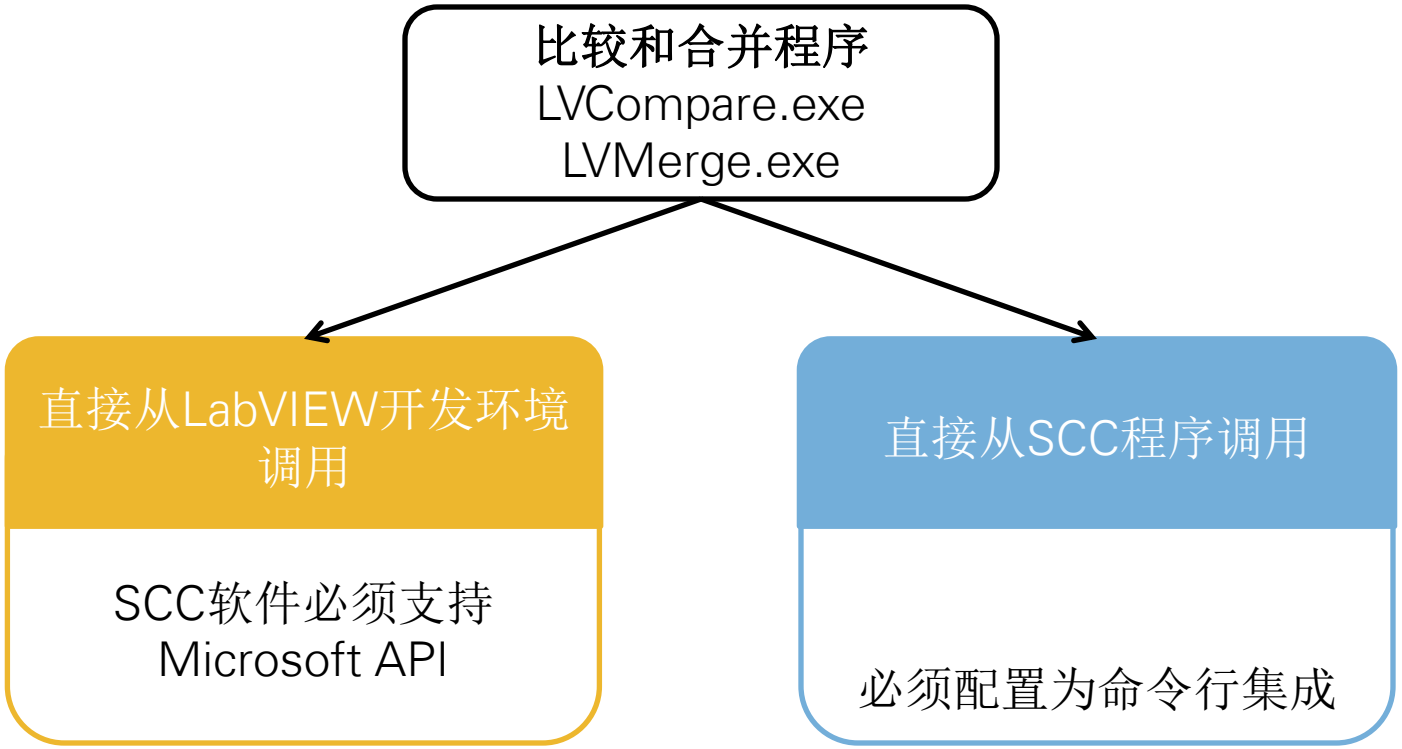
- 简化源代码控制
- 便于SCC的VI升级至新版本的LabVIEW
- 加快VI的加载时间

