跨平台人机交互软件组件集成框架①

陈宏君, 刘克金, 张 磊, 王国栋

(南京南瑞继保电气有限公司, 南京 211102)

摘 要: 提出一种跨平台的人机交互软件组件集成框架, 包括三部分内容: 一是人机交互软件的框架结构, 包括引导程序、主框架、组件实体、消息总线. 二是组件的管理方式, 组件是跨平台的可单独加载的单元, 引导程序将组件二进制代码加载到内存. 框架通过虚函数体系动态地派生出不同组件实体对象, 并以组件名为关键字散列挂载到消息总线上, 组件对应的界面放置在主框架的界面容器内. 三是基于领导者-追随者模型的组件状态切换和交互协同工作方式, 组件通过鼠标或键盘操作响应获得焦点, 进入领导活动状态, 处理完事件后进入休眠追随状态. 组件之间通过消息交互协同, 并支持快速消息总线和事件循环总线. 该框架在保护测控装置配置软件、工业控制编程软件进行了应用, 实践表明基于该架构显著降低了组件间耦合关系, 提高了开发效率和组件复用度.

关键词: 跨平台组件; 集成框架; 消息总线; 状态切换; 组件协同

Cross-Platform Integration Framework for HMI Software

CHEN Hong-Jun, LIU Ke-Jin, ZHANG Lei, WANG Guo-Dong

(NR Electric Co. Ltd., Nanjing 211102, China)

Abstract: The paper promotes a cross-platform components integration framework for human-machine interaction (HMI) software and illustrates it from three aspects. First of all, the paper introduces its frame construction which includes bootstrap program, mainframe, component entity and message bus. Then it presents the management approach of all components: each component entity is a cross-platform unit loaded by the mainframe separately with the bootstrap program. The framework derives all component entities dynamically through virtual function system, then mounts them to the message bus with their name as hash keyword and inserts their interfaces into the mainframe interface container. Thirdly the paper elaborates the approach for component status switch and interactive collaboration which is based on the leader-imitator pattern and supports fast message bus and event loop bus. In the approach a component gets focus by mouse or keyboard event, it is selected as active leader and enters dormant imitator status again after processing the event. And all components interact with each other through messages. The framework was already put into application in configuration software for protection and monitoring control device and programming software for industrial control, and was demonstrated observably efficient in reducing coupling relationship of components while boosting the development efficiency and component reusability.

Key words: cross-platform component; integration framework; message bus; status switch; component cooperation

在人机交互平台软件的生命周期内, 由于面向的领域和用户非常复杂, 软件需求变动、运行环境改变等均要求系统具有较强的适应能力. 需要一种通用的方法来实现满足特定的需求, 确保个性化的需求实现不影响已经成熟稳定的组件. 其中基于组件模式的软件开发和集成, 已经有研究和应用[1-5]. 采用组件的好处: 接口和实现分离(将组件的通信协议和组件的内部实现细节分开)、去耦合(尽可能地消除软件之间或者

① 收稿时间:2015-03-12;收到修改稿时间:2015-04-26

软件的不同部分之间的联系)、可部分升级. 文献[1]提出了一种基于组件的程序产物形成架构和应用, 但只涉及了纯文件数据处理组件的调用替换过程. 对于纯数据处理的组件开发实现, 由于是一趟式线性函数调用, 其难度可控. 文献[2]提出了基于“防火墙”方法系统层面的组件修改影响分析模型, 文献[3]以“感知-决策-执行”的软件自适应周期为主线, 分析了组件的感知-决策各个环节特征. 文献[4-6]主要是组件开发方法和接口描述的抽象阐述, 通用组件技术发展已经相对成熟. 文献[7]分析了通用组件模型不能在嵌入式系统中应用原因, 对嵌入式组件模型进行分类和层次化分析.

