1. 活动注册：

RegisterActor->GetActorInfoList->CreateActorTree(创建整个活动树)

{

Boost::Tokenizer库:包含四个分词对象(char\_separator/char\_delimiters\_separator/escaped\_list\_separator/offset\_separator)，其中char\_delimiters\_separator已被废弃。

Char\_separator有两种构造函数:char\_separator()和char\_separator(const Char\* dropped\_delims, const Char\* kept\_delims = 0, empty\_token\_policy empty\_tokens = drop\_empty\_tokens),参数解释如下：dropped\_delims表示在分词后不被保留的分隔符，kept\_delims则被保留的分隔符，drop\_empty\_tokens不保留空格，keep\_emptry\_tokens保留空格。

}

1. 由活动注册->整体界面设计:

通过wxRibbonBar添加wxRibbonPanel的方式设计主菜单栏，按钮为wxRibbonButton，

主界面由wxNotebook控件将其分为两部分，一部分为左侧的视图栏，另一部分为右侧的绘图操作栏。

1. 宏wxDynamicCast(ptr,className),将对应的类转化为指针形式返回，如果不存在返回NULL。
2. WorkFlowSystem类中定义了得到所有活动的接口。
3. wxTreeCtrl类：构造一棵树型结构，用来存储文件信息。方法getFirstChild()得到除根意外的第一个子节点，getNextSibling()得到兄弟节点。
4. PortElement.h:有关端口的属性和相关操作。
5. DesignViewMode.h:包括活动的添加，活动之间的连线等鼠标事件的操作。
6. ChoseTriggerPortDlg.h:主要是连线中对端口功能的选择对话框。
7. LinkElement.h:连接两个活动。
8. 在整个界面中，分为三层设计:mode->view->window，在mode中可通过获得指针方法得到最终的显示窗口。
9. wxDC系列：

wxDC为所有绘图设备的父类，其子类包括wxBufferedDC,wxMemoryDC,wxScreenDC,wxClientDC,wxPaintDC,wxWindowDC等等；

wxClientDC：对应窗口的client区域.一般情况下,我们都应该使用这个来修改我们的窗体显示

wxPaintDC：wxPaintDC仅限于在wxPaintEvent事件中使用(也就是OnPanit)

wxWindowDC：并且这个只在windows平台可用,象linux中的Xwindow,边框这些东西都是窗口管理器负责,自然是没有这个wxWindowDC的概念了.如果你想全面修饰窗口就需要用到这个.

wxScreenDC：这个就是整个屏幕了,如果你想捕捉屏幕,就可以用到它.还有一个小技巧,你可以通过wxScreenDC获得当前系统的分辨率

wxMemoryDC：通过名字就能了解,是内存DC,实际上对应一个bitmap.可以通过SelectObject选择操作的bitmap.所有针对该DC的操作,实际都是画到bitmap上的

.

wxBufferedDC：我们在自己用DC画东西的时候,都会碰到背景闪烁的问题,一般经典的解决方法就是双缓冲技术.就是说我们构建一个中间的DC,先把我们需要的图象都画到这个DC上,最后在整个更新到目标DC上,这样就避免了闪烁的问题.wxBufferedDC,就是完成这样的任务的，它的构造函数如下：

wxBufferedDC();

wxBufferedDC(wxDC \*dc, const wxSize& area, int style = wxBUFFER\_CLIENT\_AREA)

wxBufferedDC(wxDC \*dc, const wxBitmap& buffer, int style = wxBUFFER\_CLIENT\_AREA)

如果使用默认的构造函数，需要调用wxBufferedDC::init()，原型如下：

void Init(wxDC \*dc, const wxSize& area, int style = wxBUFFER\_CLIENT\_AREA);  
void Init(wxDC \*dc, const wxBitmap& buffer, int style = wxBUFFER\_CLIENT\_AREA);

wxBufferedPaintDC:是wxBufferedDC的wxPaintEvent版本, 由于wxPaintDC只针对client区域,所以能看到它的构造函数和wxBufferedDC有些差别,不需要指定wxDC,因为就是针对wxPaintDC的.构造函数如下:  
wxBufferedPaintDC(wxWindow \*window, const wxBitmap& buffer, int style = wxBUFFER\_CLIENT\_AREA)  
wxBufferedPaintDC(wxWindow \*window, int style = wxBUFFER\_CLIENT\_AREA);

1. 在canvas画布中，实现了OnPaint(wxPaintEvent &)的事件处理，将其中的wxPaintDC的传递过程如下：canvas::OnPaint->workflowView::OnPaint->ViewMode::OnPaint(纯虚函数)->designModeView::OnPaint(具体实现).
2. 双击活动界面，弹出属性设置对话框：

DoubleClick事件->确定是否为有效碰撞控件区域->设置碰撞控件->execute(获得属性对话框模式(monitor mode 还是 Design mode)->总体界面设计->各部分区域参数设置(解析XML文件中各活动的参数名称以及参数获得所要设置的参数))

设置属性：

designViewMode(doubleClick)->showPropertyCommand(execute)->ActorElement(showProperty)->ActorPropertyDlg(ActorPropertyDlg)->ActorPropertyDlg { UserParameterPropertyDlg，SystemParameterPropertyDlg，ShowPropertyDlg}最终完成属性的设置

1. 修改：在某些活动中，选择文件时，删除了该文件名，结果再次打开时还是显示该文件名，/\*bug====🡺>>>>>定位在ActorPropertyDlg的1316行\*/

原因：在删除路径的时候，只删除了text中的数据，而没有更新在XML中，导致下次打开时依然读取了上次打开的文件名。

1. 从创建模板界面，使用快捷启动监控程序按钮启动监控程序，其方式是：通过文件系统类获得监控程序所在的路径，再启动命令行自动运行监控程序。同时在是否连接的状态下自动启动到断开和连接的状态。
2. 在监控软件中，ProjectMonitorFrame是整体界面的设计类。由ProjectMonitorApp类在构造函数中进行构造。
3. Wxwidgets下的命令行解析：

wxCmdLineEntryDesc作为命令行选项的描述，结构如下：

struct wxCmdLineEntryDesc

{

wxCmdLineEntryType kind; //wxCMD\_LINE\_OPTION…

const char \*shortName;

const char \*longName;

const char \*description;

wxCmdLineParamType type; //wxCMD\_LINE\_VAL\_STRING…

int flags;

};

wxCmdLineParser作为解析类，构造函数如下：

wxCmdLineParser()；

wxCmdLineParser(int argc, char \*\*argv)；

wxCmdLineParser(int argc, wxChar \*\*argv)；

wxCmdLineParser(int argc, const wxCmdLineArgsArray& argv)；

wxCmdLineParser(const wxString& cmdline)；

wxCmdLineParser(const wxCmdLineEntryDesc \*desc)；

wxCmdLineParser(const wxCmdLineEntryDesc \*desc, int argc, char \*\*argv)；

wxCmdLineParser(const wxCmdLineEntryDesc \*desc, int argc, wxChar \*\*argv)；

wxCmdLineParser(const wxCmdLineEntryDesc \*desc,

int argc,

const wxCmdLineArgsArray& argv)；

wxCmdLineParser(const wxCmdLineEntryDesc \*desc, const wxString& cmdline)；

1. boost::thread: