# 摘 要

工作流技术是实现企业业务过程重组、过程管理的企业信息化技术，最初用于生产过程及办公自动化领域，后来在企业的业务流程管理领域得到充分的应用。虽然工作流产品和技术日新月异，但是就目前的工作流技术而言，仍然有一些明显的缺陷，主要缺陷在于：异种工作流产品之间的缺乏交互机制，工作流建模技术不够成熟，数据规范性得不到统一。

工作流编辑作为工作流管理技术的基础，它对企业生产过程进行建模，并把这些模型描述成计算机可以处理的语句。工作流编辑工具就是这样一个软件包，包含了图形化定义工具，用户通过工作流编辑工具实现企业业务过程定义、过程重组和过程管理。

本文从理论和实践两个方面对基于 BPEL4WS 的工作流管理技术进行了较为深入的研究和探索，基于 Eclipse 框架设计与实现了可视化工作流编辑器。其中作者主要的工作如下：

本文首先介绍了工作流的基本概念、工作流管理系统及 WebServices。接着对 BPEL4WS 和基于服务的流程描述语言进行研究、分析和比较。重点分析和研究了 Web Service 技术和基于 Web Service 定义的商业流程执行语言 BPEL4WS。

然后分析了 Eclipse 的体系结构及如何利用 GEF 来实现图形库。接着对基于 BPEL4ws 的工作流模型进行分析和属性设计。设计并实现了支持 BPEL4ws 工作流流程定义的图形库。

接着分析了基于 BPEL4ws 的工作流编辑器的功能需求，并利用 Eclipse 框架，设计并实现了可视化工作流编辑器。该编辑器支持基于 BPEL4ws 的流程描述和流程仿真，并能自动验证 BPEL4ws 表达式的规范性。

最后通过具体的应用案例演示了本系统的工作过程，证明了本系统的实用性和易用性。测试结果表明：基于 Eclipse 的可视化工作流编辑器设计出的流程可以满足企业级业务需求，可以通过可视化的方式定制复杂的商业逻辑，可以进行流程实例的创建和启动，还可以将图形化流程描述自动转换为 BPEL 和 workflow 文件，并对流程定义进行有效性检查。

关键词：工作流，工作流建模，流程编辑，Eclipse ， BPEL4ws

# ABSTRACT

As the new technology of enterprise information, workflow technology originated in the field of production process and office automation.When applied to manage business process, workflow technology gets more and more attention.With the developing of workflow technology, workflow products and technology keep growing and transforming. However,there are a lot of obvious defects in workflow technology currently available.It is manifested that different workflow product provided by different developer can not be inter-operated,with lack of supporting to distributing environment with different architecture,the immaturity of workflow modeling technology and shortage of an uniform specification.

As the basic of Workflow Management technology, the basic requirement for workflow editing including workflow modeling is that the processes are modeled in a simple and virtual way.The workflow editing tool is one such package including the definition of the graphical tools, and users use workflow editing tools to achieve enterprise business process definition, process reengineering and process management.

After the deep research and study of the present workflow technologies based on BPEL4ws,the author designs and implements a workflow editing system based on Eclipse.The main works done by the author are listed below:

Firstly, the paper introduces the basic concept of workflow, workflow managementsystem and WebServices.Then it analyses BPEL4WS and flow description language based on servevice. The focus of analysis and research is web service technology and business process execution language based on web service definition.

Secondly, the paper analyses the architecture of Eclipse and how to use GEF to implement a graphics library .Then it analyses workflow model and designs the properties of model.After that, a graphics library based on BPEL4ws is design and implemented to support workflow process definition.

Thirdly, a Workflow editor based on Eclipse is design and implemented after analysis of functional requirements Workflow editor based on BPEL4ws.The workflow editor support flow description, flow simulation and automatic verification of BPEL

expression.

Finally, an application case is presented to show how to use the workflow editor.It is shown the system is pratical and convenient.Test results show that the workflow editor based Eclipse can design workflow process to meet the needs of enterprise-class, finish creation and end of flow examples, transform graphical description of flow to BPEL and workflow files and check the workflow definition.

**Keyword:** Workflow, Workflow Modeling, Workflow Edtion, Eclipse, BPEL4ws.

# 缩略词

| | |
|---|---|
| WfMS | 工作流管理系统 |
| Wf4EIB | 企业流程综合业务 |
| WfPD | 工作流设计器 |
| WfMC | 工作流管理联盟 |
| WSDL | Web 服务描述语言 |
| WSFL | Web 服务流程描述语言 |
| BPEL | 业务流程执行语言 |
| BPEL4WS | 基于 Web 服务的业务流程执行语言 |
| BPML | 业务流程建模语言 |
| BPMI | 业务流程管理协议 |
| BPSS | 业务过程规范模式 |
| B/S | 浏览器/服务器 |
| CORBA | 公共对象请求代理体系 |
| C/S | 客户端/服务器 |
| MVC | 模型/视图/控制器 |
| UDDI | 通用描述发现与集成 |
| GEF | 图形编辑器框架 |
| VCM | 版本与配置管理 |
| BPR | 业务流程再造 |

# 独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：石一峰　　日期：　　年　月　日

# 关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：石一峰　　导师签名：谭浩

日期：　　年　月　日

# 第一章 绪论

工作流技术是实现企业业务过程重组、过程管理的企业信息化技术，最初用于生产过程及办公自动化领域，后来在企业的业务流程管理领域得到充分的应用。工作流技术标准制定的组织—工作流管理联盟于1993年成立。自此，工作流的技术研究以及相关产品的开发进入了繁荣的阶段。虽然工作流产品和技术日新月异，但是就目前的工作流技术而言，仍然有一些明显的缺陷，主要缺陷在于：异种工作流产品之间的缺乏交互机制，工作流建模技术不够成熟，数据规范性得不到统一。

目前的工作流系统存在的最大问题是互联网上的互操作问题，虽然一些分布式中间件如 DCOM、CORBA 在一定程度上解决了上述问题，但是随着互联网的发展，基于 Web 的分布式应用越来越广泛，应用领域越来越深入，互操作带来了许多新的难题，而如何解决这些问题成为影响企业自动化高速发展的关键因素。在这样的背景下，WebService 的出现解决工作流发展的最大难题。因为 WebService 是基于 XML 的使用标准化协议的服务架构，可以很方便地实现互联网上跨平台、语言独立、松散耦合的异构应用的交互和集成。

## 1.1 工作流技术的发展概况及现状

### 1.1.1 国内外研究现状

工作流技术在国外得到较快的发展，成立于 1993 年的工作流管理联盟(Workflow Management Coalition)，是一个非赢利性的国际性的标准化组织。其成员主要包括工作流产品的供应者、应用者，以及有关研究机构和个人。该联盟颁布了一系列工作流产品标准，包括工作流参考模型、工作流管理系统各部分间接口规格、工作流产品的互操作性标准等，这些举措加速了工作流技术的商品化。

尽管供应商很多，然而实际应用的工作流系统的数量却很有限。这主要由于以下几个原因。首先，这门技术还很年轻，所以系统开发商并没有认识到工作流管理技术的潜能。同时某些工作流管理系统还不成熟，其功能有限且可靠性不能让人满意。而且现在为某一特定应用对工作流系统进行改造并不容易。最后，尽

管工作流管理联盟做出了大量的努力，但系统链接和功能方面的标准还能匮乏。例如，许多工作流管理系统使用一种自己特有的图形技术来定义过程 。这样做的缺点是，使得在不同的供应商的系统间交换过程描述变得非常困难。

为了对当代的工作流管理系统有一定的印象，下面介绍市场上的 3 种主流产品：Staffware、COSA 和 ActionWorkflow。Stafffware 是领先的工作流产品之一，占据大约 25%的市场份额。因此它可以作为展示当前的工作流管理系统能力的典范。COSA 和 ActionWorkflow 各具特色，一定程度上体现了工作流管理系统应用的极限。COSA 具有管理复杂业务过程的广泛潜能，而且相当健壮。ActionWorkflow 表现了一种完全不同的方法，它注重团队协作而不是过程管理。

相对国外而言，国内对于工作流技术的研究起步较晚，虽然有许多研究机构和部分公司在进行工作流产品的研究和开发，但是由于没有形成统一的联盟和标准组织，产品和成果相对较少。目前国内市场上的工作流产品还是处于萌芽阶段，其可靠性、开放性、稳定性及标准化方面都还有待加强。

### 1.1.2 工作流发展阶段

从工作流的应用范围、功能的强弱，可以把它划分为三个阶段：

(1). 电子数据流阶段。工作流管理系统的初期阶段，功能简单，适用性不强，多用于单元批量计算的领域，如统计帐目。

(2). 事务处理流阶段。工作流管理系统的发展阶段，功能由单一的计算统计扩展到企业的局部业务管理，如进帐和出纳。

(3). 信息管理流阶段，工作流管理系统的高级阶段，通过全局地把握企业业务流程，设计出一套自动化的流程运转系统，可以极大地提高企业生产效率。

### 1.1.3 工作流技术研究热点

工作流技术的发展，经过十几年的努力，取得一定的结果。近年来，在工作流理论与实施技术方面，研究的热点包括：

- 工作流过程建模技术：包括工作流过程模型和过程描述语言方面的研究，前者是对业务流程的计算化描述，概括了实现流程所需的各种必要信息：如流程开始和结束条件、组成该过程的各个步骤。
- 工作流的分析技术：为了判定业务过程是否在定量（完成时间、资源利用率等）和定性(正确性)方面满足要求，就必须对其进行分析。在现有过程被改进后，

也应该在改变实施前对修改后的过程进行分析。为了进行分析，可以使用仿真和一些形式化验证技术。 这些能力的进一步扩展显然是未来的发展方向。

- 工作流的规划技术：现代的工作流管理系统，在给任务分配资源和决定使用相同资源的多个任务按什么次序执行方面能力有限。现有系统没有足够地重视在安排人力资源时可能发生的“时间表问题”。而且由于劳动力灵活度的提高及组织业务时间的延长，这个问题会越来越突出。这些功能是必须的 ，尽管目前还没有工作流管理系统对此给予支持。
- 分布式工作流技术：现有的工作流管理系统多数面向某个独立组织的业务过程，工作环境单一，而随着互联网的应用，以后的工作流系统将向跨网络、分布式运作的方向发展。
- 丰富工作流模型的表达能力：现代企业的种类越来越多，对应的业务单元也越来越多，以后工作流的一个研究方向是扩展工作流模型的表达含义，适应种类繁多的企业业务逻辑的表达。

## 1.2 工作流技术的不足

工作流技术从出现到成熟取得了长足的发展。但是就目前的工作流技术而言，仍然存在很多不足，具体分为以下几点[5]：

1、标准化程度差,系统的集成性不理想

虽然现在已经存在 WFMC 专门负责工作流的管理及标准的制定，但是和很多新兴技术一样，各大开发商并没有一个权威的第三方标准，因此开发出来的工作流产品存在互不兼容的问题，很难将不同的工作流产品融合起来形成一个更大的应用。

2、目前的工作流不适用于企业级的工作流管理

工作流系统最初并不是大规模企业设计的，由于初期的工作流系统本身的一些缺陷，故而不可能直接将现有的工作流系统应用到企业级流程中。

3、系统中对于并发访问和异常错误缺乏可靠的支持和恢复机制

数据是企业的灵魂，一旦数据丢失将会为企业带来巨大的损失。在现今分布式网络环境中，可靠的工作流系统需要能够处理这些异常情况并且能够及时地恢复关键数据。

4、实施的复杂性

为企业定制一个完善的工作流系统是一件庞大的工程，不仅需要对整个企业

的流程的运行有一定的了解，而且再进行企业流程处理分析和改造、管理规程和操作规范建立的过程中还必须有不同软件供应商的全力配合。

## 1.3 本论文的选题和研究内容

本课题来源于国家863项目“支持数字媒体内容创作的集成环境”，通过研究工作流技术及相关标准，开发具有自主知识产权的工作流平台。该平台是支持“数字媒体内容创作的集成环境”的中间件服务平台，提供商业业务过程的建模、流程编辑、工作流流程管理等功能。平台将提供一套定义良好的服务组件，以便用户程序可以采用可视化流程编辑的方式使用系统平台进行工作流建模及编辑。系统采用MVC框架结构，分离了模型和业务逻辑从而保证了系统的松散耦合性，同时可以满足企业业务的快速变化。本课题是工作流系统的核心组成部分，主要对可视化工作流建模、编辑及仿真子系统进行设计和实现。

该项目主要是开发基础平台软件，该平台不仅支持“数字媒体内容创作的集成环境”，还支持各个领域的商业业务过程的工作流建模和管理，这是一套基础服务系统。具体而言，该系统应该具有稳定性、可靠性和可扩展性的基础平台软件，它能够满足商业业务管理系统的开发和管理，能够支持业务流程定义、可视化流程编辑，适应业务逻辑的变化。

基于上述目的，本文主要工作和具体的研究内容如下：

本文首先介绍了工作流的基本概念、工作流管理系统及WebServices。接着对BPEL4WS和目前主流的基于服务的流程描述语言进行研究、分析和比较。重点分析和研究Web Service技术和基于Web Service定义的商业流程执行语言BPEL4WS。

然后介绍了Eclipse的体系结构，设计并实现了支持BPEL4ws工作流流程定义的图形库。分析了支持BPEL4ws的工作流编辑器的功能需求，然后利用Eclipse框架设计并实现了可视化工作流编辑器。该编辑器支持基于BPEL4ws的流程描述和流程仿真，并能自动进行BPEL语法的规范性。

最后通过具体的应用案例演示了本系统的工作过程，证明了本系统的实用性和易用性。

本论文结构为：第一章为绪论，主要介绍工作流的基本概念。第二章为关键技术研究，详细介绍了相关技术。第三章为基于Eclipse的可视化工作流编辑器的分析与设计。第四章为基于Eclipse的可视化工作流编辑器的实现与仿真。第五章

为全文总结，总结了本文的创新点及不足之处。

# 第二章 关键技术研究

## 2.1 工作流技术概述

根据负责工作流管理系统 WfMS（Workflow Management System）标准化工作的工作流管理联盟（WfMC）的定义[8]:“工作流是一类能够完全或者部分自动执行的经营过程，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传递、执行，从而实现预期的业务目标”。工作流需要解决的问题是如何协同多个伙伴之间的信息传递及任务的交互执行，以达到预期的目的。

工作流基本概念如表 2-1 所述：

表 2-1 工作流基本概念

| 概念名称 | 具体描述 |
| --- | --- |
| 工作流管理系统 WFMS(Workflow Management system) | 专门负责定义和管理工作流程的一种软件系统，同时负责管理工作流的在运行环境中的执行过程 |
| 工作流(Workflow) | 将企业流程抽象为计算机可以理解的一种流程，可以由计算机自动执行而不需要人工干预 |
| 工作流引擎(Workflow Engine) | 支持工作流执行的软件环境和工具的集合 |
| 流程(Process) | 通过在伙伴、人物和任务之间传递消息，规范操作来表达商业逻辑的一种模型 |
| 活动(Activity) | 流程的基本单元，用于实现具体功能的工作单元 |
| 任务(Task ) | 活动实例的输入对象，等待活动实例进行处理 |
| 操作者(Operator) | 任务的执行者 |
| 任务列表(Tasklist) | 需要操作者完成的一系列的相关任务列表 |

### 2.1.1 工作流管理技术

工作流技术是对商业业务过程进行抽象建模，管理业务过程的办公自动化的核心技术。对企业利用工作流方法进行业务过程的建模和深入分析不仅可以规范

化企业的业务流程，发现业务流程中不合理的环节，进而对企业的业务过程进行优化重组。工作流管理联盟对工作流管理系统的定义为[9]：工作流管理系统是一个软件系统，它负责工作流的定义和管理，并按照在计算机中预先定义好的工作流逻辑推进过程实例的执行。

### 2.1.2 工作流管理系统

工作流管理系统是运行在工作流引擎上的一套软件系统，其目的在于建立和管理工作流的运行。工作流管理者使用该系统来推进工作流实例的执行，并通过监控软件来监控工作流的运行状态。工作流管理系统是抽象在具体领域企业业务之上的管理系统，它与具体的商业逻辑并不相关，故而可以使用该系统建模和管理各个领域的商业业务过程。从比较高的层次上来抽象地观察工作流管理系统，可以发现所有的工作流管理系统都提供了以下基础功能[10]：

- 建立阶段功能：通过对业务流程进行建模，用计算机可以识别的方式来表达真实的业务逻辑，建模和表达的过程中主要要用到流程编辑工具。
- 运行阶段的控制功能：使用流程编辑器将业务流程转换为工作流程之后，需要把工作流程放到特定的环境中去运行，这个环境称之为工作流引擎。而控制功能就是负责调度这些工作流程。
- 运行阶段的人机交互功能：工作人员可以通过工作流引擎提供的接口，如命令行、可视化界面等监控各种活动运行的状态，在必要时人为进行调控。

### 2.1.3 工作流参考模型

目前各大厂商的工作流产品在所需的执行环境、功能特性等方面各有不同，为了到达统一的标准，以便充分发挥各个工作流系统的优势，WfMS 就实现互连和互操作的标准做出了努力，并于 1994 年 11 月推出了工作流系统参考模型[10]，如图 2-1 所示。

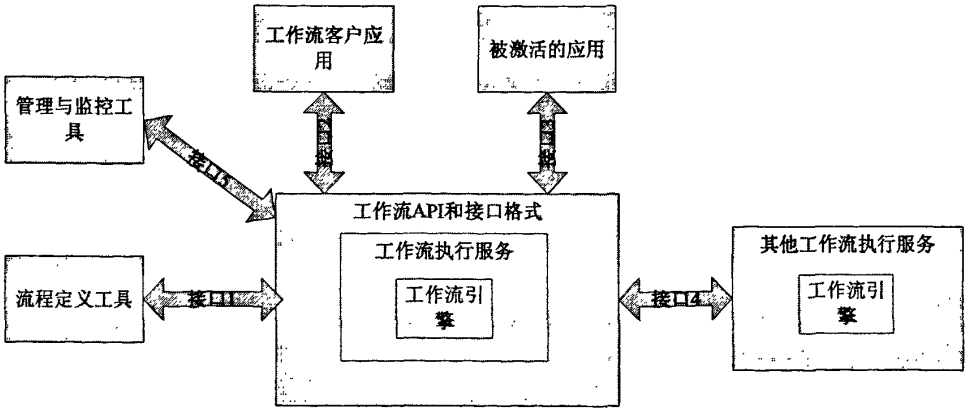



图 2-1 工作流系统参考模型

接口 1：工作流运行服务和工作流定义工具间的接口，作用在于对工作流模型的解释和读写访问。接口 1 提供了创建和修改工作流定义的工具与工作流执行服务之间的关系。这个 WAPI 包括如下功能：打开和关闭一个连接，获取工作流定义（过程定义和资源分类）以及打开、创建和保存一个过程定义。

