(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 101410824 B
(45)授权公告日 2012.03.21

(21)申请号 200480012009.9

(22)申请日 2004.07.30

(30)优先权数据
60/567,165 2004.04.30 US
60/567,149 2004.04.30 US
60/567,153 2004.04.30 US
10/903,765 2004.07.30 US

(85)PCT申请进入国家阶段日
2005.11.03

(86)PCT申请的申请数据
PCT/US2004/025060 2004.07.30

(87)PCT申请的公布数据
WO2005/111851 EN 2005.11.24

(73)专利权人 微软公司
地址 美国华盛顿州

(72)发明人 P·瑟沙德瑞 H·奈特 R·H·格伯 S·E·多西科 V·H·柯利

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张政权

(51)Int.Cl.
G06F 17/00(2006.01)
G06F 7/00(2006.01)
G06N 5/02(2006.01)
G06F 9/44(2006.01)

(56)对比文件
US 5917489 A,1999.06.29, 说明书第9栏第15行至第16栏第6行、说明书附图1.
US 5495603 A,1996.02.27, 说明书第3栏第61行至第9栏第17行、说明书附图1-2.
US 2002/0049715 A1,2002.04.25, 全文.
US 6341369 B1,2002.01.22, 全文.

审查员 胡雅娟

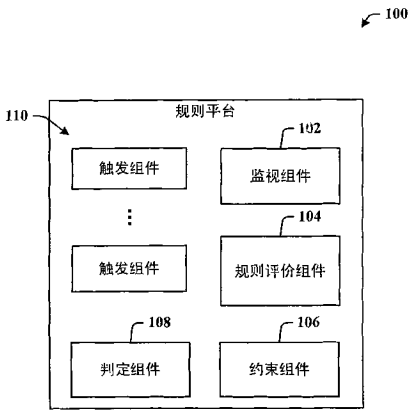
权利要求书 3 页 说明书 35 页 附图 19 页

(54)发明名称

用于终端用户规则逻辑的定义和执行的规则框架

(57)摘要

一种基于规则的软件架构，它为终端用户规则逻辑的定义和执行提供了基础结构。这使得使用简单的IF-THEN规则，在统一的存储平台内的数据能够终端用户自动化操作。这种架构包括跟踪与该数据相关联的项的监视组件，以及与监视组件接口使用与被跟踪的项相关的元数据以提供这些项的子集的自动化处理的规则组件。该系统进一步提供使用基于内容的逻辑定义系统内的虚拟集合和项。该系统进一步包括一个或多个触发器组件，按照触发器逻辑动态地将项和项集合设置为活动的。额外的组件可包括用于向项施加约束的约束组件，以及支持在判定点启动应用程序客户化的判定组件。



CN 101410824 B

1. 一种方便了数据管理的系统，其特征在于，包括：

监视组件，跟踪与数据相关的项；和

规则评价组件，使用与被跟踪的项相关的元数据，以方便项的子集的自动处理；

一个或多个触发器组件，按照触发器逻辑将项和项的集合动态地设定为活动的；

约束组件，在项内施加约束逻辑；

判定组件，标识数据内的一个或多个判定点并在一个或多个判定点处启动应用程序客户化逻辑。

2. 如权利要求 1 所述的系统，其特征在于，进一步包括用于定义系统中的虚拟集合和项的基于内容的逻辑。

3. 如权利要求 1 所述的系统，其特征在于，所述规则评价组件方便了自动化操作数据的规则的输出。

4. 如权利要求 1 所述的系统，其特征在于，所述规则评价组件方便了包括自动导出的条件的规则的输出。

5. 如权利要求 1 所述的系统，其特征在于，所述规则评价组件方便了包括明确指定的条件的规则的输出。

6. 如权利要求 1 所述的系统，其特征在于，所述规则评价组件方便了包括作为自变量的终端用户输入的规则的输出。

7. 如权利要求 3-6 中任一权利要求所述的系统，其特征在于，进一步包括解决组件，它解决所述规则间的冲突。

8. 如权利要求 7 所述的系统，其特征在于，解决组件通过终端用户规则优先级和预先定义的解决策略中的至少一个来解决规则冲突。

9. 一种用于管理文件系统的基于规则的系统，其特征在于，包含：

监视组件，使用终端用户创建的一个或多个规则来跟踪与文件系统的数据相关的项；和

规则组件，处理与被跟踪的项相关的、方便了数据的自动处理的规则；

一个或多个触发器组件，按照触发器逻辑将项和项的集合动态地设定为活动的；

约束组件，在项内施加约束逻辑；

判定组件，标识数据内的一个或多个判定点并在一个或多个判定点处启动应用程序客户化逻辑。

10. 如权利要求 9 所述的系统，其特征在于，所述终端用户使用规则从其他项定义虚拟集合和项。

11. 如权利要求 9 所述的系统，其特征在于，其中所述一个或多个触发器组件是同步或异步的。

12. 如权利要求 9 所述的系统，其特征在于，所述规则包括自动导出的条件和明确指定的条件。

13. 如权利要求 9 所述的系统，其特征在于，所述规则包括作为自变量的终端用户输入。

14. 如权利要求 9 所述的系统，其特征在于，进一步包含解决组件，它通过使用终端用户规则优先级和预先定义的解决策略中的至少一个来解决多个规则间的冲突。

15. 一种方便了数据管理的方法，该方法包括：

跟踪与该数据相关的项；以及

提供与被跟踪的项相关的元数据以方便项的子集的自动处理；

按照触发器逻辑将项和项的集合动态地设定为活动的；

在项内施加约束逻辑；

标识数据内的一个或多个判定点；以及

在一个或多个判定点处启动应用程序客户化逻辑。

16. 如权利要求 15 所述的方法，其特征在于，进一步包含使用基于内容的逻辑来定义系统内的虚拟集合和项。

17. 如权利要求 15 所述的方法，其特征在于，进一步包含输出方便数据的自动化操作的一个或多个规则。

18. 如权利要求 15 所述的方法，其特征在于，进一步包含输出包括自动导出的条件和明确规定的条件的规则。

19. 如权利要求 15 所述的方法，其特征在于，进一步包含输出包括作为自变量的终端用户的输入的规则。

20. 一种用于管理文件系统的基于规则的方法，其特征在于，包含：

使用终端用户创建的一个或多个规则来跟踪与文件系统的数据相关的项；以及

处理与被跟踪的项相关的、方便了数据的自动处理的规则；

按照触发器逻辑将项和项的集合动态地设定为活动的；

在项内施加约束逻辑；

标识数据内的一个或多个判定点；以及

在一个或多个判定点处启动应用程序客户化逻辑。

21. 如权利要求 20 所述的方法，其特征在于，进一步包含通过终端用户规则优先级和预先定义的解决策略中的至少一个来解决规则间的冲突。

22. 如权利要求 20 所述的方法，其特征在于，进一步包含使用基于内容的逻辑来定义虚拟集合和项。

23. 如权利要求 20 所述的方法，其特征在于，进一步包括：定义包含一个或多个规则的逻辑，所述逻辑被用于组成方便数据的自动处理的终端用户程序。

24. 如权利要求 23 所述的方法，其特征在于，所述逻辑用于使用 IF-THEN 规则构造，定义询问、复杂动词和约束中的至少一个。

25. 如权利要求 23 所述的方法，其特征在于，所述逻辑方便了跨多个应用程序的规则逻辑的公共视觉显示。

26. 如权利要求 20 所述的方法，其特征在于，进一步包含以下动作的至少一个：

用规则定义值以生成属性；

将规则与事件结合以创建代理；

过滤列表以生成新的列表；以及

将新的列表与另一个列表相组合。

27. 如权利要求 20 所述的方法，其特征在于，进一步包括：定义逻辑以允许终端用户执行以下至少一种动作：

定义与现有类型不同的新的终端用户类型；

动态地修改现有的类型；

定义并持续询问；以及

定义影响特定项的逻辑。

28. 如权利要求 20 所述的方法，其特征在于，进一步包含持续第一应用程序中作出的跨越另一个应用程序的基于规则的变化。

用于终端用户规则逻辑的定义和执行的规则框架

[0001] 有关申请的交叉参考

[0002] 本申请要求以下在先申请的优先权：于2004年7月30日提交的美国专利非临时申请号（未知），名称为“RULES FRAMEWORK FOR DEFINITION ANDEXECUTION OF END-USER RULES LOGIC”；于2004年4月30日提交的美国专利临时申请第60/567,165号，名称为“RULES FRAMEWORK FORDEFINITION AND EXECUTION OF END-USER RULES LOGIC”；于2004年4月30日提交的美国专利临时申请第60/567,149号，名称为“DERIVED SET-ARULES-BASED QUERY LIKE MECHANISM THAT DEFINES CONTENTS OF ACOLLECTION”；以及于2004年4月30日提交的美国专利临时申请第60/567,153号，名称为“END-USER APPLICATION CUSTOMIZATION USING RULES”。本申请也涉及共同待批的于2004年7月30日提交的美国专利申请号（未知）（代理人档案号MSFTP669USA），名称为“END-USER APPLICATIONCUSTOMIZATION USING RULES”。这些专利申请整体被引用在此作为参考。

技术领域

[0003] 本发明涉及基于规则的软件架构，它方便了数据的终端用户自动化。

背景技术

[0004] 计算机和计算已经将世界上的用户分为两类：有知识的“高手”，他们知道怎样以复杂的方式使用计算机，以构成程序并赋予有价值的和丰富的行为；以及新手，他们是那样无助，以致于不能轻易或便宜地访问知识或信息或教育，以使计算机很好的服务于他们的需要。然而，当技术已经打破了这些访问障碍时，计算方面的主要突破已经出现了。

[0005] 在大型机的世界里，除了可以担负得起的最大的公司以外，计算机对所有人来说都太昂贵了。小型计算机的出现，以及后来的个人计算机(PC)，打破了费用壁垒，并使计算机对小型商业和个人来说是可用的。

[0006] 20世纪80年代，程序员努力构建图形用户界面(GUI)应用程序，没有丰富、一致的GUI，他们不能为PC用户构建有价值的应用程序。Visual Basic的革命及对于控件和基于事件的GUI结构的使用，使得应用程序开发者的整个大军能容易地构建丰富的应用程序。这也建立了具有能使用这些应用程序的更多终端用户的良好的循环。

[0007] 20世纪90年代，终端用户努力克服信息访问的缺乏。因特网和网络的增长改变了这个空白，使得任何具有浏览器的人都可以访问几乎所有有价值的信息。然而，仍然存在需要克服的重大障碍。

[0008] 计算不是个人的。关于PC，只有极少是真正“个人的”。本地盘上的数据是个人的。但是机器的行为（代表用户所做的）对数百万用户几乎是相同的。尽管拥有令人称奇的功能强大的通用计算机，一般用户还是把它当作静态的工具，作为通信端点使用，作为搜索入口点使用，作为执行一些预录制的大量销售的应用程序来使用，但却不能是这个单词的真正意义上的任何“个人计算”。当前的应用程序中可用的个性化能力仅仅是其可能的和想要的皮毛而已。

[0009] 计算是手动的。考虑大多数典型的计算机终端用户的日常例程。PC 收集信息、对通信做出反应、作决定以及根据决定行动—启动通信或对其作出反应、组织信息、买卖物品、旅游等等。计算机已经改善了人们之间的通信，而且改善了对信息的访问。然而，PC 对减轻终端用户适时作决定并根据决定行动的责任方面，却做的很少。在商业世界中，存在用于主要的有组织的决定的决定支持系统。在许多日常的、但重要的和个人的决定中，软件仍不能帮助一般的 PC 用户。

[0010] 计算与上下文无关。计算机软件一般提供可选设定，它们是相当静态并与用户的实际上下文无关（例如，“为何我在工作时和在家时都要有关于通信伙伴的相同设定”）。

[0011] 因此，由于日益陷入手动信息处理的苛刻环境，即每天花费大量的时间用于筛选、分类、搜索和对电子邮件、文档和其他个人数据作出反应，用户仍然处于软件的“前工业时代”。

[0012] 终端用户软件应该是个性化的，知道终端用户的需要和偏好，并以由那些需要和用户上下文的所引导的方式来行动。而且，计算机系统和软件应该给每个终端用户提供个人执行助理，它每天 24 小时工作来收集并筛选终端用户感兴趣的信息，并对这些信息作出反应。

[0013] 最有价值的一类终端用户的计算活动涉及信息流和搜索，如确保终端用户看到相关的信息（例如，“告诉我是否因为天气不好，学校关闭了）、增强人与人之间的具有个性化行为的通信（例如，“如果当我妻子呼叫我时我不在办公室，让她知道我何时回来”）、确保重要的信息不会丢失（例如，“如果有紧急的电子邮件，确保转寄到我的移动设备上”）、以及信息管理自动化（例如，“当新的照片到达时，将它们放到正确的文件夹并基于其时间戳、GPS 位置和任何相关的日历条目来共享它们”）。

[0014] 完成这个目的的方法是通过允许终端用户“编程”其计算机的行为。然而，传统的编程语言很明显不是答案，因为，终端用户并非（且不可能成为）训练有素的开发者。

[0015] 因此，对于改善的编程平台有一种从未满足的需求，该平台方便了这种手动处理的自动化，因此允许用户使计算机和软件个性化，以代表他或她来自动行动。

发明内容

[0016] 下面为了提供本发明的一些方面的基本理解，给出了本发明的简要概述。这个概述不是本发明的广延综述。它并不意图标识出本发明的关键 / 重要元素或者描绘本发明的范围。其唯一的目的是以简化的形式显示给出本发明的一些概念，作为此后给出的详细描述的序言。

[0017] 本发明的焦点在于数据和信息、存储中的和通信中的中心值，并且在于一般是无知的非开发者的终端用户（消费者、有知识的工作者和商业用户），虽然如此，他们希望使他们的 PC 和应用程序以定制的和自动化的方式工作。

[0018] 在此揭示并要求保护的本发明，在其中的一方面中，包含基于规则的软件架构，它为自动化操作数据的终端用户定义的规则逻辑的定义和执行提供了基础结构。

[0019] 规则创建框架包括项和关系，它们在创建规则时以及构建使用规则的应用程序时被使用。这些项和关系由规则框架的规则引擎用于确定响应于特定的事件，计算机应该做什么。为了定制基于终端用户规则的应用程序行为，程序员“在规则上允许”(rule enable)

软件的功能，终端用户为这些功能创建规则。

[0020] 本发明允许终端用户定义逻辑，该逻辑至少以以下四种方法来“自动化处理”终端用户数据：数据导出、触发激活、数据约束和判定点。数据导出规则依附于输入项范围。触发激活规则依附于集合内改变的项。数据约束规则依附于集合内改变的项。数据驱动的判定规则显然由应用程序应用到项。统一的规则存储平台将所有有知识的工作者应用程序集合在公共存储内的系统化数据的主题周围。

[0021] 规则是用于复杂逻辑的单个说明性的 IF-THEN 编程构造。每个终端用户逻辑组件被定义为规则。所有其他的构造是简单的二进制代数运算符。规则在特定项类型的输入自变量上定义了说明性的无副作用的功能。

[0022] 规则创建框架使用项类型，以支持规则创建：判定点项、规则项、以及规则集附件项。表示为 RuleSetAttachment，它表示判定点（表示为 DecisionPoint）和规则（表示为 Rule）的项之间的连接。可以把这些连接建模成物理的、存储的“链接”或计算出的“公值联合”。在任何一种情况下，其功能是相同的，即从规则到为其创建该规则的判定点的连接。应用程序的每个规则允许的特征由判定点项来定义。终端用户为特定的判定点创建规则，并且应用程序使用判定点对象向规则引擎提交数据。每个条件 / 结果语句（statement）是一个规则项。规则是用于执行的逻辑单元。规则可以是多重语句的。用户创建规则。用户可以控制规则组织，但应用程序或显示给用户的视觉浏览 metaphor（比喻）也可以控制规则组织。应用程序也可以创建规则。规则通过规则集附件（RSA）项与判定点相关联。规则引擎使用该附件来确定为给定的输入应该执行哪个规则。

[0023] 规则平台评价使用数据库询问处理器的终端用户规则。这会导致有效的评价，同时语义仍与平台的剩余部分保持一致。

[0024] 既然终端用户一般不是训练有素的开发者，那么期望他们用传统编程语言来编程就是不合情理的。相反，提供了系统化的逻辑构件块，因此终端用户可以通过以简单而丰富的组合将它们结合在一起来进行编程。来自现有的应用程序的根据经验的证据表明，终端用户可以使用 IF-THEN 规则轻松地定义判定逻辑。因此，本发明的基于规则的平台能够使用简单的 IF-THEN 规则，在统一的存储平台内使终端用户自动化操作数据成为可能。这进一步支持使用询问包括布尔（Boolean）条件的询问构造器的大量应用程序。

[0025] 为了实现前述内容以及相关的目的，在此结合以下叙述和附图描述了本发明某些说明性的方面。然而，这些方面仅仅指示出本发明的原则可在其中应用的各种方式中的的一些，本发明意欲包括所有这样的方面及其等价物。当结合附图考虑时，从以下对本发明的详细描述中，本发明的其他优点和新颖的特征会变得更为明显。

附图说明

[0026] 图 1 是示出依照本发明的规则架构，方便终端用户数据自动化操作和管理的系统。

[0027] 图 2 是示出依照本发明的规则架构的一种系统化的方法。

[0028] 图 3 示出依照本发明的规则架构的一种数据导出的方法。

[0029] 图 4 是示出依照本发明的规则架构的一种副作用逻辑的触发激活的方法。

[0030] 图 5 示出依照本发明的规则架构的一种数据约束的方法。

[0031] 图 6 示出依照本发明的规则架构，在判定点提供应用程序客户化的方法。

[0032] 图 7 示出本发明的规则架构的项类型及其关系的示意图。

[0033] 图 8 示出依照本发明的规则架构的条件和结果处理的示意图。

[0034] 图 9A 示出依照本发明的规则的一般格式。

[0035] 图 9B 示出依照本发明，用于活动规则 (active rule) 的一般规则格式。

[0036] 图 9C 示出依照本发明，用于应用程序客户化规则的一般规则格式。

[0037] 图 9D 示出依照本发明，用于数据导出规则的一般规则格式。

[0038] 图 9E 示出依照本发明，用于数据约束规则的一般规则格式。

[0039] 图 10 示出依照本发明的规则架构，通过将规则依附于输入项的集合来应用规则的一种方法。

[0040] 图 11 示出依照本发明的规则架构，通过将规则依附于判定点来应用规则的另一种方法。

[0041] 图 12 示出依照本发明的规则架构，通过直接调用规则来应用规则的另一种方法。

[0042] 图 13 示出依照本发明的规则架构来应用规则的另一种方法。

[0043] 图 14 示出依照本发明，用于规则冲突解决的一种方法。

[0044] 图 15 示出依照本发明的规则架构的项类型以及相关关系。

[0045] 图 16A 显示本发明的 EvaluationResultElement 嵌套类型。

[0046] 图 16B 显示本发明的 InputScope 嵌套类型及其派生。

[0047] 图 16C 显示本发明的 LogicConstraint 嵌套类型。LogicConstraint 被 RuleLogic（和其子类）和 DecisionPoints 所使用。

[0048] 图 16D 显示本发明的 LogicStatement 嵌套类型。LogicStatements 对条件树和一组结果（动作或布尔逻辑 (Boolean)）编码。

[0049] 图 16E 显示本发明的 LogicResult 嵌套类型及其派生。

[0050] 图 16F 显示本发明的 Condition 嵌套类型及其派生。

[0051] 图 16G 显示本发明的 ArgumentValue 嵌套类型及其派生。

[0052] 图 16H 示出本发明的 FunctionInfo 嵌套类型及其派生。

[0053] 图 17 示出依照本发明的规则引擎。

[0054] 图 18 示出代表本发明的规则架构中的丰富程度的尺度 (dimensions) 的框图。

[0055] 图 19 示出表示本发明的规则架构的输入的示意图。

[0056] 图 20 示出可以执行揭示的架构的计算机的框图。

[0057] 图 21 示出依照本发明的一个示例性的计算环境的示意性的框图。

具体实施方式

[0058] 现在参考附图描述本发明，其中始终使用相同的符号指代相同元素。下面的描述中，为了说明的目的，提出了很多特定的细节以便提供对本发明的全面理解。然而，明显的是，没有这些特定的细节，本发明也是可以实施的。在其它实例中，众所周知的结构和装置以框图形式示出，以便方便描述本发明。

[0059] 作为在本发明中所使用的，术语“组件”和“系统”意指计算机相关的实体，它们是硬件、硬件和软件的结合、软件或者执行中的软件。例如，组件可以是但不限于：在处理器上

运行的进程、处理器、对象、可执行的、执行线程、程序、和 / 或计算机。为了说明，服务器上运行的应用程序和服务器都可以是组件。一个或多个组件可以驻留在进程和 / 或执行线程中，且组件可以位于计算机中和 / 或分布于两台或多台计算机之间。

