

# 基于 Flex 和 Jbpm 的自定义 workflow 研究与实现<sup>①</sup>

王超, 戴牡红

(湖南大学 信息科学与工程学院, 长沙 410082)

**摘要:** 针对当前企业业务流程的个性化需要和业务需求多变的问题, 研究了基于 flex 和 JBPM 实现自定义 workflow 来解决上述问题. 采用 flex 技术和设计模式实现了 flex gef workflow 框架, 该框架实现了流程模型到 xml 数据的转换, 通过动态路由实现了 workflow 多步回退. 最后 Jbpm workflow 引擎载入符合 jpdL 规则的流程定义文件生成 workflow 实例. 经运行后表明, 该系统可实现用户自定义 workflow 且可实现多步回退的功能.

**关键词:** Flex; 自定义 workflow; Jbpm; Gef 框架; workflow 回退

## Study and Design of Workflow Based on Flex and Jbpm

WANG Chao, DAI Mu-Hong

(College of Information Science and Engineering, Hunan University, Changsha 410082, China)

**Abstract:** Aiming at the problem of personalized needs and changing needs in enterprise business processes, this paper researches the method of realize custom workflow based on the Flex and Jbpm to solve these problems. The Flex technology and design patterns is applied to Flex gef workflow frame. The framework realized conversion of the process model to xml data. Via dynamic routing to achieve a multi-step workflow rollback. Finally, the workflow instance is generated and run by load line with jpdL files into Jbpm workflow engine, after running shows that the system can achieve user custom workflow and multi-step rollback functionality.

**Key words:** Flex; user-defined workflow; Jbpm; gef framework; workflow rollback

近年来, 在激烈的市场竞争下越来越多的企业为了提高办公效率和企业核心竞争力而对企业信息化日益重视, workflow 作为一种过程建模和过程管理的技术也得到了进一步应用<sup>[1]</sup>. 目前企业 workflow 开发主要有两种方式: 一种是基于某种 workflow 引擎进行开发, 例如 jboss 的 Jbpm workflow, 开发人员利用 eclipse Jbpm 流程设计器来设计流程, 同时根据业务需求编写表单页面并利用配置文件来关联各个处理节点对应的表单<sup>[2]</sup>. 这种方式对流程设计和相关表单采用了硬编码, 当业务需求变更时需由开发人员修改相关流程、表单和业务逻辑处理, 这样做缺乏灵活性, 对开发人员和用户都带来了不便; 另一种是以 html 表单形式来实现, 这种方式实现的工作流的优点是简单易用, 并且可跨平台运行, 同时支持用户自定义表单, 用户友好性较高. 但是这种方式也有一些缺点: 繁琐的页面脚本代码用以支持自定义

功能, 不易进行后续的二次开发和维护. Html 表单只能对表单数据做一些简单的处理, 若实际应用中碰到复杂流程和业务时不能很好的对其进行处理. 本文借鉴了两种方式的优点, 对现有的基于 Flex 和 Jbpm 的自定义 workflow 方案进行了可行的技术实现: 利用 Flex 技术来实现 web 端的表单和流程设计器, 生成符合 jpdL 规则的 xml 文件. Jbpm 引擎载入该文件并加载运行流程. Flex 作为一种客户端技术界面展现十分强大, 用户友好性强, 为用户自定义表单等操作提供了良好的体验<sup>[3,4]</sup>; 借助 flashplayer 的安装普及度, 轻松实现跨浏览器跨平台, 相比 html 表单省去了繁琐的 js 脚本编写, 同时将数据展现和数据处理很好的进行分离; 后台流程业务处理采用 Jbpm 开源 workflow 来驱动. Jbpm 有优秀的工作流引擎架构设计和良好的可扩展性. 综上所述, 本文最终选择基于 Flex 和 Jbpm 技术实现一套自定义 workflow 系统.