接口 2：工作流运行服务和客户方应用之间的接口，用于工作列表处理器和执行服务之间的交互。其 WAPI 支持以下功能：打开和关闭一个连接，生成案例和工作项状态的摘要信息，新案例的生成以及活动的开始、中断和完成。同时它还约定了客户方应用和工作流运行服务之间的功能访问方式。

接口 3：工作流引擎和供调用应用间的接口，实现该接口规范的功能应用可以被工作流引擎调用。基于工作流运行控制服务标准接口的“激活”方法很多，但都可以被“应用代理者（ApplicationAgent）”概念所包含。通过一个标准的 API 集与工作流运行控制服务进行通信，接收应用数据、信号和响应活动事件等。过程可以直接使用这些 API，与传统应用或其它非工作流应用进行交互。

接口 4：工作流管理系统之间的互操作接口，这是构造大规模分布式工作流系统的重要接口标准。该接口使能了若干自治工作流系统之间的工作交换。通过该接口，不同的工作流管理系统可以实现互连或集成。

接口 5：工作流运行服务和工作流管理工具之间的接口，该接口侧重于管理和监控工具与工作流执行服务之间的链接。它可分为两个部分：工作流系统管理功能和工作流追踪功能。前者包括员工的添加、授权的许可以及过程定义的执行。[1]。

工作流管理联盟仍然致力于 WAPI 的标准化。迄今为止在接口 3 和接口 5 的标准化方面进展甚微。

## 2.2 Eclipse 及相关技术

### 2.2.1 Eclipse 概述

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。它本身是一个框架和一组服务的集合，是通过插件组件搭建起的开发环境。与此同时 Eclipse 附带了一个标准的插件集，其中包括 Java 开发工具。

从最终用户的观点来看，Eclipse 平台使用工作台模型来集成各种工具，使用扩展点来将用户开发的工具插件插入到平台中。Eclipse 平台正是根据扩展点这一概念来建立的，扩展点是系统中定义的一些位置，同时其他工具可以在此添加功能。平台中的每个主要子系统本身是由一组插件来构成的，这些插件实现一些重要功能并定义扩展点。Eclipse 系统本身是通过向第三方插件供应商提供扩展点来构建的，插件供应商可以通过这些扩展点来添加各种功能。插件可以定义它们自己的扩展点，或者只是将扩展点的功能添加到其他插件的扩展点上。

虽然大多数用户将 Eclipse 当作 Java 的集成开发环境来使用，但 Eclipse 的目标远不止于此。Eclipse 还包括插件开发环境(Plug-in Development Environment PDE)，这个组件主要适合希望扩展 Eclipse 的软件开发人员。通过这个插件可以开发与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的所有功能都是插件，任何人都可以开发和提供 Eclipse 插件，也可以根据自己的喜好搭建自己的集成开发环境，所有插件的安装和卸载都非常的容易。用户可以选择一个一致又富有个性的集成开发环境[11]。Eclipse 体系结构图如图 2-2 所示:

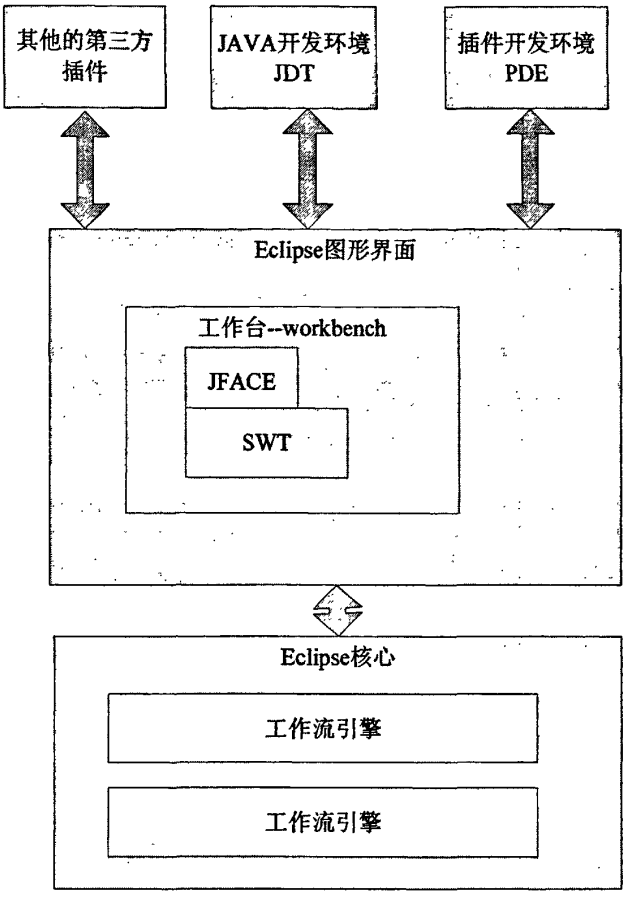



图 2-2 Eclipse 体系结构

### 2.2.2 插件概述及原理

插件是 Eclipse 的功能扩展点，插件与 Eclipse 本身是松散耦合的。在开发的过程中需要用到某个功能插件时，只需下载到相应的插件，然后通过 Eclipse 预留的扩展点来配置使用这个插件。当然，开发出自己的插件后，同样可以预留自己的扩展点以便别人可以在此插件上进行 2 次开发。

正因为插件的即插即用的方便性及与 Eclipse 之间的松散耦合性，Eclipse 本身就可以得到最大限度的精简。虽然许多常规功能都是绝大多数 IDE 环境所必备的功能，Eclipse 却也把它们都做成了插件模式，甚至用来开发 Java 程序的开发环境也只不过是 Eclipse 系统中的一个普通插件而已。整个 Eclipse 体系结构就像一个结构图，可以不断的向上加插件，同时，现有插件上还可以再加插件。

### 2.2.3 GEF 概述

GEF（Graphical Editor Framework）最早是 Eclipse 的一个内部项目，后来逐渐转变为 Eclipse 的一个开源工具项目。GEF 是一个基于 MVC 体系结构的图形化编辑框架，它允许开发人员以图形化的方式展示和编辑模型，从而提升用户体验。与其他一些 MVC 编辑框架相比，GEF 的一个主要设计目标是尽量减少模型和视图之间的依赖，好处是可以根据需要选择任意模型和视图的组合，而不必受开发框架的局限。GEF 的架构如图 2-3 所示。GEF 包括了控制层，即一些基于 SWT 的控制组件；视图层，包括图形元素、上下文菜单、连线等；编辑域、命令栈等其他一些核心组件。

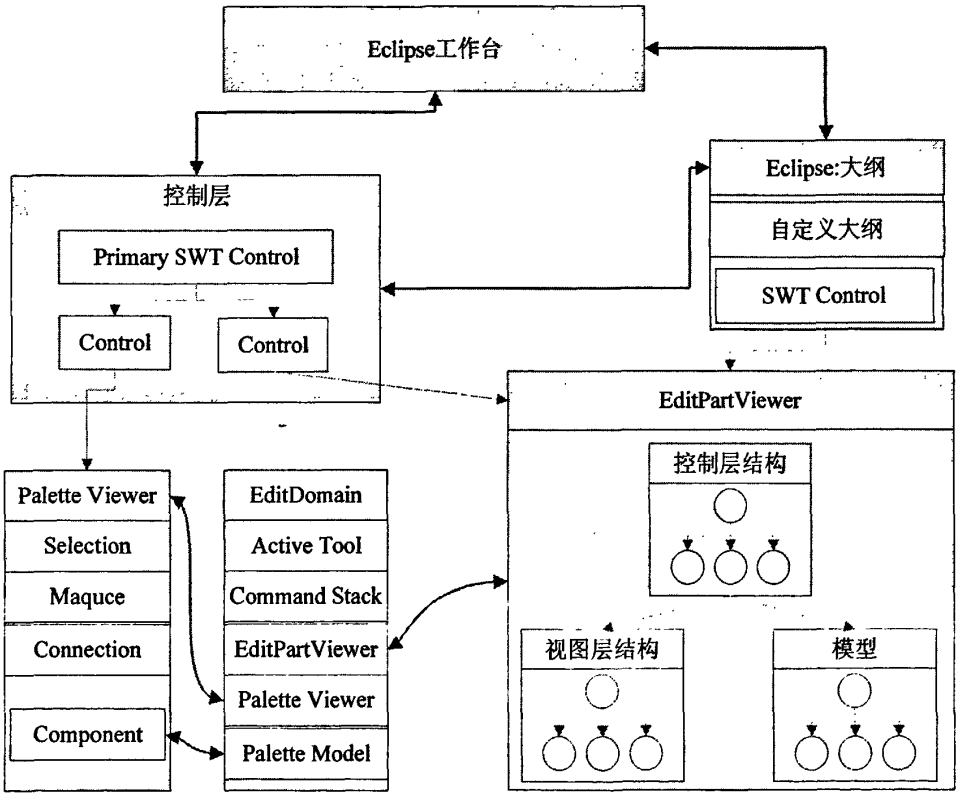



图 2-3 GEF 整体架构图

## 2.3 WebService

Web Service 是这样一种软件平台：它通过标准的协议和文档描述 Web 服务接口，该接口支持分布式计算，是发布到互联网上的基于服务组件的平台。它描述

了服务操作的接口，该接口通过隐藏服务实现的细节使得整个体系呈现出一种松散耦合的关系，用户不必关心服务的具体实现，只需通过查询获得特定的 Web 服务列表，然后调用合适的服务即可。Web 服务是一种部署在 Web 上的服务和对象，它具有以下特征[17]:

- 完好的封装性，向外界提供功能接口，屏蔽底层实现。
- 松散耦合：对于使用 Web 服务的客户而言，整个底层的功能实现都是透明的，只要 Web 服务的接口不发生变化，底层实现不会对用户产生任何影响。
- 使用协约的规范性：通过统一的标准将 Web 服务发布到公共注册库中，方便用户查询和访问。
- 使用标准协议规范：使用标准的协议进行数据传输，并有公认的第三方维护这些标准。
- 高度可集成能力：因为不同平台下的 Web 服务都使用标准的 Web 协议描述服务接口，所以可以很方便的集成起来。

虽然 Web Service 具有许多优点，但是它和许多新兴的技术一样还没有一套成熟的标准和体系。目前的 Web 服务在互联网上的应用单一而且种类很少，从而不能体现强大的功能和体系上的优越性。

在这样的情况下， BEA、IBM 和 Microsoft 在 2002 年 联合推出了一套关于 Web 服务的新规范 BPEL4WS(Business ProcessExecution Language for Web Services，简称 BPEL4WS 或 BPEL)。即基于 Web Service 的业务流程执行语言。

## 2.4 BPEL4ws 技术

### 2.4.1 BPEL4ws 概述

BPEL4WS 提供了一种 XML 格式的用来描述业务交互协议和商业流程的语言，它通过这种方式扩展了 Web 服务的交互模型来支持业务的事务处理。同时，BPEL4WS 还定义了一个可互操作的集成模型，该模型使得企业内部以及业务之间的自动化流程集成变得更为容易。

### 2.4.2 BPEL4ws 基本结构

BPEL4WS 支持的业务流程能够指定一些关键信息，包括一组 Web Service 操作的可能执行顺序、Web Service 共享的数据、业务流程涉及的伙伴以及这些伙

伴扮演的角色。为了实现这些功能，BPEL4WS 引入了活动(Activity)、伙伴(Partner)、伙伴类型（Partner Link Type)、端点引用（Endpoint Reference)、变量(Variable)、相关性、错误处理、作用域和消息属性等关键元素。图 2-4 为 BPEL4WS 语言的基本结构[15]：

```
<process name="ncname" targetNamespace="uri"
         queryLanguage="anyURI"?
         expressionLanguage="anyURI"?
         suppressJoinFailure="yes|no"?
         enableInstanceCompensation="yes|no"?
         abstractProcess="yes|no"?
         xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/">

  <partners>?
    <!-- Note: At least one role must be specified. -->
    <partner name="ncname" serviceLinkType="qname"
             myRole="ncname"? partnerRole="ncname"?>+
    </partner>
  </partners>

  <containers>?
    <!-- Note: The message type may be indicated with the messageType
         attribute or with an inlined <wsdl:message> element within. -->
    <container name="ncname" messageType="qname"?>
      <wsdl:message name="ncname">?
        ...
      </wsdl:message>
    </container>
  </containers>

  <correlationSets>?
    <correlationSet name="ncname" properties="qname-list"/>+
  </correlationSets>

  <faultHandlers>?
    <!-- Note: There must be at least one fault handler or default. -->
    <catch faultName="qname"? faultContainer="ncname"?>*
      activity
    </catch>
    <catchAll>?
      activity
    </catchAll>
  </faultHandlers>

  <compensationHandler>?
    activity
  </compensationHandler>

  activity
</process>
```

图 2-4 BPEL4WS 语言的基本结构

## 2.4.3 活动（Activity）

BPEL4WS 流程类似于用来表达算法的流程图。流程的每一步称为一个活动。活动包括基本活动(Basic Activity)、结构化活动(Structured Activity)和特殊活动（Specail Activity)，基本活动负责实现一定的原子功能，如赋值(Assign)、调用 Web 服务(Invoke)等。特殊活动是指变量的作用域、异常的处理等，如补偿

（compensate）、中止流程执行（Terminate）等。结构化活动则利用循环(While)、分支(Switch)等元素对基本活动进行组装整合以实现系统所需的逻辑功能。BPEL4WS 允许递归地组合结构化活动，以表达任意复杂的算法，这些算法代表和体现了服务的具体实现。

BPEL4WS 的活动有基本活动和结构化活动两种类型。其中，基本活动包括：

- Receive：在活动中阻塞等待并获取同伴发来的消息。
- Reply：接受并处理从活动 receive 发来的消息。
- Receive 和 Reply 组合起来可以在业务流程中表达请求和响应操作。
- Invoke：调用 Web 服务接口。
- Assign：为相关活动赋值，并负责活动之间数据转换。
- Throw：当活动出现异常时，向异常处理器抛出异常。
- Wait：为活动或者某个事件的执行构造时间条件，这个时间条件可以基于时间点也可以基于时间段。
- Empty：通过不同的参数设定表达不同的功能，例如构造在业务流程中插入“no-op”指令。这个构造可用于并行活动的同步。

而结构化的活动通过组合基本活动来表达负责的业务逻辑，有以下一些活动：

- Sequence：定义 Sequence 结构化活动中的基本活动按顺序执行。
- Switch：这个有点类似于高级语言的 Switch，通过 case 和 othercase 来指定活动的执行条件。
- While：定义 While 中的基本活动循环执行，直到出现临界条件。
- Pick：构造阻塞并等待某一个合适的消息的到达或超时警报响起。当其中一个触发器触发后，相关的活动就被执行，pick 也随即完成了。
- Flow：比较开放的结构化活动，可以灵活的定义单个或者多个并行执行的基本活动。可以在并行的活动中使用链接来定义任意的控制结构。

此外还有一类特殊活动，它们既不是基本活动也不是结构化活动，而且不能单独存在，它们只能依附于结构化活动或者在特定的场景中存在。这类活动包括以下几种：

- Catchall 和 Catch 用于获取异常信息，在错误处理器中经常使用。
- OnMessage 和 OnAlarm 通常用于 pick 和事件处理器中，负责响应消息和警告。
- Case 和 Otherwise 是活动 switch 的基本构成单元。
- Scope 表示活动域，并在自己的域活动中定义相匹配的参量及异常处理和

补偿处理程序。

- Compensate 构造用来在已正常完成执行的内层作用域上调用补偿。当流程正常结束时，在作用域内调用补偿活动处理善后事宜；通常会在异常捕获器和补偿处理器中使用 Compensate。

### 2.4.4 抽象流程和可执行流程

如前面所述，BPEL4WS 流程分为抽象流程和可执行流程。抽象的 BPEL4WS 商业流程的定义：一个未完全定义的且不打算被执行的商业流程。因此在一个抽象流程中，一些操作细节会被省略使得流程是未完全定义的，并且通过设置 process 元素的 abstractProcess 属性为 yes 来标识流程的抽象状态。对于抽象流程可以做如下理解:

- 可以通过添加某些未定义的属性将抽象流程转化为可执行流程。
- 抽象流程和可执行流程一样，对于已经定义的属性是清晰的和统一的。而且可执行流程中所有的元素结构同样适用于抽象流程。
- BPEL4WS 流程可以通过使用抽象流程来定义业务协议角色。

其实抽象 BPEL4WS 流程和可执行 BPEL4WS 流程最大的区别也可能仅仅在于 process 元素中 abstractProcess 属性的值。换而言之，要将一个抽象 BPEL 流程装化为可执行流程其实很简单，我们只需为这个抽象流程添加某些必要的属性并且将 abstractProcess 属性的值设置为 no 或者删除 abstractProcess 属性。这样该抽象流程就可以转换成相应的可执行 BPEL4W S 流程，而不需要进行任何进一步的细化或者扩展。

### 2.4.5 错误处理（Fault Handler）

当商业流程执行出错时， Fault Handler 负责处理出错消息，处理操作类似于数据库中的回滚。一个错误处理器可以拥有多个 catch 和 catchAll 处理器中。

当流程执行错误时，所产生出错信息的区域的流程会马上终止，在任何情况下这种终止操作都是必需，不管出错的严重性和出错频度。一个活动在发生错误的情况下会抛出一个出错消息。该消息首先会被自身的错误处理器所处理。错误处理器会尝试三种解决方案：

- 解析出错消息，依照错误处理规则寻找出错根源并处理。
- 同高级语言的异常处理一样，暂时不处理错误，而是构造一个 throw 活动

向上级发送一个出错消息。

- 第三种处理方式是直接中止流程的运行。

### 2.4.6 事件处理(Event Handler)

事件处理主要负责两类事件的处理：消息事件和警告事件。前者是从活动外部发来的消息，而后者是到了预设的时间点而发出的警告。事件处理机制从作用域的一开始就激活，一直等待事件的到来而执行内部行为，也会随着作用域的结束而结束。

