ICS 35.080
L 77

SZDB/Z

深圳市标准化指导性技术文件

SZDB/Z 17.5-2008

深圳市电子政务应用服务规范 第5部分：应用服务运行管理框架规范

Electronic Government Application Service Specification—

Part 5 : Application Service Running and Management Framework Specification

2008-11-18 发布　　2008-12-01 实施

深圳市质量技术监督局发布

目　　次

前　言 II
1 范围 1
2 规范性引用文件 1
3 术语和定义 1
3.1 名词解释 1
3.2 术语定义 1
4 元数据 2
4.1 元数据描述方法 2
4.2 应用服务及服务组件元数据 3
4.3 服务模块元数据 4
5 应用服务框架组成 5
5.1 工作原理 5
5.2 总体框架 6
6 应用服务框架技术要求 6
6.1 服务库 6
6.2 发布模块 7
6.3 调用模块 7
6.4 管理模块 7
6.5 监控模块 7
7 接口定义 7
7.1 服务组件管理接口 8
7.2 服务模块管理接口 9
7.3 应用服务管理接口 10
7.4 服务库管理接口 12
7.5 获取应用服务接口 13
7.6 异常约定 14
附 录 A （规范性附录） 服务组件描述模式 Schema 16
附 录 B （资料性附录） 服务组件描述样例 20
附 录 C （资料性附录） 深圳市福田区某并联审批系统应用示例 22
参考文献 38

前　言

SZDB/Z 17-2008《深圳市电子政务应用服务规范》目前分为10个部分：

——第1部分　《总则》
——第2部分　《应用系统分类及代码规范》
——第3部分　《应用系统描述规范》
——第4部分　《组织身份模型数据规范》
——第5部分　《应用服务运行管理框架规范》
——第6部分　《组织身份服务接口规范》
——第7部分　《访问控制服务接口规范》
——第8部分　《单点登录服务接口规范》
——第9部分　《电子表单服务接口规范》
——第10部分　《业务流程服务接口规范》

本部分为SZDB/Z 17-2008的第5部分。

本技术规范适用于深圳市各级党政机关的信息化建设工作。对于本部分未能涵盖的内容将依据本技术规范的编写原则对本部分内容进行扩充。

本技术规范文件由深圳市信息化领导小组办公室、深圳市福田区信息中心提出。

本技术规范文件由深圳市信息化领导小组办公室归口。

本技术规范文件由深圳市信息化领导小组办公室、深圳市福田区信息中心、北京有生博大软件技术有限公司共同起草。

本技术规范文件主要起草人：贾兴东、陈朝祥、张雁、高新辉、王克照、石卫宁、赵斌、李淼、周礼洪、杨海波、王姝、张焕焕、刘用军、梁文龙等。

本技术规范文件为首次发布。

深圳市电子政务应用服务规范
第5部分：应用服务运行管理框架

1 范围

本部分定义了应用服务、服务组件和服务模块，以及它们之间的关系，规定了相应的元数据和元数据的扩展原则及方法，给出了应用服务运行管理框架的组成部分和各部分的功能和技术要求，提供了应用服务框架之间的分布式调用机制，规定了应用服务发布、调用和管理的技术要求和服务接口。

本部分主要用于深圳市各级党政机关的信息系统规划与建设，以及电子政务信息系统建设的系统集成商、软件开发商和监理单位进行信息化规划、建设。适用于构建共性的、基础的、开放的应用支撑平台，为应用系统建设提供基础的应用服务。并可将原有应用系统中可重用、可共享的功能单元服务化，利于应用系统整合。各应用系统在不同应用服务框架之间，采用对等的分布式调用机制，通过应用支撑平台的互操作调用，可实现互联互通。

2 规范性引用文件

下列文件中的条款通过本部分的引用而成为本部分的条款。凡是注日期的引用文件，其随后所有的修改单（不包括勘误的内容）或修订版均不适用于本部分，然而，鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本部分。

GB/T 19488.1—2004　电子政务数据元 第1部分：设计和管理规范
GB/T 21062.3-2007　政务信息资源交换体系 第3部分：数据接口规范
GB/T 21062.4-2007　政务信息资源交换体系 第4部分：技术管理要求
GB/T 21064-2007　电子政务系统总体设计要求
GB/Z 19669-2005　XML在电子政务中的应用指南
SZDB/Z 17.1-2008　深圳市电子政务应用服务规范 第1部分：总则

3 术语和定义

3.1 名词解释

local**调用：**

本地调用，指调用方与被调用方处于同一个运行环境中，并可直接通过API的方式进行调用。

remote**调用：**

远程调用，指调用方与被调用方处于不同的运行环境或不同的主机、网络环境中，通过Web服务的方式进行远程调用。

lazy **调用：**

延迟的、延后的调用，指在调用服务时先获得服务的代理对象，只在实际执行调用时才真正发起调用。

3.2 术语定义

3.2.1 应用服务

应用服务是电子政务应用系统中公共的、可定义的、可注册的和可调用的功能单元。

应用服务作为公共服务，在更高层的基础上提供软件复用和业务复用，可通过应用服务元数据明确定义，可通过应用服务框架发布并注册在服务库中，可将第三方发布好的服务注册在服务库中，可通过应用服务框架进行监控和管理，可随时透明的被本地或远程查找和调用。

应用服务应该满足如下技术要求：

1. 同时以API和Web服务的形式提供服务。
2. 应用服务都是无状态的，每次对应用服务的调用都具有完整语义，与上下文无关。
3. 消息传输基于HTTP/1.1(RFC 2616)或JMS协议。
4. 采用W3C的SOAP 1.2作为消息封装格式。
5. 采用W3C的WSDL 1.2作为服务描述规范。

3.2.2 应用服务运行管理框架

应用服务运行管理框架简称应用服务框架，是应用服务的运行、监控、管理的框架。

应用服务框架提供了统一的服务库注册、存储、查询的应用服务元数据信息，提供了发布、调用应用服务的功能，可对应用服务及调用进行监控、管理，同时提供了本地和远程调用，可支持分布式应用和负载均衡。

在不同的应用服务框架之间，采用对等的分布式调用机制，可注册远程的服务库到本地。可通过应用服务框架之间的互操作调用，实现互联互通。

应用服务框架本身不提供访问控制的功能，但可借助访问控制服务模块实现对应用服务的认证、授权和访问控制。

应用服务框架作为软件基础设施，通过部署在上面的服务模块，以标准的协议对外提供服务，可实现更高层次的软件复用和业务复用，可将原有应用系统中可重用、可共享的功能单元服务化，利于应用系统整合。

应用服务框架提供高度可集成的能力，采用标准的Web服务协议组作为服务接口描述和调用规范，可屏蔽不同软件平台的差异，实现透明的互操作。

3.2.3 服务模块

服务模块是进行部署的最小单位，是满足某些特定功能需求的一组相关应用服务的集合，可以是软件包的形式，也可以是第三方提供的应用服务集合的形式。服务模块可通过元数据描述文件（附录A：服务组件描述模式schema）描述并部署在应用服务框架上，也可通过应用服务框架提供的界面或API来部署，由应用服务框架实行统一的监控和管理。

3.2.4 服务组件

服务组件是服务模块的基本组成元素和基本构建单位，是粒度最小的实现和发布单元，是相关的一组应用服务的具体实现，它的功能以应用服务的形式提供。服务组件具有可设置的属性，其属性是可以改变服务功能的数据。

服务模块由一个或多个服务组件及相关配置信息构成。

4 元数据

4.1 元数据描述方法

采用摘要表示的方法定义和描述元数据，摘要包括以下属性：中文名、英文名、数据类型、值域、约束、说明。

4.1.1 中文名

元数据的中文名称，中文名称在同一类元数据中是唯一的。

4.1.2 英文名

元数据的英文名称，英文名称在同一类元数据中是唯一的，比较时不区分大小写。

可包含的字符为大小写的英文字母、数字，所有组成词汇为无缝连写。

4.1.3 数据类型

元数据的数据类型。

4.1.4 值域

元数据可以取值的范围。

4.1.5 约束

元数据的约束性条件，包括是否非空、最大出现的次数、是否唯一。

4.1.6 说明

对元数据含义的进一步的解释及补充说明。

4.2 应用服务及服务组件元数据

通过应用服务元数据对应用服务进行描述，发布、查找和调用应用服务时都需使用元数据信息。应用服务和服务组件都使用相同的元数据描述。

本部分定义了核心元数据，即所有应用服务描述中共性的、必不可少的元数据。

序号	中文名	英文名	数据类型	值域	约束	说明
1.	唯一标识	id	字符串	ID..200	非空	应用服务的唯一标识，如ProcessInstanceService。对同一服务组件、同一版本的“唯一标识”保持唯一。
2.	中文名称	name	字符串	C..200	非空	应用服务的中文名称，如流程实例服务组件。
3.	版本	version	数值	从1.0开始	非空	应用服务的版本号，如1.0，前面为主版本号，小数点后面为小版本号。
4.	描述	description	字符串	C..2000		应用服务的描述信息
5.	服务WSDL的URL地址	wsdl	字符串	C..2000		应用服务WSDL的URL地址，要求符合RFC 1738,Uniform Resource Locators (URL)规范，仅当注册的服务为第三方提供时才需要，否则此值由应用服务框架自动生成。
6.	所属分类	category	字符串	ID..200		应用服务所属的分类，以“/”开头，如workflow下的/instance分类，可划分多级分类。
7.	服务接口	service	字符串	ID..200		应用服务提供的接口，采用interface属性描述本服务提供的接口。如果前面有WSDL的地址，不需要本属性。
8.	具体实现	implementation	字符串	ID..200		应用服务接口的实现类，采用class属性描述实现类，用factory-method属性描述实现类的工厂方法，如果有factory-method则使用此方法实例化，否则直接创建实现类。如果前面有service属性，必须指定实现类。

