(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 101842781 B
(45)授权公告日 2012.10.03

(21)申请号 200880113955.0
(22)申请日 2008.11.03
(30)优先权数据
11/982,675 2007.11.02 US
(85)PCT申请进入国家阶段日
2010.04.29
(86)PCT申请的申请数据
PCT/US2008/082240 2008.11.03
(87)PCT申请的公布数据
WO2009/059291 EN 2009.05.07
(73)专利权人 TTI发明有限责任公司
地址 美国特拉华州
(72)发明人 穆尼尔·科钦瓦拉
约瑟芬·米卡勒夫
约翰·R·乌尔特II
(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 王波波
(51)Int.Cl.
G06F 15/177(2006.01)
(56)对比文件
US 5826005 A,1998.10.20, 说明书第2栏第65行至第4栏第57行,图2－5.
US 5317757 A,1994.05.31, 全文.
US 5826005 A,1998.10.20, 说明书第2栏第65行至第4栏第57行,图2－5.
CN 1959652 A,2007.05.09, 全文.
CN 1648873 A,2005.08.03, 全文.
审查员 吴媛媛

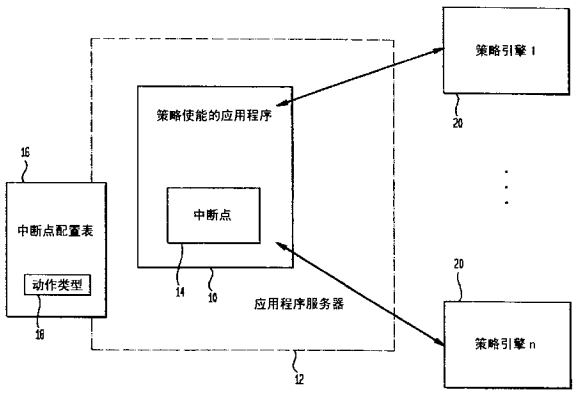
权利要求书 2 页 说明书 7 页 附图 3 页

(54)发明名称

用于策略使能编程的方法和系统

(57)摘要

在不对应用程序代码进行修改的情况下允许在电信系统中对应用程序的应用程序行为进行外部执行时适配的系统和方法包括:具有至少一个中断点并驻留在应用程序服务器上的应用程序;至少一个可识别的决策引擎;以及针对每一个中断点,具有决策引擎的至少一个标识符的中断点列表,使得在中断点中的一个处,应用程序访问中断点列表,调用与一个中断点相对应的决策引擎的所列实例,并基于该决策引擎对应用程序的行为进行适配。此外,中断点列表中的每一个条目可以具有序列号,使得如果针对同一个中断点的两个条目具有相等的序列号,可以并行地调用在这些条目中识别出的决策引擎。



CN 101842781 B

1. 一种系统，用于允许对应用程序的行为进行外部执行时适配，所述系统包括：

至少一个决策引擎，被适配为在决策引擎计算设备上操作，并且被配置为与驻留在应用程序服务器上的应用程序交互，所述应用程序服务器包括与所述决策引擎计算设备分离的服务器计算设备，其中，所述至少一个决策引擎在所述应用程序外部，并且所述应用程序具有至少一个断点；以及

中断点配置列表，所述中断点配置列表针对至少一个中断点，包括所述至少一个决策引擎的至少一个标识符，所述应用程序被配置为在所述至少一个中断点处访问所述中断点配置列表，所述中断点配置列表被配置为允许在所述应用程序执行时修改所述至少一个决策引擎的至少一个标识符；

其中，在所述至少一个中断点处，所述应用程序访问所述中断点配置列表，使用所述至少一个标识符调用针对所述至少一个中断点的所述至少一个决策引擎，以及

所述中断点配置列表中的至少一项以及所述至少一个决策引擎被配置为：至少部分地基于对所述中断点配置列表中的所述至少一个决策引擎的至少一个标识符的修改以及对所述至少一个决策引擎的修改，在所述应用程序执行时对所述应用程序的行为进行适配。

2. 根据权利要求 1 所述的系统，其中，所述中断点配置列表在所述应用程序外部。

3. 根据权利要求 1 所述的系统，其中，所述中断点配置列表还包括：针对所述至少一个标识符的属性映射以及针对所述至少一个标识符的动作类型。

4. 根据权利要求 1 所述的系统，其中，所述中断点配置列表还包括：与所述列表中的每一个条目相关联的序列号。

5. 根据权利要求 1 所述的系统，其中，所述中断点配置列表被配置为：动态地将所述至少一个中断点映射到相应的所述至少一个决策引擎。

6. 根据权利要求 1 所述的系统，还包括：驻留在应用程序服务器上的应用程序。

7. 根据权利要求 6 所述的系统，其中，所述中断点配置列表还包括：与所述列表中的每一个条目相关联的序列号；并且

如果针对所述至少一个中断点的每一个条目的序列号是相等的，则并行地进行调用。

8. 一种用于允许在电信系统中对应用程序的行为进行外部执行时适配的方法，所述方法包括：

在应用程序中的中断点处，访问所述应用程序外部的中断点配置列表，针对所述中断点，所述中断点配置列表具有至少一个决策引擎的至少一个标识符，所述至少一个决策引擎在所述应用程序外部，所述中断点配置列表被配置为允许在所述应用程序执行时修改所述至少一个决策引擎的至少一个标识符；