### 2.4.7 补偿处理(compensation Handler)

Compensation Handler 的目的在于恢复活动的前一系列行为。所做的操作就是调用一个效果相反的服务来撤销作用域内已执行的相应行为。正因为如此，一个服务的补偿操作通常就是由该服务提供的，该服务比其他服务更容易处理自己已进行的操作。补偿处理器只是作为补偿活动的容器，本身不表达任何附加的信息。与错误处理器不同的是，补偿处理器被当作业务逻辑正常的一部分，补偿处理器的调用不会使当前流程区域变为非正常状态。

## 2.5 本章小结

本文首先介绍了工作流的基本概念、工作流管理系统及 WebServices。然后介绍了 Eclipse 及插件原理，分析了 GEF 的体系结构，为下面利用 GEF 设计可视化图形库打下了基础。接着对 BPEL4WS 和基于服务的流程描述语言进行研究、分析和比较。重点分析和研究了 Web Service 技术和基于 Web Service 定义的商业流程执行语言 BPEL4WS。在接下来的章节中将对基于 Eclipse 的工作流编辑器的设计进行讨论。

# 第三章 基于 Eclipse 的可视化工作流编辑器的分析与设计

在其它工作流管理系统中，可视化工作流编辑器可能被称为可视化工作流建模及仿真子系统。它以计算机能够处理的形式进行过程的定义，可以基于形式化的过程定义语言、对象—关系模型来进行过程模型定义。可视化工作流流程编辑器应输出一个能被工作流引擎解释并执行的过程定义。不同的工作流产品其建模工具输出模型的存储格式是不同的，在本系统中产生的过程定义是基于 BPEL4WS 规范的，BPEL4WS 是目前最主流的流程定义语言规范。

要实现 BPEL4WS 可视化流程编辑器实际上是实现一种基于 Web 服务的工作流建模工具。该建模工具将提供一套良好的访问和管理系统的 API 组件，以便用户程序可以采用可视化流程编辑的方式使用系统平台进行工作流建模及编辑。本系统在保证了系统具有良好松散耦合性的基础上，满足企业业务的快速变化。同时还应该保证系统具备以下几个特点：具有开放性和可扩展性；具有稳定性和可靠性。

可视化工作流编辑器的应用场景如图 3-1 所示：设计人员采集商业流程需求，使用工作流编辑器为商业逻辑建立流程模型，形成一系列完善的商业流程。然后通过 UDDI（命名注册）查询可用的 WebService，通过调用和组合的方式将商业流程装化为计算机可以识别的 WebService 流程描述，并且生成可以执行的版本然后发布到支持组合服务的 Web 服务环境中。监控人员通过监控核管理工具观察和管理服务实例的整个运行过程，同时可以通过在管理工具上预留的接口上使用第三方工具来进行交互和调用。整个系统的角色包括：工作流编辑器的设计人员、Web 服务环境的维护人员和组合服务的管理人员。

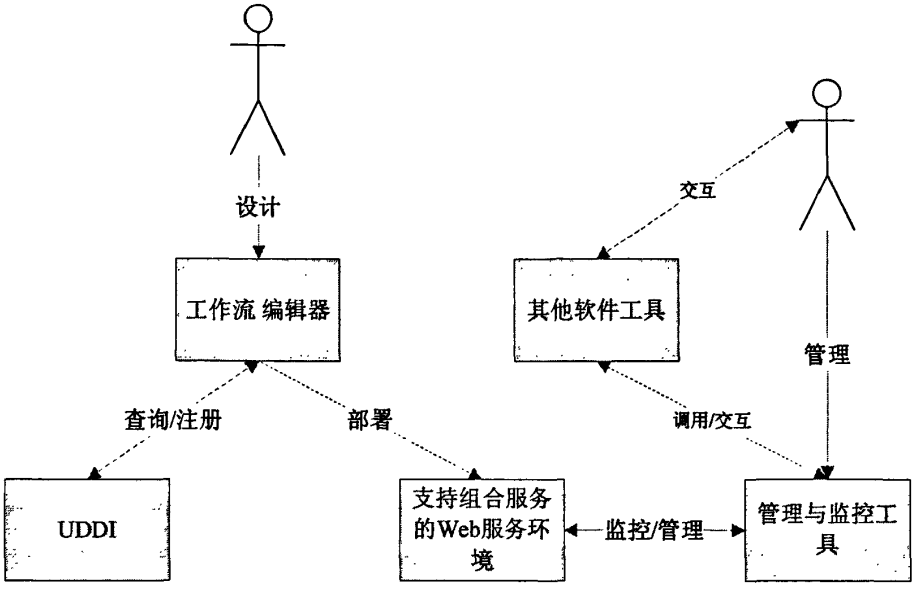



图 3-1 工作流编辑器的应用场景

基于上述分析，一个基于 BPEL4WS 的工作流编辑器的功能需求包括：

- 对建模过程的支持：能够支持 BPEL4WS 的工作流模型的定义，通过一组基础元素以及元素的组合描述一个组合服务的执行细节。对服务查询的支持：由于组合服务的定义过程关注如何将已有的、实现子功能的 Web 服务组合成为新的服务。
- 对流程描述和仿真的可视化支持：提供图形化界面用于绘制流程，流程对应企业商业逻辑，通过对业务流程的组合和描述生成可以被工作流运行环境支持的可执行工作流。

通过对基于 BPEL4WS 的工作流编辑器的功能需求分析，本系统必须实现以下几个部分：

- 定义 BPEL4WS 的工作流模型，建立模型映射机制，即工作流模型与可视化图元模型之间的映射。由于 WebService 具有分布性和异构性等特性，所以建立这种工作流模型的难度较大。而采用基于有向图的建模方法，可以简化这种复杂性，使建模过程清晰明了。可视化工作流编辑器的最主要功能是：根据具体的商业业务过程来建立计算机可以理解的流程描述，该流程描述必须符合 BPEL4WS 的业务流程描述语言的语法格式。
- 设计并实现一个支持 BPEL4WS 模型可视化表示的图形库。工作流编辑器是提供给用户用于设计流程的最重要工具，所以必须必须支持用户通过鼠标点击、

选择，键盘操作，实现活动图元之间的连线与属性定制。这就需要设计和实现一个能够满足 BPEL4WS 的工作流模型要求的图形库提供可视化的支持。例如：对模型中的一些结构化活动，图形库应提供能支持嵌套的图元的原型。

- 基于 BPEL4WS 的描述文件的生成与管理，一个在 Web 服务环境中运行的工作流可以被计算机理解的就是可执行组合服务的描述文件，这些描述文件必须符合行业标准，才可能实现 Web 服务之间的无缝结合。最重要的一个问题是如何解决 BPEL4WS 流程描述语言版本问题，也就是活动模型与 BPEL4WS 语言版本映射。因此，需要实现一个可扩展的格式转化机制。
- 设计工作流流程编辑器的总体结构，并实现其各个功能模块。工作流流程编辑工具在前三项工作的基础上进行设计，包括多个功能模块，例如：实现多种视图的用户界面可以提高建模工具的易操作性，根据定义的 BPEL4WS 工作流模型设计实现绘图系统，模型的属性编辑，模型存储，图元模型的本地持久化及反向生成流程定义文件，根据生成器框架实现描述语言生成器和 Web 服务封装格式生成器等等。

## 3.1 基于 BPEL4ws 的工作流模型的分析与设计

基于 Web 服务的工作流建模的对象是 Web 服务，具有分布性、自治性和异构性特征，这增加了建模过程的复杂性。而基于有向图的建模方法，直观、易于理解，可以简化这种复杂性。业务流程描述语言是一种规范化的流程描述方式，用来描述和整合具体业务功能，为业务流程的自动化实现提供基础。

为了建立用户需要的流程描述，并符合 BPEL4WS 的业务流程描述语言的语法格式,这部分主要内容包括：（1）分析及定义 BPEL4WS 的工作流模型。（2）设计可视化图元模型（3）建立模型映射机制，即工作流模型与可视化图元模型之间的映射。

### 3.1.1 建立模型映射机制

BPEL4WS 的活动可以分为两类：基本活动，结构化的活动。基本活动定义为商业流程内的原子操作，结构化的活动用于定义流程的逻辑层次，它通过定义一组活动的执行顺序来描述一个复杂的商业流程。实现可视化流程编辑的关键就是通过特定的属性编辑器来编辑模型的属性，从而映射到商业流程的对应属性。从功能角度来看，除了为图元模型定义 BPEL4ws 活动本身的属性，还需扩展许多其

他属性，比如 BPEL 的各种表达式属性，执行条件属性，用于仿真的属性。表 3-1 定义了 BPEL4ws 商业活动与可视化图元模型的映射关系。如何设计各种可视化图元模型将下节详细阐述，这些模型的属性总体上可分为 3 类：

- 仿真属性：商业流程执行时，需要使用仿真属性来表示流程的执行状态，仿真结果等信息。
- 基本属性：根据 BPEL 活动的相关语法规定，所必须的属性。
- 高级属性：涉及活动故障处理、事务回滚，XML 语言相关的高级属性。

表 3-1 BPEL4WS 商业活动模型与可视化图元模型的映射

| BPEL4WS 活动模型 | 可视化图元模型 | BPEL4WS 活动模型- |
|---|---|---|
| Invoke（基本活动） | Invoke（简单图元模型） | 基本活动--简单图元模型 |
| Receive（基本活动） | Receive（简单图元模型） | 基本活动--简单图元模型 |
| Reply（基本活动） | Reply（简单图元模型） | 基本活动--简单图元模型 |
| Assign（基本活动） | Assign（简单图元模型） | 基本活动--简单图元模型 |
| Throw（基本活动） | Throw（简单图元模型） | 基本活动--简单图元模型 |
| Terminate（基本活动） | Terminate（简单图元模型） | 基本活动--简单图元模型 |
| Wait（基本活动） | Wait（简单图元模型） | 基本活动--简单图元模型 |
| Empty（基本活动） | Empty（简单图元模型） | 基本活动--简单图元模型 |
| Sequence（结构化活动） | Sequence（结构化图元模型） | 结构化活动--结构化图元模型 |
| Switch（结构化活动） | Switch（结构化图元模型） | 结构化活动--结构化图元模型 |
| While（结构化活动） | While（结构化图元模型） | 结构化活动--结构化图元模型 |
| Pick（结构化活动） | Pick（结构化图元模型） | 结构化活动--结构化图元模型 |
| Flow | Flow | |
| Scope | Scope | |
| Case（非活动） | Case（内嵌图元模型） | 其他--内嵌图元模型 |
| Otherwise（非活动） | Otherwise（内嵌图元模型） | 其他--内嵌图元模型 |

| | | |
|---|---|---|
| OnAlarm（非活动） | OnAlarm（内嵌图元模型） | 其他--内嵌图元模型 |
| OnMessage（非活动） | OnMessage（内嵌图元模型） | 其他--内嵌图元模型 |
| Link（连线） | Connection（连线模型） | 连线--连线模型 |

### 3.1.2 简单图元模型的设计

简单图元模型映射为 BPEL4ws 的简单活动，负责处理商业流程中的原子操作，可以由结构化活动调用形成特定的业务逻辑。设计一个抽象类 ActionElement 来描述简单图元模型的公共属性、操作。简单图元模型包括 invoke, receive, reply,assign, throw, wait 和 empty。设计类图如图 3-2 所示。

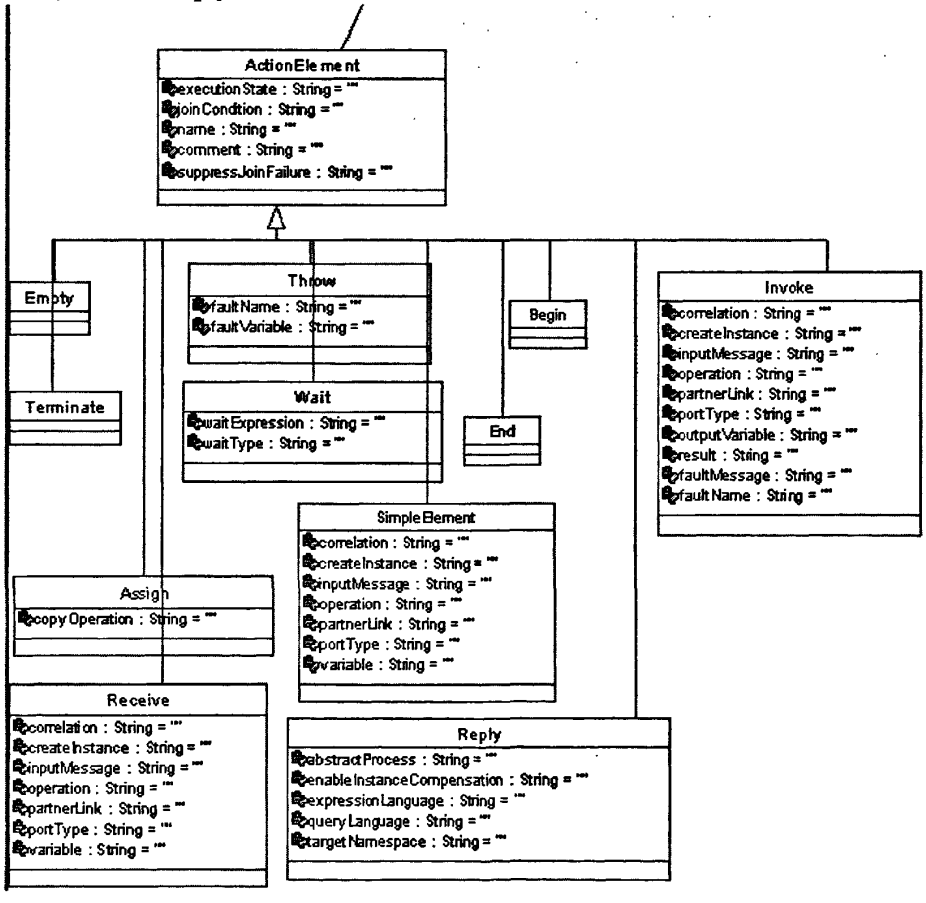



图 3-2 简单图元模型设计类图

这些简单图元模型有一些共同的属性 standard-attributes 如 name 表示模型的名字；joinCondition 指明活动的执行条件；suppressJoinFailure 指明若执行条件不满

足是否忽略由此产生的错误。当为 true 时忽略该错误。这些基本模型还包括子元素(standard-elements)，如 source 和 target 两个元素，这两个元素用来定义并行活动之间的同步关系。下面逐个对简单图元模型进行设计。

Invoke:

在 invoke 图元模型的语法当中（为了描述的简洁性，下面的活动设计均简称图元模型为模型），Join Condition 为活动的执行状态，通常用 BPEL 表示式来表示；catch、catchAll 以及 compensation 是 invoke 的一套异常捕捉和错误补偿机制，其中 catch 和 catchAll 负责捕获工作流程运行期间的异常信息，而 Compensation Handler 专门负责恢复或者采用其他补偿机制处理异常发生时的状态改变。

Invoke 设计为功能强大的模型，除了上述特性，Invoke 不但可以和流程打交道，还可以非流程单元进行信息交换。由于 WSDL 可以把不同的功能定制成统一格式的服务，所以 Invoke 活动能交互功能种类很多，可以是 WebService、EJB 组件、或者是其他 BPEL 流程。Invoke 的属性设计如表 3-2 所示。

表 3-2 Invoke 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Correlation | 在有状态会话当中用来定义会话的状态 |
| Execution State | Invoke 活动的执行状态 |
| Name | 活动名称，基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数来表达 |
| Input Variable | 发送给被调用服务的信息， |
| Output Variable | 返回的信息，请求/回应调用需要有此属性 |
| Partner Link | 说明跟哪个角色交互 |
| Port Type | 说明跟 Partner Link 指定角色的哪个类交互 |
| Operation | 说明使用 Port Type 指定类的哪个函数 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略出此 |

| | |
|---|---|
| | 产生的错误。当为 true 时忽略该错误。 |
| Fault Message | 仿真的默认返回信息，仿真属性 |
| Result | 仿真结果，仿真属性 |
| Fault Name | 仿真的默认名字，仿真属性 |
| Output Message | 仿真时的控制台输出信息，仿真属性 |

Receive 和 Reply

设计 Receive 和 Reply 模型主要功能为：一个商业流程这个两个活动为伙伴提供服务。

当商业流程为同步调用时， receive 模型接受 partner 的服务请求，而 reply 模型将处理请求并将结果返回；当商业流程为异步调用时，invoke 模型取代了 reply 的功能，它直接调用 partner 的回调操作，而回调操作将负责将返回结果。

其中 receive 模型通过对 createInstance 属性的设定来初始化一个流程，当然商业流程中可能存在多个标注了 createInsatance 的 receive，在这种情况下只能指定一个 receive 来初始化商业流程。当 receive 和 reply 成对出现时，此时商业流程表现为同步调用方式，当 receive 单独出现时，商业流程表现为直接回调式服务。Receive 可以单独存在，但是 reply 却不可以，这可以在上述 reply 的描述中得到解释。所以出现一个 reply 模型，还必须在 reply 之前定义一个 receive 模型，并且这个 receive 模型和 reply 模型的以下属性应该相同：PartnerLink, PortType 和 Operation 属性。

表 3-3 和表 3-4 分别描述了 receive 和 reply 模型的属性设计。

表 3-3 receive 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Correlation | 在有状态会话当中用来定义会话的状态 |
| Execution State | Receive 活动的执行状态 |
| Name | 活动名称，基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数来表达 |
| Input Variable | 发送给被调用服务的信息， |
| Output Variable | 返回的信息，请求/回应调用需要有此属 |

| | 性 |
|---|---|
| Partner Link | 说明跟哪个角色交互 |
| Port Type | 说明跟 Partner Link 指定角色的哪个类交互 |
| Operation | 说明使用 Port Type 指定类的哪个函数 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Input Message | 仿真的默认返回信息，仿真属性 |
| Create Instance | 可选值：TureFale，true 的 receive 活动来初始化商业流程。 |
| Variable | 获得的信息，包含消息和变量。 |

表 3-4 reply 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Correlation | 在有状态会话当中用来定义会话的状态 |
| Execution State | 活动的执行状态 |
| Fault Name | 出错命名 |
| Name | 活动名称，基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数来表达 |
| Partner Link | 说明跟哪个角色交互 |
| Port Type | 说明跟 Partner Link 指定角色的哪个类交互 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Variable | 获得的信息，包含消息和变量。 |

