分类号 密级

U D C 编号

中南大學

CENTRAL SOUTH UNIVERSITY

硕士学位论文

论 文 题 目	柔性工作流的研究
学科、专业	计算机应用技术
研究生姓名	徐海军
导师姓名及专业技术职务	李建华 教授

2005 年 5 月

摘　要

工作流技术是实现企业业务过程自动化的核心，有助于提高企业的生产率与竞争力。狭义的说，工作流的柔性是指工作流系统执行松散、部分定义的工作流程的能力，工作流定义往往在运行时才完全确定。广义上还包括工作流技术的可扩展性、重用性及与现有应用于企业的其他 IT 技术的融合能力。

本文在总结前人对工作流柔性的研究、理解与定义的基础上，分别尝试了从理论、工程及应用等不同角度分析了工作流柔性并给出了一个较详细的界定。

提出将根据其生命周期分为建立、配置与运行阶段，并在此基础上提出了工作流建立-配置-运行柔性框架，在深入研究工作流管理联盟元模型和流程定义语言 XPDL 标准的基础上，讨论了柔性框架下可配置的内容、配置的基础理论、分类、规则、机制及其实现。

由于目前对工作流定义语言元模型的扩展存在不兼容的缺陷，本文对工作流管理联盟的流程定义元模型的基础上兼容地进行了柔性扩展，扩展后的流程定义语言 XPDL 支持空子流程及动态绑定、流程跳转以及五种控制关联。

同时在研究多个开源工作流定制工具和引擎的基础上，选择了定制工具 JaWE 和引擎 Shark 组成工作流管理系统作为改进的基础。深入地研究了 JaWE 和 Shark 的源代码，并在此基础上完成了支持工作流柔性框架的 WfMS 原型。该系统具有运行不确定性定义流程的能力、较好的松散耦合性及工作流定义的重用性。

本文改进工作流柔性的方法与同类方法相比，具有兼容性好、非程序化等优点。最后提出了一个能够反映工作流柔性核心的评价标准并对改进前后的工作流系统进行了对比分析，指出改进后的原型系统在可变的流程定义、重用性以及动态绑定等方面比原有系统更具柔性。

最后指出，工作流柔性与其对业务流程自动化程度的提高是一对必然的矛盾。因此在追求工作流系统柔性的同时，必须把握好柔性与自动性及效率之间的关系，于二者之间折衷以达到最好的效果。

关键词：工作流，柔性，XPDL，松散耦合，重用性

Abstract

Workflow is the core technology of achieving enterprise business process automatization,which helps to improve the enterprise's productivity and competition. In narrow sense,the flexibility of workflow means the ability of Workflow Management System(WfMS) to execute process on the basis of loosly and partially defined model while the full specification may be made at runtime.It also include the extensibility, reusability and the ability to integrate with other IT technology currently applied in enterprises in a broad sense.

This paper respectively tried to analyse workflow from different viewpoint such as theory,engineering and application and give a definition in comparatively detail on the basis of summarizing others researches,understandings and definitions.

The life cycle of workflow is divided into build stage,configuration stage and execution stage, and this paper present a flexible frame of workflow build-configuration-execution on this basis. configurable content, configuration theory, category, rules, mechanism and implementation under the flexible frame has been discussed on the basis of lucubrating on Workflow Management Colation's specification of mata model and XML-based process definition.

Due to current extend to the workflow definition meta model is lack of compatibility, we extend the Workflow Management Colation's process definition meta model with compatibility, the process definition language XPDL(XML-based process definition language) is able to support null subflow, dynamic binding, subflow jump and five kinds of controlling association.

Workflow design tool JaWE and engine Shark are choosed to compose a WfMS as the original system to be improved on the basis of research on a few open souce workflow design tools and engines. JaWE and Shark have been lucubrated and improved to be a WfMS proto type supporting the flexible frame of workflow.This system is able to execute process defined partially ,reuse the process definition and loosely coupled.

In comparation with congeneric method, this paper present one that is more compatible and non-programming. A evaluating standard that's able to reflect the core flexibility of workflow has been put forward and comparing analysis between the original system and improved system has been made. Conclusion that the improved system is more flexible than the original one at changeable process definition, reusability and dynamic bind aspects has been reached.

At last, the flexibility of workflow and its improvement on the automazition of business process is a pair of logical contravention. Therefore, we must have a good command on the relationship between flexibility and automazition , efficiency while pursuing flexibility of workflow and compromise to archive the best result.

KEY WORDS：Workflow,Flexibility,XPDL,Loosly Coupled,Reusability

目 录

第一章 绪论

1.1. 工作流研究现状

21 世纪以来，信息技术发展一日千里，办公自动化（Office Automation,OA）的口号仿佛就在昨日，而今我们却已经进入了工作流的新时代。从电子商务（Electronic Commerce,EC）到电子政务（Electronic Government,EG），从供应链管理（Support Chain Management,SCM）到客户关系管理（Customer Relationship Management,CRM）甚至于 ERP 以及最常见的 MIS 都可以见到工作流的身影。工作流对我们日常生活的影响正日益显著，工作流技术也正逐渐成为计算机应用领域一个新的研究热点。对工作流技术进行深入的研究对于加速我国企业的信息化进程，提高其运行效率以及竞争能力都有着重要的意义。

1.1.1. 工作流标准

工作流标准组织——工作流管理联盟（Workflow Management Colation,WfMC）成立于 1993 年。WfMC 成立后为工作流标准开发了一套框架，并制定了一系列的工作流相关标准以及规范，包括工作流参考模型[1]（Workfolw Reference Model,WfRM）、基于 XML 的工作流定义语言[2]（XML-based Process Definition Language,XPDL）、基于 XML 的引擎运行集成协议[3]（Wf-XML）、工作流管理应用编程规范[4]等，WfMC 制定的标准规范框架如图 1-1。

虽然 WfMC 拥有从工作流生产及销售商、用户到学术及顾问机构共 300 余组织成员，其中包括国际上著名的 Adobe、BEA Systems、FileNet Corporation、IBM、NEC Soft Ltd、Oracle、Sun Microsystems、SAP AG 等软件公司。但是由于最初各大公司工作流的技术路线差异导致的利益之争，工作流存在着不同的事实标准以及流派。如果说 WfMC 的标准血脉正统，保守而高贵的话，其他标准则倾向于技术的革新。在这些技术流派中，大致可以分为两类[5]。第一类侧重基于纯 XML 技术，如 WfMC 的 XPDL、BPMI 的 BPML 以及 OMG 的 Workflow Management Facility。第二类则侧重基于 web 服务，如由联合国/贸易促进和电子商务中心（UN/CEFACT）和 OASIS（结构化信息标准发展组织）共同倡导、全球参与开发和使用的支持模块化电子商务框架的规范集合 ebXML 以及 2002 年 8 月 9 日由 Microsoft, BEA, IBM, SAP & Siebel 联合提交发布的商业流程执行语言（Bussiness Process Execution Language for Web Service,BPEL4WS）。

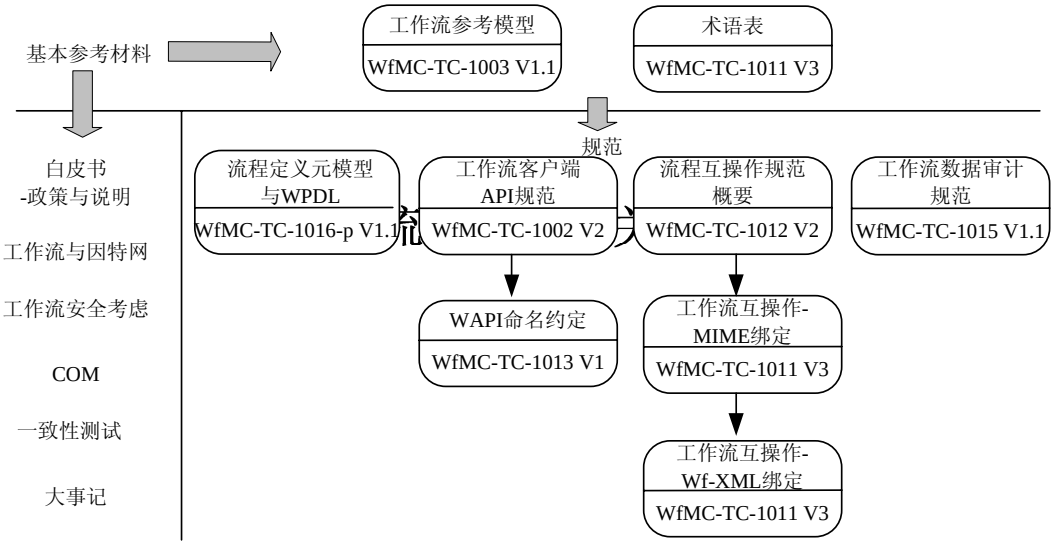


图 1-1 工作流标准及相关规范

1.1.2. 工作流产品、研究成果及应用技术

经过十余年的发展，工作流经历了概念逐渐由模糊到清晰、应用由局部到整体、技术由简单到复杂的过程。随着 WfMC 的成立、标准的制定、网络的普及以及新技术的发展，工作流研究及产品成果不断推陈出新。

目前市场上的产品根据采用的任务项传递机制的不同可以分为四类[6]：(1)基于文件的工作流系统——以共享文件的方式来完成任务。这种类型的产品是产生最早、发展最成熟、最具多样性的，通常包含有 Client/Server 模式的图像、文档与数据库管理系统。代表产品有 FileNet 的 Visual WorkFlow，IBM 的 FlowMark，InConcert 的 InConcert。(2)基于消息的工作流系统——通过用户的电子邮件系统来传递文档信息。这种类型的产品都已实现了与一种或多种电子邮件系统的集成。代表产品有 Novell 与 FileNetet 合作开发的 Ensemble，JetForm 的 InTempo，Keyfile 的 Keyflow。 (3) 基于 W eb 的工作流系统——通过 WWW 来实现任务的协作。这一类产品起步较晚(在 1995 年以后)，但是发展迅速，已成为一种最新的市场流行趋势。许多供应商纷纷开发新产品或者在原有产品的基础上增加对 Web 的支持。代表产品有 Action Techno logies 的 Action Works Metro，Ultimus 的 Ultimus。(4) 群件与套件系统——依据划分标准，这一类产品与前面 3 种有很大程度的重叠，但是在这里却有必要把它们单独划分成一类，因为这类产品都需要依赖自己系统的应用基础结构，包括消息传递、目录服务、安全管理、数据库与文档管理服务等，它们本身就构成了一个完整的应用开发环境。代表产品有 IBM/Lotus 的 Lotus Notes,Microsoft 的

Office 与 Exchange，Novell 的 GroupWise。

在学术界，工作流研究与产业界相比也毫不逊色。美国 Georgia 大学计算机系 METEOR (managing end to end operations) 研究项目的工作流管理原型系统 ORBWork[7]是基于 CORBA 的完全分布的工作流执行系统，以 CORBA 产品 Orbix 作为底层的通信支持，并使用 CORBA 来实现系统的互操作和数据源的封装。METEOR 研究项目的另一个工作流管理原型系统 WebWork 完全基于 web。IBM Almaden 研究中心所进行的研究项目 Exotica 在工作流分布执行方面提出了一种能够完全分布的执行模型[8]，Exotica/FMQM 是该项目完成的基于永久消息队列的分布式工作流系统，它通过持久消息(persistent messages) 的方式来保存工作流相关执行信息，使得每一个执行节点都是相互独立的，工作流过程的执行不以某一个节点为中心，完全实现了分布。这种方式大大提高了系统的可靠性、可扩展性以及柔性。除此之外还有瑞士苏黎士大学计算机系的事件驱动分布式工作流系统 EVE[9]以及达特茅斯大学计算机系设计开发的一种基于可移动代理的工作流系统 DartFlow[10]。国内有清华大学研究开发的用于集成制造的工作流管理系统 CIMFlow[11]和西北大学软件工程研究所研究开发的基于信牌驱动式工作流计算模型的工作流管理系统[12]。

1.1.3. 百家争鸣的开源工作流项目

随着开源项目的兴起，工作流开源项目在工作流领域内也独树一帜，为工作流的发展及繁荣做出了不可磨灭的积极贡献。开源工作流项目按照功能可以分为两类，工作流定制工具和工作流引擎。开源工作流定制工具有 JaWE[13]、nezha[14]等，它们都支持 WfMC 定义的 XPDL 规范，其中 JaWE 身出名门，为多个开源工作流引擎所采用。 开源工作流引擎则可以说是遍地开花，备受关注的有 OFBiz（不仅仅是工作流引擎，它是一整套的开发基于 Java 的 web 应用程序的组件和工具）、OBE、Shark、OpenebXML、Bonita、Twister、ActiveBpel、OpenWFE、jBpm、OSWorkflow、willow[15]等，其中 willow 是国内的一个开源项目。 同时，开源工作流支持的标准不一，也可以分为两个流派，即上面所说的侧重于基于 XML 的和侧重于基于 Web service 的。上述工作流定制工具 JaWE、nezha 以及工作流引擎 OFBiz 、OBE、Shark 都是完全支持 WfMC 标准的。由于目前 OFBiz 采用了 Shark 作为其工作流引擎，OBE 的载体公司 Zaplet 被合并，新公司无继续发展 OBE 的计划，Shark 不但功能丰富、系统可扩展性良好，尤其是其对 WfMC 参考模型中定义的接口 5——管理与监视接口的实现无人能敌，其前景被许多人看好。其余的如 OpenebXML、Bonita、Twister、ActiveBpel 等则是革新派的支持者，它们都侧重于对 Web service 的应用。

1.1.4. 工作流的发展趋势及研究前沿

工作流技术最初起源于办公信息系统领域（Office Information System,OIS）和计算机辅助协同工作(Computer Supported Cooperative Work,CSCW)领域[16]。到目前为止，工作流的发展主要经历了三个阶段[5]：电子数据流（Electronic Data Flow, EDF）阶段、事务处理流（Transaction Process Flow，TPF）阶段以及信息管理流（Information Management Flow, IMF）阶段。EDF 阶段工作流在信息技术中的应用，仅着眼于利用信息技术减轻人们在流程中的计算强度，如设计一个流程用来协调多个会计统计帐目。所以，EDF 最主要的特点是仅对企业单项业务进行处理,基本不涉及管理的内容。而 TPF 阶段并没有形成对企业的全局业务的管理,而着眼于对企业局部业务的管理,比如,设计一套工作流程,来管理物资的采购和供应。当今阶段，IMF 强调对企业业务全局整体性的管理，工作流就是为了完成同一目标而相互衔接、自动进行的一系列业务活动或任务。

随着工作流技术的发展及其在各行各业日益广泛的应用，目前，工作流技术与信息技术以及企业管理紧密结合，已经悄悄渗入 MIS 系统、ERP 系统和 CRM 系统等企业级关键系统中，并迅速成为这些系统的核心。因此工作流系统在企业应用集成领域（Enterprise Application Integration,EAI）中的作用越来越重要。

在工作流的研究方面，主要可分为三大类[17]：(1) 工作流的理论基础：包括工作流管理系统的体系结构、模型、定义语言等。(2) 工作流的实现技术：包括工作流的事务特性、先进的软件技术的应用[18]、工作流仿真等。(3) 工作流技术的应用:工作流实施技术在不同应用领域的应用方法、应用软件集成等。

随着工作流技术应用日益广泛，人们对工作流提出越来越多的需求。工作流平台既需要具有通用性的特点以适用于不同的应用场景，同时又需要具有一定的柔性与适应性以与特殊的应用场合紧密衔接，最大限度的帮助用户提高工作效率，提高企业在复杂商业环境中的竞争力并促进企业的持续发展。与这些需求相对应，工作流研究热点包括工作流模型的研究[17,18]、工作流时间管理[19]、工作流柔性[20-24]、动态性[25-30]、适应性[31,32]、事务性[33]、工作流访问控制[34]、分布式工作流[35]及工作流在企业信息集成中的应用、工作流的性能评价及仿真等。

目前，国内外对于工作流柔性的研究也取得了一定的进展，有关柔性工作流的相关内容在本文第二章有详细的阐述。

1.2. 研究问题描述及目标

工作流及其应用目前存在如下几大特点：

1、 工作流应用相当广泛。几乎所有的组织内部、组织与组织之间都存在着业务流程，对这些业务流程有效的执行、管理、监控以及改进深刻的影响着组织的工作效率。近年来有关 B2B（Bussiness to Bussiness）、B2C（Bussiness to Customer）、BPR（Bussiness Process Reengine）、BPM（Bussiness Process Management）的研究都是以组织内部及之间的流程管理为研究对象的。工作流在电子商务、电子政务、供应链管理、客户关系管理、ERP 等以处理流程为主的系统中的应用大大提高了业务流程的执行效率及效果。可以说，只要有业务流程存在的地方，就有工作流。

2、 随着互联网的普及，分布式技术的发展， 系统的不同实例之间，不同的系统实例之间的数据交换以及协同工作需求迫切。工作流系统的发展也呈现出分布式的特征。

3、 工作流技术与其它软件应用的融合。IT 技术在各大企业中的应用日益广泛，工作流技术与信息技术以及企业管理结合日益紧密，工作流系统之间的互操作以及工作流与其他系统之间的集成给工作流提出了灵活性（定义时的不确定性）、可重用性、松散耦合性以及通用性、可扩展性及更高的稳定性等需求。

4、 商业竞争日益激烈，敏锐的觉察瞬息万变的商业环境并迅速地做出反应不仅仅是企业发展的需要也是企业生存的需要。业务流程的高度的变化，对工作流技术提出了前所未有的挑战。

从工作流应用的特点可见，在工作流系统帮助企业提高工作效率及竞争力的同时，工作流在应用中的缺陷逐渐的暴露。这些问题主要反映在工作流流程定制以及执行不灵活，工作流定义与执行联系过于紧密、对异常情况的处理欠缺、工作流定义版本管理复杂，以及工作流在 MIS 系统、ERP 系统和 CRM 系统等企业级关键系统的渗入过程中可扩展性不足。

柔性工作流致力于解决目前工作流应用中出现的部分问题，这个问题的范围取决于工作流柔性的界定。目前工作流柔性上没有一个统一的定义，对于工作流的柔性、动态性、适应性等术语的使用也经常混淆。

针对上述问题，本文将研究的内容锁定为：工作流柔性范畴的界定，即从不同的角度讨论工作流的柔性及其改进方法，如何在该范畴下从理论的角度以及工程应用的角度改进工作流的柔性以及实现基于一组开源工作流管理系统包括定制工具及引擎的工作流柔性改进。

