计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(2):76−82 [doi: 10.15888/j.cnki.csa.007294]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

通用虚拟仪器前面板运行平台设计①

陈昌浩, 李晓明

(浙江理工大学 机械与自动控制学院, 杭州 310018)
通讯作者: 李晓明, E-mail: lxm@lxm.name

摘　要: 随着科学技术的进步, 测量仪器技术逐渐向虚拟仪器技术方向发展, 开发简单、通用、可拓展已成为虚拟仪器开发的重要指标. 为了解决目前市面上虚拟仪器前面板开发软件不易于拓展、运行效率较低的问题, 提出了一种通用虚拟仪器前面板的运行平台的实现方法, 其主要原理是: 界面组件化, 前面板界面由组件组合而成, 仪器组件以 jar 包形式存在, 方便拓展, 平台加载不同的仪器脚本, 即可生成不同的虚拟仪器. 提出了一种基于观察者模式的一对多的数据交换机制, 方便组件间通讯, 也可拓展与外部系统通讯. 信号发生显示器的例子证明了该通用仪器前面板运行平台的可行性.

关键词: 虚拟仪器; 运行平台; 组件

引用格式: 陈昌浩,李晓明.通用虚拟仪器前面板运行平台设计.计算机系统应用,2020,29(2):76–82. http://www.c-s-a.org.cn/1003-3254/7294.html

Design of Universal Virtual Instrument Front Panel Running Platform

CHEN Chang-Hao, LI Xiao-Ming

(Faculty of Mechanical Engineering and Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: With the advancement of science and technology, measuring instrument technology is gradually developing towards virtual instrument technology. Simple, universal and expandable development has become an important indicator for virtual instrument development. In order to solve the problem that the virtual instrument development software on the market is not easy to expand and the operation efficiency is low, a method for realizing universal virtual instrument front panel running platform is proposed. The main principle is that the platform instrument component exists in the form of jar package, which is convenient for expansion and platform loading. Different instrument scripts can generate different virtual instruments, and a one-to-many observer-based data exchange mechanism is proposed to facilitate communication between components and facilitate communication between platform and hardware system. An example of a signal generation display demonstrates the feasibility of the universal operating platform.

Key words: virtual instrument; operating platform; component

虚拟仪器 (Visual Instruments, VI) 是指结合软、硬件资源构造测量仪器, 即用户在通用计算机的硬件和软件平台上, 可自定义仪器操作面板和测量功能的软件系统[1]. 软件在虚拟仪器中有着举足轻重的地位, 美国国家仪器公司 (NI) 曾提出“软件即仪器”(Software is Instrument) 这一理念, 形象的概括了软件的重要地位, 它担负着仪器系统数据分析处理、实时显示、动态修改等重担[2]. 而衡量一款虚拟仪器软件的好坏很大程度上取决于前面板设计的方便与否. 目前, 虚拟仪器前面板的开发主要有两种: 一种是基于文本语言来编写仪器软件的底层驱动、数据处理算法、虚拟仪器面板显示等, 这类语言主要有 C/C++, Java, C#等. 近年也出现

① 收稿时间: 2019-07-05; 修改时间: 2019-08-20; 采用时间: 2019-08-30; csa 在线出版时间: 2020-01-16

了使用诸于 G 语言[3], Unity3D 等新兴编程语言来完成虚拟仪器软件的开发[4], 另一种是以 NI 的 LabView 为代表的可视化图形编程语言开发方式[5]. 两种方式各有优缺点, 前者运行效率高, 容易实现复杂的数据处理算法, 有较为完备的图形框架, 容易自由定制复杂功能的仪器面板界面, 但是编程语言上手门槛较高, 学习成本大, 开发周期较长, 后者使用图形化界面开发, 容易上手, 开发周期短, 但是有不易于拓展、运行效率低等缺点.