[0060] 规则平台

[0061] 文件系统是终端用户数据的仓库。本发明允许终端用户定义以至少以下四种方式“自动化操作”终端用户的数据的逻辑：数据导出、触发激活、数据约束和判定点。统一的规则存储平台将所有知识工作者应用程序集合在公共存储内的系统化数据的主题周围。它是关于发现、组织数据和对数据起反应。丰富的询问和丰富的规则是事务的两方面。当发现数据时，丰富的询问和视图带给终端用户增加的控制。丰富的规则提供了增加的数据组织的自动化操作，还有可以在系统中构建主动 / 反应行为的能力。这些是当今的 PC 用户正开始对个体应用程序的期望的能力。一个特征是跨所有数据和应用程序使这些能力标准化。

[0062] 本发明的基于规则的架构方便了系统化、数据共享和信息管理。数据的普遍存在的公共系统化是终端用户的“程序”开发的关键。共享数据对终端用户程序来说是普通和自然的。当决定对到达的电子邮件消息进行优先次序排序时，为了作决定，很自然地要将当天的日历条目、参与的人和组、用户的当前位置（由不同的应用程序创建的所有系统化的数据）集合在一起。揭示的架构能够跨应用程序共享数据。本发明的一个重要方面是能够通过组织（文件夹 / 列表）和关系来管理信息过载的能力。在大多数信息过载的情况下，这个问题不仅是具有充足的文件夹和分类，而且是要有手动组织 / 分类这些数据的时间。如果系统使对大量的组织构造自动化，那么它们对终端用户才变得真正有价值。

[0063] 规则创建框架由项和关系组成，它们在创建规则时以及构建使用规则的应用程序时被使用。规则平台的规则引擎使用这些项和关系来确定响应于特定的事件，计算机应做些什么。

[0064] 为了基于终端用户规则来定制应用程序的行为，发生两件事情。程序员“在规则上允许”软件的功能，而终端用户为那些功能创建规则。例如，规则允许的操作系统特征被包括在内，以便用户可以在如文件夹和文档的项上定义规则。这支持终端用户编写表示他们希望的来自其计算机的行为的规则“程序”。而且，这个行为应该随他们的移动而移动。因为揭示的规则架构将规则建模成数据，所以该数据可以被同步和移动。

[0065] 规则是用于复杂逻辑的单个说明性的编程构造。每个终端用户逻辑组件被定义为规则。所有其他的构造是简单的二进制代数运算符。规则在特定的项类型的输入自变量上定义说明性的无副作用的功能。规则有两部分，即特征标（它定义输入项类型和输出类型）和定义。一般来说，输出类型是简单的标量类型、项类型或集合类型（反映一组结果的期望值）。

[0066] 规则在逻辑上表示为在单个输入上的动作，尽管在评价期间，它经常被用于一组实际输入。作为连续的例子，考虑针对 Message 项类型定义的规则。Message 类型是关系类型 Recipient 的源，其目标是 Contact 项类型。它捕捉邮件消息的“To”字段内的信息。为了说明这个例子，使用类似英语的伪程序设计语言来描述规则。这不应被误解为表示实际的规则语言，相反，规则是通过使用 API 构造的代码以及使用具有有意义的用户友好描述的友好的 UI 操作的终端用户来表示的。

[0067] 规则定义是零个或更多个 IF-THEN 语句的有序列表。规则内的每个 IF-THEN 语句

具有条件表达式(IF 部分)和结果表达式(THEN 部分)。条件表达式是布尔表达式树,其叶子是终端用户能够推论的基本条件运算符。结果是一列常量表达式或终端用户可以推论的简单表达式。

[0068] 具有多重 IF-THEN 语句的规则的语义根据为某一输入项的输出而定义。如果没有其条件在输入项上评价为真的语句,则结果为 NULL。如果恰好有一个其条件评价为真的语句,则由该语句的结果来定义输出。如果存在多个其条件评价为真的语句,则结果依赖于对所有这些语句的结果应用冲突解决策略。普通的冲突解决策略是基于规则中的语句的顺序,即最高优先级的取得胜利。

[0069] 规则创建框架使用项类型来支持规则创建:判定点项、规则项和规则集附件项(也表示为 RuleSetAttachment 或 RSA)。应用程序的每个规则允许的特征由判定点项来定义。终端用户为特定的判定点创建规则,且应用程序使用判定点对象向规则引擎提交数据。每个条件/结果语句是规则语句项。规则是用于执行的逻辑单元。应用程序也能创建规则。应用程序能提供默认规则作为起始点。这个默认规则将被用作判定点,直到终端用户选择修改或提供替代规则为止。规则通过 RSA 项与判定点相关联。由规则引擎使用该附件来确定为给定的输入应该执行哪些规则。

[0070] 可能定义加入/结合其他数据集合的条件。而且,可能指定包括可用于其他规则的虚拟集合的组件化的(componentized)逻辑。

[0071] 对于异步行为规则,其结果描述了关于文件系统类型、任何 CLR 类型的方法或任何静态方法的动作。对同步行为规则,其结果是应用程序特定的类型的一组值,例如,文件名、字符串、要评价的对象、颜色枚举列中的一个颜色。

[0072] 规则平台包括标准的丰富的规则用户界面(UI)。所遵循的原则如下:存在公共 UI 用于规则和视图/询问;整个创作过程可发生在单个屏幕内;支持模板的统一抽象;值选择能支持丰富的类型特定的控件(例如,日期挑拣器);以及 UI 可以扩展,即,当增加新的项类型时,UI 可以自动为它们创作规则并包含它们的条件和动作。对操作系统规则来说,用户通过带有视觉浏览 metaphor 的 UI 动作将规则依附于判定点。然而,其他应用程序可以为用户创建附件,因为该应用程序为该用户定义了判定点。

[0073] 现在参考图 1,示出依照本发明的规则架构,方便了终端用户数据自动化操作和管理的系统 100。该规则架构允许终端用户用以下四种方式定义“自动化操作数据”的逻辑。第一,数据导出方便了使用基于内容的逻辑从系统中的其他项定义“虚拟”集合和项。例如,定义一组叫做 PeopleWhoLiveInBellevue 的联系人,它是地址在 Bellevue 中的联系人的全集的子集。第二,副作用逻辑的触发激活方便了通过将集合和项与触发逻辑相关联而使集合和项处于活动状态。例如,通过说明任何正被添加到规范(specification)集合的新文档应能导致它们的创作者被电子邮件告知。第三,数据约束方便了在项上施加约束逻辑。例如,通过要求 Indigo 文件夹内的任何电子邮件的主题内有“Indigo”或已经被发送到 Indigo 邮件发送列表。最后,数据驱动的判定方便了在应用程序拦截器(interceptor)点支持丰富的应用程序客户化逻辑。例如,通过基于消息的内容和目的地,指定哪个签名文件应被附加到输出的电子邮件消息中。

[0074] 作为其支持,系统 100 包括监视组件 102,它跟踪与数据相关的项。规则评价组件 104 使用与被跟踪的项相关的元数据来提供对项的子集的自动处理。该系统也方便了使用

基于内容的逻辑在系统中定义虚拟集合和项。该系统还包括一个或多个触发组件106，该组件按照触发逻辑的功能将项和项的集合动态地设定为活动。其它组件可包括用于在项上强加约束的约束组件106，以及支持在判定点允许应用程序客户化逻辑的判定组件108。判定组件108不仅代表可由应用程序暴露的一个或多个判定点，而且代表提交事件或判定点的API。

[0075] 现在参考图2，示出了依照本发明的规则架构的一种系统化的方法。虽然，为了简化说明起见，这里显示的一种或多种方法（例如，以流程图的形式）是按照一系列动作显示并描述的，但是应该理解和认识到，本发明并不限于这些动作的顺序，除了依照本发明在这里显示和描述的顺序之外，一些动作可以不同的顺序出现和/或与其他动作同时发生。例如，本领域的技术人员应该理解并认识到，方法可被替代地表示为一系列相关的状态或事件，如在状态图中。而且，并非所有例示出的动作都是实现依照本发明的方法所必须的。

[0076] 系统化被定义为将数据和逻辑结构化成众所周知的和已定义的模式，这些模式使多个应用程序能够识别数据和逻辑并与之交互。数据的普遍存在的一般系统化是终端用户“程序”开发的核心。至少三种“系统化”允许依照本发明的终端用户数据自动化操作。在200，接收数据和逻辑用于系统化以方便终端用户数据自动化操作。在202，数据被系统化。系统化的信息是数据，该数据是终端用户应用程序（例如，电子邮件、人、组和位置）的基础。这允许应用程序对数据的一致性解释。在204，为逻辑构件块将事件系统化。系统化的信息事件是提供挂钩来附加逻辑的事件。这些事件是高级的并依靠对终端用户有意义的信息流。例如，电子邮件消息的到达就是这种事件。事件可以是应用程序内部的拦截器点，或者是可绑定逻辑的异步事件。在206，逻辑构件块被系统化以通过规则处理数据（或信息）和事件。既然终端用户一般不是训练有素的开发者，那么期望他们用传统的编程语言来编程就是不合情理的。相反，提供了系统化的逻辑构件块，因此终端用户可以通过简单却丰富的组合将它们组合在一起来进行编程。

[0077] 现在参考图3，示出了依照本发明的规则架构的数据导出的一种方法。在300，接收数据用于处理。在302，使用规则架构的一个或多个规则，通过使用基于内容的逻辑从系统内的其他项定义虚拟集合和项，来自动化操作数据。例如，可通过定义叫做PeopleWhoLiveInCity的一组联系人，它是地址在该城市中的Contacts（联系人）的全集的子集，来创建虚拟集合。

[0078] 现在参考图4，示出了依照本发明的规则架构的副作用逻辑的触发激活的一种方法。在400，接收数据用于处理。在402，通过将触发事件的活动与触发逻辑相关联，来使集合和项处于活动状态。例如，说明正被添加到Specification（规范）集合的任何新文档可导致它们的创作者被电子邮件告知。

[0079] 现在参考图5，示出了依照本发明的规则架构的数据约束的一种方法。在500，接收数据用于处理。在502，在项上施加约束逻辑。数据约束的一个例子是，要求Name（名称）文件夹内的任何电子邮件在主题内具有“Name”或者已经被发送到Name邮件发送列表。

[0080] 现在参考图6，示出了依照本发明的规则架构，在判定点提供应用程序客户化的一种方法。在600，接收数据用于在判定点处理。在应用程序拦截器（或判定）点提供丰富的客户化逻辑。因此，在602，处理依附于判定点的规则。数据驱动的判定点的一个例子是，基于消息的内容和目的地，指定哪个签名文件应被附加到输出的邮件消息中。在604，返回应

用程序判定。

[0081] 现在参考图 7，示出了依照本发明的规则架构的项类型及其关系的示意图。输入范围是任何类型的任何项。操作系统使用输入范围来将规则评价的范围限制到特定的项或文件夹，但个体应用程序一般不使用该输入范围。上图中的线上的标记显示了项之间的关系的名称。RuleSetAttachment（规则集附件）项表示 DecisionPoint（判定点）和 Rule（规则）项之间的连接。这些连接可被模拟为物理的、存储的“链接”，或是作为计算出的“公值联合”。另一种情况下，其功能是相同的，即从 Rule（规则）到为其创建该规则的 DecisionPoint（判定点）的连接。判定点项使应用程序能使用规则平台。规则描述规则项，包括约束、条件和结果。规则包含规则语句。规则被依附于判定点，且应用程序提供输入给判定点以返回结果。

[0082] 现在参考图 8，示出了依照本发明的规则架构的条件和结果处理的示意图。该规则平台为终端用户规则逻辑的定义和执行提供了基础结构。该规则模式提供了与项的类型相关的参数化的条件和结果。例如，EmailMessage（电子邮件消息）项类型可能有与其相关联的条件 IsFrom（来自）、IsTo（去往）等等。为该规则平台提供了完整的一组项类型 800，使用它们就方便了终端用户依照本发明进行编程。项类型 800 表示为 ITEM $TYPE_1$：$CONDITION_1$/$RESULT_1$、ITEM $TYPE_2$：$CONDITION_2$/$RESULT_2$、.....、ITEM $TYPE_N$：$CONDITION_N$/$RESULT_N$。作为输入 802，自动从项类型中导出一些较为简单的条件，而其他条件 804 可由开发者明确指定。用户向项类型 800 输入自变量值 806。所选择的项类型 800 的这些条件和结果成为用于终端用户逻辑的输出“指令集”808。规则中的条件和结果是开发者定义的条件和结果的实例，自变量值由终端用户指定。每个规则存留为项，具有项的全部共享、安全和系统化能力。

[0083] 终端用户逻辑“程序”，即输出指令集 808，是一组规则，它是规则语句的集合。一个完整的逻辑单元包括一个或多个规则。每个规则是一创作单元。注意，输入到规则的是数据项。规则是有关某一项类型的项的说明性语句。作为基本模型的扩展，可以提供非项数据（过渡过程数据或 XML）作为规则输入。规则应用于其上的项依赖于规则的部署。规则是可部署的终端用户逻辑的单元。通过将规则依附于项范围（item scope）（项输入或判定点的源）来部署该规则。这种联合以 RuleSetAttachment（规则集附件）来记录。规则和 RuleSetAttachment(RSA) 可以是项（如前所述，它可以是项之间有名的公值关系）。

[0084] 在活动规则 (active rule) 的情况下，项范围是项集合。当使用规则来截取并客户化应用程序时，该应用程序提供对其应用了规则的项。

[0085] 规则共同定义了终端用户客户化请求。规则内的每个规则语句对应于终端用户意图的一个语句。如果可使用多个规则，可在动作上使用冲突裁决器。

[0086] 现在参考图 9A-E，示出了依照本发明的规则结构的规则的代表性格式。从终端用户的角度，Rules == Views == Queries（规则==视图==询问）。规则平台至少遵循这些原则：规则是说明性的 IF-THEN 表达式；用于规则的数据模型是数据模型，并且因此任何项类型可以参与规则；以及，在数据库询问处理器内针对该数据模型来评价规则。

[0087] 视图是持续的询问，但却在表达力方面不同于询问。实际上，视图和规则共享对持续性的公共要求。

[0088] 尽管这种通用性解释对过滤器询问是有意义的，但对使用更为丰富的语义（例

如，聚集）的成熟的 SQL 询问来说却显然不是这样。这种询问对开发者使用应用程序来说是有关的。那些询问具有与更为丰富的规则系统的通用性。

[0089] 终端用户询问一般是简单的过滤器。有趣的是，这类过滤器询问也正是由 OPath 语言所提供的。

[0090] 图 9A 示出了依照本发明的规则的一般格式。规则简单地是一组采用形式 ON（每个数据项）IF（条件为真）THEN（结论）的语句。它可被用于表示判定、事实，并且一般来说，可以表示各种说明性的语句。询问是规则的受限形式，其中规则的结论被限制成在结果中包含项（一般来说，排除项也是可能的结论）。定义询问结果的内容的说明性语句如下：ON（每个数据项）IF（条件为真）THEN（将该项包含于询问结果中）。实际上，基于最初模拟哪种 SQL 语言，这就是在关系演算中怎样模拟询问。在活动规则的情况下，规则的结论被解释成要执行的副作用动作。

[0091] 既然规则是说明性的，那么不同的评价算法都是可能的。低延迟算法适合于客户机，而高吞吐量的算法适合于服务器。规则存留为项，导致安全、共享和系统化范围内的标准行为。

[0092] 规则的条件是结合由输入项类型所定义的基本谓语 (predicate) 的布尔表示。针对项属性的公共比较条件，以及由模式开发者定义的高级条件方法都被支持。第一个原则是终端用户规则内的条件和终端用户询问内的条件在表达力和表达的容易程度和 metaphor 方面应该相同。第二个原则是终端用户视图应该可与其他规则 / 询问 / 视图相组成 (composable)。这允许通过组成的构件块来构建丰富的终端用户逻辑。注意，跨关系和跨数据集来表达条件也是可能的（在数据库用法中，是“加入”(join)，但对一般的终端用户来说，是以有意义的程式化的样式来表示）。

[0093] 规则的结果简单地是数据。当将规则用于应用程序客户化时，它们需要由消费应用程序来进行解释。在激活规则的情况下（这些提供带有副作用动作的事件触发的行为），结果对应于要执行的动作。规则平台提供主控服务，这些动作在该主控服务中被执行。

[0094] 规则项是包括关系（例如“包含”规则项）的集的项。规则的逻辑行为是在具体类型的单个输入项上定义的。规则被应用于输入项。

[0095] 数据库内的编程构造的主要类型是触发器、存储的过程 (sprocs)、视图 / 询问、以及约束。同样地，依照本发明的终端用户编程具有活动规则（触发器）、客户化规则 (sprocs)、导出规则（视图）和约束规则（约束）。

[0096] 图 9B 示出了依照本发明的用于活动规则的一般规则格式。在计算机需要代表用户执行动作的任何情况下，使用活动规则，这些情况包括例如：自动组织从照相机下载的照片；处理、过滤并转寄收到的电子邮件、电话呼叫和 IM；通知用户相关的警告；以及由终端用户定义的简单的专门的“工作流”。

[0097] 活动逻辑由形式为 ON（项）IF（条件）THEN（动作）的规则来定义。THEN 子句中的结果对应于从存储系统中可用的一组基本动词中选择出或者存储模式开发者定义的可执行的动作（例如，移动和添加）。为了为终端用户规则定义条件和动作，高级开发者能定义专门的模式或模式扩展。

[0098] 有两种方法使活动规则可依附于项范围中。第一，可用周期性语义将活动规则依附于项集合。该规则周期性地应用于每个项并且出现适当的动作（在任何冲突解决以后）。

第二，可用事件语义将活动规则依附于项集合。无论何时在项集合中发生变化，针对变化的项执行该规则。

[0099] 图 9C 示出了依照本发明的用于应用程序客户化规则的一般规则格式。如今的应用程序的行为并不是非常客户化的。一般来说，任何客户化基于简单的选项设置。相反，规则允许终端用户基于当前的应用程序和用户的上下文，以数据驱动的逻辑在各种拦截器点客户化应用程序。客户化规则可由希望为终端用户控制提供指向的任何应用程序使用。一种考虑方法是，通过应用程序的工具→选项 (Tools->Options) 窗格设定的每个值应该是可定义的，而不仅作为常量，而是可通过判定逻辑定义。例如，确定哪个签名文件要应用于发出的邮件消息的规则、是否用警告打断用户的规则等等。

[0100] 应用程序客户化规则采用 ON（项）IF（条件）THEN（结果）的形式。应用程序提供项或项的集作为输入。结果实际上不是作为规则评价的一部分而自动执行的，相反，它们只是简单地作为规则执行的结果来返回，用于应用程序正确解释。

[0101] 图 9D 示出依照本发明的数据导出规则的一般规则格式。终端用户能定义使用基于内容的过滤器定义项的集的规则。这些集叫做导出的项集合 (itemsets)。操作系统视觉浏览 metaphor 具有 Dynamic Sets（动态集）的概念，它表示最相关的概念。所有的终端用户的询问和视图落入该范畴内。

[0102] 导出规则采用 ON（项）IF（条件）THEN（包括 / 排除）的形式。注意，唯一允许的结果是包括和排除。通过将规则依附于项范围来使用它。结果产生的导出的 ItemSet 包含按照规则“包括在内”的源集合内的那些项。实际上，这些导出规则在项范围上定义了丰富的过滤器。

[0103] 导出的 ItemSets 的一些例子如下：一组叫做 MyFriends 的人，定义为家庭电话号码不为空的联系人 (Contacts)，和不标记为商业联系人的联系人 (Contacts)；以及一组叫做 InBoxMailFromMyFriends 的电子邮件，定义为 InBox（收件箱）中来自 MyFriends（我的朋友）中的任何人的任何电子邮件。

[0104] 注意，导出的项集合的定义是组成性的，即 InBoxMailFromMyFriends 使用了 MyFriends 的定义。为了终端用户构建模块化的定义，这种组成性是很重要的。导出的项集合的模块化的定义也使其他种类的规则更富表达力。例如，InBox 内的“活动”规则指示出如果发送者在 MyFriends 内，则该邮件应被标记为“个人的”。

[0105] 图 9E 示出依照本发明的数据约束规则的一般规则格式。终端用户也能使用形式为 ON（项）IF（条件）THEN（允许 / 不允许）的规则在项集合上指定约束。约束规则依附于现有的项集合（这是用于规则的项范围），并且由项集合的变化（插入 / 更新 / 删除）激活。该激活与变化同步。有多种可能的失败行为（如果该规则结果为“不允许”）：变化（一般为项添加）本身可能失败。并且可选地，正被添加到集合的项可被修改，以符合约束。

[0106] 每个规则与指定规则输入的类型、允许的条件和允许的规则结果的规则约束相关。例如，用于定义导出的项集的规则或在询问约束中，其结果是包括输入项或排除输入项。终端用户不处理规则约束。它是应用程序和开发者的概念，约束终端用户可以在某一判定的上下文中指定的逻辑。RuleConstraint 类型用于描述这些约束。

[0107] 可能观察到约束规则和导出规则非常相似。然而，差异在于约束规则仅为集合内的成员指定必要的条件。它们不指定集合应该怎样被填充（换句话说，没有针对其应用规

则逻辑的项的域）。

[0108] 终端用户编程 (EUP) 模型

[0109] 传统地，大量的为文件系统的设计工作已经基于对“手动”交互范围中终端用户的要求的理解，例如，复制语义、使用期限管理等等。这些要求已经被记录并插入到核心文件系统数据模型和 API 中。然而，通过包含更为丰富和更多的动态交互，并带有终端用户编程的数据操作，终端用户与文件系统数据进行交互的全部要求比“手动”交互更为宽泛。

