文章编号：1006-4710(2013)01-0020-07

# 一种轻型工作流引擎的设计与实现

杨明顺[1]，韩周鹏[1]，余婷[1]，李言[1]，邵利真[2]
(1. 西安理工大学 机械与精密仪器工程学院，陕西 西安 710048；
2. 中铁电化(西安)通号设备有限公司，陕西 西安 710000)

**摘要**：针对目前常见大型工作流引擎存在结构复杂、灵活性差等问题，采用分层存储思想，将工作流过程模型中节点的静态属性和节点之间的动态连接关系分开存储，优化了工作流过程模型文件的存储方式。在保存好的工作流过程模型的基础上，制定了简化的模型解析加载流程，根据节点之间的动态连接关系，提出了节点递归排序算法，保证了节点的有序运行。基于.NET 编程平台，开发了一种轻型工作流引擎，实现了工作流实例的解析加载和各活动节点的有序仿真运行，通过案例验证了所开发工作流引擎的正确性和有效性。

**关键词**：工作流引擎；分层存储；活动节点；模型加载；实例运行
**中图分类号**：TP311.11　　**文献标志码**：A

## Design and Implementation of a Lightweight Workflow Engine

YANG Mingshun[1], HAN Zhoupeng[1], YU Ting[1], LI Yan[1], SHAO Lizhen[2]
(1. Faculty of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an 710048, China;
2. China Railway Electrification (Xi'an) Communication & Signal Equipment Co. Ltd, Xi'an 710000, China)

**Abstract**: Aiming at the complex structure and poor flexibility of large workflow engine, the static attributes of nodes and the dynamic connection relations between nodes in workflow process model are stored separately to optimize the storage of model file using tired storage thinking. Based on the saved workflow process model, the simple model analytical and loading process is made. Furthermore, according to the dynamic connection relations between nodes, a node recursion sort algorithm is proposed to ensure the orderly operation of the nodes. On the basis of the .NET programming platform, a kind of lightweight workflow engine is designed and developed, whereby realizing the analytical loading of workflow instance and the orderly simulation running of activity nodes. Finally, the correctness and effectiveness of the developed workflow engine is tested via the practical cases.

**Key words**: workflow engine; tiered storage; activity nodes; model loading; instance running

工作流是一类能够完全或者部分自动执行的经营过程，它根据一系列过程规则、文档、信息或者任务能够在不同的执行者之间进行传递与执行[1-2]。工作流管理系统是工作流技术应用的载体，工作流引擎则是工作流管理系统的核心，其性能的优劣直接影响着工作流管理系统的执行效率[3]。轻量级工作流引擎具有简洁、灵活等特性并能被方便地集成到面向流程的应用，是当前研究的热点。

近年来国内外众多学者展开了针对工作流引擎的研究。文献[4]基于 Web 的工作流引擎，支持企业级应用；文献[5]研究了基于.NET 平台下实现图形化工作流引擎机制；文献[6]分析研究了 MINI 引擎的结构框架和各个功能模块的设计；文献[7]建立基于对象有色 Petri 网的工作流模型，并根据工作流的执行过程，进行了工作流引擎体系结构以及相关类的设计；文献[8]针对产品设计任务节点的工

**收稿日期**：2012-09-11
**基金项目**：国家自然科学基金资助项目(60903124)；高等学校博士学科点专项科研基金项目(20096118120003)；西安市科技计划项目(CX1255-4)。
**作者简介**：杨明顺，男，副教授，博士，主要研究方向为网络化设计与生产优化控制。
E-mail：Yangmingshun@xaut.edu.cn。

作流过程中出现的表单设计问题,提出了表单系统与工作流引擎集成的一种设计与实现方案;文献[9]在分析工作流引擎主要功能的基础上,以 XPDL 作为流程定义语言,采用 Java 技术,实现工作流引擎的基本功能和特征;文献[10]提出一种基于 BPEL 的工作流引擎调度技术;文献[11]从工作流的标准出发,结合对已有工作流产品的研究与分析,设计了一个适应 WFMC 标准的工作流引擎,并着重研究了工作流数据接口和工作流执行解释的相关问题。

