(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101706901 B
(45)授权公告日 2012.07.25

(21)申请号 200910204570.1

(22)申请日 2009.12.01

(73)专利权人 中国建设银行股份有限公司
地址 100032 北京市西城区金融大街 25 号

(72)发明人 张峰 王斯洋 郑志远 陈铭新 刘立 李光宇 何银行 尹振宇 邱炜亨 潘舒

(74)专利代理机构 广州三环专利代理有限公司 44202
代理人 郝传鑫 潘中毅

(51)Int.Cl.
*G06F 9/54*(2006.01)
*G06F 13/14*(2006.01)

(56)对比文件
US 7065493 B1,2006.06.20, 全文.
CN 1466051 A,2004.01.07, 全文.
CN 1512370 A,2004.07.14, 全文.
US 6148323 A,2000.11.14, 全文.

审查员 陈丽娜

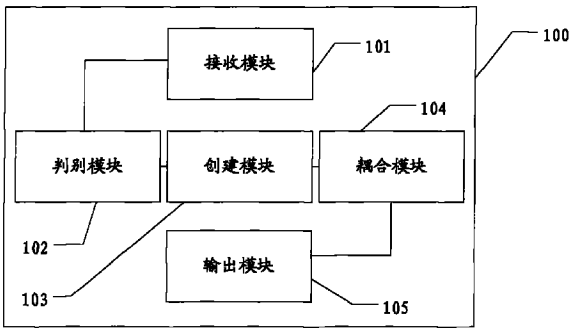
权利要求书 2 页 说明书 10 页 附图 4 页

(54)发明名称

实现屏蔽不同工作流产品的装置及方法

(57)摘要

本发明公开了一种实现屏蔽不同工作流产品的装置及方法，所述装置包括接收模块，用于接收应用系统服务器发送的应用上下文；判别模块，用于根据接收模块接收到的应用上下文判别调用方式是本地调用还是远程调用；创建模块，用于创建统一接口本地实现模块和统一接口远程实现模块；耦合模块，用于将统一接口本地实现模块或统一接口远程实现模块与预设的工作流统一接口耦合，由统一接口本地实现模块或统一接口远程实现模块实现工作流统一接口的功能；输出模块，用于向用户输出工作流统一接口。根据本发明的一种实现屏蔽不同工作流产品的装置及方法实现了屏蔽不同工作流产品的目的，使得工作流产品和应用系统之间形成松耦合。




CN 101706901 B

1. 一种实现屏蔽不同工作流产品的装置，其特征在于，所述装置包括：

接收模块，用于接收应用系统服务器发送的应用上下文；

判别模块，用于根据所述接收模块接收到的应用上下文判别调用方式是本地调用还是远程调用；

创建模块，用于根据所述判别模块的判别结果执行如下操作：当所述判别模块判别出调用方式为本地调用时，创建统一接口本地实现模块；当所述判别模块判别出调用方式为远程调用时，创建统一接口远程实现模块；

耦合模块，用于将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能；

缓存模块，用于将通过所述耦合模块与所述工作流统一接口耦合的统一接口本地实现模块和/或统一接口远程实现模块缓存于内存；

输出模块，用于向用户输出所述工作流统一接口。

2. 如权利要求 1 所述的实现屏蔽不同工作流产品的装置，其特征在于，所述装置还包括：

初始化模块，用于对所述装置进行初始化，核检所述接收模块接收到的应用上下文，并且传送所述应用上下文到所述判别模块。

3. 如权利要求 1 或 2 所述的实现屏蔽不同工作流产品的装置，其特征在于，所述应用上下文包括：调用方式、工作流服务器 IP 地址和端口、用户名和密码。

4. 如权利要求 3 所述的实现屏蔽不同工作流产品的装置，其特征在于，当所述判别模块判别出调用方式为本地调用时，所述创建模块还用于传送用户名和密码到所述统一接口本地实现模块，所述统一接口本地实现模块将所述用户名和密码翻译成本地工作流产品能识别的信息，并且传送所述信息到所述本地工作流产品，从而调用所述本地工作流产品。

5. 如权利要求 3 所述的实现屏蔽不同工作流产品的装置，其特征在于，当所述判别模块判别出调用方式为远程调用时，所述创建模块还用于传送工作流服务器 IP 地址和端口、用户名和密码、应用程序编程接口方法名称、输入参数到所述统一接口远程实现模块，所述统一接口远程实现模块根据所述工作流服务器 IP 地址和端口查找远程工作流产品的适配器模块，并且传送所述用户名和密码、应用程序编程接口方法名称、输入参数到所述远程工作流产品的适配器模块，所述适配器模块将所述用户名和密码、应用程序编程接口方法名称、输入参数翻译成所述远程工作流产品能识别的信息，并传送所述信息到所述远程工作流产品，从而调用所述远程工作流产品。

6. 如权利要求 5 所述的实现屏蔽不同工作流产品的装置，其特征在于，所述统一接口远程实现模块通过 JAVA 远程调用、网络服务或超文本传输协议方式传送所述用户名和密码、应用程序编程接口方法名称、输入参数到所述远程工作流产品的适配器模块。

7. 一种实现屏蔽不同工作流产品的方法，其特征在于，所述方法包括如下步骤：

a，接收应用系统服务器发送的应用上下文；

b，根据所述应用上下文判别是本地调用还是远程调用；

c，当判别为本地调用时，创建统一接口本地实现模块；当判别为远程调用时，创建统一接口远程实现模块；

d，将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能；

e，向用户输出所述工作流统一接口；

其中，在步骤 d 和 e 之间，所述方法还包括：

将与所述工作流统一接口耦合的统一接口本地实现模块和 / 或统一接口远程实现模块缓存于内存。

8. 如权利要求 7 所述的实现屏蔽不同工作流产品的方法，其特征在于，所述应用上下文包括：调用方式、工作流服务器 IP 地址和端口、用户名和密码。

9. 如权利要求 8 所述的实现屏蔽不同工作流产品的方法，其特征在于，

在步骤 c 中，当判别为本地调用时，还传送所述用户名和密码到所述统一接口本地实现模块；

在步骤 d 中，由所述统一接口本地实现模块实现所述工作流统一接口的功能包括：所述统一接口本地实现模块将所述用户名和密码翻译成本地工作流产品能识别的信息，并且传送所述信息到所述本地工作流产品，从而调用所述本地工作流产品。

10. 如权利要求 8 所述的实现屏蔽不同工作流产品的方法，其特征在于，

在步骤 c 中，当判别为远程调用时，还传送所述工作流服务器 IP 地址和端口、用户名和密码、应用程序编程接口方法名称、输入参数到所述统一接口远程实现模块；

在步骤 d 中，由所述统一接口远程实现模块实现所述工作流统一接口的功能包括：所述统一接口远程实现模块根据所述工作流服务器 IP 地址和端口查找远程工作流产品的适配器模块，并且传送所述用户名和密码、应用程序编程接口方法名称、输入参数到所述远程工作流产品的适配器模块，所述适配器模块将所述用户名和密码、应用程序编程接口方法名称、输入参数翻译成所述远程工作流产品能识别的信息，并且传送所述信息到所述远程工作流产品，从而调用远程工作流产品。