9.	签名信息	signature	字符串	ID..2000		应用服务的签名信息,对服务的名称、版本号进行签名运算，可以验证服务的提供方。
10.	发布时间	publishDate	日期时间	D17		应用服务的发布时间,按照GB/T 7408—2005执行，格式为CCYYMMDD HH:MM:SS。
11.	更新时间	updateDate	日期时间	D17		应用服务的更新时间,按照GB/T 7408—2005执行，格式为CCYYMMDD HH:MM:SS。
12.	运行状态	state	字符串	running|stoped | unknown		应用服务的运行状态,应用服务注册后，即处于running状态，通过changeServiceState接口可改变运行状态。
13.	调用风格	style	字符串	rpc|document		应用服务的调用风格,最大出现次数为1。缺省为rpc风格。rpc指接口驱动的风格，document指文档驱动的风格。
14.	调用模式	mode	字符串	sync|async		应用服务的调用模式,最大出现次数为1。缺省为sync。sync|async指同步|异步模式。
15.	传输协议	transport	字符串	HTTP|JMS		应用服务传输协议,最大出现次数为1。缺省采用HTTP传输协议。
16.	扩展元数据名称	name	字符串	ID..200		扩展元数据的名称,最大出现次数为N，不可重复。
17.	扩展元数据值	value	字符串	ID..2000		扩展元数据的值,最大出现次数为N。

4.3 服务模块元数据

序号	元数据	英文名	数据类型	值域	约束	描述
1.	唯一标识	id	字符串	ID..200	非空唯一	服务模块的英文名称，不能重复。
2.	中文名称	name	字符串	ID..200	非空	服务模块的中文名称
3.	版本	version	数值	从1.0开始	非空	服务模块的版本号,从1.0开始,最大出现次数为1，如1.0。
4.	服务提供方	provider	字符串	C..200		服务的提供方的名称,最大出现次数为N。
5.	厂商	vender	字符串	C..200		服务的厂商，最大出现次数为N
6.	单位名称	name	字符串	C..200		服务的提供方|厂商的名称，最大出现次数为N。
7.	地址	address	字符串	C..200		服务提供方|厂商的联系地址
8.	联系人	contact	字符串	C..200		服务提供方|厂商的联系人

9.	联系电话	tel	字符串	C..200		服务提供方|厂商的联系电话
10.	电子邮件	email	字符串	C..200		服务提供方|厂商的电子邮件
11.	网站	website	字符串	C..200		服务提供方|厂商的网站
12.	依赖	dependence	字符串	C..200		服务模块依赖的其它服务模块的唯一标识，最大出现次数为N。如workflow服务组件依赖org、ac、form服务组件。

5 应用服务框架组成

5.1 工作原理

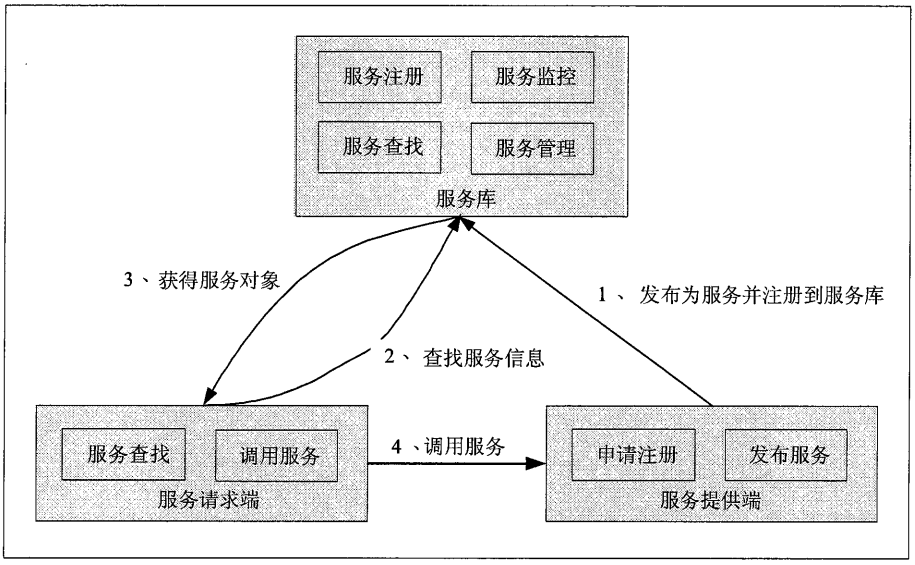


图1 应用服务工作原理图

应用服务框架提供了应用服务的发布、注册、查找、调用、监控、管理功能，其中涉及三个角色：服务提供者、服务请求者、服务库。图1所示，应用服务工作原理如下：

1. 服务提供者开发符合应用服务技术要求的功能单元，将开发好的功能发布为应用服务，并将应用服务在服务库中注册。
2. 服务请求者在服务库中查找所需服务，根据返回的结果确定需要调用的应用服务。
3. 服务请求者根据需要调用的应用服务，获得应用服务的代理对象。
4. 服务请求者发起调用请求，对应用服务进行实际调用，并获得服务返回的结果。

在整个提供服务的过程中，三个角色的基本功能如下：

1. 服务提供者：发布服务、进行注册。
2. 服务请求者：查找服务、调用服务。
3. 服务库：服务注册、服务查找、监控管理。

5.2 总体框架

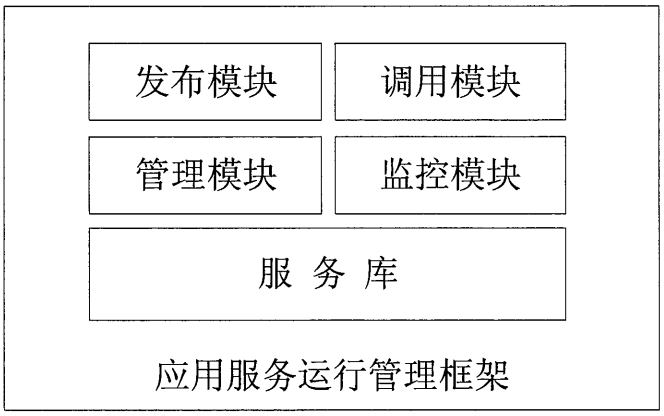


图2 应用服务框架基本结构示意图

如图2所示，应用服务框架由服务库、发布模块、调用模块、管理模块和监控模块五部分组成。

服务库提供对应用服务元数据的注册、存储和查找功能，可以将服务模块、服务组件和应用服务注册在服务库中。

发布模块提供将功能单元服务化的功能,读取并解析注册描述文件，将描述在其中的接口发布为应用服务，并自动注册到服务库中。

调用模块封装对应用服务的调用过程，屏蔽技术实现细节，直接实现对应用服务的调用，可与应用服务框架部署在一起,也可单独部署在应用服务的调用端。调用模块可将远程应用服务框架注册到本地，能对远程服务库查找和调用，实现不同应用服务框架之间的分布式调用。

管理模块提供对服务模块、组件和应用服务的管理功能，通过访问控制服务模块实现对应用服务的授权和访问控制，并提供UI界面实现人机交互。

监控模块在记录应用服务调用日志的基础上提供审计、统计和分析功能，可监控和改变应用服务的运行状态。

6 应用服务框架技术要求

应用服务框架作为应用支撑平台的基础架构，要求稳定、可靠，能够满足分布式、负载均衡、高可用的要求，能够动态部署和更新服务模块，能够实现对服务的实时管理。

6.1 服务库

服务库在应用服务框架中是服务模块、服务组件和应用服务信息的存储仓库,并提供管理服务接口，包括注册服务、修改服务、删除服务、改变服务状态等接口。为使服务能够被发现、被使用，还提供服务信息的查询、获取接口。

服务库作为应用服务框架的核心部分之一，要求稳定、高效及支持集群。对应用服务信息的存储可以采用多种方式，包括目录服务、数据库、文件系统或其他方式。

6.1.1 注册服务

应用服务框架应提供注册服务的功能。它可注册两类服务，一类由应用服务框架发布的服务，这类应用服务在发布的同时，自动在服务库注册，并将服务的状态标记为活动状态。另一类是第三方发布的Web服务，可通过人工或自动的方式注册。人工方式可以通过服务库提供的UI界面进行注册，自动方式可以调用服务库提供的服务注册接口注册。

通过管理服务接口可对应用服务进行变更、删除、改变状态等操作。

6.1.2 查找服务

6.1.2.1 人工查找服务

服务库提供UI界面，通过此界面可对服务进行查找，可查看服务的元数据属性，了解服务的运行状态及服务质量（QoS）等信息。

6.1.2.2 自动查找服务

服务的调用方可通过筛选条件进行服务查询。

6.2 发布模块

应用服务框架提供发布服务的功能，可将封装良好的功能单元发布为应用服务。

服务提供方在描述文件中对服务模块和组件的实现进行描述和配置，应用服务框架读取此配置文件，将其中服务组件的具体实现发布为应用服务，并自动将其注册到服务库。

本部分不规定服务模块描述文件的位置，而是通过META-INF/MANIFEST.MF文件中的Asf-Definition属性确定服务模块描述文件的位置。

应用服务框架应支持服务模块的动态部署和更新，不必重新启动系统。

6.3 调用模块

通过查找服务确定需要调用的服务后，可通过应用服务接口获得服务对象，此接口返回服务的代理对象，只有在实际执行调用时，才会根据应用服务的位置，发起local调用或remote调用。如果应用服务在远程，发起remote调用；如果应用服务在本地，发起local调用。