目前,已有研究大都是基于工作流管理联盟(Workflow Management System,WFMC)给出的工作流过程定义语法 XPDL(XML Process Definition Language),在活动节点定义过程中,XPDL 规定了严格的语法约束,定义活动节点的 XML 文件不仅包含活动节点的各属性信息,而且还囊括了节点间连接关系信息,增加了活动节点定义的复杂度。由此带来的节点静态属性信息和节点间动态连接关系信息的紧密耦合,导致过程定义和过程实例化脱节。

因此,在注重模型敏捷性的中小型工作流管理系统中,需要研究更为灵活的工作流引擎实现技术,以轻量级和灵活性为目标,基于简单、松散耦合的节点定义语法,简化 XPDL 语法,解决现有工作流引擎结构复杂、流程脱节的问题,实现工作流过程的形式化建模和实例化仿真运行。本文基于静动态信息分层思想,简化原有工作流模型的存储结构,制定模型解析加载流程并设计节点排序算法,在仿真运行时通过工作流结构的逻辑判断决定节点运行的先后顺序,实现对传统企业应用级工作流引擎的简化,并通过实例仿真运行进行验证。

## 1 工作流引擎

工作流管理系统主要实现流程建立阶段时过程模型相关活动的定义和建模、流程运行阶段时活动的排序、调度以及人机交互。WFMC 给出的工作流管理系统体系结构模型包括过程建模工具、组织/角色/资源建模工具、工作流引擎、工作流控制数据和工作表处理程序等,其体系框架如图 1 所示。

工作流引擎是工作流管理系统的核心,其主要功能是在流程定义阶段提供规则支持,在流程运行阶段提供解析和执行服务。为了实现上述功能,工作流引擎必须包括过程模型定义、过程模型解析、模型实例化运行、工作流资源管理配置、工作项管理、工作流相关数据的维护、与外部的信息交互等。因此,一般工作流引擎主要包括解析加载模块、流程执行模块和工作项管理等模块。工作流引擎各模块如图 2 所示。

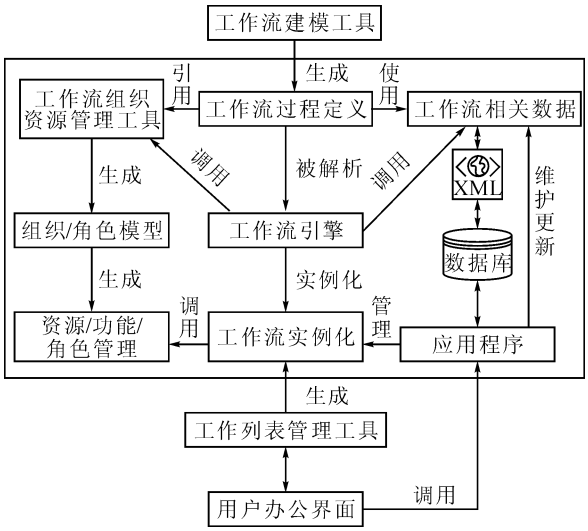


图 1 工作流管理系统架构图

Fig. 1 The structure of WFMS

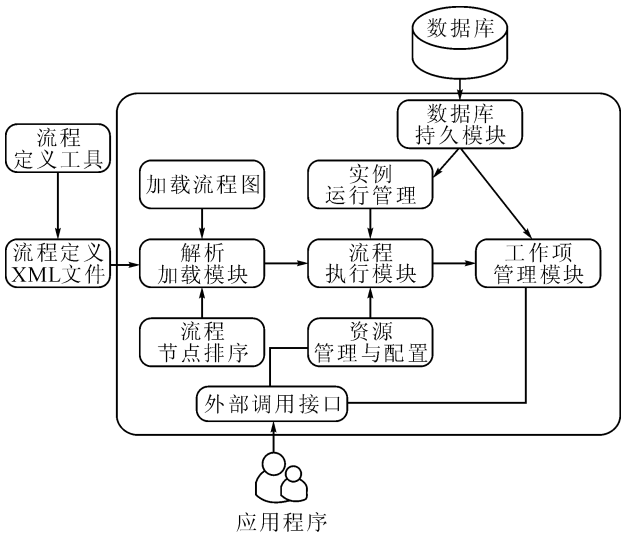


图 2 工作流引擎框架图

Fig. 2 Framework of workflow engine

工作流引擎解析加载模块首先需识别过程模型中产生的模型文件,然后按照预先定义好的解析规则,加载所定义的工作流程,并将流程图显示到客户端用户界面,进而依据节点之间的逻辑连接关系设计工作流节点排序算法,生成实例化节点执行顺序表,使实例各节点依次有序执行,完成流程的实例化仿真运行。

