(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 103543684 B
(45)授权公告日 2017.10.24

(21)申请号 201210452149.4

(22)申请日 2012.11.12

(65)同一申请的已公布的文献号
申请公布号 CN 103543684 A

(43)申请公布日 2014.01.29

(30)优先权数据
61/558,987 2011.11.11 US
13/662,215 2012.10.26 US

(73)专利权人 洛克威尔自动控制技术股份有限公司
地址 美国俄亥俄州

(72)发明人 道格拉斯·W·里德
约瑟夫·布罗尼科夫斯基
苏比安·戈文达拉杰
塔里尔·贾斯珀
迈克尔·D·卡兰恩
史蒂文·约翰·科瓦尔
肯尼斯·S·普拉赫
道格拉斯·J·赖夏德
查尔斯·M·里斯查尔
克里斯多佛·E·施塔内克

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 王萍 陈炜

(51)Int.Cl.
G05B 19/05(2006.01)

审查员 袁珑瑜

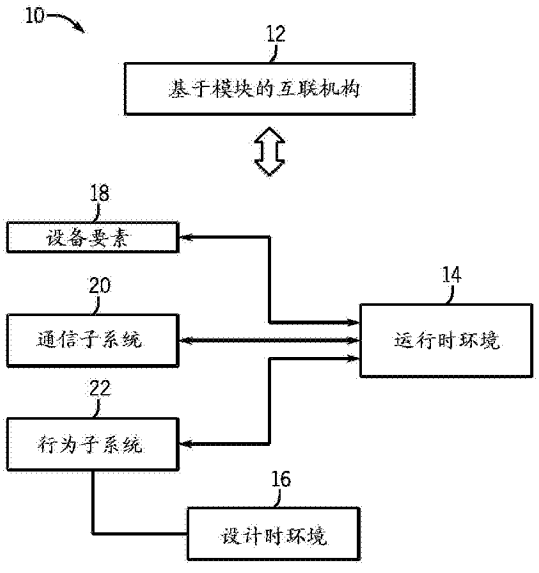
权利要求书4页 说明书21页 附图9页

(54)发明名称

用于传达对象的状态信息的变化的方法和自动控制部件

(57)摘要

提供了一种自动控制系统，该自动控制系统包括存储自动控制系统的对象的状态信息的第一部件。另外，第一部件生成对所存储的状态信息的一个或更多个变化进行描述的一个或更多个增量脚本。此外，第一部件将一个或多个增量脚本发送到控制系统的一个或更多个其他部件，且一个或更多个其他部件基于一个或更多个变化应用该一个或更多个增量脚本以更新存储在一个或更多个其他部件上的状态信息。



CN 103543684 B

1.一种自动控制部件，被配置成：

通过处理器检测自动控制系统的对象的状态信息的一个或更多个变化的指示，所述变化涉及对所述对象的修改、添加、删除或以上的组合；

通过所述处理器创建对自动控制系统的对象的状态信息的所述一个或更多个变化进行描述的一个或更多个增量脚本，而不提供对象的整个状态信息；

通过所述处理器检测提交指示，所述提交指示表示所述对象的状态信息的一个或更多个变化应被提交给所述自动控制系统的一个或更多个其他自动控制部件；以及

当检测到所述提交指示时，通过处理器来将所述一个或更多个增量脚本传输到所述自动控制系统的所述一个或更多个其他自动控制部件，使得能够在检测到所述一个或更多个变化之后且在检测到所述提交指示之前，提交并应用其他自动控制部件的其他变化。

2.根据权利要求1所述的自动控制部件，其中，所述一个或更多个其他自动控制部件被配置成应用所述一个或更多个增量脚本，以基于所述一个或更多个变化对存储在所述一个或更多个其他自动控制部件上的状态信息进行更新。

3.根据权利要求1所述的自动控制部件，其中，所述自动控制部件被配置成：

识别所述一个或更多个变化；以及

至少部分地基于所识别的所述一个或更多个变化来创建所述一个或更多个增量脚本。

4.根据权利要求1所述的自动控制部件，其中，所述自动控制部件包括编程终端、可编程逻辑控制器、输入/输出I/O模块或人机接口HMI终端中的至少一种。

5.根据权利要求1所述的自动控制部件，其中，所述对象包括控制程序、标签、模块配置或人机接口屏幕中的至少一种。

6.根据权利要求1所述的自动控制部件，被配置成创建和/或传输能够与特定的编程技术无关地被解释的增量脚本。

7.根据权利要求1所述的自动控制部件，被配置成基于数据处理和/或数据通信效率创建和/或传输以下增量脚本：该增量脚本选择性地对对象的全部状态信息或仅对所述对象的已经改变了的状态信息进行描述。

8.根据权利要求1所述的自动控制部件，被配置成：当全部状态信息的阈值量变化时，创建描述对象的全部状态信息的增量脚本，否则创建仅描述所述对象的已经改变了的状态信息的增量脚本。

9.根据权利要求1所述的自动控制部件，包括持久化对象模型，其中，所述持久化对象模型被配置成：当所述一个或更多个其他自动控制部件离线时存储所述一个或更多个变化，并且当所述一个或更多个其他自动控制部件返回在线时通过所述增量脚本将所述变化传达给所述一个或更多个其他自动控制部件。

10.根据权利要求1所述的自动控制部件，被配置成创建和/或传输包括有以下中的至少一种的增量脚本：所述对象的标识符、所述对象的已经改变了的要素、与所述要素的变化有关的数据或所述变化的环境的属性。

11.根据权利要求1所述的自动控制部件，被配置成创建和/或传输包括修订版本号的增量脚本，以使得被配置成应用所述一个或更多个增量脚本的一个或更多个部件能够：基于所述对象的状态信息的本地副本的当前修订版本号与所述一个或更多个增量脚本的修订版本号的比较，来对是否应该将所述一个或更多个增量脚本应用到所述本地副本进行认

证。

12.根据权利要求1所述的自动控制部件，其中，所述一个或更多个增量脚本包括能够用于区分对于所存储的状态信息的不同的修订版本的唯一标识符。

13.一种自动控制部件，被配置成：

通过处理器接收包括以下数据的一个或更多个增量脚本：所述数据描述自动控制系统的对象的状态信息的一个或更多个变化，其中，所述变化涉及对所述对象的修改、添加和/或删除，基于通过变化手段提供的提交指示确定所述变化被提交，所述提交指示表示状态信息的一个或更多个变化应被提交给所述自动控制部件；

从所述增量脚本提取描述所述一个或更多个变化的所述数据；以及

至少部分地基于所提取的所述数据来确定所述对象的状态信息。

14.根据权利要求13所述的自动控制部件，被配置成：

从所述增量脚本提取生成所述一个或更多个变化的用户的标识符；以及

基于所述标识符对所述变化的源进行认证。

15.根据权利要求13所述的自动控制部件，被配置成：

将所述状态信息的所述一个或更多个变化应用到描述所述对象的状态信息的所述自动控制部件本地的数据。

16.根据权利要求15所述的自动控制部件，被配置成：

以全或无的方式来应用所述一个或更多个变化，使得应用所述增量脚本中的所有变化或不应用所述增量脚本中的任何变化。

17.根据权利要求13所述的自动控制部件，其中，所述增量脚本包括修订版本号，并且所述自动控制部件被配置成：仅当所述修订版本号能应用于与存储在所述自动控制部件本地的数据中的所述状态信息相关联的当前修订版本号时，才应用所述一个或更多个增量脚本。

18.根据权利要求13所述的自动控制部件，其中，所述增量脚本包括能用于区分对于所存储的所述状态信息的不同的修订版本的唯一标识符，并且所述自动控制部件被配置成基于所述唯一标识符来应用所述一个或更多个增量脚本。

19.一种用于传达自动控制系统的对象的状态信息的变化的方法，所述方法包括：

通过处理器来生成以下一个或更多个增量脚本：该一个或更多个增量脚本代表由变化手段做出的、所述自动控制系统中的一个或更多个对象的状态信息的一个或更多个变化；

通过所述处理器检测提交指示，所述提交指示表示所述对象的状态信息的一个或更多个变化应被提交给订阅了状态信息的一个或更多个变化的通知的受众；以及

当检测到所述提交指示时，通过变化仲裁器来将所述增量脚本公布给订阅了所述状态信息的变化的通知的所述受众，使得能够在做出所述一个或更多个变化之后且在检测到所述提交指示之前对所述受众提交和应用其他变化手段做出的其他变化。

20.根据权利要求19所述的方法，包括将所述自动控制系统中的所述对象的所述状态信息的黄金副本中的变化存储在数据存储器中，其中，所述黄金副本被所述自动控制系统认为是正确的，且用作所述对象的所述状态信息的权威参考副本。

21.根据权利要求19所述的方法，包括：

检测请求订阅所述变化的新受众成员；以及

将关于所述变化将所述新受众成员带至最新所需的所有增量脚本公布给所述新受众成员。

22.根据权利要求21所述的方法，包括：

基于所述受众所请求的订阅，通过所述变化仲裁器来确定代表存储在所述受众上的状态信息的当前修订版本的受众修订版本号；

通过所述变化仲裁器来确定代表存储在所述对象的所述状态信息的权威参考副本上的状态信息的当前修订版本的参考修订版本号；以及

通过所述变化仲裁器，仅提供根据持久化对象模型的指示将所述受众修订版本号更新到所述参考修订版本号所需的所述增量脚本，其中，所述持久化对象模型被配置成存储与所述一个或更多个对象的状态信息的每个变化相关联的增量脚本。

23.根据权利要求19所述的方法，包括：

生成多于一个的增量脚本；

将所述多于一个的增量脚本聚集成复合增量脚本；以及

通过所述变化仲裁器将所述复合增量脚本公布给订阅了所述一个或更多个对象的所述状态信息的变化的通知的受众。

24.一种用于传达自动控制系统的对象的状态信息的变化的方法，包括：

通过处理器或多个处理器确定所述自动控制系统中的所述对象的状态信息的要素的由变化仪器做出的一个或更多个未定变化，其中，还没有将所述变化提交到所述自动控制系统；

利用所述处理器或所述多个处理器之一来生成表示未定变化的一个或更多个增量脚本，其中，所述增量脚本是数据驱动的，不需要耗用特定的编程技术，且被配置成对所述对象的状态信息的所述要素的变化进行描述；

将所述增量脚本和代表会将所述增量脚本中的变化反向的变化的一个或更多个反向增量脚本存储在存储器中；

使用所述处理器或所述多个处理器之一来应用所述增量脚本，以生成所述对象的状态信息的新的修订版本；

通过所述处理器或所述多个处理器之一检测提交指示，所述提交指示表示所述未定变化应被提交给订阅的受众成员；以及

仅当检测到所述提交指示时，才利用所述处理器或所述多个处理器之一来将所述数据驱动的增量脚本提供给所述订阅的受众成员。

25.根据权利要求24所述的方法，包括：

利用所述至少一个处理器来对以下进行检测：所述未定变化的取消，以及通过应用所述反向增量脚本对用于所述未定变化的所述增量脚本的应用的撤销。

26.根据权利要求24所述的方法，包括：

检测在对所述自动控制系统提交所述未定变化之前对所述自动控制系统提交的所述对象的状态信息的第二变化；

使用一个或更多个反向增量脚本收回所述未定变化；

应用所述第二变化；以及

在首先应用所述第二变化的情况下将用于所述未定变化的所述增量脚本重新应用到

所述对象的状态信息的修订版本。

27.根据权利要求24所述的方法，包括：

以所述至少一个处理器确定一个或更多个增量脚本改变了相同元素的值，其中，所述一个或更多个增量脚本将所述元素改变到中间值，并且所述增量脚本之一将所述元素改变到最终值；以及

通过生成将所述相同元素的值改变为所述最终值的单一增量脚本来压缩改变所述相同元素的值的所述增量脚本，而不考虑所述中间值。

用于传达对象的状态信息的变化的方法和自动控制部件

[0001] 本申请是2011年11月11日提交的题为“Control Environment Change Communication”的美国临时专利申请No.61/559,003和于2011年11月11日提交的题为“Automation Control System Change”的美国临时专利申请No.61/558,987的非临时申请，通过引用将其合并到本文中。

技术领域

[0002] 本公开内容的实施方式总体上涉及自动控制与监视系统的领域。更具体地，本公开内容的实施方式涉及自动控制系统的部件之间的状态变化传达。

背景技术

[0003] 对于自动控制与监视系统存在广范围的应用，尤其在工业设置中。这样的应用可以包括广范围的致动器诸如阀、电动机等的供电以及经由传感器的数据收集。典型的自动控制与监视系统可以包括一个或更多个部件，例如：编程终端、自动控制器、输入/输出(I/O)模块和/或人机接口(HMI)终端。