11. 如权利要求 10 所述的实现屏蔽不同工作流产品的方法，其特征在于，所述统一接口远程实现模块通过 JAVA 远程调用、网络服务或超文本传输协议方式传送所述用户名和密码、应用程序编程接口方法名称、输入参数到所述远程工作流产品的适配器模块。

12. 如权利要求 7 所述的方法，其特征在于，在步骤 e 之后，所述方法还包括：

根据所述工作流统一接口输入所述应用上下文，判别调用方式为本地调用还是远程调用，当判别为本地调用时，运行存储的统一接口本地实现模块；当判别为远程调用时，运行存储的统一接口远程实现模块。

# 实现屏蔽不同工作流产品的装置及方法

## 技术领域

[0001] 本发明涉及工作流技术领域，具体而言，本发明涉及一种实现屏蔽不同工作流产品的装置及方法。

## 背景技术

[0002] 工作流技术是90年代以后计算机应用领域的新技术，工作流技术对于提高企业的信息化程度、运行效率以及竞争能力都有着重要的意义。随着银行业务的不断发展，越来越多的银行的应用系统开始采用工作流产品来满足实际业务的需求。然而，银行业界工作流产品推陈出新发展较快，每个厂商提供的工作流客户端程序会存在差异，而应用系统与工作流产品之间的联系是通过工作流产品自身提供的客户端程序来完成的，这样应用系统与不同工作流产品在交互上会存在差异。当应用系统与不同工作流产品进行协同工作时，应用系统需要根据不同的工作流产品的客户端程序进行不同的配置，这使应用系统与工作流产品之间形成紧耦合。当应用系统需要替换、更新工作流产品时，银行需要派大量的技术人员去学习、熟悉新的工作流产品的客户端程序，进而根据新的客户端程序更新应用系统配置，这在银行技术人员有限和众多应用系统需要建设的情况下，不仅会严重浪费人力资源，还会增加项目实施的时间和成本，甚至影响项目质量，很大程度上会影响银行的运行效率和竞争能力。

[0003] 综上，目前不同的工作流产品的客户端程序存在差异，没有统一的标准，致使应用系统与工作流产品之间形成紧耦合，不利于应用系统和工作流产品的维护与替换，也不利于对不同工作流产品客户端程序的集中管理，制约了银行业务的发展。

## 发明内容

[0004] 本发明的目的是提供一种实现屏蔽不同工作流产品的装置及方法，通过工作流统一接口实现应用系统与工作流产品之间松耦合。

[0005] 为实现上述目的，本发明提供了一种实现屏蔽不同工作流产品的装置，所述装置包括：

[0006] 接收模块，用于接收应用系统服务器发送的应用上下文；

[0007] 判别模块，用于根据所述接收模块接收到的应用上下文判别调用方式是本地调用还是远程调用；

[0008] 创建模块，用于根据所述判别模块的判别结果执行如下操作：当所述判别模块判别出调用方式为本地调用时，创建统一接口本地实现模块；当所述判别模块判别出调用方式为远程调用时，创建统一接口远程实现模块；

[0009] 耦合模块，用于将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能；

[0010] 缓存模块，用于将通过所述耦合模块与所述工作流统一接口耦合的统一接口本地

实现模块和 / 或统一接口远程实现模块缓存于内存 ;

[0011] 输出模块,用于向用户输出所述工作流统一接口。

[0012] 本发明还提供了一种实现屏蔽不同工作流产品的方法,所述方法包括如下步骤 :

[0013] a,接收应用系统服务器发送的应用上下文 ;

[0014] b,根据所述应用上下文判别是本地调用还是远程调用 ;

[0015] c,当判别为本地调用时,创建统一接口本地实现模块 ;当判别为远程调用时,创建统一接口远程实现模块 ;

[0016] d,将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合,由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能 ;

[0017] e,向用户输出所述工作流统一接口 ;

[0018] 其中,在步骤 d 和 e 之间,所述方法还包括 :

[0019] 将与所述工作流统一接口耦合的统一接口本地实现模块和 / 或统一接口远程实现模块缓存于内存。

[0020] 通过实施以上技术方案,本发明具有以下优点 :

[0021] 采用本发明可以形成工作流统一接口,应用系统通过工作流统一接口接入不同的工作流产品,实现屏蔽不同工作流产品与应用系统交互上的差异,使应用系统与工作流产品之间形成松耦合,便于应用系统和工作流产品的维护与替换 ;工作流统一接口还可实现对工作流产品客户端程序的集中管理,使技术人员可以忽略不同工作流产品客户端程序的差异,利用所掌握的工作流统一接口技术来实现对不同工作流产品的操作,从而有效地节约人力和时间成本,提高银行的运行效率和竞争能力,促进银行业务的发展。

## 附图说明

[0022] 图 1 是本发明的一种实现屏蔽不同工作流产品的装置的第一实施例的结构示意图 ;

[0023] 图 2 是本发明的一种实现屏蔽不同工作流产品的装置的第二实施例的结构示意图 ;

[0024] 图 3 是本发明的一种实现屏蔽不同工作流产品的装置的第三实施例的结构示意图 ;

[0025] 图 4 是本发明的一种实现屏蔽不同工作流产品的装置在本地调用工作流产品时的系统位置示意图 ;

[0026] 图 5 是本发明的一种实现屏蔽不同工作流产品的装置在远程调用工作流产品时的系统位置示意图 ;

[0027] 图 6 是本发明的一种实现屏蔽不同工作流产品的装置在远程调用工作流产品时的系统位置示意图 ;

[0028] 图 7 是本发明的一种实现屏蔽不同工作流产品的方法的实施例的流程图。

## 具体实施方式

[0029] 为使本发明的实施例的目的、技术方案和优点更加清楚,下面将结合附图对本发

明作进一步地详细描述。

[0030] 图 1 是本发明提供的一种实现屏蔽不同工作流产品的装置的第一实施例的结构示意图，所述实现屏蔽不同工作流产品的装置 100 包括：

[0031] 接收模块 101，用于接收应用系统服务器发送的应用上下文，所述应用上下文包括：调用方式、工作流服务器 IP 地址和端口、用户名和密码；

[0032] 判别模块 102，用于根据接收模块 101 接收到的应用上下文判别调用方式是本地调用还是远程调用；

[0033] 创建模块 103，用于根据判别模块 102 的判别结果执行如下操作：当判别模块 102 判别出调用方式为本地调用时，创建统一接口本地实现模块，并且传送用户名和密码到所述统一接口本地实现模块；当判别模块 102 判别出调用方式为远程调用时，创建统一接口远程实现模块，并且传送工作流服务器 IP 地址 和端口、用户名和密码、应用程序编程接口 (Application Programming Interface，API) 方法名称、输入参数到所述统一接口远程实现模块；

