(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 101257494 B
(45)授权公告日 2010.12.22

(21)申请号 200810082260.2
(22)申请日 2008.02.29
(30)优先权数据
07103381.5 2007.03.02 EP
(73)专利权人 国际商业机器公司
地址 美国纽约
(72)发明人 格奥尔格·奥克斯 马丁·亨克
迈克尔·贝伦特 戴尔特玛·库伯勒
(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 鲍进
(51)Int.Cl.
H04L 29/06(2006.01)

(56)对比文件
WO 2005/062571 A2,2005.07.07, 全文.
EP 1061710 A2,2000.12.20, 全文.
CN 1480885 A,2004.03.10, 全文.

审查员 刘庆峰

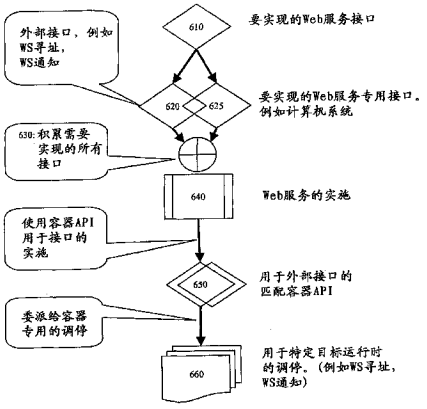
权利要求书 2 页 说明书 6 页 附图 6 页

(54)发明名称

用于访问在计算机网络中实现的资源的方法和系统

(57)摘要

本发明涉及用于访问在计算机网络中实现的资源的方法和系统，其中存在由所述资源的任一个或由访问一个所述资源的系统管理应用所使用的多个不同的运行时环境，其中所述不同的运行时环境需要使用包括多个各个不同的运行时专用的web服务标准的web服务资源框架。为了提供一种对于客户机透明地访问资源的方法，即，使得访问方法与基础运行时环境无关，提出了以下步骤：a)使用(640)资源的通用资源编程模型，该模型定义：a1)访问所述资源所需要的访问形态，以及a2)与资源相关的功能和属性的最小技术规范，b)使用包括多个“代码功能的差异”的适配机制(650、660)来使得所述通用资源编程模型适配于各个多个运行时专用的资源模型，所述运行时专用的资源模型包括各个不同的资源功能和特定的各个不同的访问形态，c)通过使用包括专用于所述资源的运行时环境的各个代码功能的差异的请求，访问在其特定的运行时环境中的所述资源。



CN 101257494 B

1. 一种用于访问在计算机网络中实现的资源的方法，其中存在由所述资源的任一个或由访问所述资源之一的系统管理应用所使用的多个不同的运行时环境，其中所述不同的运行时环境需要使用包括多个各个不同的运行时专用 web 服务标准的 web 服务资源框架，其特征在于以下步骤：

a) 使用 (640) 资源的通用资源编程模型，该模型定义：

a1) 访问所述资源所需要的访问形态，以及

a2) 与资源相关的功能和属性的最小技术规范，

b) 使用包括多个代码功能的差异的适配机制 (650、660) 来使得所述通用资源编程模型适配于各个多个运行时专用的资源模型，所述运行时专用的资源模型包括各个不同的资源功能和专用于各个多个所述不同的运行时环境的各个不同的访问形态，

c) 通过使用包括专用于所述资源的运行时环境的各个代码功能的差异的请求，访问在其特定的运行时环境中的所述资源。

2. 按照权利要求 1 的方法，其中所述资源由以下资源组组成：

a) 服务器资源，

b) 存储器资源，或

c) 网络资源。

3. 按照权利要求 1 的方法，其中处理所述不同的运行时环境的子组，其中所述运行时环境是以下运行时环境组的成员：

a)IBM Websphere 应用服务器，

b)Apache 技术，

c)“.net”技术，

d)OSGI，

e)JBOSS，

f)BEA。

4. 按照权利要求 1 的方法，其中所述适配机制包括：

a) 通用接口实现要在所述资源上执行的期望行动，

b) 将所述接口调停到所述不同的运行时环境的特定运行时环境内的适当的运行时功能。

5. 按照权利要求 4 的方法，其中通过接口到参数映射来实现所述调停步骤。

6. 按照权利要求 4 的方法，其中通过用于实现在所述通用资源编程模型中丢失的功能的编程代码来实现所述调停步骤。

7. 一种用于访问在计算机网络中实现的资源的系统，其中存在由所述资源的任一个或由访问所述资源之一的系统管理应用 (1) 所使用的多个不同的运行时环境，其中所述不同的运行时环境需要使用包括多个各个不同的运行时专用 web 服务标准的 web 服务资源框架，其特征在于：

a) 用于存储资源的通用资源编程模型 (18) 的装置，该模型定义：

a1) 访问所述资源所需要的访问形态，以及

a2) 与资源相关的功能和属性的最小技术规范，

b) 用于存储实现包括多个“代码功能的差异”的适配机制 (650,660) 的功能部件的

装置，所述功能部件用于使得所述通用资源编程模型适配于各个多个运行时专用的资源模型，所述运行时专用的资源模型包括各个不同的资源功能和专用于各个多个所述不同的运行时环境的各个不同的访问形态，

c）用于发送包括专用于所述资源的运行时环境的各个代码功能的差异的请求的装置，用于访问在其特定的运行时环境中的所述资源。

用于访问在计算机网络中实现的资源的方法和系统

技术领域

[0001] 本发明涉及系统管理的领域。具体地，本发明涉及用于访问在计算机网络中实现的资源的方法和系统，其中存在由所述资源之一或由访问所述资源之一的系统管理应用(1)使用的多个不同的运行时环境，其中所述不同的运行时环境需要使用包括多个各不同的、运行时专用的web(网络)服务标准的web服务资源框架。

