基于 SharePoint 的工作流引擎的实现

金飞腾 赵正德 张 东 卢志国
(上海大学计算机工程学院,上海 200072)

摘 要 工作流管理系统能提高企业的生产效率,而工作流引擎是工作流管理系统的核心。为了对工作流进行更有效的管理,提出了一种新的基于 Microsoft 的 SharePoint 协作平台的工作流引擎实现方案,即先利用嵌入 C#逻辑代码的 XML 模板定义流程,再利用 Agent 服务框架实现引擎的逻辑组件。该方案使用户可轻松实现流程自定义和扩展出其他现有系统的接口,以保证工作流管理系统的灵活性和可扩展性。

关键词 工作流引擎 XML 工作流模板 WSS/SPS Agent 服务
中图法分类号: TP315 **文献标识码**: A **文章编号**: 1006-8961(2006)11-1552-05

Realization of Workflow Engine Based on SharePoint

JIN Fei-teng, ZHAO Zheng-de, ZHANG Dong, LU Zhi-guo
(*School of Computer Engineering and Science, Shanghai University, Shanghai* 200072)

Abstract The workflow management system can improve enterprise's productivity and working efficiency. The workflow engine is the core of the workflow management system. This paper realizes a workflow engine based on Microsoft's cooperative platform: SharePoint, using XML Template with embedded C# code to define workflow, using agent service to realize the engine's logic components. Users can easily define their workflows and extend existing systems' interface by defining new port. This engine ensures the workflow management system's flexibility and extensibility.

Keywords workflow engine, XML workflow template, windows sharepoint services/sharepoint portal server(WSS/SPS), agent service

1 引 言

工作流管理是近年来计算机应用领域发展最为迅速的几项新技术之一,其可广泛应用于办公自动化、文件管理、群件应用、BPR(business process reengineering)等领域。

由于工作流管理系统能合理、高效地支撑企业业务流程的运转,实现业务流程电子化、信息化,并能将业务流程中的信息流转与控制管理从手工填写、传递纸介质表单、文件的手工作业方式中解放出来,因此可提高企业的生产效率和减少企业的资源消耗[1]。

工作流引擎是工作流管理系统的核心,这种工作流管理系统的好坏就在于是否有一个功能强大的工作流引擎。本文提出了一个基于 SharePoint 的工作流引擎,并且在 windows SharePoint service 上得到了实现。

2 相关概念

2.1 SharePoint、WSS、SPS

Microsoft SharePoint 产品与技术利用友好的 Web 服务和应用程序,为用户提供在一个单一汇聚工作台中连接人员、流程和数据的能力。

WSS(windows sharepoint services)是用于创建 Web 站点的引擎,它不仅为团队协作提供了站点,而且实现了信息共享和文档协作。

SPS(SharePoint portal server)利用 WSS 平台提供的 Web 部件和文档库等技术,将站点、区域、个人、知识和业务流程连接在一起,以使用户能够协同处理文档、任务、联系人、事件及其他项目。

收稿日期:2006-06-28;**改回日期**:2006-08-03

第一作者简介:金飞腾(1982 ~),男。2004 年获辽宁石油化工大学学士学位,现为上海大学计算机工程学院硕士研究生。主要研究方向为 Web 应用技术、协同技术研究。E-mail: ykjxk@yahoo.com.cn

2.2　工作流、引擎、模板、实例

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,使得文档、信息或任务能够在不同的执行者之间进行传递与执行。工作流中有多个参与者按照一定的规则进行活动,如相互传递文档、信息、任务等[2]。它包括了不可缺少的两部分:

(1)控制流:其决定了流程执行的步骤,由工作流引擎负责实现;

(2)数据流:其决定了流程所处理的具体内容,并需要一个载体来实现。

工作流引擎:其是工作流管理系统中的调度器,其不仅能自动维护工作流经营过程的状态和接收外界传递的数据或信息(这些信息可以是数据库的增删,文档库的文档被修改等),而且可根据内部的预定义的规则、状态、外部信号、当前的上下文等来推演,并执行后续的执行过程。

工作流模板和实例[3]:模板是一系列需要动态调整、修改的过程和执行规则的集合,可用于描述一种经营过程。工作流管理系统可把不同的流程、不同的规则,基于集合封装为工作流模板。而模板的加载、执行、卸载则是由工作流引擎来处理的。工作流实例是一种经营活动的一个实例,也是工作流引擎处理消息的基本对象。工作流管理系统则把某一个工作流实例相关的事件、消息、规则封装到一个单独的上下文中。