目前国内外对仪器软件开发平台的研究不多, 大多数科研人员都是应用仪器技术来解决自身领域的问题[6,7]. 特别是国内, 虽然有一些学者, 如华中科技大学的学者基于 iOS 的虚拟仪器浏览器[8]、基于前端响应式技术的可重构手机虚拟仪器[9]、基于 Unity3D 的虚拟仪器模型[4], 浙江大学学者设计的虚拟仪器共享平台[10]等, 虽然取得了一些成果, 但大部分虚拟仪器测试系统主要还是基于国外软件平台的二次开发和硬件方案提供. 虽然国外以 NI 的 LabView 为代表的虚拟仪器开发软件在重用和硬件解耦等方面有很大的改善, 但还是有些不足: 软件模块的开发闭源, 用户只能使用软件提供的模块, 对于软件进行功能拓展还是需要重新开发, 开发成本高[11].

基于以上的不足, 本文借助 Java 语言和 SWT/JFace 图形包, 研究并实现了通用虚拟仪器前面板的运行平台, 定义了一套仪器显示组件的标准, 平台界面由各个不同的组件组合完成, 开发者可根据需求自己设计组件拓展. 提供了仪器软件与硬件系统的数据交换的接口, 减少了软件硬件的耦合性, 使得应用软件能适用非主流特别是自主研发的硬件系统.

1 通用虚拟仪器前面板运行平台工作原理

仪器前面板运行平台既可独立运行, 也能作为基础平台嵌入其他自动测试系统, 它的工作原理如图 1 所示, 平台加载保存有写好的组件的 jar 包, 加载来自网络或本地的仪器脚本, 解析脚本生成仪器前面板虚拟界面, 并与底层硬件系统通讯, 交换硬件系统和虚拟面板中的数据. 具体工作方式如下:

脚本: 一般是 XML 文件, 存放界面中组件的信息[12], 包括容器组件和一般组件, 脚本文件可以是本地文件也可以是网络文件. 本地文件存放在指定的文件夹中, 在 Windows 和 Linux 操作系统中, 一般存放于项目所在根目录下的 UI 文件夹中, 网络文件存储在其他 PC 机器之中, 如果远程用户想运行平台, 就可以通过 TCP 协议传输网络脚本数据到平台. 为区别于以 .xml 结尾的配置文件, 脚本文件的格式设置为.uixml

组件 jar 包: 平台采用模块化的设计, 组件独立于平台, 存放在 jar 包中, 方便拓展. 一般将 jar 包放入一个文件夹中, 统一管理, 在 PC 系统中, 组件 jar 包通常存放于 UI 文件夹下的 component 文件夹.

界面运行: 平台加载脚本解析之后生成可视化图形界面, 用户可以直接操作面板, 平台提供数据交换支持. 平台使用 Java 语言和 SWT/JFace 图形包来生成图形界面, 满足跨平台需要, 该平台和界面可以运行在 Windows 系统和 Linux 系统之上.

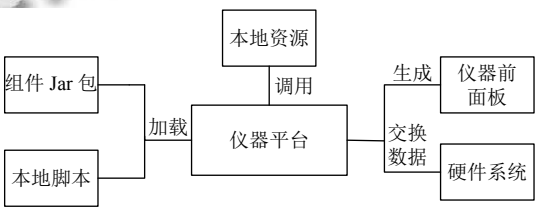

图 1 前面板运行平台工作原理

2 虚拟仪器前面板运行平台实现方法

虚拟仪器的基本组成模块是组件, 组件是具有某种功能的独立模块, 依据实现功能的不同有容器组件和功能组件, 组件之间的关系如图 2 所示. 容器组件是用来安放其他组件的组件, 功能组件是实现人机交互功能的组件. 将容器组件和功能组件按照一定顺序组装好就能形成一个虚拟仪器. 下文将围绕控件介绍运行平台的实现方法.

