# 基于 Ruby 和 Flash 的数据交换系统研究与实现

Research and Implementation of Data Exchange System Based on Ruby and Flash

鲍 陈 朱 伟 张云华 （浙江理工大学信电学院 杭州 310018）

**摘 要：**本文针对目前 Ruby 平台与 Flash 平台不能进行数据互操作的缺陷，通过在 Ruby 平台上调用 ActiveX 控件的方法来实现 Ruby 平台与 Flash 平台数据端口的协同访问，结合 XML 技术，给出了一种基于 XML 的自动售货机用户界面管理系统数据交换模型，最后具体阐述了在该模型中实现 Ruby 和 Flash 异构平台数据交换的关键部件及解决方法。

**关键词：**Ruby Flash 可扩展标记语言 ActiveX 数据交换

## 1 引言

Flash 是 Maromedia 公司推出的一种多媒体交互式矢量动画创作软件[1]。目前 Flash 技术作为一种全新的设计方式，已经渗透到音乐、传媒、游戏等各个领域。

Ruby[2]（中文意思为红宝石）是一种纯面向对象编程的解释性脚本语言。具有如下的特点：任何事物都是一个对象，迅速和简便，无须变量声明，变量无类型，语法简单而坚实，自动内存管理，多精度整数，异常处理模式，动态装载，线程安全。Ruby 运行平台目前的稳定的版本是 Ruby 1.8.5 可以从网上下载，同时下载添加 ActiveX 控件 VisualuRuby 来支持 Ruby 平台与 Flash 平台交互控制。

为了扩展现有投币式自动售货机的销售方式，我们开发了自动售货机用户界面管理系统[3]，它是我们开发的新型多媒体自动售货机的一部分，该系统是自动售货机软件系统应用层的重要组成部分。该系统中采用了多媒体表现形式，用 Flash 技术实现用户交互和自动售货机销售产品信息展示。

由于目前 Flash 平台和 Ruby 平台之间不能进行交互操作和信息共享。本文提出了一种新的解决方案：通过调用 ActiveX 控件方法来实现 Flash 展示平台与 Ruby 控制平台的互操作，以 XML 作为中间数据交换格式，通过 DOM 对 XML 文件进行信息抽取，实现异构平台数据交换[4]使得系统更加健壮和灵活。

## 2 ActiveX 互操作协议

ActiveX[5] 是 Microsoft 提出的一组采用 COM（Component

Object Model，组件对象模型）使软件组件能够进行交互的技术集，与具体的编程语言无关；它包括 ActiveX DLL 组件和 ActiveX 控件。ActiveX 控件同其它的 ActiveX 组件相比具有以下特点：能通过设置属性控制其行为，从而实现与数据接口交互。其通讯协议描述如下：

(1) ActiveX 控件

ACTIVEXCINFO = [ " ShockwaveFlash. ShockwaveFlash" ," _IShockwaveFlashEvents" ]

(2) Flash→Ruby 发送命令

Flash 动画的 AcitonScript 中提供一个 Fscommand 动作，提供只要 Flash 动画发出这个动作，就触发 Shockwave Flash 控件的一个 Fscommand 事件。Fscommand 命令的语法如下：Fscommand（command，arguments）；其中参数 command、arguments 都是向 Ruby 所发送的字符串命令。对应的 Ruby 响应事件为 fl_fscommand（cmd，arg）其中两个字符型参数 cmd 和 arg 都是由 Flash 端 Fscommand 动作传递过来的参数。因此我们可以编写事件响应代码，根据 Flash 的 Fscommand 动作传递过来的 Command，在 Ruby 应用程序中完成相应功能。

(3) Ruby→Flash 发送命令

Ruby 的 ActiveX 控件可通过一定的方法直接操作或者调用 Flash 动画的 Action 脚本代码。主要包括：SetVariable，这是向 ActionScript 脚本中的一个变量赋值。