3　工作流引擎的设计原理

3.1　开发环境

Windows Server 2003、Windows SharePoint Services2.0、VS.NET2003、ASP.NET、SQL Server2000、Microsoft Office System

3.2　支持特性

本论文实现的工作流引擎将支持以下功能:

(1)流程高度自定义:可用 XML 格式的流程定义模板,并用嵌入标准的 C#代码描述流程逻辑,然后更改模板便可以实现流程的自定义。

(2)可扩展的端口系统:它提供了对 SPS 文档库、表单库的支持,并支持对磁盘文件系统、邮件系统、数据库系统以及 WebService 端口的扩展。

(3)完善的状态监视功能:用于监视当前运转工作流实例,如运转到哪一步、是否触发异常等。

(4)基于 Agent 服务底层框架,用于支持工作流集群及负载均衡等高级特性。

3.3　引擎构成结构

引擎构成结构如下:

(1)Agent 服务层:作为 Container,可提供以下一些基本的集成环境:线程池、消息路由器、类工厂、各种 Register 注册器等;

(2)逻辑组件层:系统实例中的各种逻辑组件,包括工作流的模板库、端口扫描器、各种监护例程等,该层与领域知识相关;

(3)工作流实例层:每一个工作流实例派生于不同的工作流模板,且有不同的生命周期、生命周期管理、状态保存、逻辑代码调用等。

该结构优点有:

(1)Agent 服务可同时部署于两台机器,可实现消息路由、基本的负载均衡。

(2)可以同时存在多个工作流实例、多个数据流载体,不仅每一个实例都有独立的上下文环境,且每个工作流之间相互不受干扰。

引擎构成结构图如图 1 所示。

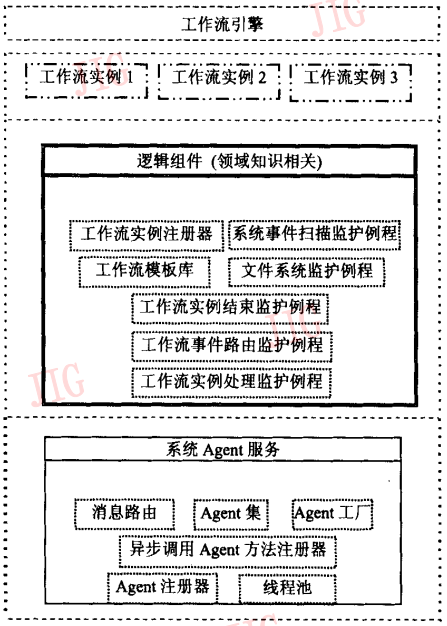

图 1　引擎结构图

Fig. 1　Engine's Structure

3.4　事件处理流程

每一种工作流程信息都使用一个 XML 格式的工

作流模板来描述,用户能快速地替换模板和更改流程定义。一个工作流实例通过一个运行在底层的线程池的看护例程(daemon)来维护其状态和迁移等操作。

引擎中的各种 Adapter 捕获外部端口的消息事件,如实例中的 SPS Adapter 用于捕获文档库的文档创建和文档修改等事件消息。Adapter 根据消息的 ID 提交给对应的看护例程。看护例程随后根据消息/事件的内容、当前上下文(context)来执行对应匹配的逻辑代码。事件处理流程伪代码如下:

```
//消息扫描 Agent;
从消息队列里得到一个工作流事件;
发送事件到事件路由 Agent;
//事件路由 Agent
If(该事件含有 GUID){
    根据 GUID 查找定位绑定到该实例的 Agent;
    把事件发送给该 Agent;}
Else{
    //新工作流实例创建 Agent
    调用工作流实例生成器;
    创建新的工作流实例 Agent;
    定位到匹配该实例的工作流模板;
    根据模板创建新的工作流实例;
    创建一个新的 Agent 绑定到该实例;
    运行该 Agent;
    把事件加入到 Agent 的事件处理队列中;}
//与工作流实例绑定的 Agent
处理该事件;
    //工作流实例结束 Agent
    If(工作流实例结束){
        释放该工作流实例;
        注销与该工作流实例绑定的 Agent;
        结束;}
    Else{等待下一个事件;}
```