本文主要基于分层存储的思想,综合运用 XML 和数据库技术存储工作流过程模型,制定模型解析规则,确定模型加载步骤;同时设计节点排序算法,求解模型中所有节点的执行顺序,实现工作流引擎中实例解析加载模块和实例仿真运行模块的功能。

## 2 工作流模型解析与加载

工作流引擎解析加载模块解析加载的是事先存储完好的、结构正确的工作流过程模型文件。首先需要合理地持久化保存工作流过程模型,进而确定模型的解析规则和加载步骤。

### 2.1 工作流过程模型的分层存储

针对传统工作流建模语法 XPDL 中存在的活动节点定义复杂、耦合度高等问题,采用分层存储思想,将工作流过程模型中节点的静态属性信息保存到 XML 文件,将节点之间动态的连接关系信息保存到数据表中,实现二者的分离,降低过程模型的复杂度和耦合度。活动节点的属性主要包括节点的编号、名称、位置等静态信息,节点之间的逻辑连接关系分为五种,分别是顺序、或分、或合、与分和与合[12]。工作流过程模型的分层存储如表1所示。

表1 工作流过程模型分层存储

Tab.1 Workflow process model storage table

| 存储方式 | 字段名称 | 属性说明 |
|---|---|---|
| XML 文件格式(静态层) | ID(节点编号) | 节点编号 |
| | LCID(流程编号) | 确定节点的流程归属 |
| | Name(节点名称) | 节点的简易名称 |
| | Left(左位置) | 流程设计完成后节点与设计画布左侧的距离 |
| | Top(上位置) | 流程设计完成后节点与设计画布上部的距离 |
| | Width(节点宽度) | 节点本身的宽度 |
| | Height(节点高度) | 节点本身的高度 |
| | BusinessName(节点备注) | 节点详细备注信息 |
| 节点信息表(ActivityInfo)(动态层) | activityID(节点编号) | 节点编号 |
| | LCID(流程编号) | 确定节点的流程归属 |
| | activityName(节点名称) | 节点的简易名称 |
| | BusinessName(节点备注) | 节点详细备注信息 |
| 节点连接关系表(actRelation)(动态层) | LCID(流程编号) | 确定连接关系的流程归属 |
| | QDID(起点编号) | 连接的起始节点 |
| | ZDID(终点编号) | 连接的终止节点 |
| | GX(连接关系) | 连接关系(顺序、或分、或合、与分、与合) |
| | GL(概率) | 连接关系为或分情况下两分支的执行概率 |

采用分层思路的模型存储方法能够大大简化工作流过程模型的 XML 文件,改变了传统过程模型 XML 文件复杂冗余的现状,方便后期工作流过程模型实例化时模型的加载、运行、分析优化和资源配置;与数据库技术的结合也发挥了其数据易于存储和修改的优势,实现了模型的持久化。

### 2.2 工作流过程模型的解析加载

针对保存好的工作流过程模型,工作流引擎在解析加载过程中首先对模型的静态 XML 文件进行解析,提取各活动节点的编号、位置等数据,并将节点呈现到流程编辑器页面上展示;然后从模型对应数据表中查找节点间连接关系信息,通过节点操作类中相关方法生成各连接弧线,完成整个解析加载过程。工作流过程模型的解析加载步骤如下。

1)导入模型对应的 XML 文件,读取 XML 文件中各节点的编号、左位置和上位置属性。

2)把 XML 文件读取的节点属性信息作为参数传入节点操作类(ActivityManage)中的节点添加(ActivityAdd)方法中,在画布中依次按照 XML 文件中定义的节点位置加载各节点。

3)从节点信息表中查找每个要加载的节点的属性信息,赋给节点信息类(ActivityClass)。

4)从连接关系表中查找该流程所有的连接关系,读取起点编号、终点编号和连接关系属性,然后作为参数传入节点操作类(ActivityManage)中的连接关系添加(RelationAdd)方法中,执行该方法,根据事先定义好的关系连接线符号依次在画布上绘制各连接关系线。