对于人机交互频繁、基于界面操作非线性触发、存在长久活动生命周期的人机交互工具软件组件化开发, 存在组件生命期管理、组件间通信和协同、界面组织管理等难点, 同时基于 Microsoft 的 COM 组件存在机制复杂、不能跨平台复用的问题, 需要设计一种轻量级、跨平台的组件集成协同工作方式和界面分层管理架构. 本文介绍了一种可跨平台的通用的人机交互软件集成框架和基于组件化的编码集成协同交互方法, 在不修改主框架程序和不重新编译的情况下, 面向不同的应用场景, 可配置加载不同组件, 基于 C++/Qt4.8 跨平台库形成对应的界面, 响应相关人机鼠标键盘操作. 本方法可降低组件的更新替换对软件系统的稳定性影响, 提高人机交互软件的集成和适应能力. 本文详细阐述了基于组件的跨平台的人机交互软件通用集成框架和对应组件的协同交互方法, 并介绍了其应用实例.

1 组件集成框架

1.1 组件系统原理

组件(package/component)是可独立发布的二进制单元. 组件是 1 个黑盒子, 对外的接口主要有 API 和端口, 其中 API 是组件对外提供的功能, 端口表示组件内部调用外部其它组件功能的函数[2]. 图 1 是组件系统原理图.

一个使用组件结构的软件, 由一个可执行宿主程序和多个完成子功能的组件组成, 宿主程序负责启动整个系统, 将所需的组件加载到自己的进程地址空间中.

1.2 人机交互软件组件集成框架

本文提出的人机交互软件集成框架包括: 引导程序、主框架、组件库、消息总线, 如图 2 所示.

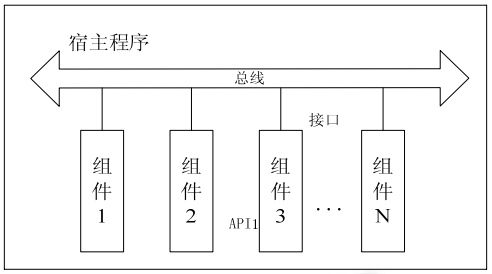

图 1 组件系统原理图

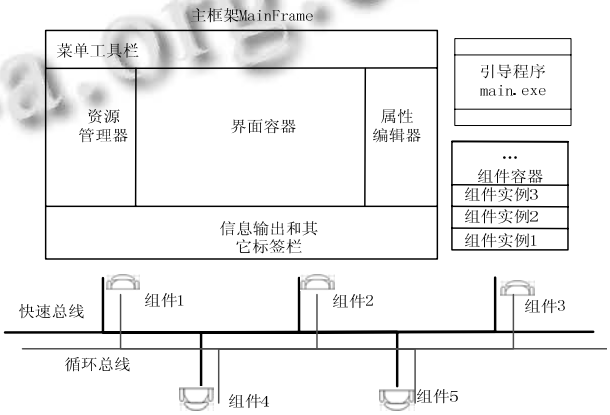

图 2 人机交互软件集成框架

引导程序是 main.cpp 文件编译后形成的可执行程序, 仅包括一个 main 入口函数. 在 main 函数中创建主框架(MainFrame)实例. 组件库包括各个顶层组件编译后的二进制目标文件, 在 main 函数中调用其构造函数, 形成组件对象实例, 将组件对应的主窗体界面放入到主框架容器内. main 函数还调用消息总线构造函数, 将组件接口对象指针放入消息总线散列管理.

主框架由菜单工具栏、界面容器、资源管理器停靠栏、属性编辑器停靠栏、信息输出停靠栏 5 个区域构成. 在顶层主框架的界面容器管理各顶层组件的主窗体界面, 每个组件在主窗体容器内管理各自子界面. 界面容器支持后入先出栈、并行标签、多窗体工作空间 3 种排列显示方式. 图 3 是两层界面容器管理示例.