Assign

Assign 模型用于为变量（variable）赋值，它单独存在时没有任何意义。Assign

模型需要和其他模型配合属性。其中 invoke, receive 和 reply 模型都使用变量(variable)来进行消息和数据的传递，而变量的赋值则是通过 assign 活动来进行的。

变量可以 messageType、type 和 element 三种定义方式。其中 messageType 是 Web 服务定义语言的 in/out 消息类型，element 为 Web 服务定义语言 的 types 部分中的自定义 XML Schema 元素，而 type 属性的值为 XML Schema 的内建的简单类型。此外，如果要定义一个类型为 XML Schema 复杂类型的 variable，必须在 XML Schema 类型定义中声明一个类型为复杂类型的 XML Schema 元素，然后在 variable 中设置 element 属性的值为该元素。

Throw、Wait、Empty

Throw 模型用于为运行中的流程抛出异常，这个异常消息可以由错误处理器来捕获并处理，也可以直接终止运行的流程。异常消息中有个重要的属性叫 QName，它用来标注异常消息，以免和其他异常消息发生命名冲突。一般情况下可以使用默认命名即 faultName 来指定所要产生的错误的标识符。throw 模型可以在流程内部抛出特定的错误。faultVariable 是可选属性，表示 throw 模型会抛出一个有关错误的变量，此变量方便 faultHandler 的 catch 节点获得错误的信息，并根据信息采用相应的处理机制。

wait 活动允许流程暂停一段时间或者当系统时钟达到某个具体时间的时候再继续执行，常用于长时间运行的流程。wait 相当于一个定时器，根据预先设定的时间，在特定的条件下触发事件。Wait 定时有两种方式，其一是通过时间点，如明天早上 7 点整触发；其二是时间段，如一个小时后触发事件。

empty 活动不做任何事情，可以被用来作为补偿处理器和错误处理器的默认处理方式。Empty 多用于流程设计的初始阶段，在对整个商业流程把握不是很清晰的情况下，可以预先使用 empty 来替代一些概念模糊的活动，到了设计的后期再使用正确的活动替换之。

表 3-5、表 3-6 和表 3-7 分别对 throw、wait 和 empty 模型的属性进行了详细设计。

表 3-5 wait 属性设计

| 属性名称 | 功能 |
| --- | --- |
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |

| | |
|---|---|
| Execution State | Wait 活动的执行状态 |
| Wait Expression | 等待时间表达式，与 Wait Type 联用。 |
| Wait Type | 等待时间的类型，deadline 暂停到某个时刻为止；duration 表示暂停多长时间。 |

表 3-6 throw 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | Throw 活动的执行状态 |
| Fault Name | 出错名称 |
| Fault Variable | 出错时返回的变量 |
| Name | 名称 |

表 3-7 empty 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | Empty 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 Empty 活动的名字 |

### 3.1.3 结构化图元模型分析与设计

结构化图元模型映射为 BPEL4ws 的结构化活动，用来指定一组活动的执行顺序。这些活动描述了如何通过将活动组合为一些特定的结构来创建商业流程。这些结构则表达了控制流，数据流，错误和外部事件处理方式以及在业务协议中涉及的消息交换方式。结构化图元模型之间能够以任意的方式进行嵌套。根据控制商业流程执行的方式可以将结构化活动分为三种类型：sequence、switch 和 while

活动提供顺序控制；Flow 活动提供并发和同步控制。Pick 则提供了基于外部事件的不确定性选择。同简单图元模型一样，设计了一个抽象类 ContainerElement 来表达结构化图元模型的公共属性和操作。图 3-3 描述了结构化图元模型的设计。

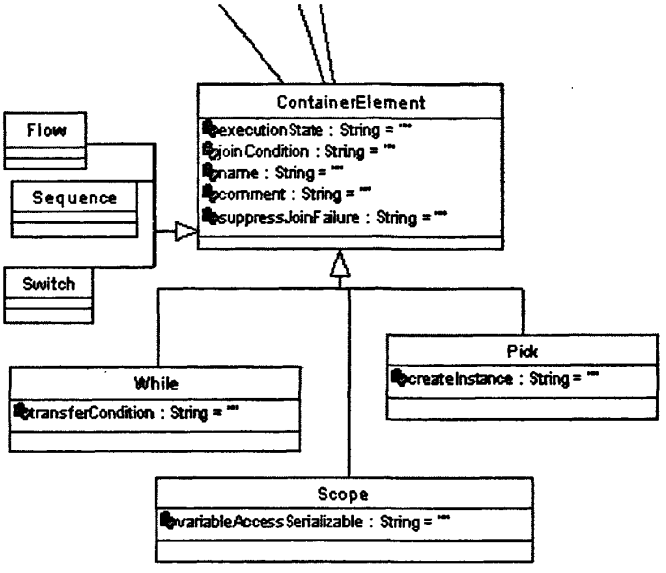



图 3-3 结构化图元模型的设计类图

以下是对这三种流程控制对结构化模型分析与设计：

Sequence：

结构化活动通过组合简单活动来表达复杂的商业逻辑，商业逻辑的执行顺序可以分为两大类：顺序执行和并发执行。顺序执行逻辑用 sequence 来表达，在一个 sequence 模型中，包含了一个或者多个顺序执行的活动，这些活动按照他们被罗列的顺序一个接一个执行。并发执行逻辑用 flow 来表达。但是可以使用 flow 来取代 sequence，即通过设定合适的 link 属性。

Switch：

switch 模型使得流程能够根据条件选择将要执行的活动。一个 switch 模型包含了一个或多个由 case 语句表示的条件分支，以及一个可选的 otherwise 分支。但是，同 C++、JAVA 等高级语言的 switch 相比，这里的的 switch 有所不同。Switch 只会执行一个 case，而不会像高级语言那样会执行其他 case。举个简单的例子，变量 i，如果 i=0 则执行 case 0，如果 i=2 则执行 case 1，其他情况下执行 otherwise。给定参量 i=0，则执行 case0，其他 case 不执行，这点与高级语言有所不同，因为在高级语言中会执行 otherwise。这个有点类似于 if/else，而不是高级语言的 switch。

当然，BPEL 的 switch 的执行条件没上面所述的那么简单。

While：

while 模型用于表达商业流程中的循环逻辑部分。while 循环执行简单活动，直到 while 中的临界条件表达式的值为假。恰好与上述的 switch 不同，while 活动的用法与高级语言的 while 类似。

表 3-8、表 3-9 和表 3-10 分别展示了 sequence、witch、while 模型的属性设计。

表 3-8 sequence 属性设计

| 属性名称 | 功能 |
| --- | --- |
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 sequence 活动的名字 |

表 3-9 switch 属性设计

| 属性名称 | 功能 |
| --- | --- |
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 switch 活动的名字 |

表 3-10 while 属性设计

| 属性名称 | 功能 |
| --- | --- |
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函 |

| | |
|---|---|
| | 数组合来表达 |
| Execution State | 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 while 活动的名字 |
| TranslateCondition | 描述 While 循环的转移条件 |

Flow

前面在分析 sequence 的时候明确了商业逻辑的两大分类，Flow 模型用于表达商业逻辑的并发执行部分，并且其属性 source 和 target 提供了并发活动之间的同步功能。flow 下的活动都将被并发执行。Flow 下活动并发执行是通过 link 属性来控制的，一个 link 包含一个 target 和一个 source，通过对这两个值的设定可以表达活动之间的同步关系。表 3-11 展示了 flow 模型的属性设计。

表 3-11 flow 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 flow 活动的名字 |

### 3.1.4 其他图元模型的设计

其他模型是指简单图元模型和结构化图元模型之外的特殊模型，主要包括：pick、onMessage、onAlarm、connection 等。pick 模型由若干事件/活动对组成。在 pick 模型的语法中，onMessage 表示消息到达事件，而 onAlarm 表示计时器事件。一个 switch 模型包含了一个或多个由 case 语句表示的条件分支，以及一个可选的 otherwise 分支。虽然 onMessage、onAlarm、case、otherwise 都不属于 BPEL 的基本活动或者结构化活动，但是在可视化流程编辑中，却和基本活动及结构化活动有着同样这样的地位，比如在表达一个复杂业务流程逻辑过程， onMessage、

onAlarm、case、otherwise 是必不可少的图元模型。图 3-4 描述了其他模型的类设计图，其中 EmbedElement 作为公共的抽象类描述了公共的属性和方法。表 3-12、表 3-13、表 3-14、表 3-15 和表 3-16 分别为 pick、case、otherWise、OnAlarm、OnMessage 的属性设计。

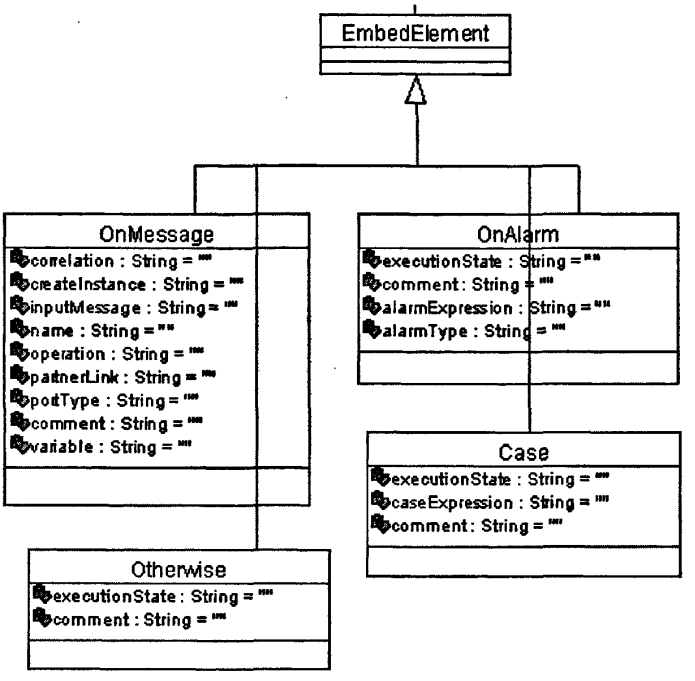



图 3-4 其他模型的类设计图

Pick

Pick 模型使得当某个事件发生的时候能够执行相应的活动。事件由特定消息的到达或计时器引起。比如在早上闹钟响起，相当于 omMessage 到达，便要起床，相当于执行流程。

Pick 模型由对事件和活动组成。在一个 pick 活动中，通过设定属性 onMessage 和 onMessage，前者表示信息到达事件，后者表示计时器事件。同 receive 模型一样，pick 也有一个 createInstance 属性，这个属性只能与 onMessage 事件一起用，表明是否在消息到达的时候初始化一个商业流程，其作用和 receive 模型相同。表 3-12 展示了 pick 模型的属性设计。

表 3-12 pick 属性设计

| 属性名称 | 功能 |
| --- | --- |
| Comment | 对活动的注释，大多数活动具有这个基 |

| | 本属性 |
|---|---|
| Join Condition | 指明活动的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | 活动的执行状态 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Name | 表示 pick 活动的名字 |
| CreateInstance | 功能同 Receive 的 CreateInstance |

表 3-13 case 属性设计

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Case Expression | 指明 Case 的执行条件，用 BPEL 基本函数组合来表达 |
| Execution State | 活动的执行状态 |

表 3-14 otherwise

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Execution State | 活动的执行状态 |

表 3-15 onAlarm

| 属性名称 | 功能 |
|---|---|
| Comment | 对活动的注释，大多数活动具有这个基本属性 |
| Alarm Expression | 计时表达式，用 BPEL 基本函数组合来表达 |
| Alarm Type | 计时器类型，deadline 计时到某个时刻为止；duration 表示计时多长时间。 |
| Execution State | 活动的执行状态 |

表 3-16 onMessage

| 属性名称 | 功能 |
|---|---|

| | |
|---|---|
| comment | 对活动的注释，大多数活动具有这个基本属性 |
| correlation | 在有状态会话当中用来定义会话的状态 |
| Execution State | onMessage 活动的执行状态 |
| Name | 活动名称，基本属性 |
| Partner Link | 说明跟哪个角色交互 |
| Port Type | 说明跟 Partner Link 指定角色的哪个类交互 |
| Operation | 说明使用 Port Type 指定类的哪个函数 |
| Suppress Join Failure | 指明若执行条件不满足是否忽略由此产生的错误。当为 true 时忽略该错误。 |
| Input Message | 进行仿真时通过此属性来传递信息，仿真属性 |
| Variable | 获得的信息，包含消息和变量。 |

Connection 分析与设计

Connection 模型用来连接两个其它的模型，对应的 BPEL 活动是 link，虽然该元素本身来说没有什么操作， 通常也是一个结构化行为内不可缺少的元素。但是还能表达一定的条件选择机制。connection 模型包含 benPoint 用于控制 connection 的可视化折点，它还包含一个 sourceNode 和一个 targetNode。对于一个模型来说，可以跟多个相关 connection，对应到描述文件里面就是一个活动内部可以有多个 sourceConnection 和多个 targetConnection。从设计图来看，connection 本身比不具备自主功能，它的创建、修改和删除是由 connection 内部元素，sourceNode 和 targetNode 的第一个公共祖先决定的。

## 3.2 基于 BPEL4ws 的可视化图形库的设计

### 3.2.1 图论原理

图论（Graph Theory）是数学的一个分支。它以图为研究对象。图论中的图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有

这种关系。

图论逻辑：一个图包含一个非空的元素及 V(G)和一 个 E(G)。其中 E(G)是 V(G)中两个无序元素组成的二元组。V(G)称位图 G 顶点的集合,如果任意集合 V(G)中的顶点 x 和 y,(x,y)在 E(G)中,边(x,y)可能以连接顶点 x 和 y 的边(弧)所代表,x 和 y 就称为邻接的,否则 x 和 y 不邻接。将图形应用系统中的图形分解成： Vertex (顶点)、Edge(边)、Port(连接点)三个基本元素。

系统中的 Connection 类就是按这种思想设计的。图 3-5 描述了 Connection 的类设计图。其中 ConnectionNode 相当于 V(G)，Connection 本身就是 E(G)，它有一个 sourceNode 和一个 targetNode，从而形成一条可以表示事物关系的线。

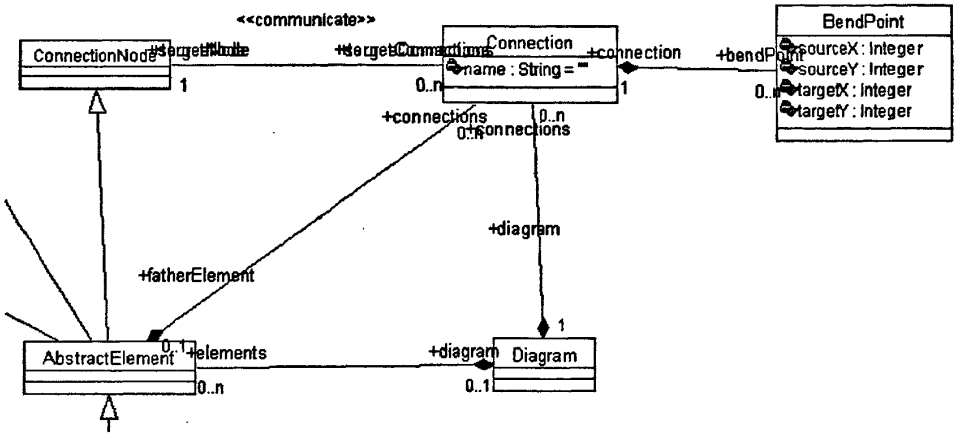



图 3-5 Connection 的类设计图

## 3.2.2 可视化图形库的基本架构

可视化图形库是利用 Eclipse 中的内建框架 GEF 来实现的，GEF(Graphical Editing Framework)是一个开源框架。它使用基于 SWT 的 Draw2D 插件建立图形显示环境。完全采用了 MVC 模式(model-view-controller)，如图 3-6 所示。其中，关键的技术是使用了一类叫 EditPart 的控制器类，负责在操作图形视图时修改业务模型的数据，以及在模型属性发生变化时进行显示更新。

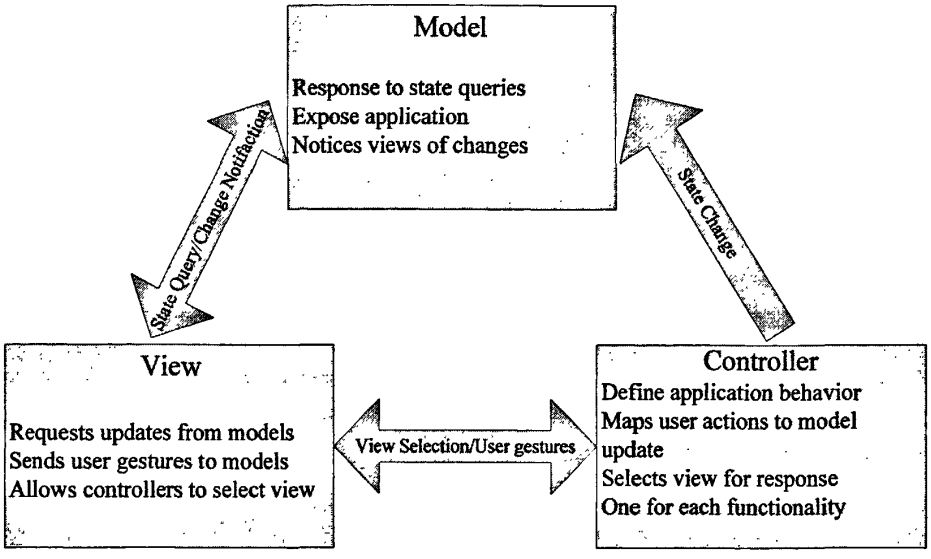



图 3-6 MVC 基本结构图

MVC 通过创建下面三个层将面向对象的设计与可视化接口分开：

- 模型（Model）：Model 代表可视化图元模型，包含完成任务所需要的所有的行为和数据。模型是整个架构的基础，它可以通过有状态的对象记录必要的信息，当模型的状态改变时，模型将会自动刷新与之相关的视图。
- 界面（View）：View 表示图形视图的显示，使用 Draw2d API 的基本元素 Figure 来表示。界面的定义很宽泛，从局部看来包括活动图元、基本连线，从全局看来包括对话框、上下文菜单等。本系统最重要的界面元素就是支持 BPEL4ws 的可视化图形元素。
- 控制器（Controller）：控制器是 view 和 model 之间的桥梁，将模型映射到界面中。控制器处理用户的输入，每个界面有一个控制器俄 ditpart。它负责相应 view 的事件，修改模型数据。控制器的功能在于：监控模型的状态，获取用户输入如鼠标选择、键盘操作，实时更新界面元素。