执行local调用时，根据不同的开发语言，对象是按照传引用方式（传地址）或传值方式引用；执行remote调用时，对象是按照传值方式引用。

关于调用服务的技术要求：

1） 要求支持Lazy调用，首先获得的是服务的代理对象，只有在实际执行调用时，才真正发起调用。

2） 要求透明的支持local调用和remote调用，在客户端代码及配置不做改动的情况下，可以动态调整应用服务提供方的位置。

6.4 管理模块

6.4.1.1 应用服务管理功能

应用服务管理功能包括应用服务的注册、删除、修改、改变服务状态等功能，并提供对服务模块、服务组件的管理功能。

6.4.1.2 应用服务列表

应用服务框架应提供应用服务列表，其中的信息包括服务的名称、所属的服务库、服务的状态、服务的处理日志和服务的性能记录等。

6.4.1.3 应用服务认证、访问控制

应用服务框架本身不提供访问控制的功能，可借助访问控制服务模块实现对应用服务的认证、授权和访问控制，如只允许经过认证的服务请求者才可访问指定的应用服务。

6.5 监控模块

6.5.1.1 监控

应用服务框架可以监控服务模块、组件和应用服务的运行状态，并可改变应用服务状态。

6.5.1.2 日志

应用服务框架详细记录每次服务调用的日志信息，包括调用的发起者、调用的服务、发起时间、响应时间、成功/失败等信息。

6.5.1.3 审计

应用服务框架提供审计功能，对服务的调用情况进行审计。

6.5.1.4 统计分析

应用服务框架提供对应用服务的运行状况的统计、分析功能，包括应用服务的运行状态、调用次数、平均响应时间、成功 / 失败次数等信息。

7 接口定义

本接口所处的命名空间为：egov.appservice.asf。

7.1 服务组件管理接口

7.1.1 注册服务组件接口

服务名称	ServiceComponentManager.register	
服务说明	注册服务，用于注册服务。	
参数列表	参数名称	参数说明
	moduleID	String 类型，是应用服务所属服务模块的唯一标识，对应于服务模块的 id 元数据，组件注册在此服务模块下。
	serviceComponent	ServiceComponent 对象，表示一个应用服务的元数据信息，具体内容参见“4.2 应用服务及服务组件元数据”。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	ServiceComponentManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceComponent 对象，表示本次注册生成的服务详细信息，包括为此服务新生成的唯一标识符 uid。	
备注		

7.1.2 修改服务组件接口

服务名称	ServiceComponentManager.update	
服务说明	修改服务组件，修改已有服务组件的属性。	
参数列表	参数名称	参数说明
	serviceComponent	ServiceComponent 对象，表示应用服务的信息，具体内容参见“4.2 应用服务及服务组件元数据”。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceComponentManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceComponent 对象，表示本次修改操作完成之后新的服务信息。	
备注		

7.1.3 删除服务组件接口

服务名称	ServiceComponentManager.delete	
服务说明	删除服务组件。	
参数列表	参数名称	参数说明
	Uid	指定服务组件的唯一标识符
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceComponentManagerException	如果删除失败，则抛出此异常。
返回值	无	
备注		

7.1.4 查找服务组件接口

服务名称	ServiceComponentManager.search

服务说明	查找服务组件。	
参数列表	参数名称	参数说明
	whereCase	String 类型，查询条件，不包含 where 字符串。用服务的元数据信息查找服务。查询条件的格式应符合 ANSI SQL 92 中 where 子句对查询条件的要求。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	ServiceComponentManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 数组，包含符合查询条件的服务组件的唯一标识符。	
备注		

7.1.5 获取服务组件接口

服务名称	ServiceComponentManager.get	
服务说明	获取服务组件。	
参数列表	参数名称	参数说明
	uid	String 类型，服务组件的唯一标识符
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceComponentManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceComponent 对象，包含服务的元数据信息。	
备注		

7.2 服务模块管理接口

7.2.1 创建服务模块接口

服务名称	ServiceModuleManager.create	
服务说明	创建服务模块。	
参数列表	参数名称	参数说明
	id	String 类型，服务模块的唯一标识
	name	String 类型，服务模块的中文名称
	version	String 类型，服务模块的版本
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务模块不存在，则抛出此异常。
	ServiceModuleManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceModule 对象，表示服务模块的信息，具体内容参见“4.3 服务模块元数据”。	
备注		

7.2.2 修改服务模块接口

服务名称	ServiceModuleManager.update	
服务说明	修改服务模块。	
参数列表	参数名称	参数说明
	serviceModule	ServiceModule 对象，表示服务模块的信息，具体内容参见“4.3 服务模块元数据”。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常

	NoSuchElementException	如果指定的服务模块不存在，则抛出此异常。
	ServiceModuleManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceModule 对象，表示本次修改操作完成之后新的服务模块信息。	
备注		

7.2.3 删除服务模块接口

服务名称	ServiceModuleManager.delete	
服务说明	删除服务模块。	
参数列表	参数名称	参数说明
	modelID	String 类型，服务模块的唯一标识
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务模块不存在，则抛出此异常。
	ServiceModuleManagerException	如果无法正常返回结果，则抛出此异常。
返回值	无	
备注		

7.2.4 查找服务模块接口

服务名称	ServiceModuleManager.search	
服务说明	查找服务模块。	
参数列表	参数名称	参数说明
	whereCase	String 类型，查询条件，不包含 where 字符串。查询条件的格式应符合 ANSI SQL 92 中 where 子句对查询条件的要求。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务模块不存在，则抛出此异常。
	ServiceModuleManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 数组,包含符合查询条件的服务模块的唯一标识	
备注		

7.2.5 获取服务模块接口

服务名称	ServiceModuleManager.get	
服务说明	获取服务模块。	
参数列表	参数名称	参数说明
	modelID	String 类型，服务模块的唯一标识
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务模块不存在，则抛出此异常。
	ServiceModuleManagerException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceModule 对象，表示服务模块的信息，具体内容参见“4.3 服务模块元数据”。	
备注		

7.3 应用服务管理接口

7.3.1 获取应用服务状态接口

服务名称	ServiceManager.getState	
服务说明	获取服务状态。	
参数列表	参数名称	参数说明
	uid	String 类型，服务的唯一标识符
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 类型，值为 running|stoped|unknown	
备注		

7.3.2 改变应用服务状态接口

服务名称	ServiceManager.changeState	
服务说明	改变服务状态。	
参数列表	参数名称	参数说明
	uid	String 类型，应用服务的唯一标识符
	state	String 类型，变更服务状态，可选取的值： running：运行状态 stoped：停止状态
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceManagerException	如果无法正常返回结果，则抛出此异常。
返回值	boolean 类型，成功为 true，失败为 false。	
备注		

7.3.3 查询应用服务接口

服务名称	ServiceManager.search	
服务说明	查找应用服务。	
参数列表	参数名称	参数说明
	whereCase	String 类型，查询条件，不包含 where 字符串。用服务的元数据信息查找服务。查询条件的格式应符合 ANSI SQL 92 中 where 子句对查询条件的要求。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	ServiceManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 数组，其中包含符合查询条件的服务的唯一标识符。	
备注		

7.3.4 应用服务日志接口

服务名称	ServiceManager.getLog	
服务说明	获取应用服务调用日志信息。	
参数列表	参数名称	参数说明
	uid	String 类型，指定应用服务的唯一标识符
	whereCase	String 类型，查询条件，不包含 where 字符串。用

		服务日志的信息查找日志。查询条件的格式应符合ANSI SQL 92 中 where 子句对查询条件的要求。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	ServiceManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 数组，指定应用服务的调用日志信息。	
备注		

7.4 服务库管理接口

7.4.1 注册远程服务库接口

服务名称	RepositoryManager.registerRepository	
服务说明	将远程的服务库注册在本地。	
参数列表	参数名称	参数说明
	name	String 类型，远程服务库的本地别名，不能重名。可使用此名称调用远程应用服务框架上发布的服务。
	ip	String 类型，远程服务库的 IP 地址
	port	int 类型，远程服务库的端口号
	description	String 类型，远程服务库的描述和说明信息
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务库不存在，则抛出此异常。
	RepositoryManagerException	如果无法正常返回结果，则抛出此异常。
返回值	无	
备注		

7.4.2 移除远程服务库接口

服务名称	RepositoryManager.removeRepository	
服务说明	从本地注册中移除远程服务库，不影响远程服务库。	
参数列表	参数名称	参数说明
	repositoryName	String 类型，远程服务库的本地别名
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务库不存在，则抛出此异常。
	RepositoryManagerException	如果无法正常返回结果，则抛出此异常。
返回值	无	
备注		

7.4.3 查询远程服务库接口

服务名称	RepositoryManager.searchRepository	
服务说明	查询远程服务库	
参数列表	参数名称	参数说明
	whereCase	String 类型，查询条件，不包含 where 字符串。查询条件的格式应符合 ANSI SQL 92 中 where 子句对查询条件的要求。

异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务库不存在，则抛出此异常。
	RepositoryManagerException	如果无法正常返回结果，则抛出此异常。
返回值	二维 String 数组，包含符合查询条件的远程服务库的唯一标识符和信息。	
备注		

7.4.4 查询远程应用服务接口

服务名称	RepositoryManager.search	
服务说明	查找注册在远程服务库中的应用服务信息。	
参数列表	参数名称	参数说明
	repositoryName	String 类型，远程服务库的本地别名。
	whereCase	String 类型，查询条件，不包含 where 字符串。用服务的元数据信息查找服务。查询条件的格式应符合 ANSI SQL 92 中 where 子句对查询条件的要求。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	RepositoryManagerException	如果无法正常返回结果，则抛出此异常。
返回值	String 数组，包含符合查询条件的服务的唯一标识符。	
备注		