消息总线是个全局对象类, 根据组件名和编号形成组件的全局 ID, 散列到总线节点上, 消息总线提供 2 种消息发送方式: 各组件可调用总线的快速发送接口, 向目标组件发送消息, 并阻塞等待目标组件处理完后返回状态和数据, 用于需要立即返回的高优先级事件处理. 各组件也可将消息发往事件处理循环队列

中, 总线周期定时触发后, 将消息转发给目标组件, 用于普通优先级的事件处理. 相关消息由哪条总线发送由组件开发人员决定, 参考标准是: 对于在 500ms 内需处理完毕的消息, 放在快速总线上发送, 例如解析网络报文后通知界面刷新数据的消息. 对于无需界面显示响应的后台处理消息, 放在普通优先级总线发送, 例如形成装置配置文件的消息. Debug 版本提供日志功能, 可记录相关事件从发送到响应结束后的时间, 便于组件开发人员衡量选择. 本文的框架适用于单机系统的 HMI 软件, 经过测试, 在 2G 内存的计算机上, 软件管理并编辑 150 台装置的配置数据时, 快速总线能满足相关处理需求.

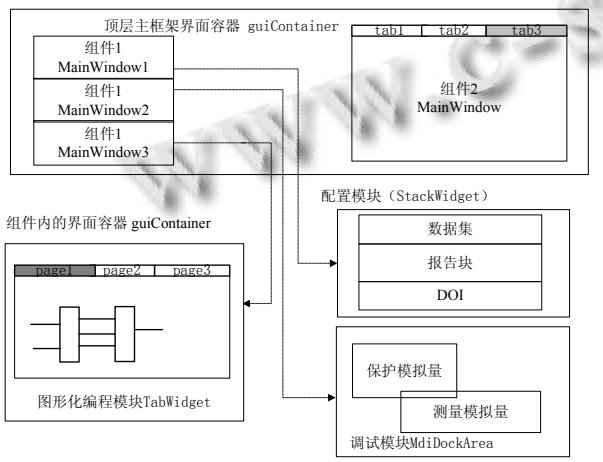

图 3　分层界面容器管理示例

1.3 组件间协同交互

组件的状态切换采用领导者-追随者模型: 在组件配置文件中, 预设一个领导者角色的组件. 引导程序加载完组件后, 主框架文件菜单中的新建文件、打开文件、保存文件等常规操作, 交由领导者组件响应处理, 此时其它组件处于追随休眠状态. 领导者组件打开工程管理文件后, 在资源管理器显示绘制层次节点树. 当鼠标或键盘点击相关节点时, 领导者组件将相关节点编码成消息, 以广播方式通知消息总线上的其它组件. 各组件根据节点类型判断是否属于处理范畴, 当属于此组件处理时, 该组件立即进入领导者角色, 显示相关组件界面, 上一个活跃者则保存对应的视图数据, 进入追随休眠状态.

以文件保存为例, 鼠标点击工具栏的保存按钮, 工程管理组件响应该命令, 向所有组件群发保存消息:

```
CMSG* pmsg =
new CMSG(getName(),"MSG_SaveGui");
CMsgBus::instance()->quickSend("all", pmsg);
delete pmsg;
```

消息总线串行发送各消息, 各组件接收到该命令后, 依次进入领导者状态, 保存对应的数据.

1.4 具体实现

人机交互软件组件集成框架, 按照分层划分实现. 结合图 4 所示, 具体实施步骤包括如下四个部分:

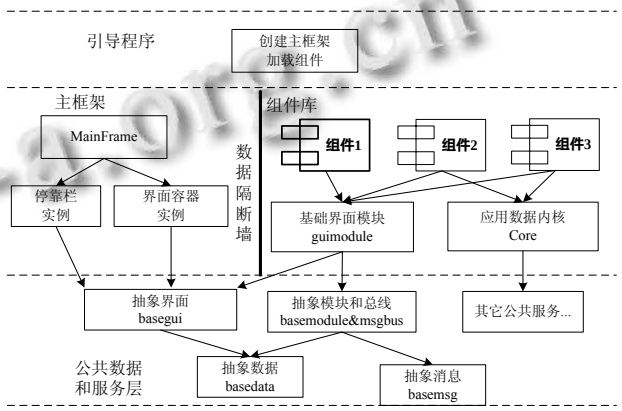

图 4　基于组件的人机交互软件分层结构图

第一部分, 编写公共数据服务层, 包括抽象数据、抽象界面、抽象消息、抽象组件和消息总线等, 可被其它组件依赖调用. 其中: a)抽象数据用于管理层次化的 XML 节点数据, 存储中间产物数据, 是个通用递归的数据模型, 抽象数据包括若干属性和若干子节点, 其类结构如图 5 所示. b)抽象界面包括停靠栏、资源管理器树形层次展示节点、顶层组件的主窗体基类、界面容器共 4 个类. c)消息是组件交互的报文, 事件是消息总线的传输媒介. 消息的成员变量包括: 发送者、接收者、消息类型、消息编码、关联数据地址值.

