第35卷 第18期 计 算 机 工 程 2009年9月
Vol.35 No.18 Computer Engineering September 2009
·软件技术与数据库· 文章编号：1000—3428(2009)18—0069—03 文献标识码：A 中图分类号：TP302.1

基于 IoC 模式的工作流与应用系统的集成

敖丽敏[1]，祝晓东[2]，周 炜[3]
(1. 东北电力大学信息工程学院，吉林 132012；2. 群硕软件开发(上海)有限公司，上海 201203；3. 华北计算技术研究所，北京 100083)

摘 要：针对在应用系统中引入工作流技术进行集成的问题，提出应用控制反转模式对嵌入式工作流系统进行集成的方案，以降低集成耦合度、减少系统复杂性、提高组件重用率。避免不适当的集成方法导致应用系统代码的侵入、结构的破坏以及系统紧耦合所造成的系统灵活性和可扩性差、维护难度大的问题。给出开源工作流系统 OSWorkflow 和基于 Java EE 轻量级架构的应用系统的集成实例和实现过程。
关键词：控制反转模式；工作流；应用系统；集成

Integration of Workflow with Application System Based on Inversion of Control Pattern

AO Li-min[1], ZHU Xiao-dong[2], ZHOU Wei[3]
(1. Dept. of Information Engineering, Northeast Dianli University, Jilin 132012; 2. Augmentum, Inc., Shanghai 201203;
3. North China Institute of Computing Technology, Beijing 100083)

【Abstract】The application of Inversion of Control(IoC) pattern is introduced to resolve the problem of integration while introducing embedded workflow technology into the application system, so that the coupling of integration and system complexity can be reduced, and the rate of component reuse can also be improved effectively. The problems such as the invasion of system codes, the damage of structure, less flexibility and expansionary led by the tightly coupled system and difficult maintenance can be avoided which result from an inappropriate integration method in the application system. An integration example of Open Source Workflow(OSWorkflow) system and the application system based on Java EE lightweight architecture are given.
【Key words】Inversion of Control(IoC) pattern; workflow; application system; integration

工作流技术主要用于企业业务过程分析、模拟、定义以及操作实现，是实现企业业务过程管理和控制、过程集成、过程重组的核心技术[1]，它改变了企业的信息处理方式和企业应用间的协作方式，对企业经营和管理产生了重要影响。工作流技术已成为企业信息化建设方案中必不可少的内容之一，但是目前的工作流管理系统产品大都要和一定应用系统结合起来才能发挥作用。由于应用系统的多样性和差异性，工作流管理系统和应用系统的集成常常因方法不当而造成对应用系统代码的侵入和结构的破坏，导致系统紧耦合，降低系统的灵活性和可扩性、增加维护的难度。

1 系统集成模式介绍

根据工作流管理系统与应用系统的关系，目前可以将工作流管理系统分为自治式和嵌入式 2 大类[2-3]。自治式工作流具有独立的数据库和消息中间件，将所有的业务处理功能作为外部应用来调用，有的还提供业务表单设计的功能；而嵌入式工作流是嵌入在应用程序中，它的流程控制通过应用程序调用 WAPI(Workflow API)来实现，是流程管理和控制功能的基础构件[2]。本文介绍的集成模式主要参考文献[3]，即自治式工作流系统集成模式和嵌入式工作流系统集成模式。

1.1 自治式工作流系统的集成模式

在自治式工作流系统集成模式中工作流系统与业务应用系统的交互方式上，工作流引擎以远程调用方式提供 WAPI。工作流引擎必须为如何调用业务应用提供解决办法。一般情况下，自治式工作流引擎应该能够为直接调用外部业务应用提供远程接口(如基于 RMI, JMS 或者 Web Service 的业务接口)，另一方面，业务应用必须为自治式工作流引擎提供远过程调用业务方法[4]。如图 1 所示，应用系统通过 WAPI 访问和驱动工作流引擎，工作流引擎则通过流程中的设定代理触发和调用应用系统。