7.5 获取应用服务接口

7.5.1 获得本地平台客户端接口

服务名称	ServiceClientFactory.getServiceClient	
服务说明	获得本地平台应用服务客户端。	
参数列表	参数名称	参数说明
	无	无
异常处理	ServiceClientException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceClient 对象，可用此对象获得具体的服务引用。	
备注	ServiceClientFactory 是定义在 egov.appservice.asf 下的类，不是接口，通过它可获得 ServiceClient 对象引用。	

7.5.2 获得远程平台客户端

服务名称	ServiceClientFactory.getServiceClient	
服务说明	获得远程平台应用服务客户端。	
参数列表	参数名称	参数说明
	repositoryName	String 类型，远程服务库的本地别名。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	ServiceClientException	如果无法正常返回结果，则抛出此异常。
返回值	ServiceClient 对象，可用此对象获得具体的服务引用。	
备注	ServiceClientFactory 是定义在 egov.appservice.asf 下的类，不是接口，通过它可获得 ServiceClient 对象引用。	

7.5.3 通过 UID 获取服务接口

服务名称	ServiceClient.getServiceByUID	
服务说明	通过 UID 获取服务对象。	
参数列表	参数名称	参数说明
	uid	String 类型，应用服务的唯一标识符
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceClientException	如果无法正常返回结果，则抛出此异常。
返回值	Service 代理对象，可向下造型为实际的服务类。	
备注		

7.5.4 通过 Name 获取服务接口

服务名称	ServiceClient.getServiceByName	
服务说明	通过服务的英文名称获取服务对象。	
参数列表	参数名称	参数说明
	name	String 类型，应用服务的英文名称（命名规则为所属服务组件的 id + “.” + 应用服务的 id,如 workflow.ProcessInstanceService）。
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务对象不存在，则抛出此异常。
	ServiceClientException	如果无法正常返回结果，则抛出此异常。
返回值	Service 代理对象，可向下造型为实际的服务类。	
备注		

7.5.5 通过 WSDL 获取服务接口

服务名称	ServiceClient.getServiceByWSDL	
服务说明	通过服务的 WSDL 地址获取服务对象。	
参数列表	参数名称	参数说明
	wsdl	String 类型，服务的 WSDL 地址
异常处理	IllegalArgumentException	如果传入的参数错误，则抛出此异常
	NoSuchElementException	如果指定的服务不存在，则抛出此异常。
	ServiceClientException	如果无法正常返回结果，则抛出此异常。
返回值	Service 代理对象，可向下造型为实际的服务类。	
备注		

7.6 异常约定

应用服务运行管理框架应包含以下异常：

异常名称	异常描述
ApplicationServiceFrameworkException	应用服务运行管理框架异常
ServiceComponentManagerException	服务组件管理异常
ServiceModuleManagerException	服务模块管理异常
ServiceManagerException	应用服务管理异常
RepositoryManagerException	服务库管理异常
ServiceClientException	应用服务调用异常

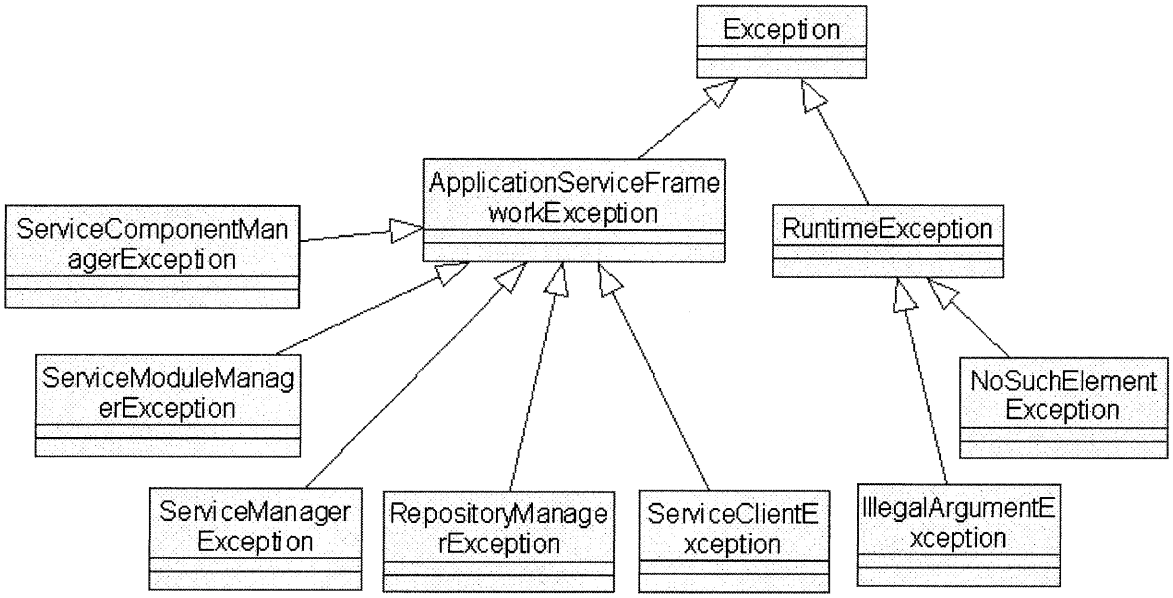


图3　应用服务框架异常关系示意图

附 录 A
（规范性附录）
服务组件描述模式 Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:element name="ServiceModule">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="info" minOccurs="0"/>
                <xs:element ref="ServiceComponent" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" use="required"/>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="version" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="info">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="provider" minOccurs="0"/>
                <xs:element ref="vender" minOccurs="0"/>
                <xs:element ref="dependence" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="provider">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="business" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="vender">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="business" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="business">
        <xs:complexType>
```

```
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="website" type="xs:string" minOccurs="0"/>
                <xs:element name="address" type="xs:string" minOccurs="0"/>
                <xs:element name="contact" type="xs:string" minOccurs="0"/>
                <xs:element name="tel" type="xs:string" minOccurs="0"/>
                <xs:element name="email" type="xs:string" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="id"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="dependence">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="module" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="module">
        <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="required"/>
            <xs:attribute name="version" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ServiceComponent">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="description" minOccurs="0"/>
                <xs:element ref="category" minOccurs="0"/>
                <xs:element name="wsdl" type="xs:string" minOccurs="0"/>
                <xs:element name="service" minOccurs="0">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                                <xs:attribute name="interface" use="required"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="implementation" minOccurs="0">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                                <xs:attribute name="class" use="required"/>
```

```
                                    <xs:attribute name="factory-method"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element ref="invoke" minOccurs="0"/>
                    <xs:element name="signature" type="xs:string" minOccurs="0"/>
                    <xs:element ref="properties" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:string" use="required"/>
                <xs:attribute name="name" type="xs:string" use="required"/>
                <xs:attribute name="version" type="xs:string" use="optional"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="category" type="xs:string"/>
        <xs:element name="invoke">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="style" minOccurs="0"/>
                    <xs:element ref="transport" minOccurs="0"/>
                    <xs:element ref="mode" minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="style">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="rpc"/>
                    <xs:enumeration value="document"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="transport">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="HTTP"/>
                    <xs:enumeration value="JMS"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="mode">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="sync"/>
```

```
                <xs:enumeration value="async"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="properties">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="property">
        <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="value" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

附 录 B
（资料性附录）
服务组件描述样例

```
<?xml version="1.0" encoding="GBK"?>
<ServiceModule id="workflow" name="流程服务模块" version="1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ServiceModule.xsd">
    <info>
        <provider>
            <business>
                <name>深圳福田区信息办</name>
                <website>http://www.szft.gov.cn</website>
                <address>深圳市福田新区委大楼1406</address>
                <contact>张三</contact>
                <tel>12345678</tel>
                <email>zhangsan@szft.gov.cn</email>
            </business>
        </provider>
        <vender>
            <business id="soft ">
                <name>某软件公司</name>
                <website>http://www.software-corp-website.com</website>
            </business>
        </vender>
        <dependence>
            <module id="org" version="1.0"/>
            <module id="ac" version="1.0"/>
            <module id="form" version="1.0"/>
        </dependence>
    </info>
    <ServiceComponent id="ProcessInstanceService" name="流程实例服务组件"
version="1.0">
        <description>
            <![CDATA[
                流程实例服务，包括创建、删除流程实例等方法
            ]]>
        </description>
        <category>/instancce</category>
        <service interface="egov.appservice.workflow.ProcessInstanceService" />
        <implementation class="net.risesoft.workflow.ProcessInstanceServiceImpl"
factory-method="getInstance" />
        <invoke>
            <style>rpc</style>
```

```
            <transport>HTTP</transport>
            <mode>sync</mode>
        </invoke>
        <signature>Here is signature info</signature>
        <properties>
            <property name="firstKey" value="aStringValue"/>
            <property name="secondKey" value="anotherStringValue"/>
        </properties>
    </ServiceComponent>
    <ServiceComponent id="ProcessDefineService" name="流程定义服务组件"
version="1.0">
        <category>/define</category>
        <service interface="egov.appservice.workflow.ProcessDefineService" />
        <implementation class="net.risesoft.workflow.ProcessDefineServiceImpl" />
    </ServiceComponent>
</ServiceModule>
```

附 录 C
（资料性附录）
深圳市福田区某并联审批系统应用示例

政府网上并联审批系统融合了政府业务中的事务办理、过程查询、数据交换、结果通告等常用功能，本示例以深圳市福田区审批系统中某并联审批流程作为案例，结合代码样例，详细描述本规范在电子政务中的应用和实现。

1.并联审批业务分析

1.1 业务描述

统一受理

客户通过服务大厅申报或者通过统一门户登陆系统，提交需要审批的材料和相关信息，经过初审，检查是否符合受理条件、材料是否齐全，初审合格后提交信息到相关主办单位，并打印受理通知书。