## 3 半结构化数据模型建立

XML 是标准通用标记语言 SGML（Standard Generic Markup Language）的一个子集。XML 文档内部采用树形结构的形式描述数据，可以表示各种结构化与半结构化的数据形式。各种各样的页面元素如图片、声音文件是使用得最多的非结构化数据，通常也是页面显示的重要依据。对于非结构化数据的处理通常是将其转化为结构化数据来处理，具体到对图形、声音文件的处理是将图形文件所在的路径、类型等结构化的数据信息保存到 XML 文档中。需要提出该图形、声音文件时通过读出 XML 文档内容获得文件的路径，然后再根据路径提取文件。为了能够方便地进行同类型而不同结构的数据之间交换及转换，需要找到一个高度灵活的、能够表达同类而异构数据的模型。XML 目前用于建立数据模型的手段有两种：文档类型定义（Document Type Definition，DTD）和 XML 大纲（XML Schema）。本文以前者为背景对 XML 的建模方法进行介绍。以下是页面 XML 文件的 DTD 格式：

```
<!ELEMENT screen (page+)>
<!ATTLIST screen
    name CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    w CDATA #REQUIRED
    h CDATA #REQUIRED
    soundPath CDATA #REQUIRED
    imagePath CDATA #REQUIRED
    welcomePage CDATA #REQUIRED
    desc CDATA #REQUIRED
>
<!ELEMENT page (image | button | keyboard | animateImage | radioButton)+>
<!ATTLIST page
    name CDATA #REQUIRED
    bkImage CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    desc CDATA #REQUIRED
>
<!ELEMENT image EMPTY>
……
<!ELEMENT button (hitArea?)>
……
<!ELEMENT keyboard (image, textbox, button+)>
<!ATTLIST keyboard
    name CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    keyChars CDATA #REQUIRED
>
<!ELEMENT animateImage (animateItem+)>
……
<!ELEMENT radioButton (radioItem+)>
……
```

## 4 基于 XML 数据交换系统模型

自动售货机用户界面管理系统为顾客、商家提供了一个宣传和展示产品形象的舞台，它具有如下基本特点：①内容和形式多样化：图像、文本、声音和动画等媒体形式。②页面显示方式多样化：显示页面由多媒体信息组成；页面轮流交替显示。③系统包括各种与商品销售相关的，如顾客登录、顾客信息调查、商品详细说明等。

基于 XML 的系统数据交换模型如图 1 所示，它包括两个平台，分别是 Flash 展示平台和 Ruby 控制平台。Flash 展示平台根据展示的实际需求定义了页面 XML 文件，通过 Flash 的 XML 类直接从页面 XML 文件中加载数据使用，实现系统的数据展示的功能。Ruby 控制平台包括：①配置管理子模块：实现了系统的动态配置和扩展功能。②主控制管理子模块：包括事件收发器的容器和页面对象容器，它通过调用 ActiveX 控件方法实现与 Flash 展示平台进行交互式操作。③任务调度子模块：包括事件解析器和迁移状态收集器，它的用途是将收到的事件进行解析，然后将解析后的事件发送到主控制管理子模块。④页面调度子模块：包括页面

元素解析器和页面容器,它实现整个系统页面数据的转换工作,将转换的页面元素和页面响应动作通知任务管理子模块。最后利用 Socket 通讯自动售货机通讯 Ruby 程序库,实现与自动售货机之间的通讯,从而真正意义上实现自动售货机与 Flash 页面的协同工作。

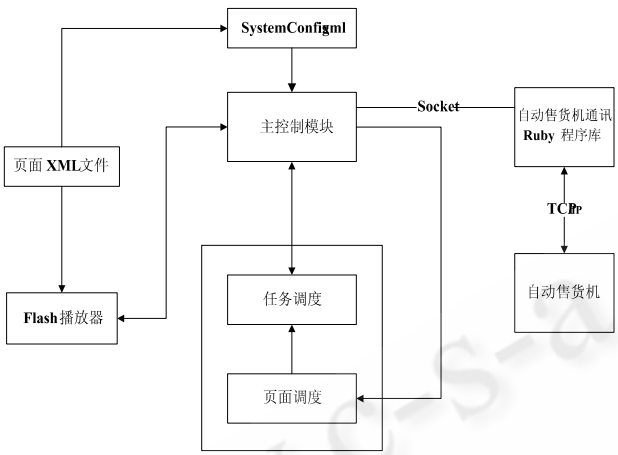


图1 基于 XML 的系统数据交换模型

## 5 关键部件及技术

本系统由两个平台组成,分别作为 Flash 展示平台和 Ruby 控制平台。

Flash 展示平台:开发环境为 Flash MX 和外部页面 XML 文件。通过 Flash 的 XML 类来对 XML 数据绑定和解析,实现了页面动态显示,并利用 Flash 的 ActionScript 脚本所提供的 Fscommand 命令可以使界面层接收它的运行环境发送来信息。

Ruby 控制平台:开发环境为 Ruby RDE 平台和支持 Flash 格式动画显示 ActiveX 控件,它可以从网上下载后安装,来支持与 Flash 的交换控制。Ruby 平台负责控制页面调度和事件响应处理。