无需分开编译代码的情况：

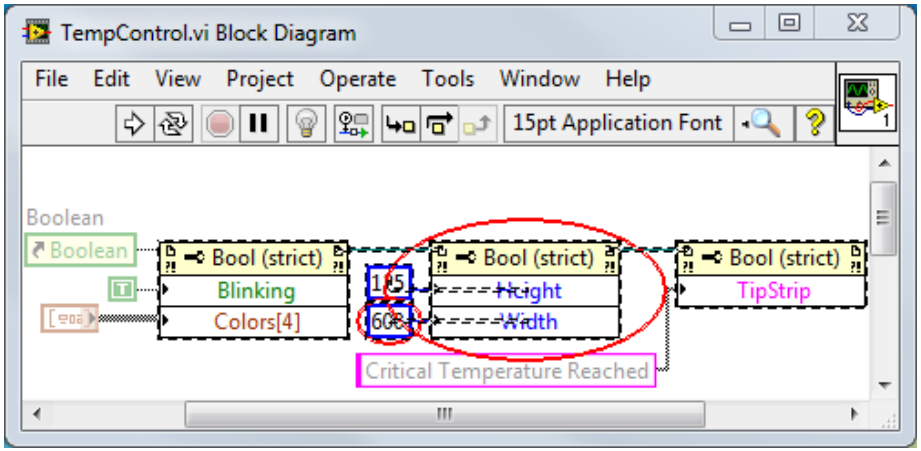
- 使用LabVIEW运行引擎运行VI 此时，可考虑构建一个源代码分布包

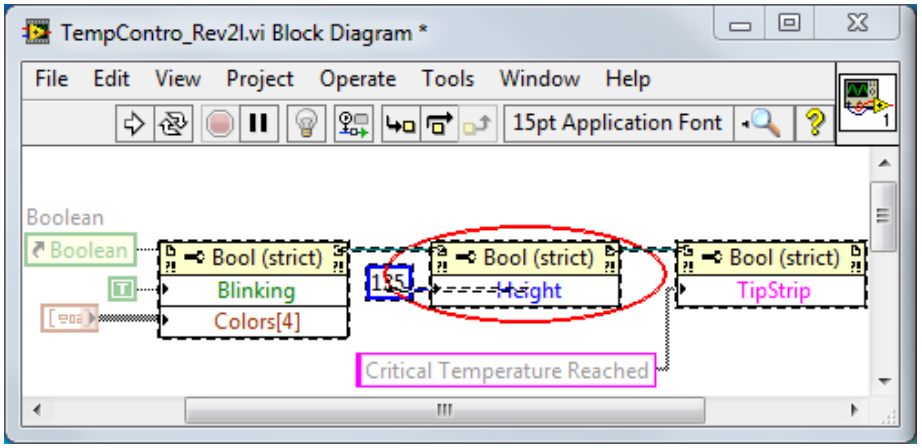
图形化差异和合并程序

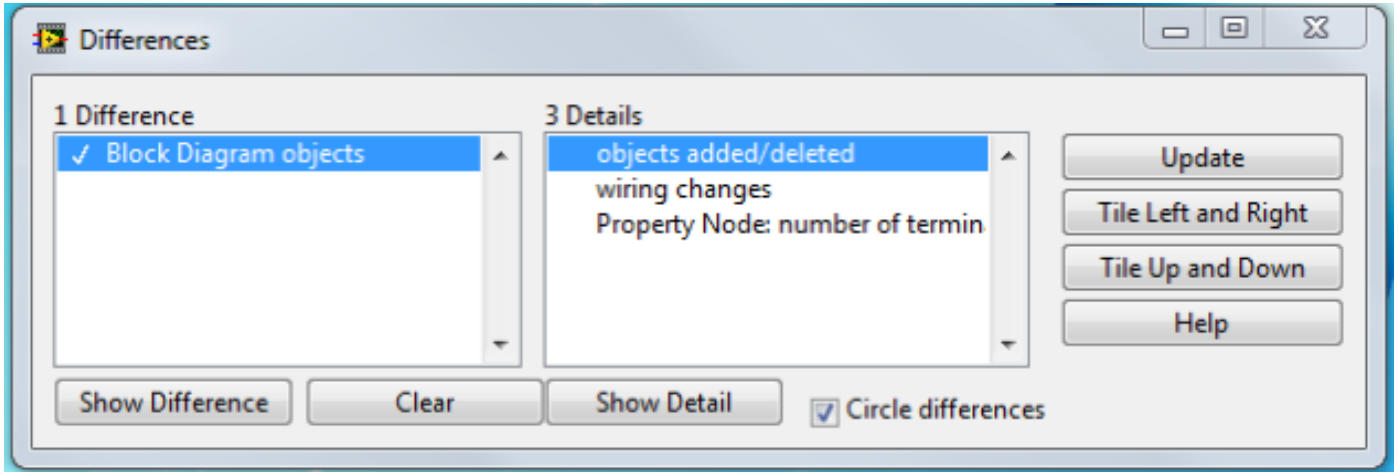
由于VI的源代码是二进制的，必须使用特定比较和合并程序。



图形化比较







NATIONAL INSTRUMENTS

图形合并

简化开发周期。

当多个人员对某个VI进行修改并分别保存时可使用这一工具

结合SVN使用图形化比较工具

演示

团队开发建议

- **使用源代码控制**
- 每次提交时更新文档
- 使用VI比较工具查看改动
- 使用VI合并工具整合代码修改

构建和发布可复用程序

挑战

如何在多个开发周期不同的独立项目之间利用通用代码？

- 如果我想要复用的代码仍在开发中该怎么办？ 如果管理多个版本的复用代码？
- 复制通用代码可能非常繁琐，而且会导致交叉链接

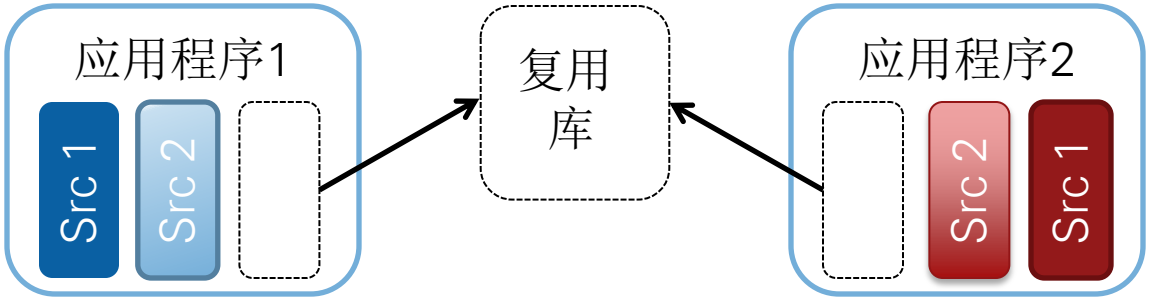
解决方案: 创建一个复用库

管理复用库

复用库专为跨项目开发而设计。 复用库的开发周期可能与所服务的项目不一致。

复用库应该：

- 与所服务的项目独立开来
- 在维护多个版本的应用程序时可轻松升级或降级



源代码发布包

打包并发送给用户的文件集合。