[0004] 人机接口或者“HMI”通常用于监视或者控制各种处理。HMI可以从具体的寄存器读取或者写入具体的寄存器以使得它们可以反映各种机器、传感器、处理等的操作状态。接口还可以写入寄存器以及存储器以使得它们在一定程度上可以控制处理的功能。单独在监视功能中，执行很少或不执行实际的控制。在许多其他设置中，采用类似的设备，例如在汽车、飞机、商业设置以及许多其他应用中。在许多应用中，接口可以不与远程设备或者处理通信，而是可以以独立的方式被操作。

[0005] 在这些接口设备中，用在接口中的对象可以与工业自动化设备的不同的控制、监视或者任何其他参数相关联。这些对象中的一些对象可能在接口设备上具有视觉表示，而其他对象可能不能被视觉上表示然而可以易于通过用户来配置和编程。用户可能期望例如通过创建新的对象、复制对象、编辑对象等来操作这些对象以创建并且定制接口。

[0006] 自动控制与监视系统中的各部件可以利用控制与监视系统的一个或更多个对象(例如，控制程序、标签、模块配置以及HMI屏幕)的状态信息。有时，部件可以被用于修改对象的状态信息。因此，部件可能需要将状态的变化传达给控制与监视系统，以使得其他部件可以获悉控制与监视系统的对象的状态变化。实际上有时候状态的变化可以包括添加或者删除控制与监视系统内的特定对象。例如，传达控制与监视系统对象的状态的传统方法包括了将对象的整个状态提供给控制与监视系统。现在认识到这样的方法往往效率低，提供了比描述控制与监视系统内的对象的变化状态所需更多的信息。提供对象的整个状态可能导致传达状态数据中的带宽低效率以及消耗和使用数据中的处理低效率。此外，现在认识到这样的提供全部状态数据的方法有时可能增加因疏忽而对设置在控制与监视系统中的其他状态变化进行了覆写的可能性。

[0007] 此外，传统方法依靠集中控制与监视。例如，传统的控制与监视系统依靠描述控制系统的集中式数据模型。对集中式数据模型的依赖可能导致处理低效率以及增加的对托管

集中式数据模型的部件(例如,控制器)的依赖。

发明内容

[0008] 以下概述与最初要求保护的发明范围相当的某些实施方式。这些实施方式不意图限制要求保护的发明的范围,相反地这些实施方式仅意图提供本发明的可能的形式的简要摘要。实际上,本发明可以包含可以与以下陈述的实施方式类似或者不同的各种形式。

[0009] 本实施方式提供在自动控制与监视系统中的部件之间传达对象的状态变化的新颖的方法。由于状态变化发生在控制与监视系统内,所以只是改变的数据被传达给控制与监视系统内的其他部件。例如,控制与监视系统对象可以包括控制程序、标签、模块控制以及用于HMI屏幕的图形。当这些对象的要素变化时,变化的要素可以被以数据驱动的方式提供给存储对象的状态信息的部件。通过仅提供变化的要素,而不是提供对象的全套的要素,可以显著地减少传递到部件的数据量。此外,当对象被删除时,可以不需要对象的整个状态。反而,可以仅提供删除的对象的指示,由此在对象被删除的情况下减少要传递的数据量。此外,以数据驱动的方式提供变化可能使得传达是不可知的,或者不取决于具体的编程技术。

[0010] 此外,本发明提供使用遍及控制与监视系统分布的执行引擎来施加传达的变化和/或分布的命令以基于变化来异步执行命令的新颖的方法。例如,控制与监视系统的部件中的一个或更多个部件(例如,智能I/O设备、编程终端、PLC以及HMI等)可以各自包括嵌入式执行引擎。执行引擎可以被存储在部件的有形的、非暂时的、计算机可读取的介质上。当被触发时(例如,通过接收改变的状态信息),在控制与监视系统的各种部件上的嵌入式执行引擎可以基于触发器或者安排的执行时间来异步地响应。例如,分布的命令可以是用户和/或系统定义的以一种或更多种方式对状态变化起反应的命令脚本。通过借助于嵌入在控制与监视系统的部件上的执行引擎使能够执行控制逻辑,可以产生更有效的处理。例如,这样的执行方案可以通过将逻辑分布遍及控制与监视系统来更好地利用多个中央处理单元(CPU)核。

附图说明

[0011] 当参照附图阅读以下详细说明时本实施方式的这些以及其他特征、方面和优势将变得更好理解,在全部附图中同样的标记代表同样的部分,在附图中:

[0012] 图1是根据本发明的某些方面的自动控制与监视系统的部分的框架的总体概观图;

[0013] 图2是根据本发明的实施方式的自动控制与监视系统的图解概观图;

[0014] 图3是根据本发明的实施方式的接口和编程终端中的某些功能部件的概观图;

[0015] 图4是根据本发明的实施方式的设备要素的某些视图或者容器的概观图;

[0016] 图5是图1的控制与监视系统的图,示出了根据实施方式的用于传达状态变化的持久化对象模型的使用;

[0017] 图6示出了根据实施方式的变化的仪器、变化的仲裁器以及受众成员之间的状态变化传达的进程;

[0018] 图7示出了根据实施方式的状态变化被撤销的处理;

[0019] 图8示出了根据实施方式的在未定编辑期间做出外部变化的处理；

[0020] 图9示出了根据实施方式的用于中止未定变化的处理；

[0021] 图10示出了根据实施方式的用于把未定变化压缩成一组变化的处理；

[0022] 图11示出了根据实施方式的使用分布式执行引擎来执行控制命令的自动控制与监视系统；

[0023] 图12示出了根据实施方式的通过执行引擎执行的处理循环；以及

[0024] 图13示出了根据实施方式的用于调度命令的处理。

具体实施方式

[0025] 通常，当状态变化被传达时，控制与监视系统严重地依赖自动控制器，例如可编程逻辑控制器(PLC)以及自动控制器编程(例如，PLC编程)来影响控制与监视系统。自动控制器编程严重地依赖于任务和/或逻辑的基于事件和/或基于进度表的执行(例如，用编程语言写的机器可读的指令，例如继电器梯形逻辑)来影响控制与监视系统的变化。自动控制器经常被用于消耗所有的输入数据、计算并且分布输出数据、处理数据的变化以及将数据分布到控制与监视系统的部件。不幸地，这样的对由控制与监视系统的部件(例如，自动控制器和自动控制器编程)影响和托管的集中式数据模型的严重依赖提供了一些低效率。例如，随着用于集中式模型的预定的和基于事件的任务的数目增加，由于可能导致单个模型的许多附加的变化所以可能出现退化的性能。此外，集中式模型的大量使用(例如，经由自动控制器)产生了处理控制逻辑的更加集中的方法，导致控制逻辑的低效率的执行、单节点故障(例如，当自动控制器发生故障时，整个控制与监视系统可能发生故障)，并且可能造成对自动控制器的处理压力(processing strain)。

[0026] 根据本实施方式，通过利用分布式数据模型、分布式状态变化传达以及分布式命令执行，控制与监视系统可以变得更加灵活。例如，通过在整个控制与监视系统提供增加的协作能力、增加的数据冗余以及处理负载平衡，本实施方式展示了更加鲁棒及灵活的自动控制与监视环境。

[0027] 鲁棒控制与监视系统

[0028] 将通过以下讨论来描述许多方面、部件以及处理。通过引言，总体系统综述目的在于将这些创新置于文中。图1是根据本公开内容的实施方式的用于接口的控制与监视软件框架10的图解表示。框架10有利于通过利用基于模块的互连机构12来构建功能软件，该互连机构12内在地支持动态操作和配置。该动态操作和配置能力利于有效地为可配置的接口提供特征丰富的配置环境。也就是说，如下所述，单个设备要素被设置成可以被单独地编程的独立操作的代码，在库中时预写以便使用，在其功能以及在屏幕中的外观方面被自定义，并且互相连接以提供信息给用户以及控制与监视功能。

[0029] 框架10包括可以属于单个系统(例如，计算机)的两个相关的软件环境。具体地，运行时环境14使操作者(例如，个人用户)能够与应用交互，例如在运行期间(例如，在使用接口期间，通常在与操作中的处理交互期间或者在观察操作中的处理期间)的处理。设计时环境16允许设计者配置接口及其部件。例如，系统可以以图形的方式经由运行时环境14在显示器(例如，计算机或者接口设备屏幕)上将运行信息呈现给操作者。此外，系统可以包括用于接受可以经由运行时环境14来检测和管理的操作者输入的装置(例如，小键盘)。环境如

以下详细描述的那样相互作用，以创新的方式来提供非常增强的接口的使用以及编程。

[0030] 运行时环境14包括或提供对设备要素18的访问。设备要素18为软件部件，可以包括任何在软件环境中可访问或者可配置的要素。例如，设备要素18包括通过运行时环境14来管理的软件部件，例如“ActiveX”控制或者“.NET”部件。“ActiveX”和“.NET”指的是面向对象的概念、技术和工具。通常本领域的技术人员将非常熟悉这样的编程方法。在本文中，这样的标准应该仅被当作是示例，并且“设备要素”应该被理解为包括可以作为拟独立的要素运行的任何大体上类似的部件或者自足的(self-sufficient)程序，有时称之为“对象”。针对这样的要素存在其他标准和平台，通常被不同的公司或者工业群体所拥护。

[0031] 因为这样的设备要素对本文中陈述的概念中的某些概念来说是基本的，所以准备了一些介绍。设备要素通常包括四个特征：性质、方法、连接(或者连接点)以及通信接口。在本文中，性质是可以被调整的属性，例如用于定义要素在屏幕视图中的表示或者图像，以及要素在屏幕上的位置等。在本文中，方法是可执行的函数(有时本文中称作要素“功能性”或者“状态引擎”)，并且定义通过执行要素所进行的操作。在本文中，连接是要素之间的链接，并且可以被用于使数据(从存储器读取或者写入存储器)被发送到另一个要素。

[0032] 设备要素18的具体示例可以包括软件按钮、计时器、计量表、PLC通信服务器、可视化(例如示出在自动控制与监视系统内的部件的状态的屏幕)以及应用。通常，实质上任何可识别的功能都可以被配置为这样的要素。此外，如以下所讨论的那样，这样的要素可以互相通信以进行各种各样的显示、监视操作和控制功能。应注意，设备要素18不需要用于支持设计模式的特殊限制。而且，虽然与图像关联的要素非常有用，尤其是用于可视化，但是许多要素可以不具有视觉表示，而是可以执行HMI内的功能，例如计算，或者甚至管理以及在其他要素之间的数据交换。

[0033] 运行时环境14通常使用通信子系统20来进行操作。通信子系统20适于与设备要素18互相连接。实际上，通信子系统20可以被视为包括设备要素18的连接。然而，通信子系统20可以包括发送数据到外部电路以及从外部电路接收数据的一系列软件、硬件以及固件，例如自动控制器、其他计算机、网络、卫星、传感器、致动器等。

[0034] 运行时环境14通常使用行为子系统22来进行操作，该行为子系统22适于管理设备要素18的行为。例如，行为子系统22的职责可以包括如下：放置和移动设备要素、修改设备要素、在可交换屏幕上集合设备要素、保存并且恢复屏幕布局、管理安全性、保存并且恢复连接列表，以及提供远程访问给运行时环境14。实际上，在此再次地该行为可以被定义为每个设备要素的轮廓(profile)(即，“方法”或者“状态引擎”)的一部分。

[0035] 设计时环境16包括行为子系统22的高等的实现，该行为子系统22的高等的实现利于在不妨碍或者损害运行时环境16的行为的情况下直接或者间接地操作运行时环境14。也就是说，即使在接口工作的情况下也可以设计和重新配置设备要素18。在某些实施例中，行为子系统22可以经由设计时环境16的远程供应来扩展对运行时环境14的访问，例如在常规的浏览器中。行为子系统22允许设计者经由远程编程终端通过将设计时环境16或者设计时环境16的方面从HMI提供到编程终端来与HMI的运行时环境14的方面交互以及改变HMI的运行时环境14的方面。例如，经由网络耦接到膝上型电脑的HMI可以通过经由网络将具体的设计时环境16提供给膝上型电脑来给用户提供配置权能。

[0036] 以下提供如何完成该操作的细节和示例。在当前实施方式中，设计时环境16可以

是结合动态超文本标记语言(DHTML)和动态服务器页面(ASP)服务器脚本来将动态内容提供给浏览器的产物。ASP脚本是专门编写的代码,包括在页面被发送到用户之前在服务器(例如,网络服务器)上处理的一个或更多个脚本(即,小的嵌入式程序)。通常,在常规应用中,这样的脚本提示服务器从数据库访问数据以及在数据库中做出改变。接下来,在将页面发送给请求者之前脚本通常建立或者自定义页面。如以下所讨论的那样,这样的脚本非常不同地用在本框架中,例如在没有设备要素的功能性或其相互关系的先前知识的情况下来建立可视化。