背景技术

[0002] 诸如WSDM(web服务分布式管理，见www.oasis-open.org)和WS-Man(web服务管理)的web服务(WS)标准以及将来汇聚的标准描述了IT资源(操作系统、打印机、应用等)如何使用web服务向客户机应用暴露它们的可管理性能力。这些客户机应用通常是由IBM或由互联网服务供应商或其它临售商提供的系统管理应用。现有的web服务用遵循WSDL(web服务定义语言，见www.w3.org)标准的XML文档描述它们的接口。它们伴随有通常在应用服务器上部署和执行的各个web服务的实施(implementation)。

[0003] 所以，存在描述使用WSDL定义的能力的接口的公开标准，但WS-实施的实现是随其被部署到的运行时环境(WAS，OSGI，ApacheTomcat等等)而改变的。

[0004] 应当注意，本发明应用于这个领域的所有标准。当前，WSDM(在OASIS标准化)和WS-Man(在DMTF标准化)代表当前的现有技术标准。支持WSDM的和支持WS-Man的组同意把这些标准和新的标准连接在一起，这将在接下来的几年内规定。所以，WSDM构成物，诸如可管理的资源，应当被理解为可互换地用于相应的WS-Man构成物。

[0005] 参照图1，图1显示用于现有技术的基于web系统管理方法的现有技术硬件和软件环境的最基本的结构部件，一个最新现有技术的基于web服务的系统管理实施1是WSDM客户机1，它针对特定的运行时环境8A到8E，诸如在WAS 6.0上的ODI:RM、在Apache Tomcat上的AIDE、BEA等等。WSDM管理性接口4是通过提供服务接口定义的WSDL 1.1文档、相关的资源特性机制和资源元数据文档表示的接口。

[0006] 这些实施的每个实施都定义了用于资源类型的它本身的编程模型5,6,7。这样的编程模型5,6,7规定了必须如何实现各个可管理的资源，以便能够在各个运行时环境内运行。

[0007] 与单元1,4,5,8E协作的、如图1所示的当前实施有各种缺点：

[0008] 第一，它们不是便携式的，并且在对于部署目标的决定方面没有灵活性。这是由于如下事实：每个可管理的资源运行时环境规定了它本身的编程模型。

[0009] 第二，不同的运行时环境8A,...,8E实现WSDM可管理的资源以及在上述的WS-Man标准的情形下WS-资源传送所需要的WS技术规范的不同的各个完整性等级。这是由方块5,6,7的不同高度来表示的。

[0010] 此外，对于所实现的技术规范，支持各种不同的服务质量属性，诸如性能、可靠性等等。由于用于可管理资源的每个当前可得到的运行时环境8A,...,8E根据不同的编程模型而支持不同的服务质量，所以可管理资源实施的供应商不可能依赖于由基础的运行时环

境所提供的公共的、标准化的特征集。例如，运行时环境 8E，其被表示为“WAS”--Websphere 应用服务器 -- 实现运行时环境没有实现的属性。作为一个例子，环境 8E 实现 8A-8D 所没有提供的端对端 WS 安全功能。

[0011] 其中每个运行时支持不同的服务水平和服务质量的另一个例子是对 WS 寻址的支持。WS 寻址存在多个不同的版本。每个运行时环境选择在运行时的开发和测试期间可用的版本。运行时供应商提供哪些 API 来暴露 WS 寻址的功能性和互操作性水平，也是运行时供应商的自由。所以，例如，WAS 6 选择版本 2004/08，其中 WAS 6.1 具有 WS 寻址的 2006 版本。另一个例子是，从 WAS 6 到 WAS 6.1，暴露 WS 寻址功能性的 API 大大地改变。所以把 WSDM 兼容的 web 服务从 WAS 6 移动到 WAS 6.1，牵涉到很大量的转向努力 (portingeffort)。

[0012] 在此情况中，讨论不是有关“低层”编程模型（即，J2EE 相对于 OSGi）和它相应的实施（例如，IBM WebSphere 应用服务器，JBoss，Bea WebLogic，Apache Geronimo 等等）。讨论更多的是关于在 J2EE/OSGi 之上实现“容器 (container)”所需要的编程模型，这样，可管理的资源可以在它之内运行。这样的容器克服了现有技术的缺点，诸如在不同运行时之间的互操作性。

发明内容

[0013] 本发明的目的是提供对客户机透明地访问资源的方法，即，使得该访问方法与基础的运行时环境无关。

[0014] 本发明的目的是通过在所附独立权利要求中阐述的特性来实现的。本发明的其它有利的安排和实施例在各个从属权利要求中阐述。现在应当参考所附权利要求。

[0015] 简言之，本发明提供了一种在 WSDM 可管理性接口与不同的各个运行时环境之间的接口层，这缓和了各个运行时环境的实现差别。这个接口层提供对于每个资源类型可共用的编程模型，以及提供在通用编程模型与各个独立的运行时环境之间的适配机制。

[0016] 按照本发明的最广义的方面，公开了一种用于访问在计算机网络中实现的资源的方法，其中存在由资源的任一个或由访问一个资源的系统管理应用所使用的多个不同的运行时环境，其中不同的运行时环境需要使用包括多个各不相同的运行时专用的 web 服务标准的 web 服务资源框架，所述标准是诸如不同的各个目标 -- 例如 web 服务寻址、web 服务通知、web 服务安全 -- 需要使用的 WSDM、WSMAN，其中本方法的特征在于以下步骤：