1.3. 本文所作的主要工作

本文所作的主要工作如下：

1、 工作流柔性相关研究的分析、评价与总结

2、 工作流柔性范畴的界定

3、 工作流柔性框架及其理论基础与实现机制的研究，包括柔性框架下的可配置内容、配置基础理论、分类、规则、机制及其实现等内容，并对框架的可行性进行了分析。

4、 WfMC 元模型以及流程定义语言 XPDL 规范标准的研究以及 XPDL 的柔性扩展

5、 开源工作流管理系统的研究

6、 改进开源工作流管理系统，提高其柔性，实现了一个工作流柔性框架的原型，并进行了相关测试工作

7、 本文采用的柔性改进方法与其他方法优缺点比较分析

8、 工作流柔性的评价机制讨论及对系统原型改进前后柔性进行了对比

1.4. 论文结构及内容

本文共分六个章节，各章节内容如下：

第一章：对工作流的研究现状、发展趋势及本文的研究目标、方法及意义进行了综述。

第二章：介绍了工作流相关基本概念，并对工作流柔性作了广泛深入的讨论。在总结前人对工作流柔性研究与理解的基础上，对工作流的柔性进行了详细的定义，并深入的讨论了改进工作流的各种方法和途径。

第三章：提出了工作流建立-配置-运行柔性框架，讨论了框架可配置内容、配置基础理论、分类、规则、机制及其实现等内容，并对框架的可行性进行了分析。

第四章：在深入分析开源工作流管理系统定制工具 JaWE 和工作流引擎 Shark 的源代码的基础上设计与实现了一个支持柔性框架的 WFMS，包括 F-JaWE 和 F-Shark。同时完成了系统的柔性功能测试工作。

第五章：对本文采用的柔性改进方法与其他改进方法进行了对比，总结出了能够反映工作流柔性核心的柔性评价标准，同时在该标准下对本文提出的柔性工作流框架及其实现进行了分析与评价。

第六章：总结了本文工作的长处与不足以及需要进一步研究的内容与方向。

第二章 工作流及其柔性

2.1. 工作流概述

如果数据库系统像受人尊敬的智者讲述的条理清晰的故事，那么工作流就像是一群乳臭未干的小子在大谈各自的“哲理”。之所以这样讲，是因为工作流系统还处于技术发展曲线上的初级阶段。在这个领域我们将面临一个激动人心的阶段[36]。开源工作流 jBpm 的奠基者 Tom Baeyens 如是说。工作流的标准组织及其标准众多、工作流产品及开源工作流百家争鸣的现状印证了 Tom Baeyens 的陈述。在工作流领域还有很多事情等待我们去做，值得我们去做。

2.1.1. 什么是工作流

处于发展初级阶段的工作流，无论是在技术上还是在标准上都存在着不同程度的不成熟与不统一。不同的工作流学者和工程师从不同的角度对工作流进行了定义。下面是一些有代表性意义的定义，它们分别从不同的角度对工作流概念进行了描述，有助于我们对工作流的一些基本特征的理解。

1、 IBM Almaden Research Center 的定义[37]

工作流是经营过程中的一种计算机化的表示模型，定义了完成整个过程所需用的各种参数。这些参数包括对过程中每一个单独步骤的定义、步骤间的执行顺序、条件以及数据流的建立、每一步骤由谁负责以及每个活动所需要的应用程序。

2、 Am it Sheth 的定义[38]

工作流是涉及到多任务协调执行的活动，这些任务分别由不同的处理实体来完成。一项任务定义了需要做的某些工作，它可用各种形式来进行定义，包括在文件或电子邮件中的文本描述、一张表格、一条消息以及一个计算机程序。用来执行任务的处理实体可以是人，也可以是计算机系统(比如一个应用程序、一个数据库管理系统)。

3、 Giga Group 的定义[39]

工作流是经营过程中可运转的部分，包括任务的顺序以及由谁来执行、支持任务的信息流、评价与控制任务的跟踪、报告机制。

4、 W. M. P. Van der Aalst 的定义[40]

工作流是一系列工作的偏序集。工作的序列可以有多种方式，比如工作 X

与 Y 满足 X；Y 当且仅当 X 在 Y 开始之前就已经就绪。

简单的说，工作流就是全部或部分的由计算机支持或自动处理的业务流程[1]，工作流管理系统(Workflow Management System,WfMS)则是一个工作流运行的平台。工作流系统支持活动或任务的顺序执行，能够根据需求将活动或任务自动分配给相关的执行人以及在不同的流程参与者之间传递执行结果信息。WfMC 对工作流相关术语简要定义可参考文献[41]。

2.1.2. 工作流分类

根据工作流的特性及特性的组合，工作流可以被分为不同的类型。Martin Ader[41]根据相关流程间的相似度及其对企业的价值将工作流分为生产型、管理型、协作型和 Ad hoc 型，如表 2-1 所示。

表 2-1 Ader 的工作流分类

	生产型	管理型	协作型	Ad hoc 型
场景	变化少，高度可控以及高吞吐量的常规流程	良定的、没有迫切的性能要求的日常操作流程	提供结构化协作支持的流程	未事先定义的流程，特定于流程实例
用例	保险索赔流程	购买订单，费用表统计	预算与审批文档的准备	医院诊断病人流程

同时我们也可以根据任务的复杂度和结构来对工作流进行划分[43]。不同类别的工作流在其商业价值、流程的重复执行程度、任务的复杂度及任务的结构并不是完全不同的，如图 2-1 所示。虽然 WfMC 的规范应用到了各种类型的工作流及系统当中，但是这些应用大部分属于管理型或生产型工作流。

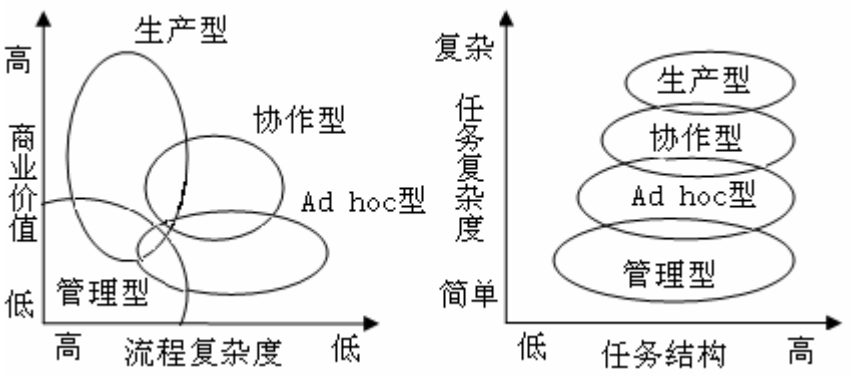


图 2-1 根据任务特点的工作流分类

2.2. 柔性工作流

目前对于工作流的柔性国内外还没有统一的定义，不同的研究项目、学者所指的柔性意义不尽相同。虽然对工作流柔性的讨论比较多而杂，但对其本质的认识基本上是一致的。在这些讨论中，工作流柔性和动态性重叠的地方较多，这也说明了工作流柔性、动态性的定义及范畴不明确的特点。有的说的是动态性，而其中讨论的许多本质的问题又与柔性有关。

Steinar Carlsen 指出柔性不仅仅是产品和技术的一个特性，同时也是在组织设置中的使用与部署[16]。Shazia Sadiq 等将工作流所面临的变化分为三个方面：动态性、柔性和适应性，并给出了工作流的柔性、动态性及适应性的明确定义[22]。本文在综合参考各大家对工作流柔性理解的基础上并结合工作流目前的发展趋势及应用环境于 2.2.2 一节对工作流柔性的范畴进行了一个较清晰的界定。

2.2.1. 动态与适应性工作流简介

学界以及业界对工作流动态性、适应性与柔性的认识存在一定的混淆和重叠。所谓动态性是指工作流随业务流程进化的能力，其面临的最大问题是工作流定义新旧版本交替时如何处理正在运行的旧版本的流程实例。Pinar Koksal 等认为对于存在正在运行的实例却又发生变化的工作流定义，可以通过如下三种方式来处理[44]：放弃策略、并行策略和转换策略。转换策略中可以采用C.Liu 等人提出的动态工作流的接管策略规范语言[45]来描述。

工作流的适应性是工作流对异常环境的反应能力。工作流异常指工作流执行过程中出现的可预见及不可预见错误，包括工作流机交互时出现的通信错误、资源不足、任务执行错过时机或执行时间超长等等。异常处理能力与工作流动态修改密切相关，异常处理需要动态修改能力的支持，而对动态修改的需求，往往又源于对异常的发现和分析，两者相辅相成。

对于工作流的动态性，有的学者将其分类两类，永久动态性和临时动态性。所谓永久动态性就是上面所说的工作流定义版本的升级切换，临时的动态性则是指在工作流运行过程中的对工作流定义的修改与运行。显然，永久动态性影响到的是所有的流程实例，临时动态性即动态修改影响到的只是暂时的，仅影响一个流程的运行。在本文中，临时的动态性被视为柔性的范畴之内，它涉及到对不确定性流程的处理，为柔性提供执行不确定性定义的工作流程的能力。因此，我们需要注意区分动态修改与动态性，动态修改并不等同于动态性。

2.2.2. 柔性工作流的范畴

目前对柔性工作流的定义主要是从工作流程执行的角度来考虑的。鉴于工作流与程序语言之间的相似度以及目前工作流日渐广泛的应用及与其他系统的渗透与结合越来越深入越来越紧密，工作流的柔性需要多方位的审视，而不仅仅是工作流的执行，还应该包括从程序理论、工程的角度、应用的角度来考察与界定。

狭义的工作流的柔性即从工作流执行的角度来定义的柔性通常是指工作流系统执行松散、部分定义的工作流程的能力，工作流定义往往在运行时才完全确定。

由于工作流的执行与工作流的定义之间的关系密不可分。我们可以将工作流定义划分成不同的层次。Peter Mangan 和 Shazia Sadiq 等将工作流的定义划分为结构层、数据层、时间层和执行层四层[24]。Pinar Koksal 等在论述工作流定义变化时，指出需要考虑结构层、任务层、资源层和系统层四个层次[44]。本文经过综合分析，认为工作流定义的内容可以划分为三个层次，所谓数据、时间、资源等属性都可以看作是工作流定义的属性。如图 2-2 所示，工作流定义被分为执行层、属性层、拓扑层三个层次构成。拓扑层由活动以及转移组成，它们共同描述流程的拓扑结构，即工作流的任务执行的先后关系。属性层则由时间、资源、相关数据及其他属性组成，以辅助说明拓扑层，如为活动和转移提供标识、名称及描述，活动的时间、资源、调用的外部应用，转移选择的条件等。执行层则由流程或者活动的状态转换规则、活动执行的启动结束模式及优先级组成，它描述工作流运行时的动态行为。通过对工作流定义内容层次的划分，可以得出工作流系统执行松散、部分定义的工作流程中待确定的内容可以是流程执行的步骤即活动、与活动相关的属性如活动的最晚执行时间、活动执行最长时间、活动所需资源及相关数据等以及流程和活动状态转换的规则。

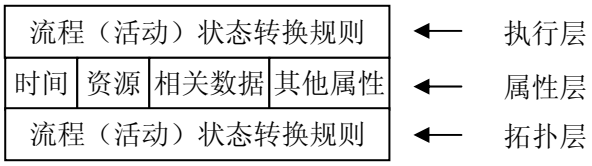


图 2-2 工作流定义的三个组成层次

工作流定义语言是一种面向过程的流程语言，它支持模块化、数据的作用域以及包之间的引用。而程序设计语言已经由面向过程发展到面向对象了，具有良好的继承重用及多态性，灵活性更大。鉴于程序设计语言的发展，工作流定义语言中引入面向对象的思想，有助于提高工作流柔性。

工作流管理系统最初直接将业务流程编码，结果当业务流程发生变化时，

系统不得不重建，代价昂贵。目前，工作流管理系统由工作流定制工具和执行引擎组成。定制工具完成对业务的建模，引擎负责执行流程。当业务流程发生变化时，只需重新建模即可。但是这样的建模方法没有重用性，同时也没有统一建模的思想。因此从工程的角度上，进一步降低工作流建模与执行之间的耦合，有利于工作流系统的柔性提高。

综合以上不同角度工作流柔性的考察，本文对工作流的柔性界定如下：工作流的柔性是指工作流定义与执行松散耦合、可扩展性及可重用强、具有模块化、继承性、分布性等特点、能够执行不确定性定义的工作流程并能够在执行时逐步完善之。

2.3. 提高工作流柔性方法的研究

2.3.1. 提高工作流柔性方法分类

目前全球工作流学者在解决工作流柔性问题上提出了各种各样不同的方法，提高工作流柔性的研究主要集中在建模方面，通过引入能够表示不确定性的活动来提高工作流的柔性是研究主流。提高工作流柔性的方法可以按照不同的角度来进行划分。Shazia Sadiq 等将提高工作流柔性的方法分为了三类[22]：

1、 通过定义的方法（flexibility by definition）：该方法主要是通过增强工作流定义语言的描述能力。

2、 通过控制粒度的方法（flexibility by granularity）：如 WfMC 的过程定义元模型存在原子活动、块活动和子工作流三种粒度的活动，后两者是粗粒度的活动，增加工作流定义时的不确定性。文献[23]则细化活动粒度，提高工作流运行时的灵活性。文献[46]采用问题分离原则把汇聚、分支及其约束说明从活动中提取出来组成新的元素“连接符”，把控制信息说明从活动说明中独立出来，分别封装，使变化局部化，提高工作流的松散耦合性。

3、 通过模板提高工作流的柔性。

本文在 2.2.2 节中对工作流定义进行了层次划分。通过对工作流层次的划分，使得提高工作流柔性的方法有规可循，工作流柔性的分析随之提升到方法学的高度。根据工作流层次的划分，本文将工作流柔性的改进又分为以下三种方法：

1、 从拓扑层提高柔性：拓扑层提高工作流的柔性主要体现在对活动执行顺序的不确定性描述、活动的重用方面，如 WfMC 的过程定义元模型中的分支与合并转移，文献[22-25，47]所采用的类黑盒机制支持对活动执行顺序的不确定性描述，而子工作流、通过外部引用使用 web 服务等提高了活动的重用性。

2、 从属性层提高柔性：WfMC 参考模型中，一些相关数据在定义时作为拓扑层转移选择的条件被定义，在运行时，由于相关数据的取值不同导致一个流程定义具有不同的执行路线，我们称之为流程实例类型，流程定义的某些流程实例由于在决定转移选择的相关数据上具有相同取值，即具有相同的执行路径，从而属于同一个流程实例类型。目前工作流系统在属性层上对柔性的支持仍然不够，而且有很大的改进的潜力。如果对工作流执行时有影响的所有属性都能像工作流相关数据一样在运行时改变，工作流系统的柔性将大大加强。

3、 从执行层提高柔性：目前 WfMC 的工作流参考模型仅提供了一种流程和活动转换有限自动机，对活动执行的描述也仅限于启动和结束活动是否为自动，所有的活动一律都可以被放弃、终止和挂起。从执行层的描述能力上来讲，该参考模型不是很强，还有很多地方可以扩充以加强工作流的柔性。

无论是从什么角度来对工作流柔性的改进方法分类，工作流柔性的改进对象的具体性质决定了不同工作流改进方法所涉及到的事物的本质及其局限性。不同类别的改进方法之间也存在着一定的联系和重叠，有的甚至是需要同时使用两类甚至更多的改进方法或者涉及到了工作流的不同应用阶段的相互配合才能达到改进工作流柔性的目标。

2.3.2. 面向对象工作流

Sadiq 的分类没有囊括面向对象工作流，根据本文对柔性工作流的定义，面向对象所能提供的继承重用性和多态性都能够提高工作流的柔性。

面向对象工作流有两种解释，其一为应用面向对象技术的工作流系统，其二为将面向对象思想引入到工作流模型、语言中。本文的面向对象方法即指第二种解释——在工作流中引入面向对象思想。

在软件设计思想的发展过程中，大致经历了面向过程、面向对象两个阶段。工作流与程序之间存在着极大的相似性，同时其本身又是建立在程序之上的。因此软件设计领域中先进的思想非常值得工作流领域借鉴。

程序设计思想中最先出现的便是面向过程的设计思想。面向过程的程序设计方法将解决问题的重点放在模拟问题的过程方面，它在解决问题时具体采用的是自顶向下、逐层分解、逐步求精、模块化的结构化程序设计技术，把大的程序划分为若干相对独立、功能简单的程序模块，强调的是过程，强调功能模块化，通过一系列过程调用和处理完成相应的任务。

从前面的分析来看，工作流中无时不刻的体现着面向过程的思想，大致可归纳成如下几个方面：

1、 模块化：通过子工作流、块活动等元素支持自顶向下、逐层分解、逐

步求精、模块化的结构设计

2、 数据作用域：与结构化程序语言一样，工作流定义语言中的相关数据都具有一定的作用域，包中的相关数据作用域为整个包中的所有流程，流程中的相关数据局限于流程内部。同时，相关数据存在同名时，服从小作用域相关数据屏蔽大作用域相关数据的规则。

3、 模块与数据的引用：允许通过导入的方式引用包以外定义的相关数据、参与者、流程。

面向对象的基本原则是封装、继承与多态[49]。将信息与处理信息的功能组合起来，然后将其包装成对象成为封装。封装有利于将系统的改变影响限制在对象内。所谓继承是指一个类拥有另一个类的数据和操作[50]，在面向对象的程序设计中，采用继承的方式来组织设计系统中的类，可以提高程序的抽象程度，更接近人的思维方式，使程序结构更清晰并降低编码和维护的工作量。简单的说多态是指特定功能有多种表现形式和实现方法，在运行时动态绑定其具体的表现形式和使用的实现方法。显然，面向对象的设计思想从重用性、不确定流程的定义与执行等方面对于工作流柔性的提高都有着很大的帮助。目前工作流不存在流程封装、流程之间的继承关系以及流程定义与执行的动态绑定。从 WfMC 对工作流定义语言的设计来看，最初就从根本上承认了工作流与程序之间一衣带水的血缘关系，吸收了面向过程的程序设计思想。90 年代正是面向对象设计思想从少年走向成熟的阶段，工作流还没有来得及将面向对象设计思想吸收，面向对象便已经在不知不觉中成熟壮大起来了。如果能够在工作流中融入面向对象的思想，工作流便可以轻易的沿着程序设计思想之路乘风而上，绕过程序设计思想发展过程中所走过的弯路。面向对象思想在提高工作流柔性中的应用是本文的一个重点，将会在后面有较详细的阐述。

目前，Bell 实验室 Guangxin Yang 提出了一种支持流程继承与动态修改的命名为 P 的工作流语言[54]，该语言最终将被编译后执行，其设计与实现与目前的面向对象程序设计语言有着极其密切的关系。文献[55]使用网状约束作为其元模型。一个约束实际上是某个任务的描述，它可以分解成多个通过子约束的输入输出端口连接的子约束网络。其继承机制是通过从一个约束网络栈产生组合约束来支持的。该元模型约束过多是很多错误的根源，如非法实体参考（悬挂实体）、不兼容端口的连接以及令牌的丢失，同时它不能保证每次继承的结果是正确有效的。文献[59]致力于建立一个流程描述库，将描述组成一个体系，通过继承而不是编程进行分类。遗憾的是流程继承的实现机制没有论及。文献[60]所描述的 WF-net 是一个 Petri-net 的变体，该元模型具有四种类型的继承：协议、投影、协议/投影以及生命周期继承。其他的包括文献[56-58]都声称是面