[0110] EUP 的设计是基于与把终端用户看作要求的首要驱动者有关的四个主要原则：

[0111] Compositional Logic（合成逻辑）。复杂的逻辑是使用较小的合成单元以较简单的结构构造的。所有的终端用户程序都是说明性的。终端用户程序由逻辑组件的合成来构造。存在一种定义逻辑组件怎样能被装配在一起的 EUP 代数。每个逻辑组件是对终端用户有意义的逻辑的片断。无论该逻辑组件用于定义询问，还是用于定义复杂的动词或约束，它都是使用普通的 IF-THEN 规则构造来定义的。除普通的逻辑模型外，也允许跨应用程序的规则逻辑的普通视觉显示。

[0112] 作为数据管理程序。每个 EUP 程序和每个 EUP 逻辑组件被表现为文件系统项。从终端用户的角度看，他们定义的复杂的动作或过滤器与他们已经编辑的文档并无不同。从共享、同步、备份、安全性等等角度来看，它是一块数据。有用的副作用是与 EUP 概念交互的 API 表面是与任何数据项交互的标准 API 表面。

[0113] 终端用户灵活性。它都是关于给终端用户以能力和灵活性。例如，终端用户能定义新的终端用户类型、动态地修改现有的类型、定义并持续询问、定义影响特定项的“商业逻辑”等等。在说明性的、代数的和基于规则的适当的抽象水平上来描述这些。

[0114] 跨应用程序的公共模型。终端用户在呈现给用户的应用程序或视觉浏览 metaphor 内所做的工作可继续在其他的应用程序内。例如，如果为在该 metaphor 内的 Document（文档）类型设置终端用户属性，当针对该 metaphor 内的一些其他部分中或在另一个文件系统应用程序内的文档来定义询问时，终端用户可期望能够使用该属性。

[0115] 对于数据的 EUP，存在五个基本方面：数据类型、逻辑、询问和视图、动词和自动化操作 (automation)、判定、以及约束。数据类型涉及类型映射和属性映射。类型映射包括终端用户可以看见并可以推论的数据类型，以及这些类型如何映射到文件系统。属性映射涉及该用户看见的与终端用户数据类型相关的属性，以及这些属性如何映射到底层的存储类型。

[0116] 逻辑涉及询问、自动化操作、约束和应用程序客户化。编程模型定义终端用户逻辑，而创作模型定义终端用户怎样能定义复杂的逻辑。提供了终端用户逻辑的持久性，而管理描述终端用户逻辑如何被共享和管理，以及用于命名、共享、系统化等的机制等等。

[0117] 询问和视图提供表达力的模型，例如，它描述要呈现给终端用户的正确的过滤和询问抽象。行为考虑当终端用户把项拖入或拖出自动列表 (auto-list)（例如，行为象列表的自动列表）时，期望的行为应当是什么。就呈现抽象而言，有一组与自动列表向终端用户的呈现相关的逻辑信息（例如，投影属性、分类顺序、分页），也有一些实际信息（页面尺寸等等）。

[0118] 动词和自动化操作描述根据底层数据模型，终端用户“动词”是什么，终端用户如何构造新的复杂的条件动词，对项的动词的应用如何可共同运用于一组项以及更为丰富的

动词，、以及动词的应用如何能够自动化操作。

[0119] 判定描述应用程序客户化判定（由终端用户客户化规则所定义）如何被建模。

[0120] 约束描述终端用户如何可在项 / 项集合的内容上定义约束、以及可实施的语义。

[0121] 在终端用户编程 (EUP) 代数中存在五个基本的术语：

[0122] Property(T)（属性 (T)）：它描述终端用户类型 T 的属性。该用户使用属性来描述过滤器和动作。

[0123] Filter(T)（过滤器 (T)）：它定义可用作类型 T 的项上的过滤器的布尔逻辑函数。它是返回类型是布尔值的规则。

[0124] Event(T)（事件 (T)）：它定义所关心的事情的出现。它一般与其类型为 T 的数据项（事件数据）相关。

[0125] Action(T)（动作 (T)）：它是项类型 T 的副作用方法，并且一般要求其他参数。

[0126] Set(T)（集 (T)）：它是类型 T 的项的集。

[0127] 这些术语的每一个的实例可以由开发者从系统内定义的项模式中获得。在 Set(T) 的情况下，这个术语的实例是文件系统中的任何集合。然而，这些术语的每一个的实例也能由终端用户通过简单的代数构造器或通过规则来构造。以下的基本术语可以使用规则来构造：

[0128] Property(T)：输入项类型为 T 而输出类型为 O 的规则在类型 T 上定义类型 O 的属性。

[0129] Filter(T)：输入项类型为 T 而输出类型为布尔值的规则在类型 T 的项上定义过滤器。

[0130] Action(T)：输入项类型为 T 的规则和输出类型在类型 T 的项上定义动作。

[0131] 在每个规则内，可以使用包括以下条件的各种条件：< 属性 >< 比较运算符 >< 表达式 >、< 现有的过滤器 >、ANY/EVERY（任何 / 每个）< 关系目标 >MATCHES（匹配）< 过滤器 >、以及 ANY/EVERY< 关系目标 >IN< 集 >。这些条件示出可以出现在 Rule 内的不同种类的组合。这也包括聚集，聚集是集中的项的计数。可以使用简单的二进制代数运算来构造下面的术语：Event(T) = Event(T)+Filter(T)，其中导出的事件由过滤器应用于另一个事件的事件数据来定义，例如，NewInterestingItem(Doc) = NewItem(Doc)+IsInteresting(Doc)；Filter(T) = Filter1(T_SubType1)（过滤器 1(T_ 子类型 1)）Union（联合）Filter2(T_SubType2)（过滤器 2(T_ 子类型 2)）；Set(T) = Set1(T)+Filter(T)，其中导出的集由过滤器对另一个集内的每个项的应用来定义；导出的集成员是过滤器评价为真的那些项；以及 Set(T) = Set1(T)Union Set2(T)。

[0132] 个体 EUP 逻辑组件可直接在应用程序内进行评价。这是用于应用程序客户化的方法（例如，对于类似 Outlook 的应用程序，以允许终端用户通过终端用户逻辑来客户化该应用程序的行为）。EUP 的中心元素通过环境视觉浏览 metaphor 来显露出，并包括以下三种程序：

[0133] Derived Set（导出的集）：导出的集（先前作为代数术语描述过）本身是完整的终端用户程序。当打开它时，执行该集定义的逻辑，且结果一般被显示出来。下面的代数运算也被支持用于导出的集，Set(T) = Set1(T)Union Set2(T)。

[0134] Batch（批）：Batch = Set(T)+Action(T)。批定义为要执行的面向集的任务。批

的语义是在集内的每个项上执行该动作。可以手动执行或计划执行。

[0135] Agent(代理):Agent = Event(T)+Action(T)。代理定义当事件发生时要执行的动作。

[0136] 所有这三个类型的EUP程序被定义为文件系统项,也作为逻辑组件。

[0137] 可以包括进一步的代数运算。例如,考虑各种集组合运算(交集、集差异等等)来定义导出的集是可行的。

[0138] RuleLogic(规则逻辑)项定义终端用户逻辑的单元。QueryFilters(询问过滤器)、复杂的动词、复杂的事件和计算出的属性是RuleLogic的实例。从逻辑上讲,RuleLogic项记录了用户定义的功能在输入项上的定义。每个RuleLogic项包含RuleConstraint(规则约束),它定义该功能的输入项类型和输出类型。存在零或更多IF-THEN语句。每个语句有Condition(条件)(对应于语句的IF部分)和Result(结果)(对应于THEN部分)。Condition是嵌套元素的布尔表达式树,该树的叶子是LeafCondition(叶条件)元素。Result是零个或更多的ResultElements(结果元素)的嵌套元素集合,每个ResultElement有一个名称和一组参数值。在QueryFilters的情况下,语句的Result是具有名称为True(真)或False(假)、且没有参数的单个ResultElement。

[0139] 这里简要地描述了任一RuleLogic定义(因此,还有QueryFilter)的语义。如果QueryFilter没有语句,则输出为NULL(空)。如果QueryFilter有一个语句,那么,如果Condition评价为True且Result为True则输出为True。如果Condition评价为True而Result为False则输出为False。另外的情况则输出为NULL。如果QueryFilter有多个语句,那么在Condition评价为真的那些语句的Result上使用聚集函数来计算输出。最常见的聚集函数是基于语句顺序(提供嵌套的if-then-else或switch(开关)语句语义)。

[0140] 现在参考图10,示出了依照本发明的规则架构,一种通过依附于输入项的集合来应用规则的方法。术语“集合”指项引用(item reference)的任何集(在shell(外壳)术语中称为集,在规则术语中称为关系的多集(multi-set))。在1000,接收提供输入项范围的项集合。在1002,规则可以依附于提供了输入项范围的项集合。在1004,调用该规则。在1006,当调用时,将该规则应用到项范围内的每个项。

[0141] 现在参考图11,示出了依照本发明的规则架构的通过依附于判定点来应用规则的另一种方法。在1100,暴露判定点。在1102,将该规则依附于判定点。该规则可以依附于用于为规则进行评价提供项的判定点。

[0142] 现在参考图12,示出了依照本发明的规则架构,通过直接调用规则来应用规则的另一种方法。在1200,接收输入项用于处理。在1202,以规定了其输入的参数直接调用规则。

[0143] 现在参考图13,示出了依照本发明的规则架构的应用规则的另一种方法。在1300,接收输入项范围用于处理。在1302,该规则应用到在输入项范围内变化的那些项。上一种情况模拟规则的公共使用以提供活动行为(有时叫做处理事件的能力)。在规则的这种使用中,输入项是“由事件产生的”并且规则评价的结果可能导致“动作”被执行。新颖的规则平台提供了这种活动处理事件能力,作为规则系统的顶部的一层。

[0144] 现在参考图14,示出了依照本发明的用于规则冲突解决的一种方法。作为缺省方式,终端用户逻辑的行为是个体规则的累积行为。然而,在许多情况下,规则可能具有“冲

突”的交互。在1400,接收包括用于处理的多个规则的用户定义的逻辑。一些不同的冲突解决策略得到支持。例如,在1402,可以通过便于规则的终端用户规则优先化的判决组件解决规则冲突。在1404,判决组件的一种替代方法提供了可通过开发者定义的判决策略(例如,定制聚集)来解决规则冲突。

[0145] 规则创建框架由创建规则时和构建使用规则的应用程序时所使用的项和关系组成。规则平台的规则引擎使用这些项和关系来确定响应于特定的事件,计算机应做些什么。为了客户化基于终端用户规则的应用程序行为,发生两件事情:程序员“在规则上允许”软件的功能,终端用户为那些功能创建规则。“规则允许”意味着为用户合成规则而定义与应用程序定义的类型或OOTB(框外)类型相关联的EUP组件。“允许”也意味着创建可对其依附规则的判定点。例如,规则允许的操作系统特征被包括在内,以便用户可以在项的变化上定义规则,这些项如文件夹和文档。

[0146] 规则API用于将规则输入引入到规则评价引擎,维护终端用户规则,以及注册(register)和考虑规则架构的各个部分,包括条件、动作、绑定等等。规则API作为助手方法来实现,它完成由规则类型定义的部分类,但有三种例外:规则输入、规则输入集合和规则异常类型。这意味着用于规则的API是针对规则类型的API。本发明的规则API提供以下内容:

[0147] 从Query类型分离的AppCustomization/Active Rules类型。分离允许在与Query相关的类型和AppCustomization类型上构造不同的助手方法。这也是从开发者的角度的简化;只对使用规则引擎用于在其应用程序内进行应用程序客户化感兴趣的开发者只需要学习那些API。

[0148] 组合的Rule、Rules概念。Rules是由约束信息(输入类型和输出类型)和一个或多个LogicStatement构成的。每个LogicStatement由Condition树和一个或多个结果组成。

[0149] LogicResult类型族(为了以后的扩展)。LogicResult是结果类型继承的基类型。支持两种结果类型,现有的“Action”,它包含已编码的方法调用(FunctionInfo),以及Boolean(布尔值)返回类型(主要用于QueryFilters/AutoLists)。在另一个实现中,可以扩展到覆盖标量和XML(可扩展标记语言)结果。

[0150] EUP项之间基于值的连接(Value-based Connections Between EUP Items)。这是关联项的附加方式,而非坚持项之间的清楚的关系/链接。规则API使用ItemIds用于编码规则附件等等。以这种机制来使用Association概念,这个概念是一种已说明的基于值的关系。

[0151] 规则API(指代EUP组件的另一种方式)由System.Storage.Rules名称空格内的类型组成。它提供了引入新的事件的手段(RuleInput,RuleInputCollection)。除了极少的例外之外,这些类型是先前列举的EUP组件和程序。这些类型是系统化的类型,这意味着它们是实例化的、持久性的和使用标准存储操作API来被发现。这些类型提供创建并维护用户规则(从RuleLogic中导出的类型)的手段。这些规则可以是活动规则,用于应用程序客户化的规则,或是结构化的询问规则。

[0152] 现在参考图15,示出了本发明的规则架构的项的类型和相关联的关系。规则平台所暴露的大多数公共API包括在System.Storage.Rules模式中说明的类型。这个规则模

式定义了 EUP 组件项类型。这些用于代表用户来实例化规则逻辑，允许应用程序说明可对其依附规则逻辑的判定点，以及允许应用程序保持并传递规则评价的结果。

[0153] RuleLogic。保持用户规则逻辑的所有项的基类型。它包含对所有规则公共的基本“形状”。

[0154] 源自 :(OS 文件系统). 项

[0155] 类型内添加的属性 :

[0156] ● Constraint :用于该规则的 LogicConstraint。

[0157] ● Statements :LogicStatement 的多集 (multiset)。每个规则包含一个或多个语句 (条件 / 动作对)。评价这些条件，如果为真，则返回动作作为结果。多集内的排序暗指语句间的优先级排序 ;Statement[0] 具有最高优先级。也可以实现有序的多集。没有语句的 RuleLogic 项被认为是不合法的。在一个实现中，当商业逻辑存在时，可以抛出运行时间异常。

[0158] ● Enabled :布尔逻辑标记这条逻辑为允许或者不允许 (默认情况为 :真) 添加到该类型的方法 :无。

[0159] QueryFilter。QueryFilter 表示用户创作的过滤逻辑。QueryFilter 内的语句具有指定布尔输出类型的 LogicConstraints。

[0160] 源自 :RuleLogic

[0161] 添加到该类型的属性 :无

[0162] 添加到该类型的方法 :

[0163] ● QueryFilter(Type inputItemType) :允许设定输入项类型的构造器。将在 LogicConstraint 中设定布尔逻辑结果类型。

[0164] ● QueryFilter(Type inputItemType,LogicStatement 1) :设定输入项类型 (同上) 以及添加单个 LogicStatement 的构造器。

[0165] ● Collection Evaluate(CollectionScope cv) :给定输入集合，评价询问过滤器。返回结果作为 ItemIds 的集合。

[0166] ● Collection Evaluate(AutoList a) :给定 Auto List，评价询问过滤器作为输入集合。

[0167] ● Boolean Evaluate(Item i) :在 QueryFilter 内保持的 LogicStatements 上评价单个项 ;返回最终结果。

[0168] AutoList。AutoList 表示 AutoList 的 windows 浏览 metaphor 概念。它指 QueryFilter+InputScopes，并可以用通过 ItemSearcher 返回的结果来评价。

[0169] 添加到该类型的属性 :

[0170] ● InputScopes :InputScope 的多集。AutoLists 有一个或多个输入范围，当进行评价时，可以在这些输入范围上应用它们的过滤器。有 0 个输入范围的 AutoLists 是非法的。当商业逻辑存在时，可以抛出运行时间异常。

[0171] ● QueryFilters :QueryFilter 项的 ItemId 的多集，包含评价该 AutoList 时要使用的逻辑定义。

[0172] 添加到该类型的方法 :

[0173] ● AutoList(Type inputItemType,InputScope InputScope) :允许设定输入项类

型和输入范围的构造器。将在 LogicConstraint 中设定布尔逻辑结果类型。

[0174] ● AutoList(Type inputItemType,InputScope InputScope,QueryFilter):设定输入项类型、一个输入范围、以及添加对单个 QueryFilter 的引用的构造器。

[0175] ● ItemSearcher GetItemSearcher():在评价这个 AutoList 的结果上返回 ItemSearcher。

[0176] ● Internal string GetViewName():返回支持这个 AutoList 的生成的视图的名称。用于 ItemSearcher 集成。

[0177] AppRule。在应用程序客户化的情况下使用 AppRules。

[0178] 源自:RuleLogic

[0179] 添加到该类型的属性:

[0180] ● DecisionPoint:该 AppRule 所依附于的判定点的 ItemId。

[0181] 添加到该类型的方法:

[0182] ● AppRule(LogicConstraint lc):允许设定 LogicConstraint 的构造器。

[0183] ● AppRule(LogicConstraint lc,LogicStatement 1):允许设定 LogicConstraint 并且允许添加单个 LogicStatement 的构造器。

[0184] ● EvaluationResults Evaluate(Item inputItem):对于给定的输入项,评价该 AppRule,并返回生成的结果。

[0185] ● EvaluationResults Evaluate(CollectionValue c):对指定的集合内的每个项,评价该 AppRule,并返回生成的结果。

[0186] ScopedAppRule。ScopedAppRule 主要用于 Active Rules 的情况以添加规则逻辑应被施加于其上的输入范围。某些应用程序客户化情况会要求逻辑与文件系统内的位置的联合,这是可能的。这些情况能够用这些项自己的判定点来利用这些项。

[0187] 源自:AppRule

[0188] 添加到该类型的属性:

[0189] ● InputScopes:InputScope 的多集(参见下文)。ScopedAppRules 有一个或多个其逻辑可以应用于其上的输入范围。有 0 个输入范围的 ScopedAppRules 是非法的。当商业逻辑存在时,可以抛出运行时间异常。

[0190] 添加到该类型的方法:

[0191] ● ScopedAppRule(LogicConstraint lc,InputScope InputScope):允许设定 LogicConstraint 和单个输入范围的构造器。

[0192] ● ScopedAppRule(LogicConstraint lc,InputSeope InputScope,LogicStatement 1):允许设定 LogicConstraint 和单个输入范围、并且允许添加单个 LogicStatement 的构造器。

[0193] DecisionPoint。应用程序创建 DecisionPoints 以描述何时何地它们将提交输入到规则引擎,以及它们希望接收何种结果。DecisionPoints 是一种机制,应用程序通过该机制可以提交输入到规则引擎,用于应用程序客户化的情况。

[0194] 源自:(OS 文件系统). 项

[0195] 该类型的属性:

[0196] ● ApplicationName:创建 DecisionPoint 的应用程序或模式的名称。例如,这可

由 UI(用户界面)应用程序使用来对给定的应用程序显示所有的 DecisionPoint。

[0197] ● ScopeRequired:当为真时,依附于此的任何 RuleLogic 项必须具有定义的 Input Scopes(ScopedAppRule)。

[0198] ● Constraint:描述为该 DecisionPoint 创建的规则上的约束的 LogicConstraint(参见下文)。此约束描述当应用程序到达 DecisionPoint 时,应用程序将提交的输入种类 (OS 文件系统类型 id),它期望的输出的种类也是如此。

[0199] 添加到该类型的方法:

[0200] ● DecisionPoint(string ApplicationName, LogicConstraint c, boolscopeRequired):允许设定所有属性的构造器。

[0201] ● EvaluationResults Submit(RuleInput r):提交给定的 RuleInput 并取回与其处理相关的 EvaluationResults。

[0202] ● EvaluationResults Submit(RuleInputCollection r):提交给定的 RuleInputCollection。

[0203] EvaluationResults。提交的每个 RuleInput 导致创建单个 EvaluationResults 项。这个项包含基于提交的数据进行的规则评价的结果(如果有的话)。这个新创建的项的 itemid 返回给提交应用程序;然后,它可读取该项数据以确定要采取什么动作。

[0204] 源自:(OS 文件系统). 项

[0205] 添加到该类型的属性:

[0206] ● Results:EvaluationResultElements 的多集,其中每一个包含单个规则结果(要采取的动作)。

[0207] 添加到该类型的方法:无。

[0208] 图 16A-H 示出了依照本发明的规则架构所使用的嵌套元素类型。

[0209] 图 16A 显示本发明的 EvaluationResultElement 嵌套类型。EvaluationResultElement 表示单个规则结果(要采取的 ActionResult 或要返回的其他种类的 Result)。

[0210] 源自:(OS 文件系统).NestedType

[0211] 添加到该类型的属性:

[0212] ● Result:指定从规则返回的结果的 LogicResult。

[0213] ● InputItemId:导致生成该 EvaluationResultElement 的输入项的 ItemId。

[0214] ● RuleItemId:导致生成该 EvaluationResultElement 的 RuleLogic 项的 ItemId。

[0215] ● Statement:在导致生成该 EvaluationResultElement 的逻辑项内指定 Statement 的整数。

[0216] 添加到该类型的方法:

[0217] ● void Execute(ItemContext ic):助手方法。如果 Result 属性包含 ActionResult,调用其中指定的方法。它不调用 ItemContext.Update();调用者这样做。如果 Result 不是 ActionResult,该方法应抛出 RuleException。参数替换:如果 Result 包含 ActionResult, ActionResult 内的 Arguments(自变量)将是 ConstantValues,它表示由 LogicStatement 指定的经后处理的 Arguments。

[0218] 图 16B 显示本发明的 InputScope 嵌套类型及其导出类型。为图 15 的 AutoList 或 ScopedAppRule 表示 InputScope，

[0219] 源自 :(OS 文件系统).NestedType

[0220] 添加到该类型的属性 :

[0221] ● ItemId :输入范围的 ItemId。

[0222] 添加到该类型的方法 :

[0223] ● InputScope(Guid inputScope)

[0224] CollectionScope 是作为集合的 InputScope。

[0225] 源自 :InputScope

[0226] 添加到该类型的属性 :

[0227] ● RelationshipSchema :保持定义该集合的关系的 String(串) 模式名称。

[0228] ● RelationshipName :在以上定义集合的模式内的关系类型的 String 名称

[0229] ● ShallowScope :Boolean :如果为真，输入范围应限制在一级关系遍历内。缺省为假。

[0230] 添加到该类型的方法 :

[0231] ● CollectionScope(Guid inputScope，string relationshipSchema，stringrelationshipName)

[0232] AutoListScope 是作为 AutoList 的 InputScope。

[0233] 源自 :InputScope

[0234] 添加到该类型的属性 :无。

[0235] 添加到该类型的方法 :

[0236] ● AutoListScope(Guid inputScope)

[0237] CSIDLScope 是作为视觉浏览 metaphor CSIDL 的 InputScope。CSIDL 值提供独立于系统的独特方法以标识应用程序频繁使用的特定文件夹，但它们可能不在任何给定的系统上具有相同的名称或位置。

[0238] 源自 :InputScope

[0239] 添加到该类型的属性 :

[0240] ● CSIDL :为当前用户评价的并用作为输入范围的 CSIDL 的值。

[0241] 图 16C 显示本发明的 LogicConstraint 嵌套类型。RuleLogic(及其孩子) 和 DecisionPoints 都使用 LogicConstraint。当用于 DecisionPoint 时，LogicConstraint 指定要提交用于评价的输入项类型和作出说明的应用程序预期的输出类型。当用于 RuleLogic 时，该约束说明可以针对什么输入类型来评价该规则，以及该规则产生什么类型的输出。

[0242] 源自 :(OS 文件系统).NestedType :

[0243] 添加到该类型的属性 :

[0244] ● InputItemTypeId :指定输入项类型的 OS 文件系统 TypeId。

[0245] ● Output :来自指定输出类型的 OutputType enum(参见下文) 的值。

[0246] 添加到该类型的方法 :

[0247] ● LogicConstraint(Type inputItemType，OutputType ot) :采用 typeOf(Item)

和 OutputType（参见下文），并发现适当的 OS 文件系统 TypeId 以存储在 InputItemTypeId 中。

[0248] ● OutputType(enum)，描述对于 Logic/DecisionPoint 输出什么类型（或者输出类型上的约束）。用于 LogicConstraint 中。

[0249] 值：

[0250] ● Boolean：

[0251] ● FunctionInfo

[0252] 图 16D 显示本发明的 LogicStatement 嵌套类型。LogicStatement 对条件树和一组结果（动作或布尔逻辑）进行编码。

[0253] 源自：(OS 文件系统).NestedType：

[0254] 添加到该类型的属性：

[0255] ● Condition：该逻辑语句的条件树。

[0256] ● LogicResults：LogicResult 元素的多集。如果条件评价为真，对这些结果进行编码。

[0257] 添加到该类型的方法：

[0258] ● LogicStatement(Condition c, LogicResult 1)：设定条件树和单个结果的构造器。

[0259] 图 16E 显示本发明的 LogicResult 嵌套类型及其导出类型。LogicResult 是用于所有结果类型的基类。它从不用于其自身；而是使用其孩子来编码来自 RuleLogic 项的结果。

[0260] 源自：(OS 文件系统).NestedType：

[0261] 添加到该类型的属性：无。

[0262] 添加到该类型的方法：无。

[0263] BooleanRusult。允许一段规则逻辑返回布尔值。主要由 QueryFileter/AutoList 所使用。

[0264] 源自：LogicResult

[0265] 添加到该类型的属性：

[0266] ● Result：包含该结果的布尔逻辑值。缺省为真。

[0267] 添加到该类型的方法：

[0268] ● BooleanRusult(bool result)。

[0269] ActionResult。Actions 是活动或应用程序客户化规则的结果。

[0270] 源自：LogicResult

[0271] 添加到该类型的属性：

[0272] ● FunctionInfo：FunctionInfo 对调用的方法进行编码。

[0273] ● Arguments：包含用于 FunctionInfo 的有序自变量的多集。

[0274] ● Result：标量结果是串，和描述该实际标量值的枚举 (enum)，例如，整型、日期、串等等。

[0275] 添加到该类型的方法：

[0276] ● Action(LogicRusult lr, params object[]arguments)：设定逻辑结果以及可

选地设定任何必需的自变量的构造器。

[0277] ● ScalarRusult(String result,enum ScalarType)

[0278] ScalarResult 。

[0279] 添加到该类型的属性：

[0280] ● Result：标量结果作为串，和描述该实际标量值的枚举，例如，整型、日期、串等等。

[0281] 添加到该类型的方法：

[0282] ● ScalarRusult(String result,enum ScalarType)

[0283] 图 16F 显示本发明的 Condition 嵌套类型及其导出类型。Condition 是用于对规则的条件进行编码的基类型。它不能直接被实例化；而是使用其孩子作为给定的规则实例的一部分。Condition 不包含其自身的属性。

[0284] 源自：(OS 文件系统).NestedType

[0285] 添加到该类型的属性：无。

[0286] 添加到该类型的方法：

[0287] ● public static Condition operator &(Conditionleft,Conditionright)：过载“&”以创建新的带有左和右参数的 AndCondition。

[0288] ● public static Condition operator|(Condition left,Condition right)：过载“|”算符。

[0289] ● public static Condition operator！ (Condition c)：返回带有 Condition c 的 NotCondition。

[0290] ConditionTree 源自 Condition 并对一系列条件进行编码。ConditionTree 从不直接被使用；而是使用其孩子 AndCondition 和 OrCondition。

[0291] 源自：Condition

[0292] 添加到该类型的属性：

[0293] ● Children：Conditions 的多集。它们可能是单个条件或其他树。

[0294] 添加到该类型的方法：无。

[0295] AndCondition 源自 ConditionTree。当 AndCondition 树被添加到规则时，它表示应该评价分量条件，并且当确定规则评价成功时，结果应该逻辑地 AND(逻辑与)在一起。

[0296] 源自：ConditionTree

[0297] 添加到该类型的属性：无。

[0298] 添加到该类型的方法：

[0299] ● AndCondition(params Condition[]operands)：允许在构造时指定分量条件的构造器。

[0300] OrCondition 也源自 ConditionTree。当 OrCondition 树被添加到规则时，它表示应该评价的分量条件，并且当确定规则评价成功时，结果应该逻辑地 OR(逻辑或)在一起。

[0301] 源自：ConditionTree

[0302] 添加到该类型的属性：无。

[0303] 添加到该类型的方法：

[0304] ● OrCondition(params Condition[]operands)：允许在构造时间指定分量条件

的构造器。

[0305] NotCondition 源自 Condition。当 NotCondition 被添加到规则时，评价构成条件(consituent condition)，然后使用结果的否定。

[0306] 源自 :Condition

[0307] 添加到该类型的属性 :

[0308] ● Condition :单个 Condition(或者任何类似 ConditionTree 的导出类型)添加到该类型的方法 :

[0309] ● NotCondition(Condition condition) :允许在构造时间指定该条件的构造器。

[0310] LeafCondition 用来表示实际条件。它描述要使用的条件函数的名称，以及要用于有名的条件函数的自变量的多集。

[0311] 源自 :Condition

[0312] 添加到该类型的属性 :

[0313] ● Name :评价期间使用的条件函数的名称。当前它可以命名任何受支持的 SQL 比较器，内建的面向集的条件或者触发项类型上的任何方法。

[0314] 支持的 SQL 内建组件(和文本等价物)是“=”,“Equals”,“>”,“GreaterThan”,“<”,“LessThan”,“>=”,“GreaterThanEqualTo”,“<=”,“LessThanEqualTo”,“<>”,“NotEqual”,“Like”

[0315] 支持的面向集的条件 :

[0316] ANY_IN :用于 RelationshipValue 和 CollectionValue。确定在有名集合内定义的任何关系是否是 RelationshipValue 内指定的类型。

[0317] ALL_IN :用于 RelationshipValue 和 CollectionValue。确定在有名集合内定义的所有关系是否是 RelationshipValue 内指定的类型。

[0318] ●自变量 :有名的条件函数的自变量的多集。因为，在一个实现中，多集是无序的，使用 LeafCondition1 和 LeafCondition2 属性。

[0319] 添加到该类型的方法 :

[0320] ● LeafCondition(string name, params_object[]arguments) :允许在构造时间设定条件函数名和自变量的构造器。

[0321] OPathCondition 表示复杂的、开发者定义的一块 OPath。为了被使用，OPath 表达式必须评价为布尔返回值，例如 :(“Count(OutDocumentAuthorRelationships) > 1”)。

[0322] 源自 :LeafCondition

[0323] 添加到该类型的属性 :

[0324] ● OPath :包含 OPath 表达式的串。

[0325] ● DisplayName :询问构建器或其他 UI 中显示的该块 OPath 的串名称。

[0326] 添加增加到该类型的方法 :

[0327] ● OPathCondition(string exp) :设定 OPath 表达式的构造器。

[0328] 图 16G 显示依照本发明的 ArgumentValue 嵌套类型及其导出类型。ArgumentValue 是用于为条件和动作编码自变量的基类型。它并不被直接实例化，而是使用其孩子类型(ConstantValue、PropertyValue、RelationshipValue、CollectionValue 等等)。

[0329] 源自 :OS 文件系统 .NestedType

[0330] 添加到该类型的属性：无。

[0331] ConstantValue 源自 ArgumentValue 并被用于表示用于规则的条件和动作的自变量值内的常量值。

[0332] 源自：ArgumentValue

[0333] 包括在该类型中的属性：

[0334] ● Value：包含要由该自变量表示的单个串值。

[0335] 添加到该类型的方法：

[0336] ● ConstantValue(string value)：允许在构造时间直接设定值的构造器。

[0337] PropertyValue 源自 ArgumentValue 并被用于表示作为 OS 文件系统类型的属性的条件和动作的自变量。注意，在另一个实现中，可使用类型 GUIDs 而不是串模式和类型名，以用于与规则架构内触发项类型的一致性。

[0338] 源自：ArgumentValue

[0339] 包括在该类型中属性：

[0340] ● PropertyName：包含作为要用作参数的入站 (inbound) 项上的属性的名称。

[0341] 添加到该类型的方法：

[0342] PropertyValue(string propertyName)：允许在构造时间指定属性名称的构造器。

[0343] RelationshipValue 源自 ArgumentValue 并被用于表示要用作任何面向集的条件的参数的关系的名称。

[0344] 源自：ArgumentValue

[0345] 包括在该类型中的属性：

[0346] ● RelationshipSchemaName：包含可在其中发现 Relationship 类型的模式的名称。

[0347] ● RelationshipName：包含 Relationship 类型的名称。

[0348] 添加到该类型的方法：

[0349] ● RelationshipValue(stringrelationshipSchemaName，stringrelationshipName)：允许在构造时间指定模式名称和类型名称的构造器。

[0350] CollectionValue 源自 ArgumentValue 并被用于表示要用作对任何面向集的条件的参数的集合的名称。

[0351] 源自：ArgumentValue

[0352] 包括在该类型内的属性：

[0353] ● CollectionItemId：项集合的 Guid 项 id

[0354] ● RelationshipSchemaName：可在其中发现该集合内的 Relationship 的模式的名称

[0355] ● RelationshipName：包含在该集合内的 Relationship 类型的名称。

[0356] 添加到该类型的方法：

[0357] ● CollectionValue(Guiditemid，string relationshipSchemaName，stringrelationshipName)：允许在构造时间设定所有属性值的构造器。

[0358] RuleValue 源自 ArgumentValue 并被用于表示要被作为 LeafCondition 内的参数

评价和使用的某个 RuleLogic。当 RuleValue 用作任何面向集的条件的参数时，期望指向 AutoList（否则应生成异常）。当 RuleValue 用作为任何内建的标量条件（“＝”，“＜”等等）的参数使用时，它可以指向任何 RuleLogic。RuleValue 也可以返回布尔逻辑值。

[0359] 源自：ArgumentValue

[0360] 包括在该类型内的属性：

[0361] ● RuleId：规则项的 Guid 项 id

[0362] 添加到该类型的方法：

[0363] ● RuleValue(GuidruleItemId)：允许在构造时间设定所有属性值的构造器。

[0364] OpathValue 允许将 OPath 表达式作为 LeafCondition 的 LHS 或 RHS 来使用。

[0365] 源自：ArgumentValue

[0366] 添加到该类型的属性：

[0367] ● OPath：OPath 表达式

[0368] ● DisplayName：在询问构建器或其他 UI 中显示的名称。

[0369] 添加到该类型的方法：

[0370] OpathValue(string exp，string displayName)：允许在构造时间设定所有属性值的构造器。

[0371] 图 16H 示出本发明的 FunctionInfo 嵌套类型及其导出类型。FunctionInfo 是用于表示要被称为 Actions 的方法上的信息的基类型。FunctionInfo 从不在其自身上被使用；而是使用其孩子 InstanceFunctionInfo 和 StaticFunctionInfo。

[0372] 源自：(OS 文件系统).NestedType

[0373] 添加到该类型的属性：

[0374] ● MemberName：要被调用的方法的名称或其“set_”方法应该被调用的属性的名称。

[0375] 添加到该类型的方法：

[0376] ● FunctionInfo(string memberName)：允许在构造时间设定成员的名称的构造器。

[0377] ● internal virtal void Execute(ItemContext context，List<ArgumentValue>arguments)：在孩子类中该方法被忽略，以执行有名的方法调用。

[0378] InstanceFunctionInfo 源自 FunctionInfo 并被用于表示在特定项实例上的方法调用。

[0379] 源自：FunctionInfo

[0380] 添加到该类型的属性：

[0381] ● TargetItemId：应在其上调用方法的项的项 id。注意，当按照将方法调用编码成 ActionResult 元素时，如果该方法调用应该在输入项上作出，你将使 TargetItemId 为空。如果该调用应该在某一特定项上做出，该项的 ID 应该在这里设定。

[0382] 添加到该类型的方法：

[0383] ● InstanceFunctionInfo(string memberName)：允许在构造时间设定成员的名称的构造器。

[0384] ● InstanceFunctionInfo(string memberName，Guid targetItemId)：允许在构

造时间设定成员的名称和作为目标的项 id 的构造器。

[0385] ● internal void Execute(ItemContext context, List<ArgumentValue>arguments) :执行由 InstanceFunctionInfo 数据规定的方法。注意, context.Update() 在结束处不被调用 ;调用者关心 :如果方法调用对内存中的任何 OS 文件系统数据有副作用,哪一个重要。

[0386] StaticFunctionInfo 源自 FunctionInfo 并被用于表示静态方法调用。

[0387] 源自 :FunctionInfo

[0388] 添加到该类型的属性 :

[0389] ● AssemblyName :在其中可以发现方法的汇编组件 (assembly) 的名称。

[0390] ● TypeName :在其上定义该方法的汇编组件内的类型的名称。

[0391] 添加到该类型的方法 :

[0392] ● StaticFunctionInfo(string assemblyName, string typeName, stringmemberName) :允许在构造时间设定 AssemblyName、TypeName 和 MemberName 的构造器。

[0393] ● Internal void Execute(ItemContext context, List<ArgumentValue>arguments) :执行由该 StaticFunctionInfo 指定的方法。注意, context.Update() 不在结束处被调用 ;调用者关心 :如果该方法调用对内存中的任何 OS 文件系统数据有副作用,哪一个重要。

[0394] 许多 non-File System(非文件系统) 类型是 OS 文件系统规则 API 的一部分。

[0395] RuleInput 被用于定义和提交输入给平台。提交的输入将触发依附于事件发生的范围的 Rules 的评价。为了提交输入,使用存在于 DecisionPoint 类型上的提交方法。

[0396] 该类型的属性 :

[0397] ● Int64 Id :RuleInput 的 id。RuleInput 被成功提交后,将由平台分配给它 Id。

[0398] ● Guid EventDataItemId :在其上发生事件的 WinFS 项的 itemid。

[0399] 该类型的方法 :无。

[0400] RuleInputCollection 源自 System.Collections.CollectionBase。它允许多个 RuleInput 同时提交。生成的 RuleSetEvaluation 项包含来自该批内所有提交的事件的组合结果。提交空的 RuleInputCollection 是合法的。返回的 Guid 将被设定为 Guid.Empty。

[0401] 该类型的属性 :无。

[0402] 该类型的方法 :

[0403] ● RuleInput this[int index] :允许使用数组索引存取器来访问包含的 RuleInputs 的索引器。

[0404] ● int Add(RuleInput value) :将指定的 RuleInput 添加到集合。如果 RuleInput 已经在该集合内,则刨除 InvalidOperationException。

[0405] ● bool Contains(RuleInput value) :如果集合包括给定的 RuleInput,则返回为真,否则为假。

[0406] ● int IndexOf(RuleInput value) :返回集合中的给定的 RuleInput 的索引。

[0407] ● void Insert(int index, RuleInput value) :在指定的索引处,将值插入集合内。

[0408] ● void Remove(RuleInput value):从集合中移除给定的文件系统规则事件。

[0409] RuleException。当这些API中的一个失败时,抛出异常。当应用为标准情况描述的“正常的”异常时,或者是系统异常就是想要抛出的异常时,使用该标准异常。例如,

[0410] ArgumentNullException

[0411] ArgumentOutOfRangeException

[0412] InvalidOperationException

[0413] 如果该密匙(key)与集合内的条目不匹配,OS文件系统规则API索引器(C#“this”属性)抛出ArgumentOutOfRangeException。索引器返回合法的对象或抛出异常,并且不以null(空值)返回,如System.Collections.Hashtable所做的那样。发出OS文件系统规则或SQL客户端检测到的其他错误作为RuleException。

[0414] 样本代码

[0415]

```
1   using(ItemContext ic) {
2       Folder myFolder = … ;
3
4       // create a simple QueryFilter for Documents starting with F
5       QueryFilter qf = new QueryFilter(typeof(Document), new
LogicStatement(
6           new LeafCondition("like", "Title", "F%"),
7           new BooleanResult(true)));
8       myFolder.OutFolderMemberRelationships.Add(new
FolderMember(qf));
9
10      Folder myDocumentsFolder = … ;
11      // create an AutoList of Documents bigger than 512k in the above
folder
12      AutoList al = new AutoList(typeof(Document), myDocumentsFolder,
new LogicStatement(
13          new LeafCondition(">", "Size", "524288"), new
```

[0416]

```
BooleanResult(true)));
   14      myFolder.OutFolderMemberRelationships.Add(new
FolderMember(al));
   15
   16      // now, modify our AutoList to use the QueryFilter as well
   17      al.Statements.Add(new LogicStatement(new RuleValue(qf), new
BooleanResult(true)));
   18      ic.Update();
   19
   20      ItemSearcher i = al.GetItemSearcher();
   21
   22
   23      // Declare a Decision Point
   24      DecisionPoint dp = new DecisionPoint("My Application",
   25         new LogicConstraint(typeof(Document),
OutputType.FunctionInfo),false);
   26      myFolder.OutFolderMemberRelationships.Add(new
FolderMember(dp));
   27
   28      // Create an AppRule "attached" to this DP
   29      AppRule ar = new AppRule(dp.Constraint, new LogicStatement(
   30         new LeafCondition("like", "DisplayName", "WinFS Rules%"),
   31          new ActionResult(new StaticFunctionInfo("System.Xml",
"System.Xml.XmlConvert",
   32    "EncodeName"))));
   33      ar.DecisionPoint = dp.ItemId;
   34
   35      // add additional LogicStatments
   36      ar.Statements.Add(new LogicStatement(new LeafCondition("<",
"Size", "128"),
   37          new ActionResult(new InstanceFunctionInfo("set_Comment"),
"Small Document")));
   38
   39      myFolder.OutFolderMemberRelationships.Add(new
FolderMember(ar));
```

[0417]

```
40
41      RuleInput r;
42      EvaluationResults er = dp.Submit(r);
43      foreach(EvaluationResultElement ere in er.Results) {
44          … ;
45      }
46  }
```

[0418] LeafConditions 内的 Operator Discovery。开发者能发现支持哪些算符的一种方法是具有所有合法的算符的枚举 (enum)。其益处在于开发者能快速地发现所有合法的算符。错误拼写的算符的问题也可以被避免。另一种方法是为每个算符定义唯一的类，例如，BetweenOperator(valuel,value2,value3),GreaterThanOperator(valuel,value2) 等等。