[0017] a) 使用资源的通用资源编程模型，该模型定义：

[0018] a1) 访问所述不同的资源所需要的访问形态 (access modality)，以及

[0019] a2) 与资源相关的功能和属性的最小技术规范，

[0020] -- 例如，发送通知到资源的功能，或请求特定资源实例的地址的功能，或寻址特定的资源特性的功能等等 --，

[0021] b) 使用包括多个“代码功能的差异 (delta)”的适配机制 -- 这基本上是不是目标运行时环境的一部分的丢失功能 -- 用于使得通用资源编程模型适配于各个多个运行时专用的资源模型，所述运行时专用的资源模型包括各个不同的资源功能和专用于各个多个不同的运行时环境的各个不同的访问形态，以及

[0022] c) 通过使用包括专用于所述资源的运行时环境的各个代码功能的差异的请求，访问在其特定的运行时环境中的资源。

[0023] 资源编程模型在这里被理解为用于实现资源的基本编程。这包括利用由基本的运行时基础结构提供的 API 或由第三方销售商提供的库。

[0024] 访问形态在这里被理解为访问资源所需要的信息的总和。这包括寻址资源的消息的结构，如何表示资源的识别的方式，为了访问资源而要传送的安全信息，以及寻址资源所需要的所有其它上下文信息。

[0025] 与资源相关的功能和属性的最小技术规范在这里被理解为要由资源提供的功能和属性的集合。例如，这可包括识别资源和读出 - 访问 (read-access) 它的特性所需要的信息。

[0026] 本方法基本上适用于访问任何类型的资源，诸如服务器资源、存储资源、或网络类型的资源。

[0027] 在 WSDM 语言中，这个接口层在标准容器中被实现。优选地，新的 Java 标准容器由本发明提供，以便实现 WSDM 可管理的资源和相应的 WS-Man 构成物。这样的容器优选地在图 1 的底部显示的不同运行时环境之上被实现。借助于以上的特性 b)，本发明的方法在 WS- 栈的实施质量和成熟度方面能够闭合不同运行时的现有缺失的间隙。

[0028] 本发明方法有利地隐藏了基础运行时环境的实现细节，这具有如下有利效果：大大提高了 WSDM 可管理的资源的实现者对部署选择的灵活性。

[0029] 而且，即使在第三方开发者致力于室内 web 服务开发的情形下，也可以大大加速基于 WSDM 的 web 服务的开发。

[0030] 本发明方法可以应用于所有的实际存在的运行时环境，诸如：

[0031] a) IBM WebSphere 应用服务器，

[0032] b) Apache Tomcat，

[0033] c) “.net”技术，

[0034] d) OSGI，

[0035] e) JBOSS，

[0036] f) BEA，

[0037] g) 任何其它类型的 (J2EE) 应用服务器。

附图说明

[0038] 本发明是通过例子说明的，而不受附图的形状限制，其中：

[0039] 图 1 显示现有技术方法所使用的现有技术硬件和软件环境的最基本的结构部件；

[0040] 图 2 显示包括本发明方法的优选实施例所使用的、被称为“标准容器”的功能接口层部件的本发明硬件和软件环境的最基本的结构部件；

[0041] 图 3 是其中显示了本发明的标准容器的细节的、按照图 2 的示意图；

[0042] 图 4 是图 3 的标准容器的放大视图的示意图，其中显示了运行时环境实现所需 WSDM web 服务技术规范的所需版本的情形；

[0043] 图 5 是其中显示所谓的“调停器 (mediator)”提供所需版本的情形的、按照图 4 的示意图；以及

[0044] 图 6 显示根据本发明容器构建通用 web 服务的优选的本发明实施例。

具体实现方式

[0045] 通常通过参照附图，并且具体地，现在通过参照图 2，本发明方法提供接口层 17，其包括两个部件，第一，可得到的所有资源类型的多个通用编程模型。在图上显示了示例性的所选择的可管理的资源“计算机系统”的编程模型。第二，用于 WSDM- 可管理资源 20 的所谓的“标准容器”。

[0046] 对于第一部件 18，除了附图所示的资源类型“计算机系统”以外，在功能部件 18 中还存在用于任何其它当前存在的资源类型的全部多个另外的通用编程模型。因此，这个部件 18 代表编程模型的全部集合。

[0047] 按照本发明的重要方面，这些编程模型是通用的，即它们是以一般特性画出的，这样通用模型覆盖资源的任何版本、任何编程表现 (programmed appearance)，与它当前的逻辑结构和功能范围和行为无关。下面参照图 6 给出它们的细节和例子。

[0048] 容器适配器 22，24 和 26 经由各个应用编程接口连接到标准容器 20，这些容器适配器 22，24 和 26 都是被编程为连接在上述的标准容器与不同的运行时环境 8A，...，8E 的各个运行时环境之间的功能部件。

[0049] 图 3 包括图 2 的接口层 17 的示例性所选细节。本发明的容器 17 把功能和服务质量添加到 WSDM 兼容的实施所需要的 WS 标准。更具体地，在图 3 上，显示了在这方面的功能：WS 寻址、WS 通知、WS 资源特性和 WS 可靠的消息传送。这些部件都由标号 32 表示。从图 3 的示意图，本领域技术人员将会明白，例如，借助于本发明的标准容器 20 将 WS 寻址栈和 WS 通知栈的功能添加到标号为 8E 的运行时环境 WAS。类似地，WS 通知栈被添加到 OSGI/LWI 的运行时环境 8D。类似地，如在方块 34 中列出的资源特性和可靠的消息传送那样的其它功能被添加到所显示的运行时环境的任一个运行时环境。