使用所述至少一个标识符调用与所述中断点相对应的所有决策引擎，以及

至少部分地基于对所述中断点配置列表中的所述至少一个决策引擎的至少一个标识符的修改以及对至少一个决策引擎的修改中的至少一项，对所述应用程序的行为进行适配。

9. 根据权利要求 8 所述的方法，其中，访问所述列表的步骤包括：访问针对所述至少一个标识符的属性映射以及针对所述至少一个标识符的动作类型。

10. 根据权利要求 8 所述的方法，其中，访问所述列表的步骤包括：访问与所述列表中的每一个条目相关联的序列号。

11. 根据权利要求 10 所述的方法，其中，如果针对至少一个中断点的每一个条目的序列号是相等的，则并行地进行调用。

12. 根据权利要求 8 所述的方法，还包括：动态地将中断点映射到至少一个决策引擎。

用于策略使能编程的方法和系统

技术领域

[0001] 本发明大致涉及通信、信息和娱乐服务的编程。

背景技术

[0002] 以传统形式编写的软件应用程序具有在应用程序代码内定义的所有功能。一旦部署了这样的应用程序，提供该应用程序的提供商便没有影响应用程序的行为的余地。应用程序具有少数可以调节的参数，然而通常除非修改了应用程序代码，否则应用程序的行为是固定的。

[0003] 希望专注于动态市场的服务提供商需要快速提供新服务的灵活性。与修改应用程序代码相关的周期令人不可接受地延缓了该过程。从而，对于应用程序代码，需要在不等待软件发布的情况下便可以对应用程序行为进行修改。

[0004] 在用于构建智能电话网络服务的智能网和高级智能网概念的开发中，涉及到类似的动机。通过改变在电话交换机内的代码来构造新的服务是可能的，然而这相当的慢和麻烦。所选方式是精确地定义呼叫模型，该呼叫模型指定了在建立双方之间的连接时电话交换软件的状态。在呼叫模型中的各个点处，可配置电话交换机向外部服务控制点查询关于怎样进行处理的指令。通过改变服务控制点中逻辑的行为而不对交换系统内的软件自身做出改变，便可以产生新的服务功能。

[0005] 该方式极大地依赖于对交换系统的状态进行指定的公共呼叫模型和所期望的行为。因为从一个系统到另一个系统，建立呼叫的处理是一致的，因此可以定义这样的呼叫模型。呼叫模型的定义和商定花费相当多的时间，这是可以容忍的，因为电话呼叫的处理在时间上是相对静态的。在提供商频繁地提供新的并且变化的服务的环境中，为服务逻辑的内部状态定义如此详细的模型将是不可能的。虽然该公共呼叫模型技术具有所期望的特性，然而不能将其直接应用于对紧急的、新的并且变化的服务所进行的处理。

[0006] 电信设备厂商和运营商通过向呼叫处理系统增加附加特征来解决类似的问题。该商定的解决方案（高级智能网 (AIN) 的基础）定义了另一公共呼叫模型，作为对电话呼叫的建立进行处理的基础。该公共呼叫模型提供了事件的预定义集合，其中，可以向外部系统查询决策。例如，美国专利 5,940,487，“Programmable call processing systemand method”，Bunch 等人示意了在与 AIN 耦合的分布式电信交换系统上的这种方式。这种处理呼叫的服务交换点（交换机）与处理服务逻辑的服务控制点的分离允许在不对交换机内的软件进行改变的情况下对新的呼叫服务进行了定义。这种方式对于控制电路交换网内的电话呼叫起到很好的作用，并通过诸如 Parlay 组织规范和 Java 社区处理 JAIN 规范等工业论坛的结果扩展到下一代分组交换网络中。需要这样的标准化的事实指示了该方式的局限在于，其处理特定的功能（呼叫控制）并要求事先协定和 / 或对功能的标准化。从而，该方式适于在时间上静态的应用程序功能。

[0007] 在美国专利 6,970,901，“Device and method for swapping out a partof a service logic program”中，Moritz 教导了用于在多个实体上分布服务逻辑的机制。

