# 面向工作流的分析、设计与开发

侯 潇，高延武
邢台学院信息科学与技术系，河北邢台（054000）
E-mail: houxiao86@yahoo.com.cn

**摘　要：**应用工作流技术分离传统企业级应用系统开发中的流程需求和非流程需求。通过对WfMC 工作流参考模型的研究，提出工作流三个不同层次的定义以及基于此对复杂工作流分解的实践参考。基于工作流驱动思想给出一种设计方案，采用此方案可以使工作流管理系统的表现层和业务层更加统一。

**关键字：**工作流技术；工作流管理系统；jBPM；工作流驱动；工作流变更
**中图分类号：**TP311

## 1 引言

工作流（Workflow），是对工作流程及其各操作步骤之间业务规则的抽象、概括、描述。工作流要解决的主要问题是：为实现某个业务目标，在多个参与者之间，利用计算机，按某种预定规则自动传递文档、信息或者任务。工作流管理系统（Workflow Management System, WfMS）的主要功能是通过计算机技术的支持去定义、执行和管理工作流，协调工作流执行过程中工作之间以及群体成员之间的信息交互。工作流需要依靠工作流管理系统来实现[1]。

业界的工作流主要遵循两个标准：WfMC 和 BPEL。WfMC 的全称为 Workflow Management Coalition（工作流管理联盟），此组织于 1993 年成立，并于 1994 年制定了一系列工作流参考模型的工业标准[2]。BPEL 全称为 Business Process Execution Language（业务流执行语言），是由 IBM，Microsoft 等公司于 2002 年联合发布的标准，他们的工作流产品也都实现了 BPEL 标准[3]。目前国内采用 WfMC 标准的比较多。

国外各大软件公司，如 IBM，Microsoft，Oracle 和 JBoss 等，都有工作流管理系统或工作流引擎。包括 IBM(c) Lotus Workflow，Microsoft(c) Windows Workflow Foundation，Oracle(c) BPEL 流程管理器，以及 JBoss(c) jBPM。本课题采用的是 JBoss 公司的 jPDL（JBoss jBPM Process Definition Language）流程语言，其已经实现的产品是 jBPM。jPDL 相比其他工作流语言是轻量级的，因此更加简单易读。

目前不少 ERP（企业资源计划）、CRM（客户关系管理）和 OA（办公自动化）系统已经引入了工作流技术，但大多数应用系统仍然是直接基于数据模型开发的，具有流程特性的需求也在设计阶段分散在业务逻辑当中，没有提取出来。工作流可以规范业务参与者的信息交互行为，为企业级应用引入工作流技术比直接采用数据模型开发更便于业务逻辑整合，使数据模型之外的规则包含在业务逻辑而非表现层当中。业务层之上的功能，例如表现层，可以设计的更加松散，降低其耦合性，便于调整。本课题重点探索复杂工作流的分解，以及研究工作流和其他模块的交互方式。

## 2 工作流模型

根据 WfMC 工作流参考模型（Workflow Reference Model）的定义，工作流管理系统包括以下几个部分[1]：

- **流程定义工具**（Process Definition Tool），是创建流程定义的工具，通常作为工作流开发环境的一部分被提供。

- **流程定义**（Process Definition），是一个工作流从开始到结束除数据模型之外的所有必要信息，包括所有步骤（Step）以及步骤之间的流转（Transation）和步骤中的动作（Action）。
- **工作流表现服务**（Workflow Enactment Service），通常为工作流引擎（Workflow Engine），解释流程定义并为用户的工作列表添加工作项（Work Item）或者调用其他服务。
- **工作流相关数据和应用程序数据**（Workflow Relevant Data and Application Data），通常为数据模型。
- 除此之外还有**工作列表**（Work Lists），**工作列表处理程序和用户界面**（Worklist Handler & User Interface），**监控操作**（Supervisory Operations），以及**暴露和内涵的接口**（Exposed and Embedded Interfaces）。

工作流引擎的工作方式通常被理解为它是一种有限状态自动机（Finite State Machine，简称为 FSM）。有限状态机又称有限状态自动机或简称状态机，是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型[4]。