向对象的，而实际上它们的面向对象性仅限于以一种面向对象的方式描述不同的实体及其协作。

当代网络已经非常的普及，SOA 的概念也逐渐兴起。上个世纪末 Gartner Group 就提出了面向服务（Service-Oriented Architecture,SOA）的概念[51]，而实际上早在上个世纪 70 年代早期，采用 SOA 构建软件的方法就已经出现。CORBA 便是早期的 SOA 实践之一。目前，SOA 仍然没有一个官方的定义，SOA 到底是一个软件产品、一种设计思想和方法还是一种应用技术还在处于争论之中。IBM 认为，SOA 不是一个产品，而是一种设计概念，应用它可以灵活的将业务流程元素和根本的 IT 基础架构视作安全的标准化组件（服务），重复使用并加以组合，以满足不断变化的业务优先级。BEA 则认为 SOA 是一种方法，一种建立、维护、管理 IT 系统和业务流程的方法论。中关村科技软件则认为，SOA 是一种应用技术，应用的业务逻辑被组成模型（服务）。不管 SOA 到底是一种产品、概念、方法或者仅仅是一种技术，只要能为工作流所用，能够提高工作流的柔性即可。

从工作流的互操作以及将流程的不确定性迁移到网络上去的角度出发，SOA 为我们提供了方便，目前也有一些利用 Web Service 来提高工作流柔性的研究，如 Kwak 等人提出的支持动态工作流互操作及 EAI 的框架[52]、WfMC 提出的异步服务访问协议[48]（Asynchronous Service Access Protocol,ASAP）等。

2.4. 小结

本章对工作流及其柔性的基础概念进行了完整的阐述，同时在综合目前工作流的研究现状从不同的角度对工作流柔性进行了分析以及范畴界定。对工作流改进的方法分类及方法论进行了较深入的探讨，提出了工作流拓扑层、属性层、执行层三个层次的划分及各层次之上可对工作流柔性进行的改进。

总的来说改进工作流的柔性可以通过提高流程定义语言描述能力、粒度控制以及面向对象方法来完成。同时，从拓扑层、属性层、执行层我们又能够看到另一个工作流柔性的视图，从而作为上述三种方法的辅助。

第三章 工作流柔性改进框架

第二章对引入面向对象思想提高工作流柔性的研究现状进行了总结和评价，指出目前面向对象柔性工作流的研究存在仅限于研究讨论阶段，研究成果也存在着限制与不足。本章将讨论一个引入了面向对象思想的工作流框架，并讨论相关理论与实现机制。

3.1. 工作流建立-配置-运行柔性框架

工作流柔性改进框架从工作流部署的各个阶段及其关系出发，引入中间阶段以降低工作流建立和运行阶段之间的紧密耦合，提高工作流的松散耦合性、可扩展性以及定义与执行的不确定性，同时也体现了对程序设计理论相关思想的借鉴。虽然工作流柔性改进框架着重从工程性的角度结合对工作流执行的扩展讨论分析工作流柔性的改进，但是在理论及其应用方面也有所兼顾。

3.1.1. 什么是工作流框架？

所谓框架，是指事物的组织结构。本文中工作流的框架是指工作流生命周期中，工作流系统的各部件或实体组织与部署过程的结构。在工作流发展的初级阶段，工作流的生命周期中只有一个阶段，业务流程被编写到代码中，流程定义由程序语言代码完成。这样的工作流系统在业务流程发生变化时，必须修改源代码以满足用户的需求。严重的时候可能导致整个系统的重新构建，其代价之昂贵可想而知。应用程序设计领域将数据与执行代码分离为工作流将流程定义与流程的执行分离提供了借鉴。目前工作流的生命周期分为两个阶段：工作流的建立阶段和工作流的执行阶段。该时期的工作流管理系统的特点是把过程逻辑与应用逻辑分离，可以在不修改具体功能的情况下，通过修改过程模型，就可以改变系统的功能[53]。所以，WfMS 可以有效支持管理、控制过程的集成和重组。然而在这样一个两阶段的工作流生命周期框架下，工作流的建立与执行仍然存在较紧密的耦合。工作流柔性框架便是从降低工作流定义阶段与执行阶段之间的联系，在定义阶段与执行阶段之间加入配置阶段，使得工作流定义阶段的结果在运行时可以呈现出多样性，而不是单一的一个流程定义。使工作流定义在运行之前有足够的时间为运行做好充足的准备。各种框架说明如图 3-1 所示。

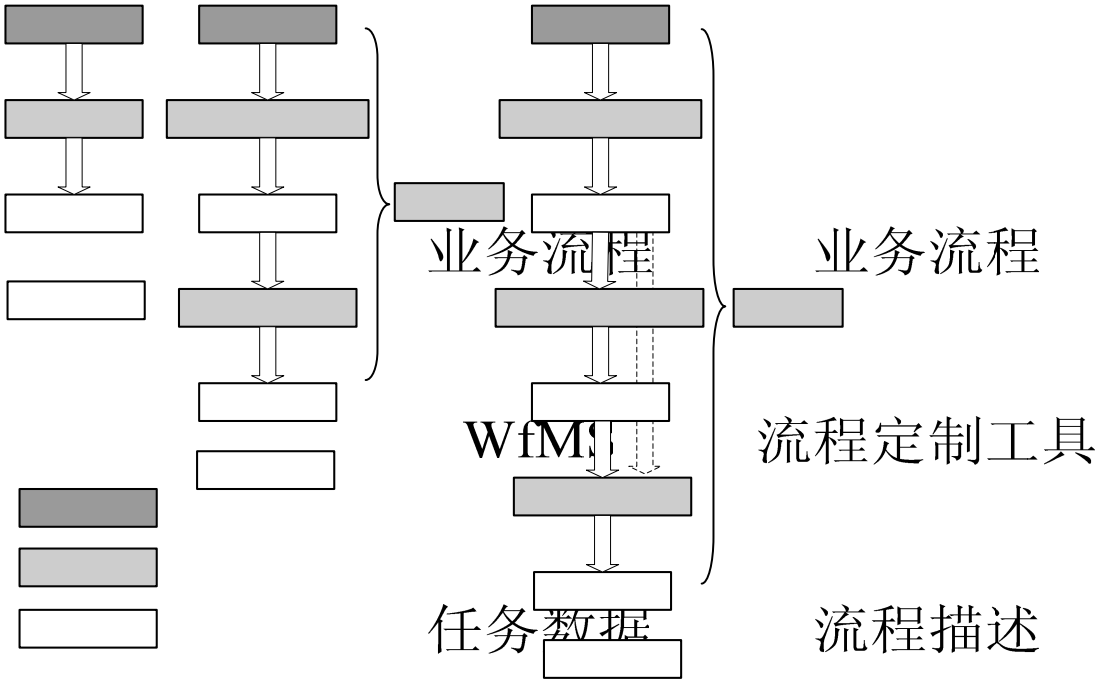


图 3-1 各种框架对比示意图

3.1.2. 工作流柔性框架的三个阶段

如图 3-1 所示，工作流柔性框架包含三个阶段：工作流建立阶段、工作流配置阶段以及工作流运行阶段。这三个阶段的工作及职能如下：

1、 工作流建立阶段

工作流建立阶段是指工作流管理系统用户根据自身业务需求，对业务流程及其相关的组织、资源、应用系统建模并利用工作流的建模工具将业务模型转化为工作流定义语言描述的流程的过程。对业务流程的建模需要对与业务流程相关组织、资源、权限、时间、数据及应用等信息的处理与组织。工作流平台作为一个流程运行的通用平台，对于一些特殊的流程的执行，业务流程的建模就显得尤为重要了。在一些工作流应用中，开发人员常常为应付特殊的流程而烦恼，有时候为了简便省事，便把一些需要特殊处理的业务流程动作放到应用层中去处理，这样便导致了代码与流程之间紧密的耦合依赖。一旦特殊业务流程需求发生变化，应用程序的代码也需要修改，程序的稳定性、可靠性以及可维护性大大降低。实际上，工作流定义语言基本上考虑到了各种应用场合，工作流定义语言与程序设计语言之间的相似性说明了我们在程序逻辑中能够解决的问题一定能够通过工作流定义语言的逻辑来解决。因此，业务流程建模的好坏将直接影响到工作流的柔性，切忌不进行业务建模的前提下就将业务建模的工作放到程序逻辑中去处理。对于一些必须通过程序逻辑处理的业务流程，仍

然可以借用外部应用来处理，因为这是一个接口，是可插拔的，业务流程需求发生改变时，只需要改变外部应用，影响的范围仅限于单个外部应用。

2、 工作流运行阶段

符合WfMC的XPDL规范的工作流定义必须被工作流引擎装载后才能够执行，工作流引擎装载流程定义的过程是一个解析的过程（也有一些其它的规范及系统是将工作流定义编译后执行的），通过解析发现工作流定义中的错误，错误的流程定义是不能被执行的，解析的结果是一个 XML 的树形文档对象，对象的根节点是包，包的结构如下：

<?xml version=”1.0” encoding=”UTF-8”?>
<Package Id=”testWRD” Name=”TestWRD” ……>
+<PackageHeader>
+<RedefinableHeader PublicationStatus=”RELEASED”>
<ConformanceClass GraphConformance=”NON_BLOCKED”>
<Script Type=”text/javascript”>
+<Participants>
+<Applications>
+<DataFields>
+<WorkflowProcesses>
+<ExtendedAttributes>
</Package>

每个带“+”号的节点都可以展开，所有这些信息构成工作流运行时的信息。工作流实例化时，流程中与开始节点相邻的活动自动启动，工作流引擎根据定义的执行人分配任务，生成任务纪录放入到任务列表中。执行人完成相关任务的执行后，将任务的执行结果作为相关数据提交，同时下一个活动实例化并被分配给相关执行人，直至流程结束。在整个流程的执行过程中，涉及到了数据在业务流程中不同的执行人之间的流转、任务的自动分配、任务的委派、根据中间执行结果选择任务执行路径等等操作。

3、 工作流配置阶段

工作流配置是指在已有的工作流定义的基础上进行继承和完成一定的修改。在继承过程中，我们将被继承的流程称作超流程，继承的流程称作子流程（注意与子工作流相区别）。继承的结构与修改内容之和将形成一个新的工作流定义——子流程，通过工作流的配置从而实现工作流的重用。配置手段实现的重用并不是纯粹的代码拷贝重用。因为配置的输出不是一个新的而且完整的工作流定义，而仅仅是修改内容及修改流程引用的绑定。工作流的配置阶段将工

作流定义阶段与工作流的执行阶段之间的紧密耦合降低到了最小的程度。

如图 3-2 所示，子流程只有在引擎运行时才会根据其超流程定义及配置数据由引擎生成一个临时的驻留内存的流程定义，该定义为引擎运行该流程提供信息支持。为保证流程实例相关数据对这个内存中的临时流程定义的引用长期有效，引擎将按照一定规则来生成运行时流程定义的 ID。

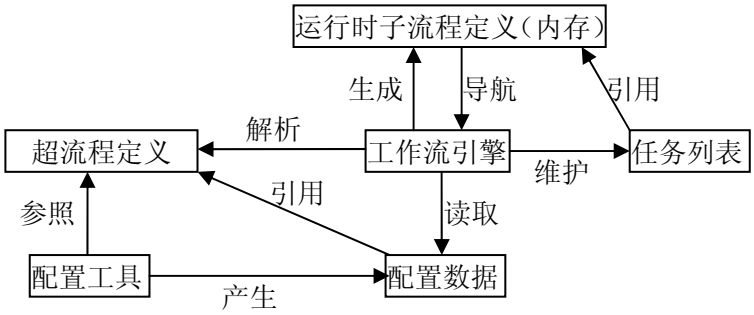


图 3-2 工作流配置及其运行

3.1.3. 工作流柔性框架体系

柔性框架下，可以通过对众多业务流程共性的提取进行系统的建模，如同设计一个面向对象的系统一样。建立阶段可以建立通用的超流程以及普通的流程(即不被继承直接运行),配置阶段则在建立阶段输出的工作流定义的基础上，可以对拓扑结构及属性进行继承或修改并形成子流程。运行阶段则会根据配置后的结果执行工作流。

工作流柔性框架系统体系结构如图 3-3 ，整个框架中主要包括了三个核心模块：流程定制模块、配置模块和导航模块。流程定制模块主要是工作流建立阶段的服务，它负责工作流的定制以及工作流定义的数据库持久化，内容包括流程定义、组织/角色模型定义、全局数据定义以及外部应用定义。配置模块负责配置数据的提取、生成以及存储工作。导航模块提供引擎的功能，由配置模块和存储为其提供数据，然后由它将这些数据组装成流程定义，并在适当的时候执行之。

<table>
<tr><td colspan="2">用户接口</td><td colspan="2">应用接口</td><td rowspan="2">流程定义</td><td rowspan="2">组织定义</td><td rowspan="2">数据定义</td><td rowspan="2">应用定义</td></tr>
<tr><td colspan="4">导航模块</td></tr>
<tr><td colspan="2">存储模块</td><td colspan="2">配置模块</td><td colspan="4">流程定制模块</td></tr>
<tr><td colspan="8">持久模块</td></tr>
</table>

图 3-3 工作流柔性框架体系结构

3.2. 工作流定义配置的分类、内容、规则及其验证

本节我们根据具体的需求决定哪些工作流元素是允许配置的，在什么情况下可以配置，在哪个阶段配置以及如何保证配置的有效性与一致性的。

3.2.1. 配置分类

根据配置影响的范围可以分为两类，一类配置完成后形成的子流程定义将会实例化多个流程，配置时机在工作流建立阶段与运行阶段之间，我们称其为静态配置。静态配置与工作流的动态性之间一眼看去有类似之处，而实际上不同，工作流的动态性存在工作流新旧版本定义流程的切换问题，而这里配置流程与原始流程之间没有流程的切换，流程实例要么按照原始流程执行，要么按照配置流程执行，它们完全是两个不同的流程定义，并不是同一个流程的不同版本。配置也有可能导致工作流的动态性，如当多个配置作为同一个工作流定义的不同版本时，就会导致一个工作流定义的多个版本及其实例的并存局面。由配置造成动态性的配置有可能是如图 3-4 所示的两种情形之一。

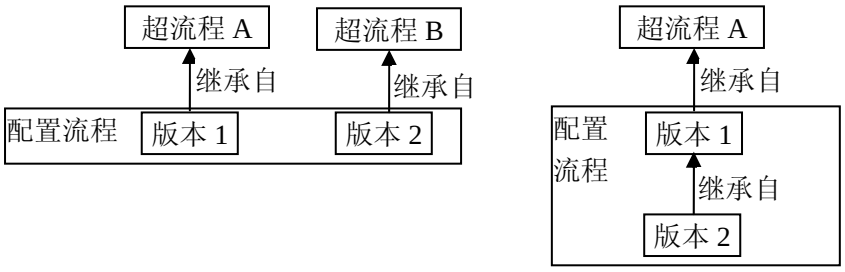


图 3-4 配置引发动态性说明

另一类是在流程的运行过程中进行配置，且仅影响当前流程实例，我们称其为动态配置。这种配置可能在根超流程[1]上进行也有可能在非根超流程上进行。我们称静态配置的配置库为静态配置库，动态配置的配置库为动态配置库。静态配置库与动态配置库之间是分开的，静态配置库与流程定义相关联，可与流程定义共同形成新的流程，而动态配置库由于在配置完后立即执行，配置信息只对当前运行流程有效，当前流程执行过后再也没有其他的流程实例按照该动态配置流程运行。由于静态配置影响范围较广，效益更高，动态配置只影响一个流程，效益低下，因此静态配置可配置的范围较动态配置的大。在柔性框架下的流程配置指导思想是坚持静态配置柔性为主，动态配置柔性为辅的原则。

[1] 没有继承其他超流程的超流程称作根超流程，其他超流程统统都称作非根超流程。

3.2.2. 可配置内容

工作流定义语言 XPDL 中的元素及其属性的可配置性如表 3-1 所示，除表头外的内容共有五行，它们表示不同类型的属性。第一行是所有主要元素的通用属性，第二行是每个元素的特有属性，第三行是元素对其它元素的参考，第四行是文档和图表元素，它包含执行机所需要的描述信息，最后一行是过程模拟和优化的相关信息。属性的配置类型可分为如下几类：

1、 必配：如果产生了新的元素时必须配置的内容，通常是那些需要区分的配置流程与原始流程之间的唯一标识。如流程 id，新配置流程如果不是原始流程的新版本，则 id 必须不同。在表中以粗体表示。

2、 选配：既可以配置也可以直接继承。这些属性是那些流程特性表征的属性，而且配置后对流程的一致性和有效性影响不大，即使有也可以通过一定的约束消除。在表中以斜体表示。

3、 限制选配：有些内容可以选配，但是存在一定的前提条件，建议尽量不要配置。某些引用如外部包、形参到实参的映射等等，如果引用被误删除，使用引用的时候便找不到被引用的内容了，但是我们可以增加引用，即使误操作也不会有太大的副作用。在表中以下划线+斜体表示。

4、 自动配置：自动配置的内容通常是可以选配的内容，但为了避免因误配而造成的不必要的麻烦，由系统自动完成配置。如 creation date（配置的系统时间）、流程的 version 信息（默认为 1，如果是流程定义的进化，则使用选配方式）等。在表中以粗体+斜体表示。

5、 禁配：即禁止配置。禁止配置的内容可以分为两类，一类是配置意义不大，配置后却有可能对流程的一致性和有效性产生副作用的内容。另一类是有意义但配置后将会产生极大副作用的内容。如 XPDL version，不同的版本有不同的规范，配置内容如果遵循配置版本进行设置，而从超流程继承的内容是在原版本的基础上定义的，那么新旧版本之间就有可能造成冲突或者不可识别的问题；而像 publication status 则必须直接继承，因为如果原始流程还处于 under test 状态或者 unreleased 状态，配置流程如何能发布（released 状态）呢？禁配内容没有使用特殊字体表示。