<sup>①</sup> 收稿时间:2013-01-23;收到修改稿时间:2013-03-05

## 1 系统模型

整个系统包含三大部分: Flex gef 框架为流程定义及表单定义进行建模、工作流引擎载入并加载运行流程、数据访问层持久化, 系统总体框图如图 1 所示。

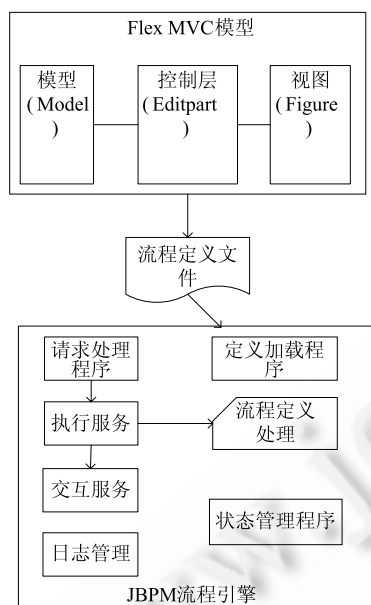


图 1 系统总体框图

Flex gef 框架为流程定义及表单定义进行建模: 以 Flex gef 框架为基础开发可视化图形界面使得用户可以在页面上自定义流程及任务表单, 最终生成符合 ipdl 规则的数据模型.

工作流引擎载入并加载运行流程: 工作流引擎载入 xml 格式流程定义文件并运行流程, 表单引擎捕获客户端提交的请求, 并将客户端提交的表单定义模型解析并持久化。

数据访问层持久化: 流程定义模型数据持久化都建立在公共的数据访问层上, 数据层还包括流程执行日志信息,

## 2 各模块的设计与实现

### 2.1 Flex gef 模型为流程定义建模

该模型采用 MVC 模式进行设计, 如图 2 所示. 整个模型由三部分组成: 流程模型、模型控制器、页面展现.

### 2.1.1 流程模型

流程模型对应的流程定义模型, 包括流程、节点、连接线等对象, 如图 2 所示, ProcessModel(流程定义模

型)、ConnModel(节点定义模型)、NodeModel(连接线定义模型)全部都实现了Imodel接口. 为了实现模型与xml数据之间的互相转换, Imodel接口提供了2个重要的方法如下:

## Public interface

```
IModel{
```

```
function Xml parseToXML():XML
```

```
function unParseXML(xml:XML):void
```

}

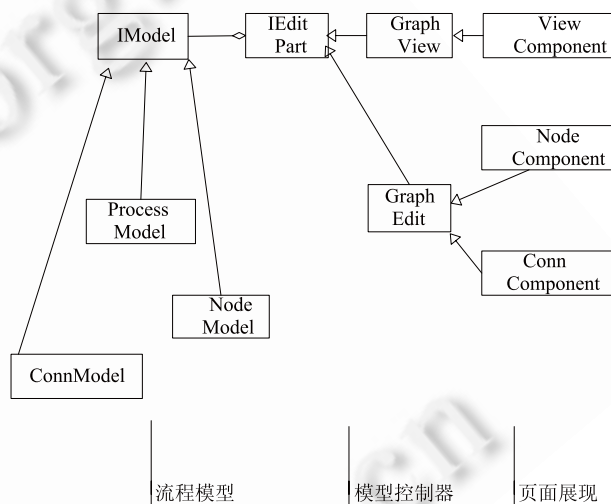


图 2 Flex gef 架构图

### 2.1.2 模型控制器

模型控制器的作用是及时对用户界面的操作进行响应并更新对应的流程模型。IEditPart 代表单个的图形编辑元素,该接口提供获取流程模型的方法:

```
function get model():Object;
```

```
function set model( _model:Object):void;
```

而代表页面展现的 **Component** 组件继承了 **IEditPart**，这样代表模型控制器的 **EditPart** 就将流程模型(**Model**)与页面展现(**Component**)关联了起来。这里实现的一个难点是对重做(**redo**)、回退(**undo**)操作的处理。应用命令模式可以很好的解决这一问题，**Component** 调用 **CommandService** 执行各个 **Command** 具体操作 **Model** 对象，以实现模型与页面展现的实时对应。此外，**CommandService** 持有 **CommandStack**，当编辑器中每次有新的操作请求时会产生一个请求对象按顺序进入堆栈 **CommandStack** 中，当用户需要执行重做或回退的操作时这些对象按后进先出的顺序依次出栈来响应。