## 3 工作流实例仿真运行

工作流实例的运行侧重于分析模型的运行时间、运行成本等特性,本文主要实现模型的仿真运行,暂不考虑节点的资源需求和各节点具体执行时间,因此,确定模型各活动节点的执行顺序成为一个首要问题。首先,设计工作流节点排序算法,生成节点排序结果集;其次,设定节点的统一运行时间,依据节点排序结果集,实现模型实例化过程中从开始节点至结束节点的各活动节点依次有序执行;最后,完成整个流程的仿真运行。流程节点排序算法步骤如下。

Step1 从 actRelation 中检索出流程所有节点对存入 RelationSet 中,把所有节点对的起点存入 StartSet 中,检索流程开始节点作为循环的 StartNode;

RelationSet = {(开始节点,$a_1$),($a_1$,$a_2$),($a_2$,$a_3$),…,($a_i$,$a_{i+1}$),…,($a_n$,结束节点)};

StartSet = {开始节点,$a_1$,$a_2$,…,$a_i$,…,$a_n$};

StartNode = 开始节点;

Step2 计算 StartNode 的出度(节点的后继节

点个数),判断是否为 1,如果是 1 则表示 StartNode 和 SubsequentNode 的连接关系是顺序、或合或与合,如果不为 1,则表示连接关系为与分或是或分;

Step3 如果 StartNode 的出度为 1,返回 StartNode 的 SubsequentNode 作为 EndNode,并存入到 ResultSet 中,Sequence 为当前循环序号,然后判断 EndNode 是否为流程结束节点,如果"是"则结束整个循环,否则将 EndNode 作为下一轮循环的 StartNode,继续执行 Step2;

Step4 如果 StartNode 的出度不为 1,返回 Start Node 的所有 SubsequentNode,存入 SubsequentSet,判断当前连接关系是与分还是或分;

Step5 如果是与分关系,分别计算两条与分支路的起点、终点和路径长度,然后选择路径较长的支路,将该支路的起点和当前循环序号存入 ResultSet,将该支路起点赋给 StartNode,同时把另一条支路的起点、终点和当前循环序号分别存入 SubsequentSet 和 ResultSet,将该支路起点赋给 StartNode,最后均返回执行 Step2;

Step6 如果是或分关系,从数据库中查找或分两支路的执行概率,按照概率选取一条支路,并将该支路的起点和当前循环序号存入 ResultSet,并将该支路起点赋给 StartNode,最后返回执行 Step2。

相关符号概念说明:

actRelation:节点连接关系数据库表;

RelationSet:节点关系集,二维数组,存放所有节点关系;

StartSet:起点集,一维数组,存放所有起点;

ID:节点编号;

Sequence:节点排序号,确定节点执行顺序;

ResultSet:排序结果集,二维数组,首列为节点 ID,次列为 Sequence;

StartNode:当前节点,每轮循环输入起始节点;

EndNode:当前终点,每轮循环输出终止节点;

SubsequentNode:后继节点;