工作流定义中，FSM 的状态对应工作流定义中各种不同功能的步骤。jPDL 是一种基于 XML 的流程定义语言其中常见的步骤有：开始（start）、状态（state，需要主动触发才能继续执行的步骤）、节点（node，不需要主动触发而自动执行的步骤）、任务（task）、判断（decision）以及分支（fork）和汇合（join）等[5]。

JBoss(c) jBPM 为开发人员提供了流程定义工具、流程定义、工作流引擎和工作列表等功能。针对本课题应为客户提供应用程序数据（数据模型）、工作列表处理程序以及用户界面。

# 3 面向工作流的分析

毕业设计(论文)的评审是本科教学工作水平评估的重要环节。大多数高校已规范化的落实了学生毕业设计的完整流程。目前毕业设计中的大量文档、资料采用人工传递的方式，效率比较低，容易出错，整个毕业设计流程控制也由人工监督完成。鉴于此现状，提出、设计并实施一套完整的自动化毕业设计管理系统将有助于提升毕业设计管理的规范化程度。

## 3.1 获取流程需求

在需求分析阶段应尽可能全面而非过分深入的获取客户需求。需求分析过程除了获得业务数据需求和功能需求之外还可能涉及到具有流程特征的需求，这会有 3 种可能：需求不具有流程性或客户不希望采用工作流技术；需求不包含明显的工作流，但可以通过工作流技术改进目标系统的开发；客户已经实施工作流但希望改进或重新实施。针对以上 3 种情况应采取不同的解决方案：

- 对于不具有流程性的系统，采用常规的方案归纳数据模型，设计并实施业务逻辑，以及开发表示层；
- 对于工作流不明显的任务需要做工作流挖掘（Workflow Mining），获取客户的业务流程，归纳并优化得出可以实施的工作流方案，最终将工作流和数据模型进行整合。
- 如果客户已经采用人工控制的工作流或者希望对现有工作流管理系统进行改进、重用，可以对人工工作流的参与过程进行建模或者在遗留系统基础上优化、重用。

根据工作流所实现的业务过程不同，工作流管理系统可以分为 4 类：管理型工作流、设定型工作流、协作型工作流、生产型工作流[6]。在需求分析阶段，针对每种不同的种类应该向客户了解工作流所涉及的数据模型或（和）文档系统等非工作流资源。这一重要的需求可

能直接影响系统的架构设计。

正确的区分流程需求（面向数据和流程）和非流程需求（面向数据）是开发工作流管理系统在需求分析阶段的关键问题。通过需求分析获取了我校毕业设计的完整流程图，它包含分布在 4 个泳道（角色）中的 44 个步骤。毕业设计工作流涵盖毕业设计初始化、课题评审、学生选题、下达任务书、提交开题报告、中期指导检查、提交成果和论文、答辩，直到最终工作结束验收。同时，通过查阅当前正在使用的各种表格、文档获得了各个工作流参与者、课题等实体信息及其基本关系。因为客户已经采用人工控制的工作流，所以对于此类需求只需进行调整和优化，使其既能满足客户要求，又适合在工作流开发工具或工作流引擎上实施。

### 3.2 工作流分解

一个工作流管理系统通常包含 3 个不同层次的工作流定义，即全局工作流、局部工作流和工作流实例。事实上，在 JBoss(c) jBPM 4.x 中抽象工作流称为“流程定义（Process Definition）”，工作流实例称为“流程实例（Process Instance）”，工作流实例的当前运行状态称为“执行（Execution）”。

当系统边界划定完成之后，整个系统拥有一个全局工作流，例如，毕业设计管理工作流、企业帐务审批工作流等。对于较为复杂的系统（例如 ERP 系统中可能拥有上百个步骤的工作流），一个全局工作流中可能包含或者可以拆分为多个局部工作流，它们之间通常是可以并发或者通过较为松散的关系交织在一起。局部工作流和全局工作流都是抽象的，用来描述其参与者的行为以及交互时序。局部工作流对某一组参与者运行起来就形成了这个局部工作流的工作流实例，即一个具有状态的任务序列。

根据全局工作流的复杂程度决定是否将其拆分为若干个局部工作流。另外如果全局工作流的某个局部在执行上具有相对独立性或者在功能上比较独立，也应该考虑分离出来形成局部工作流。适当的分解有利于模块化设计和实施，以及在系统维护阶段对工作流进行优化，甚至可以更好的应对需求变更和重构。根据本课题开发经验一般可以将局部工作流中的步骤划分在 10 个左右，但这种划分必须保证局部工作流具有相对独立性，并且可以在全局工作流中以子流程的形式表示出来。

