面向安全生产综合监管系统的工作流引擎①

滕　飞[1,2]　廉东本[1]　(1.中国科学院 研究生院 北京 100049;
2.中国科学院沈阳计算技术研究所 信息化工程技术部 辽宁 沈阳 110171)

摘　要： 针对安全生产监管系统中关键业务应用的开发离不开工作流技术支持的特点，通过对安全生产监管领域中关键业务的实际开发需求的分析，引进了工作流技术，设计并实现了一个轻量级嵌入式工作流引擎；本文首先对安全生产综合监管系统整体架构进行分析，然后提出该工作流引擎的设计方案，并详细阐述了引擎设计的关键环节；最后把此工作流引擎应用到安全生产综合监管系统中，不仅显著地缩短关键业务的开发周期，同时也规范了各种关键业务处理流程。

关键词： 工作流；工作流引擎；嵌入式；安全生产综合监管

Workflow Engine Designed for Consolidated Supervision of Safety Production System

TENG Fei[1,2]，LIAN Dong-Ben[1]
(1.Graduate University of Chinese Academy of Sciences, Beijing 100039, China; 2.Department of Information and Engineer, Shenyang Institute of Computing Technology, CAS, Shenyang 110171, China)

Abstract: Workflow technology is needed to implement many key applications in Consolidated Supervision of Safety Production system. Using the workflow technology, an embedded workflow engine is designed for Shenyang Consolidated Supervision of Production Safety System. This thesis gives an introduction to the Architecture of the application system and the workflow engine and analyzes the design of the engine. The workflow engine shortens the developmental cycle of the key application significantly. After the application system based the workflow engine is deployed, it has improved the efficiency of business greatly displaying great results.

Keywords: workflow; workflow engine; embed; consolidated supervision of safety production

本引擎研制背景基于沈阳市安全生产监管系统项目。该系统主要用于对市辖区内所有企业的生产活动进行监督管理，对事故进行模拟、预测分析、调度、救援、进行统一处理等。

该系统的很多关键业务不仅能够处理日常的业务流程，而且能够通过各种技术手段提供决策支持。并且为了直观形象，系统引入了 GIS 技术，在提供决策的同时也能够用 GIS 技术提供非常直观的地图服务。该系统中存在很多诸如救援分析之类的关键业务，这类业务都涉及了大量的 GIS 服务。

在系统的建设过程中，引入了工作流技术。因为现有的开源工作流引擎很难直接集成处理 GIS 业务，所以我们通过对安全生产监管领域中关键业务的实际开发需求的分析，设计并实现了一个轻量级嵌入式工作流引擎[1]。

该工作流引擎的使用，不仅大大缩短了关键业务的开发周期，而且也规范了各种关键业务处理流程。目前，工作流引擎已经在沈阳市安全生产监管系统中成功担当了流程控制中心的角色。取得了很好的效果。本文对该工作流引擎的设计及实现技术进行分析。

① 收稿时间:2010-01-12;收到修改稿时间:2010-02-06

1 系统总体架构

为了方便与业务系统结合，使引擎能够更加灵活处理工作流程，参照工作流管理联盟(WFMC)的参考模型[2-3]，为系统设计实现了一个嵌入式工作流引擎。基于工作流引擎的沈阳市安全生产综合监管系统的结构如图 1 所示。该系统结构说明如下：

系统采用 JSF+Hibernate 的 J2EE 三层架构模式，其中 JSF 做页面显示和逻辑控制，Hibernate 来做数据持久。系统架构分为三层：WEB 客户端，工作流引擎控制中心以及数据层。

1) 系统客户端为 Web 浏览器，主要包括系统业务界面，也就是工作流执行界面；工作流管理界面，用于对流程实例的统一管理，比如终止、启动某个流程等等。

2) 业务系统的控制中心就是工作流引擎，它为业务流程的执行提供环境。其包括两个组成部分：引擎核心控制系统和引擎管理控制台。引擎控制系统负责对流程的解析、调度、执行[4]，调用与工作流相关的外部应用和 GIS 系统，同时存储工作流控制数据和相关数据以及对外提供交互接口。引擎管理控制台是给流程管理员使用的，用来监控流程的运行状况以及控制业务流程的执行。

3) 系统采用中心数据库来做数据管理。中心数据库包括引擎数据库、业务系统数据库以及 GIS 数据库。引擎数据库存储与工作流控制、运行相关的数据表，主要包括流程模板库、实例库、组织模型库等等。业务系统数据库是和具体业务有关的数据。GIS 数据库是 GIS 服务所需的空间数据。这里所有的数据操作都用 Hibernate Api 来实现。