SubsequentSet:后继节点集,动态数组,所有后继节点组成的集合。

按照上述步骤,节点排序算法流程如图 3 所示。

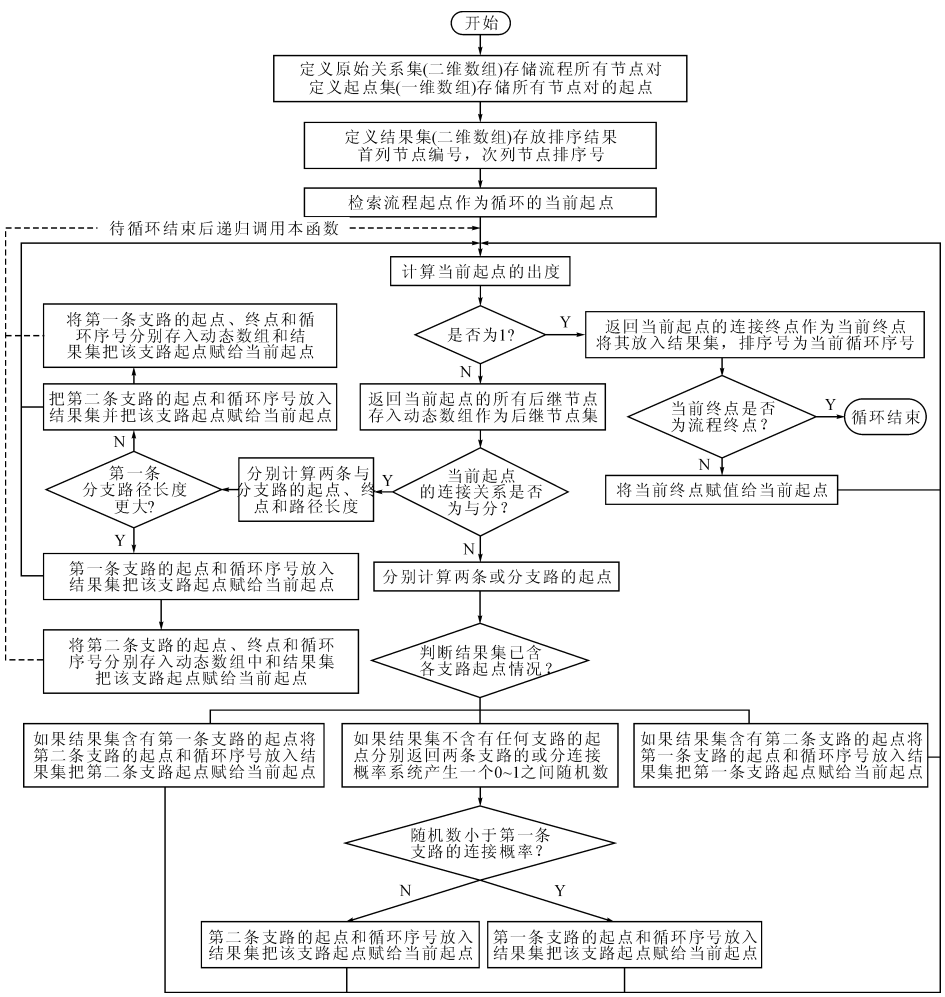


图 3 节点排序算法流程图

Fig. 3 The flow chart of example nodes sorting algorithm

该节点排序算法建立在过程模型分层存储结构基础上,采用递归的思想,通过为每个活动节点计算执行顺序号来确定节点间的执行顺序。算法支持选择、并行和循环结构的多重嵌套,具有普适性强、执行效率高等优点。

## 4 案 例

为了验证工作流引擎流程解析加载模块和实例仿真运行模块的有效性,设计了一个拥有20个活动节点,包含选择结构和循环结构嵌套、并行结构和选择嵌套结构的过程模型,如图4所示。

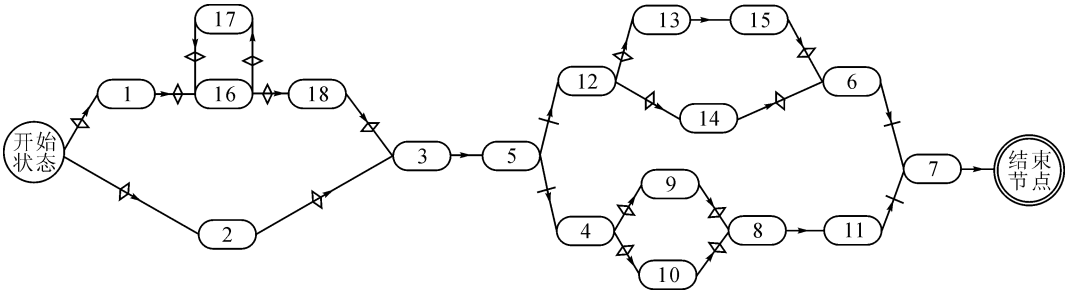


图4 案例工作流过程模型图

Fig. 4 The workflow process model of example

依托微软.NET开发平台,采用最新的WPF编程语言,在工作流建模系统平台上开发了工作流引擎,针对案例所设计的过程模型,从程序实现角度也测试了设计的工作流引擎流程解析加载和实例仿真运行模块的功能。