### 2.1.3 页面展现

页面展现负责对流程元素的展现,包括:画图板(ViewComponent)、各种节点元素(NodeComponent)和连接线元素(ConnComponent)。主要包含流程中各个组件的增删改查,及每个组件属性的设置。用户可以在导航栏中选择需要的组件以拖拽的方式进行流程设计,双击某个组件时弹出该组件属性设置页面,用户可以进行属性的设置。

总的来说,当用户通过界面进行流程设计时,模型控制器捕获到用户请求并实时的更新流程模型,这样做解除了流程模型与页面展现的依赖,弱化了用户操作与模型变化的耦合,增强了应用的灵活性。

## 2.2 工作流引擎载入并加载运行流程

### 2.2.1 流程引擎调度

JBPM(Java Business Process Management)是基于 Java 实现的业务流程框架(Business Process Management)。它是包含了业务管理、工作流、服务等领域的一个开源的、易用的、易拓展的流程语言框架<sup>[5]</sup>。JBPM 采用了 JBoss JBPM Process Definition Language(JPDL)<sup>[6]</sup>来支撑业务流程,JPDL 结合了 PetriNet 算法、UML2.0 活动图、状态机三方面的知识,利用状态机控制了工作流状态变迁,借鉴 PetriNet 算法<sup>[7]</sup>,提高了活动图的建模能力。当执行业务流程时,工作流引擎会依据流程定义文件生成流程实例,这个实例拥有一个指向开始节点的根令牌,流程向下推进本质上就是令牌在各个节点上的转移,有两种方法可以向下一个节点转移令牌,方法一是流程实例调用 signal 方法,方法二是调用当前任务实例的 end 方法,本质上,上述两种方法都是调用当前令牌的 signal 方法来实现的。节点与节点间以一个转换对象为桥梁,所以,在转移过程中,会将令牌放入到相关联的一个转换对象中,在由这个对象将令牌交给下一个节点。整个过程如图 3 所示。

### 2.2.2 动态路由的实现

Jbpm 工作流是国外的一个开源项目,因此并不能完全满足国内的项目需求,比如任意回退、会签(包括加签、减签、补签)、撤销等。本文通过动态创建 transition 接口来解决这些问题。对于不存在连接的两个节点,若要实现这两个节点之间的跳转,首先,创建一个动态连接,然后再实现任务跳转,跳转完成后在将该连接删除,这样,保证了下次进入该任务时连接不会越来越多增加流程的复杂度。以下是创建、删

除连接的核心代码:

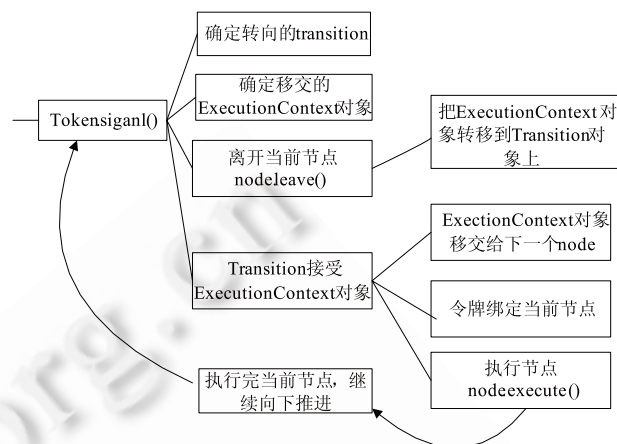


图 3 流程引擎调度图

```

/**动态创建连接当前任务节点到目的节点的
*transition
*param taskId 任务节点 ID
*param sourceName 源节点名称
*param destName 目的节点名称
*/
Public void addTransition(ProcessDefinitionImpl
pd,String sourceName,String destName){
    EnvironmentFactoryenvironmentFactory=(Environ
mentFactory) processEngine;
    Environment env=null;
    env = environmentFactory.openEnvironment();
    //取得当前流程定义
    ActivityImplsActivity=pd.findActivity(sourceName);
    //取得目的点的活动定义
    ActivityImpl dActivity=pd.findActivity(destName);
    //为两个节点创建连接
    TruansitionImpltransition=sActivity.createTransiton;
    tr ansition.setName(destName);
    transition.setDestination(dActivity);
    sActivity.addOutTransition(transition);
}
//态删除 sourceName 与 destName 间的 transition
//若存在这个连接,则需要把该连接删除
List<Transition>ts=sActivity.getOutgoingTransitions();
For(Transition trans:ts){
    if(destName.equals(trans.getDestination().getName(