抄告相关

主办单位在接到申请人或委托代理人填写的《XXXX审批申请书》后，把填有申请企业名称、地址、经营范围、审批项目、通讯方式及联系人等内容的《XXXX审批申请书》，抄告给相关审批部门。

并联审批

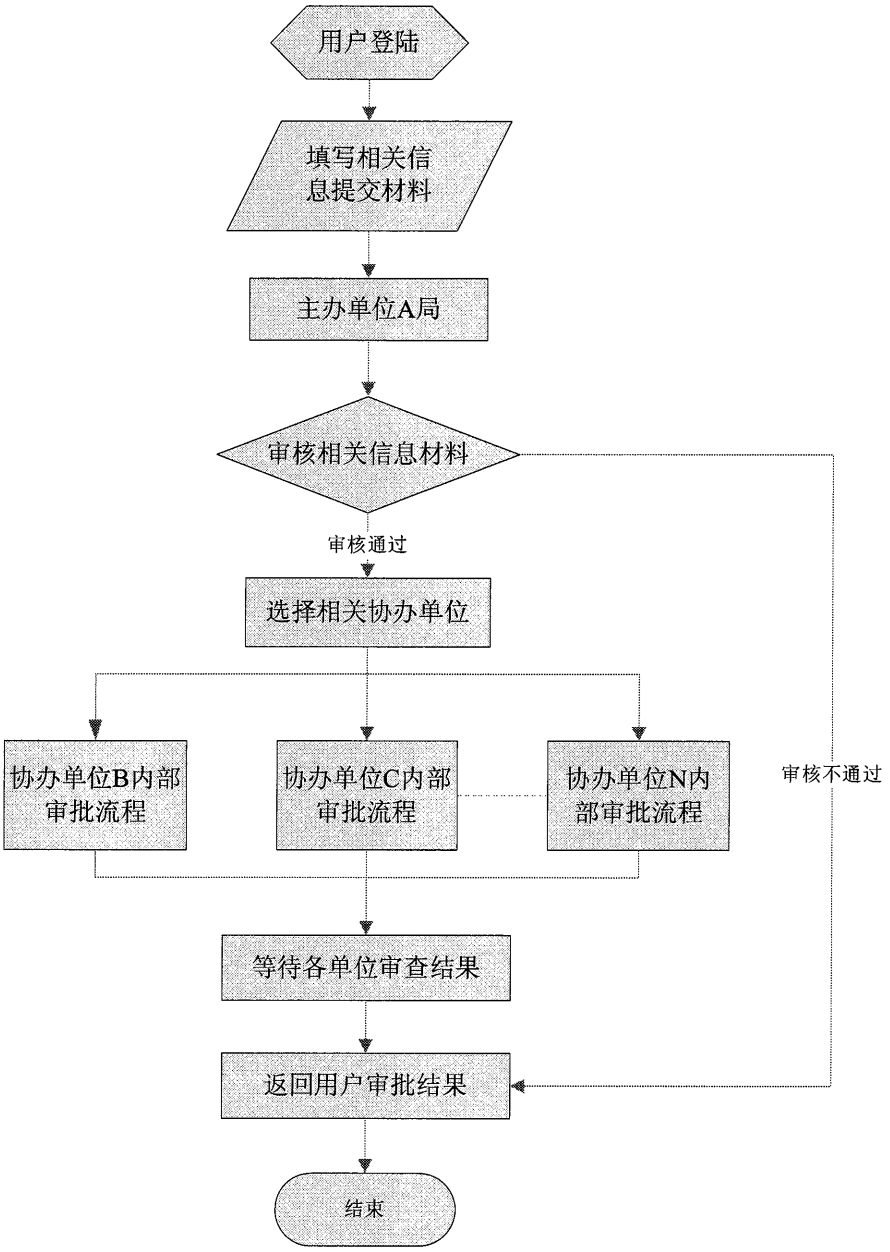
主办单位受理后，进入并联审批步骤，将审批信息并行发到协办单位，等待协办单位返回审批结果，协办单位完成审批后，主办单位汇总审批结果并告知客户，结束整个流程。

统一送达

各相关部门办理完成相关审批手续后，由主办单位统一将批准证书送达申办人。

1.2 场景描述

案例选择A局注册登记流程为示例进行描述。并联审批流程在办理过程中需要由一个主办单位发起，然后协调其他委办局在后续办理环节中同时开始审批。最后根据办理结果，由主办单位给予统一答复。场景示例如下：



用户登陆后，填写办理注册事项相关信息，提交所需要的审批材料，送交主办单位A局。

主办单位A局收到材料后，对用户所提交的材料和填写的基本信息进行核对，并进行初审，初审合格则正式受理。

如果主办单位正式受理，则根据需要将审核的材料同时抄送给B局、C局等协办单位。各协办单位接件后进入内部办理流程，开始进行并联审批。

各协办单位将办理结果反馈到主办单位。主办单位将办理结果进行汇总，并将最终审批结果返回给审批用户。

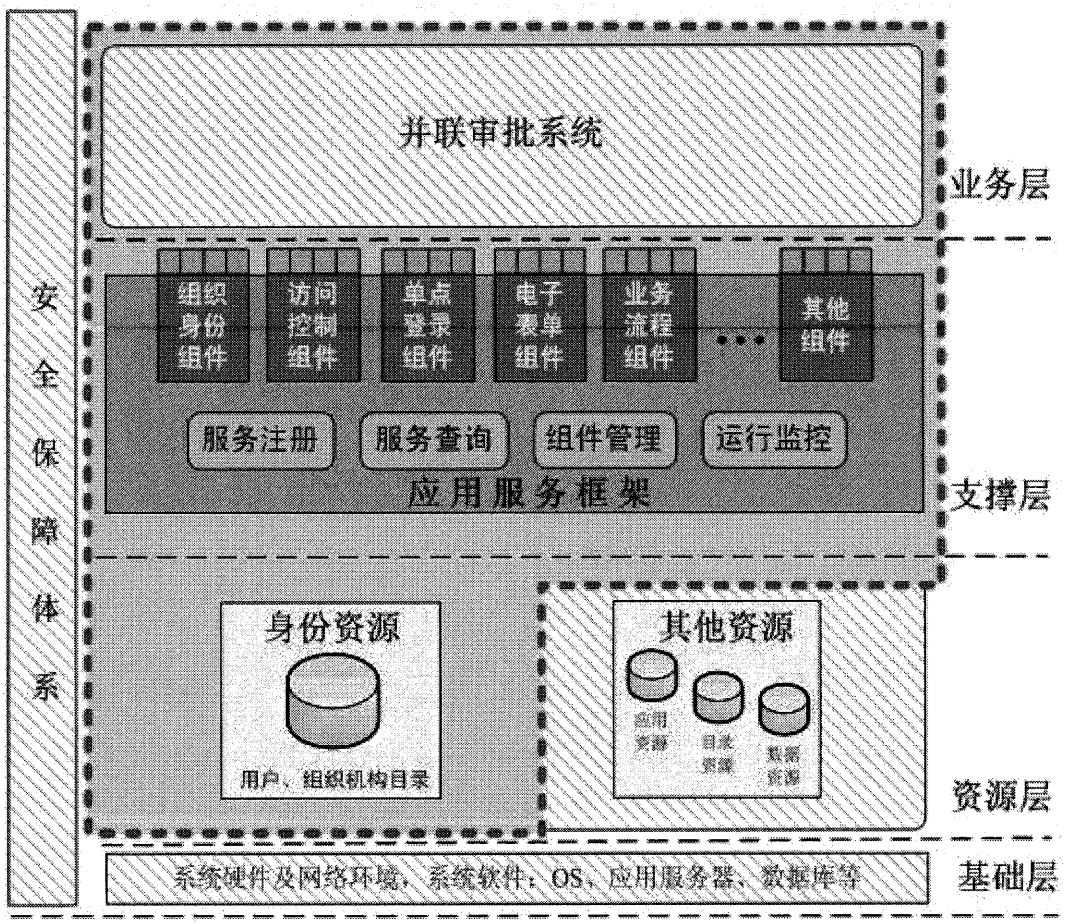
整个审批过程需要用到流程的新建、打开、发送等功能，同时需要用到辅助的用户、权限判断、表单保存等功能。

2.搭建支撑平台

2.1 平台结构框架

支撑平台分为基础层、资源层、支撑层、业务层四个部分。基础层为系统硬件和平台运行环境层，采用Linux操作系统和WebLogic应用服务，JDK版本1.5。资源层采用Oracle数据库，存储组织身份模型和业务数据。支撑层为并联审批系统应用支撑基础模块，包含应用服务框架和服务模块。业务层为并联审批应用程序。

逻辑结构图如下：



本案例介绍单一应用服务框架下的应用，所有服务模块运行于同一个支撑平台中，接口方式采用API。也可以采用分布式架构，将支撑平台和应用系统部署在不同的服务器上，其调用代码不变，由应用服务框架自动判定，通过Web服务进行通讯。

2.2 技术路线

采用J2EE技术路线，按面向对象的模块化方式设计，采用多层体系架构，通过基础组件API接口实现业务整合，集中式部署。

2.3 运行环境

JRE版本：J2SE5.0

数据库：oracle 10g

应用服务：WebLogic 9.2

操作系统：Redhat AS5

服务框架：ASF应用服务框架

2.4 基础支撑模块

2.4.1 应用服务框架

是支撑平台的底层框架，为各类基础应用服务模块提供运行管理功能。具体功能和技术规范参照本部分 《应用服务运行管理框架规范》。

2.4.2 组织身份服务

提供组织身份服务的应用模块。具体功能和技术规范参照本规范第6部分 《组织身份服务接口规范》，在场景描述中的人员登陆、联办部门选择等功能都由本服务的接口实现。

2.4.3 访问控制服务

提供访问控制服务的应用模块。具体功能和技术规范参照本规范第7部分 《访问控制服务接口规范》。对文件箱列表、menu菜单、表单等的访问控制权限判断等都需要本服务的接口来实现。