第二部分, 编写主框架. 主框架是人机交互软件的最顶层的主界面, 是一个通用的抽象的软件集成框架. 包括菜单、工具栏、停靠栏、存放顶层组件的界面容器, 其类结构如图 5 所示. 支持各顶层组件创建主框架的子菜单、图标、停靠栏内的标签窗体.

主框架提供文件的新建、打开、关闭、保存操作入口, 可显示、隐藏停靠栏. 以文件新建为例, 主框架本身不实现具体功能, 只是交由领导者负责具体处理:

```
void CMainFrame::slotNewFile()
{
  if( m_pLeader ){
```

```
    CMSG* pmsg = new CMSG("main", "NewFile");
    m_pLeader->respondServer(pmsg);   }
}
```

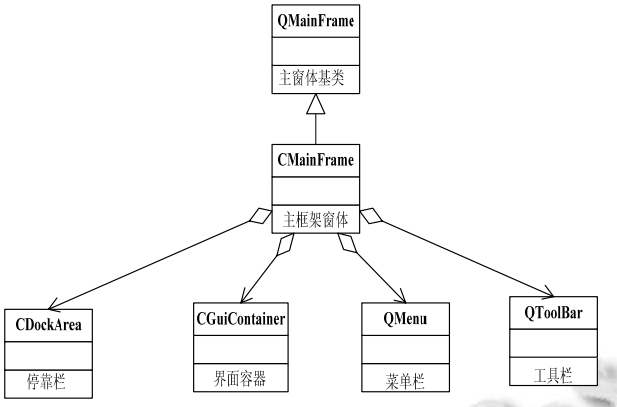

图 5 主框架 UML 模型

当应用场景切换时, 即使更换了核心组件, 只要保留对外的接口不变或者组件名不变, 主框架无需修改和重新编译, 达到通用的软件集成框架的效果.

第三部分, 编写引导程序. 在 Main 函数中, 创建主框架实体和组件容器, 读取顶层组件配置文件, 加载导入组件的二进制目标文件, 调用预定义的组件创建接口, 形成组件的数据实例. 当主框架关闭退出后, 卸载各顶层组件. 在顶层组件配置文件中可定义组件的 dll 文件名、组件对象名、领导者属性和组件树的当前活动分支等内容. 当软件存在研发版、用户版等不同版本时, 只需在组件配置树中配置各自版本所需的组件 dll 文件名. 在引导程序中创建消息总线, 消息总线基于全局 singleton(单实例) 对象创建模式, 其类结构如图 6 所示.

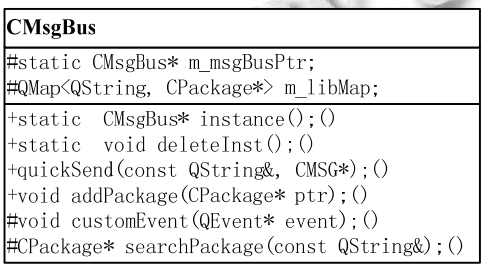

图 6 消息总线的 UML 模型

完成上述三个步骤后, 一个通用的人机交互软件集成和消息交互框架已经具备.

第四部分, 面向各种应用场景编写各具体的组件库, 完成各具体人机交互软件的集成发布. 组件有统一的构造、初始化、运行、停止、卸载、消息响应函数, 各应用组件通过 C++的虚函数派生, 进行函数功能的重定义, 其基类结构如图 7 所示.