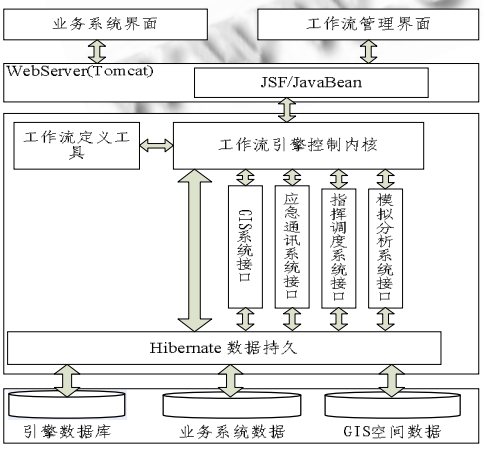

图 1 系统总体架构

2 工作流引擎设计

工作流引擎为工作流实例提供执行环境，负责解释流程定义，调度控制流程实例的执行，它是工作流管理系统的核心服务。因此，工作流引擎设计的好坏直接关系到工作流的执行效率和整个系统的性能[5]。图 2 为工作流引擎结构图，说明如下：

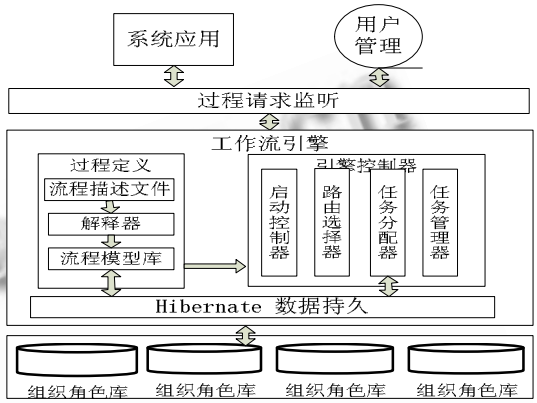

图 2 引擎结构图

工作流管理系统采用 XML 作为流程描述语言，在引擎内部用 DOM 方法来解析流程定义文件[6]。为了解决频繁解析流程定义文件而造成系统效率低下的问题，采用在数据库存储流程定义信息的方法，使工作流管理系统的效率得到明显改善。

对于工作流引擎来说的外部请求，都用“系统应用”表示，这里主要包括业务子系统的请求，引擎和引擎之间的调用请求，以及其它外部系统请求[7]。对工作流的监控管理请求，都简称为“用户”。

工作流引擎各个部分的主要功能如下：

2.1 解释器

解释器负责解析由流程建模工具所创建的流程模型文件(XML 文件)，然后将流程模型所包含的信息(创建者、名称、包、创建时间等相关属性)存入到由工作流模型 ID 唯一标识的流程模板库，从而为运行时期的工作流管理软件提供所需的模型静态信息。解析的过程中，同时用 XML 绑定技术解析出其中的 activity(活动)和 transaction(变迁)信息分别存入 WorkFlow_Model_Activity(该表存储与流程活动相关的信息，其中起始活动和结束活动必须标识)和 WorkFlow_Model_transaction(该表存储与流程变迁相关的信息，其中主要包含与变迁有关的两个活动的 ID 和变迁的条件)表中。从起始活动开始，在

WorkFlow_Model_transaction 表中查找后继活动，直到结束活动，就可以遍历出整个流程。工作流引擎运行期间也是以这两个表中的数据作为依据进行控制，从而不必频繁解析流程定义文件。

流程模型实际上是企业经营活动过程的一个模板，可以有多个相同的实例同时在运行，因此本系统在设计时支持多线程。另外，在企业活动过程中，模型的某个环节有可能会变更，如果采用直接在数据表中删除原来的模型信息再建立新的模型信息的方法，就要停止正在运行的实例，这样会造成效率低下。为了解决这个问题，我们在流程模型库里面的每条流程模型信息里面加一个模型版本字段，当流程模型有变更时，在模型库里面会产生一条新的记录，同时模型版本自动加一 。这样原来按照旧模型信息生成的实例就一直运行到结束为止，新流程实例都按照新流程模型来产生。

2.2 动控制器

启动控制器主要负责业务流程运行的初始化，根据解析过的工作流模型创建工作流程实例，此时流程实例被置为运行状态，同时将已经运行的工作流实例存入 WorkFlow_Instance)和 WorkFlow_Activity_Instance 中。