2.4.4 电子表单服务

提供电子表单服务的应用模块。具体功能和技术规范参照本规范第9部分 《电子表单服务接口规范》。表单打印、审批表单和表单数据操纵都需要本通过本服务的接口服务实现。

2.4.5 业务流程服务

提供业务流程服务的应用模块。具体功能和技术规范参照本规范第10部分 《业务流程服务接口规范》。所有的业务流转控制、业务流程数据交换都通过本接口实现。

2.5 应用服务模块注册及调用

2.5.1 应用服务模块注册

各服务模块需要注册到应用服务框架中才能被查找和调用，应用服务框架提供三种方式的服务注册方式：

- 调用应用服务框架的客户端（WEB页面）进行注册；
- 通过配置标准的注册文件进行自动注册；
- 通过应用服务框架提供的接口API进行注册

以下为注册文件示例：

```
<?xml version="1.0" encoding="GBK"?>
<ServiceModule id="form" name="表单服务模块" version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="ServiceModule.xsd">
    <info>
        <provider>
            <business id="a.gov">
                <name>A局信息办</name>
                <website>http://www.szft.gov.cn</website>
                <address>深圳市福田新区委大楼1406</address>
                <contact>张三</contact>
                <tel>12345678</tel>
                <email>zhangsan@163.com</email>
            </business>
        </provider>
        <vender>
            <business id="soft">
                <name>某软件公司</name>
                <website>http://www.software-corp-website.com</website>
            </business>
        </vender>
        <dependence>
```

```
            <module id="org" version="1.0"/>
            <module id="ac" version="1.0"/>
        </dependence>
    </info>
    <ServiceComponent id="FormTemplateService" name="表单模板服务组件" version="
1.0">
        <description>
        <![CDATA[对表单模板进行管理的组件，包括表单模板的部署、更新、删除等。]]>
        </description>
    <service interface="egov.appservice.form.FormTemplateService" />
    <implementation class="net.risesoft.soa.form.service.impl.FormTemplateServi
ceImpl" />
        <invoke>
            <style>rpc</style>
            <transport>HTTP</transport>
            <mode>sync</mode>
        </invoke>
    </ServiceComponent>
</ServiceModule>
```

2.5.2 应用服务调用

应用服务通过应用服务框架提供的接口统一进行调用，提供三种调用方式：

getServiceByName，通过服务名方式调用；

getServiceByUID，通过唯一标识符进行调用；

getServiceByWSDL，通过WSDL地址进行调用。

示例代码如下：

```
//从应用服务框架引用中获得服务客户端
ServiceClient sc = ServiceClientFactory.getServiceClient();

//通过byName方式调用应用服务管理接口
ServiceManager sm = sc.getServiceByName("asf.ServiceManager");
//sm对象实例就是服务的本地代理，获得sm对象后，就可以直接发起对应用服务的调用。

//查找所需要的服务UID
String[] orgs = sm.search("name like 'workflow%' and state = 'running'");
//通过byUID方式生成对象
ProcessInstanceService pis = sc.getServiceByUID(orgs[0]);
//pis对象实例就是服务的本地代理，获得pis对象后，就可以直接发起对应用服务的调用。
...
```

2.6 采用标准和规范

- SZDB/Z XXXX -2008 电子政务应用服务规范 第1部分：总则
- SZDB/Z XXXX -2008 电子政务应用服务规范 第4部分：组织身份模型数据规范
- SZDB/Z XXXX -2008 电子政务应用服务规范 第5部分：应用服务运行管理框架规范
- SZDB/Z XXXX -2008 电子政务应用服务规范 第6部分：组织身份服务接口规范
- SZDB/Z XXXX -2008 电子政务应用服务规范 第7部分：访问控制服务接口规范

- SZDB/Z XXXX -2008 电子政务应用服务规范 第9部分：电子表单服务接口规范
- SZDB/Z XXXX -2008 电子政务应用服务规范 第10部分：业务流程服务接口规范
- GB/T9704-1999 国家行政机关公文格式
- GB/T19487-2004 电子政务业务流程设计方法通用规范
- GB/T19488.1-2004 电子政务数据元第1部分：设计和管理规范
- 电子政务标准化指南

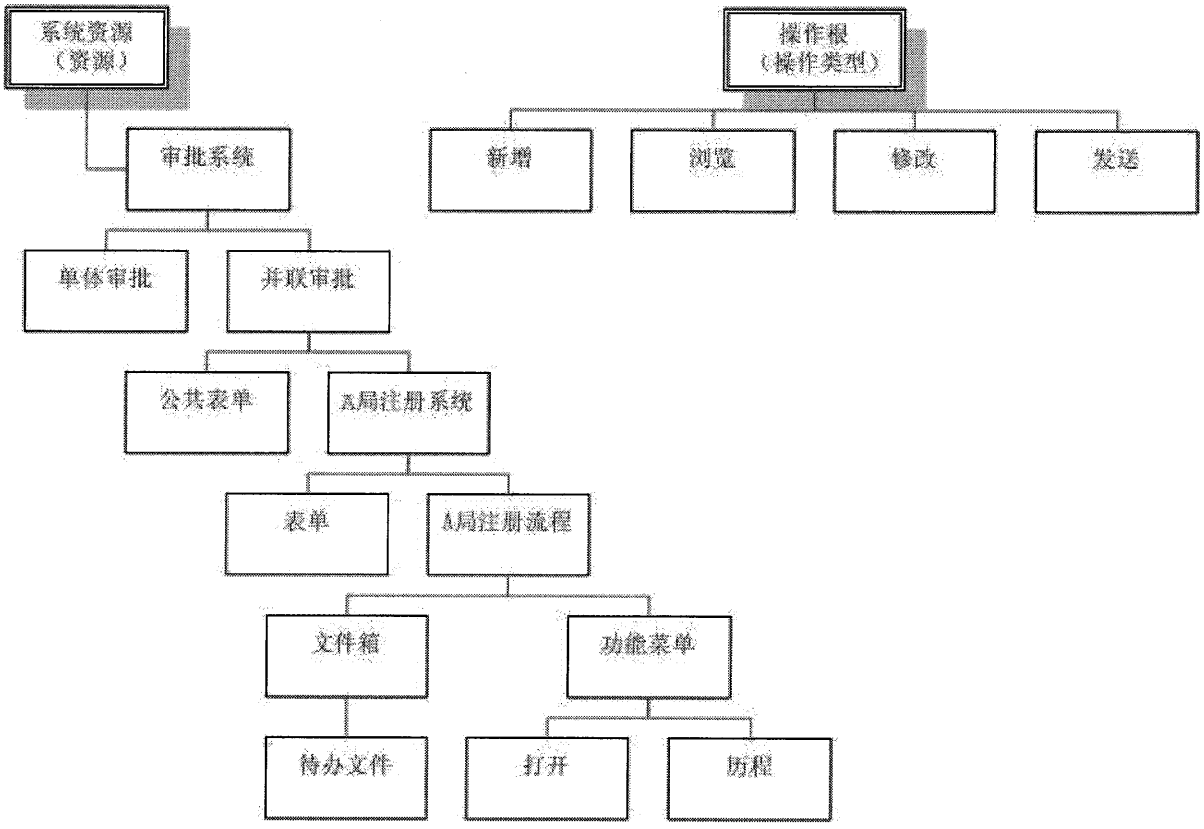
3.系统初始化设置

系统初始化设置是指构建支撑审批系统运行所需要的各种资源，如审批流程中需要用到的流程定义、审批表单、部门、人员、权限等。这些资源由客户端调用相应的服务组件接口进行设置。

3.1 资源注册

用到访问控制服务模块中的资源管理服务接口，在并联审批系统中的应用如下：

- 注册基础资源，生成资源目录树，包括整个审批系统结构
- 注册示例中需要的表单、流程定义、文件箱、功能菜单等
- 注册审批系统需要用到的操作类型，包括：新增、打开、发送、浏览等。



如上图所示，首先创建“系统资源”作为所有资源的根节点，然后在下面创建审批系统所需的单体审批和并联审批资源，并联审批下面的“A局注册系统”就是本示例要用到的各种资源。然后，创建“操作类型”为所有操作的根节点，在下面创建新增、浏览、修改、删除等操作。最后，将“系统资源”、“操作类型”和组织身份模型都加入域中，进行授权和访问控制。

3.1.1 创建资源根目录

用于系统初始化时，创建资源根节点。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();
//创建Resource管理服务接口对象
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");

//创建根资源对象，参数type为空指创建普通资源
Resource root = rm.createResource("", "系统资源",);
```

3.1.2 新增操作类型

增加新的操作者类型，用于授权。

```
//创建操作类型根节点
OperationManager om = sc.getServiceByName("ac.OperationManager");
Operation oRoot = om.createOperation("operationRoot", "操作类型",null);

//循环创建操作类型
operation = om.createOperation("add", "新增", oRoot.getUID());

operation = om.createOperation("browse", , "浏览" oRoot.getUID());

operation = om.createOperation("modify", "修改", oRoot.getUID());

operation = om.createOperation("send", "发送", oRoot.getUID());
```

3.1.3 创建审批系统资源

创建审批的基本资源，包括审批流程中所用到的流程定义、表单等相关资源。