[0034] 耦合模块 104，用于将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能，其中，预设的工作流统一接口是根据不同工作流产品提供的 API 的接口信息生成的；

[0035] 输出模块 105，用于向用户输出所述工作流统一接口。

[0036] 具体而言，所述统一接口本地实现模块实现所述工作流统一接口的功能包括：所述统一接口本地实现模块将所述用户名和密码翻译成本地工作流产品能识别的信息，并且传送所述信息到所述本地工作流产品，从而调用所述本地工作流产品；

[0037] 所述统一接口远程实现模块实现所述工作流统一接口的功能包括：所述统一接口远程实现模块根据所述工作流服务器 IP 地址和端口查找远程工作流产品的适配器模块，并且传送所述用户名和密码、API 方法名称、输入参数到所述远程工作流产品的适配器模块，所述适配器模块将所述用户名和密码、API 方法名称、输入参数翻译成所述远程工作流产品能识别的信息，并传送所述信息到所述远程工作流产品，从而调用所述远程工作流产品；另外，优选地，在远程调用工作流产品时，所述统一接口远程实现模块通过 JAVA 远程调用、网络服务 (WebService) 或超文本传输协议 (hyper text transport protocol，Http) 方式传送所述用户名和密码、API 方法名称、输入参数到所述远程工作流产品的适配器模块；

[0038] 其中，在远程调用工作流产品过程中，具有将所述用户名和密码、API 方法名称、输入参数等应用参数数据对象进行序列化和反序列化的过程。因为计算机中的复杂应用参数数据对象不能通过网络直接传输，需要将所述应用参数数据对象转换成一种二进制的串才可以在网络上传输，因此需要先将所述用户名和密码、API 方法名称、输入参数进行序列化形成 XML 报文，以便所述统一接口远程实现模块通过 JAVA 远程调用、WebService 或 Http 方式传送所述用户名和密码、API 方法名称、输入参数到所述远程工作流产品的适配器模块，所述适配器模块将所述用户名和密码、API 方法名称、输入参数翻译成所述远程工作流产品能识别的信息的过程即是将 XML 报文进行反序列 化的过程。

[0039] 通过实施本实施例，可以实现屏蔽不同工作流产品的目的，技术人员在使用工作流客户端程序时只需采用工作流统一接口，而不必考虑接口后面的内容。当需要替换或维

护工作流产品时，技术人员可以忽略不同工作流产品的客户端程序的差异，通过统一接口技术来对工作流产品进行替换或维护，这样节省了对不同工作流产品分别掌握所花费的时间，可有效提高工作效率，降低银行运营成本。

[0040] 图 2 是本发明提供的一种实现屏蔽不同工作流产品的装置的第二实施例的结构示意图。在本实施例中，如图 2 所示，所述实现屏蔽不同工作流产品的装置 100 还包括缓存模块 106，用于将通过耦合模块 104 与所述工作流统一接口耦合的统一接口本地实现模块和 / 或统一接口远程实现模块缓存于内存。

[0041] 通过实施本实施例，可以把与所述工作流统一接口耦合的统一接口本地实现模块和 / 或统一接口远程实现模块缓存在内存中，供以后调用工作流产品时直接使用，方便快捷。

[0042] 图 3 是本发明提供的一种实现屏蔽不同工作流产品的装置的第三实施例的结构示意图。在本实施例中，如图 3 所示，所述实现屏蔽不同工作流产品的装置 100 进一步包括初始化模块 107，用于对装置 100 进行初始化，同时对接收模块 101 接收到的应用上下文做核检工作（例如 :输入参数不能为空、IP 地址是否合法、用户名密码的验证工作等），并且传送所述应用上下文到判别模块 102。

[0043] 通过实施本实施例，可以实现上述第一实施例的有益效果，另外，通过使用初始化模块 107 对装置 100 进行初始化，由初始化模块 107 控制工作流统一接口创建过程中涉及到的模块调用，可确保装置 100 每次执行调用工作流产品工作时都能恢复初始化设置，提高装置 100 的易用性，使开发人员不需了解创建过程而直接使用装置 100 即可。装置 100 根据每次调用工作流产品的不同和调用方式的不同创建准确的统一接口本地实现模块或统一接口远程实现模块，确保了装置 100 执行调用不同工作流产品的工作的准确性。

[0044] 为使本领域技术人员能够更好地理解本发明，下面结合不同工作流产品（如：BPS、WLI）提供的相关 API 对如何统一不同的工作流产品、实现工作流统一接口的功能进行实例说明。

[0045] 实例一 :对于工作流统一接口“根据标识查找工作项信息”的功能

[0046] 在工作流统一接口中对应的方法为 :

[0047] public WFWorkItem queryWorkItemDetail(String workItemID) ;

[0048] 而在两种工作流产品 API 中对应的方法分别为 :

[0049] 1.BPS6 中对应的方法 :public WFWorkItem queryWorkItemDetail(longworkItem ID) ;

[0050] 2.WLI 中对应的方法 :public TaskInfo getTaskInfo(StringtaskId) ;(com.bea. wli.worklist.api.WorklistManager 类）。

[0051] 其中，在输入参数方面，输入参数分别为 :String 型和 long 型，long 型的参数可以转成 String 型，而 String 型转 long 型存在限制，即只有在 String 型中的值是由数字组成的整型字符串时才能成功转换，因此，工作流统一接口中采用 String 型。

[0052] 在输出参数方面，TaskInfo 是 WLI 提供的实体信息，而 WFWorkItem 为 BPS6 提供的实体信息，由于 BPS6 产品使用范围更广泛，所以以 BPS6 的 API 为主，由于 TaskInfo 与 WFWorkItem 存在较大差异，通过在 WFWorkItem 中增加一个 TaskInfo 类型的属性来包含 TaskInfo，这样两种方法的输出参数都被新的 WFWorkItem 涵盖了，因此，工作流统一接口

中采用 WFWorkItem。

[0053] 综上，在统一输入、输出参数时的原则为：类型、属性相近的做交集或转换（例如前面例子中的 long 型和 Stirng 型的关系），差别较大的以属性的形式纳入进来，即做并集（例如 WFWorkItem 和 TaskInfo）。

[0054] 实例二：对于工作流统一接口“找出当前用户可执行或待处理的工作项”的功能

[0055] 在工作流统一接口中对应的方法为：

[0056] public List queryPersonWorkItems(String personID, String permission, Stringscope, PageCond pagecond)。

[0057] 而在两种工作流产品 API 中对应的方法分别为：

[0058] 1. BPS6 中对应的方法：public List queryUserWorkItems(String personID, String permission, String scope, PageCond pagecond)；