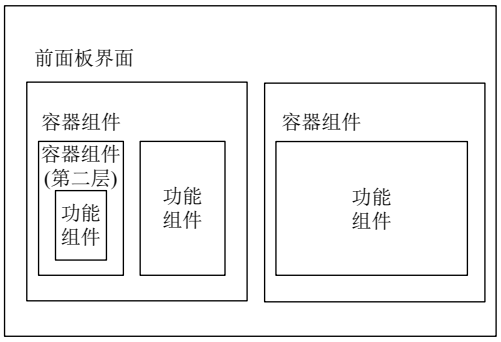

图 2 前面板界面示意图

2.1 基于观察者模式的数据交换机制

观察者模式是一种一对多的设计思想, 多个观察者订阅同一个主题时, 当主题有变化时, 通知观察者获

取新的主题, 当一个观察者改变主题的属性值时, 其他观察者也能及时接收到改变[13]. 组件与组件之间需要通讯, 组件与硬件系统之间也需要通讯, 为了解决这种一对多的依赖关系, 本课题采用了数据池数据绑定的数据交换机制: 平台中开辟一个数据池, 采用 KVO (Key-Value-Object) 的方式存储数据, key 对应一个属性, value 对应组件的属性值, 一个属性可以被多个组件绑定; 同时, 数据池中的数据也可以通过接口和硬件系统交换数据.

数据池数据绑定的工作原理是: 数据池中定义一个数据对象 A, 组件 B 需要对象 A 的数据, 组件 D 也需要对象 A 的数据, 硬件系统中的硬件模块 C 输出的数据对应对象 A 的数据, 当 C 交换数据将对象 A 的数据改变时, 数据池会发送通知给组件 B 和 D, 告诉 B 和 D 对象 A 的最新数据. 数据绑定的关系如图 3 所示.

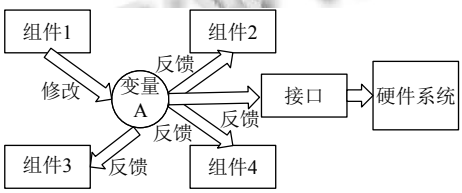

图 3 数据绑定关系

数据池使用单例设计模式, 平台中只有一个数据池, 要绑定的数据都在数据池中操作, 使用数据绑定的方法有: register (String tag, IDataBinder binder); //注册变量到数据池, bind (String tag, IDataBinder binder); //组件绑定变量, publish (String tag, Object value, IDataBinder binder); //改变变量的属性值, 通知绑定的其他组件新的属性值; IDataBinder 是一个数据绑定的接口, 组件都要继承该接口, 接口中只有一个方法: newValue (String tag, Object value), 每个组件都要实现该方法, 数据池使用组件的该方法通知到各个组件新的属性值.

2.2 组件热加载机制

平台使用了模块化的设计思想, 组件是一个个独立的模块, 可以自由的拓展. 为了方便组件的添加, 可将设计好的组件打包成 jar 包, 存放到项目根路径下的 UI 文件夹下面, 在使用到这个组件时, 平台动态加载 jar 包寻找该组件, 并使用 Java 反射机制创建组件.

2.2.1 双亲委派机制

双亲委派机制是 JVM (Java 虚拟机) 加载类的机制. JVM 虚拟机根据类的全限定名来加载类, 而类加载器是 JVM 加载类的实现, 图 4 是类加载器之间关系, 双亲委派机制是为了避免不同的类加载器加载同一个类[14].

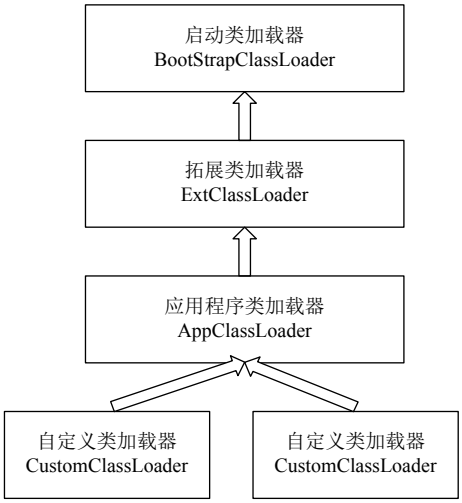

图 4 类加载器之间关系

双亲委派机制的工作流程如下:

(1) 类加载器收到类加载请求;

(2) 把这个请求委托给父加载器去完成, 一直向上委托直到启动类加载器;

(3) 类加载器检查能不能加载这个类, 如果可以就加载这个类并结束加载, 如果不能, 抛出异常, 通知子加载器进行加载;

(4) 循环步骤 (3).

2.2.2 运行时类加载器

编写好的静态代码通常是由 AppClassLoader 类加载器来加载, 但是这种类加载器只能加载在编译之前就已经写好的代码, 这样组件信息和平台耦合在了一起. 根据双亲委派机制, 本方案使用了 URLClassLoader 来加载 jar 包中的组件, URLClassLoader 是拓展类加载器的一种, 是 JDK 提供的一种可加载外部文件中的类的信息的类加载器, 配合应用程序加载器即可加载平台中预写好的组件和外部拓展的组件. 本文中, 我们对 URLClassLoader 进行封装, 使用 RuntimeClassLoader 来处理组件 jar 包的加载, 组件加载的步骤如图 5 所示.

平台每次启动的时候读取路径信息, 加载组件 jar 包到 classPath 中, 让 JVM 能够寻找到组件的信息, 加载的过程使用 RuntimeClassLoader 的 addJarToPath (String path) 方法; 当有脚本运行时, 平台解析脚本文件, 读取其中的组件类名和初始化参数, 使用 Runtime

ClassLoader 的 loadClass (String className) 提供反射支持, 生成组件, 同时调用组件的 init() 方法加载初始化参数完成组件的初始化.

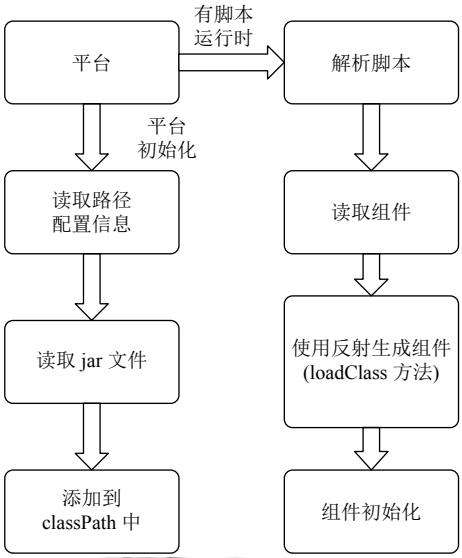

图 5 组件加载过程

2.3 组件定义及装配

2.3.1 组件模型

组件根据功能侧重点的不同, 可分为容器组件和功能组件, 这两种组件之间有一些共同点, 也有一些不同点, 容器组件可看做是功能组件的拓展, 具有可放置组件的容器. 组件的模型如图 6 所示.

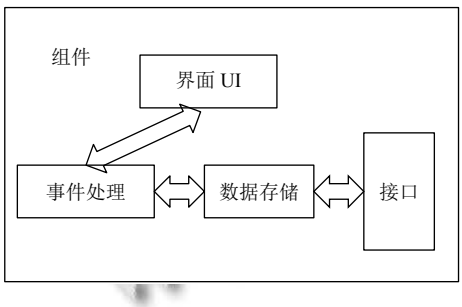

图 6 组件模型

组件包括界面显示的内容以及人机交互处理, 包含两大部分: 显示部分和数据部分, 显示部分显示组件的外貌, 数据部分处理与用户有关的输入和输出, 以及相关的事件. 本方案设计中事件也是按照数据的方式进行处理.

容器组件可看做特殊的组件, 其界面 UI 不止有实现部分, 还有可放置组件的容器凹槽, 起到布局定位的作用, 容器组件界面 UI 如图 7 所示.