### 5.1 Flash 展示模块

在 Flash 中已经内置了 XML 对象,我们可以很方便的通过生成一个 XML 对象实例来直接对 XML 对象中的属性数据进行操作。这样只要我们指向 XML 对象的不同部分,就能访问所形成的 XML 树的任意部分。Flash XML 解析器和对象,它们都是在“瘦客户”的基础上达到的。Flash 只把 Ruby 运行平台当作加载 Flash 动画的容器,而不是显示具体内容的工具。Flash +XML 是一种执行效率高,跨平台兼容性好,Flash 展示模块通过 ActionScript 脚本所提供的 Fscommand 命令接收 Ruby 控制平台控制指令,通过约定 Flash 与 Ruby 两端统一的交信格式指令,利用 Flash 调用 XML 文件方法来加载页面 XML 文件相关部分,通过 Flash 的 ActionScript 脚本语言的 LoadSound()、LoadMovie()函数实现相应的 jpg 图片和 mp3 声音文件的加载。

### 5.2 Ruby 控制模块

#### 5.2.1 配制管理

配置管理:在本系统中,配置信息主要包括系统基本信息,数据源信息(页面 XML 文件)、目标对象(Flash 播放器),这些信息都是以 XML 格式的文档保存的。对配置进行管理是 SystemConfig 类,它完成了系统基本信息的配置:Flash 播放器路径,页面 XML 文件路径。利用 Ruby 的 XML DOM 解析器 REXML 解析配置文件(SystemConfig.xml)。

其代码如下所示:

```
require 'rexml/document'   #需要 Ruby 的 DOM 解析器支持
class SystemConfig
    #基本配制信息管理
    ……
    xmlConfig = File.new("SystemConfig.xml") #载入配制文件 SystemConfig.xml
    xml = REXML::Document.new(xmlConfig) #得到一个 DOM 解析器对象
    #解析配制文件 SystemConfig.xml
    ……
end
```

#### 5.2.2 主控制管理

主控制类是整个系统的起点,也是用户或应用程序与系统进行交互操作的接口。在主控制类 MainForm 中通过调用 ActiveX 控件的 LoadMovie 方法加载 Flash 播放器目标对象,并实例化 EventParser 类、StateTrans 类、XMLDataParser 类、SystemConfig 类和 Page 类,利用事件收发容器和页面对象容器 Pages 向 Flash 平台发送事件来进行相应的页面的转换。图 2 体现上述对象间传递事件的时序。

如图 2 所示,首先 EventParse 对象接收到来自页面类 Page 对象传递 ParseEvent()消息。在 EventParse 对象中,首先通过调用 StateTrans 对象的 searchState()、getNextPageName()和 getNextCmd()方法来进行

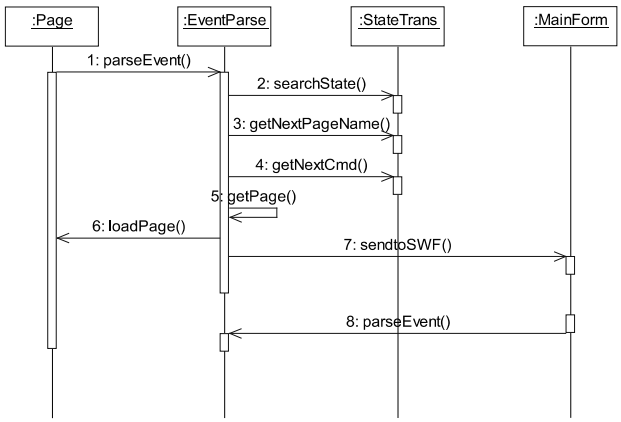


图 2　主控制管理时序图

相应的消息的解析，接着调用 getPage()方法，取得页面名，紧接着进行页面加载（调用 loadPage()方法），最后在 EventParser 对象通过调用 MainForm 对象中 sendtoSWF()方法将解析的消息发送给主控制类的事件收发器，在 MainForm 对象中调用 EventParser 对象 parseEvent()方法。

5.2.3 任务调度

任务调度是整个系统的核心部分，所有任务的解析、加载、发送都在此处完成。用户所有的工作都是通过任务管理器以执行任务的方式来完成的。它是由迁移状态收集器 StateTrans 类、任务解析器 EventParser 类、事件队列 Task，3 个核心类组成。下面是 3 个类的功能及实现。

