基于 JBPM 流程设计器①

董崇杰

(东莞职业技术学院 计算机工程系, 东莞 523808)

摘 要: 针对目前开发工作中需要把业务办理流程设计成一个个配置页面, 配置的页面不直观也很繁琐等问题. 本文在开源工作流 JBPM 的基础上进行二次开发, 采用 eclipse SWT+GEF(图形编辑框架)技术开发流程设计器, 它提供了流程图的可视化设计功能, 通过拖放及对各相关属性的编辑完成流程图的设计. 并且采用 JWS(Java web start)技术将该 eclipse plugins 改造成 RCP 应用以脱离 eclipse 环境运行.

关键词: JBPM; 流程设计器; GEF; RCP

Process Designer Based on JBPM

DONG Chong-Jie

(Department of Computer Engineering, Dongguan Polytechnic, Dongguan 523808, China)

Abstract: In view of the need for the development of business processes designed to handle each configuration page, configuration page is not intuitive and very cumbersome and so on. In this paper, the open source JBPM workflow based on the secondary development, using eclipse SWT and GEF (Graphical Editing Framework) technology to develop process designer, it provides a flow chart of the visual design features, through drag-and-drop and edit the relevant properties to complete the design of flow chart. Using of JWS (Java web start) technology transform the eclipse plugins into RCP applications running from the eclipse environment.

Key words: JBPM; process designer; GEF; RCP

1 引言

随着计算机技术和信息技术的进步和发展, 电子政务得到迅速的发展, 电子政务[1]有效解决了当前政务工作中存在的工作任务繁重和办事效率低等问题, 满足当前实际的需要. 电子政务目前还处于发展初期, 但是电子政务的优越性非常的明显, 极大地提高了工作效率.

电子政务的不断发展极大的促进了工作流技术在 OA 系统开发中的广泛应用, 特别是开源工作流技术在审批业务系统中的应用, 开发工作中需要把业务办理流程设计成一个个配置页面, 配置的页面不直观也很繁琐等问题, 为更好的有利于开发人员高效的完成业务办理流程的设计与工作流管理信息系统的发布等工作, 提高工作效率. 本文设计了基于 JBPM 的流程设计器, 该流程设计器应该具有以下的特点: 图形化的方式显示流程; 支持拖拽创建和修改流程; 导出图形对应的 xml 描述文件; 根据流程 xml 描述文件显示流程图.

2 相关技术介绍

2.1 JBPM

JBPM[2](Java Business Process Management1 是一种基于 J2EE 的轻量级的、灵活可扩展的工作流管理系统. JBPM 为设计及开发工作流和业务流程管理系统提供了一个先进的平台. JBPM 实现了流程逻辑与业务逻辑的分离. 能够可视化的进行业务流程的分析、定义和业务单元的组装, 从而使应用开发人员更关注于业务逻辑的实现. 降低了复杂流程应用的开发难度. JBPM 主要由工作流引擎、流程设计器和流程监控工具三部分组成, 主要组件关系图如图 1 所示[3]:

① 收稿时间:2013-03-20;收到修改稿时间:2013-05-02

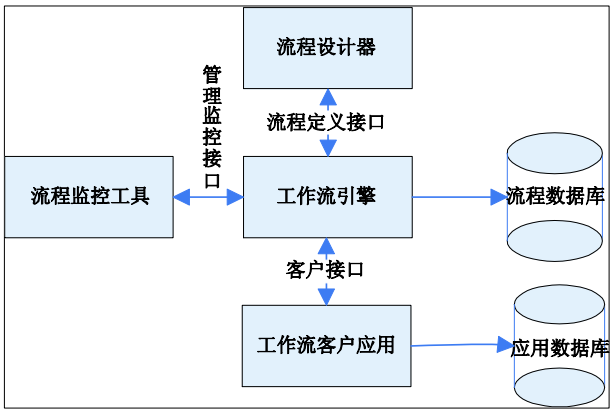

图 1　JBPM 主要组件关系图

工作流引擎完成对运行时流程的控制功能，它对外提供了流程定义接口、管理监控接口、客户接口. 应用系统可以通过这些接口同工作流引擎进行交互，流程设计器通过调用引擎的流程定义接口完成流程定义的功能，并把定义数据保存到数据库中，流程监控工具通过调用工作流引擎的接口对运行中的流程进行监控和管理.