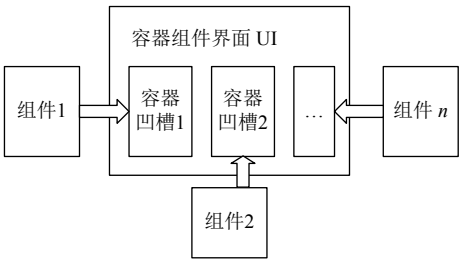

图 7 容器组件界面 UI

组件的装配原理: 将组件放入容器凹槽即可.

2.3.2 组件类定义

平台的所有组件必须遵从相同的接口原则, 这样才能相互通讯, 也方便拓展. 为了统一组件的开发方法, 提高开发效率, 本文设计了组件的接口, 及其基础实现类, 开发的组件只需要继承基础实现类, 按照功能需求重写相关的方法即可快速开发该组件, 组件接口和实现类的关系如图 8 所示.

Component 接口定义了所有组件都需要用到的一些方法, Container 接口继承 Component 接口定义了容器组件需要用到的方法, ComponentAdapter 抽象类实现了 Component 接口, 给一些方法提供默认实现, ContainerAdapter 抽象类继承 ComponentAdapter 并实现 Container 接口给容器组件的一些方法提供默认实现. 除默认实现方法外, 拓展的组件还需重写和实现部分方法, 如表 1 所示.

2.3.3 组件组合原则

容器组件规范了组件的相对位置,、大小以及其他装饰物等. 而功能组件专门处理人机交互功能, 功能组件可以简单到类似于一个标签, 一个按钮, 也可以复杂到一个功能完备的示波器界面. 容器组件侧重布局, 而功能组件侧重交互处理, 所以需要将两种组件恰好的组合在一起, 形成前面板的界面显示[15]. 下面是组件组合的一些原则:

(1) 容器组件上可以放置任意界面组件, 功能组件只能放置在容器组件内.

(2) 任何一个程序界面至少包含一个容器组件.

(3) 容器组件放置组件的位置如果已经放置有组件, 不能再放置组件.

2.4 组件装配与解析

2.4.1 仪器脚本描述方法

平台采用脚本作为描述组件组合的工具, 脚本语言采用可拓展标记语言 (eXtensiable Markup Language,

XML)[16]. XML 语言支持拓展, 我们使用自定义标签描述组件以及组件的属性和数据绑定[17,18]. 为方便识别组件的属性和数据绑定, 采用组件标签嵌套属性标签的格式来描述组件. 根据组件功能特性的不同, 我们采用<component>, <container>两种标签分别来描述功能组件和容器组件.