Moritz 特别地专注于使用作为客户端的智能终端的日益增加的能力，这是通过将服务逻辑的一部分分布到客户端来实现的。Moritz 公开了用于使用这种分布式的方式来确定与计费相关的信息的方法。对客户设备的分布可以实现个性化，然而并未易于实现对整个服务的行为的修改，因为必须将改变传播到所有的客户端设备。因为可以事先对通信机制 (Moritz 中的“计费票”) 进行定义和编码，简化了对单个主题 (如，计费) 的分布的实现。从而，Moritz 解决了灵活的业务逻辑编程的需求，然而是在非常有限的领域中解决的。

[0008] 在美国专利 6,967,957，“Architecture for the rapid creation oftelephony services in a next generation network”中，Anjum 等人描述了面向对象的呼叫模型，该呼叫模型“向应用程序隐瞒了基本的呼叫状态管理协议和硬件”。特别地，Anjum 等描述了新的呼叫模型，该呼叫模型被设计为抽象得足以表示在电路交换和分组交换电话网络中的呼叫控制。这种呼叫模型比其所基于的 Java 电话应用程序接口 (JTAPI) 模型更加灵活，然而仍然表示单个功能。从而，该模型不能够应用于更一般的情况中，在更一般的情况中，事先没有很好地定义行为或对行为有清楚地了解。

[0009] 开放移动联盟 (OMA) 内的策略评估执行管理 (PEEM) 研究描述了用于策略评估和执行的架构来作为 OMA 的服务使能器 (enabler) 的支持，在草案需求文档“Policy Evaluation，Enforcement andManagement Architecture”(OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-200600625-D) 中对此进行了指定。这些使能器包括诸如群列表管理、消息收发和位置等功能。该架构被设计提供针对这些使能器的公共框架，以查询策略决策。虽然不要求，但作为用于在此描述的策略使能编程的构造块，这种能力将是有用的，因为这种能力可以简化系统间映射接口的处理。然而，OMA 的工作没有指定使能器将如何确定何时查询策略引擎 (PEEM 使能器) 或者查询哪个策略引擎。

[0010] 另一种方式为工作流系统所支持。在这些常常应用于复杂的命令处理的系统中，处理步骤的集合是以在运行时进行翻译的文本的形式定义的，而不是以软件代码的形式编译的。这提供了灵活性，因为可以在不改变工作流引擎的代码的情况下修改工作流。然而，翻译工作效率低下，导致性能足以用于命令处理，然而对于实际业务的执行来说通常是不足的。

[0011] 数据库系统还提供了另一种机制，在该机制中，可以调用触发器和所存储的流程来执行逻辑，该机制提供了大量的灵活性。然而，仅存在针对于对数据库中数据的插入、删除、更新和选择的数据库操作的，可以执行所存储的流程的操作。该方式的问题在于对数据库操作的限制；整个规范在数据库系统内，并且除非巡览过数据和相关联的触发器和所存储的流程，否则不是能够轻易对照或修改的。

[0012] 因此，当前的处理被限制于特定的软件应用程序类型 (如，呼叫处理)，并且在应用程序处理模型可以使用之前，具有该应用程序处理模型所需的高级协定和 / 或标准化。其它当前的处理仅存在于诸如数据库和工作流系统等中间件内，这限制了这些其它的当前处理的范围，并且不满足对于服务执行的性能需求。

[0013] 从而，需要能够动态地确定何时以及在何处查询诸如策略引擎等外部决策点的系统。还需要结构化和高效的、用于配置这种动态查询的机制，其特征在于，可以被配置为受控的选项集合的中断点。

发明内容

[0014] 本发明有利地提供了用于构造和部署应用程序软件的设计架构和方法学，该设计架构和方法学允许在执行期间，在不改变应用程序的代码的情况下，适配或配置应用程序的行为。提供了通过应用程序的配置或适配，针对不同的目的对单个软件应用程序的多次重用。此外，本发明的系统和方法实现了新的业务行为的快速部署，这是因为可以在没有与软件开发周期相关联的延迟的情况下传递这些新的业务行为。

[0015] 一种系统和方法，用于允许在电信系统中对应用程序的应用程序行为进行外部执行时适配，所述系统包括：具有至少一个中断点并驻留在应用程序服务器上的应用程序；至少一个可识别的决策引擎；以及针对每一个中断点，具有决策引擎的至少一个标识符的中断点列表，使得在中断点中的一个处，应用程序访问中断点列表，调用与中断点相对应的决策引擎的所列实例，并基于该决策引擎对应用程序的行为进行适配。

[0016] 中断点列表可以是应用程序外部的文件，并可以包含针对每一个标识符和针对每一个标识符的动作类型的属性映射，此外，中断点列表可以动态地将中断点映射到决策引擎。

附图说明

[0017] 通过以非限制地示出的本发明的实施例的方式来参考所提到的图，对随后的详细描述进行了进一步的描述，在图中，相同的附图标记表示所有图中类似的部分。然而，应该理解的是，本发明不限于所示出的精确的布置和手段。在图中：