[0059] 2. WLI 对应的方法：public TaskInfo[]getTaskInfos(TaskSelectortaskSelector)。

[0060] 可以看出，BPS6 的 API 可以满足上述功能，而 WLI 的 API 无法满足上述功能，但通过下述方法可使 WLI 的 API 能够满足需求：

[0061] public List findToDoList(BriefUserVO user)throws Exception{

[0062] ProfilingUtil.startProfiling(this.getClass()+ " . findToDoList" )；

[0063] WorklistManager worklistMgr = fetchWliContext(). getWorklistManager()；

[0064] TaskSelector filter = new TaskSelector()；

[0065] StateType[]todoStateTypes = new StateType[]

[0066] {StateType.ASSIGNED, StateType.CLAIMED}；

[0067] filter.setStateTypes(todoStateTypes)；

[0068] Assignee assignee = new Assignee()；

[0069] //modi by yuan 0112 assignee.setUsers(new String[]{user. getLoginAccount()})；

[0070] assignee.setUsers(new St ring[]{user.getUserNum()})；

[0071] filter.setAssignee(assignee)；

[0072] filter.setSortByCreationDate((short)-1)；

[0073] filter.setSortByParentProcessId((short)-2)；

[0074] filter.setSortByTaskId((short)-3)；

[0075] //filter.setMaxTasksReturned(MAX_TASK_NUM_PER_PAGE)；

[0076] TaskInfo[]todoTasks = null；

[0077] try{

[0078] todoTasks = worklistMgr.getTaskInfos(filter)；

[0079] }catch(Exce ption exception){

[0080] StringBuffer logMsg = new StringBuffer(200)；

[0081] logMsg.append(" 查找操作员的待做任务时出现了异常！ " )；

[0082] logMsg.append(" \n errorCode = workflowdelegate-e-0000011us

```
erNum =" +   user.getUserNum()+" userName =" +user.getUserName()) ;
[0083]            logger.error(logMsg,exception) ;
[0084]          throw                          new          WLI Exception(new
[0085]      SysMessage(" workflowdelegate-e-0000011" ),exception) ;
[0086]            }
[0087]            if(todoTasks == null){
[0088]                 return Collections.EMPTY_LIST ;
[0089]      }
[0090]      Integer taskState = null ;
[0091]      ArrayList clpmTasklnfoList = new ArrayList() ;
[0092]      for(int i = 0 ;i < todoTasks.length ;i++){
[0093]  //       ClpmTasklnfo clpmTasklnfo = buildClpmTasklnfo(todoTasks[i]) ;
[0094]           ClpmTasklnfo clpmTasklnfo = new ClpmTasklnfo() ;
[0095]           clpmTasklnfo.setClaimant(todoTasks[i].getClaimant()) ;
[0096]           clpmTasklnfo.setStatus(todoTasks[i].getStateType().getValue()) ;
[0097]           clpmTasklnfo.setTaskld(todoTasks[i].getTaskld()) ;
[0098]          if(clpmTasklnfo ! = null){
[0099]               taskState = clpmTasklnfo.getStatus() ;
[0100]               if(taskState.equals(StateType.ASSIGNED.getValue())||
[0101]                    (taskState.equals(StateType.CLAIMED.getValue())&&
[0102]    (clpmTasklnfo.getClaimant().equalslgnoreCase(user.getUserNum())))){
[0103]                    clpmTasklnfoList.add(clpmTasklnfo) ;
[0104]               }
[0105]          }
[0106]      }
[0107]      ProfilingUtil.stopProfiling(this.getClass()+" .findToDoList" ) ;
[0108]      return clpmTasklnfoList ;
[0109]      }
```

[0110] 由于该方法的输入参数只有BriefUserVO类型，而在程序实现中只用到了UserNum属性，同时该属性为String型，因此，两个方法的输入很容易统一，即由工作流统一接口中的方法的String personID参数取代BriefUserVO中的UserNum属性，而在WLI的工作流接口实现中根据UserNum属性创建BriefUserVO，并且把BriefUserVO传入findToDoList方法，对于输入参数，通过在findToDoList方法中将TaskInfo的数组转换成List，统一输入参数。

[0111] 综上，当不同工作流产品API提供的输入参数个数不一样时，尽量不增加参数个数，如果参数类型在程序设计语言中无法转换则采用增加输入参数的方式解决，对于多出来的参数在程序实现中可不使用。

[0112] 实例三：对于工作流统一接口“挂起工作流实例”的功能

[0113] 在工作流统一接口中对应的方法为：

[0114] public void suspendProcessInstance(long processInstID)。

[0115] 而在两种工作流产品 API 中对应的方法分别为：

[0116] 1.BPS6 中对应的方法：public void suspendProcessInstance(longprocessInstID)；

[0117] 2.WLI 中无相应的方法，因为对 WLI 而言，日常开发使用 suspendTask（挂起某个任务）方法即可满足需要。由于这个方法是 BPS6 特有的，而 WLI 产品中无相应方法，所以 WLI 产品本身没有该功能，更不会提供 API。考虑到 BPS 产品适用范围广泛，因此采取直接融入该方法来解决此问题。对于 WLI 工作流产品来说，使用该方法时，WLI 工作流无任何效果。

[0118] 实例四：对于工作流统一接口“在工作流产品中创建一个业务流程”的功能

[0119] 在工作流统一接口中对应的方法为：

[0120] public String createAndStartProcInstAndSetRelativeData(String procDefName, String procInstName, String procInstDesc, boolean transcationSpan, Map map)

[0121] 而在两种工作流产品 API 中对应的方法分别为：

[0122] 1.BPS6 中对应的方法：

[0123] public long createAndStartProcInstAndSetRelativeData(String procName, StringpInstName, String procInstDesc, boolean transcationSpan, Map map)

[0124] 2. 对于此功能，WLI 没有单独的 API，但通过两个 API 的配合，可以完成此功能，因此，可在 WLI 的两个 API 的基础上结合代码编程新的方法：public StringcreateWorkflow (ClpmBPMetaInfo bpMetaInfo, CommonBPBO commonBPBO)，

[0125] 实现代码如下：

[0126] public String createWor kflow(ClpmBPMetaInfo bpMetaInfo, CommonBPBOcommonBPBO)throws Exception{