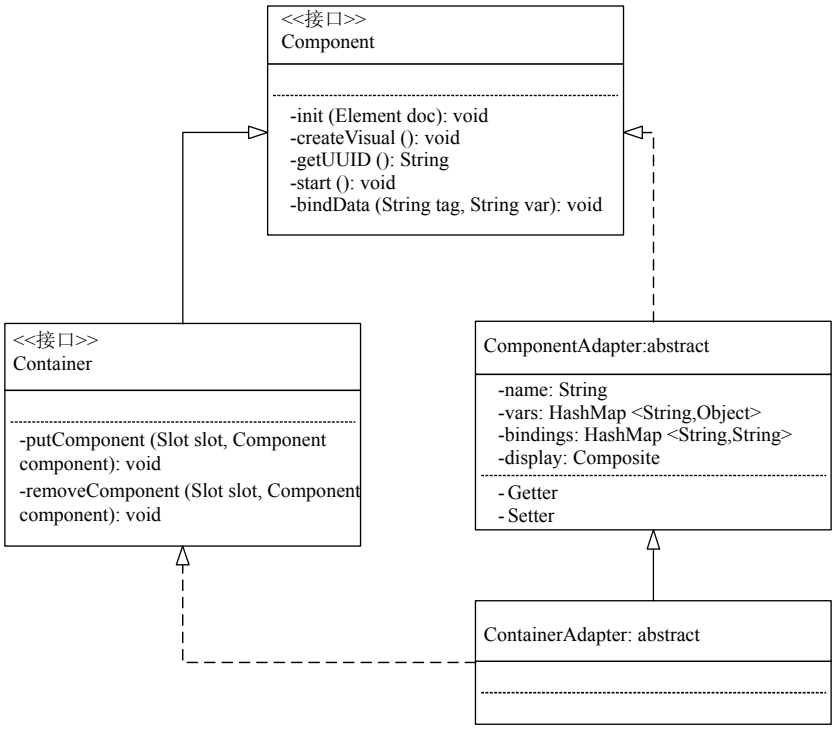

图 8 组件类图

表 1 组件类实现和重写方法

方法	说明	操作
-GetUUID(): String	获取组件的唯一 ID	实现
-CreateVisual(): void	创建组件的图形界面	重写
-bindData(String tag, String var): void	绑定数据	重写
-init(): void	初始化组件	重写
-init(Element element): void	根据脚本参数初始化组件	重写
-start(): void	启动组件的事件处理和数据交换机制	重写

功能组件的通用描述格式如下:

```
<component name="xx" class="xx">
  <param name="xx" value="xxx"/>
  <param name="xx" value="xxx"/>
  <data-binding local="xx" external="xxx"/>
</component>
```

Component 标签表示该组件是功能组件, name 表示组件的名称, class 表示组件的全限定类名, 平台解析脚本时需要根据类名使用反射来生成组件, param 标签描述组件的属性, 组件需要根据 param 标签的属性完成初始化.

Data-binding 标签表示数据绑定, local 对应组件自身的属性的数据绑定, external 表示要对外进行数据交换的属性, 即该标签描述了组件的输入输出端口. 某些仪器功能中, 有的组件可能相对独立, 不受外部系统的输入输出影响, 可以省略 external 属性, 其他属性根据组件的功能不同而变化.

容器组件的通用描述格式如下:

```
<container name="xx" class="xxx">
  <param name="xx" value="xxx"/>
  <param name="xx" value="xxx"/>
  <slot name="xx" ratio="xxx"></slot>
  <slot name="xx" ratio="xxx"></slot>
</container>
```

Container 标签该组件是容器组件, 相比于功能组件省略了数据绑定的描述, 添加了 slot 标签描述组件的布局位置, slot 标签表示容器凹槽, name 是凹槽的名

称, ratio 是凹槽占该容器组件的比例, 容器凹槽标签下可以继续嵌套其他组件标签.

仪器脚本由组件组合而成, 平台定义一个 XML 脚本表示一个仪器前面板, 使用标签<application>来描述, 即该标签是仪器脚本的根元素. 根元素下可放置组件, 若组件为容器组件, 容器凹槽内可继续嵌套组件, 若干个组件一层层嵌套就组合成一个布局和功能完整的仪器脚本.

下面一段脚本描述了一个包含启动和停止按钮的仪器界面:

```
<application>
  <container name="bottom" class="FillContainer">
    <param name="ratio" value="1:1"/>
    <param name="direct" value="HORIZONTAL">
    <slot name="start" ratio="1">
     <component name="startButton" class="InstrumentButton">
    <param name="name" value="startButton"/>
    <param name="text" value="启动程序"/>
    <data-binding local="mouse.down" external="startButtonOut"/>
    </component>
    </slot>
    <slot name="stop" ratio="1">
     <component name="stopButton" class="Instrument">
    <param name="name" value="startButton"/>
    <param name="text" value="停止程序'/>
    <data-binding local="mouse.down" external="stopButtonOut">
    </component>
    </solt>
  </container>
</application>
```

appliaciton 表示该脚本文件是仪器前面板脚本, FillContainer 容器组件包含两个 InstrumentButton 按钮组件, ratio 属性设置其各容器凹槽的布局, direct 属性设置容器凹槽的水平和垂直排列, InstrumentButton 是一个按钮组件, name 是组件的名称, text 是按钮组件显示的文本, 两个组件都绑定了 mouse.down 的内部属性, 启动按键绑定数据常量池 startButtonOut 对象, 停止按键绑定 stopButtonOut 对象, 如果其他组件想要响应按钮, 只需绑定按钮对应的数据常量池对象即可.