在此模块中，根据由解析模块中生成的过程模型类路径，创建流程实例运行所需要的实例类集，并且将其持久化。通过对启动控制器的调用来完成流程的实例化、流程的启动、活动的完成和后续活动的创建。

2.3 由选择器

路由选择器的功能是控制流程实例的流转，即根据流程模型定义和流程运行期间的应用数据，判断转移条件，进而正确选择后继活动，控制整个工作流程的正确流向。

由于工作流程定义相当于一个有向图，所以在工作流模型的解释过程中，在 WorkFlow_Model_ctivity 表中记录了每个“活动”接点的入度和出度。通过结点的入度与出度对节点的状态进行选择与并行控制。

工作流联盟(WFMC)提供的工作流基本的控制结构有：顺序(Sequence)，分支(Split)，聚合(Join)[2]。根据实际情况，分支又可以分为与分支(AND-Split)和或分支(XOR-Split)，聚合又可以分为与聚合(AND-Join)和或聚合(XOR-Join)[8]。

为了对结点状态选择和同步控制，引入一个状态控制变量 inDegreeNum，初值为 0。算法形式化描述如下：

If 该结点的 InTransCondition(输入变迁条件)为 True

inDegreeNum++;

If inDegree == 1，说明该结点和父结点的结构为顺序结构，则改变该结点状态为“就绪”。

Else If 该结点的变迁类型为 XOR-Join，则改变该结点状态为“就绪”。

End

Else if 该结点的变迁条件为 AND-Join

If inDegree == inDegreeNum,则改变该结点的状态为就绪。

End

End

End

当一个活动执行完毕后，这时需要控制其后继结点的状态，使流程能够继续往下流转。为了对结点状态选择和并行控制，引入另一个状态控制变量 outDegreeNum，初值为 0。算法形式化描述如下：

Step1:在流程模型库里面查出该结点出度并赋值给 outDegreeNum。

Step2: For i = 1:outDegreeNum

calOutTransCondition(i); //计算每一个输出变迁条件

If 该结点的类型为 XOR-Split，则说明只要有一个输出变迁条件为 True，其相应的后继结点状态就可以改为“就绪”，其它的后继结点就可以忽略。

End

If 该结点的类型为 AND-Split，这说明其所有变迁条件为 True,其后继结点的状态才能改变为“就绪”。继续等待至所有变迁条件为 True，则改变后继结点状态为“就绪”。

End

2.4 任务分配器

当由启动控制器产生一个流程实例后，引擎会产生一个新的工作流实例存入到流程实例表。对于一般的交互活动，根据模型信息，就可以把活动分配给某个执行者；如果某个活动的执行者为某一角色的话，则根据分配规则把任务分配给该角色，然后再根据业

务需要的规则来把任务分派给属于该角色的某个或者某些人员。

由于某些任务在流程模型创建过程中已经指定了由某个人来完成，也有一些任务是分配给某个角色。在系统中我们采用了“推拉”技术来保证任务分配给相应的人。如果某任务是直接指定给某个具体的人员，那么在流程执行时间就直接在这个人员的任务列表中加入该任务的 ID，也就是“推”给这个人员一个任务；如果某个流程是分配给某个角色，那么我们会把属于该角色的所有人员的任务列表中都生成一个任务实例，在流程实际执行过程中，采用竞选工作项方式，即“谁先请求谁先获得工作项”的原则，从而将任务分配给具体负责人，也就是“拉”的技术。

2.5 任务管理器

任务管理器的功能是管理每个流程执行者的任务列表。

工作流程中的所有任务按照是否由人参与的原则可以分为人工和自动两种。对于自动任务，就是程序自动执行的结点。那么当该任务为就绪状态时，就可以自动执行。对于人工活动结点，采用了基于角色的访问控制的方法。当不同角色的人员登录到任务列表管理器时，只能管理操作与本角色相关的工作项。

2.6 流程监控管理器

流程监控管理器的功能是对所有运行的工作流实例进行监控和管理，主要是通过修改工作流实例数据以及任务列表的各种状态（就绪，挂起，停止，删除，重新启动等）来实现。该功能供流程管理员来使用。例如当某个工作流任务快达到时限要求时，管理员可以修改该任务的优先级以达到使该任务被优先处理的目的，或者管理员可以给该任务的所有者发督办消息等等。