[0018] 图 1 是用于策略使能的程序执行的架构；以及

[0019] 图 2 是中断点配置的示例性的表；以及

[0020] 图 3 是示出在处理中断点配置表中的条目时，应用程序的行为的流程图。

具体实施方式

[0021] 提出了允许应用程序行为的外部执行时间适配的系统和方法。该方式的关键特征是，向应用程序开发员提供了与所编译的代码相关联的性能以及对应用程序流内的决策点和可能的动作进行定义的灵活性的，允许将这些决策点动态地映射到外部的评估引擎，以及使得当部署应用程序时或甚至在执行应用程序时能够将被配置的应用程序的行为，而不是仅在应用程序的设计和开发期间能够配置应用程序的行为。

[0022] 有两种支持这种系统和方法的特定机制。第一种是外部表，如中断点配置表。应用程序在其执行期间到达中断点时读取该表，以确定采取什么动作。基于对在给定的中断点处应用程序已定义了什么属性或变量的了解，可以与应用程序分离地定义该表中的条目，并且甚至可以在应用程序运行时改变该表中的条目。第二种适配方法包含在外部系统中。这些系统内的逻辑、策略和 / 或规则将创建将要影响到应用程序的后续流程和行为的响应。采用这两种机制，可以在不必重写或修改应用程序代码的情况下对应用程序的行为进行重大的修改。

[0023] 图 1 示出了本发明系统的一个实施例。策略使能的应用程序 10 驻留在电信系统（未示出）的应用程序服务器 12 中。策略使能的应用程序 10 是以应用程序逻辑的流程内的可配置中断点 14 的指定集合来构建的。当应用程序代码的执行到达这些中断点 14 中的

一个时，应用程序代码查看中断点配置的列表或外部表 16（如，中断点配置表），所述中断点配置的列表或外部表 16 包含包括动作类型 18 在内的数据。通过评估该数据，应用程序 10 可以确定在中断点 14 处将要采取的动作。为了管理表的复杂度，可以限制可能的动作类型的数目。对于应用程序，一般的动作可以是向外部系统 20 发送数据，并使用所产生的响应来确定应用程序的后续动作。如在其已编译的代码内所定义的那样，执行这些后续动作，作为具有相关联的性能的应用程序 10 的一部分。外部系统 20 可以是决策引擎、策略引擎或从应用程序 10 接收输入、应用所定义的一个或多个策略的集合以及提供响应的策略决策点。通过修改该策略中的一个或多个，而不是通过改变应用程序的代码来改变应用程序的行为。在备选的实施例中，可以在应用程序 10 中实现跳转表（未示出）。在这种情况下，可以通过修改外部决策点或通过修改和重新编译应用程序 10 来改变应用程序的行为。

[0024] 注意，外部系统 20 还可以是工作流系统、数据库系统或甚至另外的软件应用程序。甚至在将决策点硬编码于策略使能应用程序 10 中的情况下，可将映射改变为指向不同的软件应用程序的事实也提供了灵活性。

[0025] 策略使能应用程序 10 可以对其从外部策略引擎 20 接收到的响应做出反应的方式有很多。构造在确定其后续行为时给予外部系统 20 完全的灵活性的应用程序 10 可能是极端困难和容易出错的。然而，通过对动作类型或选项 18 的范围进行了适当的限制，可以将复杂度保持在可管理的水平。特别地，可以基于外部系统 20 提供的响应来设计应用程序，以支持数目有限的动作类型 18。这些动作类型 18 的示例可以包括分支或双路的决策点动作、空动作和变量替换动作。

[0026] 在双路决策点动作 18 中，应用程序 10 向外部系统 20 发送消息，并且外部系统提供二进制（即，“是 / 否”或“真 / 假”）响应。如果响应是“是”，那么应用程序 10 沿着一条预定义的路径前进，如果响应是“否”，应用程序 10 沿着另一条路径前进。当应用程序 10 在该决策点动作 18 中仅执行两个选项中的一个时，基于外部系统 20 用来进行决策的策略或处理，整个行为可以发生极大的改变。可以将此扩展到可能存在多条路径的多路决策动作。

[0027] 作为示例，被设计为向用户传递或发送数字内容项的应用程序 10 可以包括刚好在内容发送之前的中断点 14。响应于该中断点 14，应用程序 10 在向外部策略引擎 20 的查询中提供识别用户以及内容项列表的信息。如果响应是“是”，应用程序被编程为继续进行内容的传输，如果响应是“否”，取消传输。在一种情况下，外部系统 20 可以是确定用户是否有充足的结余来支付该项的实时计费引擎。如果用户的结余充足，外部系统 20 将用户的账户计入借方并返回“是”，如果结余不足，外部系统 20 返回“否”。在另一种情况中，外部系统 20 可以是授权系统，该授权系统仅允许将内容下载到特定物理位置内的用户，以例如作为公司的安全措施或作为餐馆吸引游客的手段。如果用户位于针对该内容项指定的区域范围内，外部系统 20 返回“是”，如果没有，系统 20 返回“否”。从而，两种使用截然不同的外部系统 20 的完全不同的服务使用同一个内容传递策略使能的应用程序 10。