[0037] 通过促进设备要素的变化,设计时环境16允许设计者进行行为子系统22的专门实现或作出可交换的设计时模型。行为子系统22的设计时实现的具体示例包括基于网络的设计时环境16,该基于网络的设计时环境16经由HMI与远程设备之间的TCP/IP连接来扩展对HMI上的运行时环境14的访问。基于网络的设计时环境16有助于在不危害运行时性能或者安全的情况下管理设备要素。在一个专门的实现中,行为子系统22给予设计者能力以使用能够访问相关接口或者HMI的网络浏览器来操作运行时环境14的方面。如上所述,以及如以下详细描述的那样,这通过使用动态内容、脚本以及设备要素性质的配置的组合来实现。

[0038] 图2是根据本公开内容的实施方式的实现上述框架的例如用于工业自动化的控制与监视系统24的图解表示。系统24包括HMI 26,该HMI 26适于与网络部件及配置装备连接。系统24示出为包括适于通过控制/监视设备30(例如,远程计算机、自动控制器,例如可编程逻辑控制器(PLC),或者其他控制器)来与处理28的部件合作的HMI 26。HMI 26可以物理地类似现有的硬件,例如面板、监视器或者单机设备。

[0039] 可以通过使用任何适当的网络策略来促进HMI 26与处理28的部件之间的合作。实际上,可以采用产业标准网络,例如DeviceNet,以使能够进行数据转移。这样的网络允许根据预定义的协议来进行数据交换,并且可以提供用于操作网络要素的能力。如上所述,虽然在当前讨论中参考了网络系统以及包括控制器和其他装备的系统,所描述的HMI 26和编程技术可以同样适用于非网络部件(例如,GPS显示器、游戏显示器、手机显示器、平板显示器等)以及适用于在工业自动化领域之外的网络系统。例如,设施管理中可以使用以下描述的布置和处理:汽车和车辆的接口、计算机数控(CNC)机器、销售点(POS)系统、用于商业市场(例如,电梯、入门系统)的控制接口等,以上仅列举一些。

[0040] 由相应的行为子系统管理和构造的运行或者操作环境14被存储并且驻留在HMI 26上。例如,这样的行为子系统可以适于例如在HMI 26的初始制造或者设置期间从存储位置加载应用程序配置框架(例如,10)。当被加载时,存储的应用程序框架可以适于创建屏幕以及在屏幕中定位用户接口设备要素(与要素对应的实际的图像或者图片表示)。这些应用程序、屏幕以及用户接口要素每个都是设备要素的类型。如以下所描述的那样,HMI 26包括指定布局和设备要素的交互的所存储的应用程序。基于运行时引擎的基于网络的设计时环境16也被加载且驻留在HMI 26上。设计时环境16可以适于为设计时环境和运行时环境两者来处理高等的特征(例如,安全管理)。

[0041] HMI 26可以适于允许用户与几乎任何处理交互。例如,处理可以包括:压缩机站、炼油厂、用于制造食物项目(food items)的成批操作、机械化的装配线等。因此,处理28可以包括各种操作部件,例如电动机、阀、致动器、传感器,或者无数的制造、加工、原料处理及其他应用。此外,处理28可以包括用于通过自动化和/或观察来调节处理变量的控制与监视

装备。示出的处理28包括传感器34和致动器36。传感器34可以包括适于提供关于处理条件的信息的任何数目的设备。致动器36可以类似地包括适于响应于输入信号来进行机械动作的任何数目的设备。

[0042] 如所示出的那样，这些传感器34和致动器36与控制/监视设备30(例如，自动控制器)通信并且可以被分配以可由HMI 26访问的控制/监视设备30中的特定地址。传感器34和致动器36可以与HMI 26直接进行通信。这些设备可以被用于操作处理装备。实际上，他们可以用在处理循环中，该处理循环由控制/监视设备30和/或HMI 26所控制与监视。可以基于过程输入(例如，从传感器34的输入)或者直接输入(例如，通过HMI 26接收的操作者输入)来激活这样的处理循环。

[0043] 接口上的服务器软件允许查看开发环境，并且直接重新配置接口(尤其是设备要素及其关联的外观和功能)，而不需要专门查看或者配置软件。该益处来自于下述事实，该事实为设备要素和设计时环境自身驻留在HMI 26中，并且通过HMI 26“提供”给编程终端46上的浏览器或者其他通用的查看器。换句话说，可以降低或者消除对外部计算机工作站(例如，膝上型电脑和台式电脑)的必要的支持。应指出，对用于查看和修改接口的配置的“浏览器”的提及不限于网络浏览器或者任何具体的浏览器。对浏览器的提及意图是示例性的。一般说来，术语“浏览器”被用在本文中来指代包括任何通用的查看器的软件。

[0044] 通过如以下所描述的那样对设备要素编程，HMI 26可以被认为包括用于呈现一个或更多个屏幕视图或者可视化的指令，和在通过参考屏幕视图(例如，按下按钮、触摸屏幕的位置等)来与HMI 26进行交互时执行的设备要素。可以通过任何期望的软件或者软件包来定义屏幕视图和设备要素。例如，屏幕视图和设备要素可以被操作系统38调用或执行。如上所述，根据本实施方式的设备要素可以是符合“.NET”或者“ActiveX”标准的对象。操作系统本身可以基于任何适当的平台，例如Window CE、OS-X等。如本文中所引用的，设备要素和工具支持用于在网络(例如，因特网)上发送数据的网络服务或者技术。因此如以下所描述的那样，这些设备要素遵循关于信息共享的一套规则并且被改适以与各种脚本和编程语言一起使用。这样的设备要素使能够将交互内容供应给外部应用程序，例如LAN、WAN、内联网、外联网乃至万维网。因此，操作系统38和各种设备要素利于借助于浏览器48通过允许对到浏览器48的配置访问(access)(例如，对其提供)来动态地配置HMI 26。

[0045] 例如，这样的配置入口包括用于例示设备要素的入口。换句话说，实际上可以从浏览器48来创建并且执行新设备要素。此外，应指出，浏览器48不需要实际的功能入口。实际上，在一个实施方式中，经由浏览器48的请求导致基于数据功能性和容器中设备要素的内容的操作的“绘制(draw)”序列，由此在实际上没有提供功能方面的情况下允许设备要素表示的示出和对其配置的访问。这允许在不需要对远程工作站的技术支持的情况下经由远程工作站进行配置。

[0046] 除了如上所述(并且如以下更详细地描述的那样)的操作系统38和设备要素之外，HMI 26包括应用或者应用层40。可以自身包括设备要素的应用利于访问并且从HMI 26的各种设备要素获取信息。具体地，应用40表示可以是可针对执行被枚举的多级设备要素中的第一级。在实际实现中应用40可以包括以XML页面形式的用户应用。然后用户应用与用户或者操作者、以及与设计者交互，如以下更详细地描述的那样。

[0047] 屏幕视图和设备要素可以被描述为独立的可执行的软件。在本实现中，通过用标

记语言(例如,超文本标记语言或者HTML)编写的适当的代码来定义屏幕视图。因此,可以在不使用转换程序的情况下进行HMI 26的图形接口屏幕的配置。此外,通过对设备要素进行编程,可以经由下述驻留的服务器软件(称为服务器42)来直接地在HMI 26上开发屏幕视图,该驻留的服务器软件使驻留的开发环境可用于远程访问。具体地,在一个实施方式中,某些设备要素的表示(例如,ActiveX控制)被提供给浏览器48而没有提供软件部件自身。因为可以经由浏览器48来访问开发或者设计时环境,因此可以消除将变化下载到屏幕以及更新远程配置软件应用的需要。

[0048] 如上所述,设备要素可以包括功能性,通过该功能性他们从通常在其他设备中(但是也可以在HMI内)的具体存储器或者存储寄存器读取或者写入。例如,具体的功能可以对应于写入控制/监视设备30的寄存器32或者从控制/监视设备30的寄存器32读取。在简单的情况下,例如,对象访问一段数据(例如,如由传感器确定的部件的状态),并且生成输出信号用于编写对应于不同的网络设备的状态的值。如以下将更加详细地讨论的那样,这样的状态信息可以经由状态增量(delta,Δ)43来传达。例如,在图2中描述的实施方式中,控制/监视设备30和HMI 26可以使用状态增量43来传达状态信息。此外,编程终端46也可以使用状态增量43来与控制/监视设备30和HMI 26进行状态信息的传达。

[0049] 当然可以配置更加复杂的功能。例如,在工业控制与监视环境中,这样的设备要素可以对一系列物理部件的操作进行仿真,例如瞬时接触按钮、具有延迟输出的按钮、开关等。许多预编程的设备要素可以供HMI 26使用。这样的功能模块可以是可经由网络访问的,或者可以驻留在HMI 26上,或者驻留在直接链接到HMI 26的单独的设备上。用这种方法,HMI供应者或者软件供应者可以提供许多可能的构建块,根据上述构建块可以对屏幕以及复杂的控制与监视功能进行编程。实际上,可利用的设备要素的库44可以驻留在HMI 26上以利于配置HMI 26,如以下描述的那样。屏幕指令可以基于操作者输入来调用用于执行期望的功能的设备要素,并且这些指令可以被编程为预编程要素的版本。例如,操作者可以通过触摸触摸屏上的位置或者压下键盘上的键来提供初始输入。然后基于屏幕指令和与指令关联的设备要素(例如,具有具体的位置触发调用或者执行预配置的设备要素)可以执行期望的功能。因此,操作者能够与处理进行交互,通常用以改变屏幕视图、写入寄存器或者命令生成其他输出或者控制信号。在单机实现中,交互可以是简单的再调用或者存储数据、改变屏幕等。

[0050] 在一些HMI具有许多这样的屏幕和大量的设备要素的情况下,可以采用一个或更多个单独的接口屏幕。每个设备要素进而可以被唯一地编程以考虑具体的输入,执行具体的功能以及生成用于具体输出的信号。如以下所描述的那样,多个这样的设备要素可以被加载并且被托管在单个软件“容器”(例如,ActiveX容器)中。

[0051] 可以通过直接地与HMI 26本身(如果存在一个)上的屏幕或者面板进行交互来配置HMI 26,但是在许多情况下将从远程编程终端46来执行配置。例如,经由浏览器48或者类似的应用来直接地提供对驻留库44和/或操作系统30及应用40的访问。在本实现中,在编程终端46处不需要其他专门的软件。实际上,驻留在HMI 26上的服务器42可以提供对库44中的设备要素的访问。通过直接地在HMI 26上存储库44中的设备要素,消除或者降低了版本冲突等的风险。另外,HMI 26可以直接地连接到编程终端46,或者通过参考分配给HMI 26的IP地址(互联网协议地址)被访问。

[0052] 访问控制方案可用于限制改变屏幕和设备要素的能力。例如，可能需要密码或者用户访问状态来获得该访问。此外，在目前构思的实施方式中，编程终端自动地识别HMI 26或者下述终端，在该终端上HMI 26作为耦接到编程终端46(例如，类似于外部存储器或者驱动)的设备驻留。因此，一旦连接到编程终端，HMI 26可以简单地被“识别”为可以被访问的设备(提供下述的配置屏幕和工具)。

[0053] 一旦驻留在HMI 26上的设备要素是编程终端46可访问的，那么可以经由来自编程终端46的通信链路直接地在HMI 26上修改或者更新HMI 26的方面。例如，用户可能希望更新具体的HMI图形以提供数据，例如历史数据或者与从重新安装的传感器34接收的信息相关的趋势。因此，用户可能发现在以离线模式(例如，不立即执行变化)的情况下更新用于表示这样的数据的HMI图形是期望的或者便利的。在这种情况下，用户可以经由编程终端46链接到可利用的设备要素的库44并且使用其来在开发环境中修改HMI图形或者功能。

[0054] 应指出，附加的设备要素可以被增加到库44。例如，如果趋势设备要素没有驻留在HMI 26上，用户可以从驻留在编程终端46上的配置库50将这样的元素下载到HMI 26。或者，用户可以从可经由网络(例如，因特网)访问的资源库52直接地到HMI 26或者通过编程终端46来访问趋势设备要素。这可以特别有益，这是因为新的且改进的设备要素可以单独地并且在定期的基础上被下载到HMI 26，由此在不需要定期释放新的转换程序或HMI操作系统，或者运行时或设计时环境软件的情况下增加新的功能。开发环境可以提供到这样的库的链接。此外，在使用嵌入代码的实施方式中(例如，操作系统、服务器软件、设备对象等)，由于嵌入代码驻留在HMI 26上，所以可以避免与嵌入代码的版本冲突并且可以消除编程终端软件升级的必要性。

[0055] 为了追踪控制与监视系统24的一个或更多个部件的状态信息，控制与监视系统24的部件可以使用代表控制与监视系统24的各个方面的分布式数据模型。例如，分布式数据模型可以使代表控制与监视系统24的数据模型的多重高速缓存的副本能够存在于控制与监视系统24内(例如，在控制与监视系统24的部件中的一个或更多个部件处)。如以下将更详细地描述的那样，分布式数据模型可以结合增量(delta)脚本和分布式命令处理来工作。增量脚本可以使控制与监视系统24的一个或更多个部件能够确定数据模型的状态变化，生成只包含数据模型的变化和/或整个数据模型的增量脚本并且将增量脚本提供给控制与监视系统24的其他部件。其他部件可以消耗增量脚本并且将包含在增量脚本内的数据施加到数据模型的本地高速缓存的副本(例如，包含在控制与监视系统24的部件之一处的分布式副本)。此外，如以下将更详细地讨论的那样，控制与监视系统24的某些部件可以利用使能够进行分布式命令处理的分布式执行引擎。这样的分布式命令处理使控制与监视系统24的分布式部件能够基于提供给分布式部件的事件或者进度表来处理命令执行。