2.4.2 仪器脚本解析

脚本文件描述了组件的属性和组件组合的方式, 平台的作用是按顺序完整读取脚本并生成前面板界面. 平台解析脚本的流程如图 9 所示.

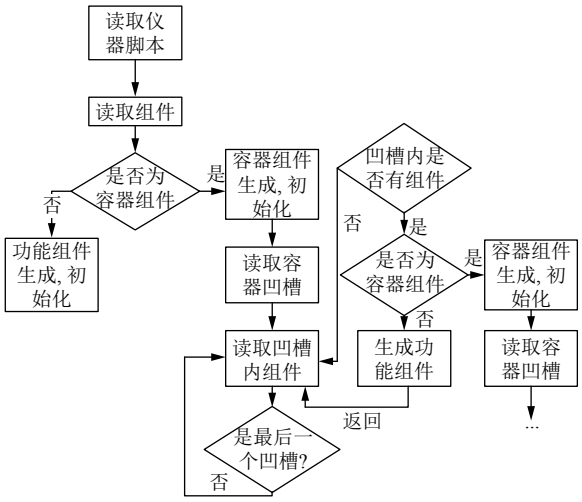

图 9 平台解析脚本流程图

一个脚本文件我们默认为一个仪器前面板界面, 则脚本文件只能有一个根节点, 且该根节点标签为 application, 如果不是 application 或根节点多于一个, 则脚本读取失败. 读取根节点下组件生成组件对象, 读取组件的参数完成组件初始化, 如果组件为功能组件, 完成组件的数据绑定, 如果为容器组件则遍历读取容器凹槽内的组件, 若凹槽内组件为容器组件, 则递归调用处理容器组件的方法, 否则调用处理功能组件的方法[19]. 待组件都处理完成, 则启动平台, 生成仪器界面, 用户可在界面中与平台交互.

3 虚拟仪器前面板运行平台应用实例

自动测试实验中经常用到信号发生器和示波器, 信号发生器可以产生多种类型的信号, 示波器负责显示输入的信号, 本文将两种装置组合在一起, 设计了一个信号发生器显示装置.

它由 5 个组件组成, 一个示波器组件, 一个信号发生器组件, 一个启动示波器按钮, 一个停止示波器按钮, 一个控制台显示组件. 它的工作原理是: 用户点击示波器启动按钮, 平台接收到按钮状态的改变, 发送通知给

示波器组件, 示波器组件开始工作, 等待信号输入, 用户点击信号发生器按钮, 启动或停止信号发生器的工作, 按钮的工作状态改变会输出到控制台中方便用户观察当前状态. 发生器工作时, 每隔一定时间就生成一定幅度、频率、相移的信号数据, 并更新信号的数据池, 平台发送通知给示波器输入信号的改变, 示波器接收输入信号, 更新显示的波形. 图 10 是脚本运行界面.