基于工作流分解的思想，将本课题中的毕业设计工作流按其功能拆分成了 4 个小的工作流，分别是：总体进度控制、教师发布毕业设计课题、学生申请毕业设计课题、学生参与毕业设计。其中第一个是全局工作流，后三个是局部工作流。显然 3 个局部工作流是可以在全局中并发进行的，比如一旦有毕业设计课题通过审核（处于教室发布毕业设计课题），学生就可以申请之。总体进度控制并不需要产生工作流实例，只是对于管理起到“开关”的作用。在完成拆分后，应采用需求审核的方法检查结果是否仍然符合需求。

## 4 面向工作流的设计

### 4.1 框架技术

本课题的工作流引擎采用开放源代码的 JBoss(c) jBPM，它是 JBoss(c)企业框架（JBoss Enterprise Framework）的组成部分，可以实现可扩展的灵活的工作流，提供基本的工作流程管理功能以及流程编排。JBoss(c) jBPM 使得企业能够创建并且自动化工作流，通过工作流协调工作人员、应用程序和服务[7]。

JBoss(c) jBPM 的特色之一是它同时支持三种工作流定义语言：jPDL、BPEL 和 Pageflow，并且将这三种语言自然的融合在流程虚拟机（Process Virtual Machine，PVM）技术上面[7]。

另外 jBPM 采用 Hibernate 作为数据持久化底层。jBPM 是轻量级的工作流引擎，开发人员通过较少的投入就能获得非常理想的效果，但是它起步时间较晚仍有诸多不足之处。jBPM 的开发工具 GPD（Graphical Process Designer）只提供了 Eclipse 的插件，并且要在运行期间进行工作流变更，也只能返回给开发人员进行。由于 jBPM 的每个版本通常会有较大的变化，比如当前 4.x 版本几乎不兼容 3.x 版本，因此可以考虑对其进行适当的封装，避免将来升级时出现不兼容的问题。

尽管 jBPM 还有不足之处，但其简单实用的宗旨具有很大的吸引力。jBPM 封装性非常好，在实际开发中只需从业务角度控制流程导向（Workflow-oriented），而不需要考虑实现细节。jBPM 已经具有了针对工作流的权限控制功能（Identity 模块），这一功能基于 RBAC-2 思想实现动态的用户授权。

表现层采用 MVC 模式（Model-View-Controller Pattern），以便彻底的分离视图（由美工人员实施）和控制逻辑（由编码人员实施）。Java Web 表示层 MVC 的实现非常多，最著名的是 Apache Struts 以及考虑周全功能强大的 Apache Tapestry，另外还有基于 SUN(c) JSF 标准的诸多产品。本课题在开发阶段决定采用 BeanStruts 作为表现层框架。BeanStruts 是一个基于 MVC 的 Java Web 表现层框架，它的特别之处是基于表单的事件驱动（Event-driven）交互方式，除此之外它还支持资源本地化（Resource Localization）、依赖注入（Dependency Injection）以及权限控制等功能[8]。

## 4.2 整体架构设计

在设计阶段可以把工作流以及工作流引擎放在不同的位置，进而产生不同的作用效果以及对其他组件的影响。下面给出两种参考方案（如图 1，图 2），在实际开发中视具体情况选择。