### 3.2.3 图形元素设计

图形元素通过 Draw2D 来实现，Draw2D 是一个轻量级的图形组件显示系统，纯粹的运用显示，它本身不包含业务模型。所有的图形元素设计为 Figure 的继承类。其中，Figure 是 Draw2d 图形 API 的基本元素。整个 Draw2d 的图形显示视图是由一颗 Figure 树组成的，如图 3-7 所示。

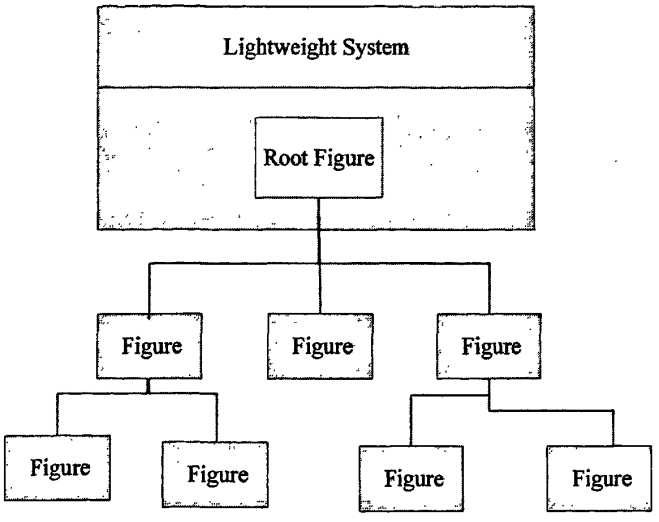



图 3-7 Figure 树结构图

一个 Figure 具有一个矩形区域，用于表示它的显示范围。一个 Figure 可以添加多个子 Figure，但子 Figure 的显示区域都必须在父 Figure 的显示区域之内。除了显示自身的功能，Figure 还应该提供比如连线控制、折点控制、缩放等接口

## 3.3 基于 Eclipse 的可视化工作流程编辑器的设计

可视化编辑器的主要功能是将用户在图形编辑器中对图形元素的操作，直接转化为对业务模型的配置，简化用户操作。其中，用户的操作包括拖拽式添加、删除图形元素、位置调整等。它提供了流程活动元素的图标及操作它们的各种工具，这样用户可以很直观地通过拖放这些元素图标并按一定的规则将它们连接起来，即可编辑出一个流程，然后，还可以对定义好的流程直接进行仿真测试和调试。

为了实现这些功能，本工作流编辑器的功能可分为五个部分，如图 3-8 所示：

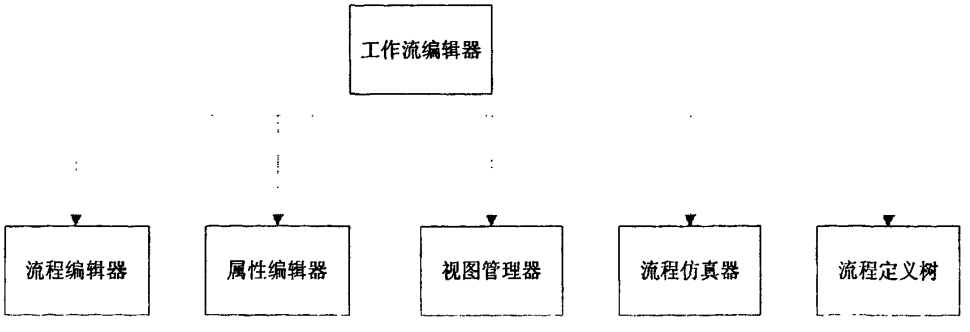



图 3-8 工作流编辑器的功能模块

### 3.3.1 基于 Eclipse 的框架设计

为了分离模型（Model）定义和视图显示（View），实现控制器（EditPart）去改变模型，总体设计采用基于 Eclipse 和 GEF 的 MVC 框架，设计框架图如图 3-9 所示。

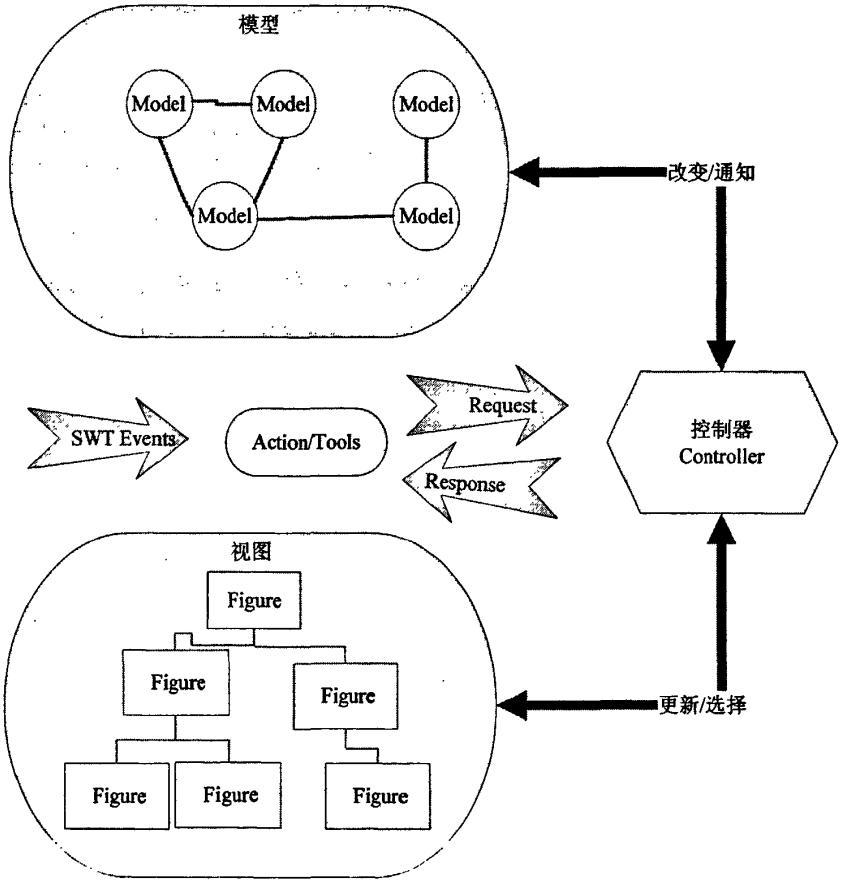



图 3-9 基于 Eclipse 的可视化流程编辑器的基础架构

其中 Model 表示可视化图元模型，它与 BPEL4WS 的活动存在映射关系，是编辑器的核心元素；View 为视图层，其基本元素是 Figure。Figure 是 Draw2D 的基础类，本文中的可视化基本图形元素为 Figure 的子类，是图元模型在编辑器中的可视化存在。本文就是借助 Eclipse 中的 Draw2D/JFace 来实现视图层的；Editpart 是控制器主要功能是监听和处理图元模型属性的变化，同时更新 View。

处理用于在利用控制器去改变模型的时候，利用了策略模式(Policy)。图形的变化不像通常的控件那样，利用添加监听器来回调函数。它把用视图层接收到的用户操作：如添加图元、移动图元、修改图元，都封装为 Request。而 EditPart（控

制器）一直处于监听 Request 的状态，当 Request 到达时，控制器就会根据自身已经安装的策略，调用相应得 Command 处理用户的请求。Policy 能够对 Request 进行一些基本的处理，如图形的创建、删除以及图形的移动。Policy 的功能是管理一组相似的 COMMAND 操作。它会根据用户的 Request，返回相应的 Command。然后由 EditDomain 的使用执行。Command 是 GEF 中的一个抽象类，需要用户根据具体的需求去实现对业务模型的操作。Command 最常见的方法有：execute、redo、undo。这三个方法分别是执行、重新执行以及取消执行。具体流程图，如图 3-10 所示。

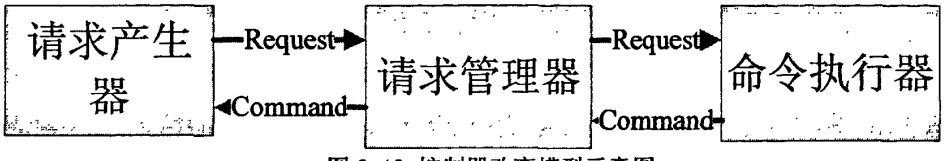



图 3-10 控制器改变模型示意图

下面将从以下几方面完成基于 Eclipse 的可视化流程编辑器的设计：其中包括流程编辑设计、可视化模型的设计、控制器的设计、属性编辑器和文本编辑器的设计。

### 3.3.2 工作流流程编辑的设计

流程编辑的主要功能是将用户在图形编辑器中对图形元素的操作，直接转化为对业务模型的配置，简化用户操作。其中，用户的操作包括拖拽式添加、删除图形元素、位置调整等。其基本的设计架构可分为如下过程：

每一个应用装配编辑器都有一个 EditDomain。它用于执行用户操作产生的命令。同时，EditDomain 有一个 CommandStack(命令堆栈)。用于保存所有用户已经执行的操作，便于撤销和重做。

(1). 当用户在应用装配编辑器中进行图形化的操作时，通过 EditPartViewer 将用户操作的原始消息传递给当前处于激活状态的工具，称为 ActiveTool；

(2). ActiveTool 把用户的原始操作序列转换为 Request 对象。Requets 对象被 GEF 框架使用，用于解释说明需要在业务模型上执行的操作但是不包含具体的操作；

(3). ActionTool 将 Request 对象传给控制器也就是 EditPart，用于获取 Command 或者 Feedback 对象。Command 对象根据 Request，实现了具体修改模型的操作。Feedback 对象用于对用户的操作进行回显。在 EditPart 中，当其接收到 Request 对象时，将 Request 对象传给其安装的 EditPolicy。EditPolicy

用于将 Request 对象构建为 Command 和 Feedback。

(4). 当从 EditPart 获取了 Command 以后，ActiveTool 通过 CommandStack(命令堆栈)执行 Command，完成对模型的修改操作；

(5). 当命令堆栈执行了 Command 并修改了模型以后，将触发对显示视图的更新操作；

(6). 同上述过程一样，Editpart 收到显示视图的更新操作 Request，Editpart 通知对应的 Figure 进行视图更新。

可视化图元模型的设计：

可视化图元模型是可视化工作流编辑器的重要组成部分，一个可视化图元模型可能对应商业流程中的一个基本操作，也可能对应一个业务逻辑。可视化工作流编辑器就是通过改变可视化图元模型的属性来定义一个商业流程。可视化图元模型分为简单图元模型、结构化图元模型及连线图元模型。分别对应 BPEL4ws 的基本活动、结构化活动及 link。

可视化图元模型映射到相应的 BPEL4ws 的活动及其他元素，然后通过活动及它们之间的连接关系来描述企业的业务流程，它定义了业务过程中包含的活动以及这些活动之间的关系。工作流编辑的一个重要过程就是完成对活动属性的设置，这里是通过描述可视化图元模型的属性来完成的，如活动由谁执行，有哪些人员、组织或盟员企业负责执行，活动执行需要的软件(如应用程序)和硬件(如机床设备)资源，以及活动的触发条件、执行状态。结构化图元模型对应 BPEL4ws 的结构化活动，描述了业务流程内部的逻辑关系，用来指定一组活动的执行顺序。这些活动描述了如何通过将基本活动组合为一些特定的结构来创建商业流程。

具体的分析与设计见 3.1，这里就不再赘述。

控制器的设计

控制层的设计和实现是 M-V-C 架构的关键，实现起来工作量也最大。控制层的 Part 与模型层和视图层的 Figure 要一一对应。控制层画布和节点的类图如图 3-11 所示。

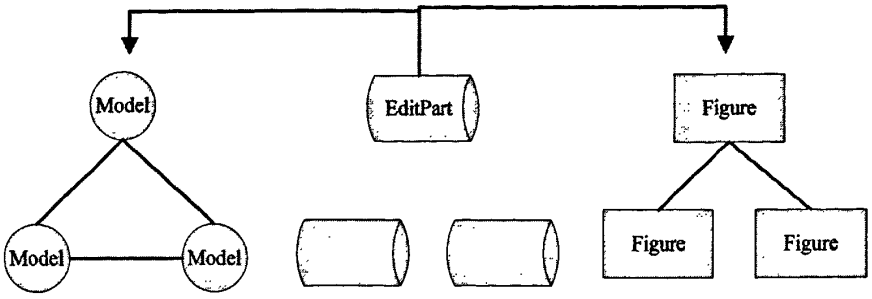



图 3-11 控制层、画布与节点的关系

控制器是流程编辑器的中心枢纽，它控制了视图与模型数据的交互。EditPart 的基类是 AbstractGraphicalEditPart。可以使用 setModel(Object)和 getModel()来获得一个 EditPart 的模型对象。同样，可以通过 getFigure()来获得一个 EditPart 的控制器对象。在工作流编辑器中，需要为每种可视化图元模型创建对应的 EditPart 即控制器，这些 EditPart 继承于 AbstractGraphicalEditPart。主要涉及重载的函数包括：

(1). createFigure() 创建新的图形元素；

(2). refreshVisuals() 当模型改变时，调用该函数进行视图更新；

(3). getContentPane() 获取 EditPart 对应的图形元素的内容面板；

(4). getModelChildren() 获取当前 EditPart 对应模型的子对象数据；

### 3.3.3 属性编辑器的设计

从功能角度来看，属性编辑器的作用是修改图元模型的属性；从底层实现的角度来看,属性编辑器持有选中图元的 Editpart 控制器，并通过 Editpart 控制器来修改图元属性。最后 Editpart 控制器更新对应的 Figure。详细过程分析如下：

- 选中某个图元后，属性编辑器中出现图元对应的属性。
- 选中图元的属性，可以进行修改。其中图元的属性不仅包含简单的文本信息，如图元的 Name，还可能包含复杂的 BPEL 表达式信息，如 Receive 的执行条件表达式。因此属性编辑器需要支持文本修改、表达式编辑等功能。
- 对于复杂的属性编辑还需进行 BPEL4WS 语法检测，比如编辑 Wait 的等待时间表达式。
- 在修改图元属性后，控制器会监听到对应的消息，进而更新表示层 Figure。

通过以上分析，本系统将属性编辑器设计为 IPropertySourceSheet 的子类，它是一个属性注册管理器，它会对实现了 IPropertySource 接口的类进行监听。属性编辑器可以对流程定义树上的非图元节点的属性进行管理。所有的图元模型均实

现了公共的接口和类，具体如下：

属性显示接口 IPropertySource 负责当一个节点被选中时，显示其所有属性。

属性存储类 SaveProperty 负责对节点属性的存储。

属性的克隆接口 Cloneable 用于图元的复制操作。

### 3.3.4 视图管理器

从用户的角度来看，视图是为了方便用户使用流程编辑器而存在的，它的主要功能应该包含：

基本视图功能：工具栏的组织、布局；图元的缩放。

扩展视图功能：流程定义结构树的显示，即大纲视图。

### 3.3.5 流程定义树的设计

流程定义树上的每个节点对于对应一个图元模型，树节点本质上是对图元模型的引用，通过图元模型本身的结构，比如结构化图元包含简单图元的组合，动态生成流程定义树，树的根节点为一个非图元模型 Diagram。大纲视图设计为一个 TreeViewer，ContentProvider 类为树节点内容提供器，主要功能是组织和显示树结构。大纲视图通过实现自己的 ContentProvider 来管理管理图元模型，即可视化图元模型。这个框架如图 3-12 所示。

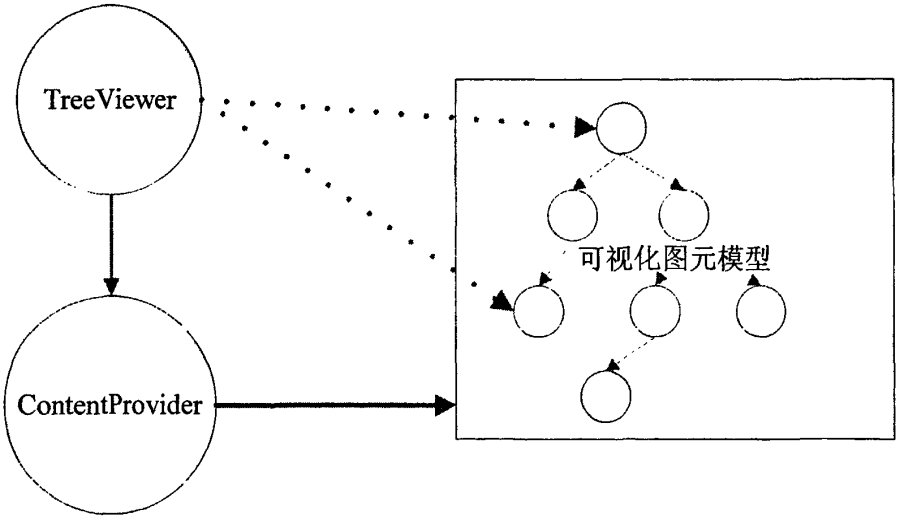



图 3-12 流程定义树的设计

## 3.4 本章小结

本章首先从从应用场景讨论工作流编辑器的抽象层功能，然后得出了设计与实现基于 Eclipse 可视化流程编辑器所必须完成的工作，其中包括 3 个大的方面：基于 BPEL4ws 的工作流模型的分析与设计、基于 BPEL4ws 的可视化图形库的设计和基于 Eclipse 的可视化工作流程编辑器。接着对基于 BPEL4ws 的工作流模型进行分析和属性设计；设计并实现了支持 BPEL4ws 工作流流程定义的图形库；接着并利用 Eclipse 框架，设计了可视化工作流编辑器其他功能模块。在接下来的章节中将对基于 Eclipse 的工作流编辑器的具体实现和具体案例进行讨论。