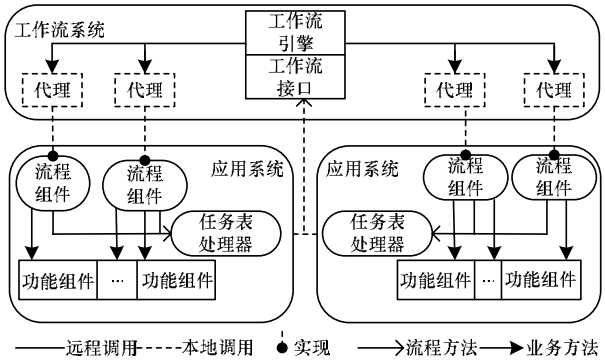

图 1 自治式工作流系统的集成模式

基金项目：东北电力大学博士启动基金资助项目(BSJXM-200601)；2006 年研究生教改立项基金资助项目“计算学科的‘并行计算’课程的课程建设”
作者简介：敖丽敏(1956－)，女，教授、博士，主研方向：计算机软件，计算机在电力系统中的应用；祝晓东，工程师、硕士；周 炜，助理工程师、硕士
收稿日期：2009-03-15 **E-mail：**aolm@263.net

自治式工作流系统本身就是一个单独的应用，作为服务应用如果没有基于某个中间件技术，自治式工作流引擎必须自己实现多线程同步、网路通信处理、资源池等服务端技术，实现的成本高、技术复杂。集成能力也会受到工作流接口和代理方式的限制。

1.2 嵌入式工作流系统的集成模式

嵌入式工作流引擎不能单独运行，一般部署在应用系统中，作为应用系统的组件，用于控制业务逻辑的执行顺序、管理任务队列和流程状态等，因此通过调用本地方法实现工作流集成。应用系统可通过工作流扩展和组件扩展 2 种方式，实现业务功能和工作流的集成，如图 2 所示。

(1)工作流扩展：业务组件为最小功能点，通过工作流代理的流程组件由引擎直接驱动。显示层组件直接访问工作流接口，适合面向流程的系统开发。

(2)组件扩展：流程组件扩展了原有业务组件，并通过调用工作流接口实现流程驱动。显示组件访问特定流程组件，实现对已有系统增加工作流管理。

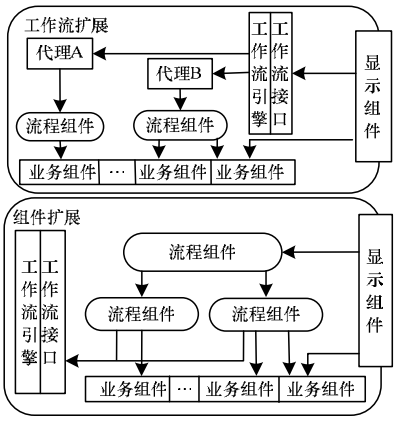

图 2 嵌入式工作流系统的集成模式

这 2 种扩展方式各有其优缺点，工作流扩展适合于设计按照工作流标准组织(WfMC)制定的标准系统，流程可完全由流程定义文件描述，能够方便地通过流程定义文件的改变调整流程，但是要求功能组件的划分必须符合流程原子事务约定；而组件扩展则适合那些业务组件不能保证都满足工作流原子事务约定，且表示层的调用方式不易改变的情形，但该方式在改变流程时需更改或添加新的流程组件，并修改显示层的调用或配置。

2 控制反转模式及其集成原理

2.1 控制反转(IoC)模式简介

控制反转(Inversion of Control, IoC)[5]是 Apache Avalon 项目创始人之一 Stefano Mazzocchi 提出的，目的是强调设计的安全性。依赖注入(dependency injection)是 Martin 对 IoC 模式的一种扩展的解释：高层不要依赖于底层，两者都要依赖于抽象；抽象不要依赖于细节，细节要依赖于抽象。IoC 是一种用来解决组件之间依赖关系、配置及生命周期的设计模式，其中对程序中的组件依赖关系的处理是 IoC 的核心。

控制反转模式原理是：组件本身不主动调用它需要完成某种功能的组件，而是声明所需组件的接口，并告知自己的上级管理者。上级管理者根据组件的声明来提供满足接口的某一组件。这样组件之间的依赖关系就被反转，组件不再和其他组件建立直接的依赖关系，依赖关系的建立可在其他地方实施。在编制程序时只要写被调用者的接口，其实例可通过配置实现。组件之间不再硬编码在一起，任何组件就可以得到最大程度的重用。

2.2 基于控制反转模式的集成原理

一个应用程序一般由多个组件组成，这些组件按照一定的方式相互依赖。在这种集成模式中，可以看出流程组件和流程组件、流程组件和业务组件、表示组件和业务组件、表示组件和流程组件之间，存在着通过对流程方法或业务方法的调用而相互依赖的情形。这种依赖关系通过被调用者组件直接硬编码到调用者组件中来表现，也就是说它们之间没有通过中介，而是以硬编码的方式耦合一起。但是，这种代码侵入性的依赖方式将导致系统的紧耦合，致使系统的可扩性和可维护性降低，组件不能得到较大程度的重用，与“高内聚，松耦合”的软件设计思想相背离。IoC 模式的引入将对这一问题的解决提供一种方案，它的实现能很好地管理组件之间的依赖，进而为工作流管理系统和应用系统的集成提供一种新的方法。