[0419] 在另一种优化中，QueryFilters 和 AutoLists 在其逻辑语句中可只有布尔结果。API 可通过创建特定的 BooleanStatement 类型，确保人们在 QueryFilters 和 AutoLists 内只有布尔结果，与用于 AppRules 的语句分离。

[0420] 下面是对终端用户的询问和过滤器、以及用于执行它们的 API 机制的描述。当代表终端用户构造询问时，或者当构造终端用户其后会看见、使用并推论的询问时，构建对象 QueryFilter 和 AutoList。

[0421] 由项的范围和运用于那些项上的 QueryFilter 来定义询问。询问的结果是项的集，实际上，终端用户在询问和实际的 List 之间看到很少的不同。因此，该用户对“询问”的概念指数据的特别的动态过滤的过程，当被持续时，就成为 AutoList，它是其内容根据要求被计算的列表。术语 List 和 AutoList 互换地使用。

[0422] 终端用户使用两种视觉机制来构造询问。第一，导航 UI，通过它，终端用户能通过执行以下的导航任务中的任何一个来运行询问：打开页面以显示 List 或 AutoLis 的内容、按照某一属性或多个属性来堆栈或分类列表（或自动列表）内的项、以及“引入”(dirilling into) 某一栈。第二，提供给终端用户的丰富的询问 UI 能通过使用集成在列表察看器页面中的询问构建器 UI，方便合成更为丰富的询问过滤器。

[0423] 询问构造不是一次性的过程，而是递增的合成的过程。例如，合成的堆栈允许用户按照标题堆积文档，然后引入 (drill into) 一个标题内。然后按照日期堆积剩余的文档，并且他或她引入一年。

[0424] 过滤器的合成。用户定义过滤器，如“ModifiedLastMonth”（上个月修改的）作为具有在（[Today] 至 30 天）和（[Today] 至 60 天）之间的 ModifiedDate（修改日期）的文档。可以创建它来运行并且可能存储在询问构建器 UI 中。

[0425] AutoList 的合成。用户按照“所有的项”的范围中具有关键词“Important”（重要的）的所有文档来定义“MyImportantDocuments”（我的重要文档），并将其存储为自动列表。用户通过打开视图，导航到 MyImportantDocuments 而启动“MyImportantDocuments”作为用于新的自动列表的输入范围。用户使用 Query Build UI 在“MyImportantDocuments”上使用“ModifiedLastMonth”询问过滤器，然后将产生的自动列表存储为“ImportantDocumentsUpdatedLastMonth”（上个月更新的重要文档）。

[0426] 多语句过滤器。用户使用询问构建UI来创建过滤器，如‘music type IS jazz’（音乐类型为爵士乐）。用户添加额外的过滤器“OR”‘music artist IS JoniMitchell’（音乐家为 Joni Mitchell)。该用户添加额外的过滤器“AND”‘musicalbum title does NOT CONTAIN “Original Soundtrack”’（音乐集标题不包含“原声道”）。

[0427] 随着时间流逝，询问可被存为 AutoList，被进一步检索和编辑。询问中使用的过滤器也可被用于终端用户编程的其他方面——用于定义丰富的动作和丰富的事件。最后，AutoList 和过滤器自身可被询问（例如，为我找到使用“MyDocuments”（我的文档）列表的 AutoList，因为我正准备删除该列表）。

[0428] 用户查看的询问结果。显示信息落入两种类别。

[0429] 询问元数据。显示所选择的项的特定的终端用户属性。终端用户能向作为询问结果的一部分显示给用户的列的列表中添加或从中移除特定的属性。这些显示细节统称为视图定义 (View Definition)。相同的询问或 AutoList 在概念上对每个应用程序和 / 或每个用户来说，可能有不同的查看定义。

[0430] 应用程序元数据。应用程序也可以将应用程序特定的数据与 ViewDefinition 相关联，以捕获用于应用程序的上下文的元数据（例如，查看 AutoList 时的背景颜色）。

[0431] 为了在丰富的导航 UI 中显示 AutoList，需要一种灵活的且功能强大的光标模块。询问或 AutoList 的显示元素对询问定义没有影响（询问定义是对询问的内容的逻辑的和说明性的定义）。

[0432] 下面是终端用户的情况。在第一种情况下，当用户过滤直到他们关心的项目，将其存储为 AutoList，并随后返回到该集时，保存和重新使用提供的功能。本地视图持久性允许用户返回其中在 AutoList 被保存时提供相同的视图的存储的 AutoList。提供了一种跨应用程序的用于询问的持久性的公共平台，从而选择查看在应用程序内定义的“People I’m meeting with”（我正与之会谈的人）AutoList 的第二用户能够浏览这个询问并看到谁将出现在此会谈上。一般来说，该用户能够看见并浏览有许多不同的应用程序创建的 AutoList。

[0433] 在第二种情况下，提供了询问合成，因此第二用户能够看见对将要出现在该会谈上的客户的所有报告。第二用户创建询问以发现由在“People I’m meetingwith”AutoList 内的人所创作的所有文档。为了分布 AutoList，第二用户创建一 AutoList 用于发现重要的客户报告（基于在他的机器上的项）并希望他的团队使用。第二用户为其团队内的每个人分配该报告，以存储为 AutoList，并针对他们各自的机器上的数据而在他们自己的机器上运行。为了分布视图定义，当第三用户打开从第二用户存储的 AutoList 时，她得到带由第二用户创建的过滤器图标、分类顺序和分组的相同的列。

[0434] 在第三种情况下，可以共享本地询问。第三用户有她希望第二用户看到的一组项。她共享对该组项的存储的 AutoList（在她自己的机器上）的访问。当第二用户和第三用户打开这个 AutoList 时（除非是安全限制），他们看到相同的该组项。在第三用户从第二用户那里得到指向一应用程序的 AutoList 的情况下，可提供一种得体的失败 (graceful failure)。第三用户没有这个应用程序。因此，AutoList 改为在 All Items（所有的项）内打开。

[0435] 合成多个 QueryFilter 和 AutoList 以构造新的 QueryFilter 和 AutoList 是可能

的。使用 AutoList 作为询问的范围也是可能的。

[0436] 现在参考图 17，示出了依照本发明的规则引擎 1700。揭示的规则平台包含接收输入并评价适当的规则的规则引擎 1700。为了调用规则引擎 1700，应用程序使用规则 API（也叫做规则评价 API）通过判定点来提交数据。该 API 提交数据到规则评价引擎，并通过 API 返回结果。执行模型是同步的并且返回的结果是简单的结构化的数据。该规则引擎 1700 使用询问处理器评价规则，并确定运行哪些规则，针对输入评价规则，然后返回结果。除作为用于开发者的存储器平台以外，用于文件系统的该 EUP 模型和抽象是整个平台的重要部分。

[0437] 现在参考图 18，示出了在本发明的规则架构内表示丰富程度的尺度的框图 1800。提供的能力落入三个一般的类别内：行为 (Behavior)、结构 (Structure) 和控制流 (Control Flow)。有关 Behavior，为两种基本类型的客户化使用终端用户规则：定义数据（存储的询问）的导出规则，和定义行为的活动规则。导出规则定义数据上的丰富询问，并且定义虚拟的数据集合（“导出的项集”），这些集合象实际的文件系统集合一样行为，用于询问和规则的目的。例如，终端用户可能定义一组叫做“”的联系人作为导出的项集。以后，用户可询问电子邮件以发现由“PeopleWhoLiveInBellevue”发出所有邮件。

[0438] 终端用户使用活动规则来定义在明确定义的判定点处的应用程序的行为，在那里应用程序允许终端用户用逻辑来扩充应用程序流。存在两种类型的判定点。同步判定点是应用程序到达判定点并发起调用终端用户规则的同步事件的地方。规则评价 API 确定一组结果。然后，应用程序根据结果而动作，可能改变其行为。异步判定点是应用程序到达判定点，发起异步事件并以“引发后忘记”(fire and forget) 的方式继续进行处理的地方。规则平台评价任何适当的规则，然后导致任何相关的结果作用于其上。

[0439] 关于图 18 的结构块，可在文件系统数据模型的所有方面写规则，以包括项类型和项类型的以下方面：作为简单嵌套元素的属性、作为嵌套元素的多集的属性、来自项的关系的端点、以及项上的扩展。也可以为关系类型以及其中的以下方面写规则：作为简单嵌套元素的属性、和作为嵌套元素的多集的属性。关于扩展，这些行为实际上类似项类型。

[0440] 关于可用函数 (Available Functions) 块，规则内的条件和动作与函数调用类似。最常见的函数是内建的算符，如“＝”等等。条件内支持任何 T-SQL(Transact-Structured Query Language)（处理－结构查询语言）可调用的函数，并且在动作内支持任何客户端可调用的语言。支持的函数是原语数据类型上的 SQL 算符，内建的 SQL 函数、用于定量 ANY_IN 和 ALL_IN 的集合算符、用于条件的存储侧的汇编组件中定义的 CLR 方法（如果注册为 UDF）、以及用于动作的客户端汇编组件中定义的 CLR 方法。

[0441] 关于合成块，除文件系统模式中定义的“元素”外，文件系统规则代数支持合成。RuleSetAttachment 是第一类对象，因为它可用于条件，或者作为另一个附件的范围。

[0442] 合成的另一种形式涉及使用强类型的 Derived Rules 作为函数。考虑，例如，定义一个或多个调用的“ISInteresting”，它对 Document 项起作用。那么，可以创建使用“IsInteresting”作为条件函数的规则。

[0443] 关于控制流块，提供控制流给使用冲突解决策略的终端用户。因为这些是复杂的代码段，因此应用程序开发者构造它们。终端用户只从可用的策略中进行选择。

[0444] 额外的控制流可在规则平台外程序地实现。既然可与规则 API 一起来使用文件系

统规则架构，开发者能使用标准的 C# 控制流构造。

[0445] 规则丰富程度

[0446] 现在参考图 19，示出了表示本发明的规则架构的输入的示意图 1900。涉及的是条件和动作作为输入作用于数据。输入的丰富程度沿着两个维度：输入的位置（例如，在存储器内或不在存储器内），以及输入的基数（单个输入或者输入集）。标准的情况使用以下的组合，显示在下面的表中。

[0447]

情况	输入的位置	输入的基数
活动规则	在存储器内（项类型）	集
应用程序客户化	在存储器内 / 不在存储器内（项、原语、XML）	单个 / 集
现在应用	在存储器内（项）	单个 / 集
导出的项集	在存储器内（项）	集
参数化项集	不在存储器内（原语）	单个

[0448] 在该例子中，输入是存储器内的消息 (Message) 项。除文件系统项以外，为 Relationship 和 XML 作为 Input 提交给规则引擎提供支持。规则本身被标量值参数化，要求在规则评价之前提供自变量。这些自变量叫做规则参数。

[0449] 条件和动作的每一个有操作数。参考下面的例子。

[0450] scalar_value = scalar_value// 两个标量

[0451] scalar_value = COUNT(set_value)// 聚集函数将集变换成标量

[0452] scalar_value_IN set_value// 标量和集

[0453] set_value_ALL_IN set_value// 两个集

[0454] 依据条件函数，每一个操作数是标量表达式或是集。

[0455] 标量表达式是其值为标量类型的表达式。支持以下的标量表达式：常量（例如 1、“Hello”）；输入的属性，包括不是多集的项类型或者关系类型的属性（例如，[Document].Title），并且如果该属性具有嵌套属性，这些也是被支持的；以及集上的聚集函数。也可以使用规则参数（以上所述）。

[0456] 基于集的聚集函数与项集、关系和嵌套元素的多集（例如，Count、MIN、MAX、SUM 和 AVG）相关。Count 是提供其自变量集内成员数量的计数的聚集函数。

[0457] 例子

[0458] Count(<ItemSet>) 计数 ItemSet 中的项数。

[0459] Count(Message.Recipients) 是提供源为该 Message 的 Recipient 关系的数量的计数的聚集函数。

[0460] Count(Message.OriginalDeliveryAccount.EAddresses) 是提供 Eaddresses 嵌套元素集中地址数量的计数的聚集函数。

[0461] 支持与 Opath 相同的聚集函数的集，它是跨整个文件系统平台的相同级别的可表达能力。

[0462] 模型中支持开发者定义的聚集函数。可扩展性也可被实现。

[0463] 类似 ANY_IN 的一些算符有作为操作数的集。集可被构造成标准集合或导出的项集。标准的文件系统集合包括关系和用于该关系的源项（例如，GroupMembership 和 MyFriends）。作为重要的特例，输入项是关系源（例如 Message.Recipient）或者关系目标（例如，Message.AttachmentOf）。作为输入项内的嵌套元素的多集，以及作为在输入项上的扩展集。

[0464] 通过将一个导出规则或多个导出规则与输入项范围相关联，来定义导出的项集。这种关联以 RuleSetAttachment 来记录。例如，导出规则 FriendlyPerson 依附于 MyContacts 作为其输入项范围，来定义导出的项集 MyFriends。导出项集可完全限制于输入范围或者只是是部分限制。完全限制的导出的项集的内容可以被枚举。部分限制的导出的项集要求提供一些或全部输入用于评价。每个导出规则也直接是部分限制的导出的项集。需要输入项以便被评价。例如，FriendlyPerson 是部分限制的项集合。这个导出可在条件内称为要在输入项上运行的过滤函数。仅部分定义规则的输入的 RuleSetAttachment 也定义部分限制的项集。例如，“People_By_Location”要求 Contact（联系人）输入项和 location（位置）作为输入。如果它依附于 MyFriends 作为其输入范围，它就变成部分限制的导出的项集 Friends_In（由 MyFriends 输入范围和 People_By_Location 导出的项集之间的 RuleSetAttachment 表示限制 (binding)）。为了评价现在的部分限制的导出的项集“Friends_In”，提供 location（位置）。

[0465] 条件的丰富程度是由对该条件的输入、以及可用于该条件的函数来定义的。对条件的输入先前已经描述过了。条件内允许的函数如下：具有标量表达式自变量的标准比较条件，比较条件包括比较算符：=、<>、>、<、<=、>=、BETWEEN 和 IN，以及支持 LIKE 和 CONTAINS 的串；与输入类型相关的存储侧成员函数，如果它们返回 SqlBoolean；采用输入项作为参数的存储侧静态函数，如果它们返回 SqlBoolean；以及基于集的条件。

[0466] 对于如先前所述的 ItemSets 的灵活定义，支持下面基于集的条件。相关的项集的一个重要类别是由以输入项作为源或作为目标的关系定义的那些条件。部分限制的导出的项集分别被考虑。

[0467] IN<TargetItemSet>。例如，<input_item>IN MyDocuments

[0468] <ItemSet>ANY_IN <TargetItemSet>。如果 ItemSet 内的任何项也在 TargetItemSet 内，则返回为真。例如，Recipient ANY_IN MyFriends。如果 IN 可选子句不存在，如果 ItemSet 内至少有一个项，则返回为真。这个条件以短形式表示为 EXISTS(<ItemSet>)。

[0469] 。如果 ItemSet 内的每个项也在 TargetItemSet 内，则返回为真。例如，Recipient ALL_IN MyFriends。

[0470] 这三个基于集的条件在两种其他的情况下也是支持的。部分限制的目标项集 (TargetItemSet 不是完全限制的；正是项集在该条件内的使用，才提供了完整的输入限制）。可以使用该类条件，针对与输入相关的项集内的项的属性，来表示丰富的条件。例如，ISANY Recipient A FriendlyPerson。该类条件也可以用于针对文件系统内的其他项

集，表示丰富的“join”条件。例如，ISANY RecipientIN Friends_In(‘Bellevue’) 以及 EXISTS(Email_By_Subject(Message.Subject))。

[0471] 关于第二种情况，虚拟目标关系集，当输入是关系（不是项）时，依然可以将基于集的条件用于由针对关系工作的规则所定义的目标集。假定采用 Message.Recipient 关系作为输入并确定电子邮件地址是否来自 Company 域的 CompanyAccount，例如，ISANY RecipientRel A CompanyAccount。

[0472] 从 RuleConstraint 可以知道允许的结果列表。这些结果的每一个有名称和已知的类型化的自变量列表。终端用户为每个规则挑选一组结果。每个结果由名称和自变量绑定来指定。这些自变量可被限制于任何标量表达式。此外，终端用户能定义数据驱动的结果集，如下：

[0473] FOREACH<ItemSet>[IN<TargetItemSet>]<Result...>

[0474] 例如 FOREACH Recipient IN MyFriends SendMail(...)

[0475] 例 如 FOREACH Msg IN InBoxMessages_By_Subject(“rules”)Msg.SetPriority(1)

[0476] FOREACH 构造内的 Result 是为这个规则的任何允许类型的结果。提供给这些结果的标量自变量包括由 FOREACH 子句指定的项的属性，以及规则输入的属性。

[0477] 规则共同定义用户想要的逻辑功能。下面是关于怎样评价规则、冲突解决如何给终端用户控制流构造的可选方案的描述。

[0478] 规则执行使用下面的用于某一特定输入项的逻辑语句：评价所有规则的条件，并且对于条件为真的那些规则跨结果解决冲突。如果有一组输入，评价的结果简单地是针对该输入范围内的每个个体的项 / 关系的规则评价的结果的联合。当针对一规则提交多个项时，没有特殊的语义。逻辑上，每个项被独立评价。来自不同输入的结果从不与其他结果冲突。

[0479] 冲突解决策略提供规则的表达能力的丰富性。支持不同的冲突策略。终端用户定义的策略方便终端用户用停止 - 处理选项对规则的优先级赋值。终端用户给每个规则分配唯一的优先级。如果也设定了停止 - 处理标记，这指示如果该规则的条件评价为真，那么所有具有较低优先级的规则都应被忽略。开发者定义的策略包括内部结果优先级赋值的能力，其中开发者可以为每个结果类型指定唯一的优先级。在可能的结果中，只有最高优先级的结果被保留。此外，开发者定义的策略方便基于内部结果聚集的冲突解决，其中开发者可以指定在相同结果类型的一组结果上起作用并的聚集函数，并聚集它们。

[0480] 使用冲突解决，支持下面的规则行为：规则语句通过伴随停止 - 处理选项，对每个规则分配唯一的终端用户优先级，象嵌套的 IF-THEN ELSE(IF THENELSE (...)) 表达式一样行为；规则语句可通过使用没有停止 - 处理选项的规则优先级，象具有失败跨越选择语句 (fall-through across case statement) 的 C++ 开关语句一样行为；规则语句可通过使用适当的内部结果冲突解决策略，提供自由复制的联合和交集语义；规则语句可通过正确的内部结果冲突解决策略，提供基于投票的语义（至少三个规则同意该结果）；以及规则语句可通过内部结果的优先级赋值提供丰富的跨结果语义。

[0481] 通过文件系统商业逻辑，能在条件和动作内提供的值上添加约束。

[0482] 使 用 LeafCondition 嵌 套 元 素 类 型 可 以 表 示 Condition(条 件)。 使 用

StaticFunctionInfo 和 InstanceFunctionInfo 可以表示 Function(函数)。自变量可以是 ConstantValue、PropertyValue、CollectionValue、RuleSetAttachmentValue 中的一个。

[0483] 使用 Action 嵌套的元素类型可以表示 Result(结果)。Action 元素内的函数和自变量与以上对条件描述的那些是相同的。一个实现中,存储器内支持多集。另一个实现中,LeafCondition 和 Action 最多支持 5 个自变量。

[0484] 现在参考图 20,示出适于执行揭示的结构的计算机的框图。为了为本发明的各方面提供附加的上下文,图 20 和下面的讨论意图提供了对适合的计算环境 2000 的简要的、一般的描述,其中可以实现本发明的各方面。尽管在上面的可在一台和多台计算机上运行的计算机可执行的指令的一般上下文中已经描述了本发明,本领域的技术人员应该理解,本发明也可以结合其他的程序模块和 / 或作为硬件和软件的结合来实现。

[0485] 一般来说,程序模块包括例程、程序、组件、数据结构等等,它们执行特殊的任务或实现特殊的抽象数据类型。而且,本领域的技术人员认识到,说明性的方法可以用其他计算机系统配置实施,包括单处理器或微处理器计算机系统、微型计算机、大型计算机、个人计算机、手提计算设备、就微处理器或可编程的消费者电子设备等等,其中每一种可以运转地耦合到一个或多个相关的设备。

[0486] 本发明的说明的方面也可以在分布式计算环境中实施,其中通过通信网络连接的远端处理设备执行特定的任务。在分布式计算机环境中,程序模块可以位于本地和远端存储器存储设备中。

[0487] 计算机一般包括各种计算机可读的介质。计算机可读的介质可以是任何可由计算机可读介质可以是由计算机访问的任何可用介质并包括易失的和非易失的介质、可移动的和不可移动的介质。作为例子,而非限制,计算机可读介质可以包括计算机存储介质和通信介质。计算机存储介质包括易失的和非易失的介质、可移动的和不可移动的介质,这些介质以用于信息存储的任何方法或技术,比如计算机可读指令、数据结构、程序模块或其它数据来实现。计算机存储介质包括但不限于,RAM、ROM、EEPROM、闪存或其它存储技术、CDROM、数字化视频光盘 (DVD) 或其它的光盘存储器、磁性磁带、磁性录音带、磁性磁盘存储器或其它的磁性存储器设备,或可用于存储需要的信息并可由计算机存取的任何其它介质。

