**工作流概念及概要设计**

目 录

[1. 介绍 4](#_Toc291229579)

[1.1 目的 4](#_Toc291229580)

[1.2 范围 4](#_Toc291229581)

[1.3 定义和缩写 4](#_Toc291229582)

[1.4 依赖文档 4](#_Toc291229583)

[1.5 概述 5](#_Toc291229584)

[2. 需求范围 5](#_Toc291229585)

[2.1 需求概述 5](#_Toc291229586)

[2.1.1 系统管理员 5](#_Toc291229587)

[2.1.2 用户 5](#_Toc291229588)

[2.1.3 维护人员 6](#_Toc291229589)

[2.2 需求界定 6](#_Toc291229590)

[2.2.1 流程定义 6](#_Toc291229591)

[2.2.2 流程执行 7](#_Toc291229592)

[2.2.3 控制管理 8](#_Toc291229593)

[3. 状态视图 8](#_Toc291229594)

[3.1 状态分类 9](#_Toc291229595)

[3.2 状态切换 9](#_Toc291229596)

[3.3 流程执行中的状态切换 10](#_Toc291229597)

[4. 过程视图 10](#_Toc291229598)

[4.1 涉及方 10](#_Toc291229599)

[4.2 线程分类 10](#_Toc291229600)

[4.2.1 消息守护线程 10](#_Toc291229601)

[4.2.2 轮循线程 10](#_Toc291229602)

[4.2.3 工作线程 11](#_Toc291229603)

[4.3 进程间通讯机制 11](#_Toc291229604)

[5. 逻辑视图 11](#_Toc291229605)

[6. 接口视图 11](#_Toc291229606)

[7. 数据视图 11](#_Toc291229607)

**本文档修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修订时间** | **修订人** | **修订内容** |
| 0.1 | 2011-4-22 | 金明岩 | 初稿 |
|  |  |  |  |
|  |  |  |  |

# 介绍

工作流引擎能够维持流程的运行状态,并在人和应用之间分派活动。工作流定义一组活动以及活动的流程逻辑，这样将流程逻辑与业务逻辑分离；工作流负责根据流程逻辑调度流程的执行路径，调用相应的业务逻辑。

## 目的

本文档描述了工作流结构设计的具体内容，为后续的开发提供指导，同时也为理解和评审提供依据，为以后理解和使用工作流框架提供了帮助。

## 范围

文档描述了工作流目前的需求范围，从几个视图描述工作流的设计思想。

## 定义和缩写

|  |  |
| --- | --- |
| 名词或缩写 | 定义 |
| 流程定义 process definition | 业务流程或过程的规格化描述。定义了工作流流程的内容和执行的顺序。 |
| 工作流实例 process instance | 已经运行同时还没有结束的一个流程， 处于中间的某个活动。 |
| 活动activity | 流程中的每一个节点，描述一个工作的逻辑单元，不可再分的过程。 |
| 工作流引擎 WorkflowEngine | 为工作流提供了运行时期的执行环境，调度、监控、管理 工作流实例。 |
| 待办事项 | 某个用户当前应该处理的流程中的活动，流程在等待这些活动完成之后才能继续执行。 |
| 轮循线程 | 工作流引擎启动后，轮循线程定期执行，到数据库查找状态为就绪的流程 ，然后新建工作线程，处理流程。 |
| 工作线程 | 当前正在工作的线程，负责处理用户的业务。可能是某个线程创建，处理完用户业务后就结束。 |
| 流程上下文 | 流程在执行过程中需要用到的一些数据。一部分是从定义文件中读出的，不保存到数据库；另一部分是保存到数据库的数据，包括流程的信息和用户的业务数据。 |

## 依赖文档

## 概述

文档首先描述了需求范围，然后从逻辑视图、过程视图、接口视图和数据视图分别介绍系统。

# 需求范围

根据工作流管理联盟（WFMC ,Workflow Management Coalition）的定义，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、信息或任务能够在不同的执行者之间进行传递与执行。“The automation of a business process, in whole or part, during which documents ,information or tasks are passed from one participant to another for action, according to a set of procedural rules.  （WFMC ，1996 ）”

## 需求概述

工作流有三种角色：系统管理员，用户 ，服务维护人员。

### 系统管理员

负责工作流的正常运行。主要的工作：1）流程定义管理：定义流程，发布流程，删除流程。2）流程运行管理：查看流程执行的状态，停止一个实例、唤醒实例，删除工作流实例。



### 用户

使用工作流完成实际的需求。主要的工作：根据流程定义启动一个流程，找到当前自己应该处理的活动，执行活动。



### 维护人员

负责工作流引擎的运行。启动工作流引擎的服务，停止工作流引擎的服务。



## 需求界定

对于工作流的功能，有些功能是我们想到但是目前暂时不实现的功能。下面对目前要实现的功能作一个界定（从三部分说明：流程定义；流程执行；控制管理），可以根据实际情况选择实现那些，不实现那些。

### 流程定义

内容主要包括：

1）工作流流程定义和保存。

2）流程定义的详细格式。

3）描述工作流中的一个活动（activity ）。（需要一个结构化的模型，具有调用层次，再引入一个状态转移模型，没有调用层次。 ）

4）角色在这里只是一个标识，工作流引擎保存实例状态时同时保存角色，而不作解释。在查询某人的待办事项时角色作为一个过滤条件 ，需要有独立的角色系统，或者与其他角色系统关联。

5) 支持嵌套process。

6) 单独定义开始节点和结束节点

7) 提供图形化界面定义流程。

8) 在工作流引擎内部权限控制，API操作权限控制.