由控制反转(IoC)模式的原理可知，IoC 解决组件依赖的实质是转移依赖。例如，流程组件 A 和业务组件 B 之间的依赖 $A \rightarrow B$，就可以通过引入交互协议 I，转换成$(A \rightarrow I)+(I \leftarrow B)$，同时引入一个装配器 Assembler。当组件 A 要调用组件 B 中的方法时，首先在 A 中声明 I，在运行时 Assembler 根据配置文件或配置代码获取 I 的一个实例 B，并将其注入到 A 中，如图 3 所示。

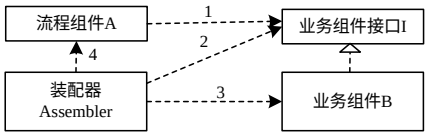

图 3 基于控制反转(IoC)的集成方法

组件在运行期不再主动调用协作组件或协作对象，而是通过装配器将所需的组件或对象在运行期动态地注入到调用者组件中。基于控制反转(IoC)的集成方法明显减少了组件或对象之间的耦合度，避免了大量硬编码，有利于组件重用和系统扩展。

3 系统集成方法

控制反转(IoC)设计模式的实现，一般是指构成框架的核心部件——容器，如 PicoContainer, Jdon, Spring 等。借助它们实现 IoC 模式的框架，简化开发的复杂性，降低难度和减小企业成本。本文采用以 Spring 框架为核心整合表现层框架 Struts 和持久层框架 Hibernate 的 Java EE 轻量级平台，结合嵌入式开源工作流软件 OSWorkflow，提出一种基于嵌入式工作流系统的集成模式的具体方案。同时给予应用验证。

OSWorkflow 是著名开源社区 OpenSymphony 发布的基于有限状态机(Finite State Machine, FSM)理论的工作流引擎。

Java EE 提供了一种企业级的计算模型和运行环境，支持开发和部署多层体系结构的应用。OSWorkflow 是嵌入式工作流引擎，可以看作业务逻辑的一部分，部署在 Java EE 框架中，软件层次如图 4 所示。从图 4 可以看出，OSWorkflow 引擎、流程组件、业务组件放置在 IoC 容器中，并且它们之间没有直接交互，依赖关系在配置文件中进行描述。当组件要调用协作组件时，IoC 容器就根据配置文件，在运行时将其动态注入。系统中工作流引擎和工作流组件被看作普通的组件，工作流系统和应用系统通过 IoC 容器集成在一起，能降低耦合程度，组件也得到一定程度的重用。

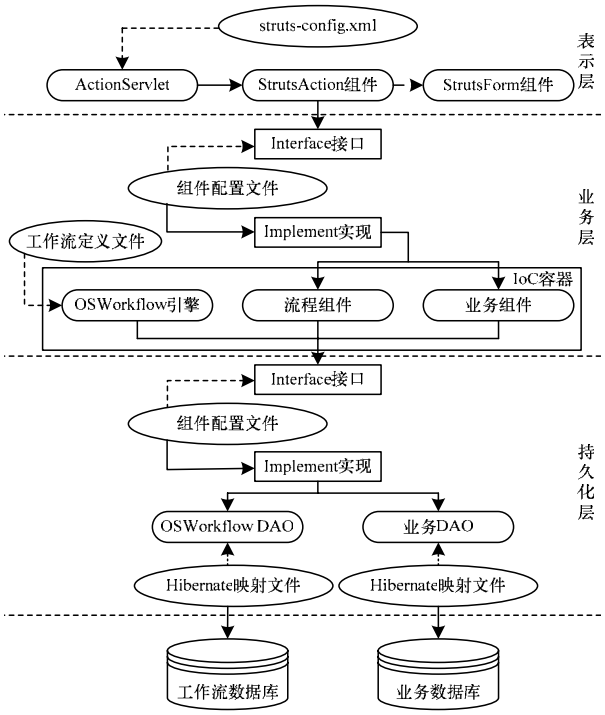

图 4　嵌入工作流引擎的软件层次

在项目中通过采用访问控制技术、工作流技术、组件技术以及数据库技术等，实现诸种类的工作流定义、调整和运行时流转控制。

因为组件的依赖关系的实现是靠 IoC 容器即上级管理者，所以首先要将组件注册到容器中。注册方式采用 XML 描述文件，包括工作流引擎、流程组件、业务组件的注册。

在构建例程中表示层组件的流程逻辑需要使用流程组件和业务组件，但是代码里并没有显式地调用那些组件，而是通过对组件接口的声明，在运行时由容器动态注入，从而避免了硬编码。