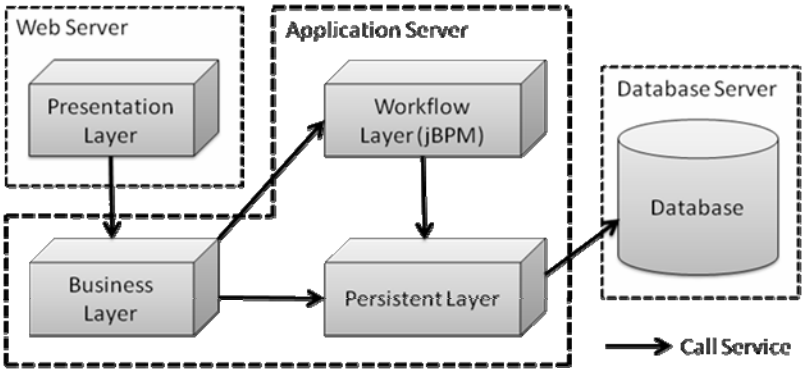


图 1，架构方案 1

Figure 1, Architecture 1

方案 1 遵循传统的 3 层架构模型，只不过业务层（Business Layer）在访问持久层（Persistent Layer）的同时还需要访问工作流层（Workflow Layer）。这种架构表达了以业务为中心的思想，传统容易实现。但是工作流加重了业务层的负担，并且可能不利于表现层、业务层和工作流层的相对独立。对于中小规模应用可以将工作流层和持久层封装在业务层，形成典型的 3 层架构模型。

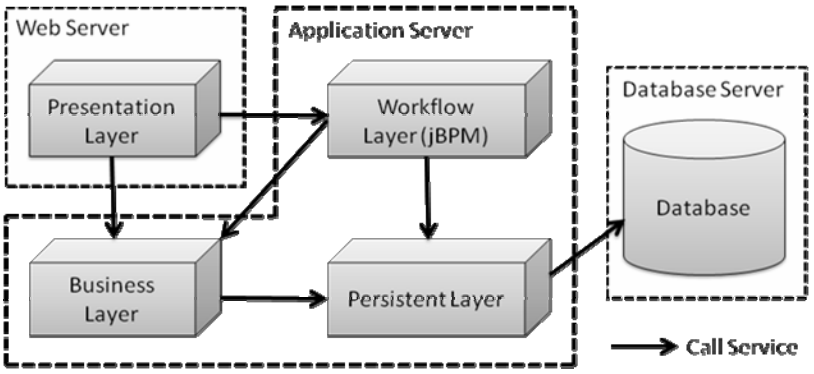


图 2，架构方案 2

Figure 2, Architecture 2

方案 2 表达完全不同的思想，即以工作流为中心的思想。对于非工作流需求（仅面向数据模型）访问业务层；而对于面向数据流的需求则可能需要同时访问工作流层和业务层，这取决于工作流是否包含对数据模型的访问。采用这种架构的工作流容易变更，业务层处于底层服务的位置因此更加独立，仅依赖于持久层。但这种方案增加了设计的难度，同时对于编码人员要求较高。对于中小规模应用可以将业务层和持久层绑定在一起实现以降低开发难度。

## 4.3 工作流层细节设计

基于方案 2（如图 2）可以采用工作流驱动的方式实现表现层和工作流层的交互，即表现层调用工作流层的服务方法，当此方法完成相应的处理之后返回当前流程处在的位置（可以是一个流程步骤名称），表现层根据此位置返回给用户相应页面。如图 3为工作流驱动的时序图。