# 第四章 基于 Eclipse 的可视化工作流编辑器的实现与应用

## 4.1 开发及运行环境

可视化流程设计器开发并运行于Windows平台上，采用Eclipse作为开发工具，系统运转所需数据存储于Oracle数据库中。

## 4.2 可视化工作流编辑器的界面

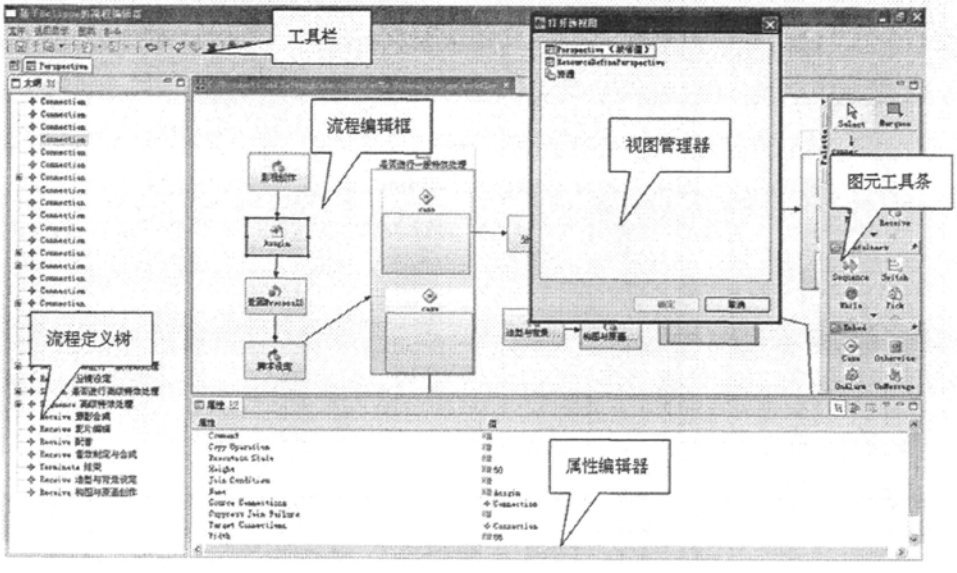



图 4-1 工作流编辑器“流程定义”窗口

## 4.3 流程编辑器的实现

### 4.3.1. 流程编辑器的功能

- 可拖拉式创建组件；
- 可对组件进行选取\删除\复制\粘贴等动作；
- 可进行两组件之间的连线；

- 可创建多种形式的连线(目前包括无箭头\空心箭头\实心箭头);
- 可拖动组件位置;
- 拖动图柄可更改大小(各个角度的图柄更改方式不同);
- 可弹出右键菜单，并可具体组件定制相应菜单;
- 连线的位置随组件的位置而自动改变;
- 每个步聚的撤消/重做;
- 对复合活动的支持;

### 4.3.2. 流程编辑器图形库的实现

图形库的基本图元包括以下 3 类:

简单活动图元:

对应简单图元模型，属于 BPEL 的原子操作，内部不会再嵌套其他图元，包括图 4-2 所述几种，具体功能见本文 3.1 节。

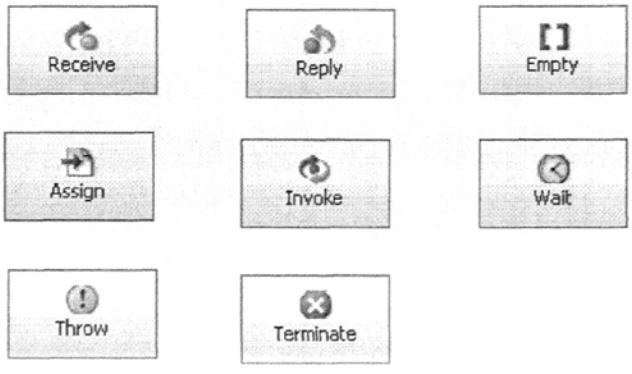



图 4-2 简单活动图元

结构化活动图元:

结构化活动图元指定一组简单活动图元的执行顺序用于定义流程的逻辑层次。这些结构化活动图元描述了如何通过将基本活动组合为一些特定的结构来创建商业流程。这些结构则表达了控制流，数据流，错误和外部事件处理方式以及在业务协议中涉及的消息交换方式。根据控制商业流程执行的方式可以将结构化活动图元分为如图 4-3 所述几种类型。其中可以通过拖拽、Menu 以及复制粘贴操作在结构化活动图元中添加简单活动图元以及其他结构化活动图元。

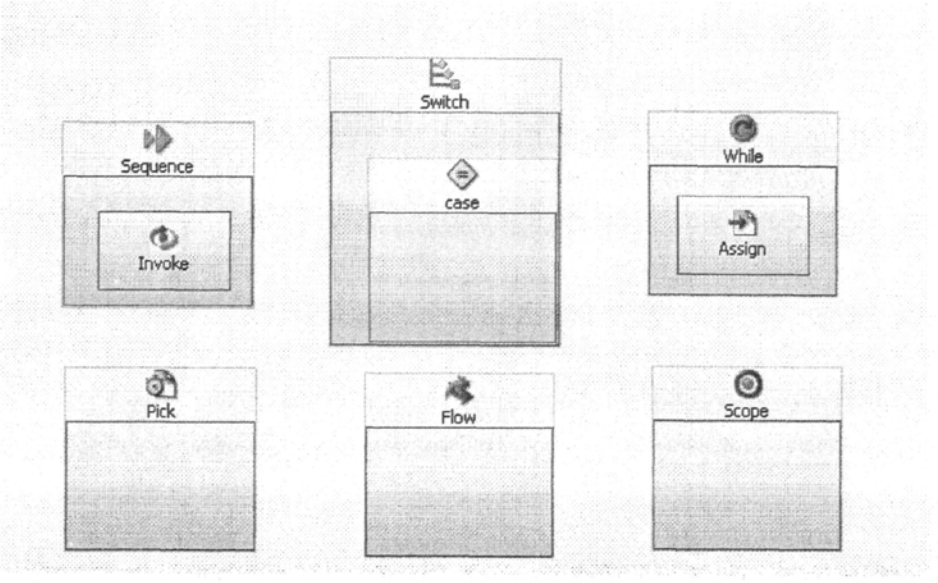



图 4-3 结构化活动图元

Connection：

Connection 用来连接两个其它的活动图元，对应的 BPEL 活动是 link，虽然该元素本身来说没有什么操作， 通常也是一个结构化行为内不可缺少的元素。connection 类中包含两个关键元素，sourceNode 和 targetNode，分别表示 connection 的发起者和接收者。图 4-4 展示了如何用 connection 连接连接两个活动图元。


图 4-4 connection 图元

## 4.3.3. 图元模型编辑的实现

### 4.3.3.1. COMMAND 实现

修改模型属性，建立图元连线，改变图元相对位置等基本操作都是使用 COMMAND 模式来实现的。Command 本来是一种设计模式，以发送一个命令来完成特定的操作，当用户进行鼠标或者键盘操作时，生成的命令会放在一个 commandStack 中。当想取消命令操作时可以对 commandStack 进行操作以恢复到命令执行前的状态。

在本系统中，Command 直接对模型进行操作，而与视图不直接发生联系。虽然模型和 Command 是分开的，但是也可以把 Command 认为是模型的一部分。Command 把对数据特定的复杂的操作封装在一起，其实实现的是模型的功能。工作流编辑器中要实现的 Command 有很多，主要完成在画布中添加、删除和移动图元时对模型的储存和修改的操作。表 4-1 描述了编辑器功能与相关的 Command 操作。

表 4-1 编辑器功能与 Command 操作的对应关系

| 编辑器功能 | 对应的核心 COMMAND 操作 |
|---|---|
| 拖拉式创建组件 | ElementCreateCommand |
| 对组件进行选取\删除\复制\粘贴等动作 | CopyNodeAction<br>CopyNodeCommand<br>PasteNodeCommand<br>ElementDeleteCommand<br>PasteNodeAction |
| 进行两组件之间的连线 | ConnectionReconnectCommand<br>ConnectionDeleteCommand<br>ConnectionCreateCommand |
| 创建多种形式的连线(目前包括无箭头\空心箭头\实心箭头) | ConnectionTypeChangeCommand |
| 拖动组件位置 | AbstractElementSetConstraintCommand |
| 拖动图柄可更改大小(各个角度的图柄更改方式不同); | AsorChangeCommad |
| 弹出右键菜单，并可具体组件定制相应菜单 | ActionCreateAction<br>ActionRetargetCreateAction |
| 连线的位置随组件的位置而自动改变 | ConnectionMoveCommand |
| 每个步聚的撤消/重做 | 所有 COMMAND 实现 REDO UNDO 方法 |
| 对复合活动的支持 | ContainerElementCreateComand<br>ContainerElementEditComand |

| | |
|---|---|
| 用树状图显示绘图区流程图的逻辑关系，根据不同的图元显示不同的节点 | 接口 ISelectionListener 的方法 selectionChanged（） |

程序中实现的 Command 要继承于 eclipse.gef.commands.Command,Command 中上要实现的方法主要有 execute()和 undo()，当 Command 被调用时主要执行 execute()方法，当要撤销 Command 操作时执行 undo()方法。

#### 4.3.3.2. 控制器的实现

EditPart(控制器)的基类是 AbstractGraphicalEditPart。可以使用 setModel(Object)和 getModel()来获得一个 EditPart 的模型对象。同样，可以通过 getFigure()来获得一个 EditPart 的控制器对象。流程编辑器抽象的所有模型对象都从 AbstractGraphicalEditPart 基类继承而来。要实现对应的 EditPart，需要重载以下函数：

protected Ifigure createFigure() 为一种类型的模型对象创建一个新的图形元素；

public void refreshVisuals() 使用与当前 EditPart 对应的模型对象数据更新当前图形元素的显示信息；

protected Ifigure getContentPane() 返回当前 EditPart 对应的图形元素的内容面板；

protected List getModelChildren() 返回当前 EditPart 对应模型对象的子对象列表。这些子对象将作为上面内容面板的显示数据。

## 4.4 属性编辑器的实现

### 4.4.1. 属性编辑器的基本功能

- 图元选择；
- 通过属性视图显示图元属性；
- 可以在属性编辑器中修改图元属性，并实时更新到流程定义树和流程编辑框
- 对仿真属性和一些高级属性的自动检查；
- 对 BPEL 表达式的编辑；

### 4.4.2. 属性编辑

图元具有各种不同的属性，具体图元所具有的属性参考 3.1 节。属性编辑需要实现对不同属性的处理，属性可以分为简单文本属性和复杂 BPEL 表达式属性：(1) 简单文本属性：直接在属性列表里面进行编辑，或者使用文本对话框进行编辑。(2) 复杂 BPEL 表达式属性。

BPEL 表达式的编辑

BPEL 表达式有 3 种类型，不同的表达式使用不同的树型对话框进行编辑：

- 布尔表达式：用于激活 link 所需的 transitionCondition、执行某个行为之前要检查该行为的 joinCondtion、while 活动的循环条件判断 condition、switch 中每个 case 的进入条件判断 condition 以及 catch 活动的 condition 等等。
- 持续时间表达式用于 wait 行为中、pick 行为的 onAlarm 子元素中、以及 Event Handler 的 onAlarm 子元素中的 for 属性里面。
- 截止时间表达式用于 wait 行为中、pick 行为的 onAlarm 子元素中以及的 Event Handler 的 onAlarm 子元素中的 for 属性。任何支持时间参数的行为都支持持续时间表达式和截止时间表达式。

表达式求值

Xpath 1.0 作为支持 BPEL4WS 的表达式语言，其特点是可以分析 XML 文档，从中提取属性，查询数据。XPath 1.0 函数功能并不完整，为了使它的表达式能从流程中获取信息，BPEL4WS 把几个扩展函数增加到 XPath 的内置函数。在 BPEL4WS 的标准名称空间（http://schemas.xmlsoap.org/ws/2002/07/business-process/）中有这些扩展函数的定义。前缀"bpws:"与这个名称空间关联。新增的三个扩展函数：

- bpws:getVariableData (‘variableName', 'partName', 'locationPath'?)
- bpws:getVariableProperty (‘variableName', 'propertyName')
- bpws:getLinkStatus ('linkName')

如果前两个函数选出的节点数不唯一，就会抛出故障 bpws:selectionFailure。为了在系统中提供这些扩展函数的功能，在代码中编写了一些扩展类。图 4-6 展示了对某个模型 JoinCondtion 属性的 BEPL 表达式编辑。

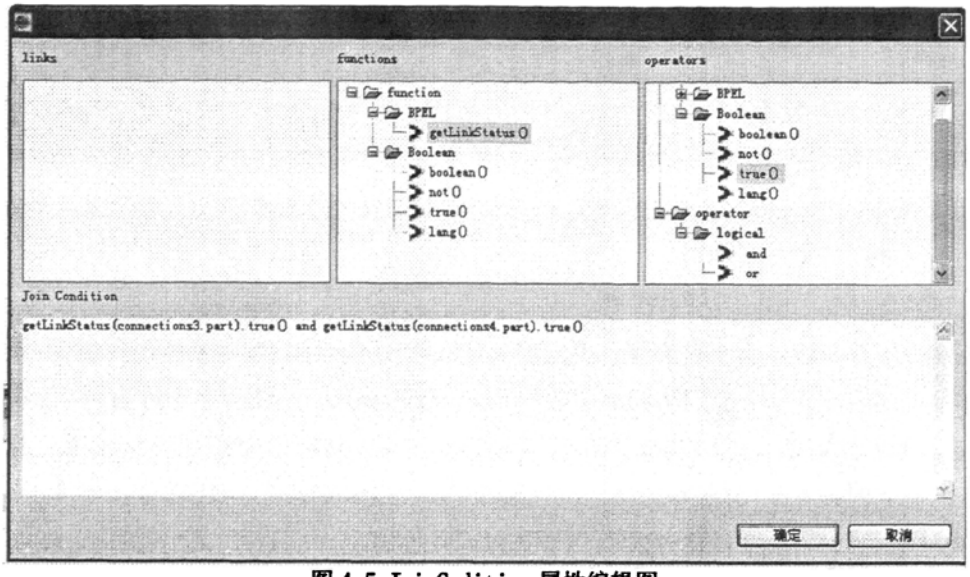



图 4-5 JoinCodition 属性编辑图

### 4.4.3. 属性编辑流程

图 4-3 展示了如何在属性视图中编辑属性，及如何实时更新对应的图形元素。

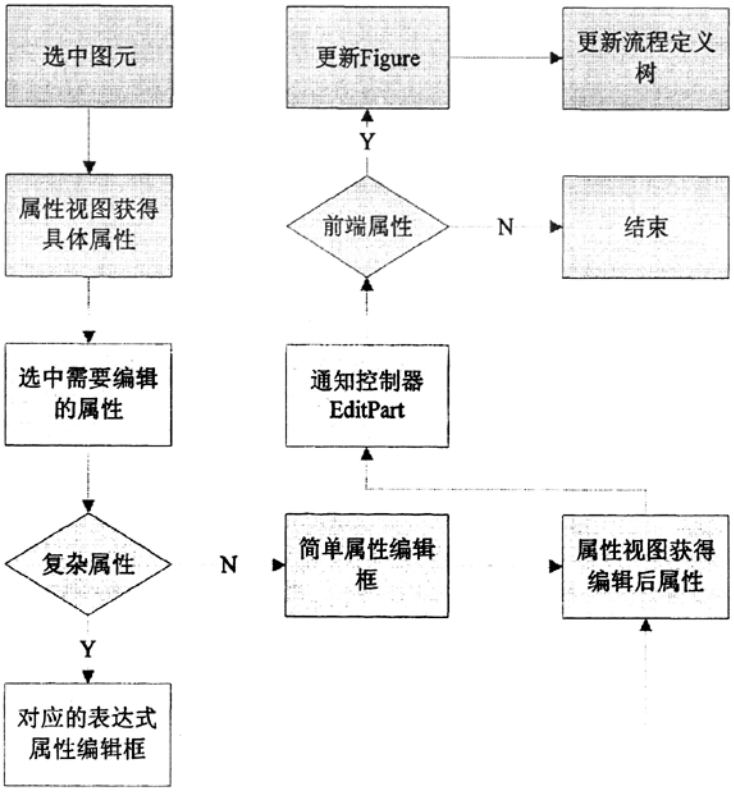


图 4-6 属性编辑流程

Itemprovider 是属性编辑中最重要的一个类，它要为图元模型提供 contentProvider，即内容显示器。下面以 Receive 为例，简单阐述对应的 Itemprovider-ReceiveItemProvider 的实现，如图 4-7 所示。ReceiveItemProvider 是 Receive 的内容提供器，当在编辑器中选中 Receive 图元时， ReceiveItemProvider 通过接口 List getPropertyDescriptors(Object obj)返回具体的属性，而属性编辑器是 PropertySheetPage 实现类，可以获得具体的 Receive 图元属性。属性视图 PropertySheetPage 显示图元属性。之后选中需要编辑的属性，属性视图 PropertySheetPage 根据其注册工厂进行处理：

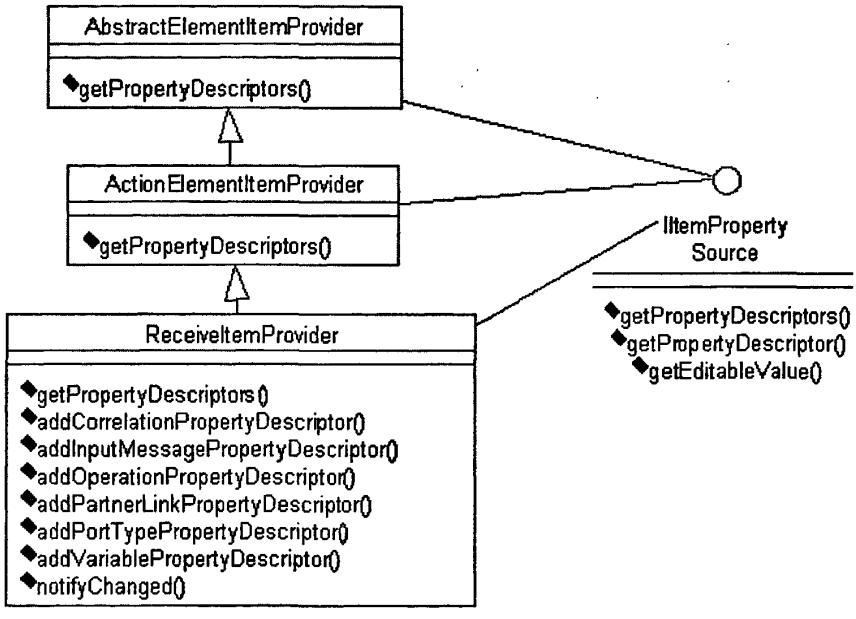



图 4-7 ReceiveItemProvider 类的设计

## 4.5 资源管理器的实现

资源管理器的基本功能：

- 图元的缩小与放大；
- 整体布局的调整；
- 多视图的管理；
- 工具栏的布局；

资源管理器的前 3 种功能在案例中会有所介绍，这里主要介绍工具栏布局的实现。工具栏是方便使用者的工具组合，相当于现实中的工具箱，因此工具在工具栏中如何摆放应该可以根据需要灵活定制，本系统实现了几种工具布局，如图 4-8 所示。