4　结束语

在应用系统中引入工作流技术的关键是解决两者的集成问题，对这一问题的解决有多种方式，但核心都是处理两者集成时的耦合度。本文提出的应用控制反转(IoC)模式能较好地解决这一问题，通过 IoC 容器转移组件间的依赖，在运行时再动态注入，并且当需要改变依赖时只需修改依赖关系描述文件，减少了硬编码的数量，极大地提高了组件的重用，使得整个系统的灵活性和可扩性大为提高，是应用系统引入工作流技术的一个有效途径。

参考文献

[1] Workflow Management Coalition. The Workflow Reference Model[Z]. 1995.

[2] Muehlen M Z, Allen R. Workflow Management Coalition Workflow Classification: Embedded & Autonomous Workflow Management Systems[EB/OL]. (2000-04-15). http://www.wfmc.org/standards/docs/MzM_RA_WfMC_WP_embedded_and_Autonomous_Workflow.pdf.

[3] 李　青, 闻敬谦, 赵　萌. 基于 AOP 的工作流系统与企业信息系统集成研究[J]. 计算机集成制造系统, 2006, 12(3): 401-406.

[4] 何建校. 基于 J2EE 的嵌入式工作流引擎[J]. 微计算机信息, 2006, 22(7): 28-30.

[5] 娄　锋, 孙　涌. 轻量级 IoC 容器的研究与设计[J]. 计算机技术与发展, 2007, 17(1): 91-93.

编辑　任吉慧

(上接第 68 页)

4.2　实验结果

应用 NS 网络模拟 DiffServ 网络，表 2 是仅加入带宽代理的 DiffServ 网络的实验结果，表 3 是使用 DRMA 方法的实验结果。其中，*TotPkts*, *TxtPkts*, *Ldrops* 和 *Edrops* 分别代表数据传输的总包数、实际传输的包数、早期丢包数和晚期丢包数，这几个参数也是衡量算法优劣的依据。

表 2　未使用 DRMA 的 DiffServ 网络实验结果

DSCP	*TotPkts*	*TxtPkts*	*Ldrops*	*Edrops*
10	1 342	1 310	0	0
11	1 745	1 554	191	11
12	25 312	23 928	1 153	231
14	179	170	9	0
总计	28 573	26 962	1 353	242

表 3　使用 DRMA 的 DiffServ 网络实验结果

DSCP	*TotPkts*	*TxtPkts*	*Ldrops*	*Edrops*
10	1 289	1 289	0	0
11	1 982	1 982	0	0
12	30 893	29 923	868	102
13	12	12	0	0
14	306	305	1	0
15	31	31	0	0
总计	34 513	33 542	869	102

实验结果表明，加入 DRMA 算法后网络接收数据的能力得到加强，即在相同的时间内接收的数据包数量增加，丢包率减小，这就提高了网络的资源管理效率。

5　结束语

本文提出一种动态资源管理算法，改善网络的管理效率，保证数据流服务质量。采用动态的资源管理方式，在边界路由器上实时地测量网络的传输速率，更合理地分配带宽资源，减少带宽资源浪费。在区分 DiffServ 网络中，带宽代理的加入使网络处理数据的能力不断增强。今后的工作将研究接纳控制方法，以对不同类型的预留请求进行有效的接纳控制，从而进一步改善网络服务质量。

参考文献

[1] Xiao Xipeng. Internet QoS: A Big Picture[J]. IEEE Network, 1999, 30(2): 8-18.

[2] 邱　瑜, 朱淼良. DiffServ 中带宽代理 BB 的实现[J]. 计算机科学, 2003, 30(4): 47-50.

[3] Mantar H. A Scalable Model for Inter-bandwidth Broker Resource Reservation and Provisioning[J]. IEEE Journal on Selected Areas in Communications, 2004, 22(10): 2019-2034.

[4] Stamos K. An Adaptive Admission Control Algorithm for Bandwidth Broker[C]//Proc. of the 3rd IEEE Int'l Symp. on Network Computing and Applications. Cambridge, BA, USA: [s. n.], 2004.

[5] 俞承志, 宋瀚涛, 宁先圣, 等. 区分服务网络中立即保留请求的接纳控制[J]. 计算机集成制造系统, 2005, 11(12): 1790-1796.

编辑　陈　文