[0028] 在空动作 18 的情况下，应用程序 10 只是执行中断点 14 并继续操作。空动作适于应用程序只是需要告知外部系统 20 某些事件的情况。在上述的内容下载服务示例中，可以指示应用程序 10 刚好在内容发送之前执行包括空动作在内的中断点 14。执行该中断点 14 可以使得外部系统记录包括内容项列表在内的下载事件，以使得在下一个记账周期期间针对该内容向具有后付费账户的用户收费。

[0029] 在变量替换动作 18 的情况下，应用程序 10 使用来自外部系统 20 的响应来改变应用程序 10 正在处理的变量的值。在以上的内容下载服务示例中，可以指示应用程序 10 刚好在内容发送之前执行包括变量替换动作在内的中断点 14。执行该中断点 14 可以使得外部系统 20 过滤内容项的列表，移除其评定信息指示其不能被用户所接受的那些内容项，这可能是基于所识别的用户的年纪或偏好来进行的。然后，外部系统 20 向应用程序 10 返回经过滤的列表，应用程序 10 使用经过滤的列表来替换初始的内容项列表，并继续进行下载。

[0030] 可以限定其它的动作类型 18，在对应用程序流的控制中提供更大的灵活性。注意，期望限制动作类型或选项 18 的数目，以避免在应用程序 10 的构建中不适当的复杂度。

[0031] 策略使能的应用程序 10 可以在应用程序服务器 12 或服务传递平台内执行，并可以本地访问中断点配置表 16，该中断点配置表 16 可以指定与每一个中断点 14 相关联的应用程序的行为。图 2 示出了可由负责产品 / 服务部署的供应系统填充的示例性的表 16。表 16 不仅包括以上讨论的动作类型 18，还包括下面将要更详细地描述的随后的字段 :策略引擎或外部系统 20 的地址、序列号 24 和属性映射 26。策略使能的应用程序 10 可以与一个或多个外部策略决策点交互。配置表 16 将程序流内的点与交互的方式（如，发送的属性）相链接，在所述程序流中，可以以与其交互的特定决策点来执行交互。

[0032] 虽然在此的讨论描述了策略决策点，在此处定义的机制中，没有将外部系统 20 限制为策略评估器。任何具有所定义的调用接口的外部系统 20 都可以被用来替换策略决策点。

[0033] 表 1 中示出了简化的抽样配置表 16。在这种情况下，存在与中断点 14 相关联的两个中断点条目。针对第一个条目，应用程序 10 必须通过经由表 16 中的系统地址 22 向策略引擎 1 发送属性 A、B 和 C 来执行属性映射，并使用响应来替换属性 C 的值。针对第二个条目，应用程序 10 必须经由表 16 中的系统地址 22 向策略引擎 1 发送属性 26A、C 和 D，并基于“是 / 否”响应来执行决策点动作 18。序列栏中的序列号 24 指示了与单个中断点 14 相关联的调用的执行顺序。可以并行地执行共享序列号 24 的调用。在没有使用序列号 24 的情况下，应用程序 10 可以以在文件中找到的顺序来执行查询。

[0034] 表 1 中断点配置表

[0035]

中断点 ID	中断点类型	序列号	系统地址	属性映射	响应替换
1	变量替换	1	策略引擎 1	A、B、C	C
1	分支	2	策略引擎 1	A、C、D	
2	空	1	策略引擎 1	X、Y、Z	

[0036] 图 3 包含示出了图 1 所示的应用程序 10 的行为的流程图。参考该图的左侧，在步骤 S 1 处发起时，应用程序 10 通过阶段 1 执行所设计的活动。在步骤 S2，应用程序到达中断点 114。在该点处，应用程序查看中断点配置表 16，看是否有任何与中断点 114 相关联的条目。

[0037] 图 3 的右侧示出了在中断点 1 模块内用于处理这些条目的流程。一旦在步骤 S3

处进入，应用程序 10 在步骤 S4 处调用外部系统 20，使用表 16 中定义的属性映射 26。当外部系统 20 响应时，应用程序 10 在步骤 S5 处确定正在处理的中断点的类型或动作类型 18。如果中断点类型 18 是变量替换，应用程序 10 在步骤 S6 处执行配置表 16 中定义的替换，并前进至步骤 S7 检验是否有更多的中断点 14 要处理。如果中断点类型 18 是空，应用程序 10 不需要等待响应，并直接前进至步骤 S7 以检验是否有更多的中断点 14 需要处理。如果中断点类型 18 是分支，应用程序 10 在步骤 S8 处检验来自外部系统 20 的响应。如果外部系统 20 的响应是“是”，应用程序 10 前进至在步骤 S7 处的检验更多的中断点 14。如果外部系统响应是“否”，应用程序在步骤 S9 处以“否”的退出条件从对任何其它中断点条目 14 的处理中断，并返回主应用程序流程。当没有更多的终端点条目 14 要处理时，应用程序 10 在步骤 S10 处以“是”的退出条件返回主应用程序流。