```
//创建审批系统资源,包含单体审批和并联审批
Resource approvalSystem = rm.createResource("", "审批系统", root.getUID());
//创建单体审批流程资源
Resource singelApproval = rm.createResource("", "单体审批", approvalSystem.getU
ID());
//创建并联审批流程资源
Resource parallelApproval = rm.createResource("", "并联审批", approvalSystem.ge
tUID());

//创建并联审批公共表单父节点
Resource publicFormTemplete = rm.createResource("formResource", "公共表单", par
allelApproval.getUID());

//创建A局注册审批业务
Resource wfResource = rm.createResource("workflowResource", "A局注册系统", para
llelApproval.getUID());
//创建A局注册流程所需要的表单父节点
Resource formTemplete = rm.createResource("formResource", "表单", wfResource.g
etUID());
//创建A局注册流程流程定义节点
Resource workflow = rm.createResource("workflowResource", "A局注册流程", wfReso
```

```
urce.getUID());
```

3.1.4 创建文件箱资源

创建登陆用户文件夹列表，常用的包括待办、已办、在办等。

```
//创建文件箱
Resource deedboxResource = rm.createResource("deedboxResource", "文件箱", work
flow.getUID());
//创建文件夹，可以循环增加
Resource folderResource = rm.createResource("folderResource", "待办文件", deedb
oxResource.getUID());
folderResource.setProperties("uri", "/workList.jsp");
rm.updateResource(folderResource);
```

3.1.5 创建功能项菜单

创建功能项菜单，如对流程的操作功能（打开、发送等），通过授权可以控制菜单的显示。

```
//创建功能菜单menu，可以循环增加
Resource menuResource = rm.createResource("menuResource", "功能菜单", workflow.
getUID());

//创建menu子节点
Resource menuItemResource = rm.createResource("menuResource", "打开", menuReso
urce.getUID());
menuItemResource.setProperties("uri", "/openInstance.jsp");
rm.updateResource(menuItemResource);

menuItemResource = rm.createResource("menuResource", "历程", menuResource.getU
ID());
menuItemResource.setProperties("uri", "/instanceTracking.jsp");
rm.updateResource(menuItemResource);
```

3.1.6 生成域对象

添加相关对象（资源、操作者、操作类型）到域中，通过授权，用于限制相关用户的权限范围，以及可以修改的范围。

```
//创建Domain域对象
//生成管理范围,主要用于管理功能
DomainManager dm = sc.getServiceByName("ac.DomainManager");
Domain domain = dm.createDomain(null);
//增加资源到域中
dm.addObject(domain.getUID(), root.getUID(), "egov.appservice.ac.model.Resour
ce");
//增加操作到域中
operation = om.getOperation("operationRoot");
dm.addObject(domain.getUID(), operation.getUID(), "egov.appservice.ac.model.O
peration");
//增加人员到域中
OrganizationManager org = sc.getServiceByName("org.OrganizationManager");
List<Organization> organizations = org.getAllOrganizations();
```

```
dm.addObject(domain.getUID(), organizations.get(0).getUID(), "egov.appservic
e.org.model.Organization");
```

3.2 生成组织身份结构

应用到组织身份服务模块中组织身份管理服务接口，生成并联审批中需要的组织身份模型，包括部门、人员、岗位、组等，如主办单位A局、协办单位B局、C局，以及各委办局的相关审批人员等。

本示例所用到的人员部门信息由系统组织身份模型数据统一提供。

3.3 表单模板的注册

在审批系统中，不同的任务阶段需要不同的表单模板。表单模板注册就是要创建应用表单模板并注册到表单服务中。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//创建Resource管理服务接口对象
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");
//查找表单资源，返回全部
String[] resourceUIDs = rm.searchResource("name = '并联审批表单'",0,0);

//得到表单模板服务接口对象
FormTemplateService fts = sc.getServiceByName("form.FormTemplateService");

//部署指定的formTemplate表单模板对象
String xml = "<xml>...</xml>";
formTemplate = fts.importXML(xml);

FormTemplate ft = fts.deploy(formTemplate, resourceUIDs[0]);
```

3.4 业务流程定制

采用的流程定义工具设计业务流程模型，形成流程定义对象ProcessDefinition。可根据系统提供的图形界面生成或者导入XML格式的流程定义对象。流程定义模型中还应该包含各个节点的数据、相关人员对象、数据模型等，并和表单模板相关联。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//创建Resource管理服务接口对象
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");
String[] resourceUIDs = rm.searchResource("name = 'A局注册流程'",0,0);

//得到流程管理服务接口
ManagementService ms = sc.getServiceByName("workflow.ManagementService");
//得到流程定义对象
processDefinition = ms.importProcessDefinitionXML("<xml>...</xml>");

//得到流程定义服务接口
ProcessDefinitionService pds = sc.getServiceByName("workflow.ProcessDefinitio
nService");
```

```
//传入指定的processDefinition流程定义对象,进行部署
ProcessDefinition pd = pds.deployProcessDefinition(processDefinition, resourc
eUIDs[0]);
//改变流程定义状态为运行
pds.changeProcessDefinitionState(pd.getProcessDefinitionUID(), pd.getVersion
(), "Running");
```

3.5 权限设置

使用访问控制服务模块中授权服务接口，对已经注册的资源和组织进行权限关联。具体应用如下：

- 创建流程的访问权限
- 创建功能菜单、文件箱的访问权限等

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

AccessGrantService ags = sc.getServiceByName("ac.AccessGrantService");

OrgUnitManager oum = sc.getServiceByName("org.OrgUnitManager");
List<Object> depts = oum.search("Department", "name='A局办公室'");
Department dept = (Department)depts.get(0);

//创建Resource管理服务接口对象
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");

//对表单进行授权,可继承
String[] resources = rm.searchResource("name = '表单'",0,0);
ags.grantPermission(dept.getUID(), resources[0], "browse", "true");

//对流程资源进行授权,可继承
resources = rm.searchResource("name = 'A局注册流程'",0,0);
ags.grantPermission(dept.getUID(), resources[0], "browse", "true");
ags.grantPermission(dept.getUID(), resources[0], "modify", "true");
ags.grantPermission(dept.getUID(), resources[0], "send", "true");
```

4.审批业务流程的实现

通过模拟并联审批业务场景的分步实现，示例服务接口在系统中的应用。

4.1 用户登陆

使用组织身份服务中的身份认证接口实现登陆身份验证，并生成相关的session信息。以普通办事人员张三进行登陆。

4.1.1 判断用户登陆信息

对用户登陆信息进行认证，并返回认证结果。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//创建用户身份认证服务接口
AuthenticateService  as = sc.getServiceByName("org.AuthenticateService");
```

```
//用户登陆认证,如果登陆成功,返回true,添加对象到session中
boolean bl = as.authenticate("zhangsan", "111111");
```

4.1.2 生成用户登陆信息

如果用户认证通过，则生成用户登陆信息。如果不通过，返回登陆页面。

```
if (bl){
    //查找登陆的用户,并生成用户信息
    OrgUnitManager oum = sc.getServiceByName("org.OrgUnitManager");
    List<Object> persons = oum.search("Person", "loginname='zhangsan'");
    person = (Person)persons.get(0);
}
```

4.2 新建审批实例，并发送到主办部门

用户创建一个新的审批实例，并发送到主办单位A局。在这个过程中，需要使用访问控制服务的权限判断接口、流程服务中的流程实例管理接口、表单服务中的表单文档服务接口、组织身份服务中的组织身份管理接口等。

4.2.1 判断用户是否具有新建流程权限

通过登陆用户信息，判断用户对审批系统是否具有新建流程实例的权限。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//获取资源管理服务接口
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");
String[] resources = rm.searchResource("name = '流程定义'",0,0);
//获取访问控制服务接口
AccessControlService acs = sc.getServiceByName("ac.AccessControlService");
//判断当前用户对审批流程是否有新建流程权限
//Person对象为登陆用户
boolean hasPermission = acs.hasPermission(person.getUID(), resources[0], "add
");
```

4.2.2 创建流程实例

如果用户具有创建流程实例权限，则创建新的流程实例，选择申报表单模板，启动流程。

```
if (hasPermission) {
    //获取流程实例服务接口
    ProcessInstanceService pis= sc.getServiceByName("workflow.ProcessInstance
Service");
    //创建一个新的流程实例对象
    instance = pis.createProcessInstance(processDefinition.getProcessDefiniti
onUID(), processDefinition.getVersion());
    //获取空白的材料申报表单文档,并展现
    FormTemplateService fts = sc.getServiceByName("form.FormTemplateService");
    FormTemplate[] formTemplates = fts.search(null, "name = '材料申报'");

    FormDocService fds = sc.getServiceByName("form.FormDocService");
    formDoc = fds.renderForm(formTemplates[0].getFormTemplateUID(), 1, formDat
a, options);
```

```
    //添加表单信息并保存
    //收到客户端数据,组装成FormData对象,提交数据
    fds.submitForm(formDoc.getFormDocGUID(), formData, options);

    //流程实例启动
    pis.start(instance.getProcessInstanceUID());
```

4.2.3 发送流程实例到主办单位

流程实例创建后，发送给主办单位。

```
    //获取活动实例UID值
    ActivityInstance[] activityInstances = pis.getActivityInstances(instance.g
etProcessInstanceUID(), new String[]{"open.running"});
    //获取后续活动列表
    String[] activityDefinitions = pis.getNextActivities(activityInstances[0].
getActivityInstanceUID());
    //添加发送部门和人员
    OrgUnitManager oum = sc.getServiceByName("org.OrgUnitManager");
    List<Object> depts = oum.search("Department", "name = 'A局'");
    Department dept = (Department)depts.get(0);
    String[] actorUIDs = new String[]{dept.getUID()};

    for (int i = 0; i < activityDefinitions.length; i++) {
    pis.assignParticipant(activityInstances[0].getActivityInstanceUID(), acti
vityDefinitions[i], actorUIDs);
    }
    //运行发送活动实例
    pis.run(activityInstances[0].getActivityInstanceUID());
}
```