[0056] 通过使用分布式的数据模型、分布式增量传达(例如，经由增量脚本)以及分布式命令执行，作为结果的控制与监视系统24可以更加鲁棒且灵活。例如，不是依赖于集中式控制/监视设备30处的集中式数据模型，可以使用数据模型的分布式副本以影响在控制与监视系统24内的变化。例如，不是依赖于在控制/监视设备30处的集中式数据模型来影响HMI 26的变化，HMI 26可以包括分布式数据模式的副本，其依赖于该副本来影响HMI 26之内的变化。此外，HMI 26可以接收下述状态增量43(例如，经由增量脚本)，该状态增量43由HMI 26所消耗并且通过HMI 26施加到HMI的数据模型的本地副本。此外，如将在以下更详细地描

述的那样，HMI 26可以包括下述本地执行引擎(例如，分布在HMI 26的执行引擎)，该本地执行引擎用于在HMI 26处执行提供至HMI 26的命令。

[0057] 此外，这种功能使同步的数据存储能够跨控制与监视系统24而存在。这些同步的数据存储可以通过使多个用户能够对将要与其他的数据存储中的每一个数据存储同步的单个数据存储做出改变而使能够协作。此外，由于数据存储可以高速缓存控制与监视系统24的数据的单个副本，所以可以进行离线修改。例如，即使当控制器不可用时，通过使用在数据存储之一中高速缓存的数据，用户可以对控制与监视系统24进行修改。当用户恢复在线(例如，可以访问处理器)时，由用户在离线情况下所做的修改可以与其他的数据存储同步。因此，用户能够以更一致且可靠的方式将变化提供给控制与监视系统24。

[0058] 例如，一个用户可以在设计软件中对标签定义、元数据定义进行改变，可以对设计的要素重命名，可以修改报警设置、改变数据类型和/或修改数据记录情况，例如Rockwell自动控制股份有限公司的RSLogix 5000™。可以对本地数据存储进行这些由用户提交的改变。在线时，这些变化可以被传递至在控制与监视系统24内的其他数据存储，从而跨系统24来应用该变化。离线时，这些变化可以保留在本地数据存储中并且当返回在线时(例如，重新连接到控制与监视系统24的控制器)可以被同步。通过变化的自动传递，可以避免冗余的变化进入而节约研发工作。此外，基于通过系统24进行的自动重命名传递，可以减少调试和初始化。此外，由于这些变化可以起源于整个系统，所以当不同用户开发控制器和HMI时可以允许灵活的工作流。

[0059] 如上所述，通过分布数据模型、经由增量脚本将变化传递至分布式数据模型以及分布命令执行，相对于传统控制与监视系统可以极大地改进控制与监视系统24。例如，可以由控制与监视系统24内分布的数据模型的多份副本中的任何一个副本来服务控制与监视系统24的客户端(例如，请求控制与监视系统24的数据模型中的数据的部件)。控制与监视系统24可以从基于许多判定因素中的一个因素确定哪一个副本来服务客户端。例如，可以选择具体的分布式数据模型副本以基于性能效率例如高效网络通路(例如，哪个副本最靠近客户端、在本地或在网络上、或哪个网络通路具有最大带宽等)将数据提供至客户端。此外，在该判定中也可以纳入处理考虑这一因素。例如，这种鲁棒的控制与监视系统24可以使数据能够被提供至使用负载均衡技术的客户端。在一个实施方式中，可以从包含已知或可能比控制与监视系统24的其他部件提供较少请求的数据模型的分布式副本的部件向客户端提供数据。在一个示例中，控制与监视系统24可以包括两个控制/监视设备30(例如，2个自动控制器)。控制与监视系统24可以预测或观察到第一控制/监视设备30比第二控制/监视设备30接收更多的关于数据的请求。因此，控制与监视系统24可以确定从第二控制/监视设备30服务客户端以避免过度利用第一控制/监视设备30。因而，控制与监视系统24可以通过基于控制与监视系统24内的部件的负载来均衡请求以避免控制/监视设备30的溢流。在某些实施方式中，这可以包括从单个部件将请求提供至请求的阈值数或数据的量并且当达到阈值时移动至溢流源处。在一些实施方式中，这可以包括在提供数据时基本上均匀分摊请求负载或数据量。

[0060] 除可以提供分布式数据模型、增量脚本以及执行引擎的负载均衡性能之外，这些性能还可以有益于控制与监视系统24中的数据冗余。例如，控制与监视系统24内的一个或多个部件可以监视数据模型的一个或多个分布式副本。在检测到副本不稳定时(例如，没有

准确地表示分布式模型的副本)，不稳定的副本可以由稳定的副本代替(例如，准确地表示分布式模型的副本)。可以从分布在被确定为具有准确地表示数据模型的副本的控制与监视系统24中的数据模型的其他副本中的任何一个副本而获得稳定的副本。

[0061] 在一些实施方式中，控制与监视系统24的部件可以访问下述冗余池，所述冗余池给分布式数据模型的有效副本或给存储分布式数据模型的有效副本的控制与监视系统24的部件提供指向器。例如，当客户端部件请求数据模型中的数据时，其可以访问传达哪里可以获得数据的冗余池。如上所述，控制与监视系统24的一个或多个部件可以监视数据模型的副本以确定不稳定的副本。当检测到一个或多个不稳定的副本时，控制与监视系统24的部件可以将指向器移动至不稳定的副本或存储不稳定的副本的控制与监视系统24的部件。因此，不可经由冗余池访问该不稳定副本。

[0062] 在某些实施方式中，如上所述，在从冗余池移除不稳定的副本(或存储不稳定的副本的部件)之后，控制与监视系统24的部件可以用稳定的版本代替不稳定的副本。在不稳定的副本被代替之后，控制与监视系统24的部件可以将替换稳定版本(或存储替换稳定版本的部件)重新增加回冗余池以用于将来使用。

[0063] 为了更好的示出设计时环境与运行时环境之间的关系，图3提供了表示HMI 26与编程终端46之间的交互作用的高级流程图。以下提供了关于该处理的更多细节。通常，用于HMI 26与编程终端46的平台包括操作系统或执行的软件38、应用软件40、以及任何通信软件、微处理器、网络接口、输入/输出硬件、一般软件库、数据库管理、用户界面软件等(图3中未具体表示)。在示出的实施方式中，设计时平台和运行时平台在HMI 26之内交互作用。设计时平台将用作设计时环境16的视图提供给台式个人计算机平台(例如运行合适的操作系统38诸如Windows XP、Windows Vista或Linux)，而运行时平台经由操作系统(例如Windows CE、Linux)与设计时平台合作。设计时平台提供动态服务器内容54，而运行时平台显示关于HMI 26自身的视图(如果在HMI 26上提供显示屏)。设计时环境16在浏览器48(例如网络浏览器或其他的通用浏览器)中显示。

[0064] 图3在非常高层面示出设计时环境16如何与操作系统38、应用40和运行时环境14交互作用。箭头56表示HMI 26与编程终端46之间内容的动态交换。通常，与设计时环境16的交互作用是最初配置HMI屏或可视化、设备要素及它们的功能和相互作用，或者重新配置该软件的设计者58的任务。通常由操作者60直接在HMI 26上与运行时环境14进行交互作用。应该注意的是，当设计时环境16具有特殊需要时，在本实施方式中，这很大程度上依赖于操作系统38、应用软件40和运行时环境14。设计时环境16和运行时环境14可以利用某些基准技术(例如DHTML、HTML、HTTP、动态服务器内容，JavaScript、Web浏览器)以分别在设计时平台和运行时平台中操作。虽然，在所示出的实施方式中运行时环境14和设计时环境16驻留在分离的平台上，但是在一些实施方式中，运行时环境14和设计时环境16可以驻留在同一平台上。例如，设计时平台和运行时平台可以被配置为或被认为是单个平台。

[0065] 在本发明的一种实施方式中，利用了设计时网络实施。如由图3中的动态服务器内容54所标注以及如下所述，该设计时网络实施通过使用具有来自HMI的DHTML支持的网络浏览器(如，48)，来提供运行在设计时平台上的软件的速度和灵活性。DHTML用于对设计时环境16中的网络内容进行动态操作。此外，在HMI中使用动态服务器内容54以将动态网络内容提供给设计时环境16。该动态的客户端-服务器环境使得网络浏览器能够在不需要针对相

关处理器编译的软件的情况下模拟运行在设计时平台上的应用。

[0066] 图4是示出了根据本技术实施方式的设计时环境中的一个或更多个设备要素的图。该图包括通过显示器100(如,用于浏览器显示的屏幕)、属性编辑器102和HMI 26之间的关系示出的交互。

[0067] 用配置屏幕或显示器100表示的设计时环境包括静态内容104和动态内容。动态内容包括与任何所显示或表示的设备要素106对应的图像(如,虚拟的开/关按钮、测量仪器)。在本技术的一种实施方式中,图像由HTML中的图像标签指定并且是由如以下所描述的HMI创建的JPEG文件的一部分。静态内容104可以由动态服务器页面(ASP)服务器来创建或它可以预先存在于HTML文件中。应当注意,在一些实施方式中,只有指定的设计者才能够对静态内容104进行编辑。

[0068] 由配置屏幕或显示器100表示的设计时环境包括静态内容104和动态内容。动态内容包括与任何所显示或表示的设备要素106对应的图像(如,虚拟的开/关按钮、测量仪器)。在本技术的一种实施方式中,图像是由HTML中的图像标签来指定并且是由如以下所描述的HMI创建的JPEG文件的一部分。静态内容104可以由ASP服务器来创建或它可以预先存在于HTML文件中。应当注意,在一些实施方式中,指定的设计者仅能够对静态内容104进行编辑。

[0069] 在图4的表示中,设备要素表示106包括在视图容器108中。如本领域技术人员所理解的,容器一般定义特定设备要素被打开且准备好使用的处理空间的一部分。因此,容器108可以与仅包括可在当前屏幕内查看的要素的第一视图容器对应。如上面所讨论的,在HMI中可以设置许多这样的屏幕。其他屏幕如替代的控制或接口屏幕可以设置在其他视图容器如容器110中。一般地,为了加速HMI的操作(如,屏幕视图之间的改变),通过限定与其相关联或在其中设置有设备要素的表示的各个设备要素来预限定这些视图容器以及将其彼此关联。可以将全局容器112限定为包括各种视图容器所必需的所有设备要素以及可在任何视图容器中不表示出的其他要素。如在图4中示出的,因此,视图容器108包括执行“轻推”功能、并且由第一屏幕中的表示来显示的虚拟按钮106。新容器110包括若干部件如“开始”按钮114、“停止”按钮116、虚拟仪表118和数字读出装置120等。全局容器112则包括用于各种视图容器的所有这些设备要素,以及操作可见设备要素所需要的、然而其自身不是可见的任何设备要素122。这些设备要素可以包括执行计算、使趋向、通信和许多其他功能的要素。

[0070] 图4还示出了在其中用户可以访问要素106的各种属性的属性编辑器102。如上所讨论的,要素106还可以包括与要素106相关联的连接和文本,其还可以由用户通过与属性编辑器102类似的编辑器来配置。

[0071] 在一种实施方式中,属性编辑器102可以通过从浏览器(如,图2的浏览器48)到驻留在HMI 26中的服务器96(如,HTTP服务器)的查询字符串来与HMI 26进行交互。服务器96与包括有如动态链接库(DLL)等基于模块的互连机构12的ASP服务器98进行协作以接收并响应查询。DLL允许将可执行例程存储为单独的文件,当程序需要或引用时,这些单独的文件能够被加载。在上述示例中,当接收呼叫时,由ASP服务器98重载该页面并且对查询字符串进行初始解析,这导致对移动命令的评估。服务器侧脚本则对由图像106表示的设备要素18进行访问以更新其位置属性。然后在网页上对新的属性信息进行更新并且将该页面传送到浏览器48。

[0072] 传达状态变化

[0073] 现已经讨论了结合通过增量脚本和分布式命令执行的分布式状态变化通知使用分布式数据模型的益处，下面将更加详细地对分布式状态变化通知进行讨论。如上所讨论的，图2是根据本技术实施方式的适于使用增量脚本来提供部件状态信息的示例性控制与监视系统24的概略表示。如所示出的，控制与监视系统24可以包括一个或更多个人机接口(HMI)26和适于与处理28的部件进行连接的一个或更多个控制/监视设备30。控制/监视设备30可以包括有助于在控制与监视系统24上执行任务(如，处理控制、远程装备监视、数据获取等)的一个或更多个处理器以及数据存储设备。此外，编程终端46可以使得一个或更多个用户能够对HMI 26和/或控制/监视设备30的属性进行配置。