[0127] ProfilingUtil.startProfiling(this.getClass()+ ″ .createWorkflow″ )；

[0128] StartProcess bp = fetchWliContext().getBP(bpMetaInfo)；

[0129] if(bp == null){

[0130] StringBuffer logMsg = new StringBuffer(200)；

[0131] logMsg.append(″ 无法获取 JPD Proxy StartProcess，不能创建流程！″ )；

[0132] logMsg.append( ″ ￥n errorCode = workflowdelegate-e-0000005

[0133] processCode =″ +bpMetaInfo.getProcessCode()+″ ServiceURI =″ +

[0134] bpMetaInfo.getServiceURI())；

[0135] logger.error(logMsg)；

[0136] throw new WLIException(new

[0137] SysMessage(″ workflowdelegate-e-0000005″ ))；

[0138] }

[0139] try{

[0140] String taskId = bp.startProcess(commonBPBO)；

[0141] if(logger.isDebugEnabled()){

```
[0142]                    logger.debug(" new first task instance created：" +taskId)；
[0143]                    }
[0144]                  ProfilingUtil.stopProfiling(this.getClass()+" .createWorkflow" )；
[0145]                    return taskId；
[0146]                  }catch(Exception exception){
[0147]                     StringBuffer logMsg ＝new StringBuffer(200)；
[0148]                     logMsg.append(" 创建流程时出现了异常！" )；
[0149]                              logMsg.append(  "  ￥n errorCode ＝
workflowdelegate-e-0000006
[0150]            processCode ＝" +bpMetaInfo.getProcessCode()+" ServiceURI ＝" +
[0151]            bpMetaInfo.getServiceURI())；
[0152]                       logger.error(logMsg,exce ption)；
[0153]                     throw                        new       WLI Exception(new
[0154]   SysMessage(" workflowdelegate-e-0000006" ),exception)；
[0155]       }
[0156]     }
```

[0157] 由于 WLI 的输入参数需要 ClpmBPMetaInfo 和 CommonBPBO，而 BPS 的输入参数需要 String、String、String、boolean 和 Map，因此输入参数差异较大。但是 BPS 中有一个 Map 类型的参数，该参数可以直接将 ClpmBPMetaInfo 和 ClpmBPMetaInfo 容纳进去 (Map 是一种 java 程序设计语言自带的数据结构，使用它开发人员可以直接将不同类型的参数放进去 Map 中去)，具体代码实现如下：

```
[0158]   map.put(" bpMetaInfo" ,ClpmBPMetaInfo)；
[0159]   map.put(" commonBPBO" ,CommonBPBO)；
```

[0160] 通过下面的代码可以直接取出 ClpmBPMetaInfo 和 CommonBPBO：

```
[0161]   ClpmBPMetaInfo bpMetaInfo ＝ (ClpmBPMetaInfo)map.get(" bpMetaInfo" )；
[0162]   CommonBPBO ommonBPBO ＝ (CommonBPBO)map.get(" commonBPBO" )；
```

[0163] 返回值一个是 long 型，一个是 String 型，因此统一接口中的方法返回值定为 String 型，由此可完成两者的融合。

[0164] 综上所述，在设计屏蔽不同工作流产品的装置的过程中，首先对不同工作流产品提供的相关 API 方法进行梳理，然后根据功能相同的方法的输入、输出参数的属性采用相关技术手段完成输入、输出参数的统一，从而统一不同的工作流产品，实现工作流统一接口的功能，达到屏蔽不同工作流产品的目的。

[0165] 图 4 是本发明提供的一种实现屏蔽不同工作流产品的装置在本地调用工作流产品时的系统位置示意图，表示应用程序和工作流产品位于同一应用系统服务器上。本发明所述装置运行在所述应用程序上，用户可通过所述装置以程序调用方式操作工作流产品，例如：将用户名和密码翻译成工作流产品能够识别的信息，并通过程序调用相关的输入参数，将输入参数发送到工作流产品。

[0166] 根据现有技术可知，不同的工作流产品会面向客户提供一套自己的 API，帮助客户通过程序代码的方式使用工作流。目前大部分工作流产品的 API 都支持远程调用，当然也

有些工作流产品的 API 不支持远程调用。这时需要根 据所述工作流产品提供的 API 开发相应的适配器模块，并且把所述适配器模块配置于所述工作流产品服务器上。所述适配器模块的作用在于：当所述接口远程实现模块要传送所述用户名和密码、API 方法名称、输入参数等应用参数数据到工作流产品时，所述应用参数数据会先经过所述工作流产品的适配器模块，由所述适配器模块将接收到的所述应用参数数据翻译成所述远程工作流产品能识别的信息，并传送所述信息到工作流产品，从而调用工作流产品。

[0167] 图 5 和图 6 是本发明提供的一种实现屏蔽不同工作流产品的装置在远程调用工作流产品时的系统位置示意图，如图 5 所示，图 5 表示当工作流产品的 API 支持远程调用时，本发明所述装置通过 JAVA 远程调用、WebService 或 Http 方式与工作流服务器上的工作流产品通信，直接将用户名和密码、API 方法名称、输入参数等应用参数数据发送到工作流产品。如图 6 所示，图 6 表示当工作流产品的 API 不支持远程调用时，在工作流服务器上配置相应的适配器模块，用于以程序调用方式操作工作流产品，本发明所述装置通过 JAVA 远程调用、WebService 或 Http 方式与所述适配器模块通信，将用户名和密码、API 方法名称、输入参数等应用参数数据发送到适配器模块，通过所述适配器模块将用户名和密码、API 方法名称、输入参数等应用参数数据翻译成工作流产品能识别的信息以操作工作流产品。

[0168] 图 7 是本发明提供的一种实现屏蔽不同工作流产品的方法的实施例的流程图，所述方法包括如下步骤：

[0169] S101，接收应用系统服务器发送的应用上下文，所述应用上下文包括：调用方式、工作流服务器 IP 地址和端口、用户名和密码；

[0170] S102，根据所述应用上下文判别是本地调用还是远程调用；

[0171] S103，当判别为本地调用时，创建统一接口本地实现模块，并且传送所述用户名和密码到所述统一接口本地实现模块；当判别为远程调用时，创建统一接口远程实现模块，并且还传送所述工作流服务器 IP 地址和端口、用户名和密码、API 方法名称、输入参数到所述统一接口远程实现模块；

[0172] S104，将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能；

[0173] 其中，所述预设的工作流统一接口是根据不同工作流产品提供的 API 的接口信息生成的，具体例子请参见上述的实例一至实例四，这里不再赘述。

[0174] S105，向用户输出所述工作流统一接口，用户可以通过所述工作流统一接口对不同工作流产品进行操作，调用工作流产品。

[0175] 在本发明实施例中，由所述统一接口本地实现模块实现所述工作流统一接口的功能包括：所述统一接口本地实现模块将所述用户名和密码翻译成本地工作流产品能识别的信息，并且传送所述信息到所述本地工作流产品，从而调用所述本地工作流产品；

[0176] 由所述统一接口远程实现模块实现所述工作流统一接口的功能包括：所述统一接口远程实现模块根据所述工作流服务器 IP 地址和端口查找远程工作流产品的适配器模块，并且传送所述用户名和密码、API 方法名称、输入参数到所述远程工作流产品的适配器模块，所述适配器模块将所述用户名和密码、API 方法名称、输入参数翻译成所述远程工作流产品能识别的信息，并且传送所述信息到所述远程工作流产品，从而调用远程工作流产