3.5 端口扫描模块

端口是引擎与外界进行沟通和访问外界资源的点(如 SPS 文档库是一个端口)。引擎可以扫描到端口的变化,并且可对端口执行某些操作,如文件的拷贝,删除[4]。

在本引擎中,实现了端口注册器和端口扫描器。注册器用于管理引擎和外界沟通的所有端口。引擎启动后,加载工作流模板和流程定义中的端口。如果某一端口已经注册(表示端口实例已经存在),就直接查找其对应的端口扫描器;如果没有注册,则生成一个端口实例,并进行注册,然后查看是否有扫描器实例对该端口进行扫描,如果没有,就创建新的端口扫描器实例。扫描器用于监视端口的变化(如用户对文件更改,Email 邮件的到达等),然后将消息提交给看护例程。

3.6 工作流模板定义

下面结合论文实现的实例来说明端口、工作流和执行逻辑的定义:

3.6.1 端口定义

该端口表示费用报销库的端口,如果有新的报销表单被创建,则引擎便可以得到这个事件,如下代码段是对该事件的定义:

```
< Port Name = "报销表单创建" Assembly = "MySPS
WFEngine. InfopathFormLibrary. dll" Type = "My
SPSWFEngine. InfopathFormLibrary. FormPort",
Url = "文档库名:http://MySPSSite/费用报销/"
EventFilter = "EventName:Created" >
```

3.6.2 审批工作流流程定义

某 XML 格式的工作流流程定义如下:

```
< MySPSWorkflowTemplate >
  < States >
    < State Name = "等待 manA 审批"
    DocumentLibraryTitle = "manA 审批库" >
      < Properties >
        < Property Name = "manA 已审批" >
        < Commands >
            < Command Value = "提交给 manB" >
            < Actions >
                < Action Assembly = "
CalledProcessAssemblyName" Type = "
ProcessTypeNameParameter" = "manB 审批库" >
                </Action > </Actions > </Command >
</Property > </Properties > </State > </States >
</SPSWorkflowTemplate >
```

该流程定义中,据销单在费用审批流程中的状态是:报销单在 manA 审批库,等待 manA 审批,若审批通过,则必须执行的下一步是提交给 manB,提交给事件处理程序的参数是:"manB 审批库"。

3.6.3 执行逻辑定义

费用审批流程中,主管通过后,若审批金额大于 500,则提交 manA 审批,若小于 500,则提交 manB 审批。int[ ]{i}表示流程中,两个状态结点之间的连接弧。在 XML 流程定义的模板里嵌入的 C#代码示例如下:

```
< Transition > < SourceCode > < ! [CDATA[
myForm = (lastEventas WFOutsideEvent
_FormLibrary). InfopathForm;
```

```
string auditorNotion = myForm.GetData("
my:FlowData/my:AuditorNotion");
if (auditorNotion = "批准") {
  string temp = myform.GetData("
  my:InfopathForm/my:TotalSum");
  float totalSum = 0;
  if (temp ! = string.Empty)
    //得到表单中员工报销的金额数
    totalSum = float.Parse(temp);
    //用户可以在 XML 文件中更改该审批阈值
  if (totalSum > 500)
    //走到弧 3,表示提交 manA 审批
    return new int[ ]{3};
  else
    //走到弧 4,表示提交 manB 审批
    return new int[ ]{4};
}
else
    //主管不批准,走到弧 7,表示表单不通过
    return new int[ ]{7};
]]> </SourceCode> </Transition>
```

3.7　引擎中对文档库的操作

WSS/SPS 提供以下 4 种方式来访问文档库:ObjectModel、WebService、WebDAV、FrontPage RPC。本引擎采用 ObjectModel 访问 WSS/SPS 中的文档库。

引擎中用到的在 Microsoft.SharePoint 名称空间下的几个 Assembly 为:

SPFile:文件对象(Infopath 表单,文档等)

SPFileCollection:文件集合对象

SPDocumentLibrary:文档库类

引擎对文档库中的文档编写了以下 4 个核心的操作代码:copy、move、delete、setAttributes。

文档库中,文档 move 操作的示例代码段如下(其作用是把所有文档都移动到 C 盘中):

```
//根据上下文,得到当前所在的 SPS 站点
SPWeb web = SPControl.GetContentWeb(Context);
//得到 SPS 中名为共享文档库的文档库对象
SPDocumentLibrary docLib = (SPDocumentLibrary) web.
Lists["共享文档库"];
//遍历文档库中的文档
foreach (SPListItem item in docLib.Items){
  SPFile file = item.File;
  byte[ ] fileBuffer = file.OpenBinary();
  string path = "c:\\" + file.Name;
   FileStream stream = new FileStream(path, FileMode.
Create);
  stream.Write(fileBuffer,0,fileBuffer.Length);
  stream.Close();
}
```