6、 联配：联配表示某些内容之间存在着某种关联关系，如果其中一部分内容被配置了，另一部分内容也需要同时配置。对于一些比较复杂的联配我们采用禁配方式。

表 3-1 XPDL 元素及其属性的可配置性

Package	Workflow Process	Activity	Transition	Application	Data Field (Workflow Relevant Data)	Participant
-Id *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*	**-Id** *-Name* *-Description* *-Extended Attributes*
-XPDL Version -Source Vender Id ***-Creation Date*** ***-Version*** *-Author* -Codepage -Country key -Publication Status -Conformance Class *-Priority Unit*	***-Creation Date*** ***-Version*** *-Author* -Codepage -Country Key -Publication Status *-Priority* *-Limit* *-Valid From Date* *-Valid To Date*	*-Automation Mode* *-Split* *-Join* *-Priority* *-Limit* *-Start Mode* *-Finish Mode* *-Deadline*			*-Data Type*	-Participant Type
-Responsible *-External Package* *-Documentation* *-Icon*	*-Parameters* *-Responsible* *-Documentation* *-Icon*	*-Performer* *-Tool* *-Subflow* -ActivitySet *-Actual Parameters* *-Documentation* *-Icon*	*-Condition* -From -To	*-Parameters*	*-Initial Value*	
-Cost Unit	*-Duration Unit*	*-Cost*				
	-Duration *-Waiting Time* *-Working Time*	*-Duration* *-Waiting Time* *-Working Time*				

3.2.3. 配置的验证，数据关联与控制关联

由于配置过程中有可能破坏原有的数据关联和控制关联，这将导致了配置后的流程完整性和一致性关系的破坏。

数据关联是一种引用关联关系，是指一个元素对另一个元素完全或部分的引用。如流程 A 中有一子工作流 B 的情形下，存在双向的引用，流程 A 对流程 B 存在完全引用，流程 B 又反过来引用了流程 A 的实际参数作为它的形式参数，即流程 B 对流程 A 存在着部分引用。在配置的过程中，为保证数据关联不被破坏，被引用的元素或者元素的属性是不允许删除的。流程 A 中被流程 B 引用的实际参数发生了变化，将会对流程 B 造成影响，而流程 B 的输入形式参数定义类型以及个数发生了变化将影响到所有调用流程 B 的流程。也就是说，它们两个中的一个发生了改变，另一个可能也需要作相应的改动。而具体如何使这种相互之间的影响最小化，需要依具体情况决定。程序设计语言的设计中通常是调用者遵循被调用者的接口规范。在这种情况下，被调用者如果多处被调用，那么被调用者接口的修改影响的范围会比较大。但是如果是调用者方发生了变化，则调用者内部处理即可。为减少数据关联的影响，设计非常重要。

文献[53]对控制关联的定义如下：WF=<P,R>是一个可适应工作流系统，其中 P＝（P1，P2，……Pn）是工作流的过程集合，R=（R1，R2……Rn）是描述过程之间关联关系的规则集合，对于 M＝（P′,R′），其中 P′ ⊂P,R′ ⊂R，如果对于 Mi∈M，当 Mi 需要动态修改时，M 中的所有过程都需要进行动态修改，则称集合 M 为 P 的一个控制关联集。如上定义所述，数据关联与控制关联之间存在着一定的联系，如在数据关联中举的例子。但是数据关联与控制关联之间是有区别的。区别如下：

1、 控制关联必将会导致相关联的元素的改变，即具有控制关联的元素要么同时改变，要么都不改变。而数据关联仅仅是一种约束，如被引用的参与者不能被删除，但参与者属性的改变等不会引起相关元素的变化

2、 数据关联存在着引用与被引用的关系，而控制关联不一定。如某个流程中的两个活动的执行顺序关联。

配置的验证工作主要验证各个元素的完整性及其之间的一致性，同时还需要验证流程中是否存在不可达或者不可推进的活动节点。验证工作保证了工作流定义的正确性。

3.2.4. 配置规则

由于上述关联及其他因素，配置后的流程可能会因不满足工作流的正确性和一致性要求而无法正常工作。为了避免因配置而导致工作流过程定义无法装载到引擎或工作流实例无法正常运行甚至出现工作流系统部分功能崩溃的情况，工作流的配置必须遵循相应规则。配置规则是一个在对工作流配置时需要遵循的规则集，该规则集通过对配置的操作进行约束和限制来保证配置后的工作流的正确性和一致性。

配置规则根据配置的分类也可以划分为两类，静态配置规则和动态配置规则。静态配置规则是进行静态配置时需要遵循的规则，它的特点是对流程的影响范围大，但是比较迟缓，在配置的过程中是不会对正在运行中的流程造成影响的。而动态配置仅影响一个流程实例，影响非常迅速。对于静态和动态配置的规则由于二者特点的不同，约束和限制也有所不同。

1、 静态配置规则

静态配置对流程的影响范围较大，大的灵活性以及少的限制有利于最大限度的提高工作流系统的柔性。同时静态配置对工作流的影响比较迟缓，通常要在配置完成以后，才能被实例化，配置过程中它不会影响任何流程实例，因此静态配置时，任意时刻，只要配置操作被允许，都可以进行。首先我们将配置操作分为如下几类：流程配置、活动配置、参与者配置、外部应用配置、相关参数配置与转移配置。静态配置需要遵守以下几条规则：

(1) 流程元素配置只能修改相应属性，或对已有流程克隆

(2) 活动、转移、参与者、外部应用以及相关参数元素可以进行增加、修改、删除操作

(3) 删除操作以遵守数据及控制关联为前提，参与者、外部应用以及相关参数被引用时不允许删除，活动删除时相邻接的转移联动删除，删除转移以及活动时不能有不可达的情况出现

(4) 尽量做到不修改拓扑层结构

(5) 没有拓扑层结构变化的配置形成不同版本引起的动态性采用简单的动态处理策略——并行策略。

2、 动态配置规则

动态配置影响的范围较小，与流程实例相联系，因此有些操作在流程实例运行过程中是不允许的。动态配置规则如表 3-2 所示。动态配置规则主要考虑了配置结果对流程实例执行的影响，对于配置后将影响到流程实例的正常执行以及一些虽然不会对流程的执行造成影响但是无意义的操作都是禁止的。

表 3-2 动态配置规则

动态配置操作	限定条件
新增活动	允许，通过子工作流动态绑定完成，新增活动必须在当前执行活动之后
修改活动（包括活动的各种属性，如执行人、对相关数据的操作权限、执行时限等）	活动运行之前或活动运行处于异常状态
删除活动	活动运行之前或活动运行处于异常状态
一切与转移相关的配置	禁止，如果需要修改转移，通过子工作流到调用流程的跳转完成
新增相关数据	允许
修改或删除相关数据	禁止
新增数据类型	允许
修改或删除数据类型	禁止

3.3. 工作流柔性框架方案

工作流柔性框架配置后的流程由两部分组成，超流程定义与配置数据。在实现上，如何配置、提取、存储配置数据，配置影响的范围及其副作用的消除，配置与原始工作流通过何种方式组成子流程等等问题都是工作流柔性框架方案设计过程中需要考虑的问题。本小节提供了两种配置方案——属性配置与操作配置，它们分别通过记录配置过程中修改的属性或记录修改操作过程来实现配置。

3.3.1. 属性配置方案

属性配置主要是通过在工作流定义属性层上的配置并保存配置的信息来完成的，它完全继承被配置的超流程的拓扑结构，不能够新增加元素也不能删除已有的元素。配置信息需要保存被配置属性所属元素到根节点包的一条路经，以保证从配置信息及超流程合成子流程时能够准确地修改相应的属性。

由于属性配置方案只能在超流程属性层的基础上进行更新操作，对于一些比较大的修改，如拓扑结构的修改即活动元素的增加与删除需要借助于工作流定义规范的扩展，通过对子工作流以及每个活动的虚拟子工作流的配置完成。

3.3.2. 操作配置方案

操作配置方案记录用户在原始流程基础上进行的每一次配置操作，这些操作可以包括元素的新增、删除与更新以及属性的更新。相对应定义三个操作原语：add(element)、update(element or attribute)和 delete(element)，分别表示增加元素、更新属性或元素和删除元素。因此操作配置方案可以对原始流程进行任意的配置。但是任意的配置将会导致配置流程对原始流程继承比率的减少，违背继承重用的初衷。因此，对操作配置方案需要添加一些约束，以限制对原始流程配置内容的无限增长。

3.3.3. 配置方案的选择

属性配置与操作配置各有优缺点，属性配置的灵活性较小，利于大范围的重用。操作配置的灵活性较大，配置内容过多将会导致配置流程对原始流程继承性下降，这样的话可能还不如重新定制一个流程。属性配置与操作配置功能对比如表 3-3（更新指元素各种属性的配置）：

表 3-3 属性配置与操作配置功能比较

配置类型 / 元素	属性配置			操作配置		
	新增	更新	删除	新增	更新	删除
流程	×	✓	×	✓	✓	×
活动	×	✓	×	✓	✓	×
相关数据	×	✓	×	✓	✓	×
转移	×	✓	×	✓	✓	✓
参与者	×	✓	×	✓	✓	×
外部应用	×	✓（部分）	×	✓	✓（部分）	×

注：外部应用的定义取决于具体的外部应用，因此只能够配置其名称，以适应具体的应用场景。考虑到关联规则，操作配置虽然可以删除的一些元素作禁止删除处理

属性配置可以通过子工作流以及虚拟子工作流在一定程度上改善对活动以及转移的新增与删除功能。鉴于操作配置方案对流程配置范围约束的复杂性，以及流程定义规范扩展对属性配置功能的加强，并基本满足需要，因此选择属性配置方案，在属性配置方案的基础上作一定的扩充，加强其在添加和删除方面的不足。

最后需要说明一点的是，无论选择哪一种配置方案，配置是为了流程定义的重用，如果配置内容在配置流程中所占的比率过大，会增加配置流程有效的

风险性，同时也不利于体现重用性。因此，我们虽然提供各种配置功能，但是常用的功能仍然是以不直接改变流程拓扑为前提的属性层配置。

3.4. 细粒度状态触发机制

通常工作流系统流程状态的转变有可能引起其他流程和活动的状态的转变，而活动的初始化仅在上一个活动执行结束时进行的，其他活动状态转变不会对其他活动造成任何影响。而有时候恰恰是活动状态的变化反映了一些影响流程高效执行的关键因素。如一个长时间执行的生产流程，在生产的过程中有可能因特殊情况出现因为原材料不够而暂时停止执行，我们称之为挂起。如果此时原材料得不到及时的采购和供应，生产过程将会永远挂起得不到及时地执行以至于影响交货日期。如果在生产过程挂起时同时能够自动启动原材料采购活动则整个生产活动将以最高效率状态运行。而此时原材料采购活动并不是因为某个活动完成开始的，而是因为生产活动的状态发生了变化，由执行转为挂起而导致的结果。因此，此时我们有必要考虑在状态级别的粒度分析活动了。图 3-5 对上述流程进行了简单的描述。

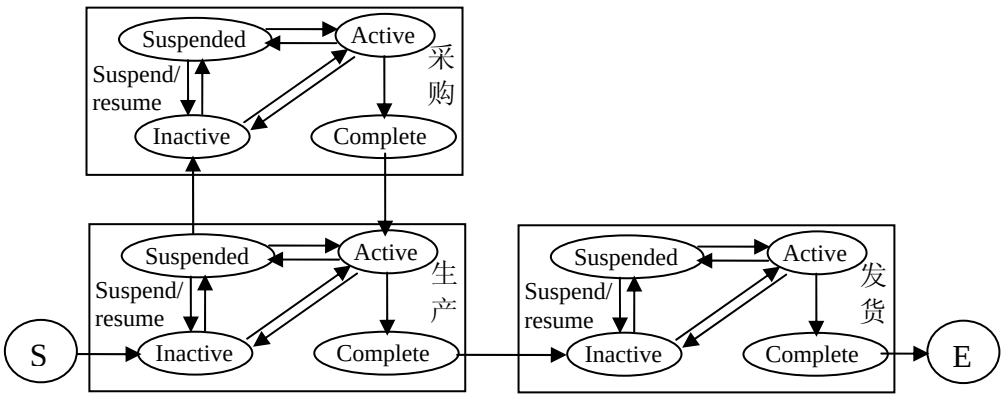


图 3-5 活动状态级触发流程描述

3.5. 小结

本章阐述了工作流建立-配置-运行柔性框架的思想，并讨论了实现该框架需要考虑的问题，包括可配置的内容、配置的规则、配置流程有效性的验证、支持配置的工作流定义扩展机制以及配置流程的合成。最后从粒度细化的角度进一步分析对工作流柔性的提高。

第四章 工作流定义语言的柔性扩展

4.1. 工作流定义语言的柔性扩展现状

第二章指出可以通过提高流程定义语言描述能力、粒度优化以及面向对象的方法改善工作流的柔性。提高流程定义语言描述能力以提高工作流柔性的研究工作是目前做得比较多的。这些工作大致可以分为两类，第一类就是通过重新建立新的工作流模型来提高柔性[17,20,25,62,63]。建立的新模型为了支持柔性，通常比较复杂，不同的模型之间无法兼容，因此该方法的兼容性差。第二类方法则是通过在现有标准模型的基础上增加新的元素实体，如黑盒(black box)或者柔性活动[22,24,25,47]，该方法由于添加的元素实体不被标准模型支持，因此也缺乏兼容性。

文献[22]，[47]的思想比较相似，前者着重讨论了流程片断的组合，而后者则给出了流程片断约束的形式化定义并实现了一个流程片断组合的算法。两者的思想大致如下：在已有的工作流定义语言模型标准的基础上进行扩展，增加了元素实体柔性活动（文献[47]中的命名，文献[22]则称之为构建活动），柔性活动的具体内容在定义时时未知的。同时，在流程库中，存在一个活动池，其中的活动有可能是事先保存的也有可能是临时定义增加的。与活动池相对应，有一个约束描述库，专门描述活动池中的活动进行组合的约束。当流程执行至柔性活动时，用户从流程池中选取需要执行的活动，由流程片断组合算法根据约束描述库中的相关约束自动组装一个合法的活动执行序列或子工作流程。

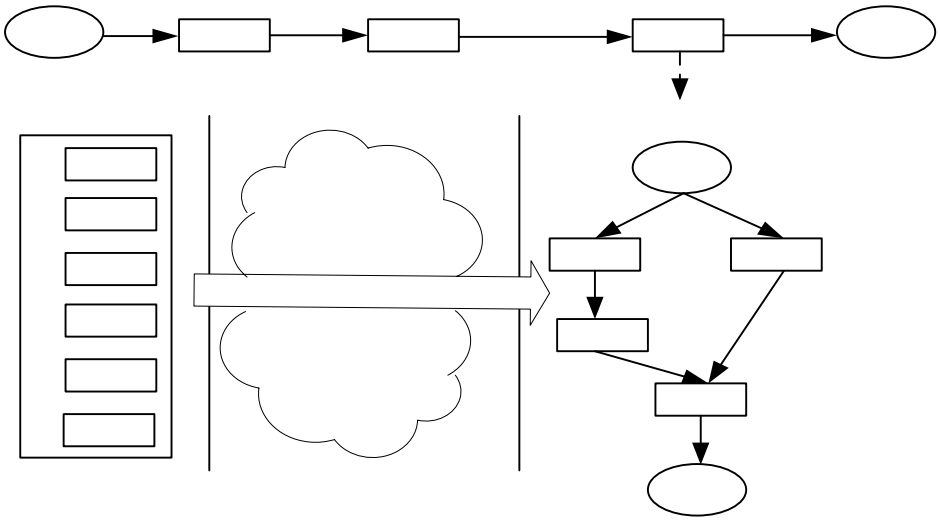


图 4-1 柔性活动的执行

柔性活动的执行如图 4-1 所示。当流程推进到柔性活动时，可由用户手动或者系统根据事先定义好的条件自动选择活动和约束，最后组装成子工作流程替换柔性活动。文献中对于如何自动选择活动和约束没有论及。

本文对上述方法分析如下：

1、 柔性活动本质上仍然是待定义的一个子工作流程。

2、 由于需要流程池（或流程片断库）及相关约束，一方面要求在业务建模的时候考虑周全（从这个意义上说，跟直接用各种模型结构描述没有区别），另一方面对于建模是没有考虑到的活动及约束，只能够在流程的运行过程中积累。

3、 由于增加了元素实体，降低了兼容性。

4.2. 工作流定义语言标准 XPDL 及其可扩展性分析

4.2.1. XPDL 简介

WfMC 于 2005 年 8 月提交了基于 XML 的工作流定义语言标准 XPDL[2](XML-based process definition)第 2 版。该标准的包定义元模型如图 4-2 所示，包是一个工作流程、参与者、工作流相关数据、工作流应用、资源库定义的集合，同时它还可以包含其他的包。流程定义也允许有属于自己的参与者、工作流相关数据、工作流应用的定义。不同层次上定义的参与者、工作流相关数据、工作流应用具有不同的作用域。

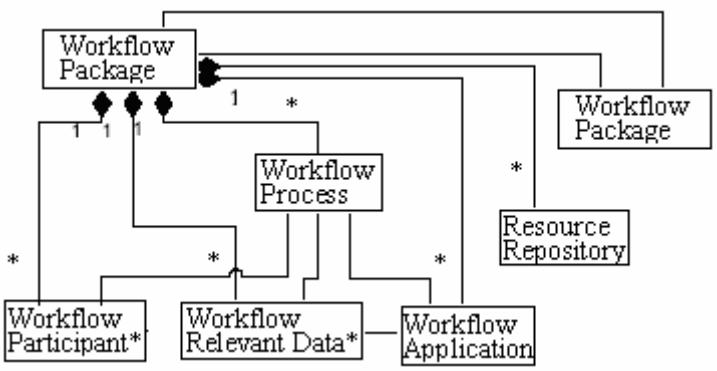


图 4-2 XPDL 包定义元模型

XPDL 对流程的描述能力较强，同时也存在一定的限制，XPDL 无法支持不确定性的流程定义，包括执行路径的临时修改、跳转以及活动内容运行时确定等。因此在 XPDL 的基础上，提高工作流柔性的扩展仍大有可为。

4.2.2. XPDL 的兼容扩展

兼容可以分为水平的兼容和垂直的兼容。水平的兼容是指系统或协议之间的兼容，而垂直的兼容则是系统或协议的高级版本向低级版本兼容。通过在现有标准模型的基础上增加新的元素实体的方法只能支持垂直兼容，而不支持水平兼容。本文在 XPDL 的基础上进行扩展，垂直兼容性由 XPDL 保证，本文着重考虑水平兼容。

对 XPDL 的扩展，目前很多研究工作都是通过在流程定义元模型的基础上增加实体来完成的，这种方法与 XPDL 的兼容性较差，新增加的实体在 XPDL 元模型无法识别。