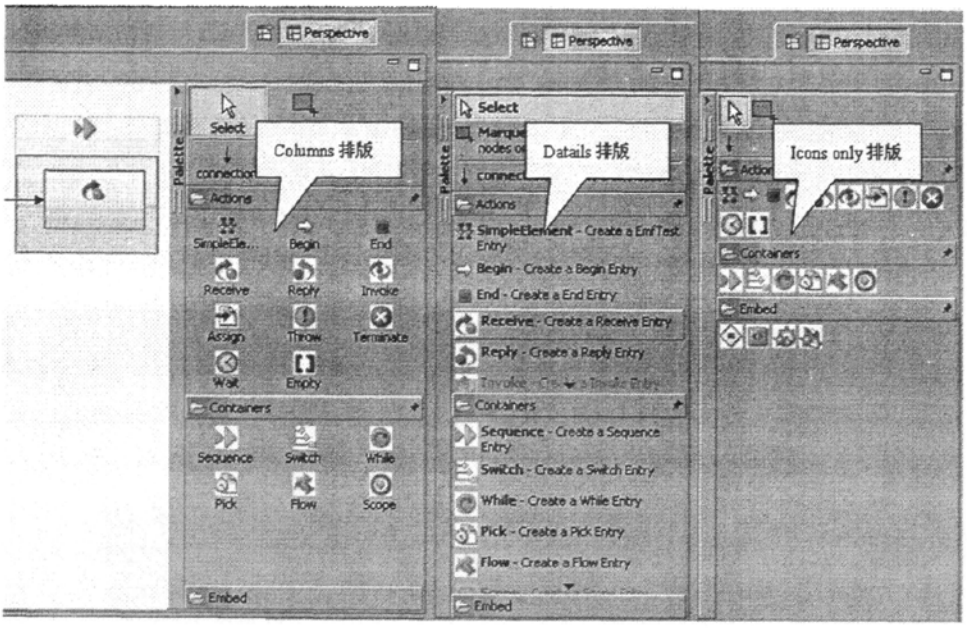



图 4-8 工具栏布局

左边的是图标样式的工具栏布局，中间的是详细信息样式的工具栏布局，右边的是简单样式的工具栏布局。当然，工具的种类不限于图中所示，可以通过 MENU 菜单灵活添加和删除。工具栏中默认的工具如表 4-2 所示。

表 4-2 工具栏中的默认工具

| 工具名称 | 包含的子元素 | 功能 |
|---|---|---|
| Select | 无 | 单选 |
| Marque | 无 | 多选 |
| Actions | Receive Reply Invoke Assign Throw Terminate Wait Empty | 在流程编辑器中创建简单活动图元 |
| Container | Sequence Switch While Pick Flow Scope | 在流程编辑器中创建结构化活动图元 |
| Embed | Case Otherwise OnAlarm OnMessage | 为结构化活动图元创建合适的子元素 |

| | | |
|---|---|---|
| Connection | Conncetion | 为活动图元之间建立联系 |

## 4.6 流程定义树的实现

### 4.6.1. 流程定义树的功能

- 显示流程定义树；
- 选取树结点，对于图元选中；
- 根据模型属性，实时更新；
- 支持树型操作，比如展开、收拢；
- 可调整树的显示格式，比如字体大小、颜色、树节点图标；

### 4.6.2. 大纲视图

大纲视图作为流程定义树的容器，主要功能是显示流程定义树。大纲视图设计为 OutLineView，如图 4-9 所示。

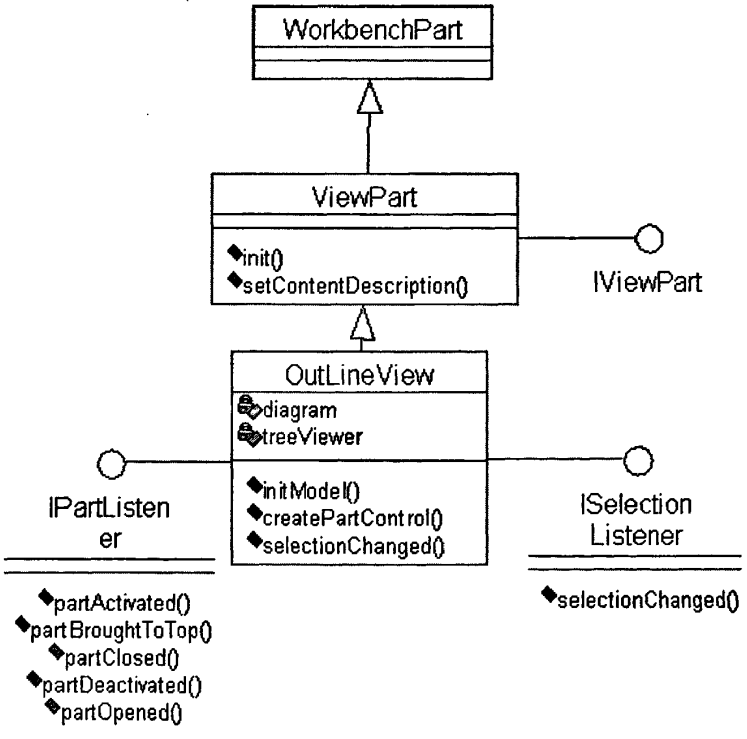


图 4-9 大纲视图的设计类图

大纲视图在流程编辑器的配置文档 plugin.xml 中的配置如下：

```
<extension point="org.eclipse.ui.views">
    <view
        name="大纲"
        class="cn.uestc.workflow.views.OutLineView"
        id="cn.uestc.workflow.views.OutLineView">
    </view>
</extension>
```

### 4.6.3. 流程定义树的显示

把模型对象的根元素，即 Diagram 传递给大纲视图 OutLineView，大纲视图持有一个 EditPartViewer，用于控制树型元素的显示，显示过程如图 4-10 所示：

- 大纲视图根据 EditPartFactory,创建对应模型对象的 EditPart；
- 使用 createFigure()和 refreshVisuals()方法建立控制器对应的图形元素，更新图形元素的属性；
- 使用 getModelChildren()，根据返回的子元素列表，决定作为当前控制器图形元素的内容面板的显示内容；
- 对于 getModelChildren()返回的对象列表，根据控制器工厂创建对应的控制器，并且将新创建的控制器添加为当前控制器的儿子节点；
- 对于没有儿子节点的控制器，使用 createFigure()和 refreshVisuals()方法，创建对应的图形元素并且进行更新。然后将它们添加到对应父图形元素的内容面板中；
- 使用 getModelChildren()建立没有个 EditPart 的儿子节点，直到整个流程定义树建立完成；

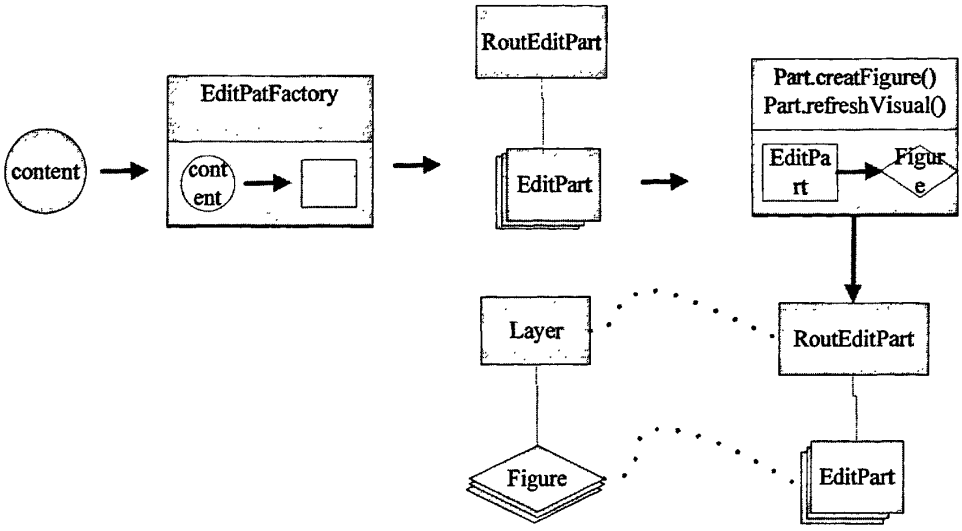



图 4-10 流程定义树的显示过程

## 4.7 应用实例

### 4.7.1. 使用基于 Eclipse 的可视化工作流编辑器的建模流程

用户使用基于 Eclipse 的可视化工作流编辑器建立流程，其流程图如图 4-11 所示，主要包括九个步骤:

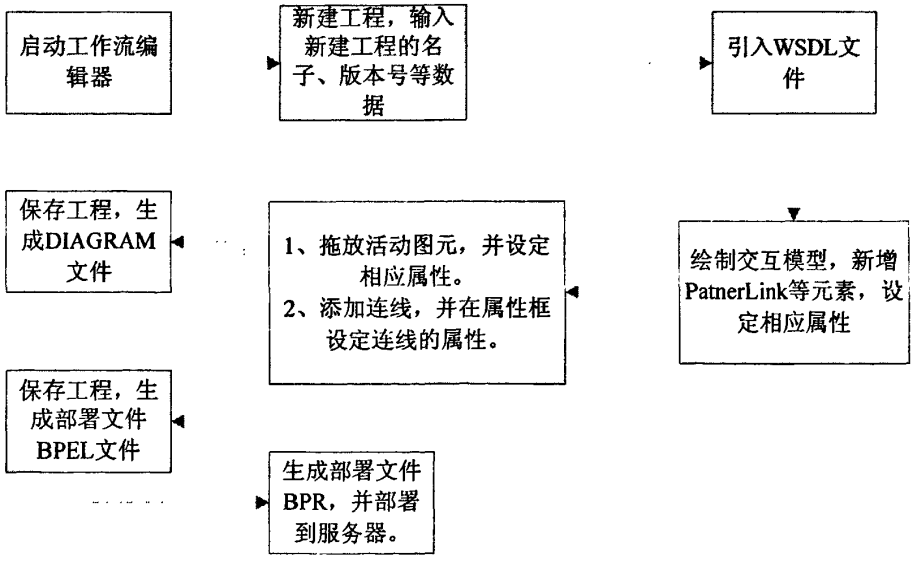



图 4-11 工作流编辑器编辑流程的步骤

首先要理解目标领域的商业逻辑，然后要明确需要流程需要实现的功能，在

WebService 运行环境下可以用哪些 WebService 来完成这项功能，从而决定采用什么方式进行服务选取。确定服务之间的消息传递类型和传递方式，从而决定如何定义活动的属性。充分思考商业流程运行过程可能出现的错误，当出现了错误时，如何进行捕获和处理，从而定义 Event Handler 和 Compensation Handler。在这些工作完成后，就可定义流程中的所有活动，包括每个活动的属性(如名称、活动类别、发送消息或存储消息的变量等)、流程执行顺序。在建模完成后，进行保存再生成 BPEL4WS 文档和本地化图元模型文档。下面以具体的应用实例说明基于可视化工作流编辑器的建模过程和流程编辑结果，体现本系统的易用性和实用性。

### 4.7.2. 项目应用

下面我们将通过一个实例来展示该流程设计器的功能。

图 4-12 是“支持数字媒体内容创作的集成环境”863 项目的具体业务流程，影视创作流程，这里影视主要指关于动漫的影视。其中包括脚本设定、分镜设定、造型与背景设定、构图与原画创作、中间画面与分层设定、上色、特效制作、摄影合成、影片编辑、配音、音效制作与合成等活动。可以看出该流程是一个较为复杂的流程。通过对本流程进行建模，可以较好地展现本编辑器的各种功能。

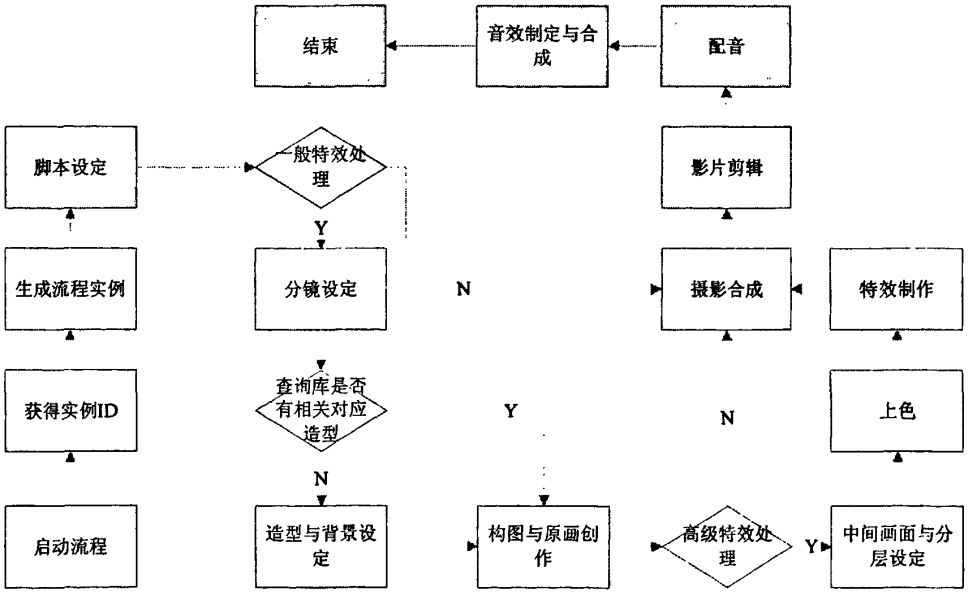



图 4-12 影视创作流程

首先，新建一个工程，取名为“影视创作（动漫）”。

然后，对流程建模所需信息进行设置。如 partnerLink，variable 等。

接着，对流程活动节点属性进行设置是最重要也是最繁琐的一步。图 4-13 显示的是对其中一个 assign 活动节点的 Copy Operation 进行属性设置时的状况。图 4-14 显示了对其中一个 case 活动节点的 case Expression 进行属性设置的情况，其他属性设置就不再这里一一赘述。