[0038] 没有示出在与外部系统 20 的交互中对异常和错误条件的处理。应用程序 10 可以被编程为具有各种异常处理行为，以处理返回的错误和无响应的超时。备选地，可以对表进行扩展以指示在特定的处理异常的情况下所采取的动作。对于本机制来说，应用程序 10 处理这些情况的方式不是关键的。

[0039] 该动态编程方式的关键方面是，可以将应用程序中断点 14 动态地映射到决策点系统 20 以及决策点系统 20 内特定的策略。有无数的可以执行该映射的方法。可能最直接的方法是人工方式，在人工方式中，人类分析者创建中断点配置表 16、定义与每一个中断点 14 相对应的策略、将配置表 16 加载到应用程序服务器 12 中以及将策略加载到决策点中。如果可以以结构化的方式使应用程序中断点 14 和策略可见，映射处理中更大的自动化和有效程度是可能的。

[0040] 设计环境可以将所有中断点 14 的结构化表示导入到用户所选的应用程序 10 中。设计环境可以从其所知的决策点取得策略。当人类用户选择中断点 14（表 2 中示出了中断点 14 的简要表示）时，设计环境可以取得并呈现那些有效的策略。例如，如果中断点 14 仅允许分支操作或两路决策动作 18，只有那些返回是 / 否或真 / 假值的那些策略才是适合的。在选择策略后，用户可以将在所选中断点 14 处从软件应用程序 20 可用的变量映射到在如表 3 所示的策略的接口中指定的变量。然后，设计环境可以使该映射生效，例如确保诸如串、整数等的类型匹配。在以这种方式成功定义所有的映射后，设计环境可以创建对应的中断点配置表 16 并将其安装在应用程序服务器 12 上。

[0041] 表 2 简化的中断点表示结构

[0042]

中断点
名称
描述
所允许的类型（空、变量替换、分支）
可用变量（名称和类型）
…

[0043] 表 3 简化的策略调用接口结构

[0044]