而另一方面 XPDL 标准设计之初，就考虑到了对扩展的需求，为其包模型中所有实体专门预留了扩展属性。扩展属性的 XML Schema 描述如下：

```
<xsd:element name="ExtendedAttribute">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="Value" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

利用扩展属性进行扩展，可以在保持 XPDL 元模型以及兼容性的基础上，通过扩展元模型中各实体的概念来支持更多的柔性需求。

4.3. 支持不确定性描述的扩展

在 XPDL 的基础上不增加元素实体进行扩展主要是通过对原有元素实体概念的扩展来完成。本节通过对子工作流的概念扩展，支持了不确定性的流程定义，提高了工作流定义语言的描述能力。

4.3.1. 空子工作流程

子工作流程是实现工作流模块化，以粗粒度的方式提高内聚、降低耦合从而达到提高工作流柔性目的的元素实体。但是由于 XPDL 中没有能够支持工作内容在定义时允许不确定，运行过程中逐步完善的元素实体。空子工作流程便是针对这一问题，在不改变 XPDL 元模型的基础上，通过概念扩展达到类似于

黑盒或柔性活动的效果。

1、 空子工作流程的定义

空子工作流程的定义如下：遵循子工作流程的 XML Schema 描述、调用及调度规则、允许工作内容在建立阶段无定义、运行阶段根据具体需求确定的子工作流程。在实现上，空子工作流程通过在标准的子工作流程定义的基础上增加一个扩展属性描述完成定义。

2、 XPDL 中空子工作流程扩展描述

利用子工作流程的扩展属性对空子工作流程的支持的扩展描述如下：<ExtendedAttribute Name="empty"/>。

说明：当子工作流程定义的扩展属性中出现了该定义时，说明该子工作流程并没有绑定实际的流程定义。该扩展在子工作流程的扩展属性中定义。

3、 支持空子工作流程需要考虑的问题及解决

空子工作流程是在子工作流程定义的基础上通过其扩展属性进行了概念上的扩展。仅当工作流定制工具以及执行引擎都能够理解该扩展概念的时候，空子工作流程才具有真正的意义。

支持空子工作流程的 WfMS 需要解决如下问题：

(1) 对空子工作流程定义的正确性校验

空子工作流程的正确性校验的解决比较简单，由于 XPDL 是基于 XML 的流程定义语言，空子工作流程是子工作流程与空子工作流程扩展描述共同定义的一类子工作流程元素实体。因此，在对子工作流程元素实体校验之前，先检查扩展属性，如果存在空子工作流程扩展描述，则取消本次子工作流程元素实体校验。

(2) 运行时确定空子工作流程

空子工作流程的具体内容通常取决于其前驱活动的执行结果，由于执行结果的多样性以及不可预测性，空子工作流程的具体内容在运行时由其前驱活动的执行人指定。在柔性框架下，也可以由系统管理员或流程管理人员在配置阶段绑定。

指定应当包括如下几种方式：

1) 绑定已有的流程。

2) 建立流程片断库及相关约束的情况下，允许从活动池中进行组装或直接指定。

3) 提供流程定制工具直接定义、上载并绑定，同时保存到流程片断库中。

(3) 空子工作流程的访问权限

空子工作流程的访问权限根据具体的场合需求不同而不同。本文提供一种

可参考的访问权限模型。该模型下，分为执行层、系统管理层和流程管理层。通常情况下，空子工作流程的具体内容在运行时由其前驱活动的执行人指定，即执行层下空子工作流访问权限的控制。如果执行人无法确定其具体内容，则交由系统管理层,由具有更大权限的系统管理人员根据历史数据分析结果确定。当系统管理层仍然无法解决时，问题将提交由流程建模人员组成的流程管理层解决。

(4) 空子工作流程的绑定

在引入工作流的建立-配置-运行柔性框架后，空子工作流的绑定分为两种类型，静态绑定和动态绑定。静态绑定是指工作流配置阶段的绑定。动态绑定指工作流运行的过程中绑定，此时的空子工作流程的作用与柔性活动相当。

空子工作流程的绑定的内容包括对被绑定的子工作流的引用及形式参数与实际参数的映射。

被绑定的子工作流的产生方式见(2)运行时确定空子工作流程。

有关参数的绑定需要遵循以下两个原则：

1) 形式参数与实际参数的完整性，即每个形式参数都需要有相对应的实际参数绑定。

2) 形式参数与实际参数的类型一致性。

由于空子工作流程存在扩展属性的说明，因此在绑定后需要同时删除该扩展属性项。

4.3.2. 子工作流程返回跳转

子工作流程执行完成后需要返回至调用流程，正常的返回将触发调用流程中其后继活动的执行。子工作流程返回跳转则允许子工作流在返回调用流程时在约束范围内跳转到调用流程中其后继活动之外的活动，从而在拓扑层增强了工作流的柔性。

在拓扑层上提高柔性需要谨慎，如果完全放开拓扑层的柔性，允许任意类型的活动任意的动态跳转，无异于在程序设计语言中引入 goto 语句，造成程序流程结构的混乱。

子工作流返回跳转的扩展在子工作流的扩展属性中定义。

本文对子工作流程返回跳转扩展内容如下：

1、 跳转返回约束

(1) 跳转层次约束：子工作流程流返回至调用流程时，调用者可能是一个流程也可能是一个块活动，或者其本身也是一个被调用流程。因此可能存在多级调用。返回的目标活动限制在调用流程一级，而不能跨越调用流程。如块活

动调用的子工作流程返回的目标活动只能是块活动中的某个活动，而不能是块活动所属的流程中的活动。如图 4-3 中虚线表示的跳转是被禁止的。跳转目标只能是与被调用子工作流程同级别的其他活动。

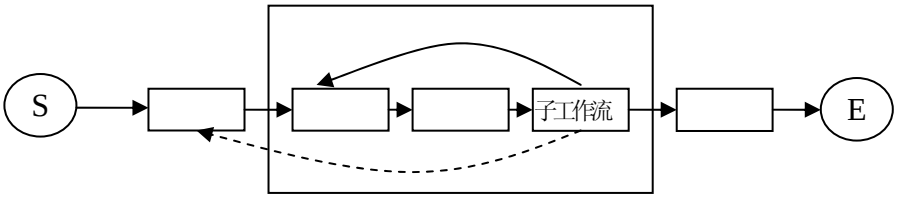


图 4-3 跳转返回约束说明

(2) 重复执行约束：在现实的业务流程中，存在着一些活动只允许执行一次的情况，导致该类活动重复执行的跳转应当禁止。

(3) 忽略执行约束：在现实业务流程中，存在着一些活动必须执行，不能绕过，如审批流程中的审查活动。导致该类活动被忽略执行的跳转应当禁止。

2、 跳转返回类型

(1) 正常跳转返回：返回后调用流程中子工作流程的后继活动被初始化，顺序执行。

(2) 回退跳转返回：返回后会回退至调用流程中子工作流程前面的某个活动并初始化之，回退有可能跨越一些活动，这些活动在业务规则中应当允许回退，如果不允许则该类返回无效。通常允许回退的活动对应于允许撤销重做的业务活动。另一点需要注意的是，回退返回将形成一个循环，因此回退返回需要设定回退返回的条件，条件满足时跳转，不满足时，正常返回，初始化被调用流程后的活动。

(3) 快进跳转返回：返回后会快进至调用流程中子工作流程后面相距两个活动及以上的某个活动并初始化之，快进有可能跨越一些活动，这些被跨越的活动相当于被删除一样不会被初始化以及执行。被跨越的活动相对应的业务活动在现实中应该是允许被忽略的。不允许被忽略的业务活动如审批流程中的审查、颁发执照等等业务流程的关键活动。

3、 跳转返回描述扩展

(1) 可回退扩展：

<ExtendedAttribute Name="reversible" Value="true/false"/>

说明：活动默认为可回退，即无需配置活动扩展属性等价于配置活动扩展属性为<ExtendedAttribute Name="reversible" Value="true"/>，该属性配置在每个活动的扩展属性中

(2) 可跨越扩展：

<ExtendedAttribute Name="ignorable" Value="true/false"/>

说明：活动默认为可忽略，即无需配置活动扩展属性等价于配置活动扩展属性为<ExtendedAttribute Name=" ignorable" Value="true"/>，该属性配置在每个活动的扩展属性中

(3) 被调用流程至调用流程返回的跳转扩展：

<ExtendedAttribute Name="destAct" Value=" Subflow_id.activity_id"/>

<ExtendedAttribute Name="Condition"

Value=" Subflow_id.con_expression"/>

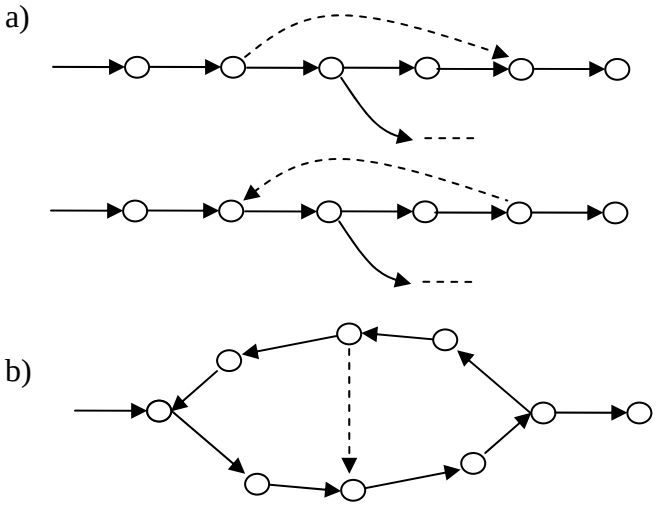
说明：该属性配置在子工作流程实现类型的活动的扩展属性中，destAct 属性记录子工作流程 Subflow_id 及返回至调用流程的目标活动 activity_id,二者以“.”分隔,condition 属性在子工作流程回退返回时使用，记录子工作流程 Subflow_id 及返回至调用流程的目标活动的条件表达式 con_expression，二者同样以“.”分隔。

被调用流程至调用流程的返回由于受到上述种种限制，需要在配置的过程中进行验证，跳转返回必须是满足约束的。

4、 跳转返回正确性验证算法

跳转返回正确性验证算法是一个基于图的遍历的算法，首先需要通过从目标活动遍历图判断跳转是前进的还是后退的。然后再从子工作流程遍历至目标活动，收集跳转跨越的活动。最后根据跳转的性质结合跳转跨越的活动的扩展属性，判断是否存在违反约束条件的情况存在。具体的验证算法在原型实现系统中有详细地描述（见 5.2.2）。

需要指出的是，当流程的拓扑结构中存在循环时，位于循环活动中的跳转将具有二义性，本文中的算法没有考虑该情况下的跳转处理。本文中实现跳转返回正确性验证算法支持带分支的线行拓扑结构的跳转。如图 4-4 所示。



a) 算法支持的跳转
b) 算法不支持的跳转

说明：由于循环拓扑结构中的节点之间相互可达，不存在偏序关系，b)中的跳转存在二义性

图 4-4 跳转返回正确性验证算法支持与不支持的情形

4.3.3. 不确定性描述的扩展柔性小结

从空子工作流程的扩展可知，空子工作流程可以完成文献[47]中的柔性活动同样的功能。空子工作流程对柔性支持的机制与图 4-1 的完全一致。但是，空子工作流程还允许直接绑定已有的工作流程或调用客户端的流程定制工具直接定制。子工作流返回跳转则从拓扑层上提高了流程执行的灵活性。

4.4. 柔性框架下符合控制关联配置的扩展

控制关联扩展的意义在于对现实业务流程中存在的控制关联作出约束，提高了工作流对现实业务流程的模拟与描述能力。同时，在支持继承的柔性框架下，保证控制关联的特性不被改变，即控制关联在被继承时，禁止重定义。

通过对现实业务流程的归纳，本文总结出了五种控制关联：活动事务选择关联、执行人关联、执行顺序依赖关联以及并行执行关联。同时规定，所有的事务关联都限制在同一层次的工作流中。因为流程、块活动、子工作流程之间存在递归关系，跨层次不利于降低耦合，同时处理起来也很困难，意义也不大。

所有的控制关联约束都是在流程元素的扩展属性中定义的。同时，由于对控制关联约束扩展的验证算法比较简单，本文没有论及。

1、 活动事务选择约束扩展

事务选择约束是指超流程中的若干个活动在现实业务流程中具有很强的联系，只要其中一个出现在业务活动中，其他的就必须出现等待调度（但不一定在每一个流程实例类型中出现）。

活动事务选择约束扩展描述如下：

<ExtendedAttribute Name="transaction" Value="id.actId"/>

说明：id 说明一个事务选择的集合，它是多个活动的集合，其作用相当于分组标记，id.actId 记录每一个在事务选择 id 中的活动 id，其中.前面的 id 表示该项内容所属的事务选择 id。该约束表示一旦流程中存在该属性，表示属于同一集合（即即 id 部分相同）的所有活动都必须存在于流程中，缺少一个都不合法。

2、 执行人约束扩展

执行人约束扩展分为两类，活动执行人互斥约束和活动执行人依赖约束。对于两种约束的定义如下：

(1) 活动执行人互斥约束是指若干个活动禁止由同一个人、角色或组织执行。该约束有利于防止业务流程执行过程中徇私舞弊现象。

(2) 活动执行人依赖约束是指若干个活动必须由同一个人、角色或组织执

行。该约束有助于职能划分，体现责权分明，防止了不同的人、角色、组织之间相互推诿扯皮，同时提高工作效率。

执行人约束约束扩展描述如下：

(1) 活动执行人互斥约束扩展：

<ExtendedAttribute Name="same_par_baned" Value="id.actId"/>

说明：Id.actId 包括两个部分，以.分隔，其中 id 表示禁止同一人执行的若干活动的集合 Id，actId 表示集合 Id 中的一个元素。

(2) 活动执行人依赖约束扩展：

<ExtendedAttribute Name="same_par" Value="id.actId "/>

说明：Id.actId 包括两个部分，以.分隔，其中 id 表示必须同一个人执行的若干活动的集合 Id，actId 表示集合 Id 中的一个元素。

3、 执行顺序依赖约束扩展

现实业务活动中，由于某些活动之间存在结果数据的依赖，即某些活动的输出数据是另外一些活动执行的必要输入数据，从而导致了它们的执行顺序的依赖。执行顺序依赖是一种偏序依赖，执行顺序依赖活动集构成一个偏序集。

执行顺序依赖约束扩展描述如下：

<ExtendedAttribute Name=" Squence" Value="id.actId"/>

说明：Id.actId 包括两个部分，以.分隔，其中 id 表示活动所属顺序执行依赖活动集合 Id，actId 表示属于集合 Id 的一个活动，属于同一个顺序执行依赖活动集合的活动以出现顺序表示其执行顺序依赖。

4、 并行活动约束扩展

并行活动约束扩展是指某些活动要么同时被执行，要么都不执行。并行活动扩展具有执行的原子性。

并行活动约束的扩展描述如下：

<ExtendedAttribute Name="concurrency" Value="id.actId"/>

说明：Id.actId 包括两个部分，以.分隔，其中 id 表示活动所属并行执行活动集合 Id，actId 表示属于集合 Id 的一个活动。

4.5. 小结

在提高工作流柔性的同时，又保证兼容性，是本文对标准工作流定义语言进行柔性扩展的目标。本文利用 XPDL 元素的扩展属性对子工作流程在概念上扩展了空子工作流程以及允许其跳转返回，从而达到了不确定性定义以及动态拓扑修改的柔性。同时，通过对现实业务流程归纳总结出了五种控制关联，一

方面提高了工作流对现实业务流程的模拟与描述能力，另一方面保证了在支持继承的柔性框架下，控制关联的特性不被改变，即控制关联在被继承时，禁止重定义。

第五章 柔性框架下工作流系统改进原型的设计与实现

5.1. 开源 WFMS 的选择

工作流管理系统 WFMS 由两部分组成：工作流定制工具——负责生成工作流定义，以及工作流引擎——负责工作流定义的解析与执行，并提供工作流执行的用户接口、监视接口、外部应用调用接口以及工作流互操作接口。

在本文 1.1.3 节对开源的工作流系统的研究现状作了比较详细地介绍。我们此次研究选择的工作流定制工具是 JaWE，工作流引擎是 Shark，它们之间互相兼容，都支持 WfMC 的 XPDL 规范。

5.1.1. 开源工作流定制工具 JaWE

JaWE 是一个流程定义建模的图形化编辑工具，完全支持 WfMC 的 XPDL 规范 1.0 版本。JaWE 主要完成三个功能：

1、 流程定义的图形化表示

2、 将流程定义导出到 XPDL 文件（结构符合 XPDL 定义的 XML Schema 的文本文件）

3、 导入所有有效的 XPDL 文件及其图形化显示

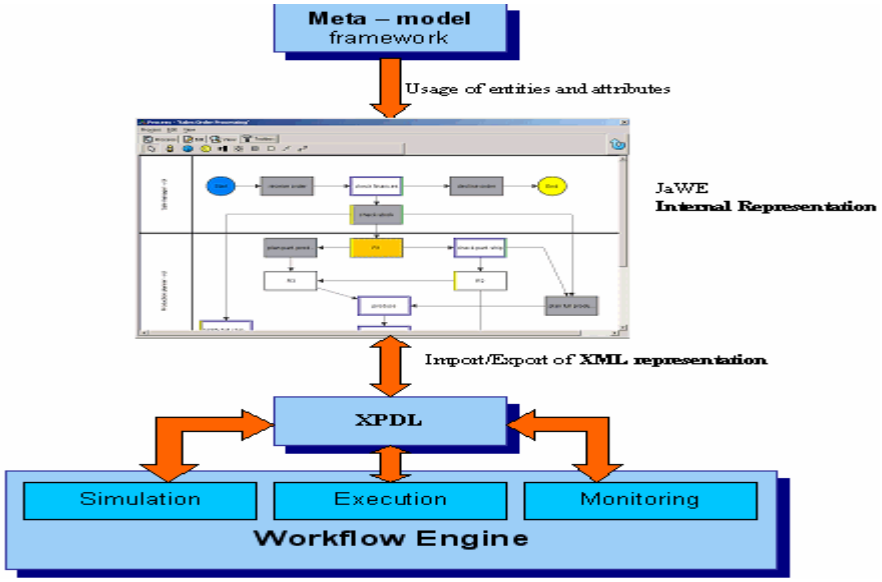


图 5-1 JaWE 在工作流元模型框架中的位置

图 5-1 说明了 JaWE 在工作流元模型框架中的位置。通过工作流定义接口，不同用户可以选择不同的工作流引擎和建模工具来组成自己的 WFMS。

在开源工作流定制工具的选择上，没有太大的余地。开源工作流定制工具 nezha 以及 jBMP 自带的定制工具（不支持 XPDL）的都依赖于另一开源集成开发环境 Eclipse（作为 Eclipse 的插件发布）。因此，本文选择了能独立运行的 JaWE。

5.1.2. 开源工作流引擎 Shark

通过对多个主流的开源工作流引擎的对比，本文决定选择 Shark 作为 WfMS 的引擎。本文主要从易用性、兼容性、跨平台性、技术先进性以及功能的完整性五大方面考察了 Shark、jBMP、bonita、Breeze、openWFE 共五个开源工作流引擎。对比如表 5-1 所示。

表 5-1 开源工作流引擎评价

	易用性	兼容性	跨平台性	技术先进性	功能完整性
Shark	较易用	好，支持 WfMC 的 XPDL 标准	好，纯 java 实现	较先进，全分布式（提供了 CORBA 实现），架构设计分层清晰，可扩展性好	完全实现了 WfMC 参考元模型中的五大接口
jBMP	简单易用	差，仅支持 JPDL，不兼容 XPDL	好，纯 java 实现	较先进，分布式，提供 EJB 实现	没有实现外部应用接口
bonita	不太易用，有时需要写代码	差，仅支持 VRML，不兼容 XPDL	好，纯 java 实现	较先进，采用 Java web start 方式运行	没有实现外部应用接口
Breeze	不友善，功能单一	差，不兼容 XPDL	好，纯 java 实现	落后，仅能单机运行	没有实现外部应用接口，图形监控功能非常简单
openWFE	较差	差，不兼容 XPDL	好，纯 java 实现	一般	没有实现外部应用接口，图形监控

开源工作流引擎 Shark 完全基于 WfMC 和 OMG 规范。它有如下特点：

1、 使用 XPDL 作为其工作流定义语言。

2、 标准的内核实现是一个不创建新线程的库，可以用于不同的环境，如 web 应用、swing 应用、也可以部署为 CORBA 服务或者 EJB 服务等。

3、 完全可配置，所有的 internal 接口甚至内核都可以实现配置

4、 可以同时运行于多个虚拟机（集群的环境下）

5、 可以配置成使用 LDAP 中定义的组织结构

6、 使用 DODS（Enhydra 提供的 O-R Mapping 工具），可以使用任意数据库存储数据，并可以通过配置轻松地在不同的数据库之间切换

7、 实现了 WfMC 中的 ToolAgent 的概念，可以执行多达六种类型的外部应用

8、 可以使用定制的 java 类作为流程变量

选用 Shark 作为柔性改进的基础除了上面所述的优点外，Shark 还是目前唯一对 WfMC 规范支持得最全面的一个开源项目，其他的开源项目虽然也支持 WfMC，但是很多功能都没有实现。

5.2. 支持柔性框架的开源 WfMS 改进原型系统 F-WfMS

5.2.1. F-WfMS 系统架构

本文在 JaWE 和 Shark 的基础上改进并实现支持柔性框架及经过柔性扩展的 XPDL 的柔性工作流系统。本文将改进后的 JaWE 和 Shark 分别命名为 F-JaWE 和 F-Shark，由它们组成的柔性工作流系统被命名为 F-WfMS。F-WfMS 具有允许不确定性定义流程、动态绑定、子工作流程返回跳转以及更强的描述现实业务流程的能力。同时实现了本文提出的柔性框架，具有良好的流程重用性。

本文改进的 F-WfMS 的系统架构如图 5-2 所示，严格遵循第三章提出的工作流建立-配置-运行柔性框架。流程定义可以直接来自流程库，或者由组装模块根据流程库和配置库中的数据组装而成，实现对超流程的重用。配置库中的数据由配置工具依据流程库中的流程生成。同时，用户还可以在运行时通过活动池选择活动组合或直接调用流程定制工具 F-JaWE 新定制一个流程组装，同时新定制的流程被存放到配置库中，以后使用（原型系统中没有实现可视化配置，可以手工配置，详细说明见 5.3 节中对流程配置的测试相关内容）。

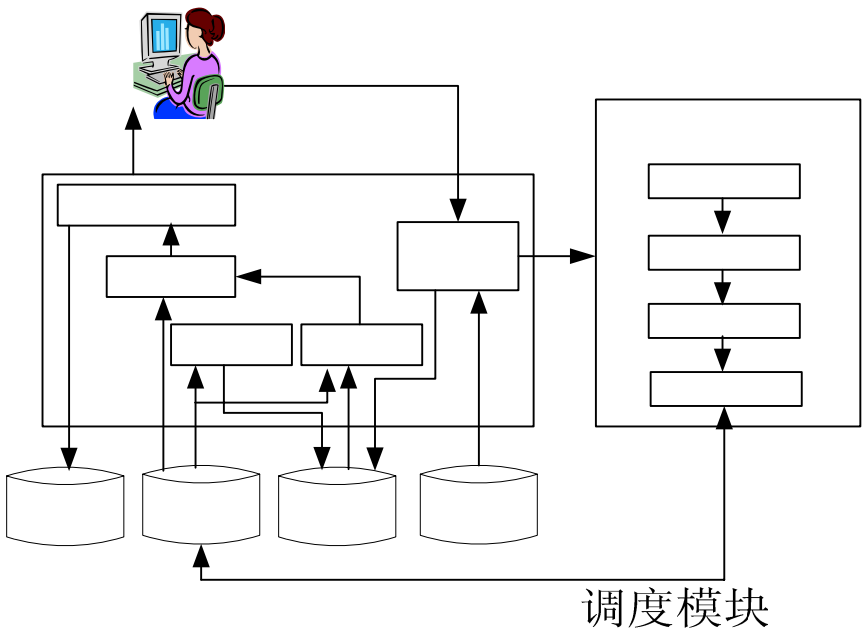


图 5-2 F-WfMS 系统架构

5.2.2. JaWE 的改进系统 F-JaWE

工作流的定制工具 JaWE 不仅提供了一个图形化的业务流程建模机制，同时还要保证定制出来的流程是切实有效合法。因此用户定制过程中每一步操作完成后都会对流程进行合法性校验。本文在对 WfMC 的流程定义语言标准 XPDL 扩展后，存在部分 JaWE 无法识别的概念扩展，JaWE 缺乏对这些概念扩展的正确性校验机制。因此改进 JaWE 的主要工作包括分析 JaWE 包校验机制并提供对各种扩展合法性校验的支持。由于改进后的 JaWE 对柔性支持更好，因此我们将其命名为 F-JaWE（Flexible JaWE）。

1、 JaWE 的包校验机制研究

包（Package）是 XPDL 元素实体中的根节点实体，由流程、参与者、相关数据、外部应用以及资源库定义组成，同时包之间可以相互引用，引用甚至可以是递归的。

由于包是一个 XML 实体，因此在校验之前先进行 XML 的结构校验以及合法性校验。XML 的编程接口分为两大类，文档对象模型（DOM,Document Object Model）和基于事件的 SAX（Simple API for XML）接口。前者将解析整个 XML 树并保存在内存中，而 SAX 则每当发现一个新的元素时产生一个报告事件。由于 JaWE 会反复的用到包的 XML 树，因此选用了 DOM 接口。

JaWE 的包校验由类 org.enhydra.jawe.xml. PackageValidator 完成。包校验接口的使用方法如下：

XMLInterfaceForJDK13 xmlI=new XMLInterfaceForJDK13();