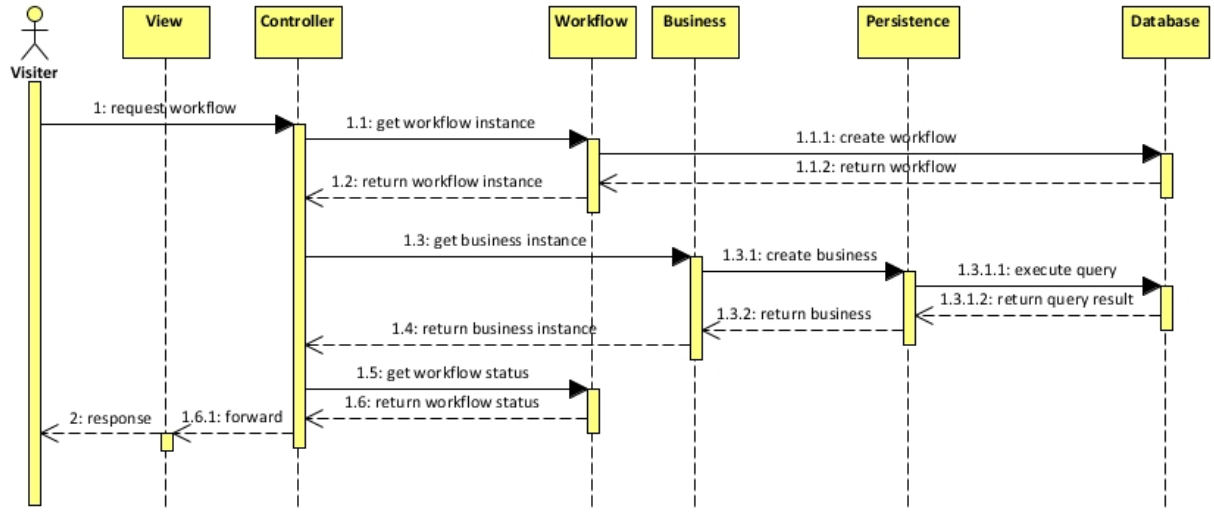


图 3，工作流驱动时序图

Figure 3, Workflow driven sequence

采用工作流驱动时应确保工作流层可以访问业务层，反之不能。工作流层可以通过访问业务层使得工作流对数据模型有控制作用，或者根据数据改变工作流的执行状态；反之，两个层次之间的相互访问会增加组件之间的耦合性，并且给工作流变更带来困难。

基于工作流驱动的思想，表现层和业务逻辑的交互设计为如图 4（已简化）所示的方案。

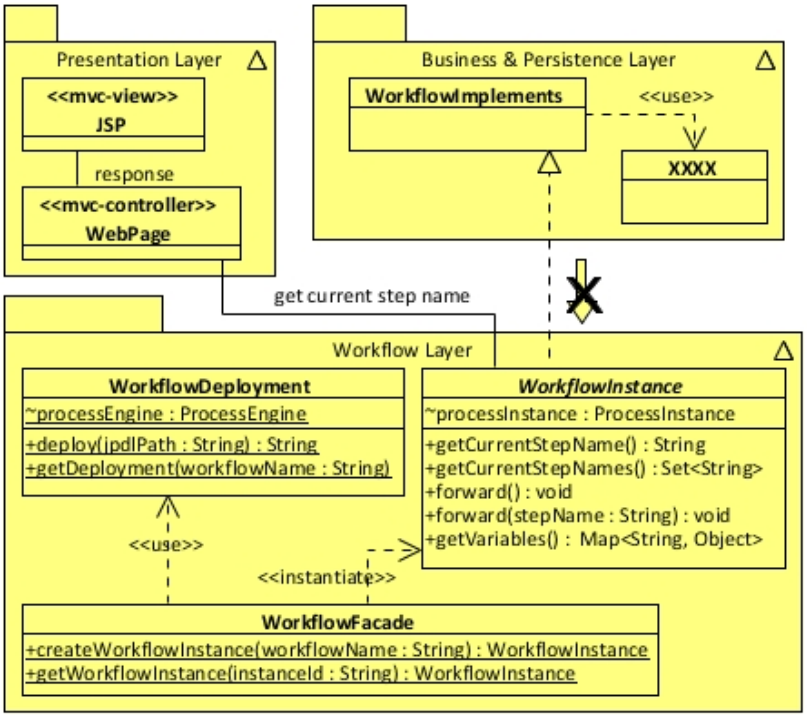


图 4，工作流层设计

Figure 4, Workflow Layer Design

**WebPage** 是 BeanStruts 的页面控制器基类，这里表示当工作流进度需要转换时，在当前工作流页面的控制器中设置工作流参数（通过 getVariables()），执行 forward()，最后获取工作流当前进度（getCurrentNodeName()），并以进度名通过 BeanStruts 的页面导航映射转到相应的工作流状态页面。**WorkflowDeployment** 是工作流部署工厂类，WorkflowFacade 用它来部署工作流。**WorkflowFacade** 是工作流管理封装类。在各种需要创建、获取工作流实例的位置使用。例如当学生申请课题时，学生的申请操作将创建一个“学生申请毕业设计课题”实例。**WorkflowInstance** 是工作流实例的接口。使用此接口可以更好的应对工作流变更以及增加、删除局部工作流。**WorkflowImplement** 实现了 WorkflowInstance 接口，它是局部工作流的具体实现，它完成相应工作流的逻辑功能并且操作底层的业务逻辑和持久化。

### 4.4 权限控制

尽管 jBPM 提供了权限控制方案，但为降低学习负担，在实际开发中采用 BeanStruts 的基于 RBAC-1[8]的实现。RBAC-1 与 RBAC-2 是根据不同思想实现的基于角色的访问控制（RBAC，Role-based Access Control），总的来说前者更注重角色的继承，后者更注重权限分配的灵活性[9]。