### 4.1 案例过程模型存储

按照分层存储的思想,把该过程模型分别保存到XML文件和数据表中。所保存的XML文件如下所示。

<? xml version = "1.0" encoding = "utf-8" ? >
< Root >
< DesignerItems >
< ActivityControl >
< Left > 84.06 </Left >
< Top > 206.09 </Top >
< Width > 72 </Width >
< Height > 52 </Height >
< LCID > 1.2 </LCID >
< ID > 1 </ID >
< Name > 提出申请 </Name >
< BusinessName > 向上级部门提出申请 </BusinessName >
</ActivityControl >
< ActivityControl >
< Left > 210.34 </Left >
< Top > 419.15 </Top >
< Width > 72 </Width >
< Height > 52 </Height >
< LCID > 1.2 </LCID >
< ID > 2 </ID >
< Name > 提交审核 </Name >
< BusinessName > 越过申请直接提交审核
</BusinessName >
</ActivityControl > ……
</DesignerItems >
</Root >

所保存的案例节点连接关系如表2所示。

表2 案例节点连接关系表

Tab. 2 The data tables of example node-relationships

| 起点编号 | 终点编号 | 连接关系 | 概率 | 起点编号 | 终点编号 | 连接关系 | 概率 |
|---|---|---|---|---|---|---|---|
| 开始节点 | 2 | 或分 | 0.3 | 5 | 4 | 与分 | – |
| 5 | 12 | 与分 | – | 6 | 7 | 与合 | – |
| 4 | 10 | 或分 | 0.6 | 11 | 7 | 与合 | – |
| 10 | 8 | 或合 | – | 16 | 18 | 或分 | 0.6 |
| 3 | 5 | 顺序 | – | 2 | 3 | 或合 | – |
| 12 | 14 | 或分 | 0.6 | 7 | 结束节点 | 顺序 | – |
| 4 | 9 | 或分 | 0.4 | 8 | 11 | 顺序 | – |
| 14 | 6 | 或合 | – | 开始节点 | 1 | 或分 | 0.7 |
| 12 | 13 | 或分 | 0.4 | 9 | 8 | 或合 | – |
| 15 | 6 | 或合 | – | 1 | 16 | 或合 | – |
| 17 | 16 | 或合 | – | 13 | 15 | 顺序 | – |
| 16 | 17 | 或分 | – | 18 | 3 | 或合 | – |

### 4.2 工作流过程模型解析与加载

按照前述的工作流过程模型的解析加载步骤加载案例的 XML 文件;同时调用后台数据表中案例数据,加载的案例过程模型如图 5 所示。

### 4.3 工作流实例仿真运行

调用前述的节点排序算法对所有节点进行排序,计算每个节点的排序号,得到表 3 所示的排序结果集。按照此结果集,根据排序号依次执行相应节点。调用节点操作类(ActivityManage)中的节点运行函数(ActivityRun),假定各节点执行时间相同,节点运行自动触发,即前驱节点执行完成之后,后续节点自动执行,实例流程仿真运行结果如图 6 所示。

表 3 案例节点排序结果集

Tab. 3 Results set of example node-sorting

| 节点编号 | 排序号 | 节点编号 | 排序号 | 节点编号 | 排序号 |
|---|---|---|---|---|---|
| 开始节点 | 0 | 3 | 6 | 15 | 10 |
| 1 | 1 | 5 | 7 | 8 | 10 |
| 16 | 2 | 12 | 8 | 6 | 11 |
| 17 | 3 | 4 | 8 | 11 | 11 |
| 16 | 4 | 13 | 9 | 7 | 12 |
| 18 | 5 | 10 | 9 | 结束节点 | 13 |