```
Package pkg=xmlI.parseDocument(args[0],true);
PackageValidator validator = new PackageValidator(pkg,false, true,false,true,
                                                  false,false,"UTF-8");
validator.validateAll(false);
```

XMLInterfaceForJDK13 首先利用 DOM 接口进行包的 XML 解析，其 parseDocument 方法将从包的 DOM 树中提取各个节点内容生成 JaWE 中为 XPDL 各个元素定义的类实例。PackageValidator 接收 Package 对象以及其他相关参数初始化。最后调用其 validateAll 方法进行校验，该方法返回一个布尔变量，true 为正常返回，false 表示发生了错误。校验的错误信息将存放在 Map 类型的 parsingErrorMessages 中。

包校验的内容包括三个方面：

(1) 包的合法性校验。检验包中各个元素的语法及语义正确性，采用递归调用的方式一次遍历包中的各个元素进行校验。

(2) 可达性校验。检查流程定义中所有活动都是可达的。

(3) 一致性校验。该校验检验专门校验 Conformance Class 属性的语义（结构描述），该属性描述包中过程定义的结构限制，取值及其语义见表 5-2。

表 5-2 Conformance Class 属性的取值及其语义

可用取值	含义
FULL_BLOCK	流程中禁止分支、合并以及循环结构
NON_BLOCK	流程中禁止循环结构
FULL_BLOCK	流程中允许使用任意结构

在对 XPDL 扩展属性校验设计时，凡涉及到影响上述校验结果的情况都需要考虑改进。

2、 空子工作流程的校验

(1) 正确性校验

空子工作流程对子工作流程进行了语义上的扩展，从语义上要求正确性校验由定义阶段推迟到静态（动态）配置绑定阶段。校验逻辑基本上与子工作流程的相同，在子工作流程的校验上增加扩展属性判断逻辑，如果扩展属性指明其未绑定（即为空子工作流程），则取消此次校验，否则对子工作流程进行校验。

(2) 可达性校验

基于流程调用结构的层次性，空子工作流程不会造成流程的不可达。

(3) 一致性校验

空子工作流程绑定后即成为子工作流程，绑定前无一致性校验。

3、 子工作流程跳转返回的校验

子工作流程跳转返回的校验实际上是对有向图的的校验过程，在 4.3.2 一节中描述了算法的大致思想。对子工作流跳转返回的校验算法设计如下：

(1) 取扩展属性，判断是否存在跳转，不存在则退出，存在则转 2；

(2) 取出跳转节点，从跳转目标节点递归遍历流程图，中止递归的条件是节点所有后继都遍历完成或当前访问节点是源跳转子工作流程。依次将将每个被访问到的节点都插入链表表头，同时如果当前访问节点是源跳转子工作流程或该条路径上最后一个活动，则将其加入到 linklist 中；

(3) 处理 linklist，删除其中不等于源跳转子工作流程的元素，如果 linklist 为空，说明是快进跳转 op=”forward”，转 3-1），否则说明是回退跳转，op=”back”，转 3-2）；

(3-1)从源跳转子工作流程递归遍历流程（过程同上，唯一区别是遍历开始节点是源跳转子工作流程，碰到目标节点中止，因此遍历结果是源跳转子工作流程到目标节点之间被跨越的活动，如果这些活动存在不允许忽略的活动则跳转不合法），转 4；

(3-2)回退跳转形成了循环，因此需判断条件表达式避免死循环。条件表达式不存在或存在且为永真式，则记录错误（死循环），转 4；

(4) 遍历 linklist 中每个节点所指的链表，根据 op 的值检查扩展属性，禁止跨越的被跨越或禁止回退的被回退则记录错误。转 5；

(5) 如果存在错误，返回 false，否则返回 true；

算法的流程图如图 5-3。图中 Node 类的定义如下：