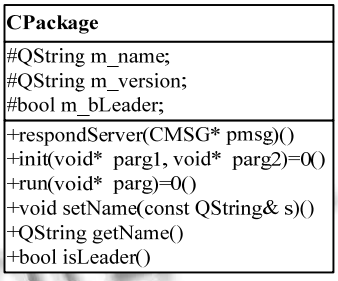

图 7 组件的 UML 模型

组件创建接口为 newPackage, 不同的应用组件返回各自实例对象, 例如工程管理组件创建接口实现为:

```
CPackage* newPackage()
{
    CProjectMng* ptr = new CProjectMng();
    return (CPackage*)ptr;
}
```

例如图形化编程组件创建接口实现为:

```
CPackage* newPackage(){
    CGuiVisProgram* ptr = new CGuiVisProgram ();
    return (CPackage*)ptr;}
```

组件加载的伪代码示例如下, 根据组件文件名, 用 QLibrary 导入, 之后调用统一的创建接口, 形成组件实例, 并放入组件库.

```
typedef CPackage* (*newFunc)();
QLibrary * plib = new QLibrary(packageName);
if ( !plib->load() )
{
    return false;
}
newFunc pfunc =
(newFunc)plib->resolve("newPackage");
if(   pfunc   )
{
    CPackage* ptr = pfunc();
    addPackage(ptr);
}
```

图 8 是跨平台组件的形成调用步骤.

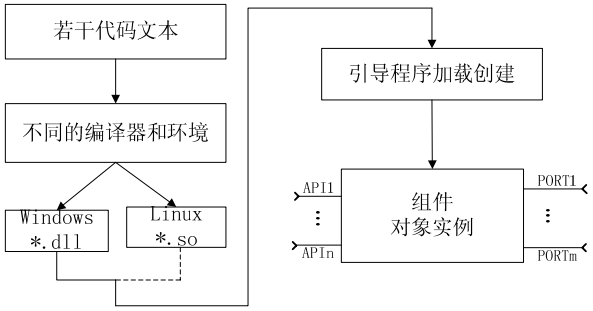

图 8　组件形成加载示例图

各应用组件在 windows 平台下编译为 DLL 文件, 在 linux 平台下编译为.so 文件, 主框架和引导程序不静态依赖具体组件.

1.5 应用实例

新一代保护测控装置配套工具软件[8]是基于组件集成框架开发的, 由资源管理器组件、装置信息组件、硬件配置组件、LCD 主画面组件、LCD 菜单组件、全局功能配置组件、图形化编程组件等构成. 资源管理器组件是默认的领导者组件, 负责文件的打开、关闭、保存和工程文件树形结构展示. 当点击不同节点时, 发送消息给相关组件, 由对应的组件展示界面, 进入活动状态. 图 9 是典型界面图.