## 5 面向工作流的开发

根据极限编程（Extreme Programming，简称 XP，是一种敏捷开发过程）的基本思想[10]，在整个软件开发生命周期（SDLC）都应遵循“辩证的简单”的原则。极限编程强调尽可能少的考虑暂时并不重要的实现，但这不意味着不能适应需求变更，而是采用类似于接口技术、依赖注入的方法将功能与实现分离。因为工作流不仅涉及其本身而且渗透到业务逻辑，所以针对工作流的需求变更有其特殊的复杂性。因此弱化架构设计，将其推迟到实施阶段是非常必要的。处于设计阶段任务应使用需求审查，确保设计结果是符合需求并且易于扩展的；处

于实施阶段的任务应及时进行单元测试（Unit Testing），这样可以确保不会在开发过程的最后出现 Bug 危机。

## 5.1 编码与测试

以“教师发布毕业设计课题”工作流中的“创建项目”页面控制器类为例给出工作流驱动的编码方法（见表格 1）。其中的 4 个 Java 元注释（Java Annotation）是 BeanStruts 提供的注入服务（Inject Service）和事件回调（Event Callback）。对于采用其他框架的设计方案（例如 Spring Framework 等），应根据具体情况编码。

表格 1，工作流驱动的 MVC 控制器

Table 1, Workflow-driven Controller of MVC

```
public class CreateProjectPage extends WebPage {
    // 持久化 DAO 实例
    @InjectInstance
    private IEntityDao entityDaoBean;
    // TeacherReleaseProjects 工作流实例
    @InjectInstance
    private ITeacherReleaseProjects teacherReleaseProjectsBean;
    // Session 中的用户 ID
    @InjectContextAttribute(scope = ContextAttributeScope.SESSION, name = "account.accountID")
    private int accountID;

    // 创建项目事件
    @EventCallback(eventName = "event_SaveProject")
    public void saveProject_click() {
        Project project = (Project) getRequest().getAttribute("project"); // 获取 jsp 页面表单信息
        Teacher teacher = entityDaoBean.find(Teacher.class, accountID);
        project = teacherReleaseProjectsBean.createProject(teacher, project);
        String workflowStatus = project.getWorkflowStatus();
        redirectMapping(workflowStatus); // 导航到新的工作流状态页面
    }
}
```

除设计阶段进行的需求审核之外，对于编码的黑盒测试（Black-box Testing）也是有必要的。对于工作流的黑盒测试用例（Testing Cases）应能尽可能根据需求覆盖一个局部工作流的各种可能。由于对工作流进行的分解，做到全面测试是可以实现的。目前 BeanStruts 没有提供其相应的单元测试（白盒测试，White-box Testing）方案，只能使用代码走查（Code Walkthrough）、黑盒测试等方法确保没有 Bug。当然也可以通过临时修改页面控制器类中属性值，使用现有的测试工具进行测试。

## 5.2 工作流监控与变更

jBPM 为工作流监控提供了必要的支持。在 jBPM 3.x 版本中提供了 jBPM web console，用这个网站可以管理工作流和工作流实例，但目前 4.x 版本还没有提供相应工具。使用 HistoryService 可以查阅正在执行或已经停止的工作流实例历史信息，或者使用

HistorySession 为工作流添加用户自定义的历史记录功能[5]。另外，jBPM 的流程定义文件（*.jpdl.xml）中的每个步骤都保留了它在设计时的位置坐标，可以设计出可视化的流程监控界面为用户呈现出工作流执行过程。