品（具体请参见上面装置实施例中的描述，这里不再赘述）；

[0177] 另外，优选地，当远程调用工作流产品时，所述统一接口远程实现模块通过 JAVA 远程调用、WebService 或 Http 方式传送所述用户名和密码、API 方法名称、输入参数到所述远程工作流产品的适配器模块。

[0178] 通过实施本实施例，利用工作流统一接口，屏蔽不同工作流产品与应用系统交互上的差异，实现了对工作流产品客户端程序的集中管理，使应用系统与工作流产品之间形成松耦合，这样便于工作流产品和应用系统的维护与替换，可节省对不同工作流产品分别掌握所花费的时间，可有效提高工作效率，降低银行运营成本。

[0179] 在本发明的另一实施例中，在步骤 S104 和 S105 之间，所述方法还包括：将与所述工作流统一接口耦合的统一接口本地实现模块和 / 或统一接口远程实现模块缓存于内存。可选地，在步骤 S105 之后，所述方法还包括：根据所述工作流统一接口输入所述应用上下文，判别调用方式为本地调用还是远程调用，当判别为本地调用时，运行缓存的统一接口本地实现模块；当判别为远程调用时，运行缓存的统一接口远程实现模块。

[0180] 通过实施本实施例，可以把与所述工作流统一接口耦合的所述统一接口 本地实现模块和 / 或所述统一接口远程实现模块缓存在内存中，在调用同样的工作流产品时，不需要重新创建所述统一接口本地实现模块或所述统一接口远程实现模块，只需在调用工作流产品时根据对本地调用或远程调用的判定直接从内存中运行相应的统一接口本地实现模块或统一接口远程实现模块即可，这样可有效提高工作效率，有利于银行业务的发展。

[0181] 通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到本发明可借助软件的方式来实现，当然也可以通过软件结合硬件平台的方式来实现。基于这样的理解，本发明的技术方案对背景技术做出贡献的全部或者部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在存储介质中，如 ROM/RAM、磁碟、光盘等，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）执行本发明各个实施例或者实施例的某些部分所述的方法。

[0182] 以上所公开的仅为本发明的具体实施方式，仅用于对本发明进行举例说明，不能以此限定本发明之保护范围，本领域技术人员在不脱离本发明实质的前提下可以进行各种修改、变化或替换，因此，依照本发明所作的各种等同变化，仍属于本发明所涵盖的范围。