2.2 GEF

GEF(Graphical Editing Framework)是图形化编辑器开发的工具，比较典型的应用就是IBM 的Rose，它是一个模型驱动的 MVC 框架，控制器(EditPart)作为模型的侦听器，侦听模型的变化，如果模型的属性发生变化，它会通知控制器，控制器就会刷新模型对应的视图(Figure). 可以看出模型和视图之间没有直接联系，它们通过控制器而间接联系，可见控制器在 GEF 框架中有着很重要的重要.

2.3 Spring

Spring 框架体系结构如图 2 所示. Spring core 包是整个框架的基础，它提供了基于依赖注入技术的构件组装机制; Spring DAO 包提供了一系列的数据库访问控制工具，免去了繁琐的数据库访问控制和异常处理的工作. Spring ORM 包允许程序设计人员将流行的 Hibernate、JDO、iBatis 等对象关系映射工具集成到现有系统中. Spring DAO 包实现了面向方面编程的支持，可以为构件提供统一的事务、日志、安全管理等服务. Spring Web MVC 包提供了对基于模型—视图—控制器模式的 Web 应用程序开发的支持[4].

2.4 RCP

RCP[5]是 Rich Client Platform 的缩写. RCP 是基于 Eclipse 项目推出的一个开发富客户端应用框架，目的在于为开发人员提供一个功能强大的、快速的、可扩展的应用平台. RCP 本质上是 Eclipse 的插件，所以当开发 RCP 应用程序时，可以利用 Eclipse 平台 UI 外观和框架来快速地进行开发. 例如创建一个菜单栏、工具栏，在 RCP 开发中很容易，只需要作一定的配置后，编写简单的代码就可以实现复杂的功能，这样就避免了许多重复性的工作.

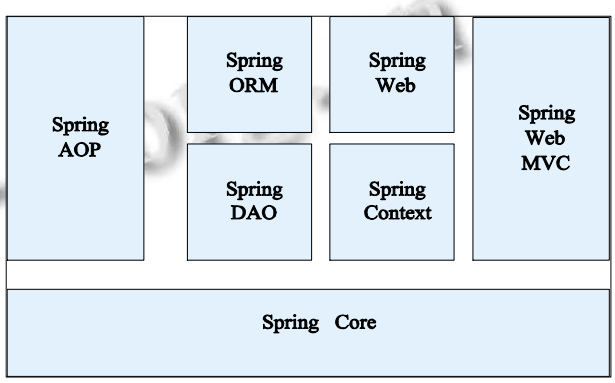

图 2　Spring 框架体系结构

RCP 的系统可以脱离 Eclipse 平台独立运行，这样大大减少了打包程序后文件的体积，使系统更加小巧和雅观. Eclipse RCP 结构关系图如图 3 所示: Eclipse RCP 具有以下优点[6]:

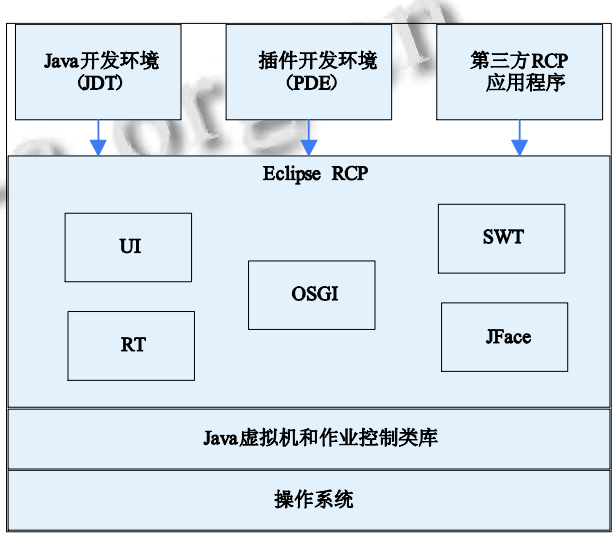

图 3　RCP 体系结构图

(1) 组件化. 基于 Eclipse 的系统设计由被称为 plug-ins 的插件构成，可以通过扩展点进行配置，也可以被不同应用程序共享.

(2) 可扩展性. Eclipse 基于插件进行扩展的思想使得用户可以方便地搭建各种规模、类型和用途的应用程序. 按照 Eclipse 官方的说法, Eclipse RCP 一开始就被设计为可扩展的.