[0488] 通信介质一般包括计算机可读指令、数据结构、程序模块或在一个调制数据信号比如载波或其它传输装置中的其它数据并且包括任何信息传递介质。术语“调制数据信号”意思是具有一个或多个它的特征集或以信号中编码信息的方式变换的信号。作为例子,而非限制,通信介质包括比如有线网络或直接有线连接的有线介质和比如声频、RF、红外线和其它无线介质。以上的任何组合也应该包括在计算机可读介质的范围内。

[0489] 现在再次参考图 20,示出了用于实现本发明的各方面的示例性的环境 2000,包括计算机 2002,计算机 2002 包括处理单元 2004、系统内存 2006 和系统总线 2008。系统总线 2008 将系统组件耦合到处理单元 2004,系统组件包括但不限于系统内存 2006。处理单元 2004 可以是任何商业上可用的处理器。双微处理器和其他多处理器结构也可以作为处理单元 2004 使用。

[0490] 系统总线 2008 可以是使用任何一种商业上可用的总线结构,进一步连接到内存总线 (用或不用内存控制器)、外围总线和局部总线的许多类型的总线结构中的任何一种。系统内存 2006 包括只读存储器 (ROM)2010 和随机存取存储器 (RAM)2012。基本输入 / 输出

系统 (BIOS) 一般存储在非易失的内存 2010 内，如 ROM、EPROM、EEPROM，BIOS 包括如启动时帮助在计算机 2002 内的元件间传输信息的例程。RAM2012 也包括告诉 RAM，如用于缓存数据的静态 RAM。

[0491] 计算机 2002 进一步包括内部的应聘驱动器 (HDD) 2014（例如，EIDE、SATA），内部的硬盘驱动器 2014 也被配置用于在适合的底盘（未显示）、磁性软盘驱动器 (FDD) 2016（例如，可读写的可移动的磁盘 2018）和光盘驱动器 2020（例如，可读 CD-ROM 盘 2022 或者可读写的其他高容量的光学介质，如 DVD）的外部使用。硬盘驱动器 2014、磁盘驱动器 2016 和光盘驱动器 2020 可以分别通过硬盘驱动器接口 2024、磁盘驱动器接口 2026 和光盘驱动器接口 2028 连接到系统总线 2008。用于外部驱动实现的接口 2024 至少包括一个或两个通用串行总线 (USB) 和 IEEE1304 接口技术。

[0492] 驱动器及其相关联的计算机可读的介质提供数据、数据结构、计算机可读的指令等等的非易失的的存储。对计算机 2002 说，驱动器和介质容纳任何适当的数字格式的任何数据的存储。尽管计算机可读的介质的描述涉及 HDD、可移动的磁性磁盘和可移动的光学介质，如 CD 和 DVD，本领域的技术人员应该认识到，计算机可读的其他类型的介质，如压缩驱动器、磁性磁带、闪存卡、录音带等等也可以用于示例性的操作环境，并且进一步的，任何这样的介质可包含用于执行本发明的方法的计算机可执行的指令。

[0493] 许多程序模块可以存储在驱动器和 RAM2012 中，包括操作系统 2030、一个或多个应用程序 2032、其他程序模块 2034 以及程序数据 2036。操作系统、应用程序、模块和 / 或数据的全部或一部分也可以缓存在 RAM2012 中。

[0494] 应该认识到，本发明可用各种商业上可用的操作系统或操作系统的结合来实现。

[0495] 用户可以通过一个或多个有线 / 无线输入设备，如键盘 2038 和定点设备，如鼠标 2040 向计算机 2002 输入命令和信息。其它输入设备（未显示）可包括话筒、IR 远端控制、操作杆、游戏垫、触针笔、触摸屏等。这些和其它输入设备通常通过耦合至系统总线 2008 的输入设备接口 2042 连接至处理单元 2004，但是也可以通过其它接口相连，如并行端口、IEEE1394 串行端口、游戏端口或 USB 端口、IR 接口等等。

[0496] 监视器 2044 或其它类型的显示设备也通过接口，如视频适配器 2046 连接至系统总线 2008。除监视器 2044 以外，计算机一般包括其他外围输出设备（未显示），如话筒、打印机等等。

[0497] 计算机 2002 可工作在使用通过有线和 / 或无线连接到如远程计算机 2048 的一个或多个远程计算机的逻辑连接的网络化环境内。尽管，为了简要的目的，只示出了内存存储设备 2050，然而远程计算机 2048 可以是工作站、服务器计算机、路由器、个人计算机、手提计算机、基于微处理器的娱乐设备、对等设备或其他公共网络节点，并且一般包括与计算机 2002 相关的许多或所有上述元件。所述的逻辑连接包括到局域网 (LAN) 2052 和 / 或较大的网络的有线 / 无线连接，例如，广域网 (WAN) 2054。这样的 LAN 和 WAN 网络环境常见于办公室、公司，并且方便了企业范围的计算机网络，如内联网，所有这些可以连接到全球通信网络，如因特网。

[0498] 当用于 LAN 网络环境时，计算机 2002 通过有线和 / 或无线通信接口或适配器 2056 连接到局域网 2052。适配器 2056 方便了到 LAN2052 的有线或无线通信，它也可以包括部属在其中用于与无线适配器 2056 进行通信的无线接入点。当用于 WAN 网络环境时，

计算机 2002 包括调制解调器 2058，或连接到 LAN 上的通信服务器，或者具有其他用于建立 WAN2054，如通过因特网的通信的其他装置。调制解调器 2058 可能是内置或外置的以及有线的或无线的，它通过串行端口接口 2042 连接到系统总线 2008。在网络化环境内，所述与个人计算机 2002 相关的程序模块或其中的一部分可存储在远程内存 / 存储设备 2050 内。应该认识到，所显示的网络连接是示例性的，在计算机间建立连接的其他装置也可以使用。

[0499] 计算机 2002 适于与操作性地布置在无线通信中的任何无线设备或实体通信，例如，打印机、扫描仪、桌面和 / 或手提计算机、手提数据助理、连接微型、与无线探测的标记相关的任何一种设备或位置（例如，公用电话亭、新的台子、休息室等），以及电话。这至少包括 Wi-Fi 和蓝牙 TM 无线技术。因此，这种连接可以是预先定义的结构，如传统的网络或简单的是在至少两个设备之间的独特的通信。

[0500] Wi-Fi，或无线保真，允许从家里的睡椅、旅馆的床上或者工作时的会议间，在没有线的情况下连接到因特网。Wi-Fi 时无线技术，类似使这种设备，例如，计算机发送和接收室内外的数据的小区电话；在基站范围内的任何地方。Wi-Fi 网络使用叫做 IEEE802.11(a、b、c 等等）的无线电技术来提供安全的、可靠的、快速的无线连接。Wi-Fi 网络可以用于彼此连接计算机到因特网、以及有线网络（它使用 IEEE802.3 或以太网）。Wi-Fi 网络工作在未经当局允许的 2.4 和 5GHZ 无线电频段，在 11Mbps(802.11a) 或 54Mbps(802.11b) 数据速率或者带有包含两种波段（双波段）的产品，因此网络可以提供类似用于许多办公室的基本的 10BaseT 有线以太网网络的实时世界的性能。

[0501] 现在参考图 21，示出了依照本发明的示例性的计算环境 2100 的示意性的框图。系统 2100 包括一个或多个客户 2102。客户 2102 可以是硬件和 / 或软件（例如，线程、进程、计算设备）。例如，客户 2102 可通过使用本发明来存储 cookie 和 / 或相关的文本信息。系统 2100 也包括一个或多个服务器 2104。服务器 2104 可以是硬件和 / 或软件（例如，线程、进程、计算设备）。例如，服务器 2104 可通过使用本发明来存储线程，以执行转换。客户 2102 和服务器 2104 之间的一个可能的通信使以适合于在两个或多个计算机进程间传输的数据分组的形式。例如，该数据分组包括 cookie 和 / 或相关的文本信息。系统 2100 包括通信框架 2106（例如，全球通信网络，如因特网），它可被用于方便客户 2102 和服务器 2104 之间的通信。

[0502] 通过有线（包括光纤）和 / 或无线技术可以方便通信。客户 2102 可操作地连接到一个或多个客户数据存储器 2108，该存储器可被用于存储本地于客户 2102 信息（例如 cookie) 和 / 或相关的文本信息。类似地，服务器 2104 可操作地连接到一个或多个服务器数据存储器 2110，该存储器可被用于存储本地于服务器 2104 的信息。

[0503] 以上讨论的东西包括本发明的例子。当然，不可能描述为了描述本发明的目的的组件或方法的可能想到的每个组合，但本领域的技术人员可以理解，本发明的许多进一步的组合和改变是可能的。因此，本发明意欲包括落入附加的权利要求的精神和范围内的所有这样的改变、修改和变化。而且，对于详细描述或权利要求中使用的术语“包括”的范围，这种术语意欲包括类似于术语“包含”，当“包含”在权利要求中作为转换的单词使用时的方式。

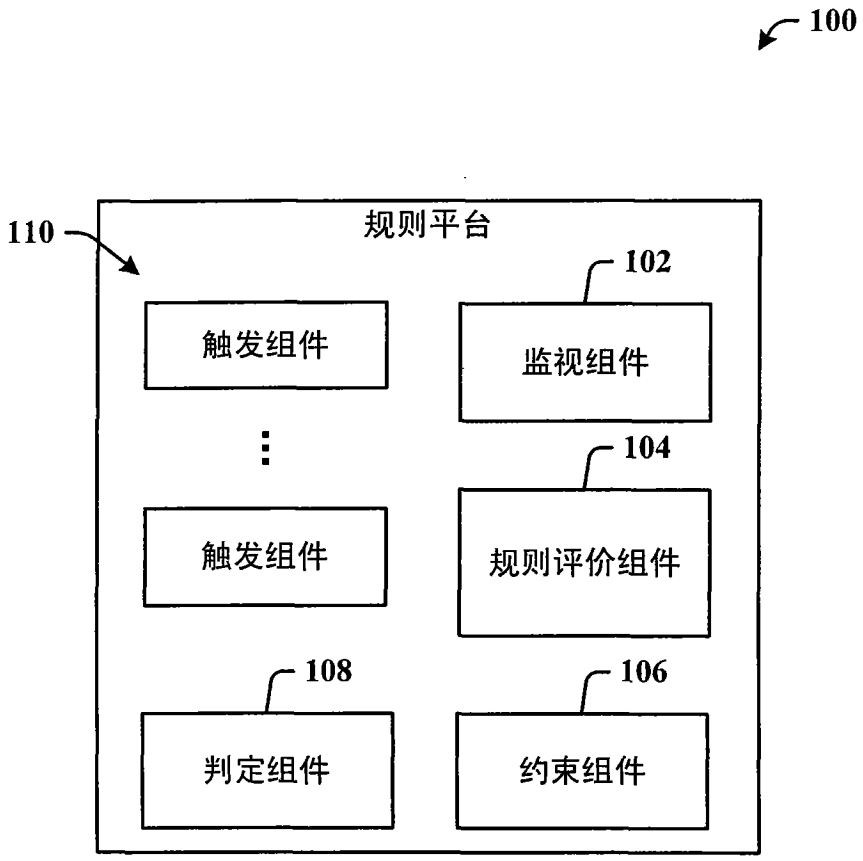


图 1

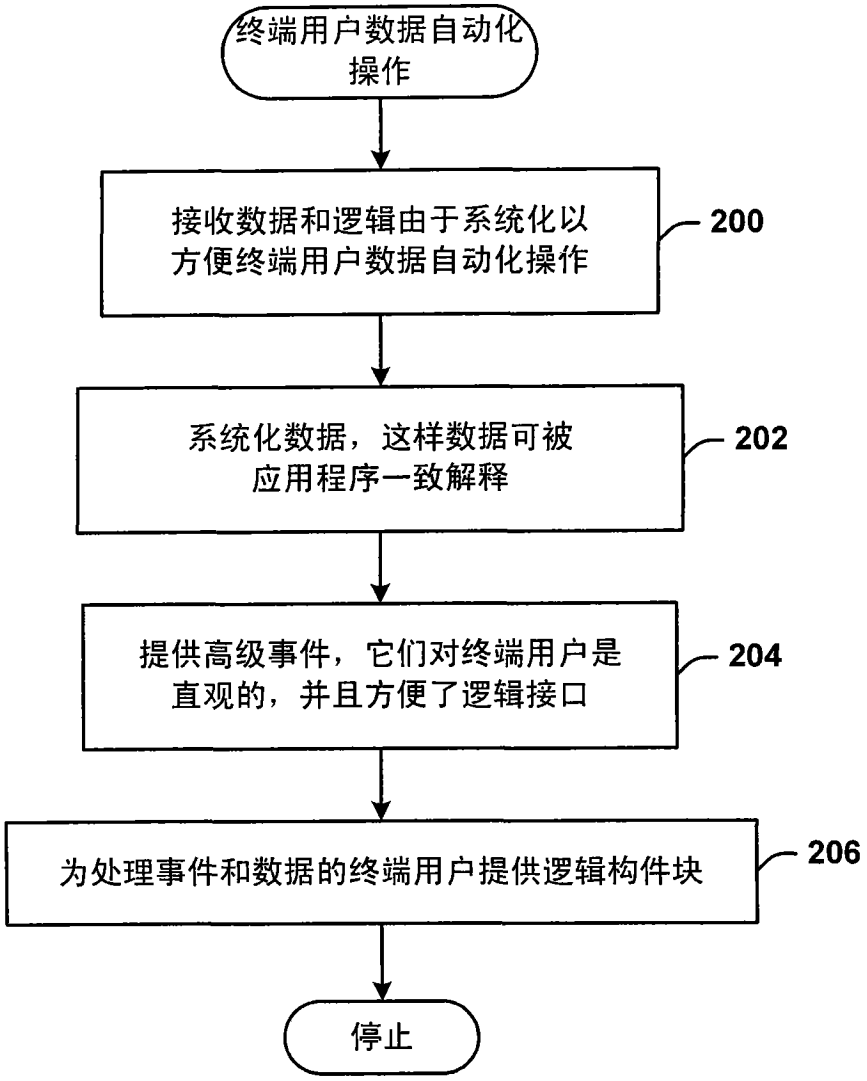

终端用户数据自动化操作
接收数据和逻辑由于系统化以方便终端用户数据自动化操作
200
系统化数据，这样数据可被应用程序一致解释
202
提供高级事件，它们对终端用户是直观的，并且方便了逻辑接口
204
为处理事件和数据的终端用户提供逻辑构件块
206
停止

图 2

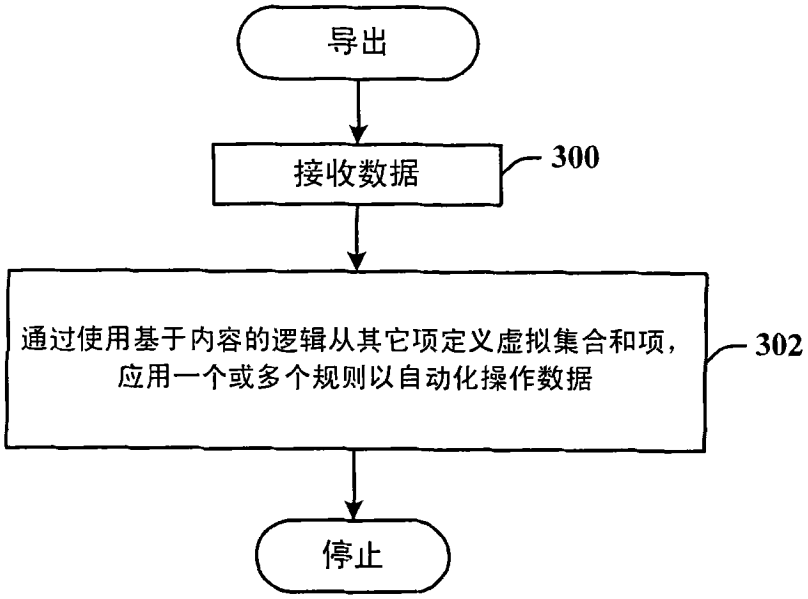

导出
接收数据
300
通过使用基于内容的逻辑从其它项定义虚拟集合和项，
应用一个或多个规则以自动化操作数据
302
停止

图 3

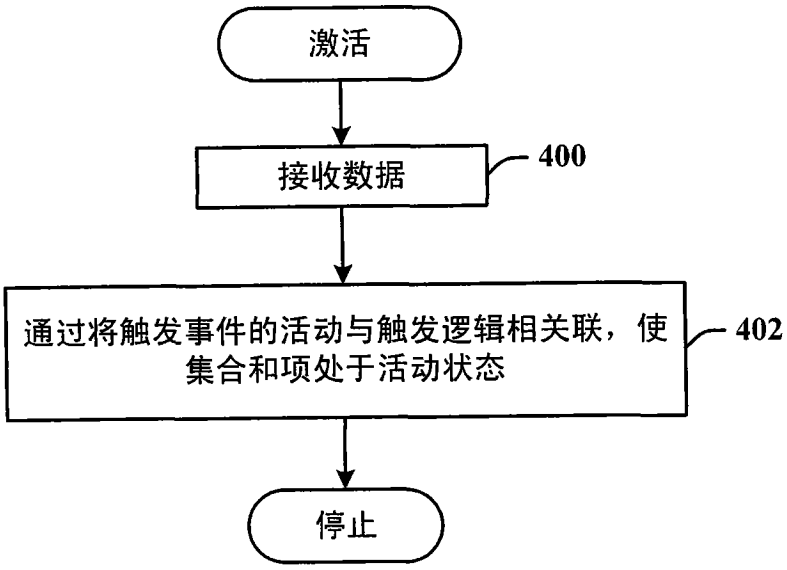

激活
接收数据
400
通过将触发事件的活动与触发逻辑相关联，使
集合和项处于活动状态
402
停止

图 4

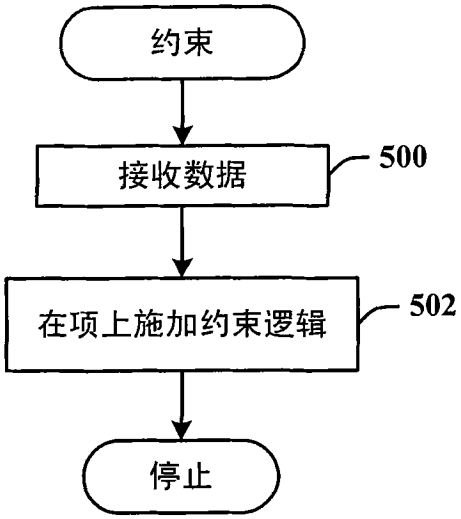

约束
接收数据
500
在项上施加约束逻辑
502
停止

图 5

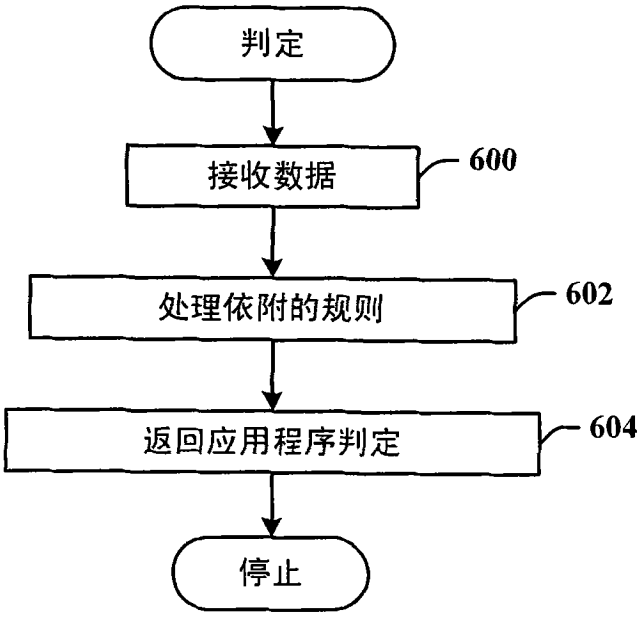

判定
接收数据
600
处理依附的规则
602
返回应用程序判定
604
停止

图 6

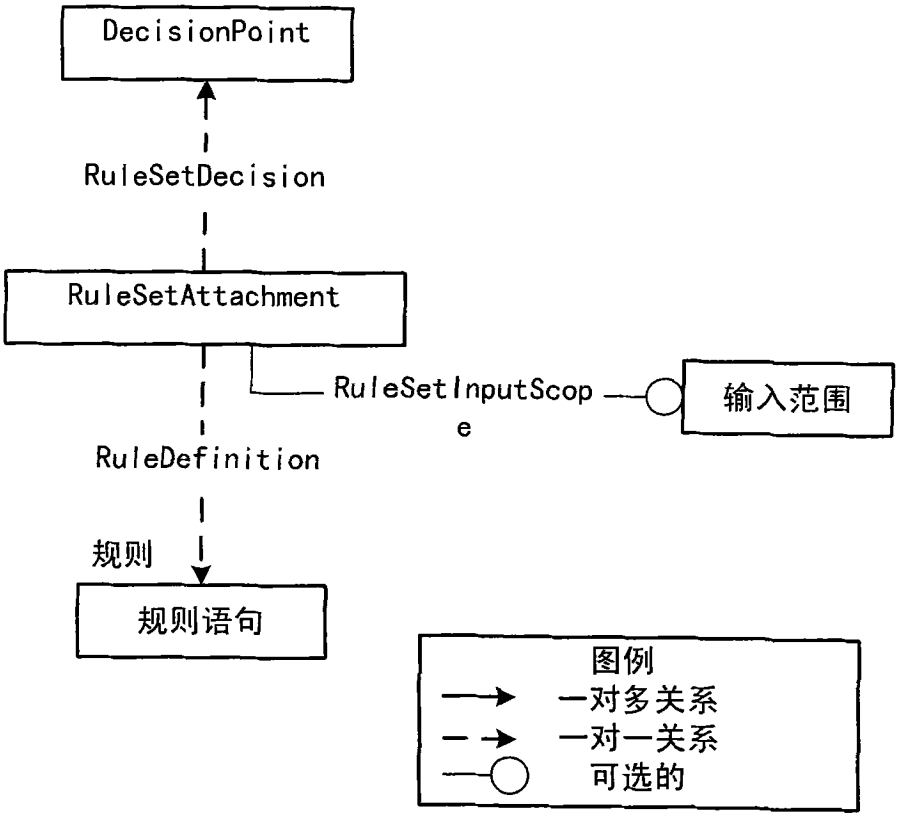

DecisionPoint
RuleSetDecision
RuleSetAttachment
RuleSetInputScop
e
输入范围
RuleDefinition
规则
规则语句
图例
一对多关系
一对一关系
可选的