[0050] 图 4 通过使用具体的和示例性的所选择的使用情况来详细地显示以前提到的本发明的功能，其中运行时环境实现 WS 技术规范的所需版本。

[0051] 在图 4 的例子中，本发明的标准容器 17 运行在与 WS 实施 22 相同的运行时环境上。如果需要的话，可以将其部署在分开的运行时环境上。WSDM 客户机与 WSDM 1.1 标准兼容的接口相接口。WSDM 1.1 需要特定的 WS 兼容的实施，这里称之为 WS*- 实施。

[0052] 标准容器 17 实现用于这些特定的 WS*- 实施的 API 层。在本例中，WAS 运行时环境包含一种实施，它精确地实现所需要的 WS 通知版本，因此容器的调停器部件正好传送来自 API 的请求。

[0053] 参照图 5，在下一个情形下，调停器 44 通过减少运行时缺陷而把所需的 WS 通知的能力提供给 API。这里，WSDM 客户机与被部署在 OSGI/LWI 运行时上的 WSDM 1.1 兼容的实施通信。这个运行时环境不提供所需要的 WS 通知版本 1.3，因此，调停器补充在版本 1.1 与 1.3 之间的差别。

[0054] 接着，参照图 6，描述按照优选的本发明实施例执行的总的控制流程，其用于根据本发明的容器 20 构建通用 web 服务。

[0055] 控制流程从步骤 610 开始，它代表要被构建的 web 服务应当实现的所有的 web 服务接口的总和。这是必须完成的第一设计水平行动。除了提供接口和接口需求的完全集合以外不需要特定的实际知识。

[0056] 步骤 620 显示所有外部的、标准化的 web 服务接口的累积。例子是标准化的接口，

如 WS 寻址、WS 通知等等，即，以上被称为“WS*”相关。

[0057] 步骤 625 提到 web 服务专用接口，或换句话说，不同于标准。对于这个接口的例子包括属性和方法，其定义了资源类型—诸如，例如计算机系统—的 web 服务专用部分。这里，例如：收集了诸如安装、开始、停止那样的方法和如 RunState（运行状态），ComputerSystemArchitecture（计算机系统架构）等的属性。

[0058] 步骤 630 组合和积累来自步骤 620 和 625 的接口，以形成应当被实现的 web 服务的全部外部接口。

[0059] 步骤 610 到 630 是运行时和平台无关步骤，有利地，它们可以由在一个现有技术开发环境中的特定工具支持。例子是 RationalApplication Developer（应用开发器）、Eclipse Tooling（工具）等等。

[0060] 在步骤 640，解决 web 服务的实施的运行时和平台缺陷。

[0061] 应当指出，在现有技术中，开发者必须决定他想要在哪种运行时和在哪种平台上实现 web 服务。更具体地，他需要使用平台和 / 或运行时专用的接口来完成 web 服务的实施。结果，适应各种不同的平台和运行时专用的要求是许多工作，正如在引言部分讨论的那样。

[0062] 相反，借助于本发明方法，在开发者使用通用的或通常可得到的接口来完成 web 服务本身的实施时，允许开发者仍旧待在与平台和运行时无关的“场所”。这把主要的分离方面添加到 web 服务实施的开发上。对于步骤 640 的实施，开发者现在优选地使用一组通用接口来表达想要实现的任何实际行动。

[0063] 例子是：

[0064] a) 经由 WS 通知式的互动发送通知，

[0065] b) 从 SOAP 消息等等中检索 WS 寻址上下文。

[0066] 在步骤 660，将这些接口调用调停 (mediate) 给在目标平台和运行时环境下可用的适当的运行时功能。

[0067] 这可以以从简单的接口到参数映射开始直到完全实现丢失运行时功能结束的很大范围的编程手段实现。

[0068] 在“接口到参数”映射的例子中，服务实施例如通过 SendNotification（参数）操作调用本发明的 API（容器）。如前所述的调停器模式的实施在容器 API 与可用的运行时栈的相应实现 API 之间进行映射。使用情况例如是：

[0069] A) 对于运行时 WAS 6.1：在这种情形下，调停器把服务实施调用映射到操作 WS_Notification_Send_Async(Param_1, Param_2)。

[0070] B) 对于运行时 OSGI：在这种情形下，调停器把调用映射到例如 Apache_Notification_Impl_Send_Async(Param_1, Param_2, Param_3)。

[0071] 在次情况中，不是目标运行时环境的一部分的丢失功能可以是服务供应商的专有实施或 Open Source（开源）实施。

[0072] 所以，本发明的适配机制提供丢失功能作为“代码的差异信息”，因此使得本发明的通用资源编程模型适配于任何想要的运行时专用资源模型。因此，按照本发明构建在 WSDM 客户机与各种不同的运行时环境之间的完全通用的 API。

[0073] 沿步骤 610 到 650 这样得到的本发明的 API 的优点在于，用户不需要知道，也不用

依赖于这个“填充－接口－缝隙”实施。为了实现这个 API 解决方案，优选地使用如“依赖注入 (Dependency Injection)”、动态策略图案或注解那样的现有技术。步骤 660 显示这样的 API 的最后实施。该实施专用于给定的运行时／平台组合。这些部分或者是统计地限于 web 服务本身，或者在 web 服务部署或执行期间动态地限于 web 服务本身。