4.3 并联审批

审批材料发送到主办单位A局后，主办单位将审批件发送到协办单位：B局、C局。各协办单位进入内部审批流程，并返回审批结果到主办单位。整个过程中，主办单位处于等待状态中，而协办单位拥有自己的审批流程，互不干涉。分步实现方式如下：

4.3.1 待办列表（文件箱列表、待办文件列表的生成）

生成待办文件列表，包括生成文件箱列表，用户的待办列表等。

本示例中使用访问控制服务中的权限判断接口，控制用户文件箱列表的显示；使用访问控制服务中的资源管理服务接口，生成文件箱列表；使用流程服务中的流程实例服务接口，生成待办文件列表。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();
//调用资源管理服务接口，查找文件箱资源
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");
String[] deedboxResources = rm.searchResource("name = '文件箱'",0,0);
//得到当前的登陆用户
String actorUID = person.getUID();
//调用访问控制判断接口
AccessControlService acs = sc.getServiceByName("ac.AccessControlService");
```

```
//判断登陆用户对文件箱及其下属文件夹是否具有浏览权限
boolean bl = acs.hasPermission(actorUID, deedboxResources[0], "browse");
if (bl){
    //得到可访问的文件夹列表
    Resource deedboxResource = rm.getResource(deedboxResources[0]);
    String[] folderResources = rm.getSubResources(deedboxResource.getUID());
    for (int i = 0; i < folderResources.length; i++) {
        String folderName = rm.getResource(folderResources[i]).getName();
    }
}

//获取工作列表
//调用流程实例服务接口
ProcessInstanceService pis = sc.getServiceByName("workflow.ProcessInstanceSer
vice");
String[] processDefinitionUIDs = new String[]{processDefinition.getProcessDef
initionUID()};
String[] processInstanceStates = new String[]{"open.running"};
String[] actorUIDs = new String[]{actorUID};
//获取工作列表数组
WorkItem[] workItems = pis.getWorkList(processDefinitionUIDs, processInstance
States, actorUIDs, null, 10, 1);
for (int i = 0; i < workItems.length; i++) {
    String uid = workItems[0].getUID();
}
```

4.3.2 打开表单

使用表单服务中的表单文档管理接口，显示审批表单；使用访问控制服务的权限判断接口，控制功能菜单的显示。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();
ResourceManager rm = sc.getServiceByName("ac.ResourceManager");
String[] menuResources = rm.searchResource("name = '功能菜单'",0,0);
//得到当前的登陆用户
String actorUID = person.getUID();
//调用访问控制判断接口
AccessControlService acs = sc.getServiceByName("ac.AccessControlService");
//判断登陆用户对文件箱及其下属文件夹是否具有浏览权限
boolean bl = acs.hasPermission(actorUID, menuResources[0], "browse");

if (bl){
    //得到可访问的menu菜单列表,其中发送子菜单包含下一步动作选项
    Resource menuResource = rm.getResource(menuResources[0]);
    String[] menuItemrResources = rm.getSubResources(menuResource.getUID());
    for (int i = 0; i < menuItemrResources.length; i++) {
```

```
        String folderResourceName = rm.getResource(menuItemrResources[i]).getN
ame();
    }
}

//打开表单,生成显示页面
FormDocService fds = sc.getServiceByName("form.FormDocService");
//生成表单文档页面,最终显示由FormDoc对象决定
FormDoc fd = fds.renderForm(formDoc.getFormDocGUID(), null);
```

4.3.3 发送到并联审批部门

挂起主办单位A局的办理流程，新建B局和C局两个协办流程，实现多个部门的并行办理。使用流程服务的流程实例管理接口来改变流程状态、生成子流程；使用组织身份服务中的组织身份管理来为子流程指派参与者。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();
//获取流程服务接口实例对象
ProcessInstanceService pis= sc.getServiceByName("workflow.ProcessInstanceServ
ice");
//改变主流程状态为挂起

ActivityInstance[] activityInstances = pis.getActivityInstances(instance.getP
rocessInstanceUID(), new String[]{"open.running"});

pis.changeActivityInstanceState(activityInstances[0].getActivityInstanceUID
(), "open.notRunning.suspended");

//创建第一个子流程实例
ProcessInstance subInstance1 = pis.createSubProcessInstance(instance.getProce
ssInstanceUID(), processDefinition.getProcessDefinitionUID(), 1, null);
//子流程实例启动
pis.start(subInstance1.getProcessInstanceUID());
//获取活动实例UID值
activityInstances = pis.getActivityInstances(subInstance1.getProcessInstanceU
ID(), new String[]{"open.running"});
//添加子流程接收的部门和人员
OrgUnitManager oum = sc.getServiceByName("org.OrgUnitManager");
List<Object> depts = oum.search("Department", "name = 'B局'");
Department dept = (Department)depts.get(0);
String[] actorUIDs = new String[]{dept.getUID()};
//为流程指派参与者
pis.assignParticipant(activityInstances[0].getActivityInstanceUID(), startAct
ivityUID, actorUIDs);

//创建第二个子流程实例,创建方法如第一个
```

```
ProcessInstance subInstance2 = pis.createSubProcessInstance(instance.getProce
ssInstanceUID(), processDefinition.getProcessDefinitionUID(), 1, null);
pis.start(subInstance2.getProcessInstanceUID());
activityInstances = pis.getActivityInstances(subInstance2.getProcessInstanceU
ID(), new String[]{"open.running"});
oum = sc.getServiceByName("org.OrgUnitManager");
depts = oum.search("Department", "name = 'C局'");
dept = (Department)depts.get(0);
actorUIDs = new String[]{dept.getUID()};
pis.assignParticipant(activityInstances[0].getActivityInstanceUID(), startAct
ivityUID, actorUIDs);
```

4.3.4 审批意见汇总

协办单位完成审批后，结束自己的审批过程，并返回审批结果到主办单位A局。当所有协办单位办理完成后，由主办单位继续执行审批流程。使用流程服务中的流程实例服务接口，用于结束子流程、改变主流程的运行状态。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();
//获取流程实例服务接口对象
pis = sc.getServiceByName("workflow.ProcessInstanceService");
//子流程审批结束，返回主流程
pis.finish(instance.getProcessInstanceUID(), "closed.completed");
//TODO 需要二次开发，添加审批需要返回的结果

pis = sc.getServiceByName("workflow.ProcessInstanceService");
//循环判断所有子流程是否都已经结束
boolean bl = true;
ProcessInstance[] subProcessInstances = pis.getSubProcessInstances(instance.g
etProcessInstanceUID());
for (int i = 0; i < subProcessInstances.length; i++) {
    String state = pis.getProcessInstanceState(subProcessInstances[i].getProce
ssInstanceUID());
    if (!state.equals("closed.completed")){
        bl = false;
        break;
    }
}
//恢复主流程状态,由挂起到运行
if (bl){
    ActivityInstance[] activityInstances = pis.getActivityInstances(instance.g
etProcessInstanceUID(), null);
    pis.changeActivityInstanceState(activityInstances[0].getActivityInstanceU
ID(), "open.running");
}
```

4.4 办理结束

主办单位完成内部审批后，即可进入结束环节，结束整个审批流程。使用流程服务中的流程实例服务接口，结束审批流程。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//调用业务流程服务接口
ProcessInstanceService pis =
sc.getServiceByName("workflow.ProcessInstanceService");

//结束指定的流程实例
Pis.finish(instance.getProcessInstanceUID(), "closed.completed");
```

4.5 审批历程查看

使用流程服务中的流程管理服务接口，可以获取流程实例的办理历程。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//调用流程管理服务接口
ManagementService ms = sc.getServiceByName("workflow.ManagementService");
//获取指定流程实例的历程
String[][] trackings = ms.getProcessInstanceTracking(instance.getProcessInsta
nceUID());
for (int i = 0; i < trackings.length; i++) {
    //列表显示每一步
}
```

4.6 获取审批日志

使用流程服务中的流程管理服务接口，可以获取流程实例的操作日志。

```
//从应用服务框架引用中获得服务客户端
sc = ServiceClientFactory.getServiceClient();

//调用流程管理服务接口
ManagementService ms = sc.getServiceByName("workflow.ManagementService");
//获取指定流程实例的实例操作日志
String[][] logs = ms.getAuditLog(4, "processInstanceUID = '" + instance.getPro
cessInstanceUID() + "'");
for (int i = 0; i < logs.length; i++) {
    //列表显示日志信息内容
}
```

5.总结

综上所述，采用应用服务构建审批系统，从以前按业务开发的模式转变成根据业务流程进行模块拼装的模式。利用支撑平台提供的组织身份服务、访问控制服务、表单服务、业务流程服务和应用服务运行管理框架，可大大减轻系统实施的工作量。同时由于采用符合应用服务接口规范的模块，实现了各模块的可替换，减少对开发商的依赖，提高了系统的可维护性和可管理性。

参考文献

[1].《Enterprise SOA中文版——面向服务架构的最佳实战》 （美）清华大学出版社. 2006
[2].《SOA概念、技术与设计》（美）Thomas ERL 机械工业出版社.2007
[3].《J2EE平台Web Services》 (美) Lai,R. 电子工业出版社.
[4]. 信息技术 程序设计语言、环境与系统软件接口 独立于语言的数据类型(GB/T 18221-2000)
[5]. 现代设计工程集成技术的软件接口规范(GB/T 18726-2002)
[6]. 基于J2EE的高可用性Web集群的设计及实现 熊忠阳 计算机工程与设计,2006
[7]. Web Services Using Apache CXf, Jinsong Yang, Java Developer's Journal, 2007