(3) 本地观感及使用体验. Eclipse 为各种操作系统提供了本地图形接口包. 当 RCP 运行时, Eclipse 首先直接调用本机窗口组件, 只有没有本机所需组件时才进行模拟. 无论 RCP 在哪种操作系统上运行, 都可以保持与本机一致的外观和行为. 一个设计优良的富客户端, 可以提供诸如拖曳操作、剪切板、导航等 UI 元素. UI 设计者也可以利用各种界面工具, 轻松设计出完美的用户界面.

(4) 脱机操作. 由于 RCP 在本机运行, 不需要网络连接, 可以充分利用本机硬件的处理能力高速进行大量数据的处理.

3 设计与实现

3.1 设计思路及概念

(1) 技术路线

流程设计器是在 JBoss 社区的开源 eclipse plugin 的基础上进行二次开发, 主要采用 eclipse SWT+GEF(图形编辑框架)技术开发. 它提供了流程图的可视化设计功能, 通过拖放及对各相关属性的编辑完成流程图的设计. 并且将该 eclipse plugins 改造成 RCP 应用以脱离 eclipse 环境运行.

(2) 名词概念

流程设计器是在 JBoss 社区的开源 eclipse plugin 的基础上进行二次开发, 主要采用 eclipse SWT+GEF (图形编辑框架)技术开发. 它提供了流程图的可视化设计功能, 通过拖放及对各相关属性的编辑完成流程图的设计. 并且将该 eclipse plugins 改造成 RCP 应用以脱离 eclipse 环境运行 .

① 扩展属性

含义: 就是扩展对象,包括流程扩展对象、节点扩展对象的列表,其中节点扩展属性包括与业务系统相关的相关文书、关联操作、可编辑区域、状态变更等扩展属性.

② 状态变更

含义: 状态变更就是指节点从某个状态变为另一个个状态, 该状态是跟某个业务表中的某个字段相对应的(即和某个状态集 id 相对应)这里用掩码表示节点的状态.

比如某个节点的状态为'000'变更为'111'其中第一位表示是否预处理, 第二位为是否上报, 第三位表示是否审批, 则根据掩码表示该节点的状态从未预处理未上报未审批变更为已预处理已上报已审批 .

3.2 详细设计

设计图总体上分成三块, 一是流程设计器部分, 一是各业务系统相关部分, 还有是服务端有关工作工作流部分, 流程设计和服务端连接是通过 GpdService 接口,服务端 GpdService 的实现类 GpdServiceImpl 通过 ProcessDefinitionService 将数据库 S_WF_PROCESS DEFINITION 表中记录通过 hibernate 映射成 Process Definition 对象再经过相关整合传回客户端. 流程设计器服务器端设计图, 如图 4 所示.

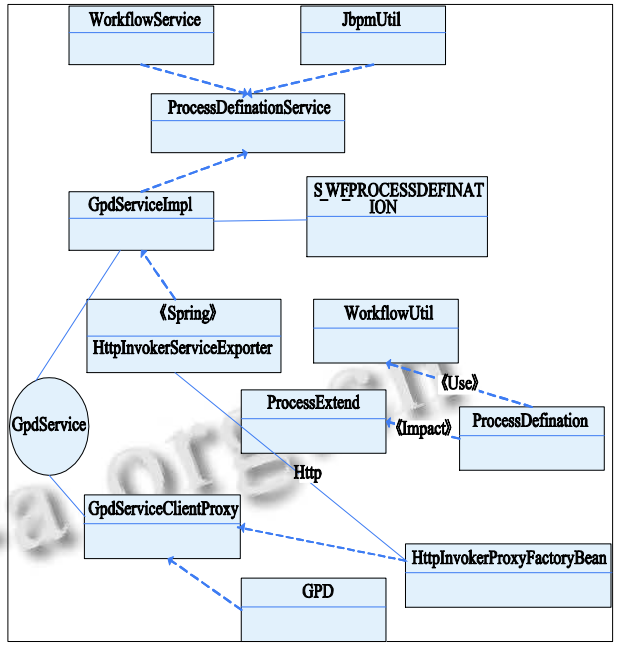

图 4 流程设计器服务器端设计图

各业务系统相关内容通过各业务系统相关接口暴露给客户端, 接口实现类从各业务系统相关元数据 xml 文件中取出, 组装成相关业务系统 java 模型对象传给客户端, 客户端拿到相关数据通过相应的工具类实现对象和 xml 的转换, 从而实现各业务系统模型对象和流程设计器扩展属性的相互转换. 这些体现在流程设计器上就是各业务系统编辑器界面元素的展示和展示完毕后将最新的界面元素的值转换为 xml 后塞回扩展属性对象. 以稽查业务系统为例进行说明, 如图 5

所示.