[0074] 本发明可以采取完全硬件实施例、完全软件实施例、或包含硬件与软件单元的实施例的形式。在优选实施例中，本发明以软件实现，该软件包括但不限于固件、驻留软件、微代码等等。

[0075] 而且，本发明可以取计算机程序产品的形式，该计算机程序产品是从计算机可使用的或计算机可读的媒体可访问的，所述媒体提供由计算机或任何指令执行系统使用的或与计算机或任何指令执行系统结合使用的程序代码。对于本说明，计算机可使用的或计算机可读的媒体可以是任何设备，其可包含、存储、通信、传播、或输送由指令执行系统、设备、或装置使用的或与指令执行系统、设备、或装置结合使用的程序。

[0076] 媒体可以是电子的、磁的、光的、电磁的、红外的、或半导体系统（或设备或装置）或传播媒体。计算机可读的媒体的例子包括半导体或固态存储器、磁带、可拆卸计算机软盘、随机存取存储器 (RAM)、只读存储器 (ROM)、刚性磁盘和光盘。当前的光盘的例子包括紧凑盘－只读存储器 (CD-ROM)、紧凑盘－读／写 (CD-R/W) 和 DVD。

[0077] 适用于存储和／或执行程序代码的数据处理系统将包括至少一个处理器，其通过系统总线被直接或间接耦合到存储器单元。存储器单元可包括在实际执行程序代码期间利用的本地存储器、海量存储器、和高速缓存器，高速缓存器提供至少某些程序代码的临时存储，以便减小在执行期间必须从海量存储器检索代码的次数。

[0078] 输入／输出或 I/O 设备（包括但不限于键盘、显示器、指向设备等等）可直接或通过介入的 I/O 控制器被耦合到系统。

[0079] 网络适配器也可以被耦合到系统以使数据处理系统能够通过介入的私有或公共网络被耦合到其它数据处理系统或远端打印机或存储装置。调制解调器、电缆调制解调器和以太网卡仅仅是几种当前可得到的网络适配器。

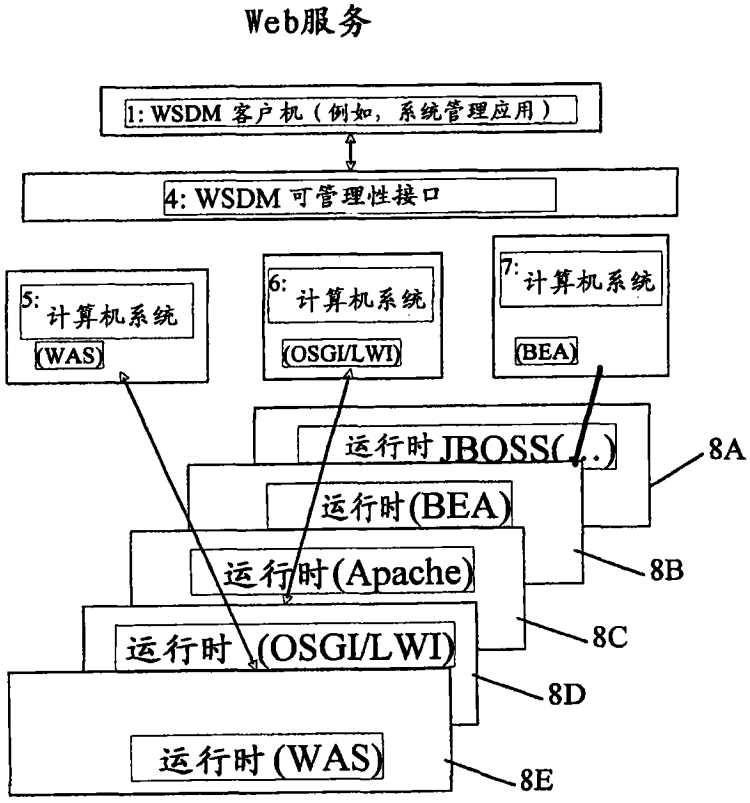

Web服务
1: WSDM 客户机（例如，系统管理应用）
4: WSDM 可管理性接口
5: 计算机系统
(WAS)
6: 计算机系统
(OSGI/LWI)
7: 计算机系统
(BEA)
运行时 JBOSS(...)
运行时(BEA)
运行时(Apache)
运行时 (OSGI/LWI)
运行时 (WAS)
8A
8B
8C
8D
8E