[0074] 在控制环境中，控制与监视系统24的各种对象(如，控制程序、标签、模块配置和HMI屏幕)的状态可以存储在控制与监视系统24的各个部件(如，编程终端46、控制/监视设备30、I/O模块和/或HMI终端26)的存储器(如，硬盘驱动器、只读存储器、和/或随机存取存储器)中。控制与监视系统24的部件中的每一个可以以松耦合、异步的方式独立地来操作。此外，部件可以以不同的编程技术(如，C++、Java和/或C#)来实施。由于对控制环境对象的状态信息做出了改变，可能需要将该状态信息与驻留在其他部件上的状态信息进行同步，以使得部件可以连续地获知控制与监视系统24内的对象的状态。根据本实施方式，为了保持被告知以状态信息，存储有状态信息的自动部件可以接收称为状态增量43的数据(如，已经改变的状态要素)，而不接收还没有改变且因而已经出现在存储有状态信息的各个部件上的所存储的状态信息中的状态要素。例如，状态增量43可以包括因控制与监视系统24内的动作而已经改变的任何数据。通过提供状态增量43而不提供未改变的状态信息，可以观测到提高了的效率。例如，在具有100个状态要素的传统的控制与监视系统24中，可以将100个状态要素中的每一个状态要素提供给存储那个对象的状态信息的每个部件。通过仅提供状态增量43，控制与监视系统24的部件可以只传输针对已经改变的要素的数据。因此，如果100个状态要素中仅有一个状态要素改变，则不会传输其他99个要素，从而减小了关于传统系统的网络流量。此外，仅提供状态增量43可以减小因疏忽而覆写了在控制与监视系统24内其他地方生成的状态变化信息的可能性。例如，在上面提到的100个状态要素的情况下，如果将所有100个状态要素传输到其他部件，则99个未改变的要素可能导致对其他地方这些99个要素之一做出的变化的覆写。通过仅提供改变的要素(如，状态增量43)，上述99个未改变的要素将不被已改变且被传达到其他部件的一个要素影响。

[0075] 现已经讨论了状态增量24的使用，图5示出了包括有用于对控制与监视系统24的部件之间的状态变化进行传达的持久化对象模型的控制与监视系统24。例如，上述部件可以包括控制/监视设备30(如，PLC)、提供有项目文件150的编程终端和如托管持久化对象模型152和协作会话154的控制/监视设备30和客户端156等部件。如先前所讨论的，控制/监视设备30可以适于与处理28(图1)的部件进行连接。项目文件150可以是对控制与监视系统24的限定并且存储在编程终端46(图1)的存储器(如，硬盘驱动器)中的各种属性进行表示的计算机文件输出。持久化对象模型152可以是控制与监视系统24中的一个或更多个部件的状态数据的计算机模型，其以持久的方式(如，通过将状态数据存储在如硬盘驱动器的非易失性存储介质中)对控制与监视系统24中的状态数据的变化保持跟踪。持久化对象模型152可以用作变化传达权威，以使得所有对对象的状态做出的已提交变化都是通过持久化对象

模型152来存储和传达的。如在以下所更加详细地讨论的，协作会话154可以是控制与监视系统24的部件之间的交互性信息交换接口，其为进行未定的变化提供了环境(如，在用户选择提交一些变化后，可以仅将这些变化应用于和传输到控制与监视系统24的其他部件中)。客户端156可以是控制与监视系统24的将对象的状态信息保存在存储器中的任何其他部件，诸如提供了对象的表示视图的部件。

[0076] 在所示出的实施方式中，所示出的部件(提供了协作会话数据154的控制/监视设备30、提供了更新的项目文件150的编程终端46、提供了持久化对象模型152和协作会话154的控制/监视设备30和客户端156)中的每一个部件都包括有数据容器158(如为数据预留的存储器)。数据容器158包括对控制与监视系统24的一个或更多个对象的状态进行定义的状态要素160。状态要素160可以以数据驱动方式来定义以使得不同的技术(如，C++、Java、和/或C#)可以使用由状态要素160表示的数据。如先前所讨论的，可能期望对存储在控制与监视系统24的各个部件中的状态信息进行有效地同步。因为存储在数据容器158中的状态要素160中的一个或更多个发生变化，所以可能需要对存储在其他部件中的数据要素160进行同步。

[0077] 如上所讨论的，持久化对象模型152可以是控制与监视系统24中的各个部件中应用状态变化时的所指定的机构。持久化对象模型152可以在其数据容器158中包括：其被称为对于一个或更多个对象的状态信息的黄金副本162(如由交叉影线所示出的)。黄金副本162包括控制与监视系统24总是认为正确的状态信息的副本。换言之，黄金副本162是状态信息的权威副本。每个状态信息具有它自己的黄金副本162，其可以或可以不与其他的状态信息的黄金副本162一起驻留在控制与监视系统24(如，在相同的计算机系统上)内。如果提交了一个或更多个状态要素变化，则以增量脚本170的形式将变化的要素提供给黄金副本162，增量脚本170基于状态要素变化来更新。然后通过增量脚本170将状态要素变化从黄金副本提供给控制与监视系统24内的其他部件。

[0078] 为了实现数据容器158内的状态变化，控制与监视系统24的部件可以扮演不同的角色，这些角色可以包括变化手段164、变化仲裁器166和受众168。变化手段164(如，通过当前实施方式中的编辑器提供修改的项目文件150的客户端)给变化仲裁器166发送变化请求。变化手段164可以通过接收关于变化请求的异步的变化响应和/或错误响应来验证变化的成功。变化仲裁器166(如，托管持久化对象模型42的服务器)对到来的变化进行排队，通过执行所请求的变化来处理变化，基于请求来做出其他副作用变化或放弃该变化。变化仲裁器166可以提供对变化手段164的变化响应，当变化出现时将变化通知公布给受众168(如，在协作会话154中涉及的客户端156和/或控制/监视设备30)，和/或将变化写入黄金副本162。受众168接收变化通知并且使用上述通知来对状态信息的存储在其数据容器158中的本地副本进行更新。

[0079] 如先前所讨论的，在控制与监视系统24的各个部件中使用的编程技术可能不是统一的。例如，一些部件可以使用C++，而其他部件可以使用C#或Java。因此，图1的设置在变化手段164、变化仲裁器166和受众168之间的状态增量43可以设置在不依赖于具体技术的数据驱动增量脚本170中。增量脚本170可以以创建、更新和/或删除(CRUD)数据的形式来描述对象状态变化。创建数据可以包括有助于对象的创建的一些或所有数据(如对于矩形，矩形的空间位置、宽度和高度)。对于任何没有提供创建请求的数据可以使用默认值。更新数据

可以包括已经在对象中更新了的数据(如,对于具有更新了的空间位置的矩形图形,更新数据可以仅包括新的空间位置)。删除数据可以标识(如,描述其标识符)已经移除的对象状态数据(如,对于已经移除了的矩形,删除数据可以包括待删除的矩形的名称)。在一个示例中,如果使用以下C#伪码产生变化:

[0080]

```
变化管理器 cm = 新建 变化管理器 ( ) ;
产生变化 c1 = 变化.复合 ( )
        创建("矩形").处于(模式).设置("X", "10").设置("Y", "10").
设置("宽度", "100").设置("高度", "200").
        创建("圆形").处于(模式).设置("X", "10").设置("Y", "10").
设置("半径", "100");
cm.执行(c1);
(ChangeManager cm = new GetChangeManager();
CreateChange c1 = Changes.Composite().
        Create("Rectangle").Under(model).Set("X",        "10").Set("Y",
"10").Set("Width", "100").Set("Height", "200").
        Create("Circle").Under(model).Set("X",        "10").Set("Y",
"10").Set("Radius", "100");
cm.Do(c1);)
```

[0081] 在一些实施方式中,数据驱动增量脚本可能与以下伪XML示例相似:

```
<脚本>
 <创建 类型="com.rockwell.矩形", 创建者ID="45:2331" 母ID
="92">
    <设置属性: "X", 值="10" />
    <设置属性: "Y", 值="10" />
    <设置属性: "宽度", 值="100" />
    <设置属性: "高度", 值="200" />
 </创建>
```

[0082]

```
 <创建 类型="com.rockwell.圆形", 创建者ID ="45:4281" 母ID
="67">
    <设置属性: "X", 值="10" />
    <设置属性: "Y", 值="10" />
    <设置属性: "半径", 值="100" />
 </创建>
</脚本>
(<Script>
 <Create type="com.rockwell.Rectangle", CreatorID="45:2331" parentID="92">
```

```
        <Setter property="X", value="10" />
        <Setter property="Y", value="10" />
        <Setter property="Width", value="100" />
        <Setter property="Height", value="200" />
    </Create>
```

[0083] `<Create type="com.rockwell.Circle", CreatorID=”45:4281” parentID=”67”>`

```
        <Setter property="X", value="10" />
        <Setter property="Y", value="10" />
        <Setter property="Radius", value="100" />
    </Create>
</Script>)
```

[0084] 在可替代的实施方式中，可以不必包括创建者ID或母ID。但是，在当前实施方式中提供了这些ID以对可能与变化一起包括的附加数据(如，做出变化的实体的标识和/或在其下创建当前对象的母对象)进行说明。因为数据驱动的增量脚本170是不可知的或不依赖于具体的编程技术，所以增量脚本170可以由控制与监视系统24的任意其他部件来消费，而不考虑使用的编程技术。

[0085] 如在上面的示例中示出的，在一些实施方式中，增量脚本170可以包括不止一个变化。因此，增量脚本170提供了以全或无(all or nothing)方式来处理整个变化集合的方式。例如，如上所示出的，在增量脚本内包括有两组创建数据用于显示器上的可视化，一组用于创建矩形图像，一组用于创建圆形图像。如果圆形图像的创建产生错误，则矩形变化可能被撤销，这导致了全或无方式。

[0086] 增量脚本170还可以包括标题信息如变化修订版本号、当提交了变化时的时间戳、做出变化的用户的标识符和/或唯一修订版本标识符等。用户的标识符可能有助于对变化源进行认证。此外，增量脚本170包括变化所应用的对象的标识符、已经改变的状态要素160和状态要素160的变化值。创建数据集合可以包括对象的全部状态(如，所有的状态要素160)，这是因为状态要素160中的每一个要素将是第一次被引入给增量脚本170的消费者。

[0087] 现转向图6，示出了变化手段164、变化仲裁器166和受众成员168之间的状态变化传达的进程190。在当前实施方式中，受众168(如，客户端156)向协作会话154提供订阅请求192。订阅请求192可以包括对于存储在受众成员168中的状态信息的修订版本194的修订版本号。如果协作会话154上的修订版本与在订阅请求192中所发送的修订版本号不匹配，则协作会话将发出将受众成员168带至存储在协作会话154中的修订版本所需的对增量脚本170集合的立即的更新通知。例如，在面板A中，客户端156发送包括有修订版本5的订阅请求192。协作会话154在修订版本8上，因此发送针对修订版本6、7和8的增量脚本170给客户端156。客户端156可以将增量脚本170应用到它的状态，因此如在面板B中所示出的，客户端被更新到修订版本8。

[0088] 如果变化手段164(如，提供更新的程序文件150的客户端或服务器)更新黄金副本162，则应该将变化通知给协作会话154和订阅受众成员(如，客户端156)。如在面板B中示出的，当黄金副本162从修订版本8更新到修订版本9时(如，经由通过发送来自变化手段164的

更新的项目文件150来精心安排的变化)，变化仲裁器166向协作会话154提供对于修订版本9的增量脚本170。如在面板C中示出的，协作会话154应用对于修订版本9的增量脚本170，因此被更新至修订版本9。增量脚本170然后被传递至受众成员168(如，客户端156)。客户端156应用增量脚本170并且更新至修订版本9。

[0089] 在某些场景中，受众成员可能需要比存储在协作会话154中更多的增量脚本170。例如，如果客户端156在处于修订版本2上时将发送订阅请求192并且协作会话154仅具有对于修订版本5至8的增量脚本170，则客户端156将仍然需要对于修订版本3和4的增量脚本170。如果协作会话154正缺少必要的增量脚本170，则它可以请求黄金副本162提供所需的增量脚本170。在一些实施方式中，黄金副本162会存储针对对象的状态信息的每个修订版本的所有增量脚本。但是，在其他实施方式中，仅会存储有限数量的脚本(如，增量脚本170的最后的5、10、50或100个修订版本)。如果黄金副本162可以提供必需的脚本，则可以通过协作会话154将其传递至客户端156。但是，如果不能传递所必需的增量脚本，则可以通知受众成员168(如，通过例外消息)和/或使用与当前状态信息相关联的整个要素集合来重载受众成员168，使得受众成员168为最新。此外，如果受众成员168遭遇应用一个或更多个增量脚本170的错误，则使用与当前状态信息相关联的整个要素集合来重载受众成员168。另外地，在某些情况下，如果将需要大量的增量脚本170以更新受众成员168，则完全地重载所有的状态信息可能更加有效或合乎需要，而不是应用状态增量。在某些实施方式中，如果需要应用的增量脚本的数量超过了最大增量脚本阈值，则使用与当前状态信息相关联的整个要素集合来重载受众成员168。最大增量脚本阈值可以基于所感知到的会倾向于使得状态信息的完全重载相比加载增加的增量脚本更加有效的增量脚本数量来自定义。