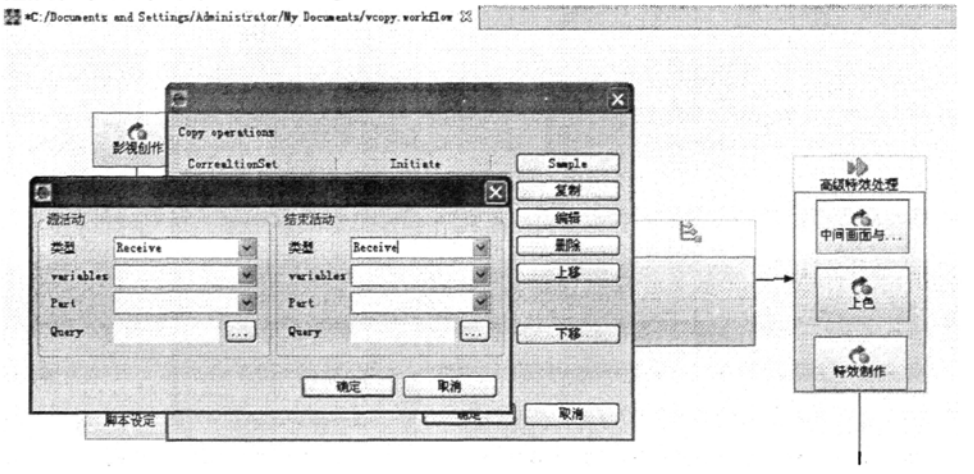



图 4-13 assign 活动节点的 Copy Operation 进行属性设置

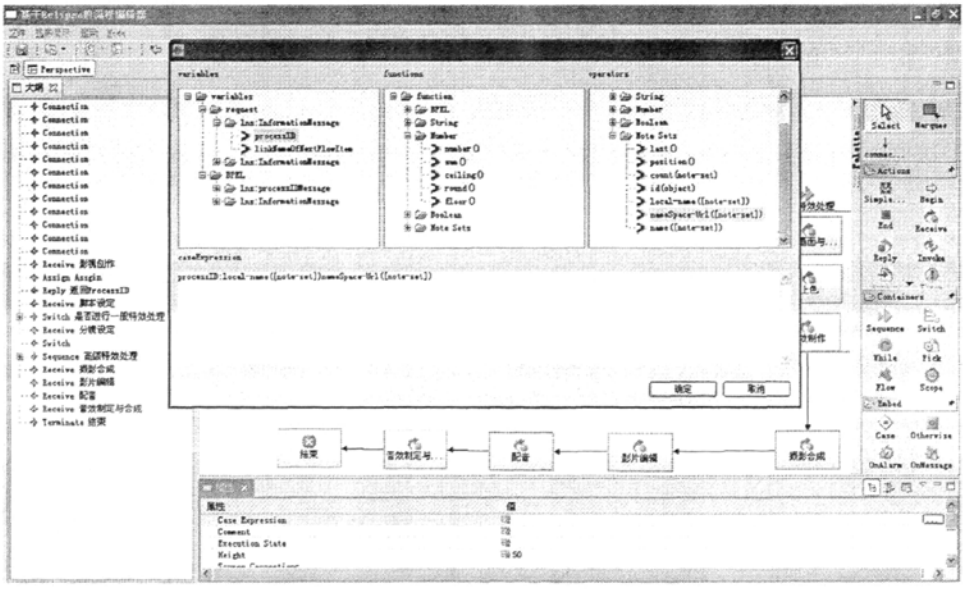



图 4-14 case 活动节点的 case Expression 进行属性设置

编辑完成后的界面如图 4-15 所示：

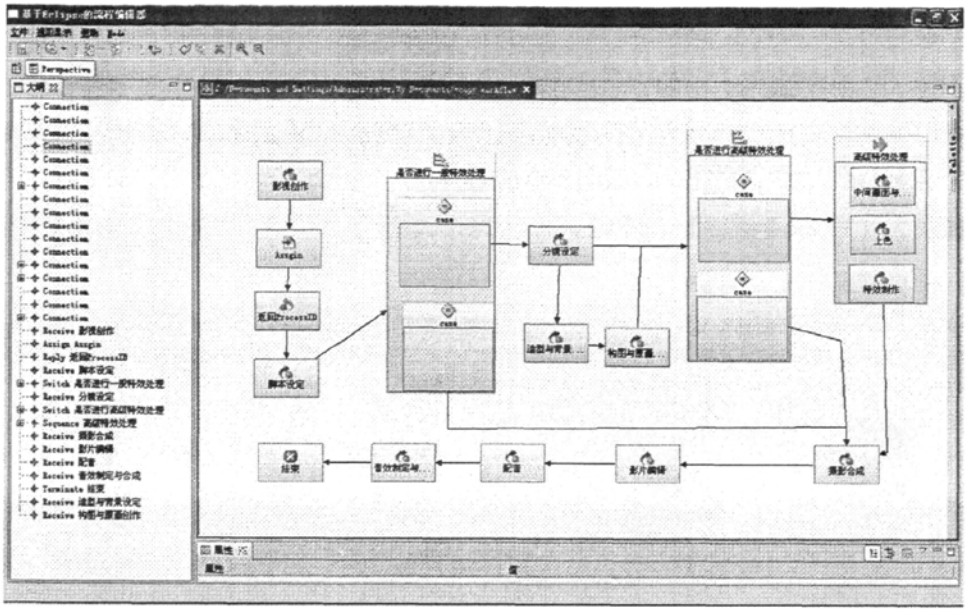



图 4-15 影视创作流程

流程设计完成后，可查看生成的项目资源，包括.bpel 文件，.wsdl 文件以及.workflow 文件等。图 4-16 为生成的.workflow 文件的部分内容，图 4-17 为生产的.bpel 文件的部分内容。

```
  </connections>
  <connections targetNode="//@elements.13" sourceNode="//@elements.5"/>
  <connections targetNode="//@elements.14" sourceNode="//@elements.13"/>
  <connections targetNode="//@elements.6" sourceNode="//@elements.14">
    <bendPoint sourceX="4" sourceY="-131" targetX="-132" targetY="8"/>
  </connections>
  <elements xsi:type="cn.uestc.workflow:Receive" sourceConnections="//@connections.0" X="77" Y="79" name="影视创作"/>
  <elements xsi:type="cn.uestc.workflow:Assign" targetConnections="//@connections.0" sourceConnections="//@connections.1" X="75" Y="167"
name="Assgin"/>
  <elements xsi:type="cn.uestc.workflow:Reply" targetConnections="//@connections.1" sourceConnections="//@connections.2" X="73" Y="247" name="返回
ProcessID"/>
  <elements xsi:type="cn.uestc.workflow:Receive" targetConnections="//@connections.2" sourceConnections="//@connections.3" X="72" Y="334" name="脚
本设定"/>
  <elements xsi:type="cn.uestc.workflow:Switch" targetConnections="//@connections.3" X="248" Y="67" name="是否进行一般特效处理">
    <childrenElements xsi:type="cn.uestc.workflow:Case" sourceConnections="//@connections.11" X="14" Y="25"/>
    <childrenElements xsi:type="cn.uestc.workflow:Case" sourceConnections="//@connections.12" X="19" Y="159"/>
  </elements>
  <elements xsi:type="cn.uestc.workflow:Receive" targetConnections="//@connections.11" sourceConnections="//@connections.4 //@connections.14"
X="439" Y="161" name="分镜设定"/>
  <elements xsi:type="cn.uestc.workflow:Switch" targetConnections="//@connections.4 //@connections.16" X="658" Y="36" name="是否进行高级特效处理">
    <childrenElements xsi:type="cn.uestc.workflow:Case" sourceConnections="//@connections.5" X="12" Y="22"/>
    <childrenElements xsi:type="cn.uestc.workflow:Case" sourceConnections="//@connections.13" X="18" Y="148"/>
  </elements>
  <elements xsi:type="cn.uestc.workflow:Sequence" targetConnections="//@connections.5" sourceConnections="//@connections.6" X="846" Y="44" name="
高级特效处理">
    <childrenElements xsi:type="cn.uestc.workflow:Receive" X="28" Y="6" name="中间画面与分层设定"/>
    <childrenElements xsi:type="cn.uestc.workflow:Receive" X="28" Y="68" name="上色"/>
    <childrenElements xsi:type="cn.uestc.workflow:Receive" X="18" Y="131" name="特效制作"/>
  </elements>
  <elements xsi:type="cn.uestc.workflow:Receive" targetConnections="//@connections.6 //@connections.12 //@connections.13"
sourceConnections="//@connections.7" X="820" Y="443" name="摄影合成"/>
  <elements xsi:type="cn.uestc.workflow:Receive" targetConnections="//@connections.7" sourceConnections="//@connections.8" X="555" Y="443" name="
影片编辑"/>
  <elements xsi:type="cn.uestc.workflow:Receive" targetConnections="//@connections.8" sourceConnections="//@connections.9" X="375" Y="442" name="
配音"/>
```

图 4-16 workflow 文件的部分内容

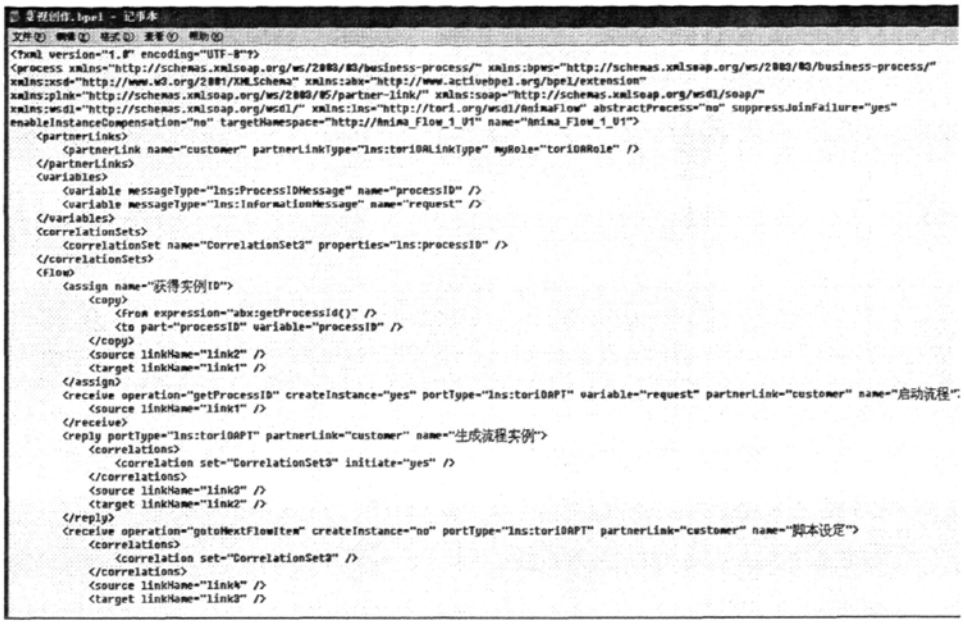



图 4-17 bpel 文件的部分内容

完成后的设计工作流程如图 4-18 所示：

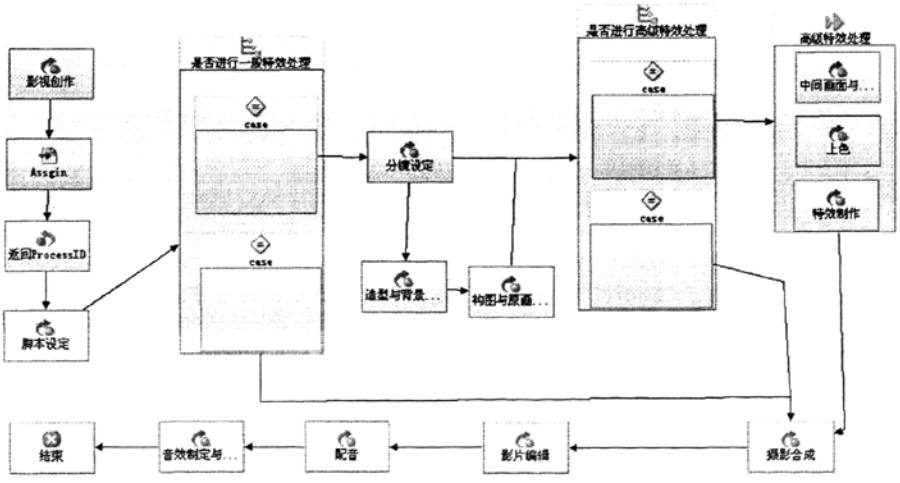
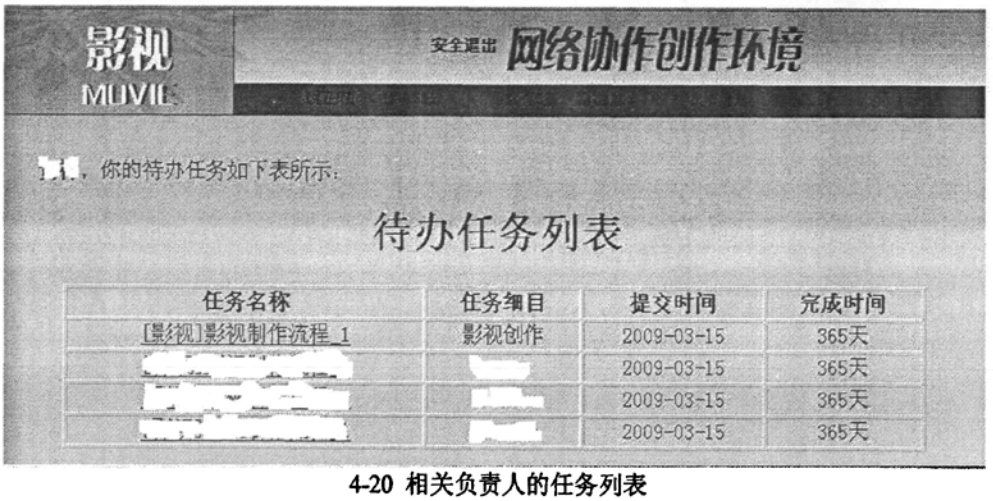



图 4-18 完成后的影视创作流程

确认流程无误，可执行模型检查和 BPEL 文件检查，然后部署到工作流服务器上去，如图 4-19 由管理员进入启动流程界面，图 4-20 表示项目负责人得到通知，需要启动“影视创作流程”，由于保密性需要，屏蔽了项目负责人姓名和其他任务信息。下一步需进行的工作就是通过流程的日常运行，不断发现流程中存在的问题，完善流程的完整性和健壮性。


4-19 流程启动管理




4-20 相关负责人的任务列表

## 4.8 本章小结

本章在上一章对可视化工作流程编辑器的设计方案的基础上，首先介绍了可

视化工作流程编辑器的运行环境和界面。然后，介绍了实现属性编辑器、资源管理器、流程编辑器、.bpel 文件的生成及可视化工作流图元的本地化过程的关键技术。其重点是流程编辑器采用 COMMAND 模式对模型的修改，以及通过控制器完成对视图层的实时更新。节点信息的保存。流程定义树的生成过程。最后，通过具体的案例，介绍了工作流程编辑器对流程编辑、.bpel 文件的生成和流程部署的工作过程。通过以上操作过程，充分展示了本系统的可用性和易用性。

# 第五章 全文总结

## 5.1 本论文研究总结

工作流技术是实现企业业务过程重组、过程管理的企业信息化技术，最初用于生产过程控制及办公自动化领域，后来在企业的业务流程管理领域得到充分的应用。虽然工作流产品和技术日新月异，但是就目前的工作流技术而言，仍然有一些明显的缺陷，主要缺陷在于：异种工作流产品之间的缺乏交互机制，工作流建模技术不够成熟，数据规范性得不到统一。本文通过研究国内外工作流的关键技术，将工作流技术与 Web 服务体系架构技术紧密地结合，提出了一种基于 BPEL4ws 的业务流程建模和编辑方案，弥补了传统工作流管理中流程编辑技术的异构环境支持不足、耦合性大等问题。系统采用 MVC 框架结构，分离了模型和业务逻辑从而保证了系统的松散耦合性，快速地通过可视化流程编辑，订制复杂商业逻辑，满足企业业务的快速变化的需求。现将本文的主要工作和创新点总结如下：

(1). 对工作流技术的研究进行了综述，分析了当前企业或部门对计算机化表示的业务流程的迫切需要。然后分析了 Eclipse 及 GEF 的框架体系。接着从 Web 服务的内容、功能、工作原理等方面介绍了 Web 服务技术。最后重点分析了 BPEL4WS 规范的基本结构，BPEL4WS 中抽象流程与具体流程的关系以及 BPEL4WS 活动的语义和限制。

(2). 分析了在工作流管理系统中，如何以计算机能够处理的形式进行业务流程的定义。继而明确设计与实现工作流流程编辑器所必须掌握的关键技术和突破关键难点：采用基于有向图的建模方法和建立支持 BPEL4ws 的可视化图形库。然后基于 Eclipse 的体系结构设计并实现了支持 BPEL4ws 工作流流程定义的图形库。最后在设计整体框架时，采用MVC框架分离了模型（Model）定义和视图显示（View），实现控制器（EditPart）去改变模型。

(3). 最终设计并实现了一个支持 BPEL4ws 的基于 Eclipse 框架的可视化流程编辑器，通过具体的案例演示了本系统的具体工作过程，证明了本系统的可用性和易用性。

## 5.2 进一步工作

(1). 为.bpel 、.wsdl、.workflow 文件提供一个可视化的元素编辑环境。

(2). 加强对模型检查的支持，本系统对模型检查的支持不够，即出现问题后，错误信息定位不是很准确。效率和安全性方面仍然存在不足，这也是今后需要研究的工作。

# 致 谢

夜阑人静，再次整理论文时让我思绪万千。在研究生期间，我感悟了许多，学到了很多。我的导师谭浩副教授一直给了我很大的帮助，在这里我要感谢您，是您让我度过了有意义的三年，是您让我感受到了朴实向学的精神，是您在我深陷知识泥潭的时候帮助了我，不管是设计模式、还是基础的 Java 编程，我还有很多东西需要向您学习，可惜毕业的时候就要到了。不过，你的宽厚、睿智可以让我用一生来学习。

谢谢我的爸爸妈妈，谢谢您们的辛劳；谢谢郑智潇，你是我坚持的源泉；谢谢我的朋友们，你们一直是我最坚强的后盾，我们在学习中相互了解，在合作中成长。我永远不会忘记我可爱的朋友们：小马的快乐，李子民稳重，陈渝的创新、张伟的合作、刘建的大气、吴艳艳的热情、虞奕霖的睿智、谢菲的美丽。。。

雷宏岩师弟，感谢你在我写论文这段时间给与的帮助；还有杨新英师妹，多谢你在算法课上的技术支持。

# 参考文献

[1] 范玉顺. 工作流管理技术基础. 清华大学出版社, 2001

[2] Amadio RM, Castellani I, Sangiorgi D. On bisimulations for the asynchronous pi-calculus. In: Montanari U, Sassone V, eds. Proceedings of the 7th International Conference on Concurrency Theory. Pisa, Italy, 1996. LNCS 1119, Berlin: Springer-Verlag, 1996. 147 ~ 162.

[3] Victor B, Moller F. The mobility workbench: a tool for the pi-calculus. In: Dill D L, eds. Proceedings of the 6th International Conference on Computer Aided Verification. California, 1994. LNCS 818, Berlin: Springer-Verlag, 1994. 428 ~ 440.

[4] van der Aalst W, Dumas M., eds. Choreography Conformance Checking: An Approach based on BPEL and Petri Nets (extended version). BPM Center Report BPM-05-25, BPMcenter.org, 2005.

[5] 汪涛, 黄力芹, 吴耿锋. 工作流管理的发展历程和趋势. 计算机工程与科学, 2001, Vol.23

[6] 周洪建. 基于SOAP的分布式Web Service的应用研究. 微机发展. 2003年10期

[7] Workflow Management Coalition. XML Process Definition Language, Document No. WFMC-TC-1025, 2002

[8] Workflow Management Coalition, Terminology and Glossary, WFMC-TC-1011. See http://www.wfmc.org

[9] Gamma E, Beck K. Contributing to Eclipse:Principles, Patterns, and Plug-ins[M]. Addison Wesley, 2003

[10]魏子鹏. BPEL商业流程建模. 科技情报开发与经济, 2004, 14

[11]Workflow Management Coalition, The Workflow Reference Model, Document Number TC00-1003.

[12]工作流管理联盟. http://www.wfmc.org

[13]Nirmal Mukhi. 使用BPEL4WS的业务流程：学习BPEL4WS. http://www-2.900.ibm.com/developerWorks/cn/webservices/ws-bpel/part8/index.shtml, 2002

[14]BPEL4WS 语 言 规 范 1.1.http://www-106.ibm.com/developerworks/library/ws-bpel

[15]柴晓路，梁雨奇 Web服务技术、架构和应用 北京电子工业出版社，2003

[16]阳红星,彭志红.基于Web的XML Web Service的研究.华东地质学院学报,2003年4期

[17]XML Base W3C Recommendation. http://www.w3.org/

[18]Seely S. SOAP：XML跨平台Web Service开发技术[M] .北京机械工业出版社,2002

[19]JDOM API Documentation.http://www.jdom.org

[20]Mendling J, Hafner M. From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL. In: Meersman R, Herrero P, eds. Proceedings of OTM 2005 Workshops. LNCS 3762, Berlin: Springer-Verlag, 2005. 506~515.

[21]Overdick H, Puhlmann F, Weske M. Towards a Formal Model for Agile Service Discovery and Integration. In: Bussler C. Proceedings of the ICSOC Workshop on Dynamic Web Processes, Amsterdam, Netherlands, 2005.

[22]G Wirtz, M Weske, H Giese. The approach to workflow modeling in object oriented system. Information System Frontiers,2001, 3(3):357-375

[23]飞思科技产品研发中心编著.Java Web 服务应用开发详解.电子工业出版社,2002

[24]Erich gamma,Richard Helm,Ralph Johnson,等著.李英军,马晓星,蔡敏,等译.设计模式——可复用面向对象软件的基础.机械工业出版社,2000

# 攻硕期间取得的研究成果

[1] Shi Yi-Feng, Tan-Hao, Ma Rong-fang, "Study on Adaptive Task Assignment Method in Media Context Invention", in IEEE International Conference on Apperceiving Computing and Intelligence Analysis 2008,ICAACI'08 p199~200(EI, ISTP, IEEE Xplore 收录)

[2] 支持数字媒体内容创作的集成环境，国家 863 项目

[3] 基于 NTP 的网络授时系统，成都第十研究所项目