图1
现有技术

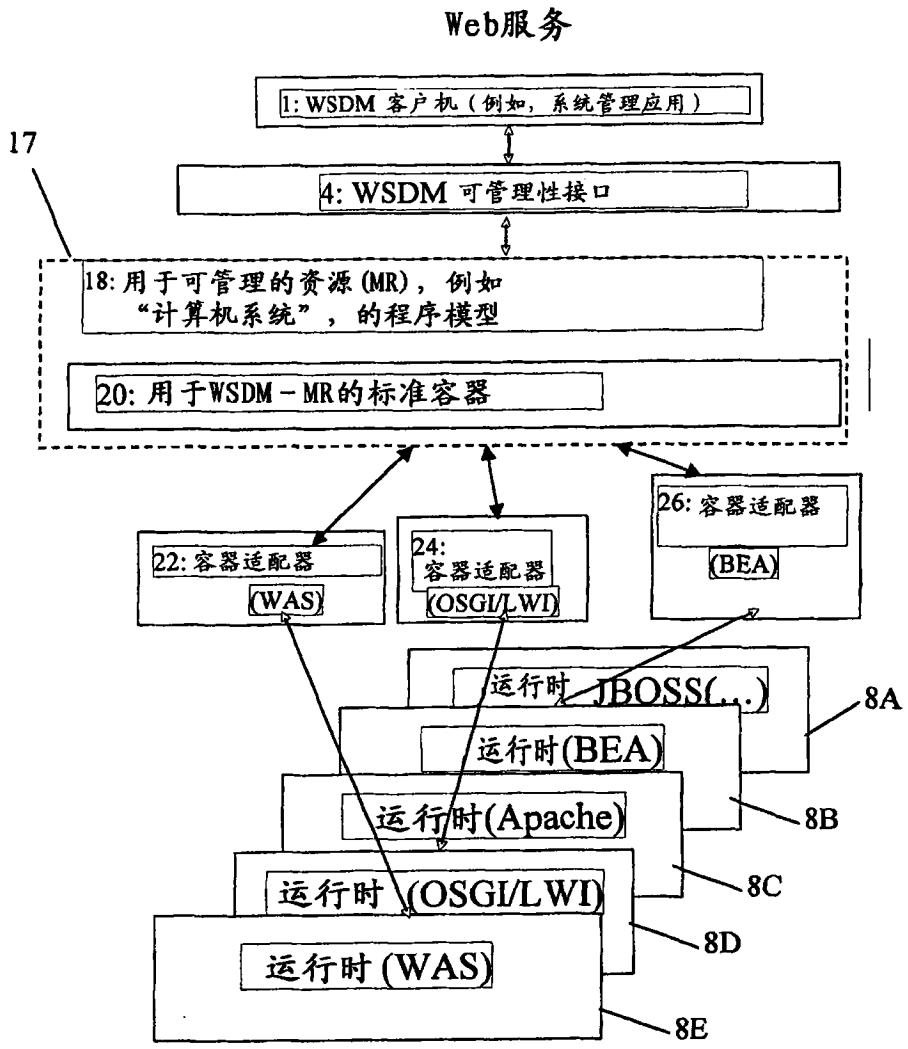
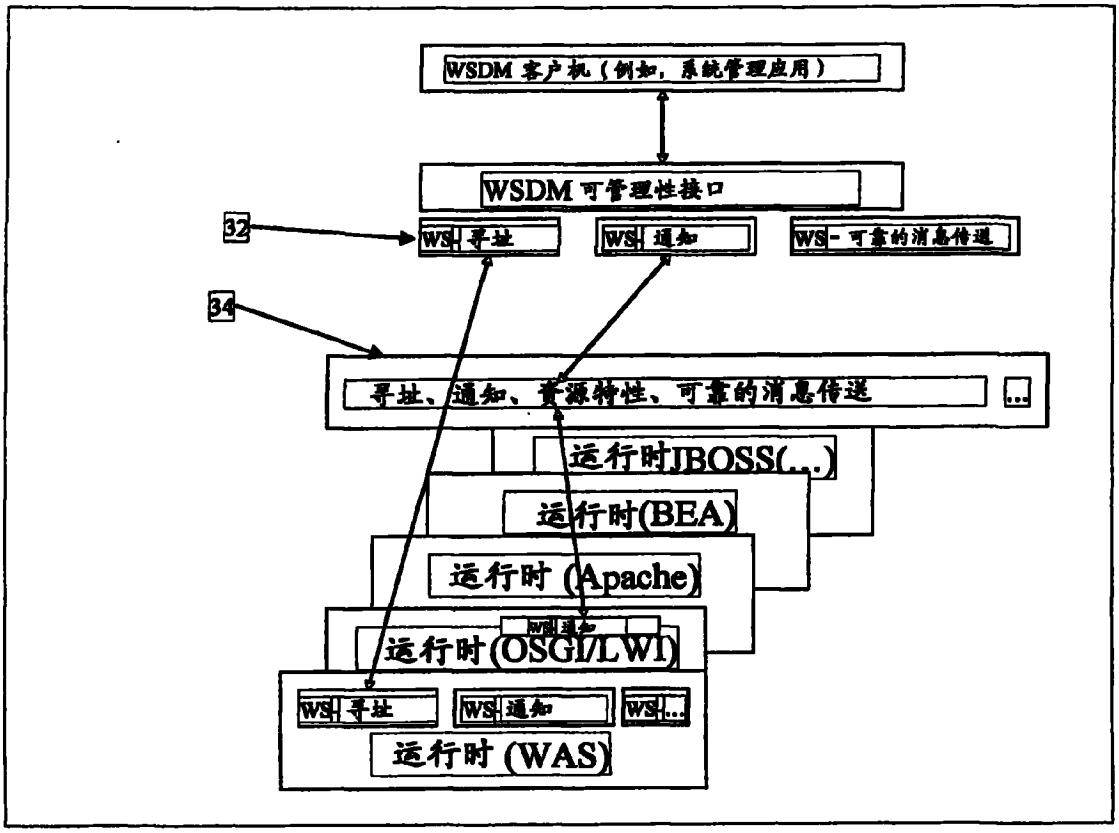

Web服务
1: WSDM 客户机（例如，系统管理应用）
17
4: WSDM 可管理性接口
18: 用于可管理的资源(MR)，例如
“计算机系统”，的程序模型
20: 用于WSDM－MR的标准容器
22: 容器适配器
(WAS)
24:
容器适配器
(OSGI/LWI)
26: 容器适配器
(BEA)
运行时 JBOSS(...)
运行时(BEA)
运行时(Apache)
运行时 (OSGI/LWI)
运行时 (WAS)
8A
8B
8C
8D
8E