[0090] 在某些实施方式中，控制与监视系统24还可以包括反向增量。反向增量对从当前修订版本变回到前一修订版本所必需的变化进行描述。如果被应用，则反向增量脚本会将对象的状态信息回退一个修订版本。将这种反向增量脚本应用到包括与反向增量脚本相同的修订版本数量的数据容器(如，图5的数据容器158)。反向增量脚本可以有助于创建针对在控制与监视系统24中提交的变化的“撤销”功能，并且还可以用于收回还没有提交的未定变化，如在提交变化之前在协作会话154中创建的那些。

[0091] 图7示出了根据实施方式的一个撤销场景。在面板A中，由第一客户端发起对象210的针对修订版本211的编辑会话。第一客户端在该会话内进行编辑，以将对象210带至面板B中的未定修订版本214。第一客户端断开，并且当被断开时，第二客户端撤销修订版本214和213，如在面板C中示出的。然后，第二客户端做出新的变化213和214。

[0092] 为了防止第一客户端仅基于修订版本号来检测它是最新的，可以赋给每个修订版本一个标识符以使得修订版本号和标识符的组合创建针对该修订版本号的唯一标识符。如果应用反向增量脚本以撤销变化，则可以保留所撤销的增量脚本，以使得可以实施“重做”功能。如果重做变化，则重新使用针对该修订版本的先前标识符，这是因为该增量脚本正在重新引入先前已经移除的相同变化。但是，如果做出新的修订版本，则使用新的修订版本标识符以使得没有控制与监视系统24的部件使用具有相同编号的新修订版本混淆所撤销的修订版本。

[0093] 例如，图7中的修订版本中的每一个都具有相关联的标识符。修订版本211具有标识符M，修订版本212具有标识符R，原始的修订版本213具有标识符T以及原始的修订版本

214具有标识符X。如果撤销修订版本214和213，则将它们从未定修订版本中移除。如果它们被“撤销”，则将它们重新添加到未定变化，重新生成具有相同标识符T和X的修订版本。但是，在当前示例中，做出新的变化，分别创建具有标识符S和Y的新修订版本213和214。因为它们是完全新的修订版本，所以使用新标识符S和Y来标识上述修订版本。一旦第一客户端返回在线状态并且针对更新进行重订阅，则毫无疑问它当前不是最新的，这是因为其最终的修订版本是214-X并且当前修订版本是214-Y。在一些实施方式中，可通过跟踪修订版本号和标识符从而找到编辑路径和更新修订版本信息来更新第一客户端。在其他实施方式中，如果发现不一致的修订版本号标识符，则使用状态信息的整个集合(如，所有的状态要素160)来重载该部件。

[0094] 可以对正在进行未定编辑的协作会话(如，图6中的协作会话154)外的黄金副本(如，图6中的黄金副本162)做出变化。图8示出了以下场景：在协作会话154中当前正在进行未定编辑的同时做出对于黄金副本162的外部变化。如所示出的，对对象210的修订版本221-B应用第一未定变化Δ1生成修订版本222-J。另外地，分别将第二未定变化Δ2和第三未定变化Δ3用于生成修订版本223-N和修订版本224-D。在提交未定变化Δ1、Δ2和Δ3之前，由控制与监视系统24的另一个部件将外部变化Δ1’应用到当前处于修订版本221-B的黄金副本162。如果协作会话154接收到新的修订版本222存在的通知，则协作会话154收回未定变化Δ3、Δ2和Δ1(将其作为未来待处理的前向增量来保存)。然后，协作会话应用针对修订版本222-H的增量脚本，并且然后再应用分别创建修订版本223-R、224-C和225-X的未定变化Δ1、Δ2和Δ3。在某些情况下，可以修改未定变化Δ1、Δ2和Δ3以在修订版本222-H后应用它们。在一些实施方式中，可以通知在协作会话154中做出未定变化的受众成员未定变化正在越过外部变化被应用到黄金副本162。

[0095] 在某些情况下，用户可能期望中止在协作会话154中做出的未定变化。图9示出了用于中止协作会话154中的未定修订的过程。如在当前示例中所示出的，用户从修订版本221-B创建未定变化Δ1，生成修订版本222-J。从修订版本222-J创建未定变化Δ2，生成修订版本223-N。进一步地，从状态223-N创建未定变化Δ3，生成修订版本224-D。用户可以确定变化不是必需的和/或不是期望的并且可以撤销变化(如，通过选择图2中的编程终端46中的撤销按钮)。为了收回未定变化，具有未定状态变化的部件可以应用针对每一个未定变化(如，Δ3、Δ2和Δ1)的反向增量，使得保留原始的非未定修订版本(如，修订版本51-B)。可替代地，因为黄金副本162具有所存储的最新的非未定修订版本(如，不包括待中止的变化的修订版本)，所以部件可以简单地从黄金副本162重载全部的状态信息。因此，如图9所示，通过反向增量或从黄金副本162重载，在时间T1处对协作会话留有修订版本221-B。因此，协作会话可用来承担从修订版本221-B进行附加编辑(如，Δ4)，在时间T2处生成新的修订版本222-R。

[0096] 在某些情况下，将多个未定变化压缩成一个修订版本是有益的，而不是针对未定变化中的每一个来创建单独的修订版本。图10示出了以下实施方式：将未定变化中的一些变化组合成一组编辑，使得生成较少的修订版本。如所示出的，在时间T0处打开编辑会话。将未定变化应用到修订版本221-B，生成修订版本222-J、223-N和224-D。未定变化可以与对共用状态要素做出的变化有关(如，每个变化可以修改矩形在显示器上的空间位置)。例如，修订版本222-J可以将矩形布置在屏幕的中央，修订版本223-N可以将矩形位置更新到屏幕

的左上角，并且修订版本224-D可以将该位置更新到屏幕的左下角。因此，虽然施加了若干值变化，但是可能仅需要从原始值(如，修订版本221-B)到最终值(例如，屏幕上左下角放置，如在修订版本224-D中所描述的)的增量。因此，协作会话154中的中间修订版本可能衰减到黄金副本162上的单个修订版本。因此，如在T2处所示出的，未定变化Δ1、Δ2和Δ3被压缩并应用到修订版本221-B，产生了修订版本222-R。在部件被配置成当检测到与修订版号相关联的冲突标识符时完全地重载所有状态信息的实施方式中，当通知部件修订版本222-R可用(如，在T3处示出的)时，部件可以重载针对修订版本222-R的所有状态信息。如本领域技术人员所应当理解的，这仅仅是可以用来组合增量的一种形式的压缩。所提供的示例无意限定针对未定变化的压缩技术。

[0097] 分布式命令执行

[0098] 现转到以下讨论中：当变化一旦被传达，如何在控制与监视系统24内来应用变化，图11示出了具有各个部件(如，HMI终端26、控制/监视设备30、编程终端46、智能输入/输出设备260和哑输入/输出(I/O)设备262)的控制与监视系统24。智能I/O设备260可以包括中央处理单元(CPU)，以使得智能I/O设备260可以基于提供给它们的数据来执行逻辑。哑I/O设备262可以不包括CPU，因此可以依赖于控制器来将逻辑应用到它们的输入。

[0099] 执行引擎264可以嵌入在控制与监视系统24的能够支持执行引擎的各个部件内。在一个示例中，具有CPU的部件嵌入有执行引擎264。执行引擎264能够将控制与监视系统24中的变化(如，状态增量43)应用到具有嵌入的执行引擎264的各个部件中。执行引擎264包括命令(如，命令脚本266)和触发条件268。当触发条件268被评估为真时由执行引擎264来执行命令脚本266。例如，触发条件268在以下情况下可以评估为真：当智能I/O设备260或哑I/O设备262的状态有变化，控制/监视设备30中的数据值有变化(如，由增量脚本170产生)和/或当用户与HMI 26进行交互。通过将执行引擎264遍及控制与监视系统24的各个部件来分布，可以更加有效地处理控制与监视系统24变化。例如，可以利用各个部件的CPU的处理能力来执行控制与监视系统24的部件所需的控制逻辑。此外，相比集中的控制器，控制与监视系统24的各种部件上的命令执行可以增加冗余度和/或提供更好的地点来执行命令。例如，响应于控制与监视系统24的变化，使智能I/O设备260能够执行智能I/O设备260所特有的逻辑，而不需要依赖于控制/监视设备30。

[0100] 如上所讨论的，一些部件(如，哑I/O设备262)可能不能支持嵌入的执行引擎264，或可能支持执行引擎264然而不具有嵌入式的。这些部件可以依赖于其他部件(如，控制/监视设备30)来执行针对不具有嵌入的执行引擎84的部件的逻辑。例如，如在图11中示出的，哑I/O设备262不具有嵌入的执行引擎264。相反，使用控制/监视设备30的传统逻辑(如，梯形逻辑(LL)、功能框图(FBD)、顺序功能图(SFC)等)来轮询数据。

[0101] 上述命令(例如，如用户和/或系统所定义的继电器梯形逻辑等命令脚本266)可以是存储在有形、非易失、计算机可读介质(如硬驱动器、数据库、只读存储器和/或随机存取存储器)中的计算机可读指令(如，对象)，以在当触发条件出现时或在调度时间处被执行。例如，命令可以存储在图5中的数据容器158中。命令可以从命令基本类继承属性和/或功能的基本集。可以将特定的属性和行为添加到基本类中以得出其他命令类如用于屏幕导航和写标签值等的类。在某些实施方式中，命令基本类可以包括参数，或可以用于输入和输出的参数数据名称/值对的集合。此外，命令基本类可以包括指示命令已经完成执行的“完成”属

性。命令基本类可以包括指示因错误而导致命令执行停止的错误属性。此外，命令基本类可以包括由控制与监视系统24用于确定谁来负责命令的存储清除(如，什么实体应当在执行之后将命令从数据容器158删除)的母属性。命令基本类可以包括对命令进行标识的名称属性。可以将名称属性用在表达与触发条件268中，以使得命令的属性可以触发附加命令。命令基本类可以包括指示命令的执行进程的进程属性并且还可以具有指示命令的执行已经超时(如，在所分配的时间段内没有执行完)的超时属性。命令基本类可以包括将命令添加到恰当的执行线程的调度属性，将在下面对其进行更加详细的讨论。此外，命令基本类可以包括包含有执行指令的执行属性。

[0102] 在某些实施方式中，可以将命令复合或放在一起。具有两种基本的复合形式：顺序命令复合和并行命令复合。在顺序命令复合中，被一起放入组中的每个命令以给定的顺序一次一个地执行。有用的顺序命令复合的一个示例可以是进行以下处理的一组命令：1)写入标签以开始罐填充(tank filling)，2)等待具体的标签值，以及3)改变图形元素的状态。以下是可能的顺序命令复合的伪代码示例：

[0103]
```
<顺序>
 <写标签 编号=”mytank” … >
 <等待 触发=”mytank.fill ==100” … >
 <设置状态 编号=”myTankDoneText” 状态=”完成” />
</顺序>
```

[0104]
```
( <Sequence>
 <WriteTag ref=”mytank” … >
 <WaitFor trigger=”mytank.fill==100” … >
 <SetState ref=”myTankDoneText” state=”Done” />
</Sequence>)
```

[0105] 在并行命令复合中，复合体中放在一起的每个命令在同一时间执行。例如，下面的写标签命令可以在同一开始时间执行：

[0106]
```
<并行>
      <写标签 编号=’valve inlet.close’ 值=’真’ />
      <写标签 编号=’valve outlet.open’ 值=’真’ />
</并行>
```

```
（<Parallel>
      <WriteTag ref=’valve inlet.close’ value=’true’/>
      <WriteTag ref=’valve outlet.open’ value=’true’/>
</Parallel>
```

[0107] 在某些实施方式中，命令复合可以包括顺序复合和并行复合的组合。例如：

```
<并行>
    <写标签 名称="cmd1" />
    <写标签 名称="cmd2" />
    <顺序>
        <等待 触发="cmd1.done && cmd2.done">
        <设置状态 ... />
    </顺序>
</并行>
```

[0108]

```
(<Parallel>
    <WriteTag name="cmd1" />
    <WriteTag name="cmd2" />
    <Sequence>
        <WaitFor trigger="cmd1.done && cmd2.done">
        <SetState ... />
    </Sequence>
</Parallel>)
```