```
class Node{
   Activity dest;
   Node pre;
   Public Node(Activity act,Node n){
      dest = act;pre = n;
   }
   }。
```

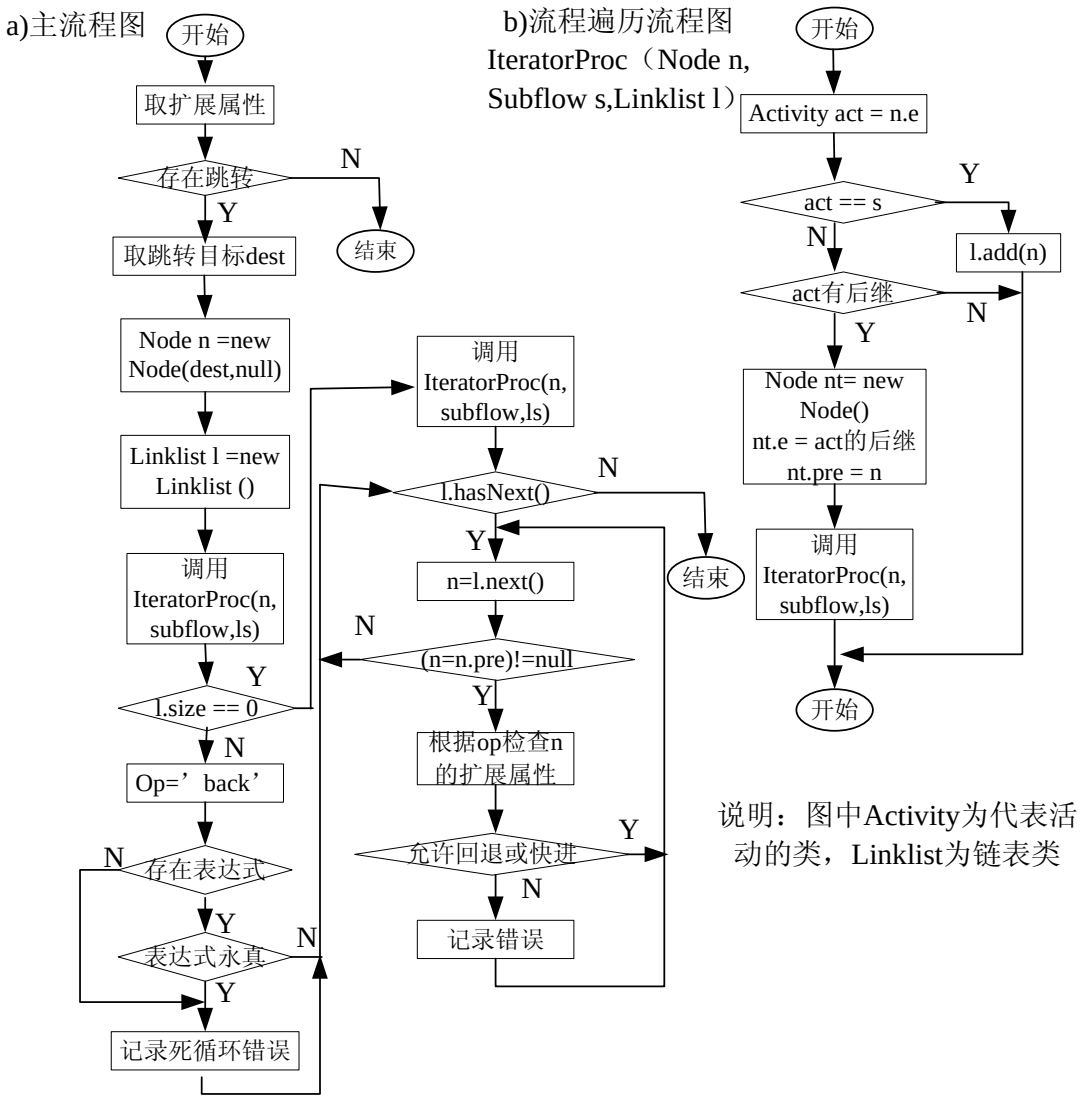


图 5-3 被调用流程向调用流程返回扩展的正确性验证流程图

4、 各种控制关联的校验

控制关联约束扩展的合法性校验算法与上述算法类似，通常也是对工作流图进行遍历，检查是否存在不符合约束要求的定义并给出相应的提示。

5.2.3. Shark 的改进系统 F-Shark

Shark 的改进内容包括配置工具、流程组装、流程正确性校验（详见 5.2.2 节）、空子工作流程的绑定与执行、子工作流返回跳转支持以及执行时对控制关联扩展的支持。由于改进后的 Shark 对柔性支持更好，因此我们将其命名为 F-Shark（Flexible Shark）。

1、 配置工具

流程配置分为静态流程配置和动态流程配置。无论哪种配置，配置信息包

括：配置内容、配置路径、配置元素及操作方式（添加、删除、修改）。

静态流程配置是指在流程定义完成后、运行前完成的配置，配置结果作为流程运行的参照，静态配置结果同样还可以被配置（因此每个静态配置项都会有一个 layer 属性说明其再配置树中的层次，保证层次高的先组装，层次低的引用了高层次流程的配置后组装——如果他们被先组装将会出错，因为它们引用的高层次的配置流程还没有组装出来，无法引用）。静态配置的流程可以实例化多个流程，其主要作用在于提供了良好的重用性，同时也具有一定的动态性。

动态流程的配置是指对处于运行状态的流程实例的流程定义进行配置，配置后的流程仅仅适用于该流程实例。动态流程配置不允许再被配置，仅影响配置时的流程实例，通常配置的内容也非常少，配置元素通常是属性级别的。动态流程配置的数据与静态流程配置的数据有所区别，动态配置由于影响范围小，配置内容少，因此不会重新组装，只会在其影响的流程实例推进时同时参考原流程定义及配置。

无论是静态配置还是动态配置，配置数据与流程定义语言的 XML 树形描述相对应，以树形的形式保存在数据库中，根节点是包元素，自根节点到每个配置节点是一条配置路径。每当用户配置一项数据时，便利用与 XPDL 元素相对应的类实例沿配置项数据的父节点追溯，直到根节点为止。途经的每个元素类型及 id 便是一条配置路径。

配置数据库表共四张：配置路径表 ConfigRoute、元素类型表 ElementType、操作权限表 OpRight 和配置数据表 ProcConfiguration。配置数据表存放的是具体针对某个 XPDL 元素的配置结果数据，但是要利用配置数据和被配置流程组装出配置流程来就需要借助于配置路径了。因为 XPDL 元素是递归定义的，这意味着 XPDL 元素就像操作系统中的文件或文件夹一样，需要上下文才有意义，仅元素的名称及其 ID 无法定为一个元素。因此在配置的过程中，同时便把配置元素只 XPDL 根节点的路径保存下来了。限于配置操作的权限，不同的元素具有不同的配置方式，故元素类型以及元素类型相关的操作权限都作为数据字典的形式保存。在系统运行前，元素类型以及相关操作权限表需要进行数据的初始化。配置数据库表设计 E-R 图如图 5-4 所示。

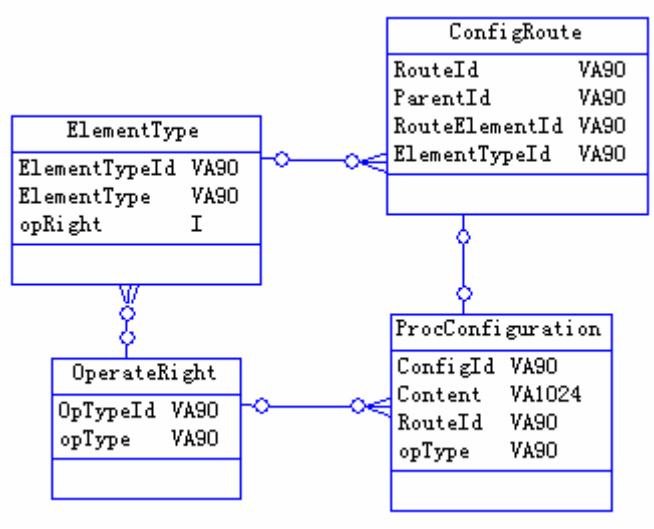


图 5-4 配置数据库表设计

元素类型为所有 XPDL 中定义的元素。基本的操作权限包括添加、删除与修改。所有的操作权限均在基础的操作权限之上进行组合得来。另外还增加一种 modifyBased 权限，指在修改的基础上生成一个新的元素，该权限只适用于 workflowprocess 元素，它有利于在包中添加一个基于已有的流程进行简单的修改而生成的新流程。

2、 空子工作流程绑定与执行

Shark 的流程以及活动的调度关系如图 5-5 所示。服务请求者可以是参与者或者子工作流程（子工作流程是活动的一种实现类型），它们可以请求流程定义相对应的管理器创建流程实例，流程实例结束后通知服务请求者。对于参与者而言，该接口没有具体实现，用户可以根据自己的需求定制，如将该模块定制成能发送短信的模块，则在流程实例完成时发送短信通知该流程实例的发起者。对于子工作流程而言，则是一种向调用流程的返回。

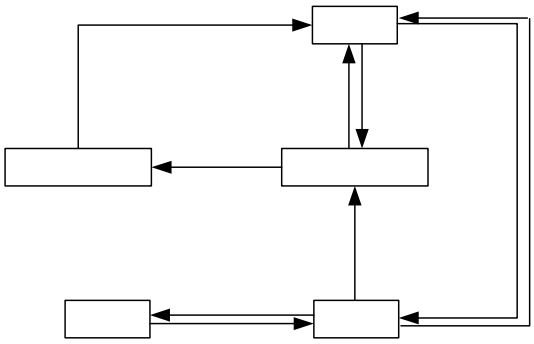


图 5-5 流程与活动的调度关系

通过对 Shark 流程与活动调度关系得分析可知，空子工作流程的绑定相当于参与者选择一个流程并启动。其执行于子工作流程相同，可以交由引擎 Shark 完成。

空子工作流程有可能在配置时完成了绑定工作，也有可能在运行时仍没有绑定。在运行时没有绑定的情况下，一旦碰到空子工作流程，引擎便无法进行流程运行导航了，此时引擎需要停止当前该流程实例的运行，提示用户指定一个流程与之绑定，完成绑定工作后再继续流程地执行。

运行时空子工作流程的绑定及运行逻辑如下：

(1) 在用户完成活动时，系统自动实例化下一活动，此时检测到该活动是空子工作流程则提示绑定空子工作流程，转 2）；否则转 6）；

(2) 提取流程相关数据用于工作流绑定时参数映射，并获取可用于绑定的流程列表；

(3) 用户选择绑定流程，系统提取该流程的形式参数；

(4) 用户完成形式参数到实际参数的映射；

(5) 检查参数类型一致性，不一致则绑定失败退出

(6) 完成绑定并实例化绑定流程；

3、 子工作流程跳转返回

子工作流的跳转返回分为两类：

(1) 静态配置跳转返回

静态配置跳转返回配置后跳转返回将影响到流程定义所有的流程实例。因此，为提高效率，静态跳转返回配置完成后，静态跳转返回扩展属性将被删除，同时根据跳转返回扩展属性描述直接在流程定义中增加转移元素。

(2) 动态配置跳转返回

动态配置跳转返回在子工作流程执行完毕时指定。动态配置跳转返回流程如图 5-6 所示。当子工作流程执行完毕时，设置跳转，设置完成后进行跳转的合法性验证，验证算法详见 5.2.2 节中对子工作流程跳转返回校验的阐述。对不合法的跳转，正常返回至子工作流程后的活动。对于合法的跳转将建立临时的跳转转移元素，并保存以备跳转返回完成时删除。接着跳转返回，实例化目标活动完成后即刻删除跳转转移元素。

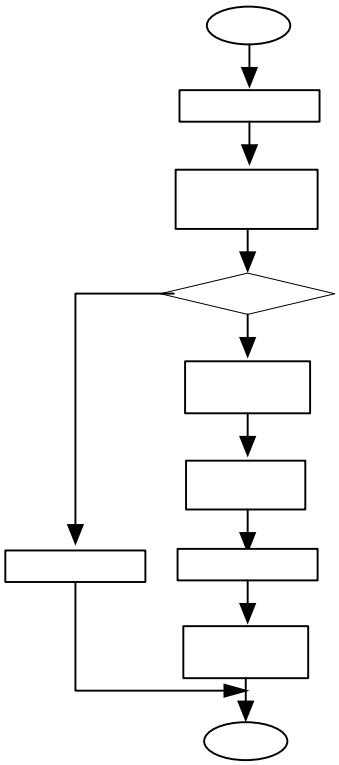
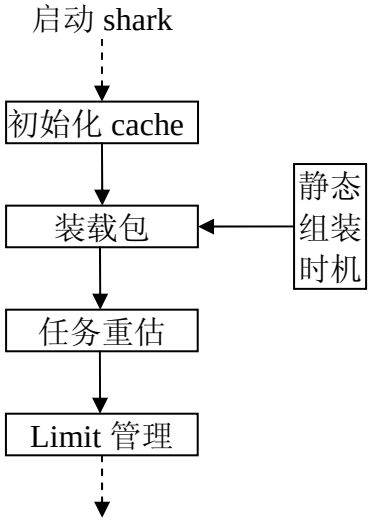


图 5-6 动态配置跳转返回流程图

4、 配置数据的组装

配置数据分为两类，静态配置数据和动态配置数据。根据配置数据的使用频度，不同的配置数据组装的时机不一。静态配置数据因静态配置流程影响多个实例将会被频繁的使用，因此在引擎启动的时候便事先根据配置数据为每个包构造一棵配置树，从书的根节点遍历每一条配置路径，最后完成配置。而动态配置仅仅影响一条流程实例，因此仅在需要的时候完成配置。



正常返回

图 5-7 静态组装时机

图 5-7 说明了 shark 的部分启动过程，初始化 cache 主要是根据系统配置文件中的配置数据决定在启动 shark 的时候加载哪些流程实例以及资源实例数据以优化系统速度，装载包实际上是在初始化 cache 中完成的，为便于说明特提取出来。静态组装的时机便选取在装载包的过程中完成。

5.3. 对 F-JaWE 和 F-Shark 的测试

通过对开源工作流定制工具 JaWE 和 Shark 的改进，F-JaWE 和 F-Shark 具备了更强的柔性。在改进的同时，我们还对改进后的系统进行了较全面的测试，提供了各种扩展合法与非法的示例流程的定义及其在F-JaWE和F-Shark的运行测试。图 5-8 是一个对子工作流程返回跳转扩展的测试示例。示例中的跳转源子工作流程是一个被调用的子工作流程，它执行完后将跳转到调用流程中的“回退目标”活动，但是由于调用流程中的 “不可重做”活动扩展属性被定义为 unreversible，即在流程实例的执行过程当中，只允许执行一次。而子工作流程返回跳转会触发“回退目标”活动的执行并进一步导致“不可重做”活动的执行，因此该流程是不合法的。

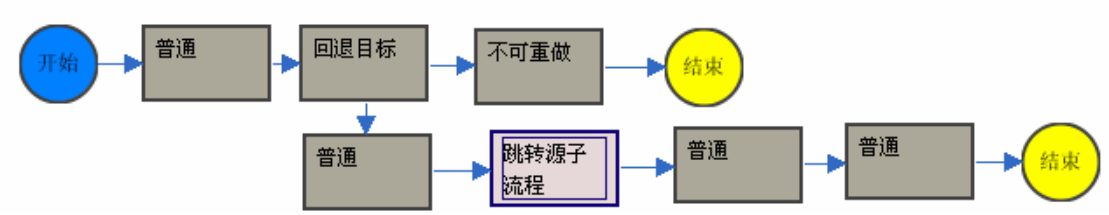


图 5-8 测试示例

测试结果如图 5-9 所示，系统提示流程的错误是因“回退跳转跨过了不允许回退的活动”引起的。

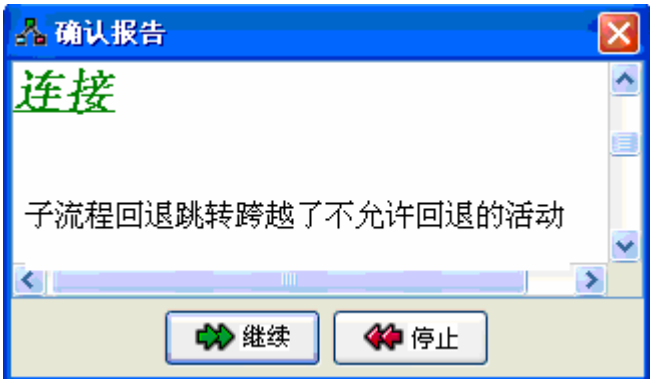


图 5-9 测试结果

上述流程在 F-Shark 上载的时候会提示同样的错误。

在流程的配置运行方面，我们提供了如表 5-3 的简单配置数据，包括配置路径与配置内容。配置内容非常简单，只是将转移的条件由原来的 Credit_Status == " OK" 修改成了 Credit_Status == "NOT_OK"。其前提条件是包 Business_Example 已经加载至工作流引擎。

表 5-3 示例配置数据

配置路径	**RouteId**	**Version**	**ParentId**	**RouteElementId**	**ElementType**
	107	1	107	Business_Example	Package
	108	1	107		WorkflowProcesses
	109	1	108	Business_Example_Wor1	WorkflowProcess
	110	1	109		Transitions
	111	1	110	Business_Example_Wor1_Tra55	Transition
	112	1	111		Condition
配置内容	**configureId**	**Version**	**ConfigContent**	**ConfigRoute**	**OpRight**
	113	1	Credit_Status == "NOT_OK"	112	modify

超流程 Business_Example_Wor1 及其子流程如图 5-10，配置前，活动 check finances 执行完后，如果 Credit_Status 的值为 OK，则继续活动 check stock，否则 decline order 后结束整个流程。配置后的条件正好相反，Credit_Status 的值为 OK 时会 decline order 并结束整个流程。

测试过程如图 5-11、图 5-12 所示。完成活动 check credit 时，我们将相关数据 credit_status 的值设置为 OK，完成后，引擎根据配置流程定义将活动执行推进到 decline order 活动，而并不如超流程所定义的推进到 check stock 活动。

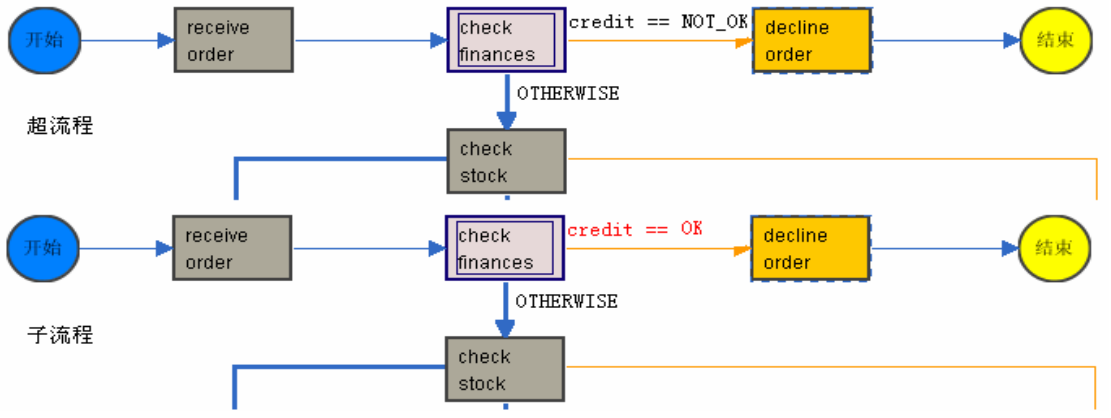


图 5-10 配置流程的超流程及子流程图示

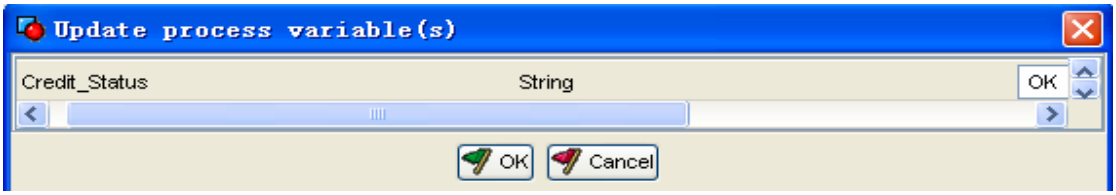


图 5-11 设置相关数据 credit_status

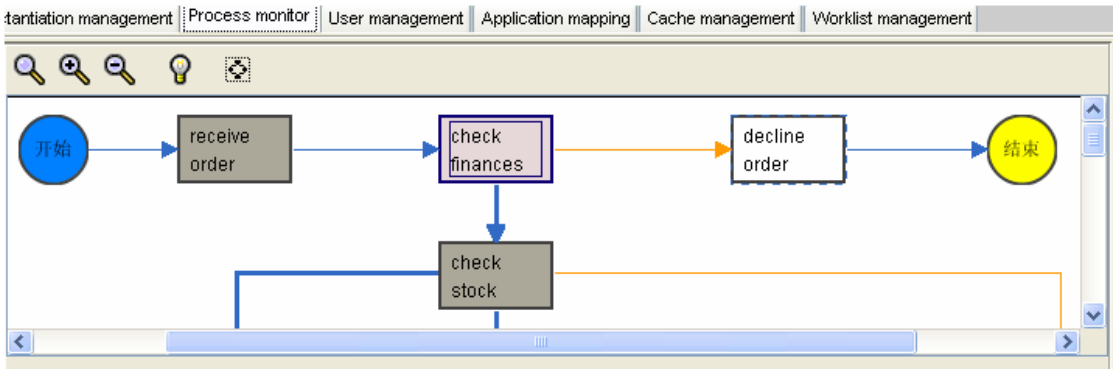


图 5-12credit_status=OK 时的执行路径

图 5-13 是子工作流程动态绑定测试界面。界面上方的组合框可选择被绑定的子工作流程，下方左侧是选中子工作流程的形式参数，中间是当前工作流的相关数据，双击相关参数可以建立实际参数到形式参数之间的映射。如图中所示，建立了 common 到 input1，test input2 的参数映射。图 5-14 是动态绑定后子工作流程的执行情况，其中灰色一项是动态绑定子工作流程的第一个活动。

图 5-13 空子工作流程的动态绑定

图 5-14 空子工作流程的动态绑定的执行

5.4. 小结

本章通过对多个开源工作流定制工具、引擎的比较与分析，最终确定选择 JaWE 和 Shark 组成 WfMS，作为柔性框架改进的基础。对开放源代码工作流管理系统的定制工具 JaWE 和引擎 Shark 进行了简要介绍。同时通过对 WfMC 制定的工作流定义语言标准 XPDL 进行空子工作流程及动态绑定、子工作流程跳转返回以及五种控制关联的扩展，完成了一个符合工作流建立-配置-运行柔性框架的系统原型的设计，并通过对 JaWE 和 Shark 的改进，实现了一个简单的原型系统，最后对该原型系统的柔性功能点进行了测试，测试结果表明该原型系统实现了柔性框架及 XPDL 相关扩展的柔性。

第六章 系统柔性分析

6.1. 系统柔性的评价指标

目前，对工作流柔性的定义仍然没有统一的标准，因此其评价指标也尚不成熟。Carlsen 在其博士论文中别出心裁的从考察同时代使用工作流技术的组织来分析工作流技术的柔性，借用 Morgan 对组织的八个比喻（组织是机器、有机体、大脑、文化、政治系统、精神监狱、变迁与转换、统治的工具） 来从组织的角度分析工作流的柔性[16]。他认为工作流的柔性应包括工作流的建模、执行、改进以及管理四个方面。其柔性需求可概括如表 6-1 所示。

表 6-1Carlsen 提出的柔性工作流支持需求

建模	执行	改进	管理
• 可变的流程定义 • 坚实的本体基础 —可扩展的元模型 • 面向用户 —图形化语言 —可理解的模型 • 覆盖 —交互 —流程目标 —组行为 —共享工作空间 —可用工具 • 重用 —模板 —交互 —个人片断	• 可变的流程定义 • 集成会话支持 • 柔性资源管理 —工作空间代理 —柔性角色 • 可调用群件工具 • 异常处理 • 活动对象 —复合文档 —软件代理 • 延迟绑定 —资源 —模型组件	• 流程建模的链接 会话支持 • 经验总结 • 辅助智能设计 —列表检查 —问题驱动	• 配置管理 • 重用模板 • 分层的资源库 —组织 —团队 —个人

Carlsen 博士对柔性需求的总结非常细致，内涵广而深。本文认为上述需求有一定的道理，相当多的要点非常的关键与精辟，但是就本文讨论的工作流平台柔性而言过于形式化，评价的可操作性不强。同时将非技术内容引入，而本文主要侧重于从技术上对工作流柔性改进进行评价。

Kammer 等人[61]对动态自适应的工作流（从本质上说其所指的动态自适应与本文定义的柔性存在一定的重叠）所需要完成的功能的定义中，也有很多值得参考的地方。这些功能包括：(1) 能动态地修改和编制工作流过程定义；(2) 工作流的执行模型可以配置：进行部分执行、指导性执行和强制执行；(3) 能够为活动、资源、agents 和 artifacts 分门别类地建立模型，保证在运行时有可选择性，并保证整个过程的一致性；(4) 具有自反性：工作流过程在执行时能够访问自己，并对自己的原模型重新修改；(5) 根据工作实例对过程模型进行演变；(6) 能够逻辑分解一个过程模型；(7) 能够利用过程片断和组件库；(8) 能够访问工作历史,并与专家交互；(9)为参与者提供相互通信的集成支持。其中(1)、(2)、(3)、(5)、(6)、(7)都与工作流的柔性有关。(4)的思想在程序设计中基本上被排斥，因为这样无论在程序的可理解性、可控性都存在着重大的缺陷。(8)侧重于适应性或者说智能性。(9)的功能更适合分布式技术来实现。

上述两种思想存在着一定的共性，一个细致入微，一个具有广泛的概括性。本文总结工作流柔性的关键点提取柔性特征如表 6-2 所示，这些特征都被细化，以增强在技术上评价的可操作性。

表 6-2 工作流柔性特征

可变的流程定义	静态（流程运行前）	属性可变
		拓扑结构可变
	动态（流程运行中）	属性可变
		拓扑结构可变
可扩展的元模型	元素实体扩充	
	元素实体语义扩展	
重用性	模块重用	
	继承重用	
延迟绑定	活动实现延迟绑定	
	任务分配	
辅助改进流程	流程历史	
	流程挖掘	
	流程改进自动决策	
松散耦合	资源映射	
	参与者映射	
	外部应用映射	

由于目前没有对工作流的柔性及其需求统一的系统的定义与评价系统，本文仅在参考 Carlsen 博士和 Kammer 等人对柔性需求说明的基础上，抽取关键点建立了一个能够反映工作流柔性核心内容的评价标准并对本文的工作进行了简要地总结。

表 6-3 改进原型系统与原系统柔性支持对比

柔性特征			原 WfMS 系统	改进 WfMS 原型系统
可变的流程定义	静态（流程运行前）	属性可变	不可变	静态配置属性，具体可配置属性参见表 3-1
		拓扑结构可变	不可变	静态配置子工作流返回跳转
	动态（流程运行中）	属性可变	相关数据可变	动态配置属性，具体可配置属性参见表 3-1
		拓扑结构可变	不可变	动态配置子工作流返回跳转
可扩展的元模型	元素实体扩充		支持 XPDL 标准，不能进行元素实体扩充，可以对元素实体进行语义扩展	同左
	元素实体语义扩展			
重用性	模块重用		块活动、子流程	同左
	继承重用		不支持	配置重用，见 5.2.3 配置工具的设计
延迟绑定	活动实现延迟绑定		不支持	支持，见 5.2.3 空子工作流的绑定与执行的设计
	任务分配		支持，任务接收、该派	支持，任务接收、该派
辅助改进流程	流程历史		保存流程执行数据	保存流程执行数据
	流程挖掘		不支持	不支持
	流程改进自动决策		不支持	不支持
松散耦合	资源映射		不支持	不支持
	参与者映射		支持	支持
	外部应用映射		支持	支持

对 JaWE 以及 Shark 组成的 WfMS 改进原型与原系统之间对柔性支持程度的对比如表 6-3 所示。从表中可看出，改进后的原型系统在柔性上比原有系统有较大的改善。

6.2. 本文采用方法的长处与不足

本文在改进工作流柔性的研究当中，主要采用了如下两种方法：

1、 提高工作流定义语言的描述能力，通过扩展 XPDL 子工作流程的概念，使其支持空子流程及其运行时的动态绑定、支持子工作流程多种跳转返回 增加了工作流定义时的不确定性，同时还允许工作流运行时的动态修改。

2、 引入面向对象思想，提高工作流定义的重用性，降低了工作流定义阶段与执行阶段之间的耦合性，使得工作流在定义后、运行前具有多态性。

目前通过提高工作流定义语言描述能力来改善工作流柔性的工作及研究比较多，他们大多对工作流进行重新建模，增加新的工作流元素实体，如专门定义一个可不确定性定义的柔性活动，或称之为黑盒，文献[22，24，25，47]采用的都是这种方法。由于建立了新的模型以及引入了新的工作流元素实体，与目前的 WfMC 的标准不兼容。而本文在采用提高工作流定义语言描述能力的方式提高工作流柔性时，充分考虑到了兼容性问题，直接在 WfMC 的流程定制语言 XPDL 的基础上进行扩展。

目前面向对象工作流的研究工作也存在着一定的局限性。对于面向对象工作流的认识主要有两类：应用了面向对象技术的工作流和引入了面向对象思想的工作流。前者仅仅是技术上的应用，对工作流特性本质没有改善。而针对后者的研究，很多文献仅仅限于面向对象的描述工作流实体及其协作[56,58,60]，文献[54]实现了面向对象的工作流语言 P 语言，这是一种类似于 C++的语言，需要编程以及编译程序以后才能够执行，对用户的计算机素质要求较高。本文没有提出新的流程定义语言，而是在 XPDL 的基础上，通过在工作流定制和运行阶段之间增加一个配置阶段，引入面向对象继承重用机制，提高了工作流定义的重用性。利用开源的工作流定制工具 JaWE 可以很方便的进行图形化的定制与配置继承。与 P 语言相比不足之处在于，缺乏一种描述工作流对象实体之间的协作关系的机制。

6.3. 小结

本章进一步的讨论了工作流柔性的需求，抽取关键点建立了一个能够反映工作流柔性核心内容的评价标准并根据该标准对本文提出的工作流建立-配置-

运行柔性框架及其原型实现系统进行了评价，结果说明该柔性框架及其原型系统具有较好的柔性，但在其分析工作流执行历史数据并辅助工作流模型改进方面存在一定的欠缺。

同时，归纳了本文研究过程中使用到的提高工作流柔性的方法，对本文的实现机制与现有同类方法的其他实现机制进行了对比，结果表明本文的实现机制具有很好的兼容性，不需要进行编程以及对程序的编译链接，接口友好。但是本文工作中没有涉及到描述工作流对象实体之间的协作关系的机制。

第七章 研究总结与展望

工作流的柔性是其在应用过程中提出的一个问题，其本身与业务流程联系得非常紧密，因此在研究工作流的过程当中我们必须要结合实际应用背景，通过不同的角度考察与研究，综合的解决工作流柔性的问题。

工作流柔性的不足限制了并在一定的程度上阻碍了其在各行业流程管理中的应用与发展。然而，工作流的柔性又是不能被无限制的放大的，因为工作流的初衷便是为了辅助企业管理业务流程并提高工作效率的，过分的强调柔性将会导致过多的要求人的参与，工作效率降低。因此在提高工作流柔性的同时，我们必须清醒地认识到，工作流的柔性不能无限制的提高，工作流的柔性与其提高业务流程自动性二者之间是一对必然的矛盾。于二者之间折衷，最大限度的满足对业务流程管理的需求才是我们研究提高工作流柔性，解决实际问题的最大目标。同时，工作流的柔性、动态性与适应性之间关系密切，存在着一定程度上的相互渗透。如本文中对工作流柔性的改进带来了动态性的问题，同时流程运行过程中的动态修改又有助于流程执行过程中异常事件的处理，提高工作流的适应性。

本文所做的主要工作总结如下：

1、 在总结前人研究的基础之上，结合工作流在应用中的实际情况，完成了对工作流柔性的综合定义。

2、 在工作流中引入面向对象思想，提出了工作流建立-配置-运行的柔性框架。深入研究工作流管理联盟元模型和流程定义语言 XPDL 标准的基础上，讨论了柔性框架下可配置的内容、配置的基础理论、分类、规则、机制及其实现。

3、 讨论了目前通过提高工作流定义语言提高柔性的相关工作及其不足。在 XPDL 的基础上利用其扩展属性进行了兼容性语义扩展。通过对子工作流程语义的扩展，支持空子工作流程及跳转返回，增加了流程支持不确定性定义的能力。同时提出了五种控制关联扩展，增强了工作流定义语言对现实业务流程的描述能力。

4、 通过对 WfMC 的 XPDL 的柔性扩展以及对开放源代码的工作流系统的研究，完成了对工作流定制工具 JaWE 和引擎 Shark 的改进，使其改进版 F-JaWE 和 F-Shark 成为支持工作流的建立-配置-运行柔性框架以及工作流定义语言 XPDL 柔性扩展的原型系统。该原型系统作为科创工作流平台研究试探性的成果，对解决目前工作流在电子政务应用中所表现出的柔性不足的问题，具有指导性的意义，并已应用到以工作流为平台的电子政务系统中。