```

```
))){//删除该连接
    ts.remove(trans);
    break;
}
}
```

2.3 数据访问层持久化

在工作流管理系统中，一个流程定义具有三种表现形态: xml 数据、java 对象以及工作流数据库中的记录<sup>[8]</sup>。前面的讨论中，系统根据用户定义的工作流图生成对应的 xml 文件，然后在 java 程序中转化为 Java 对象以进行业务逻辑处理，现在要做的是在 Java 对象与 jbpm 数据库建立转换，这里采用 Hibernate 来实现工作流数据层的持久化工作。jbpmConfiguration 维护了一系列实例工厂(ServiceFactory)，在配置文件中对实例工厂进行配置，它们只是在使用时才被创建，DbPersistenceServiceFactory 是其中的一个，它负责管理 SessionFactory。并且是在客户第一次请求时创建并调用 JbpmConfiguration.createJbpmContext.Ntext()方法，创建一个上下文(jbpmContext)，它管理数据持久化服务 DbPersistenceService，这个服务管理一个会话且所有数据访问都是通过该会话，而其他如计划会话，日志会话等都通过对应的会话类来完成。

3 实例及验证

为验证系统的功能性，使用该系统建立如图 4 的流程图，其中，新建申请节点后的分支为 and 分支类型，总裁审核为 and 型合并活动，其余的分支均为 or 类型分支/合并。

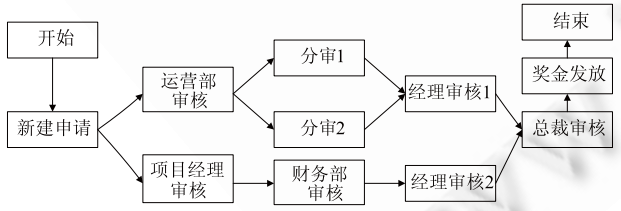


图 4 流程图实例

流程图建立后，系统生成对应的符合 Jpdl 规则的 xml 文件，如下：

```
<?xml version="1.0" encoding="GBK"?>
<process-definition xmlns="" name="分支流程">
    <start-state name="开始">
        <transition to="新建申请"></transition>
    </start-state>
```

```
<task-node name="新建申请">
    <transition to="运营部审核"></transition>
    <transition to="项目经理审核"></transition>
</task-node>
<task-node name="运营部审核">
    <transition to="分审 1"></transition>
    <transition to="分审 2"></transition>
</task-node>
<task-node name="项目经理审核">
    <transition to="财务部审核"></transition>
</task-node>
<task-node name="分审 1">
    <transition to="经理审核 1"></transition>
</task-node>
<task-node name="分审 2">
    <transition to="经理审核 1"></transition>
</task-node>
<task-node name="财务部审核">
    <transition to="经理审核 2"></transition>
</task-node>
<task-node name="经理审核 1">
    <transition to="总裁审核"></transition>
</task-node>
<task-node name="奖金发放">
    <transition to="结束"></transition>
</task-node>
<end-state name="结束"></end-state>
</process-definition>
```

系统将该流程定义文件加载到流程引擎中，引擎负责实例化流程并运行，执行回退操作来测试，获取如表 1 所示的当前活动节点的所有回退路径集合。

表 1 回退路径集合

| 当前活动   | 可回退目标集合                   |
|--------|---------------------------|
| 运营部审核  | {新建申请}                    |
| 项目经理审核 | {新建申请}                    |
| 分审 1   | {运营部审核, 新建申请}             |
| 分审 2   | {运营部审核, 新建申请}             |
| 财务部审核  | {项目经理审核, 新建申请}            |
| 经理审核 1 | {分审 1, 分审 2, 运营部审核, 新建申请} |
| 经理审核 2 | {财务部审核, 项目经理审核, 新建申请}     |
| 总裁审核   | {经理审核 1, 新建申请}            |
| 奖金发放   | {总裁审核, 新建申请}              |