图 2



包括WS-*实施的WSDM标准容器

图3

本发明

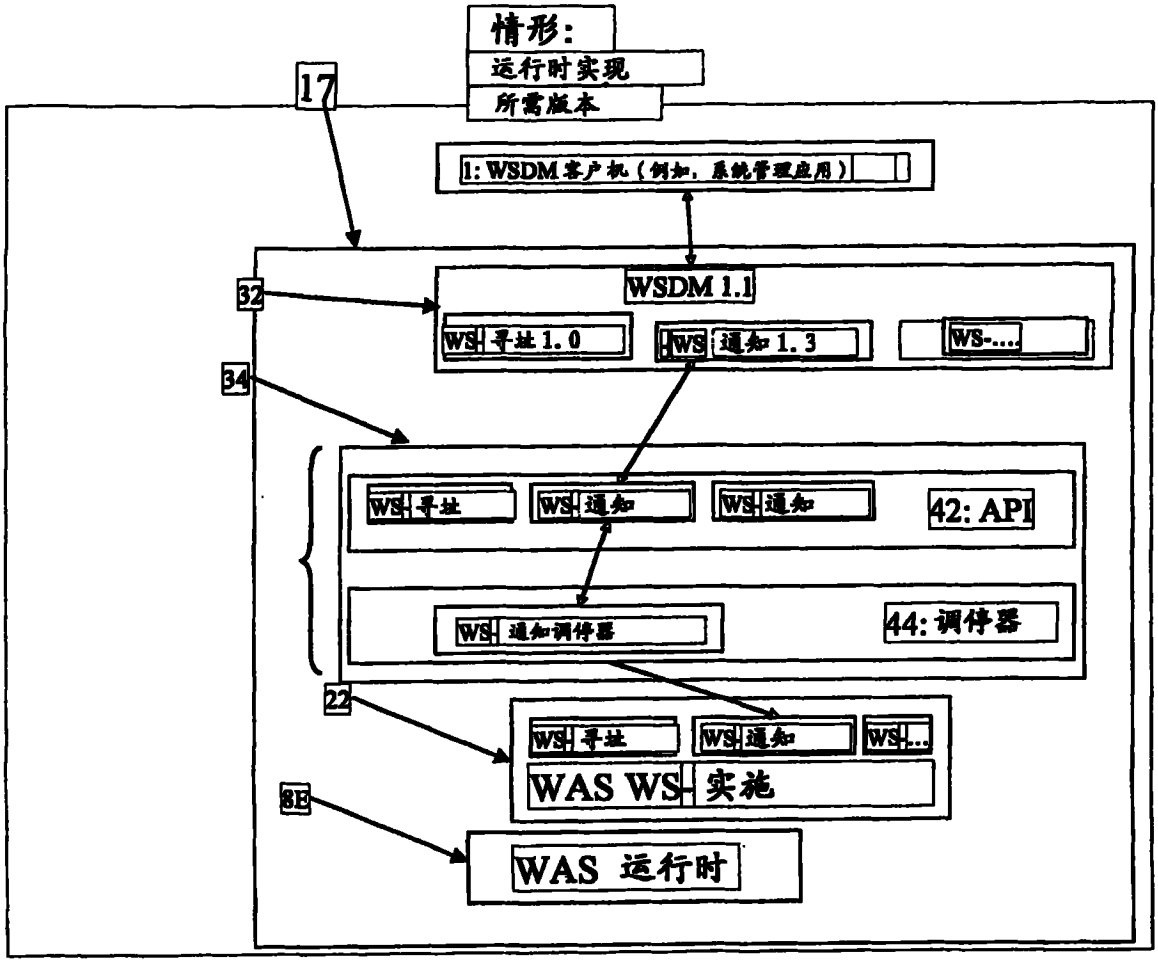
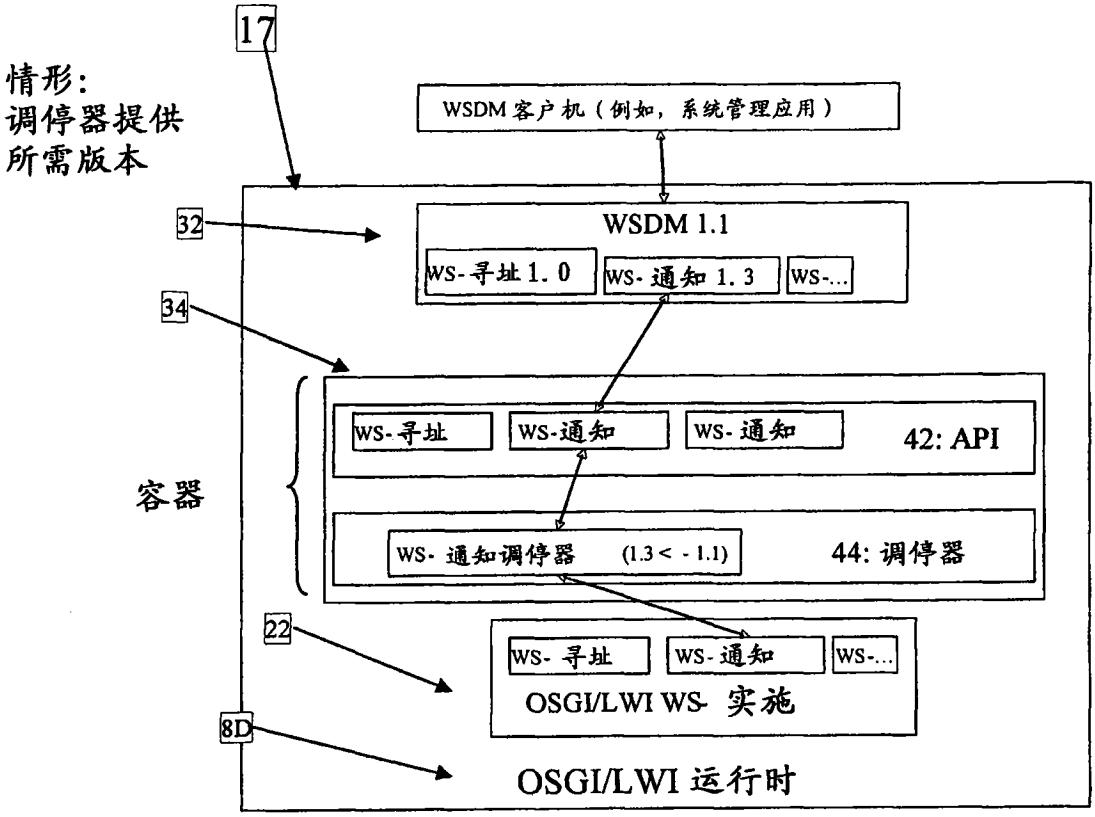