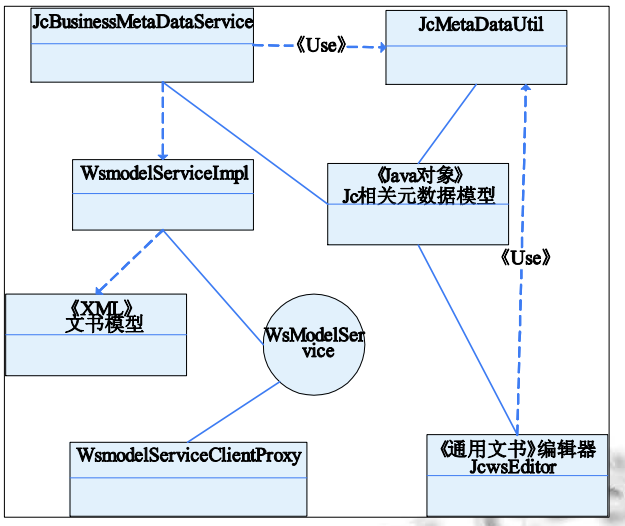

图 5 稽查业务系统设计图

流程扩展模式 XML 定义部分代码:

```
<nodeExtend name="start" isJoinSign="true">
        <desicionCodeSnippet></desicionCodeSnippet>
        <participants>
                <include>
                        <participant id="role1" name="角色1" type="role" />
                        <participant id="role2" name="角色2" type="role" />
                </include>
                <exclude>
                        <participant type="person" />
                </exclude>
        </participants>
        <extendProperties>
                <extendProperty name="稽查相关文书" type="jc_ws"><![CDATA[
                   another xml goes here ...
                ]]>
                </extendProperty>
                <extendProperty />
        </extendProperties>
</nodeExtend>
```

3.3 关键设计

(1) 对象和 xml 的相互转换

① 应用场景

主要应用在节点的扩展对象和扩展对象对应的流程扩展 xml 之间的转换, 以及扩展属性编辑器中, 各扩展属性和相应的扩展属性对应的为 xml 的一串值之间的转换.

② 设计思想

第一种应用场景的设计思想是每次打开流程图的时候从数据库元数据表(s_wf_processdefinition)表中 BUSINESS_MATA_DATA 大字段取出流程扩展属性, 通过工具类(WokflowUtil)将取出的字节数组转换为相应的对象, 再将对象中的数据映射到节点的扩展界面上; 第二种应用场景的设计思想基本类似, 是扩展属性的值转换为相应的业务系统模型对象, 再将对象的值映射到相应扩展属性编辑界面, 当界面编辑完毕又将最新的值赋给相应的对象, 最后转换为相应的 xml, 塞回扩展属性对象.

(2) 服务端和客户端通信

① 通信机制

流程设计和服务端通信采用 spring 远程服务, 通过 spring 的代理机制实现通信, 客户端通过代理类向服务端请求, 服务端根据请求的地址和配置文件中映射关系分发给相应的 bean 处理, 处理完成后再将处理后的结果返回给客户端.

② 具体实现

流程设计器通过 GpdService 和 Gzyj2GpdService 两个接口和服务端通信, 前者只是和流程设计器相关, 而后者涉及到各个业务系统的数据的获取, 当客户端调用两个接口中的方法时, 会根据 applicationContext-client.xml 配置文件中相应接口配置的 serviceUrl 向服务端请求调用该接口的实现, 服务端接收到请求后, 根据 applicationContext-httpinvoker.xml 中配置的 serviceUrl 和 bean 的映射关系, 调用 bean 所对应的接口实现获取数据传回客户端.

(3) 扩展点

① 优越性

通过扩展点实现了各业务系统编辑器的扩展性和插拔性, 从而达到流程设计器和各业务系统的松耦合.

② 实现机制

通过在 schema 文件夹中新增后缀为 exsd 的扩展点模式约束文件并在 plugin.xml 中定义相关的扩展点, 客户端获得相关的扩展点, 通过扩展点获得相应编辑器的类路径, 然后构造相应的编辑器.