从图中任一活动节点进行回退操作, 流程的节点状态及数据也跟着回到之前, 正确实现了流程回退. 如图中运营部审核节点进行回退至新建申请目标节点, 此时处于 and 分支的项目经理审核节点将被关闭, 目标节点与当前节点的状态的改变都被记录下来, 回退成功, 重新执行新建申请节点, 运营部审核、项目经理审核节点都被重新执行, 这也证明了之前流程的回退时正确的. 其余活动节点的回退操作也都实现了正确回退.

#### 4 结语

本文针对现有工作流对用户自定义支持的不足, 基于 flex 技术和 jbpm 开源工作流设计并实现了一套自定义工作流模型. 一方面, 利用 flex gef 框架, flex 技术为用户定义工作流的图形化设计和编辑提供了良好的体验, gef 框架实现了程序的松耦合, 提高了灵活性及可扩展性; 另一方面, 通过动态路由解决了流程多跨步回退和分支回退的问题. 此系统目前应用于某电信运营商的运营门户中, 高效的实现业务工作流程的运行. 今后的工作将在现有的系统模型上加以改进, 进

一步降低耦合, 完善系统架构.

#### 参考文献

- 1 兰天, 曲鹏东, 孙高飞, 等. Flex 企业应用开发实践. 北京: 机械工业出版社, 2010. 48-52.
- 2 刘晓菲, 宋朝晖. 浅谈基于 Flex 技术的 RIA 设计. 电脑知识与技术, 2010(11): 125-128.
- 3 江虹莹. 基于 JBPM 审批流程引擎的研究与实现[学位论文]. 成都: 电子科技大学, 2008.
- 4 王瑞玲. RIA 及其基于 Flex 的应用开发模型. 科技情报开发与经济, 2007, 17(14): 206-207.
- 5 WfMC. Workflow Management Coalition Technology and Glossary (WfMC-TC-1011). Technical Report Workflow Management Coalition, Brussels, 1996.
- 6 彭晓川. 基于 Flex 的 RIA 与 J2EE 应用的整合. 电脑与电信, 2008, 15(2): 37-39.
- 7 张伟滨. 自定义工作流的功能分析. 信息科技, 2007, 2(2): 149-150.
- 8 刘利坤. 基于 JBPM 和轻量级 J2EE 的办公自动化系统的研究与实现[学位论文]. 长春: 东北师范大学, 2009.

(上接第 78 页)

begin(), drawImage(), end() 函数来实现绘图事件. 主线程通过接受子线程的 senddata 信号来绘制图形以及调用 update() 函数来重绘, 从而来实现监控视频的实时显示.

#### 4 结语

本文采用嵌入式方法开发的无线视频监控系统, 已在学校的实验室监控系统中使用, 视频的传输速度在 11Mbit/s-54Mbit/s 之间, 达到了实时传输视频的需求, 取得了不错的使用效果. 该系统由于开发的设备成本低, 性能稳定, 并解决了传统的有线视频监控系统的问題, 将具有广泛的应用前景.

#### 参考文献

- 1 朱长坡. 浅析网络监控应用发展. 数字技术与应用, 2010(6): 172.
- 2 李波, 卢文科, 朱文超. 基于 3G 和 H.264 的无线视频监控系统的設計. 微计算机信息, 2011, 27(5): 78-80.
- 3 李建国, 罗代升, 吴小强. Blob 在 Intel PXA270 上的设计与移植. 成都信息工程学院学报, 2008, 23(2): 132-136.
- 4 潘学文, 文汉云. 基于 PXA270 的嵌入式 Linux 系统移植构建. 微处理机, 2011, 33(5): 89-91.
- 5 孔群娥, 曾学文, 刘马飞. 基于 x264 实现 H.264 的时域可伸缩编码. 微计算机应用, 2011, 32(5): 24-28.
- 6 丁林松, 黄丽琴. Qt 4 图形设计与嵌入式开发. 北京: 人民邮电出版社, 2003.