策略
名称
描述
输入［变量1(名称/类型),变量2…］
输出［变量1(名称/类型),变量2…］
…

[0045] 虽然已在具体的实施例中对本发明进行了描述,然而应该意识到的是,不应将本发明解释为被这些实施例所限制,而是应该根据下面的权利要求来进行解释。

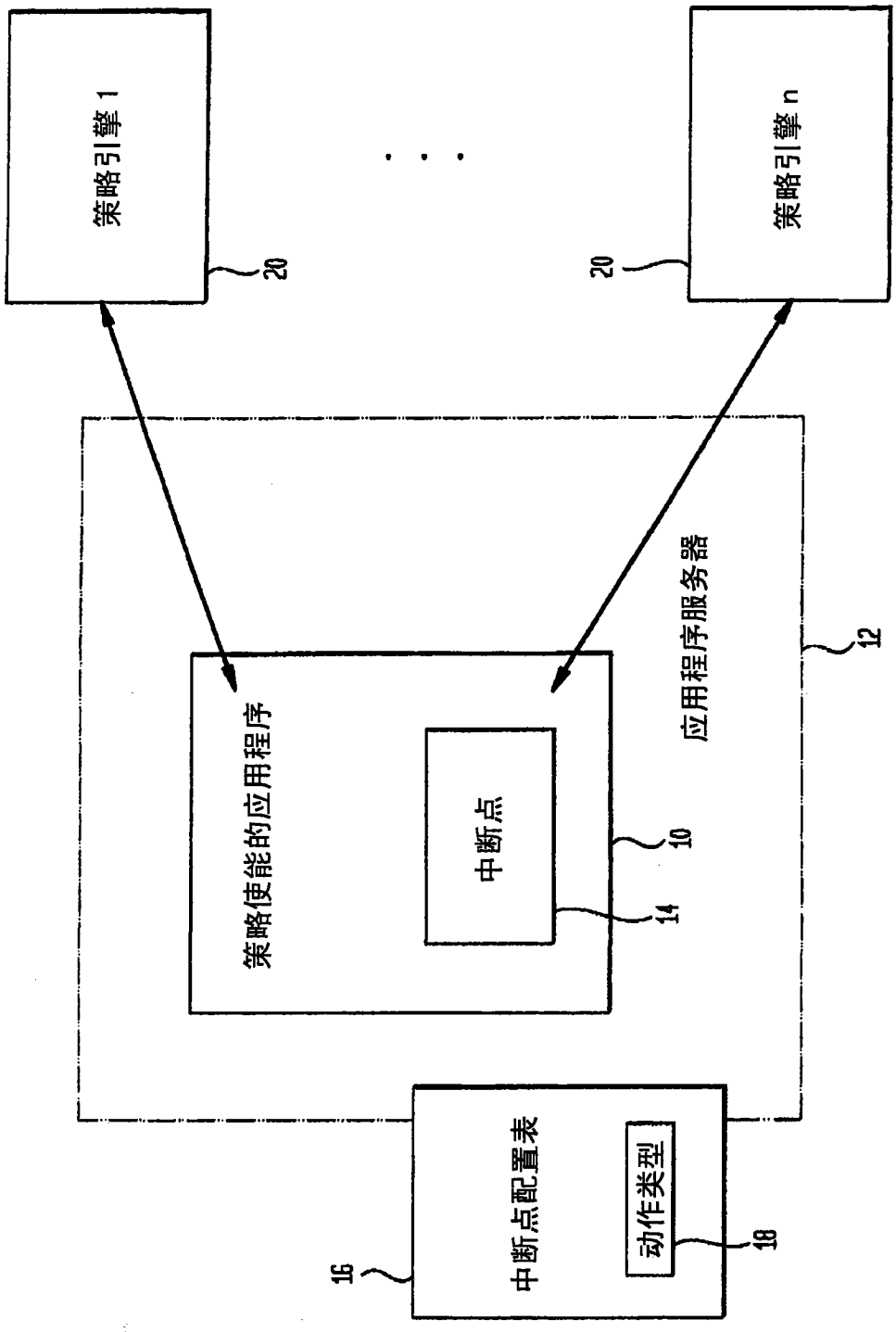

策略引擎 1
20
策略引擎 n
20
应用程序服务器
12
策略使能的应用程序
中断点
10
14
中断点配置表
动作类型