3 系统实现

工作流引擎是整个系统的核心，采用 J2EE 架构模式，WEB Server 用的是 Tomcat。下面就一个事故救援流程实例在引擎内部的流转控制过程进行说明。

首先根据具体业务需要，由流程定义工具产生如图 3 所示的流程描述文件（XML 文件），然后工作流引擎中的流程解释器解析出该模型名字以及创建人等信息，存入到流程模型库。同时也解析出来所有的 activity，transaction 以及 assignment，分别存入 WorkFlow_Model_Activity（流程模型表）和 WorkFlow_Model_transaction（流程变迁表）以及 WorkFlow_Model_assignment（流程角色定义表）。

```
<process name="事故救援">
  <start activity name="start1">
    <transition name="采集信息" to="采集事故信息"/>
  </start>
  <activity name="采集事故信息">
    <assignment handler class="cn.ac.sict.safetyProject.workflow.classA"/>
    <transition name="分析" to="事故模拟分析"/>
  </activity>
  <activity name="事故模拟分析">
    <transition name="转给科长" to="科长处理"/>
    <transition name="转给处长" to="处长处理"/>
    <transition name="转给局长" to="局长处理"/>
  </activity>
  <activity name="科长处理">
    <assignment handler class="cn.ac.sict.safetyProject.workflow.classC"/
    <transition name="实施救援" to="救援"/>
  </activity>
  <activity name="处长处理">
    <assignment handler class="cn.ac.sict.safetyProject.workflow.classD"/>
    <transition name="实施救援" to="救援"/>
  </activity>
  <activity name="局长处理">
    <assignment handler class="cn.ac.sict.safetyProject.workflow.classE"/>
    <transition name="实施救援" to="救援"/>
  </activity>
  <activity name="救援">
    <assignment handler class="cn.ac.sict.safetyProject.workflow.classF"/>
    <transition name="转到结束标记" to="结束"/>
  </activity>
   <end name="结束"/>
</process>
```

图 3 流程定义文件

当有事故救援请求时，流程启动器根据流程所需资源要求来初始化一个事故救援流程实例，将该记录存入到工作流实例库。任务路由器根据角色定义表中的类路径 cn.ac.sict.safetyProject.workflow.classA 执行指定任务函数，把相应人员或者角色的任务列表中放入任务记录，然后启动第一个任务。任务分配器根据流程描述采用“推拉”技术把第一个任务分配给相应的人员。那么这个流程就开始运行了，同时引擎根据任务优先级或者建立时间顺序把任务显示在相应工作流执行者的任务列表中。

采集事故信息的人员登录系统后，会根据任务列表中的任务清单“领取”执行任务，填写事故的各种参数，便于下一步模拟分析。当该任务执行结束后，工作流引擎路由控制器根据流程流转规则用同样的方法来确定下一步事故模拟的任务由谁来完成。同时修改当前任务结点的状态和相应人员任务列表。如此循环往复就可以实现整个工作流程的运转。当路由选择器发现后续活动结点为结束节点时，说明整个工作流程实例已经结束。图 4 为系统运行的一个界面。

图 4　系统运行一个界面

在整个流程运行过程中，管理员可以在任何时间登陆流程管理系统，根据业务需要对整个流程实例进行干预控制，以达到整个业务系统的需要。

4　结束语

本文运用工作流的理论和方法设计实现了一个针对沈阳市安全生产综合监管系统的嵌入式工作流引擎。工作流引擎以中心数据库为核心来实现，可以解决频繁解析流程定义文件的问题，并且使该系统具有一定的事务处理和错误恢复能力，大大提高了系统的健壮性。而且该引擎的运用，大大缩短了系统中关键业务的开发周期。

参考文献

1 何清法,李国杰,焦丽梅等.基于关系结构的轻量级工作流引擎.计算机研究与发展, 2001,(2):129－137.

2 Workflow Management Coalition:Workflow management coalition terminology and glossary, Technical Report, WFMC-TC-1011, 1996.

3 WFMC. Workflow Process Definition Interface—XML Process Definition Language [2008-10-20]. http://www.wfmc.org/,.

4 范玉顺.工作流管理技术基础.北京:清华大学出版社,2001.

5 van derAALSTW, vanHEE K.工作流管理—模型,方法和统.王建民,闻立杰等译.北京:清华大学出版社,2004.

6 金鑫,许静,李学孟等.基于 XML 的工作流引擎的设计与实现.计算机工程, 2007,33(23):72－73.

7 战德臣,徐晓飞.基于工作流引擎的构件组装体系结构.软件学报,2006,17(6):1401－1410.

8 叶娜,李健.工作流引擎推进过程中 m 选 n 问题的研究.计算机应用研究, 2009,26(11):4098－4100.