在做工作流变更时应慎重考虑其必要性、可行性和对现有系统影响的风险。总的来说应把握的基本原则是对于已经完成的工作流实例不做改动，对于在变更点之前正在执行的工作流不做改动，而对于在变更点之后正在执行的工作流予以终止（如果终止的代价较低）或者重新填充所涉及用户工作列表（如果终止的代价较高）。另外，必须考虑工作流引擎的实际情况，对工作流变更支持较好的工作流引擎可能已经为工作流变更做了充分的考虑，但大多数情况下需要开发人员编码解决。采用工作流驱动方案设计的系统在应对工作流变更时应考虑一下 2 种情况：增加、删除新的局部工作流可能会影响已经实例化的全局工作流和新增局部工作流在全局工作流插入点前后（如果存在）的局部工作流。对于后一种情况应注意工作流之间的衔接，包括数据格式是否一致、流程链是否仍然连续（类似向链表中插入节点）。现有局部工作流增加、删除步骤可能会影响相邻局部工作流的衔接和处理用户工作列表的问题。工作流驱动的设计方案可以尽可能减少修改已实现的表现层页面。在完成工作流变更后需要添加或修改表现层相关的页面，确保呈现数据正确以及流程链连续。

## 6 总结

工作流技术在其诞生的 40 年左右的时间中已经有了卓越的发展：工作流技术建立在数学模型（如 Petri 网）和自动化控制理论（如 FSM）基础上，其应用已经深入到办公自动化等多个领域。但处于起步阶段的工作流技术仍然有许多迫切需要解决的问题。目前仍然没有统一的流程定义标准，因此基于众多流程定义标准交集推行一种标准，可能是较为可行的有益于指导工作流技术发展的方案[11]。

实际开发工作流管理系统时，采用工作流分解技术比直接把工作流定义在需求上更安全易行。目前工作流分解面临的主要问题是很难运用现有理论（例如 ECA 规则，Event-Condition-Action[12]）指导企业级应用开发，因此只能凭借经验或参照软件工程相关理论进行。文中给出的工作流 3 层次定义能有效的指导工作流分解，以及对分解结果进行审核。

工作流驱动的设计方案充分肯定了工作流技术在系统中的地位，采用这种方案有助于以工作流为指导统一和规范系统的整体设计。而设计和开发思想的转变可能是大多数团队要面对的问题。

在本课题实际开发中，没有实施对事务的支持，最明显的可能产生数据不一致性的地方是操作数据模型和工作流两个独立的过程。对于要求严格的系统（例如金融应用）必须确保前面两个操作属于一个事务。

### 参考文献

[1] Hollingsworth David. Workflow Management Coalition The Workflow Reference Model[M/OL]. http://www.wfmc.org/standards/docs/tc003v11.pdf.
[2] Wikipedia. Workflow[EB/OL]. http://en.wikipedia.org/wiki/Workflow.
[3] 高杰. 深入浅出 jBPM[M]. 人民邮电出版社, 2009.
[4] Wikipedia. Finite-state machine[EB/OL]. http://en.wikipedia.org/wiki/Finite-state_machine.
[5] Red Hat Inc. jBPM User Guide[EB/OL]. http://docs.jboss.org/jbpm/v4/userguide/html_single/.
[6] 葛志春. 工作流管理技术介绍[EB/OL]. http://cio.csai.cn/ep/200608301722481558.htm.
[7] Red Hat Inc. JBoss jBPM[CP/OL]. http://www.jboss.com/products/jbpm/.
[8] IDreamer. BeanStruts[CP/OL]. https://code.google.com/p/beanstruts/.

[9] Sandhu Ravi S., Coynek Edward J., Feinsteink Hal L., 等. Role-Based Access Control Models[J]. IEEE Computer, 1996,29(2):38-47.
[10] Cunningham Ward. Extreme Programming[EB/OL]. http://www.c2.com/cgi/wiki?ExtremeProgramming.
[11] 刘怡, 张子刚, 张戡. 工作流模型研究述评[J]. 计算机工程与设计, 2007:448-451.
[12] Alferes J. J., Banti F., Brogi A. An Event-Condition-Action Logic Programming Language[M/OL]. http://www.di.unipi.it/~brogi/papers/JELIA06.pdf.

# Workflow-oriented Analyze, Design and Develop

HOU Xiao, GAO Yanwu
Department of Information Science and Technology, Xingtai University (054000)

**Abstract**

The usage of workflow can detach process-oriented and non-process-oriented requirements in traditional enterprise application systems. Backgrounding the study of WfMC Workflow Reference Model, gives three meanings of workflow in different levels and provides practice for complicated workflow break-up based on it. A workflow-driven design is presented in order to integrate presentation layer and business layer of WfMS.

**Keywords:** Workflow; WfMS; jBPM; Workflow-driven; Workflow change