- 包含VI文件，可允许多个开发人员将多个VI作为单个文件移动

创建包含VI的目录或压缩包

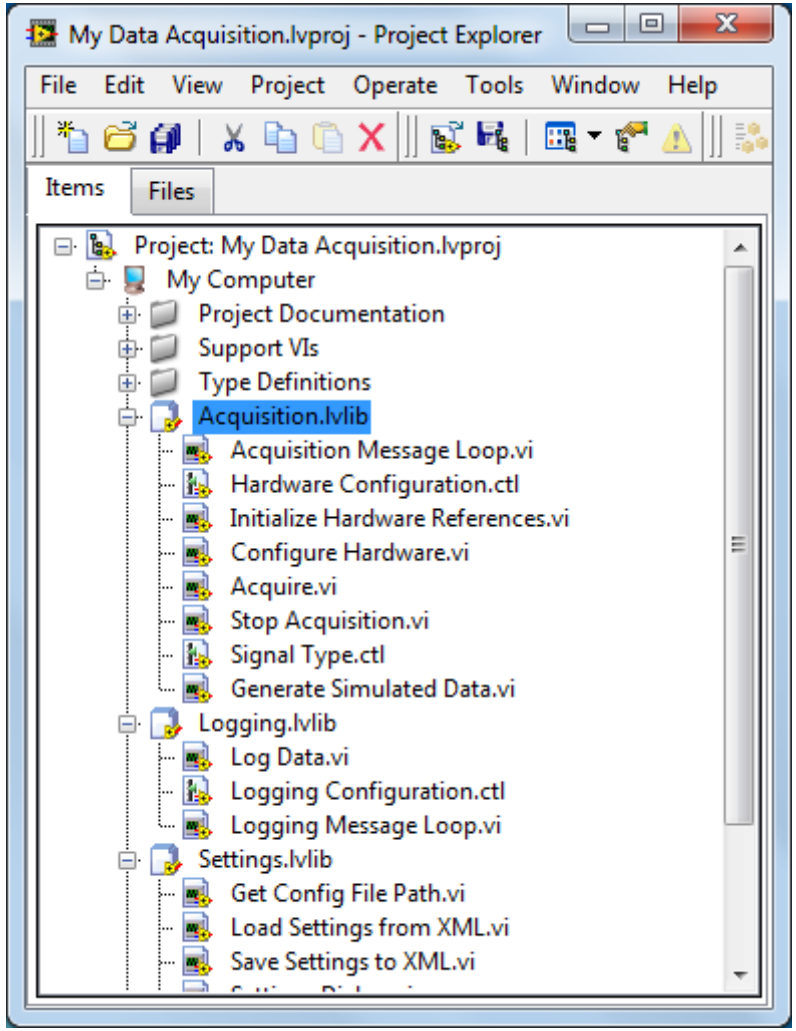
配置可包含的VI，可选择不包含vi.lib、user.lib和instr.lib

项目库

VI、类型定义、共享变量、选板文件等的集合。

.lvlib文件是一个xml文件，包含项目库自带的文件引用和库属性

.lvlib文件不包含实际的文件



NATIONAL INSTRUMENTS

何时使用项目库？

使用项目库可：

- 封装大量程序片段
- 组织项的虚拟层次结构
- 限定VI的名称，避免交叉链接
- 修改内容时无需改动项目(*.lvproj)文件
- 限制特定文件类型的访问（通过将项目库配置为公共或私有）

发布API时的建议

打包项目库文件 *.lvlibp

打包项目库文件是一个预编译的.lvlib文件，可允许用户访问库中的公共VI，但无法修改代码。

为什么要使用*.lvlibp文件？

- 减少独立应用程序的生成时间
- 通过将多个VI打包到一个.lvlibp文件中可减少部署文件的数量
- 发布不可修改的公共VI的API

使用项目库

演示

课程总结

组织和管理LabVIEW应用程序

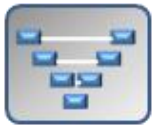
有效的团队开发实践

管理复用库

软件工程的最佳实践

ni.com/largeapps


软件工程工具


开发实践


大型APP社区

NATIONAL INSTRUMENTS