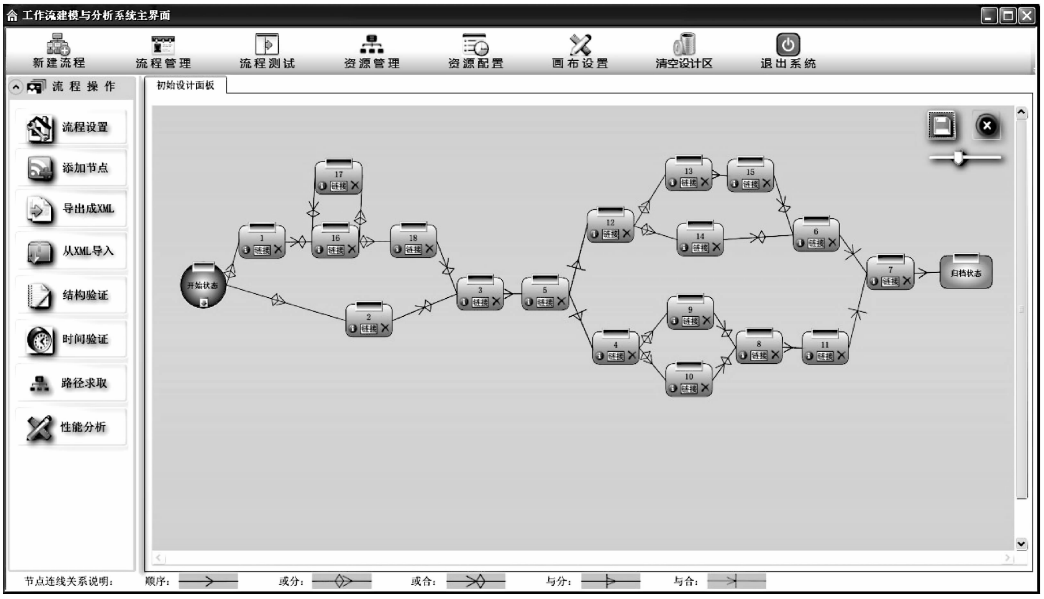


图 5 工作流过程模型加载流程图

Fig. 5 The flow chart of workflow process model loading

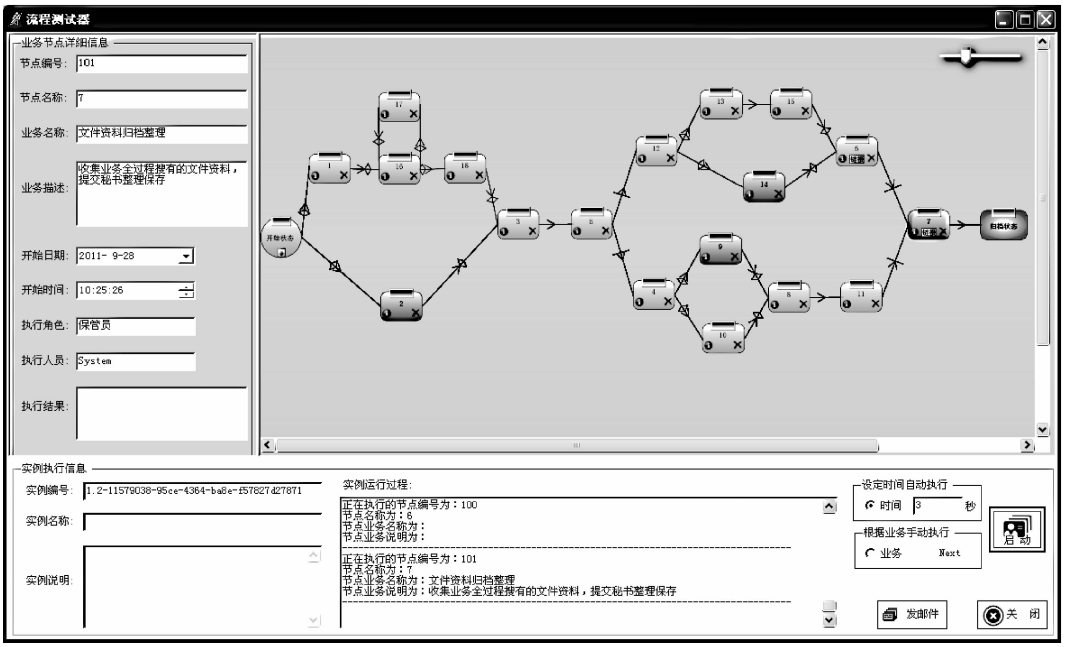


图 6 工作流实例仿真运行图

Fig. 6 Running result of workflow instance simulation

## 5 结 语

针对基于 XPDL 语法的工作流引擎结构复杂、灵活性差等问题,结合 XML 和数据库技术,提出了更为简单的模型分层存储方式,简化了节点定义语法,并在此基础上给出了实例解析加载步骤;同时,设计了节点递归排序算法,保证实例化过程中各活动节点的有序感知和仿真运行。最后设计开发出了一种轻型、灵活的工作流引擎,通过案例验证了工作流模型解析、加载以及节点有序仿真运行等模块功能的可用性和有效性。