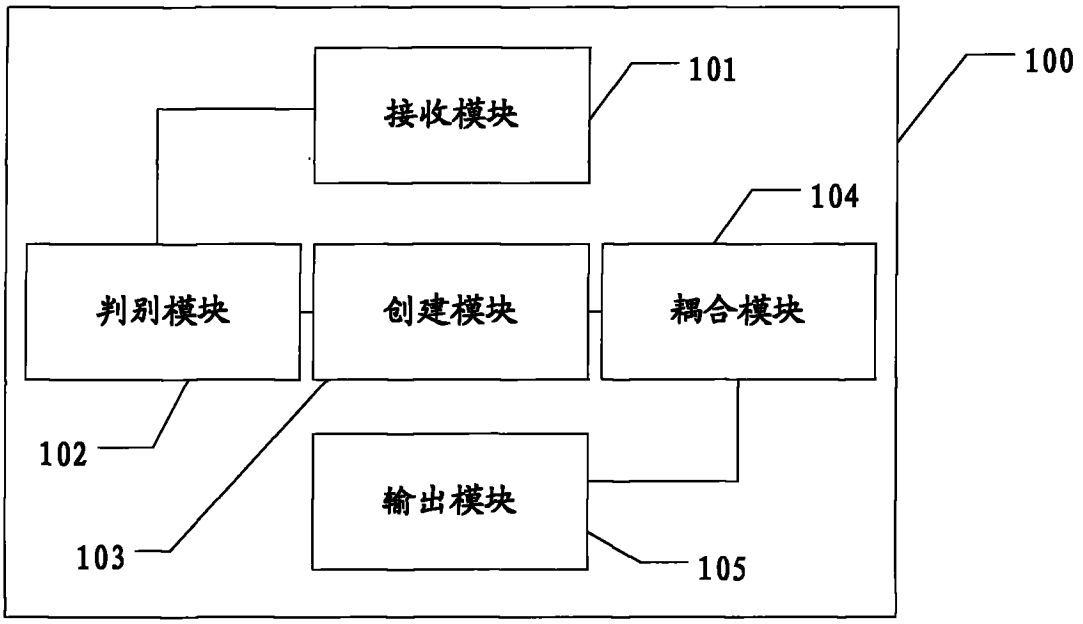

接收模块
101
100
104
判别模块
创建模块
耦合模块
102
输出模块
103
105

图 1

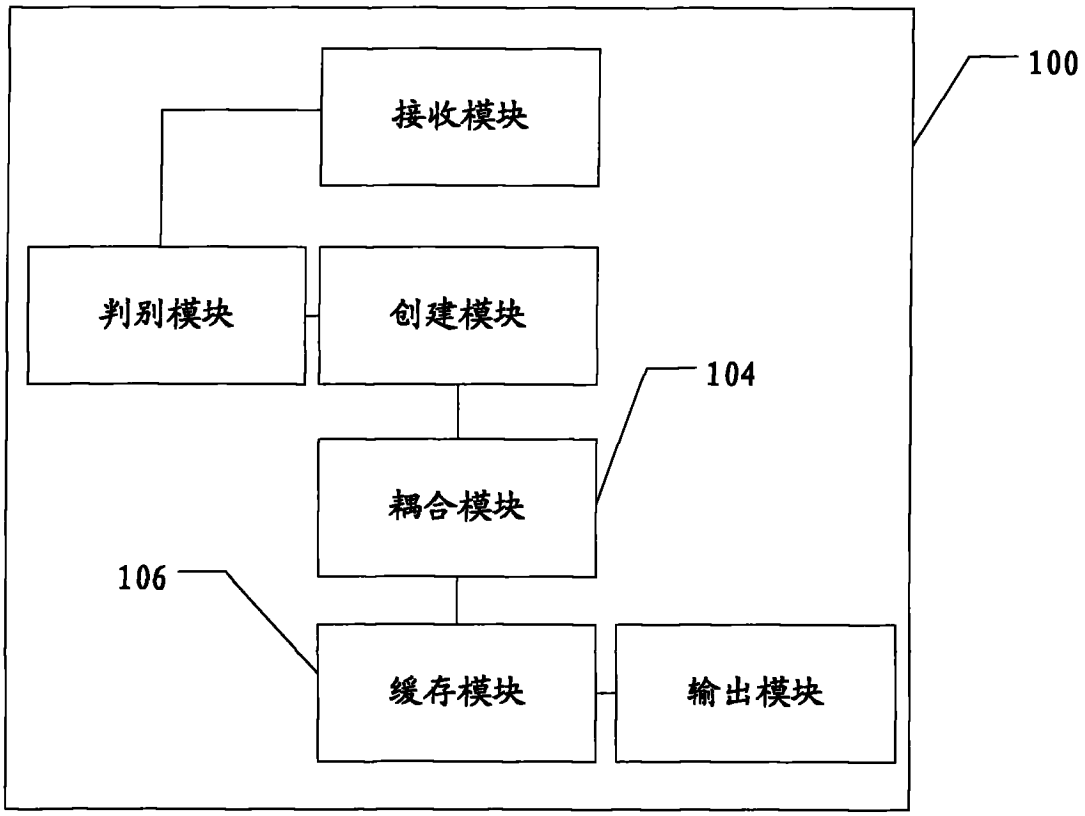

100
接收模块
判别模块
创建模块
104
耦合模块
106
缓存模块
输出模块

图 2

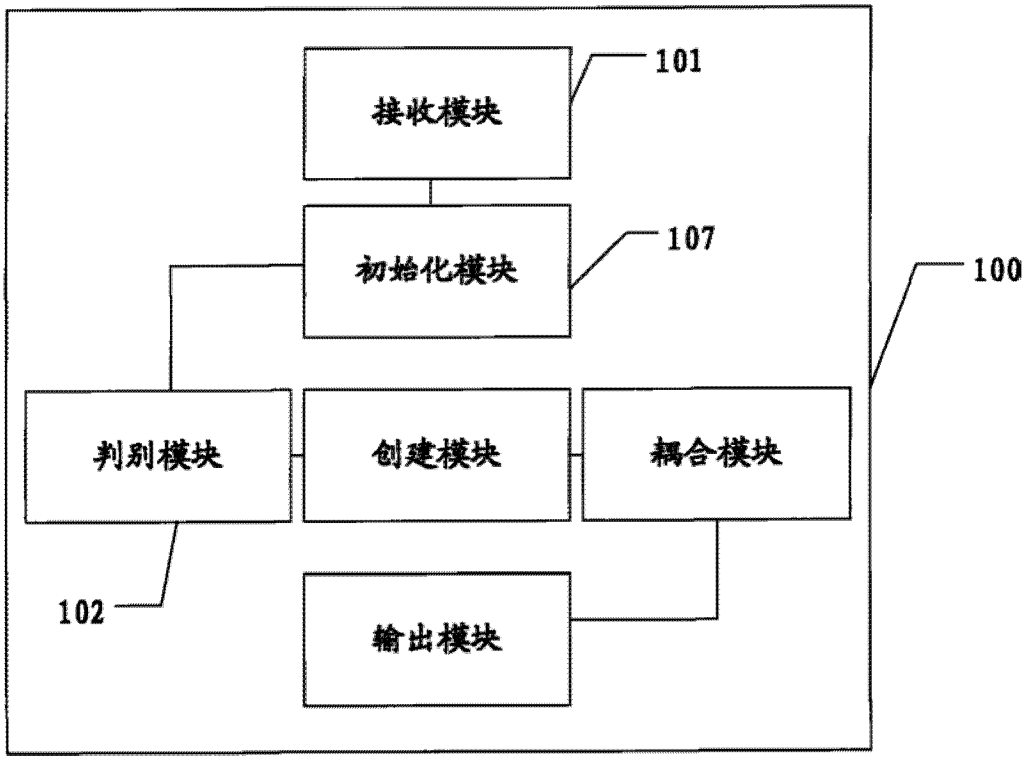

接收模块
101
初始化模块
107
100
判别模块
创建模块
耦合模块
102
输出模块

图 3

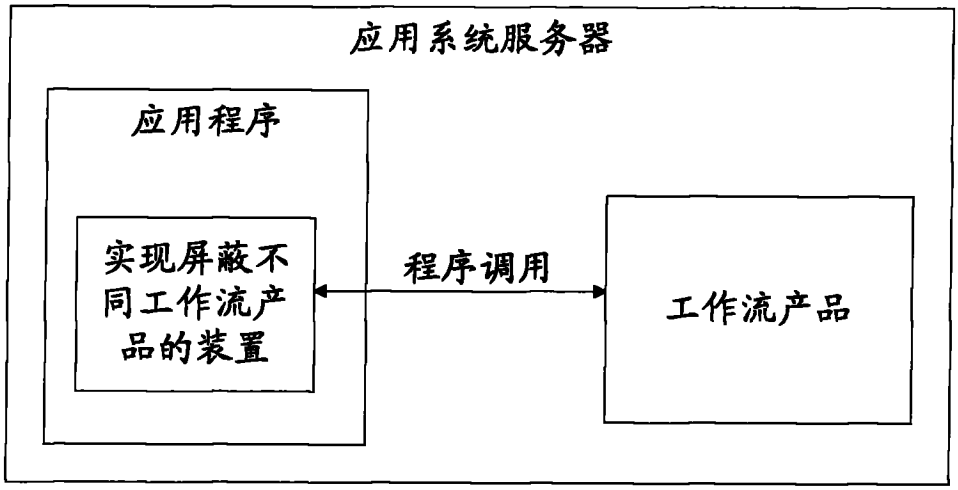

应用系统服务器
应用程序
实现屏蔽不同工作流产品的装置
程序调用
工作流产品

图 4

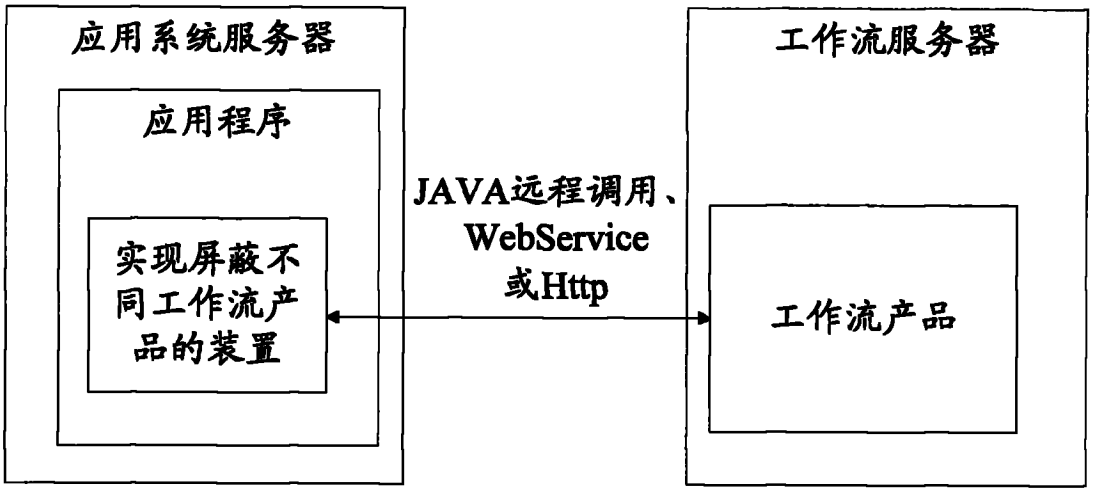

应用系统服务器
应用程序
实现屏蔽不同工作流产品的装置