情形:
运行时实现
所需版本
17
1: WSDM 客户机（例如，系统管理应用）
32
WSDM 1.1
WS 寻址 1.0
WS 通知 1.3
WS-...
34
WS 寻址
WS 通知
WS 通知
42: API
WS 通知调停器
44: 调停器
22
WS 寻址
WS 通知
WS...
WAS WS 实施
8E
WAS 运行时

图 4



WSDM标准容器：调停情形

图5

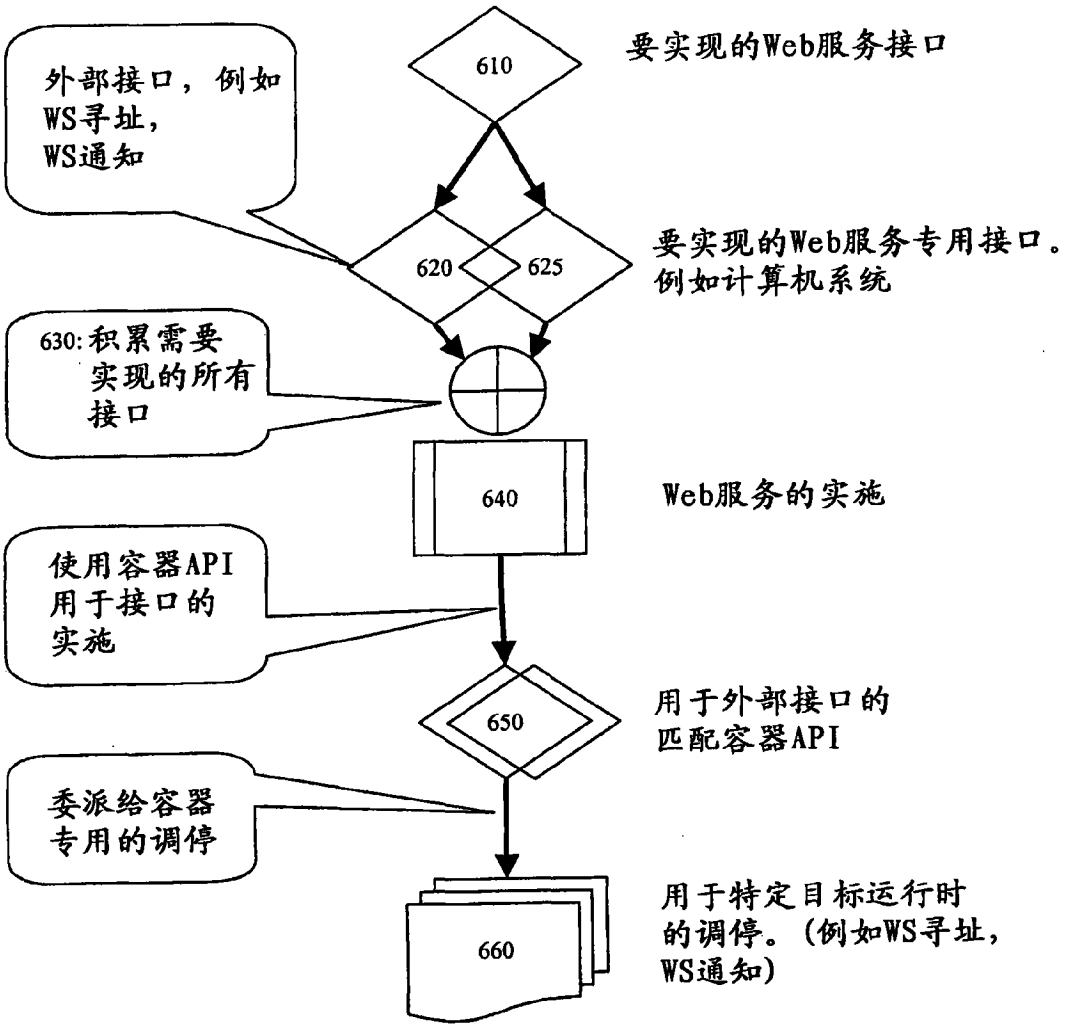

610
要实现的Web服务接口
外部接口，例如
WS寻址，
WS通知
620
625
要实现的Web服务专用接口。
例如计算机系统
630:积累需要
实现的所有
接口
640
Web服务的实施
使用容器API
用于接口的
实施
650
用于外部接口的
匹配容器API
委派给容器
专用的调停
660
用于特定目标运行时
的调停。(例如WS寻址，
WS通知)

图6