16
18

图 1

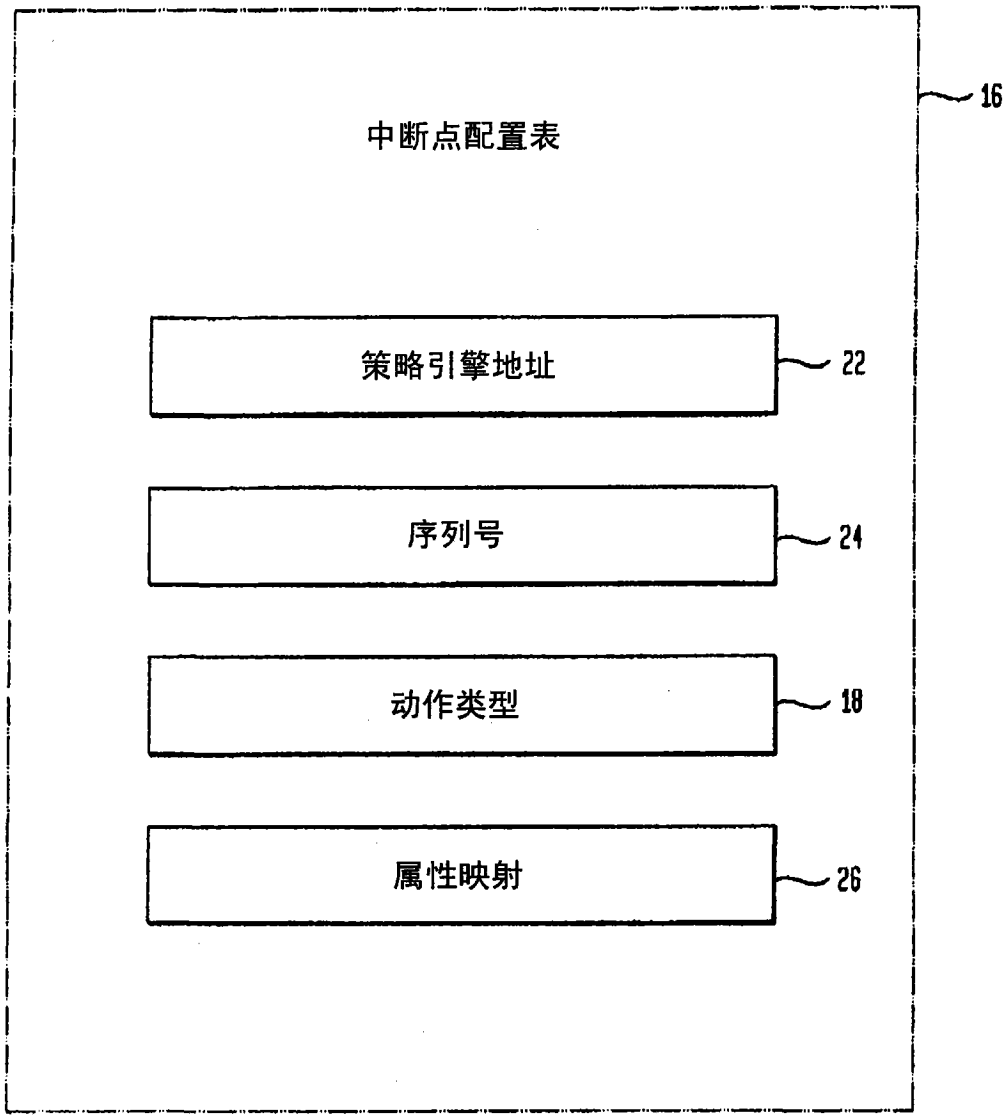


图 2

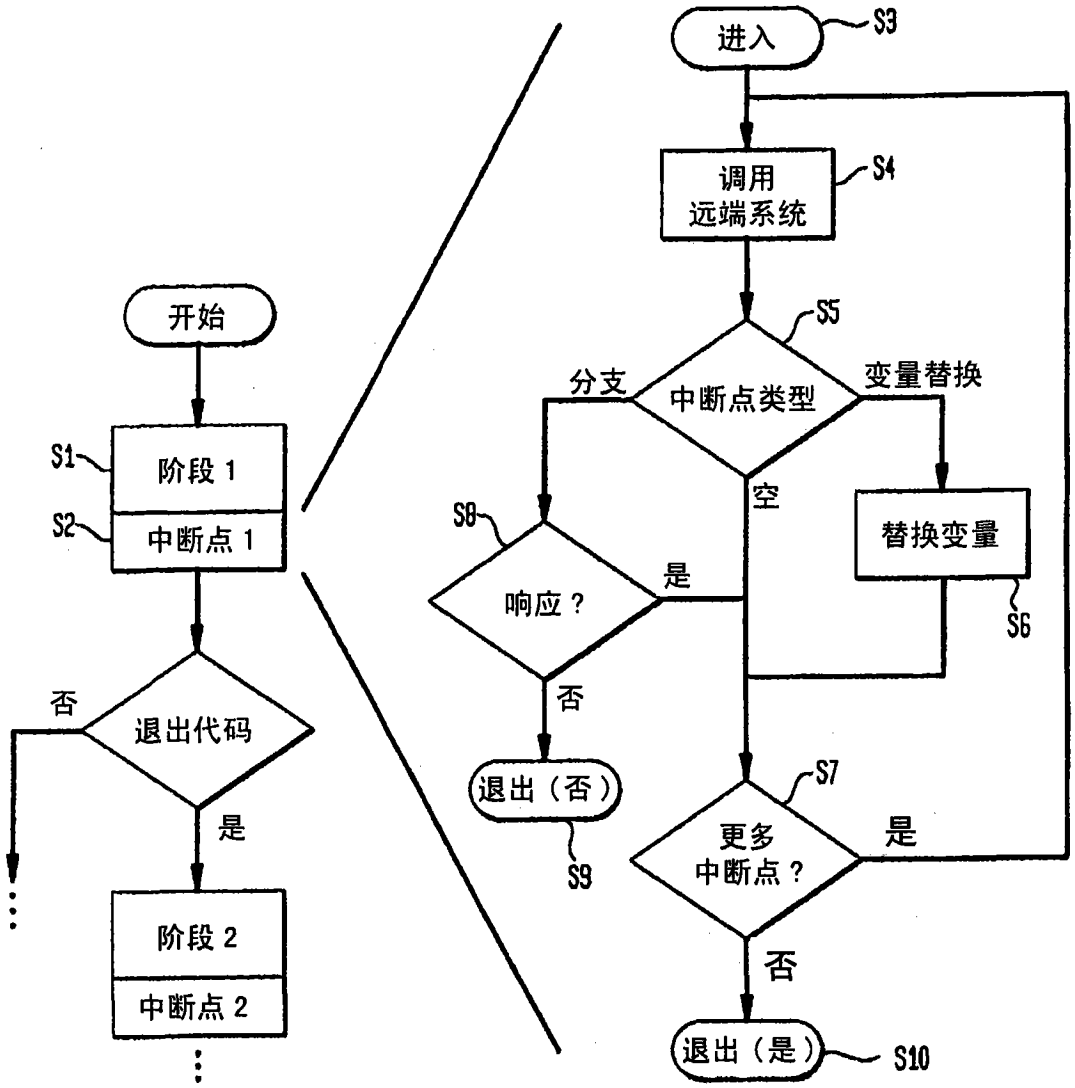

开始
S1
阶段 1
S2
中断点 1
否
退出代码
是
阶段 2
中断点 2
进入
S3
调用
远端系统
S4
S5
分支
中断点类型
变量替换
空
S8
响应？
是
否
退出（否）
S9
替换变量
S6
S7
更多
中断点？
是
否
退出（是）
S10

图 3