图 7

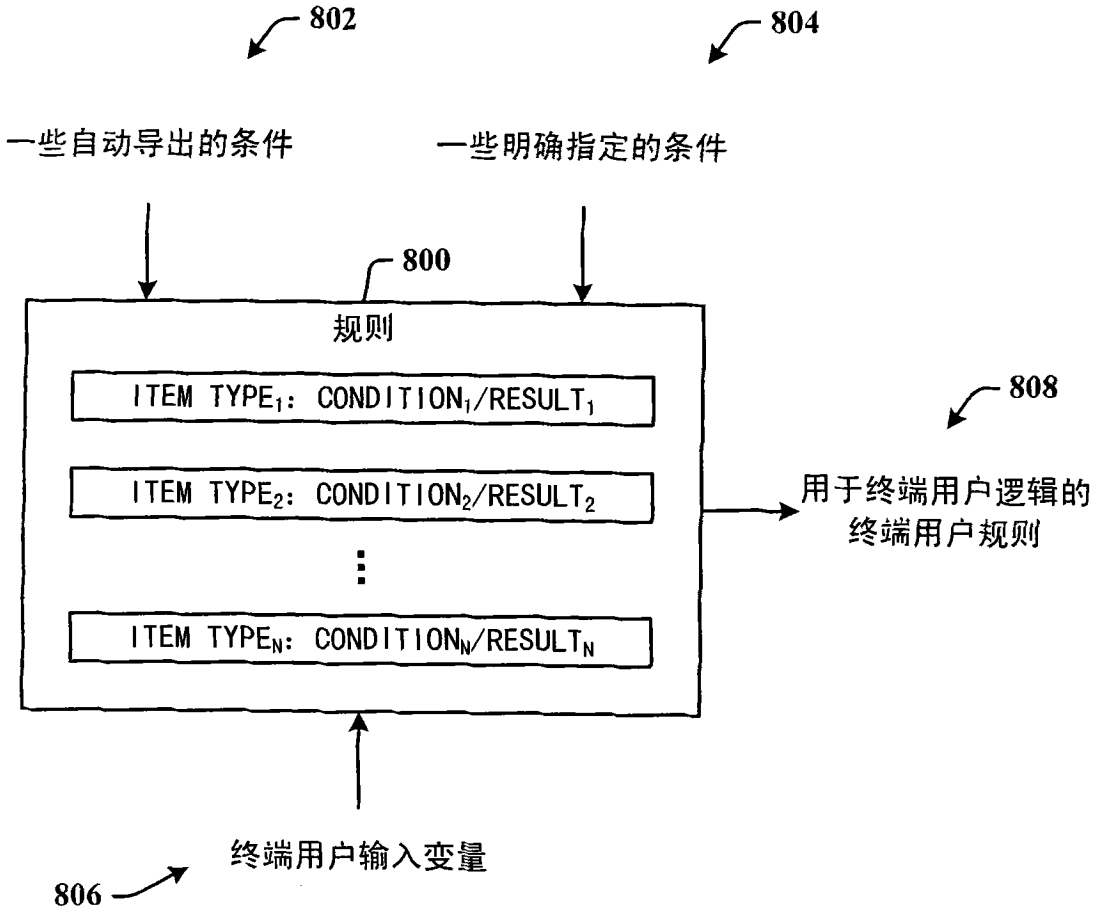


图 8

一般规则格式
ON（每个数据项）/IF（条件为真）/THEN（结论）

图 9A

活动逻辑规则格式
ON（每个数据项）/IF（条件为真）/THEN（动作）

图 9B

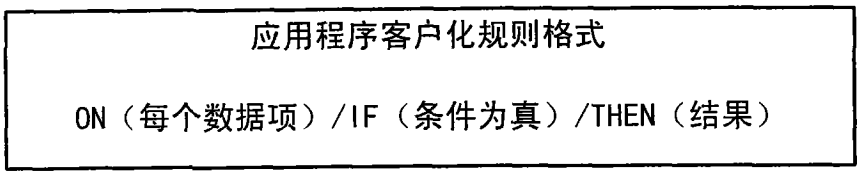

应用程序客户化规则格式
ON（每个数据项）/IF（条件为真）/THEN（结果）

图 9C

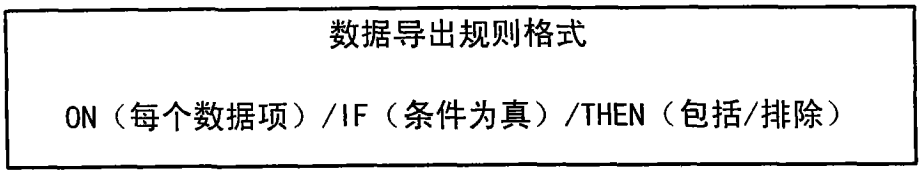

数据导出规则格式
ON（每个数据项）/IF（条件为真）/THEN（包括/排除）

图 9D

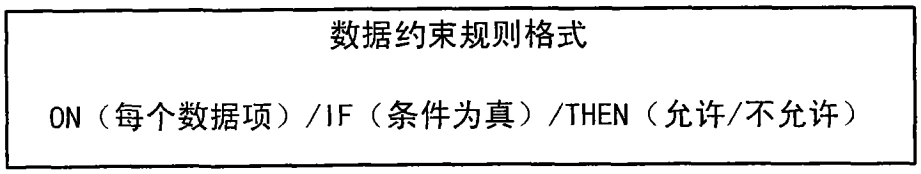

数据约束规则格式
ON（每个数据项）/IF（条件为真）/THEN（允许/不允许）

图 9E

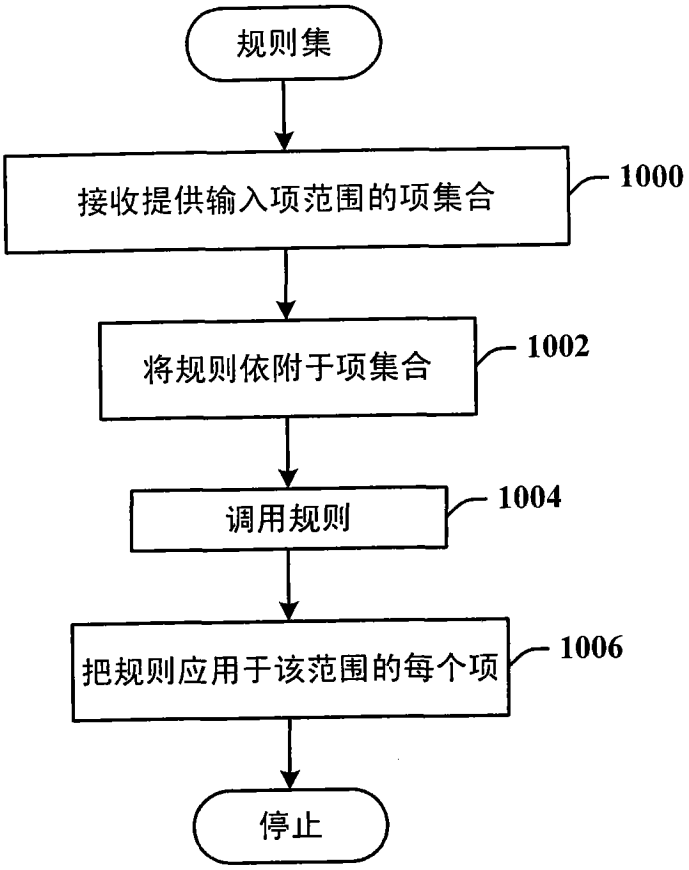

规则集
接收提供输入项范围的项集合
1000
将规则依附于项集合
1002
调用规则
1004
把规则应用于该范围的每个项
1006
停止

图 10

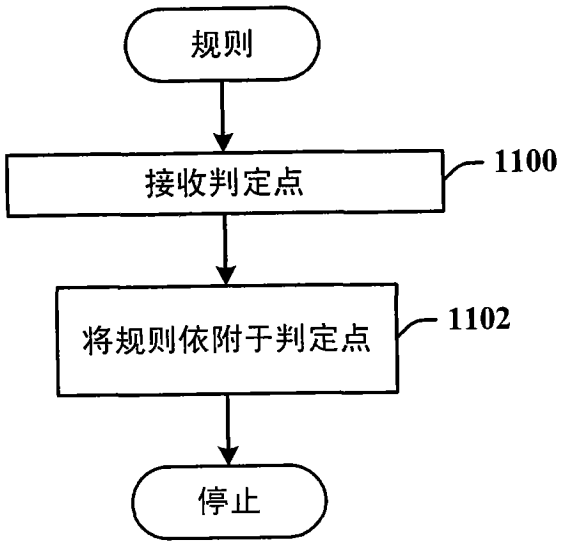

规则
接收判定点
1100
将规则依附于判定点
1102
停止

图 11

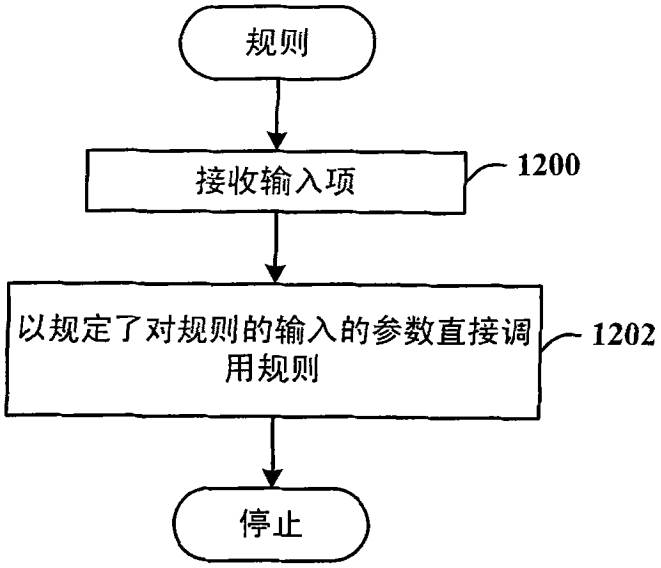

规则
接收输入项
1200
以规定了对规则的输入的参数直接调用规则
1202
停止

图 12

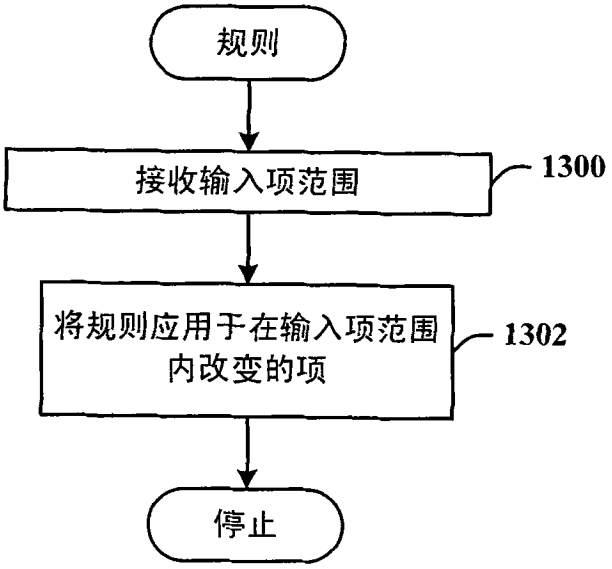

规则
接收输入项范围
1300
将规则应用于在输入项范围内改变的项
1302
停止

图 13

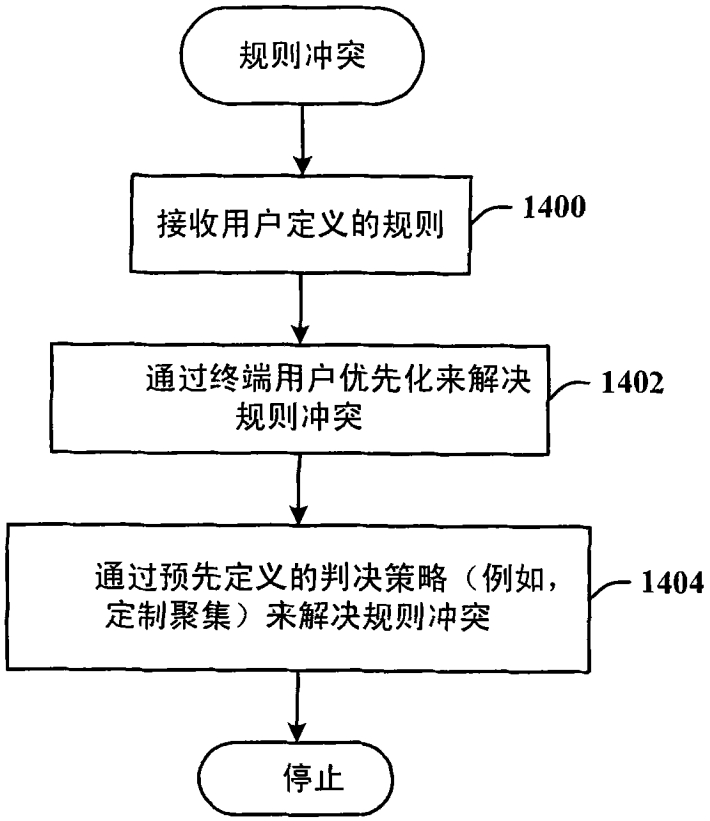

规则冲突
接收用户定义的规则
1400
通过终端用户优先化来解决规则冲突
1402
通过预先定义的判决策略（例如，定制聚集）来解决规则冲突
1404
停止

图 14

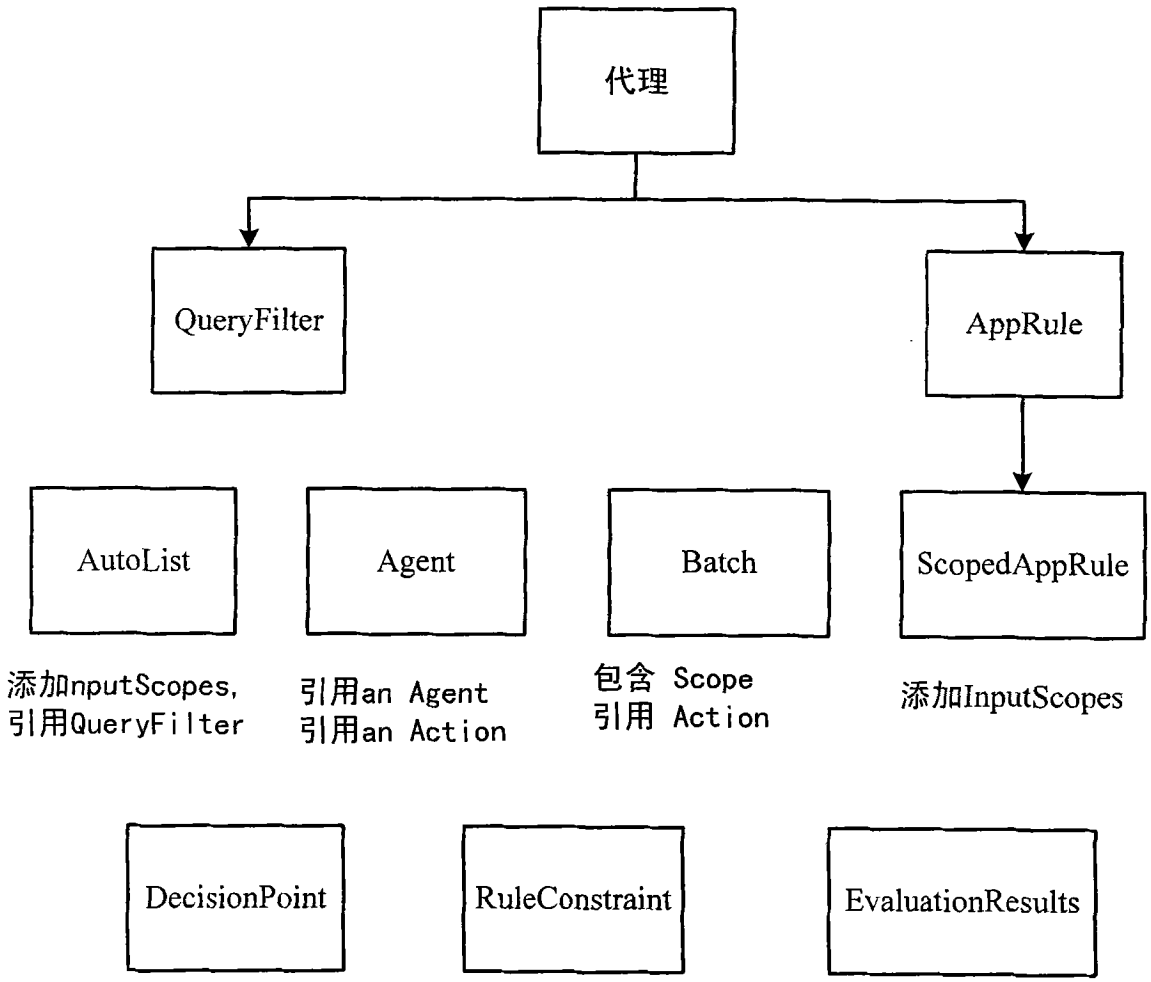

代理
QueryFilter
AppRule
AutoList
Agent
Batch
ScopedAppRule
添加nputScopes,
引用QueryFilter
引用an Agent
引用an Action
包含 Scope
引用 Action
添加InputScopes
DecisionPoint
RuleConstraint
EvaluationResults

图 15

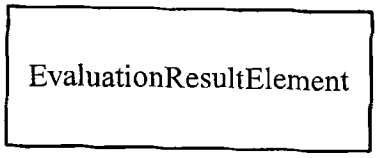


图 16A

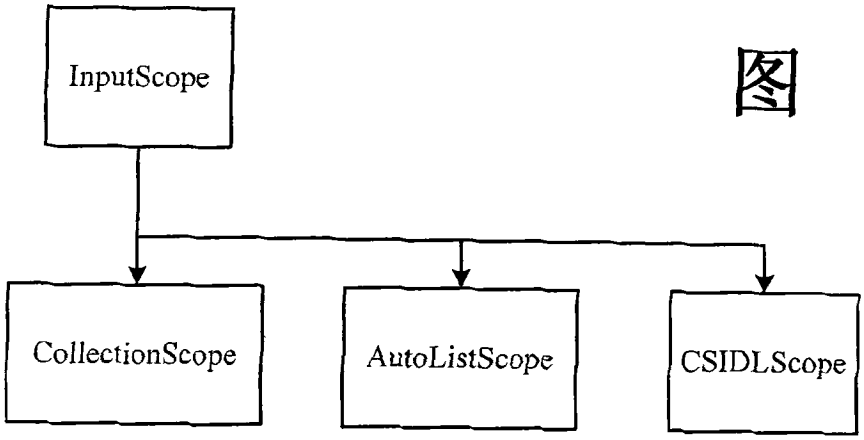


图 16B

图 16C

图 16D

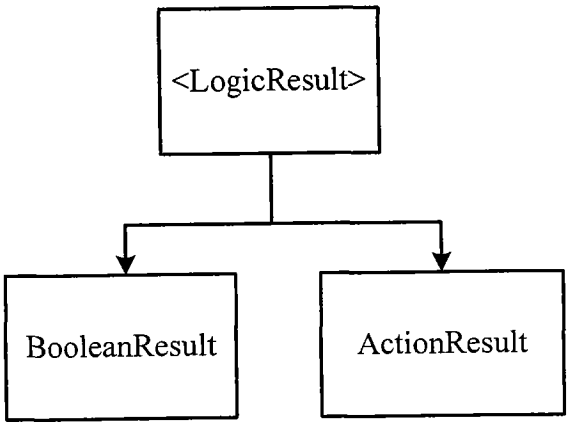

<LogicResult>
BooleanResult
ActionResult

图 16E

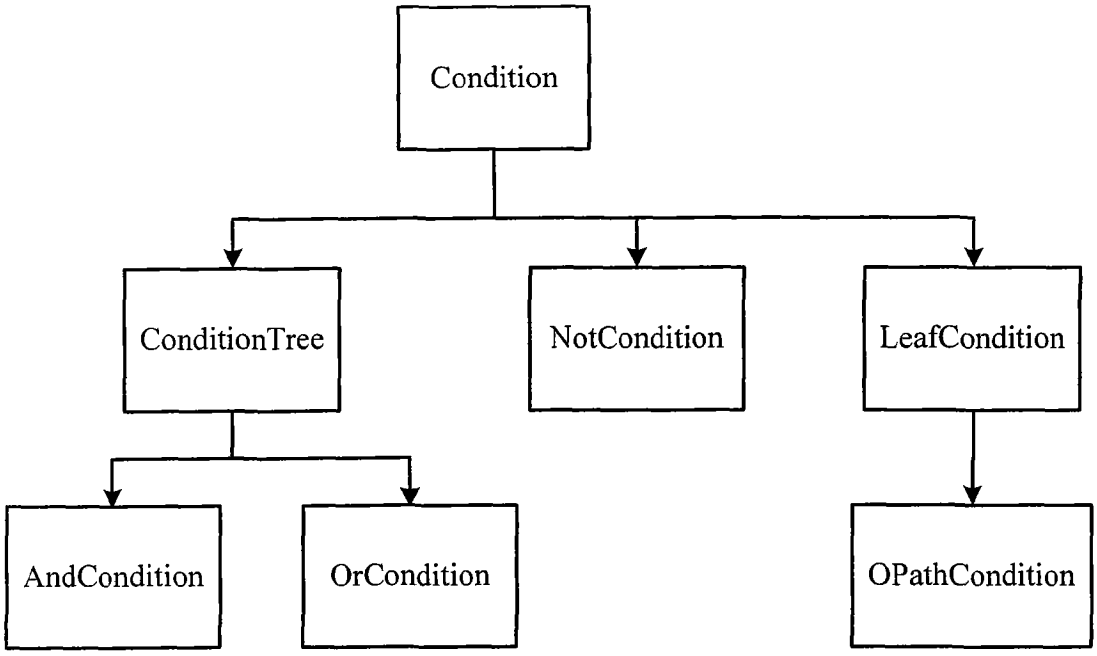

Condition
ConditionTree
NotCondition
LeafCondition
AndCondition
OrCondition
OPathCondition