由于个人资质及时间的限制，本文做出的研究工作还很肤浅，仍然有很多的

问题值得进一步的研究。这些方向大致如下：

1、 作为一个业务流程管理的平台，工作流的研究需要结合实际业务流程及其运行环境，工作流的柔性提高还可以从其部署的环境结合组织模型、流程集成等方面来进行研究。

2、 工作流的柔性与其对业务流程自动性的支持这一对矛盾之间的关系及其在应用中的取舍与折衷

3、 工作流的细粒度执行是本文提到过却没有实现的，细粒度的工作流执行不但有利于提高工作流的柔性，同时在一定的程度上由于状态机的引入可以提高工作流的智能性。

4、 本文主要研究的是从工作流平台本身内在因素改善其柔性，而一些新的技术的引入其实同样也可以改善工作流的柔性

5、 将工作流的柔性、动态性、适应性以及智能性相结合，研究它们之间相互的促进与制约关系

6、 引入数据挖掘，对工作流执行历史数据进行分析，提高工作流辅助用户改进模型的能力，使其具有一定的智能性。

参考文献

[1] WfMC. WfMC-TC-1003.Workflow Management Coalition Workflow Standard:The Workflow Reference Model . Winchester, UK: Workflow Management Coalition,1995-1-19

[2] WfMC. WfMC-TC-1025.Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface -- XML Process Definition Language (XPDL 2.0).Florida,USA:Workflow Management Coalition, Lighthouse Point, 2005-8-3.

[3] WfMC. WfMC-TC-1023. Workflow Management Coalition Workflow Standard: Wf-XML2.0,XML Based Protocl for Run-Time Integration of Process Engines.Winchester, UK:Workflow management Coalition,2004-8-8

[4] WfMC. WfMC-TC-1009.Workflow Management Coalition Workflow Standard: Workflow Management Application Programming Interface (Interface 2&3) Specification.Winchester, UK: Workflow Management Coalition,1996

[5] 工作流之大局势, http://blog.csdn.net/hongbo781202/archive/2004/09/26/117271.aspx

[6] Kobielus J G. Workflow Strategies. Foster: IDG Books WorldWide, Inc. , 1997

[7] Das S. ORBWork: a distributed CORBA-based runtime for the METEOR2 workflow management system [MS Thesis].University of Georgia, 1997

[8] Alonso G, Agrawal D etc. Exotica/FMQM: a persistent message-based architecture for distributed workflow management. Technical Report, RJ9912, IBM Almaden Research Center, 1994

[9] Geppert, A. and Tombros, D. Event-based Distributed Workflow Execution with EVE, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), The Lake District, England, September 1998. 427-444

[10] Cai Ting, Gloor A , Nog S. DartFlow: a workflow management system on the web using transportable agents. Technical Report, PCS2TR962283, Dartmouth College, 1996

[11] 范玉顺等.工作流管理技术基础.北京：清华大学出版社，施普林格出版社.2001

[12] 岳晓丽，杨斌，郝克刚.信牌驱动式工作流计算模型.计算机研究与发展，2000，37(12)：1513-1519

[13] Open source project JaWE.http://sourceforge.net/projects/jawe/

[14] Open source project nezha.http://sourceforge.net/projects/nezha/

[15] Open source project willow. http://www.huihoo.org/willow/

[16] Carlsen S. Conceptual Modeling and Composition of Flexible Workflow Models. [PhD Thesis]. Department of Computer Science and Information Science, Norwegian University of Science and Technology, Norway, 1997.

[17] 赵文，胡文蕙，张世琨，王立福.工作流元模型的研究与应用.软件学报，2003，14(06)：1052-1059

[18] 徐俊，沈康辰，黄柏林.着色 Petri 网在工作流建模中的应用.计算机应用与软件，2004，21(7)：47-48

[19] 李慧芳，范玉顺.工作流时间管理.软件学报，2002，13(8)：1553-1558

[20] 范玉顺，吴澄.一种提高系统柔性的工作流建模方法研究.软件学报，2002，13(4)：833-839

[21] 何鹤立，倪小平，盛步云，罗　丹.工作流管理系统的柔性技术.计算机工程，2003，30(6)：63-66

[22] Shazia Sadiq, Wasim Sadiq. Pockets of Flexibility in Workflow Specifications. 20th International Conference on Conceptual Modeling, Yokohama Japan, 2001.189-196

[23] Belinda M. Carter,Joe Y. etc. Customizing Internal Activity Behaviour for Flexible Process Enforcement.Fifteenth Australasian Database Conference, Dunedin,New Zealan,Conferences in Research and Practice in Infomation Technology,2004.189-196

[24] Peter Mangan, Shazia Sadiq. On Building Workflow Models for Flexible Process. In: Proceedings of the thirteenth Australasian conference on Database technologies - Volume 5 ,Melbourne, Victoria, Australia,2002.103 – 109

[25] 孙瑞志，史美林.支持工作流动态变化的过程元模型.软件学报，2003，14(1)：62-67

[26] Carlo Combi, Giuseppe Pozzi.Architectures for a Temporal Workflow Management System. Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, Database theory, technology and applications (DTTA),2004. 667 – 673

[27] Andrzej Cichocki, Marek Rusinkiewicz.Providing Transactional Properties for Migrating Workflows.Mobile Networks and Applications,2004,Vol 9:473-480

[28] Shawn Bowers, Bertram Ludascher. An Ontology-Driven Framework for Data Transformation in Scientific Workflows. International Workshop on Data Integration in the Life Sciences (DILS 2004), volume 2994 of LNCS:1-16

[29] Keith D. Swensen, Kent Irwin (1995). Workflow Technology: Tradeoffs for Business Process Re-engineering. Proceedings of ACM Conference on Organizational Computing Systems (COOCS 95), Milpitas, CA. USA, Nov 1995.22-29

[30] Yongyi Xie, Weishi Zhang. Component-Based Workflow Architecture of a Distributed Software Process Management System. Third International Conference On Quality Software,Dallas, Texas, 2003.204-210

[31] 王跃，刘卫东，王诚. 基于 Agent 的工作流系统中的异常处理. 计算机工程与应用，2003，7：177-179

[32] Martin Pruvis, Maryam Purvis, Selena Lemalu.An Adaptive Distributed Workflow System Framework.Proceedings of the Seventh Asia-Pacific Software Engineering Conference 2000. 311

[33] 丁柯，金蓓弘，冯玉琳. 事务工作流的建模和分析. 计算机学报，2003，8. 26(10)：1304-1311

[34] 王小明，赵宗涛，郝克刚. 工作流系统带权角色与周期时间访问控制模型. 软件学报，2003，14(11)：1841-1848

[35] 沈军营，黄进，严隽琪，蒋祖华. 基于 CORBA 的异地协同工作流模式. 计算机应用，1999，19(9)：19-22

[36] The state of workflow. http://www.jboss.com/products/jbpm/stateofworkflow

[37] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and Limitations of Current Workflow Management Systems. IEEE Expert, 12(5), September-October 1997

[38] M. Rusinkiewicz, A. Sheth. Specification and Execution of Transactional Workflows. In: W. Kim (ed.), Modern Database Systems --- The Object Model, Interoperability, and Beyond, Addison-Wesley 1995. 592—620

[39] C.Mohan. Recent trends in workflow management products, standards, and research.1997. http://www.almaden.ibm.com/cs/exotica/wfnato97.pdf

[40] W.M.P. van der Aalst. Three good reasons for using a Petri-net-based workflow management system. Proceedings of the International Working Conference

on Information and Process Integration in Enterprises (IPIC'96), Cambridge, Massachusetts, Nov. 14-15, 1996. 179-201

[41] WfMC. WfMC-TC-1011,Workflow Management Coalition Terminology & Glossary. Winchester, UK: Workflow Management Coalition,1999. 2

[42] Ader, M**.** seven workflow engine reviewed**.** Document World, 2(3), 1997.

[43] Gustavo Alonso, Claus Hagen, Divyakant Agrawal, Amr El Abbadi, C. Mohan. Enhancing the Fault Tolerance of Workflow Management Systems, IEEE Concurrency, 08(3) ,2000. 74-8

[44] Pinar Koksal, Ibrahim Cingil, Asuman Dogac.A Component-based Workflow System with Dynamic Modifications. Proceedings of the 4th International Workshop on Next Generation Information Technologies and Systems,1999. 238 – 255.

[45] C.Liu, M.E. Orlowska, H.Li. Automating Handover in Dynamic Workflow Environments. CaiSE 1998. 139 –157

[46] ZHAO Wen, HU Wen-Hui, ZHANG Shi-kun, WANG Li-Fu. Study and Application of a Workflow Meta-Model. Journal of Software，2003，14(6)：1052-1059(in Chinese)

[47] ShuiGuang Deng, Zhen Yu, ZhaoHui Wu, LiCan Huang. Enhancement of workflow flexibility by composing activities at run-time. Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, Database theory, technology and applications (DTTA),2004. 667 – 673

[48] Asynchronized service access protocol. http://www.oasis-open.org/apps/org/workgroup/asap/index.php

[49] [美］Wendy Boggs，Michael Boggs 著，邱仲潘等译. UML with Rational Rose 从入门到精通，北京：电子工业出版社，2000

[50] 面向对象：继承与接口. http://www.xztvu.cn/kfy/teaching/2001a/java/javazuoye5.htm

[51] 刘昕. SOA 标准之祸. 信息系统工程，2004，8：54-56

[52] Kwak M, Han D, Shim J. A Framework Supporting Dynamic Workflow Interoperation and Enterprise Application Integration. In: Proc. of the 35th Hawaii Intl. Conf. on System Sciences,Hawaii, 2002. 305-35

[53] 朱锦泉**.**可适应性工作流模型及其实现机制研究：[博士学位论文]**.**长春：吉林大学，2005

[54] Guangxin Yang, Process Inheritance and Instance Modification. In: Proceedings of ACM Conference on Supporting Group Work, Sanibel Island, Nov. 2003. 229-238

[55] Bogia, D.P. and Kaplan, S.M. Flexibility and control for dynamic workflows in the wOrlds environment. Proceedings of the Conference on Organisational Computing Systems, ACM Press, Milpitas, CA, November 1995.148-159

[56] Chen,J.CSPL:an Ada95-like,Unix-based process environment.IEEE TOSE,1997,23(3):171-184

[57] R. Conradi, et. al. EPOS: Object-oriented cooperative process modeling. In A. Finkelstein, J. Kramer, and B. Nuseibeh, editors, Software Process Modeling and Technology, 1994. 33-70

[58] Jaccheri,M.L.,Picco,G.P.,and Lago,P.Eliciting software process models with the E^3 language.ACM TOSEM,1998,7(4):368-410

[59] Malon,T.W.,et al.Tools for inventing organizations:toward a handbook of organizational processes.Management Science,1999,45(3):425-433

[60] van der Aalst , WMP; Basten, T. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science, 270(1-2): 125-203

[61] Peter J. Kammer, Gregory Alan Bolcer, Richard N. Taylor, Mark Bergman. 2000.Techniques for Supporting Dynamic and Adaptive Workflow, Computer Supported Cooperative Work, 9(3-4), 269-292.

[62] 尹建伟，陈刚，董金祥.柔性工作流动态行为建模方法.计算机辅助设计与图形学学报，2002，14(10)：933-939

[63] 解放，曹江辉，柯文，王宁生. 柔性化的产品开发工作流模型的研究. 计算机工程与应用，2003，8：39-31

致 谢

研究生生活悄然落幕，我用三年时间的付出换取亲情、友情、知识和经验的财富。也许这是一篇微不足道的学术论文，从发表的那一天起就注定使命将匆匆结束，除了作者导师、答辩委员会的委员们、所里的各位同学们以及作者本人知道其曾经存在过，再无人翻阅。但其创作过程及众多不为人知的细节将永驻我心。衷心的感谢导师李建华教授三年里对我在学术上不倦的指导、生活上无微不至的关怀和您为人师表所给予我对人生的顿悟。本文不仅仅是浩瀚学术海洋中之一粟，也是李老师辛勤培养的桃李之花。 同时也要感谢谭立球副教授不辞劳苦地组织好所里的每一次学术讨论及学术上给予我们的指引。感谢实验室各位师姐师兄、同学以及师弟师妹们与我共度了三年宝贵而美好的时光，填满了我生活中每一个空虚的缝隙。感谢我的父母及恋人给予了我默默无闻的关怀与支持。最后要感谢参加论文评审和答辩委员会的各位老师。

祝愿每一位帮助、关心和支持过我的人平安幸福！

攻读学位期间主要的研究成果

已发表学术论文：

1. 徐海军，李建华. SVG 在工作流图形监控中的应用.《信息技术》,2006,2：28-30

参加的科研项目：

1. 开源工作流平台的研究及其在电子政务中的应用
2. 长沙县电子政务系统