### 流程执行

1）流程上下文持久化保存，保存一个流程上下文，包括：当前 流程的信息和用户的业务数据。

2）流程以异步方式启动。启动可能是一个UI调用、可能是某个应用程序调用、可能是一个定时器调用、也可能是外部消息调用。工作流引擎的API提供流程的启动能力。

3）流程的起点默认为流程定义中的第一个活动，工作流引擎负责定位起点、提供启动方法。

5) 状态保持:工作流引擎在挂起流程的活动时将 流程上下文中的数据保存在数据存储中。

6）流程唤醒 ：流程的唤醒常见的有两种方式：通知和轮循。

7）流程的结束常见的有两种方式：显式结束：有一个结束步骤，只要任一分支到达结束步骤，就结束整个流程 ；隐式结束：没有显式的结束步骤，当没有后续活动时自然中止。

8）错误的处理。

流程分类包括：

1）顺序模式：按顺序执行每一个活动。



2）循环模式：某个活动执行完以后，根据一定的条件可能会跳转到前面的活动重新执行。



3）XOR分支模式 （decisions）：XOR split表示异或分支执行方式。



流程在执行完Activity1后，产生了两个异或执行的分支（Activity1—>Activity2 和Activity1—>Activity3）。这两个分支之间是替代的，也是异或执行的。在同一时间只能选择其中之一执行，或者执行Activity1—>Activity2，或者执行Activity1—>Activity3。

4）XOR聚合模式 （merges）

XOR merge表示异或聚合执行方式。不用显式支持流程定义。可以使用一个后续活动代替。



一旦聚合点之前的某一个分支（Activity1或Activity2）执行通过，不需要等待其他分支的到来，流程就会继续执行到下一个活动（Activity3）。

5）AND split分支模式 （forks）表示并发分支执行方式。



流程在执行完Activity1后，产生了两个并发执行的分支（Activity1—>Activity2 和Activity1—>Activity3）。这两个分支之间是对等的，也是并行执行的。这两条分支每次都同时执行，不存在选择，Activity1执行完以后就会同时执行Activity1—>Activity2和Activity1—>Activity3。

6) AND聚合模式（joins）。AND merge表示并发聚合执行方式。



流程执行到聚合点，会等待所有分支的到来，如果不考虑超时和异常等情况下，流程必须等待所有的分支（Activity1 和Activity2）都执行完后，才能激活后续的活动（Activity3）。

### 控制管理

工作流平台提供管理和监控工具，管理正在执行中的工作流实例，可以查看目前工作流实例执行的状态，能够唤醒、销毁某一个工作流实例。

# 状态视图

工作流实例保存流程的状态 、当前的活动、用户的业务数据。活动不保存状态，所以也就没有活动实例，只有当前活动。

## 状态分类

    流程的状态分为：

1)运行：一个流程正在执行的状态。

2)挂起：流程中的某个活动需要外部干预才能继续执行，就处于挂起状态。

3)就绪：外部干预之后流程处于就绪状态，可以由通知或轮循线程进入运行状态。

## 状态切换



**1）开始—>“就绪”状态**

    用户根据一个已经定义好的流程定义文件启动一个流程，流程处于就绪状态。

**2）“就绪”状态—>“运行”状态**

    轮循线程查询就绪状态的流程上下文，新建一个新的工作线程，将流程状态设置为运行，然后运行这个流程。

**3) “运行”状态—>“运行”状态**

    工作线程处理完流程的当前活动，查找下一个活动。如果下一个活动不需要外部UI干预，那么就继续运行 。

**4） “运行”状态—>“挂起”状态**

    工作线程处理完流程的当前活动，查找下一个活动。如果下一个活动需要外部UI干预，流程状态改为挂起。工作线程结束。

**5） “挂起”状态—>“就绪”状态**

    用户查询到自己的代办事项（也就是那些需要外部干预的已经挂起的活动），录入数据后提交。流程上下文的状态由挂起转为就绪，然后通知线程执行这个活动；如果通知没有使流程运行，就等待工作流引擎轮循线程处理。

**6） “运行”状态-->结束**

    工作线程处理完流程的当前活动，查找下一个活动。如果找不到下一个活动，就说明流程结束了。这时要删除流程的上下文。

## 流程执行中的状态切换



 流程的每一个活动从开始先进入就绪 状态；就绪状态经过外部干预进入运行状态；运行状态可能进入下一个运行状态，可能进入挂起状态，可能结束流程；挂起状态经过轮循线程激活进入就绪状态。

# 过程视图

## 涉及方

工作流涉及到的几个服务器和使用方：

客户端，通过API方式调用接口提供的方法；

工作流引擎，启动，随时创建多个工作线程；

数据存储，持久化流程上下文；

消息服务器，收发消息通知。

## 线程分类

在工作流引擎主要有如下线程：

### 消息守护线程

1）工作流引擎启动后，就启动消息守护线程。

2）消息守护线程收到消息服务器发来的消息后，新建一个工作线程。

3）新建的工作线程根据消息的内容，到数据库查出流程上下文，继续处理流程。

4）工作线程一直运行到流程结束或是流程挂起后结束。

### 轮循线程

1）工作流引擎启动后，就启动轮循线程。

2）轮循线程定期执行，查找状态为就绪的流程。

3）如果查询到了就绪的流程，轮循线程新建工作线程，继续处理流程。

4）工作线程一直运行到流程结束或是流程挂起 后结束。

### 工作线程

1）工作线程可能被消息守护线程和轮循线程创建，负责执行具体的流程活动。

2）工作线程有多个，每一个线程处理一个流程活动。

3）工作线程一直运行到流程结束或是流程挂起后结束。

## 进程间通讯机制

《待续》

# 逻辑视图

工作流涉及到的类：接口、接口的实现、引擎、流程上下文。

# 接口视图

# 数据视图