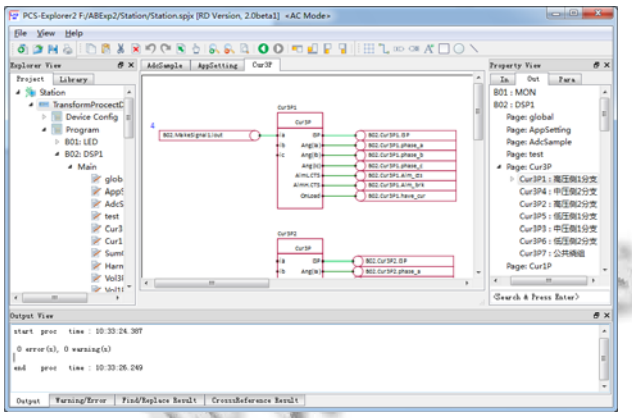

图 9　基于组件架构的工具软件示例

保护测控装置配套工具软件 PCS-Explorer 分研发版本、用户版本, 通过读取组件配置树文件, 在 main.cpp 中静态加载使能分支的组件结构. 例如: 在下述组件树配置文本中, key=”RD”的分支是研发版软件并且是使能状态, 当前平台为 win32, 其中工程管理组件是领导者组件, 在消息总线上的名字为 projectmng, 加载的文件名为 progrectmng.dll(在 linux 平台下, 加载的组件文件名替换为*.so), 平台切换时, 仅文件名变化, 组件名没有变化, 不需要修改源程序.

```
<package_tree>
  <app key="RD" enable="1" platform=”win32”>
    <item name="projectmng" file="projectmng.dll"
                                          leader="1"/>
  <item name="visprogram" file="guivisprogram.dll">
  <item name="devinfo"      file="guidevinfo.dll" >
  <item name="lcdmenu"    file="guilcdmenu.dll" >
      …
  </app>
  <app key="User"    enable="0"    />
</package_tree>
```

组件集成框架在开发过程中, 曾经出现过软件关闭时偶尔有异常退出现象, 后定位排查到是某组件调用 QScriptEngine 栈上对象的接口造成, 原因是 QtScript 模块也使用了 QLibrary 加载了系统库子组件, 组件指针不在主进程创建导致. 解决方案是在 main.cpp 中动态创建 1 个 QScriptEngine 全局指针对象 , 在所需使用组件中直接使用, 该问题解决后, 组件集成框架运行稳定, 该软件已经在 80 多个国家的用户中使用, 未出现异常情况. 基于组件化的开发方式, 模块之间高度解耦, 当修改功能时, 只修改部分组件, 修改的影响范围可控.

2　后续工作思路

本文所提出的组件集成框架, 目前还是基于组件配置树静态加载所需组件, 尚未实现动态加载功能. 当存在下述情况时, 需实现动态加载: 2.0 版本的软件需兼容支持打开 1.0 版本软件的装置驱动包, 而 2 个版本的驱动包文件格式有变化, 需根据文件后缀动态决定加载相关版本的组件. 初步思路是拓展领导者组件的功能, 当导入不同后缀的工程文件时, 将文件名信息传递给主进程的组件加载模块 CPackageMng, CPackageMng 遍历指定目录下的组件名, 用 QLibrary 导入组件. 组件增加接口, 返回是否具备支持处理该类型文件的肯定或否定应答. CPackageMng 加载支持该版本的组件列表, 卸载否定应答的组件.

3　结语

本文将人机交互软件分为多层体系, 将具体应用功能处理模块和软件的主框架本体分离; 基于公共规

范开发不同应用模块, 通过加载替换不同的模块, 可实现快速集成发布对应场景的软件. 提供了一种基于消息的模块交互通信方法, 支持广播、组播、点对点的消息发送方式, 可按照约定的规范在公用消息类基础上派生不同模块间的私有消息类, 实现模块通信的自定义灵活扩展. 基于本文提出的人机交互软件组件集成架构已经在电力系统交流保护、工业控制等领域的配套软件中应用, 实践证明该框架具有良好的适应能力, 通过功能和界面分层, 降低了软件开发难度, 提高了软件开发效率. 下一步研究方向是基于能力描述匹配的组件动态加载的具体实现.

参考文献

1 陈宏君,刘克金,冯亚东,等.一种基于组件和脚本的可视化程序产物形成架构及应用.工业控制计算机,2013,26(12): 1–5.

2 陶传奇,李必信, Gao J,等.基于模型的构件软件修改影响分析.软件学报,2013,24(5):942–960.

3 丁博,王怀民,史殿习.构造具备自适应能力的软件.软件学报,2013,9(24):1981–2000.

4 袁伟民,左春.基于样本程序的领域开发平台的研究与实践.计算机工程设计,2010,31(18):3979–3982.

5 周晓锋,马志强,刘馨月.一种基于组件的软件开发方法.软件工程与标准化,2005,(9):35–38.

6 张弛.软件组件接口扩展技术研究.微电子学与计算机,2007,24(8):35–41.

7 何鹏飞,何平,张松阳,等.组件技术在嵌入式系统中的应用 .计算机系统应用, 2014,23(6):220–223.

8 陈宏君,刘克金,冯亚东,等.新一代保护测控装置配套工具软件设计与应用.电力系统自动化,2013,37(20): 92–96.