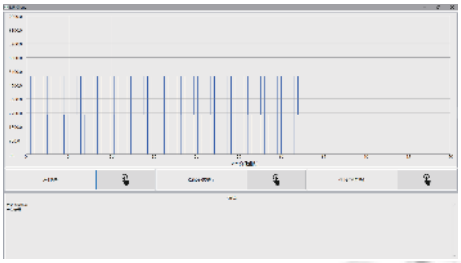

图 10 信号发生显示器运行界面

4 结论与展望

本文在分析了当前虚拟仪器的研究现状, 针对当前大多虚拟仪器开发软件不易拓展的问题, 提出了一种通用仪器软件的前面板运行平台, 采用模块化组件的方式, 用数据常量池数据绑定来交换数据, 以 XML 脚本语言为载体, 组合组件来满足不同的测试功能需求. 上文信号发生显示器的运行也说明了该通用仪器前面板运行平台的可行性. 不过仪器脚本文件的编写格式较复杂和冗余, 接下来的研究工作是设计更通用的脚本标签格式, 采用图形化的编辑界面来设计仪器前面板, 搭建一个较为完备的通用仪器软硬件开发平台.

参考文献

1 于洋, 于文波, 徐立波, 等. 虚拟仪器的发展现状及其应用. 洛阳师范学院学报, 2015, 34(2): 48–51. [doi: 10.3969/j.issn.1009-4970.2015.02.013]

2 伍星华, 王旭. 国内虚拟仪器技术的应用研究现状及展望. 现代科学仪器, 2011, (4): 112–116.

3 随阳轶. 虚拟仪器开发平台 LabScene 的软件设计[硕士学位论文]. 长春: 吉林大学, 2005

4 阚研, 何岭松, 谢道旺, 等. Unity3D 下的虚拟仪器实现. 软件导刊, https://www.cnki.net/KCMS/detail/42.1671.TP.20190820.1133.080.html. [2019-08-20].

5 汪迎, 冯家慧. 基于 PC 系统构成的虚拟仪器技术. 电子测试, 2016, (13): 4–6. [doi: 10.3969/j.issn.1000-8519.2016.13.002]

6 耿晨歌, 陈祥献, 汪乐宇. 面向虚拟仪器系统的可视化程序描述方法. 现代科学仪器, 1998, (4): 35–36.

7 谢宣松. G 语言的一种结构模型及平台实现[博士学位论文]. 长春: 吉林大学, 2006.

8 吴玉叶, 何岭松, 韦文姬, 等. 基于 iOS 的手机虚拟仪器浏览器的设计. 计算机测量与控制, 2017, 25(11): 234–238.

9 谢道旺, 何岭松, 高志强. 基于响应式编程的手机可重构虚拟仪器. 软件导刊, https://www.cnki.net/KCMS/detail/42.1671.TP.20190820.1133.082.html. [2019-08-20].

10 袁冬旭. 基于通用仪器平台的虚拟示波器设计与关键技术研究[硕士学位论文]. 杭州: 浙江大学, 2018.

11 刘传清, 胡荣玉. 虚拟仪器软面板设计技术. 襄樊学院学报, 2002, 23(5): 14–17.

12 Stoicov B, Ivanov Z. Visual designer for editing large schemaless XML file: US, 10223338. [2019-03-05].

13 欧阳宏基, 杨卫忠, 赵蕾. 观察者模式在 Java 事件处理中的应用研究. 微处理机, 2013, 34(4): 77–79. [doi: 10.3969/j.issn.1002-2279.2013.04.022]

14 孙彬. Java 动态类加载机制应用研究. 科技创新与应用, 2018, (23): 180–181.

15 钱远鹏. 基于 SWT 元数据提取的研究与实现[硕士学位论文]. 北京: 北京邮电大学, 2018.

16 王华, 雷镇. 基于 XML 的用户界面语言研究. 微计算机信息, 2006, 22(7–8): 242–243, 259.

17 吴敏, 丁永生, 陈家训. XML 的研究现状及展望. 微型电脑应用, 2001, 17(4): 5–9. [doi: 10.3969/j.issn.1007-757X.2001.04.001]

18 张迪, 朱敏, 张凌立. 基于 SAX 的 XML 解析与应用. 计算机与数字工程, 2008, 36(7): 103–106. [doi: 10.3969/j.issn.1672-9722.2008.07.030]

19 肖红德. 汉诺塔问题递归算法与非递归算法比较. 软件导刊, 2018, 17(8): 118–120.