JAVA远程调用、WebService或Http
工作流服务器
工作流产品

图 5

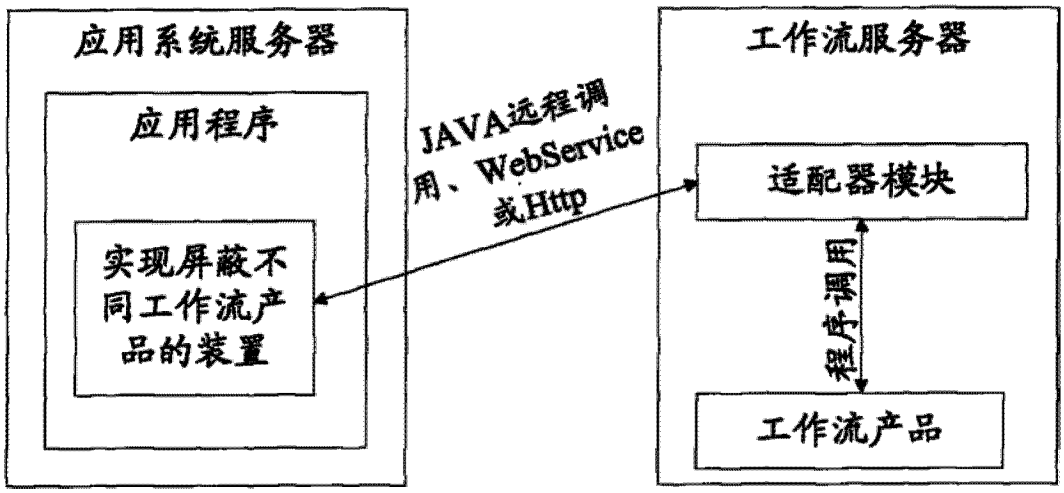

应用系统服务器
应用程序
实现屏蔽不同工作流产品的装置
JAVA远程调用、WebService或Http
工作流服务器
适配器模块
程序调用
工作流产品

图 6

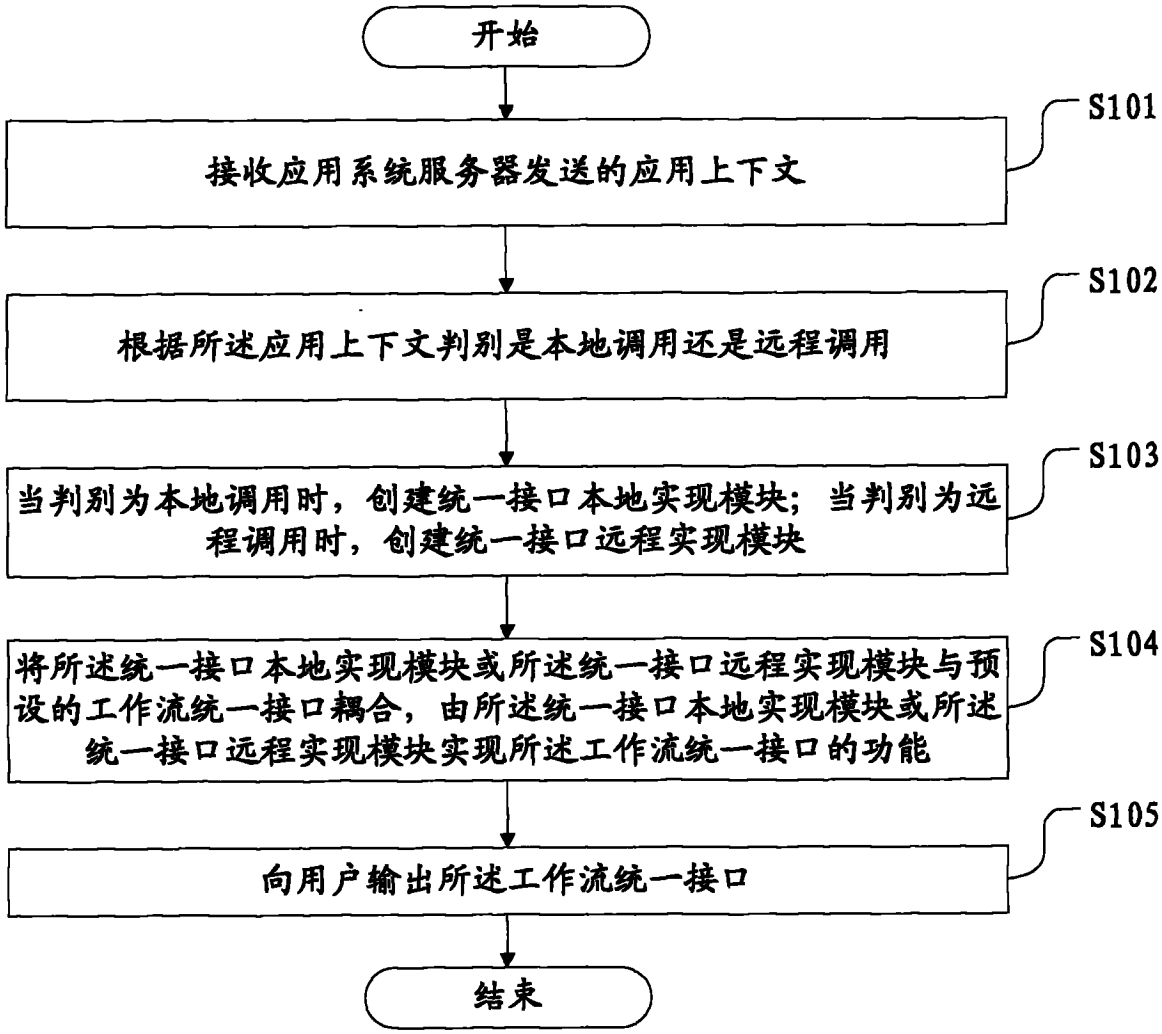

开始
S101
接收应用系统服务器发送的应用上下文
S102
根据所述应用上下文判别是本地调用还是远程调用
S103
当判别为本地调用时，创建统一接口本地实现模块；当判别为远程调用时，创建统一接口远程实现模块
S104
将所述统一接口本地实现模块或所述统一接口远程实现模块与预设的工作流统一接口耦合，由所述统一接口本地实现模块或所述统一接口远程实现模块实现所述工作流统一接口的功能
S105
向用户输出所述工作流统一接口
结束

图 7