4　采购申报流程实例的实现

4.1　实例设计

实例中,WSS/SPS 作为静态的文档共享区/服务器,数据流部分都存储在 SPS 服务器上。InfoPath 作为申报电子表单的设计器和填写器,用于快速自定义和填写基于 XML 格式的电子表单。SharePoint 中提供了对 InfoPath 表单的良好支持,其体现为表单库(form library)。InfoPath 表单能在引擎的驱动下,根据工作流定义、审批人的动作,在站点的不同文档库中进行流转。

4.2　工作流实例示例

示例流程如下:

在 WSS/SPS 中有 4 个表单库:员工库,主管库,经理库,采购部库,查看权限分别对应到员工、主管、经理、采购部。员工提交购买申报表单后,引擎会根据流程定义将文档自动提交并转移到下一个审批人的表单库,并且触发自定义的消息,发送邮件通知下一个审批人。看护例程会监视表单审批状态,并把最新的审批状态显示在提交人视图中。最后审批通过后,表单会出现在采购部文档库,供采购部人员根据表单进行采购。

在图 2 中,员工要申报机器购买,首先根据已自定义好的表单模板新建表单,并指定工作流程:审批人、审批动作、提醒方式、通过阈值。

图 2　申报 InfoPath 表单填写

Fig. 2　Writing the InfoPath form

在图 3 中，若以主管身份登录，则该用户对其他 3 个文档库都没有查看权限。机器购买申报表单是从员工提交开始，流到主管审批库，等待主管审批。若主管审批后，则将在引擎驱动下，自动提交到总经理审批库中。

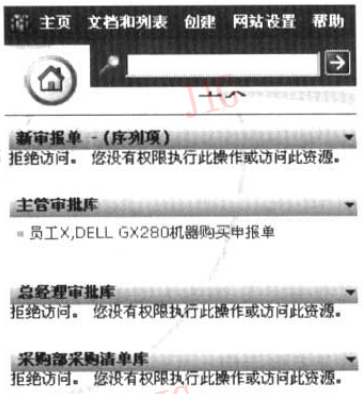

图 3 SPS 中采购申报站点图

Fig. 3 Auditing site in SPS

5 结 论

本文实现了基于 WSS/SPS 的工作流引擎，并把 WSS/SPS 和 Office 文档作为数据流载体，先使用对象模型存取文档库，然后使用嵌入 C#逻辑代码的 XML 模板来定义流程，再使用端口/适配器来获取工作流消息/事件。该引擎提供的标准可扩展的接口，可以方便地扩展出磁盘系统、Email 系统、SAP 系统等特有的接口，只要这些系统提供的 API 封装成端口便可实现扩展。有待进一步完善的功能是：①使用 Visio 控件来开发可视化的流程设计器，以使用户能在设计器中方便地设计出流程；②开发设计器的流程监视显示功能，以便能动态显示某一个工作流实例的运行状态。

参考文献(References)

1 Hu Heng-ying. The research and realization of workflow in office automation system [D]. Wuhan: The Wuhan University of Technology, 2005. [胡恒莹. 办公自动化系统中工作流的研究与实现[D]. 武汉：武汉理工大学，2005.]

2 Li Yi-jiang. Design and implementation of workflow engine based on. Net[D]. Wuhan: Huazhong University of Science and Technology, 2004. [李一江. 基于.Net 的工作流引擎的设计与实现[D]. 武汉：华中科技大学，2004.]

3 Qiu Shao-shan. A lightweight workflow engine based on XML and relationship[D]. Dalian: Dalian University of Technology, 2003. [邱少山. 基于 XML 和关系结构的轻量级工作流引擎[D]. 大连：大连理工大学，2003.]

4 Wu Qu-tao. The design and implementation of process-based lightweight workflow engine[D]. Shanghai: Shanghai University, 2004. [吴曲涛. 基于过程的轻量级工作流引擎的设计和实现[D]. 上海：上海大学，2004.]