（1）迁移状态收集器

迁移状态收集器，主要是由事件迁移状态配置矩阵组成，其结构包括当前接收到的 Flash 的脚本 Fscommand()函数发送的 Command 和 Argument 字符串参数以及该状态下一个对应的 Command 和 Argument 字符串参数。

（2）任务解析器

任务解析器负责根据主控制类 fscommand()方法接收来自 Flash 端的相应的参数 cmd，arg 字符串，任务解析器通过查找事件状态收集器，取得下一个需要加载的 pageName 名称和相应的 action 指令，发送给主控制管理类的事件收发器。

其代码如下所示：

```
require 'StateTrans' #需要调用 StateTrans 模块
class EventParser
    #基本配制信息管理
    ……
    def parseEvent(cmd,arg)
        case cmd
        when（条件）
    从 StateTrans 中取得下一状态信息 pageName 和 action
    return if action == nil and pageName == nil
    @page = getPage($StateTrans.getNextPageName)
    #从 StateTrans 状态收集器中取得 pageNextName
      @page.loadPage #调用 Page 类中的 loadPage 方法加载
          @callback.call(action, pageName) if @callback != nil  #调用回调函数
      end
    end
end
```

（3）事件队列

事件队列用于依次保存由页面调度模块对象接收的已触发的事件，是一个具有先进先出的（FIFO）特性的链表数据结构，可以保存多个事件信息的节点，这一数据结构的最大好处是无须等待前面已经触发事件全部处理完毕之后再来接收新触发的事件，从而支持页面管理模块的多次事件的触发，用于事件的注册和事件的调度。事件注册采用了 Ruby 语言的语言物性——反射技术。反射主要是用于动态地调用一些例实例或类型的方法。在 Java 与 C#等静态语言中是十分普遍的。动态地调用方法的一个途径是使用 Method 对象。有了 Method 对象，就可以在需要的时候通过发送消息 call 来运行它。在运行期可以根据任务配置信息动态加载，从而使数据交换能够随需而变，实现动态配置。

5.2.4 页面调度

页面调度子模块主要负责整个系统页面数据的转换工作，将转换的页面元素和页面响应事件通知任务调度子模块。它包括页面元素解析器 XMLPageParser 类和页面容器 Page 类。页面元素解析器 XMLPageParser 通过调用配置管理子模块中数据源信息（页面 XML 文件），利用 Ruby 的 XML DOM 解析器

REXML 进行 XML 文档解析。页面容器工作流程,首先根据 MainForm 类中提供的页面对象容器 Pages,它调用 Page 类的 initialize() 方法得到页面名 pageName;接着 MainForm 类中的通过 XMLPageParser 对象的方法解析应用级 XML 文档相应页面名 XML 片段,在 XMLPageParser 类中调用 Page 对象的 loadPage() 方法加载页面元素;每一个页面 Page 类中调用 EventParser 对象的 parserEvent() 事件解析方法对页面的状态和响应 Flash 端动作进行解析;最后通过 Ruby 反射机制 method 对象的 call 方法,把解析后的事件通过 MainForm 类的 sendtoSWF() 方法发送给 Flash 平台,Flash 平台根据相应的事件状态进行页面动态加载。

## 6 结论

本文提出了一种 Ruby 和 Flash 数据交换的新方案,利用 Ruby 调用 ActiveX 控件方法,通过 XML 文件作为中间数据交换格式,并通过 Ruby 的 XML DOM 解析器 REXML 对页面 XML 文件进行格式解析,得到 Ruby 控制平台响应 Flash 展示平台的数据信息和控制信息,然后调用 Flash 展示平台的接口指令,最终实现页面的轮流动态加载展示。应用表明,自动售货机用户界面管理系统成功的实现了 Ruby 和 Flash 异构数据平台间数据交换,具有较好的实用价值和应用前景。

## 参考文献

1 张峰. VB 与 Flash 集成开发多媒体应用程序. 计算机应用研究, 2003,20(3):103 – 104,126.

2 Ruby Home Page . http://www.ruby – lang.org/

3 吴坤,莫国良,彭维,叶修梓. 基于 XML 的用户界面管理系统. 计算机应用研究, 2006,23(2):135 – 137.

4 黄红明,尹志兵,熊桂喜. 基于 XML 的数据交换技术研究及其在大型系统中的应用. 计算机应用研究, 2003,20(12):139 – 142.

5 贺智明,彭桃发. 基于 ActiveX 数据控件的 Web 信息系统平台设计. 计算机应用与软件, 2007,24(81):36 – 137.