图 16F

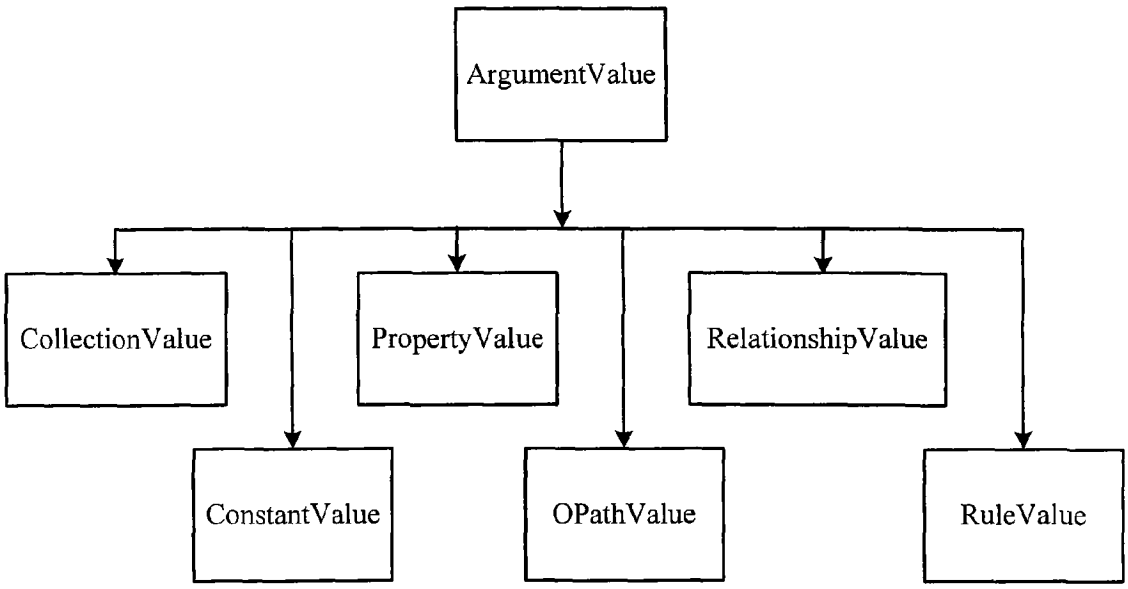

ArgumentValue
CollectionValue
PropertyValue
RelationshipValue
ConstantValue
OPathValue
RuleValue

图 16G

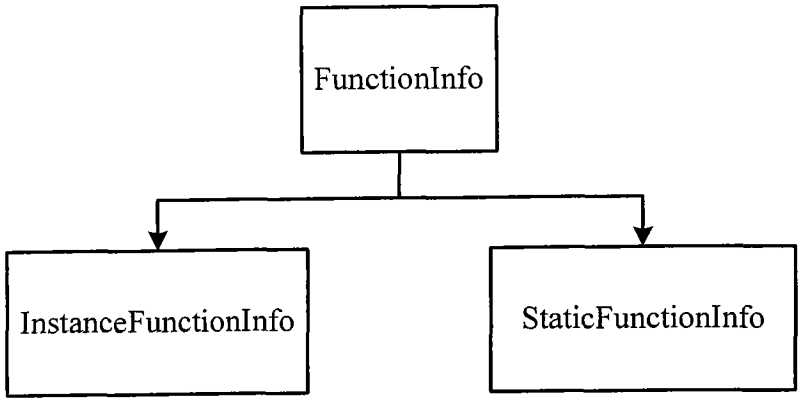

FunctionInfo
InstanceFunctionInfo
StaticFunctionInfo

图 16H

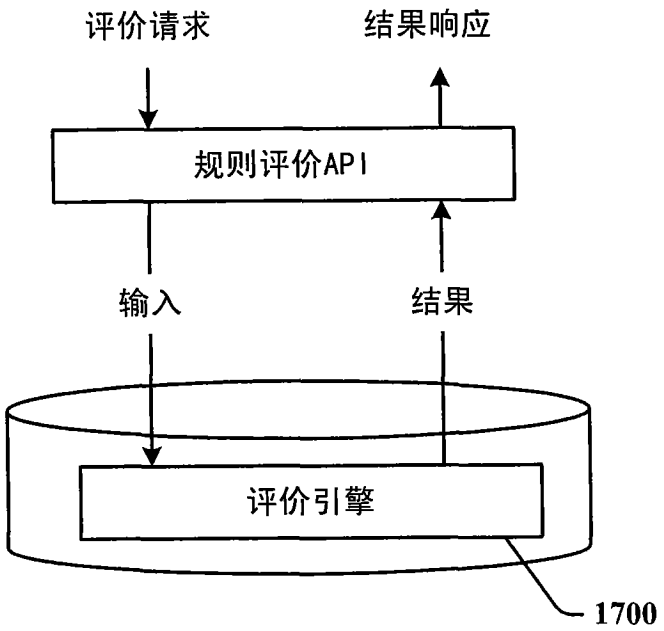


图 17

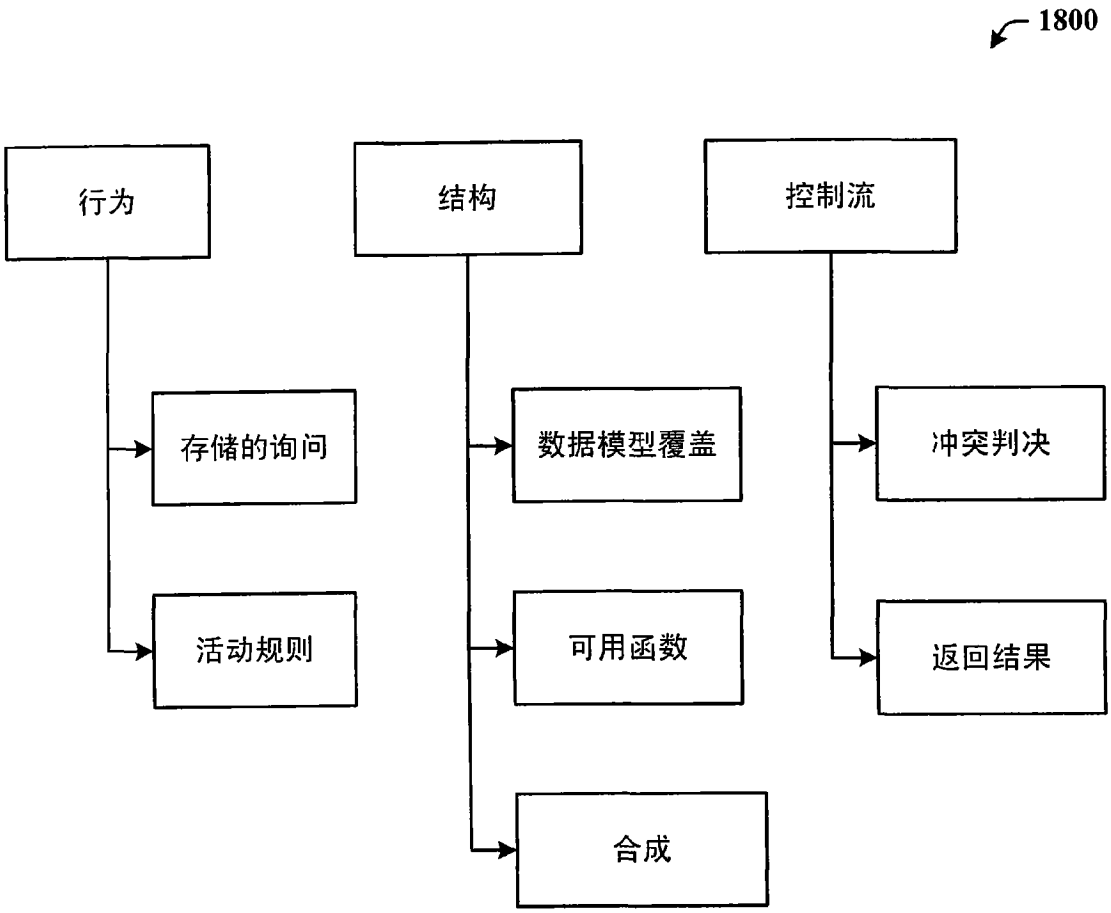


图 18

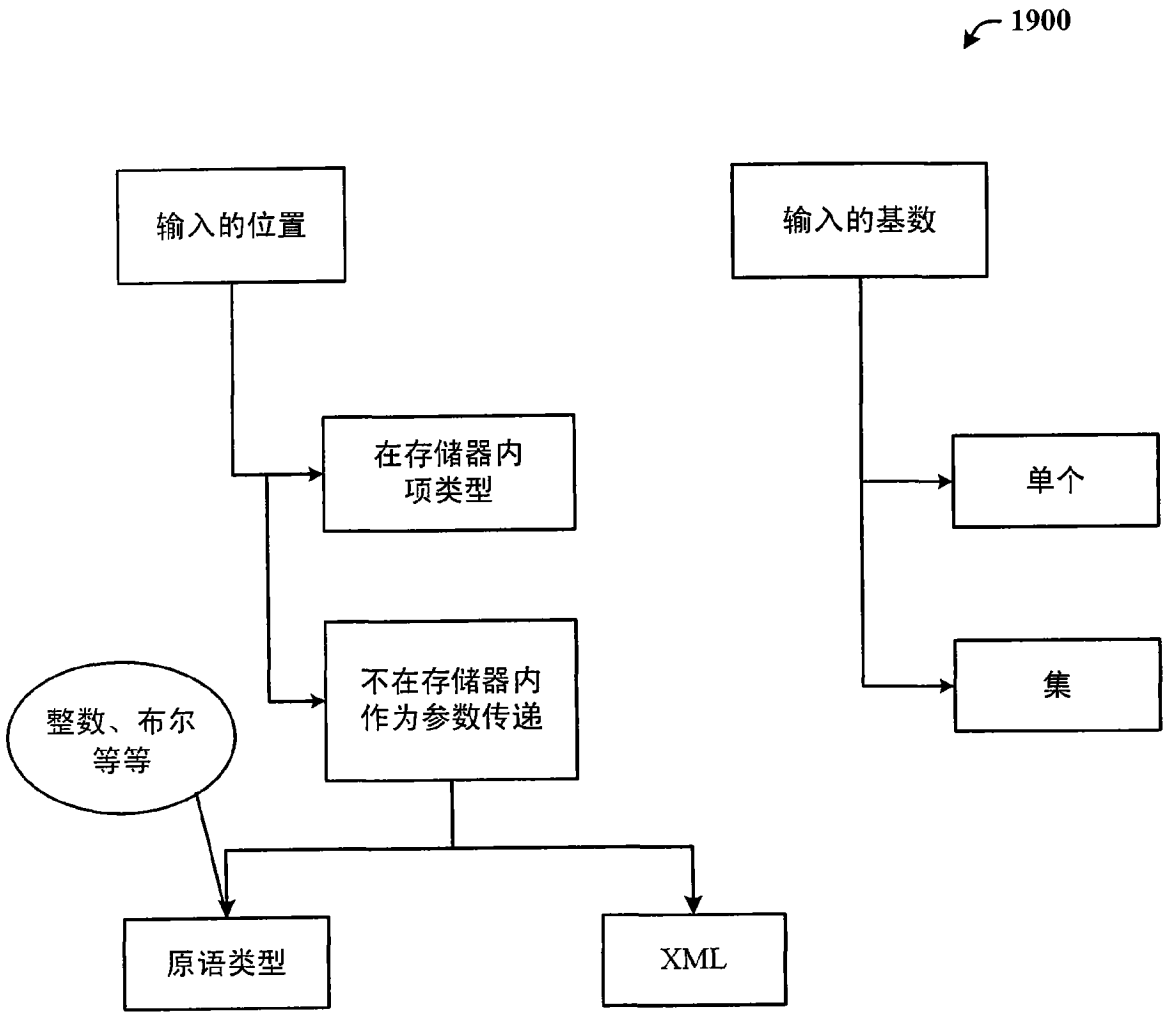

1900
输入的位置
输入的基数
在存储器内
项类型
单个
不在存储器内
作为参数传递
集
整数、布尔
等等
原语类型
XML

图 19

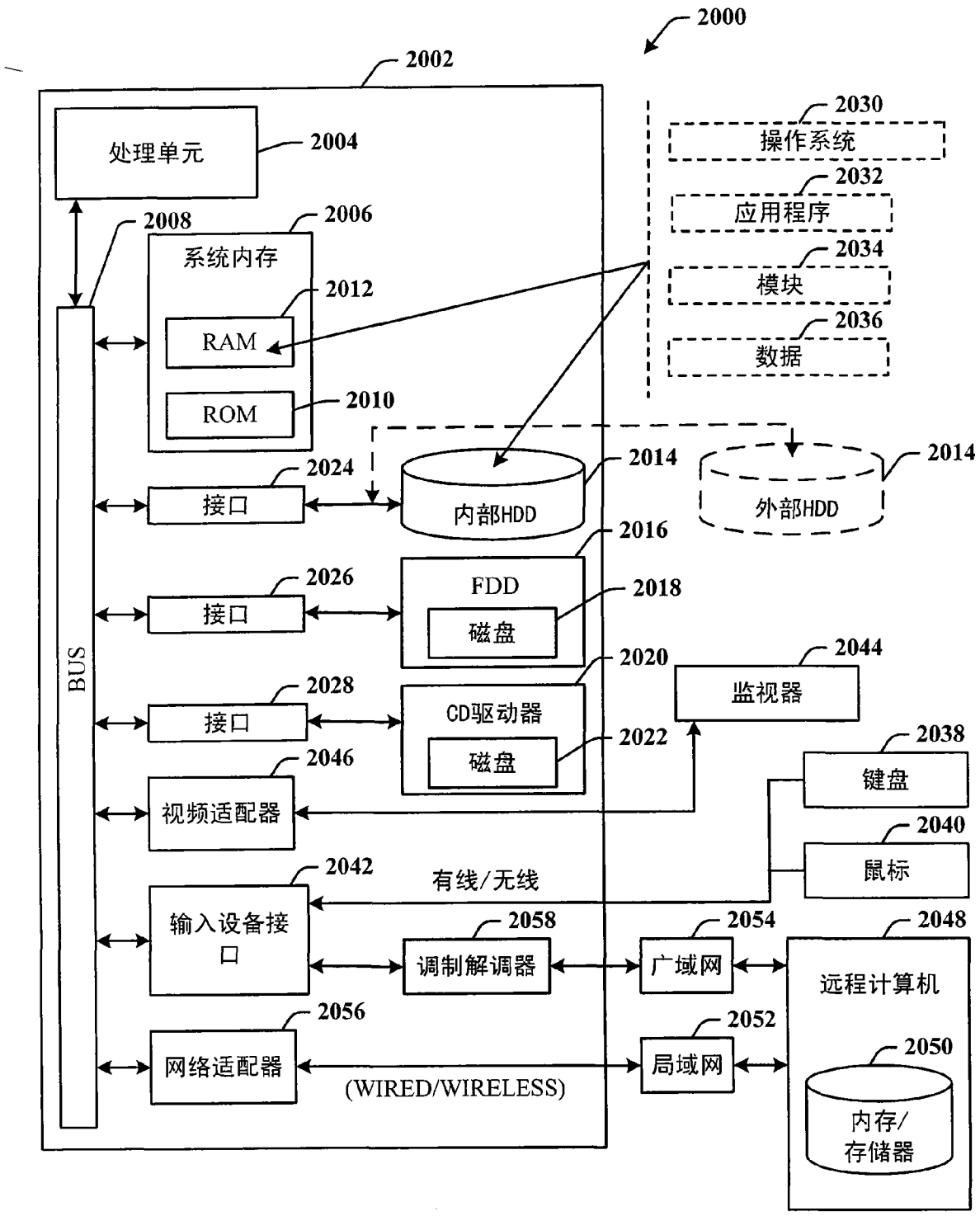

2000
2002
处理单元
2004
2030
操作系统
2032
应用程序
2034
模块
2036
数据
2008
2006
系统内存
2012
RAM
ROM
2010
BUS
2024
接口
内部HDD
2014
外部HDD
2014
2016
FDD
磁盘
2018
2026
接口
2020
CD驱动器
磁盘
2022
2028
接口
2044
监视器
2038
键盘
2046
视频适配器
2040
鼠标
2042
有线/无线
输入设备接口
2058
调制解调器
2054
广域网
2048
远程计算机
2056
网络适配器
(WIRED/WIRELESS)
2052
局域网
2050
内存/存储器

图 20

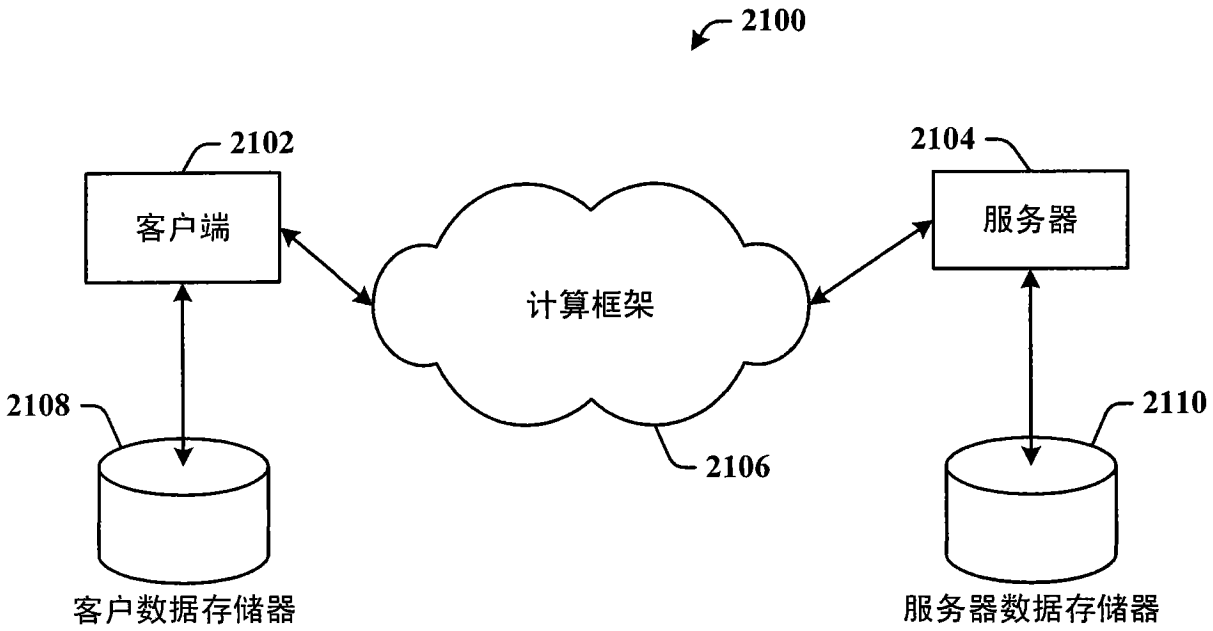


图 21