应该指出的是,本文设计的工作流引擎,没有追求功能的完备和复杂,只是实现其中必不可少的功能和特征,目的在于为中小企业和科学研究提供一个简化的工作流运行平台。设计的引擎还存在一些不足,例如工作流实例化过程中没有考虑活动实际运行时间和资源配置情况等。后续研究需要引入资源和成本等因素,以工作流动态资源配置和优化为突破口,进一步完善该轻型工作流引擎的功能,不断增强其实用性。

**参考文献:**


[1] 范玉顺,吴澄. 工作流管理技术研究与产品现状及发展趋势[J]. 计算机集成制造系统,2000,6(1):1-7,13.
Fan Yushun, Wu Cheng. Current state and development trends of workflow management research and products [J]. Computer Integrated Manufacturing System, 2000,6(1):1-7,13.

[2] Van der Aalst WMP, Kumar A. A reference model for team-enabled workflow management systems [J]. Data & Knowledge Engineering, 2001, 38(3): 335-363.

[3] 李竞杰,王维平,杨峰. 柔性工作流理论方法综述[J]. 计算机集成制造系统,2010,16(8):1569-1577.
Li Jingjie, Wang Weiping, Yang Feng. Review on approaches of flexible workflow [J]. Computer Integrated Manufacturing System, 2010,16(8):1569-1577.

[4] 张洪山,殷人昆,张素琴. 基于 WEB 的工作流引擎设计[J]. 计算机工程,2004,30(4):83-85.
Zhang Hongshan, Yin Renkun, Zhang Suqin. Design of workflow engine based on WEB [J]. Computer Engineering, 2004,30(4):83-85.

[5] 刘跃华,方俊. 基于. NET 的工作流应用系统设计[J]. 计算技术与自动化, 2009,28(2): 115-118,137.
Liu Yuehua, Fang Jun. Design of workf low application system based on . NET [J]. Computing Technology and Automation, 2009,28(2): 115-118,137.

[6] 于可新,齐璇,施海虎,等. MINI 工作流管理系统引擎的设计与实现[J]. 计算机工程与科学,2005,27(5): 88-90.
Yu Kexin, Qi Xuan, Shi Haihu, et al. Design and implementation of the MINI workflow engine [J]. Computer Engineering & Science, 2005, 27(5): 88-90.

[7] 杨军,于永利. OCPN 装备综合保障工作流引擎设计[J]. 火力与指挥控制, 2011, 36(8): 133-136.
Yang Jun, Yu Yongli. Workflow modeling and engine design of equipment integrated logistic support based on OCPN [J]. Fire Control&Command Control, 2011, 36(8): 133-136.

[8] 巫少龙,方晓汾,董建峰. 表单系统与工作流引擎集成设计与实现[J]. 中国机械工程, 2012, 23(1): 46-50.
Wu Shaolong, Fang Xiaofen, Dong Jianfeng. Design and implemantation in integration of form system and engnine [J]. China Machinical Engineering, 2012, 23(1): 46-50.

[9] 蔡孝武,韩永国,蓝科. 一种轻量级工作流引擎的研究与设计[J]. 计算机工程,2010, 36(20): 78-82.
Cai Xiaowu, Han Yongguo, Lan Ke. Research and design of lightweight workflow engine [J]. Computer Engineering, 2010, 36(20): 78-82.

[10] 郭利军,张振明,耿俊浩. 基于 BPEL 的工作流引擎调度技术研究[J]. 中国制造业信息化,2011,40(9): 7-10.
Guo Lijun, Zhang Zhenming, Geng Junhao. Research on scheduling technology of workflow engine based on BPEL [J]. Manufacture Information Engineering of China, 2011, 40(9):7-10.

[11] 钱茅. 基于行为驱动开发的轻量级工作流引擎的设计与实现[D]. 武汉:华中师范大学,2010.
Qian Mao. The design and implementation of lightweight workflow engine based on behavior-driven development [D]. Wuhan: Central China Normal University. 2010.

[12] 高新勤,李宗斌. 基于统一建模语言和多色集合理论的工作流建模方法研究[J]. 计算机集成制造系统,2006,12(7):970-975.
Gao Xinqin, Li Zongbin. Workflow modeling method based on UML & polychromatic sets theory [J]. Computer Integrated Manufacturing System, 2006, 12(7): 970-975.


(责任编辑 王卫勋)