[0109] 现转向图12，提供了通过执行引擎来执行的帧循环300的实施方式。帧循环300是运行受控的时间段(如，每秒30次)的一组计算机可读指令。帧循环300的目标是对提供给图11的执行引擎264的数据变化(如，状态增量43)做出反应。如所示出的，帧循环300可以评估框302处的表示。例如，通过数据获取线程303来提供表达数据(如，数据对象的值)，数据获取线程303可以对控制与监视系统24的状态数据进行访问。帧循环基于所评估的表达来在框304处对触发条件(如，图11中的触发条件268)进行评估。如果基于所评估的表达将触发条件268中的任意一个评估为真，则可以对与触发条件268相关联的命令(如，图11中的命令脚本)进行调度或执行。如在下面所更加详细地讨论的，关于图13，可以在帧循环300内执行某些命令，并且可以在其他线程或线程池(如，用户输入线程305和线程池307)中对其他命令进行调度及执行。帧命令或被调度以在帧循环300中运行的命令在框106处执行。接着，在框308处执行任何过渡更新(如，如何从一个值变化到另一个值的计算机可读指令)。过渡更新的一个示例可以包括表示状态变化的图形动画，例如，示出了针对打开阀的流动的动画箭头，或针对图像化地表示的最近状态变化的淡出的动画箭头。然后，帧循环300可以提供由所执行的命令应用的变化(如，提供更新的屏幕图像和/或新的数据值)。

[0110] 如上面所讨论的，帧循环300可以运行受控的时间段(如，30次每秒)。在一些实施方式中，可以通过以给定的时间间隔来跳过帧循环的一部分100来调节帧循环300执行。例如，假定帧循环300每秒运行30次，则可以将帧循环300设计成每三帧运行表达评估(框102)，可以从第二帧开始在每个第三帧处对触发器进行评估(框304)，并且可以从第三帧开始每三帧提供过渡更新(框308)。该提供(框310)可以继续在每个帧处执行，或可以最佳地仅当变化出现时运行。因此，框中的每一个依然可以按顺序执行，但是被调节成以较小的频率(如，1/3频率或每秒10帧)来执行。

[0111] 此外，可以基于运行执行引擎264的硬件来修改帧速率。例如，在一些实施方式中，如果利用较低能力的处理器如基于ARM®的系统，则帧循环可以以每秒12帧来运行，如果使用基于Atom的系统，则帧循环可以以每秒30帧来运行，如果使用台式计算机，则帧循环可以以每秒60帧来运行，以及如果使用基于浏览器的系统，则帧循环可以以每秒24帧来运行。此外，过渡选项可以允许较少的过渡(如，每6帧1次)和/或可以允许取决于所使用的平台而允许过渡呈现得较少(如，不是每帧)。执行引擎264还可以适于基于所确定的帧循环300的各种阶段的执行时间来在运行期间对帧循环300进行调节。例如，表达繁重的屏幕可能需要更多的表达评估时间，并且过渡繁重的屏幕可能需要更多过渡处理/执行时间。

[0112] 现转向关于如何调度命令以执行的讨论，图13示出了用于根据实施方式调度命令的处理320。当在框322处将触发条件268评估为真时，开始调度处理320。如先前所讨论的，可以存在与触发条件268相关联的一个或更多个命令。取决于与触发条件268相关联的命令的类型，处理320可以采取两个路径中的一个。命令可以是帧命令324或线程命令326。帧命令324影响主要的帧循环300上的数据。为了在主要的帧循环300上执行，可以将帧命令324添加到帧命令列表326。然后，可以在主要的帧循环300(图12中的框306)上执行帧命令324。通常，这些命令改变需要数据的重提供的数据。因此，可以先于提供(图12中的框310)来执行这些命令。

[0113] 线程命令326是不对帧循环300执行的存储空间中的数据进行访问的命令。可以在与帧循环300不同的线程上自由调度这些命令。因此，如果触发条件268针对线程命令326评估为真，则调度线程命令以在线程池307中运行。通过利用线程池307可以更加有效地利用资源。例如，通过从帧循环300线程中保持线程命令326，帧循环300可自由执行更重要的命令和/或必须运行在帧循环300上的命令。

[0114] 虽然在本文中仅示出和描述了本发明的某些特征，但是本领域技术人员会想到许多修改和变化。因此应当理解，所附权利要求意在覆盖所有这些落入本发明的实质精神内的修改和变化。

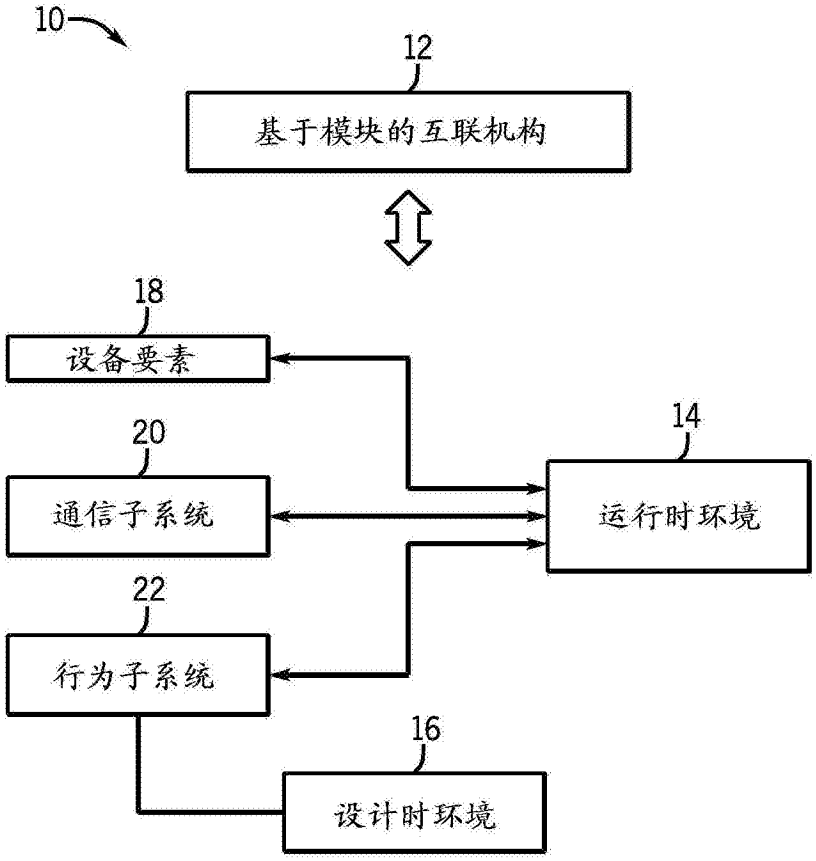


图1

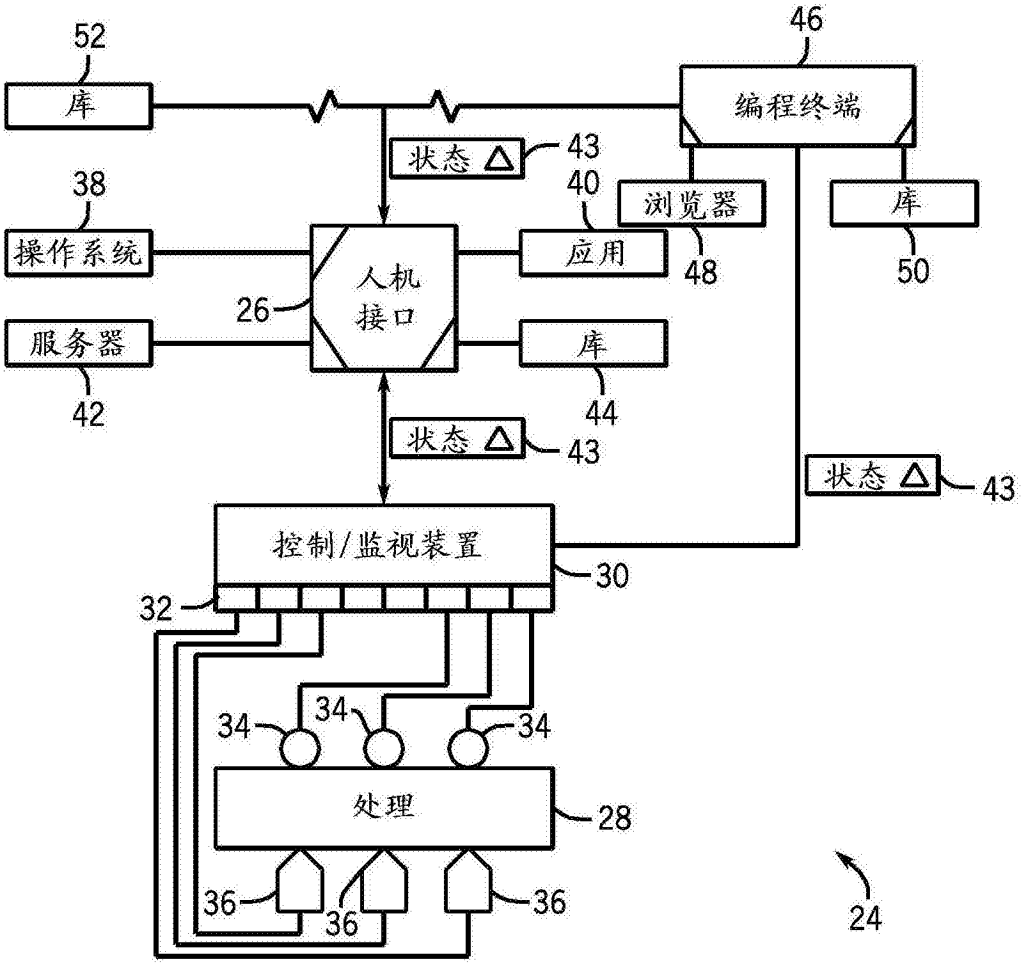

52
库
46
编程终端
状态 △
43
38
操作系统
40
浏览器
库
48
50
人机
接口
26
应用
服务器
库
42
44
状态 △
43
状态 △
43
控制/监视装置
30
32
34
34
34
处理
28
36
36
36
24

图2

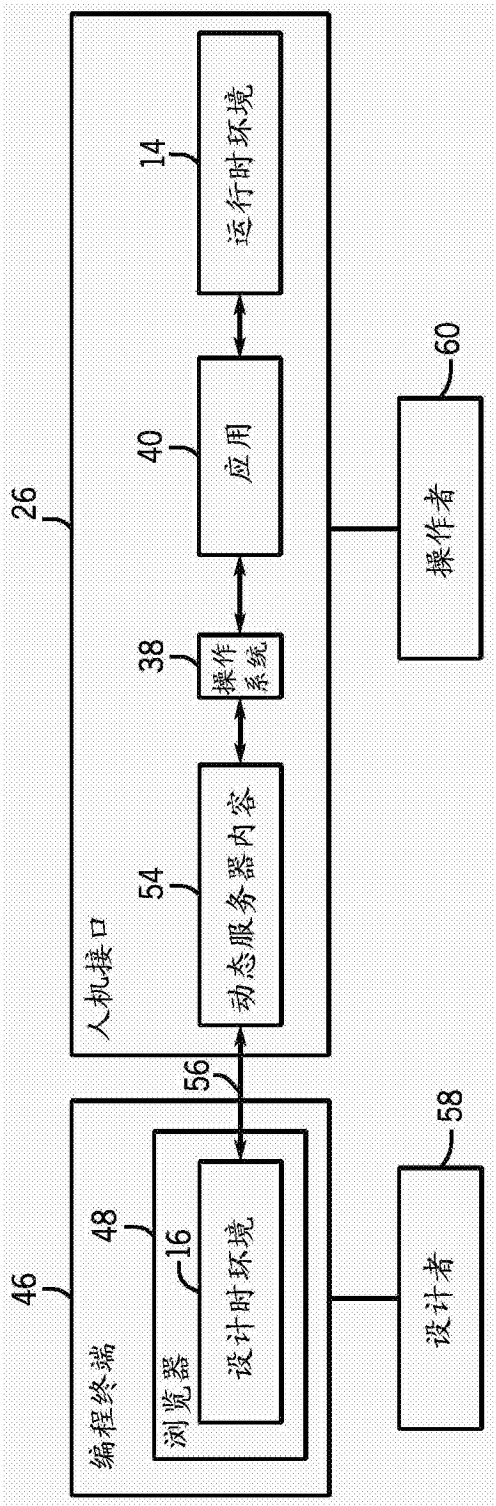

编程终端
46
浏览器
48
设计时环境
16
人机接口
26
动态服务器内容
54
操作系统
38
应用
40
运行时环境
14
56
设计者
58
操作者
60