扩展点在 pugin.xml 中的配置部分代码:

```
<extension-point id="nodeExtendProperties" name="Node ExtendProperty Editor Extensions"
schema="schema/nodeExtendProperties.exsd"/>
  <extension point="org.jbpm.gd.jpdl.nodeExtendProperties">
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.text"
       editingAdaptor="cn.com.hnisi.jbpm.gd.extend.extendproperty.TextExtendPropertyEditAdaptor"/>
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.boolean"
       editingAdaptor="cn.com.hnisi.jbpm.gd.extend.extendproperty.CheckboxExtendPropertyEditAdaptor"/>
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.multiline"
       editingAdaptor="cn.com.hnisi.jbpm.gd.extend.extendproperty.MultiLineExtendPropertyEditAdaptor"/>
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.WsExtendPropertyEdit"
       editingAdaptor="com.hnisi.gzyj.jbpm.gd.extend.extendproperty.WsExtendPropertyEditAdaptor"
       />
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.OpertionExtendPropertyEdit"
       editingAdaptor="com.hnisi.gzyj.jbpm.gd.extend.extendproperty.TransitionExtendPropertyEditAdaptor"
       />
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.EditAreaExtendPropertyEdit"
       editingAdaptor="com.hnisi.gzyj.jbpm.gd.extend.extendproperty.EditAreaExtendPropertyEditAdaptor"
       />
    <extendProperty
       id="org.jbpm.gd.jpdl.nodeExtendPropertyType.StateModifyExtendPropertyEdit"
       editingAdaptor="com.hnisi.gzyj.jbpm.gd.extend.extendproperty.StateModifyExtendPropertyEditAdaptor"
       />
  </extension>
```

4 结束语

本文结合 JBPM 工作流相关技术, 在 J2EE 开发环境下开发出具有可视化设计功能的流程设计器, 有利于开发人员高效完成具体业务办理流程的设计和工作流管理系统的发布, 提高了开发效率, 此流程设计器目前已成功应用于广州市药监局二期电子政务管理平台下的稽查、审评认证和行政审批三个子系统. 接下来的主要工作: ①进一步完善流程设计器对具体业务办理流程可视化界面的设计; ②对流程设计器中的相关组件进行二次开发, 进一步提高具体业务工作流系统的运行效率.

参考文献

1 唐协平,张鹏翥.电子政务需求研究综述.计算机应用研究, 2008,25(7):1921–1931.

2 Cumberlidge M. Business Process Management with Jboss JBPM. UK: Packt Publishing Ltd, 2007.

3 刘军,王广杰.基于 JBPM 和轻量级 J2EE 的协同办公系统的研究.福建电脑,2007,12:99–100.

4 Johnson R, Hoeller J, Arendsen A. Spring Java/J2EE application Framework. 2004.

5 强锋科技,那静.Eclipse SWT/JFace 核心应用(第 1 版).北京:清华大学出版社,2007:446–463.

6 陈冈.Eclipse RCP 应用系统开发方法与实战(第 1 版).北京:电子工业出版社,2007:20–26.

(上接第 13 页)

2011, 25(2): 99–104.

6 Veale T, Hao Y. Comprehending and generating apt metaphors: a web-driven, case-based approach to figurative language. Proc. of the 22nd National Conference on Artificial intelligence-Volume2. Palo Alto: AAAI Press, 2007: 1471–1476.

7 Strube M, Ponzetto SP. WikiRelate computing semantic relatedness using wikipedia. Proc. of the 21st National Conference on Artificial Intelligence-Volume 2. Palo Alto: AAAI Press, 2006: 1419–1424.

8 盛志超,陶晓鹏.基于维基百科的语义相似度计算方法.计算机工程,2011,37(7):193–195.

9 陆勇,章成志,侯汉清.基于百科资源的多策略中文同义词自动抽取研究.中国图书馆学报,2010,36(185):56–62.

10 苏畅.汉语名词性隐喻的计算方法研究[博士学位论文].厦门:厦门大学,2008.

11 王治敏.名词隐喻相似度及推理识别研究.中文信息学报,2008,22(3):37–43.

12 杨芸.汉语隐喻识别与解释计算模型研究[博士学位论文].厦门:厦门大学,2008.

13 孙茂松,黄昌宁,方捷.汉语搭配定量分析初探.中国语文,1997,1:29–38.

14 李玉莲.本体和喻体在中文隐喻句理解中的应用[硕士学位论文].重庆:西南大学,2007.

15 汪祥,贾焰,周斌,丁兆云,梁政.基于中文维基百科链接结构与分类体系的语义相关度计算.小型微型计算机系统,2011,32(11):2237–2242.