图3

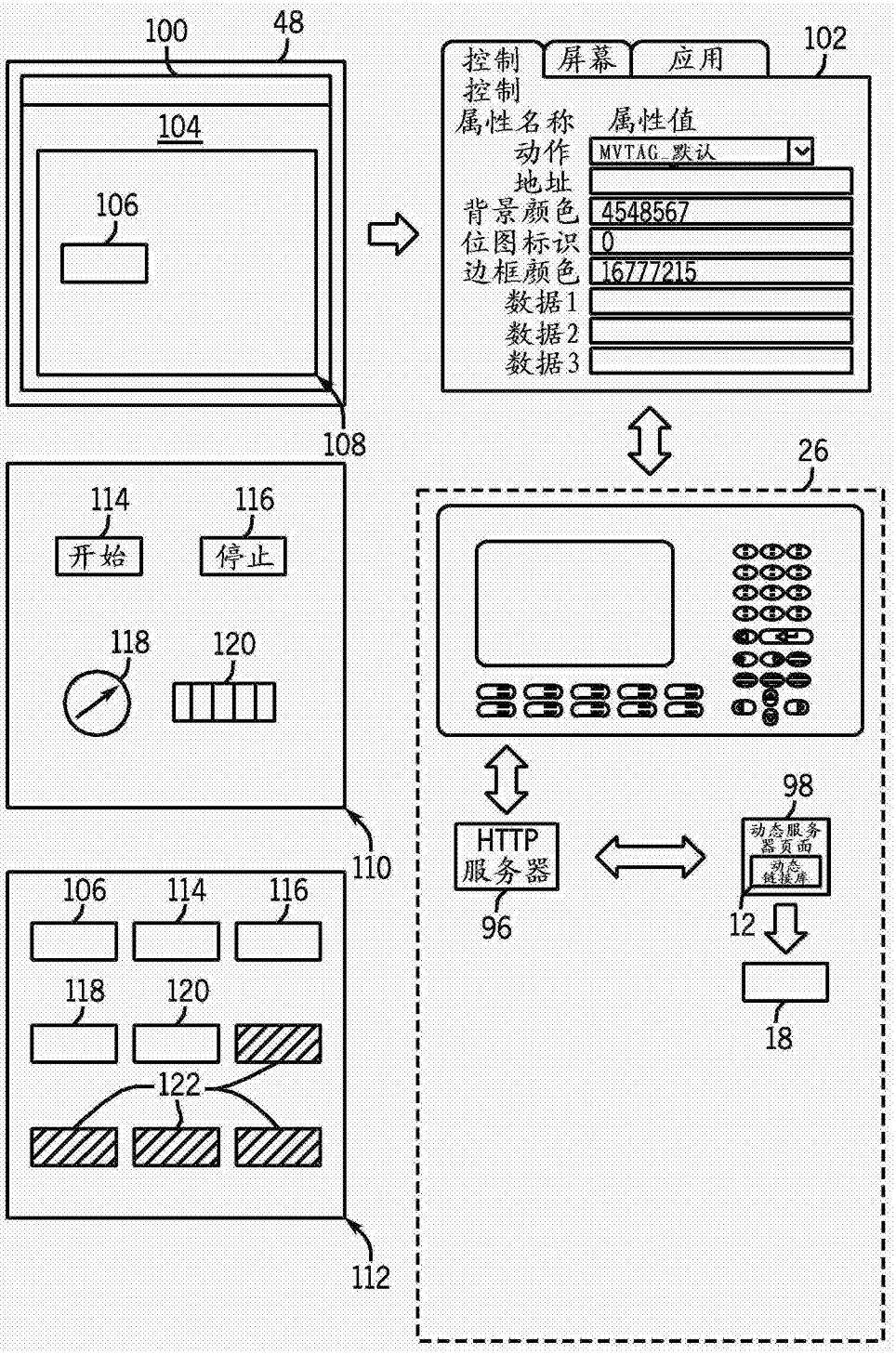

100
48
104
106
108
102
控制
屏幕
应用
控制
属性名称
属性值
动作
MVTAG_默认
地址
背景颜色
4548567
位图标识
0
边框颜色
16777215
数据1
数据2
数据3
26
114
116
开始
停止
118
120
110
106
114
116
118
120
122
112
HTTP
服务器
96
98
动态服务器页面
动态链接库
12
18

图4

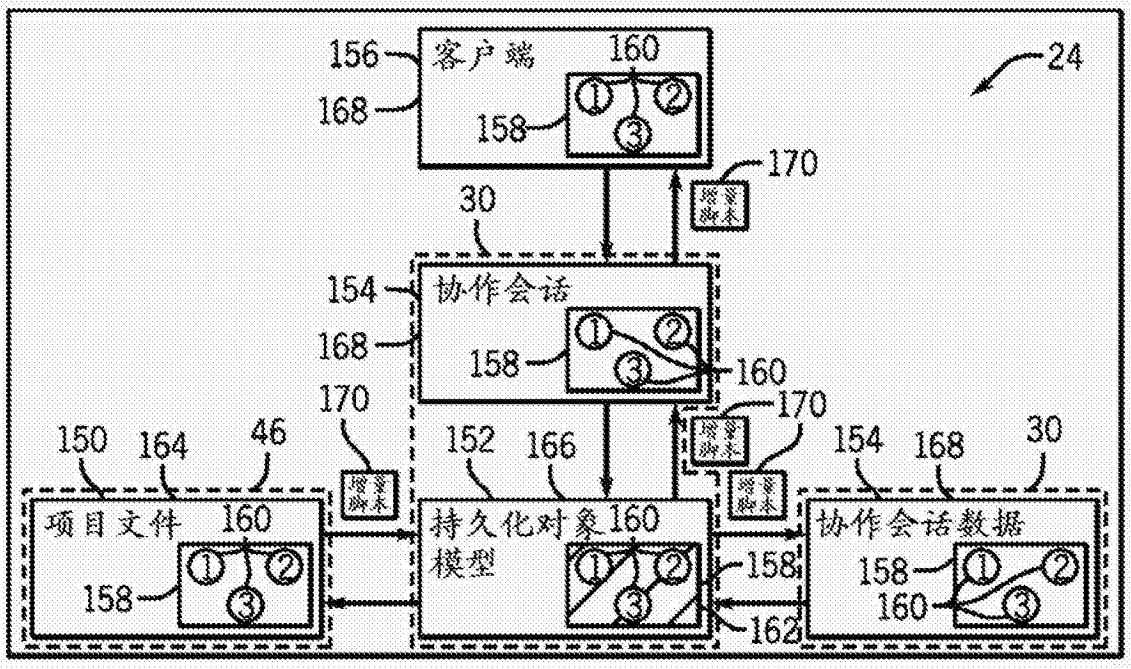

156
客户端
160
168
158
24
170
30
154
协作会话
168
158
160
170
170
150
164
46
152
166
154
168
30
项目文件
160
158
持久化对象
模型
160
158
158
162
协作会话数据
160

图5

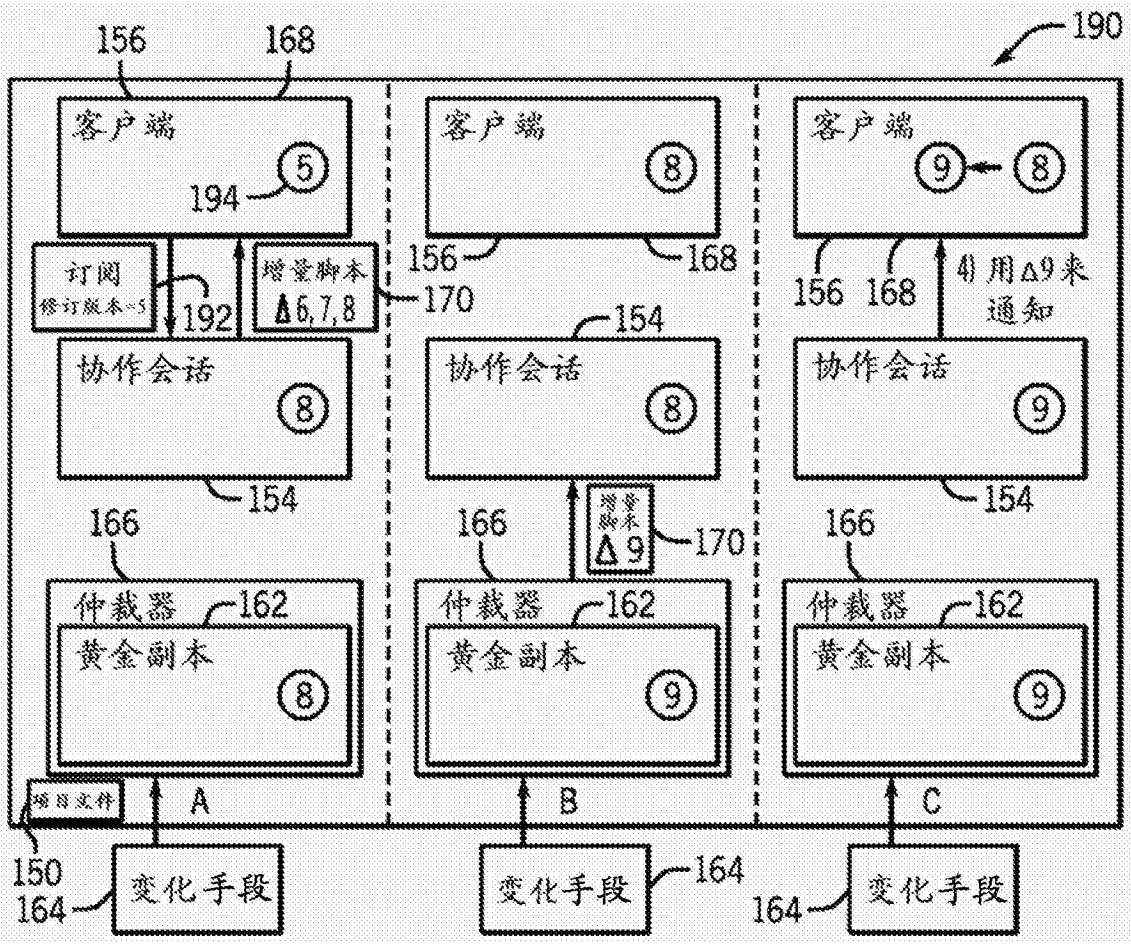

156
168
190
客户端
5
194
客户端
8
客户端
9
8
订阅
修订版本=5
192
增量脚本
Δ6,7,8
170
156
168
156
168
4)用Δ9来
通知
协作会话
8
154
协作会话
8
154
协作会话
9
154
166
166
增量脚本
Δ9
170
166
仲裁器
162
黄金副本
8
仲裁器
162
黄金副本
9
仲裁器
162
黄金副本
9
项目文件
A
B
C
150
164
变化手段
变化手段
164
164
变化手段

图6

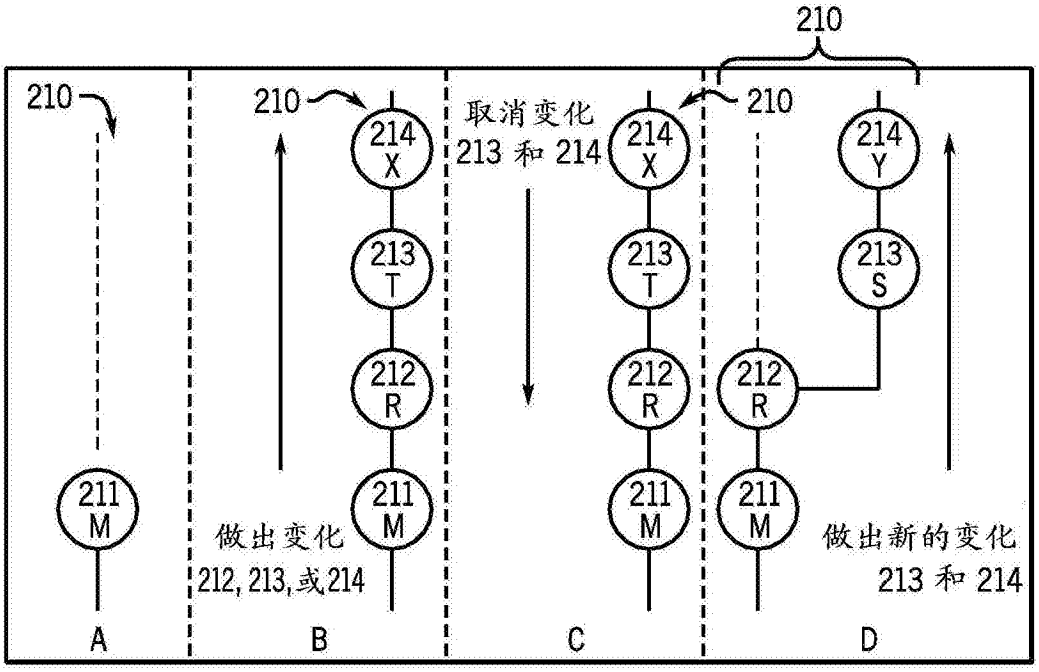

210
210
210
210
214
X
213
T
212
R
211
M
214
X
213
T
212
R
211
M
214
Y
213
S
212
R
211
M
取消变化
213 和 214
做出变化
212, 213, 或214
做出新的变化
213 和 214
A
B
C
D

图7

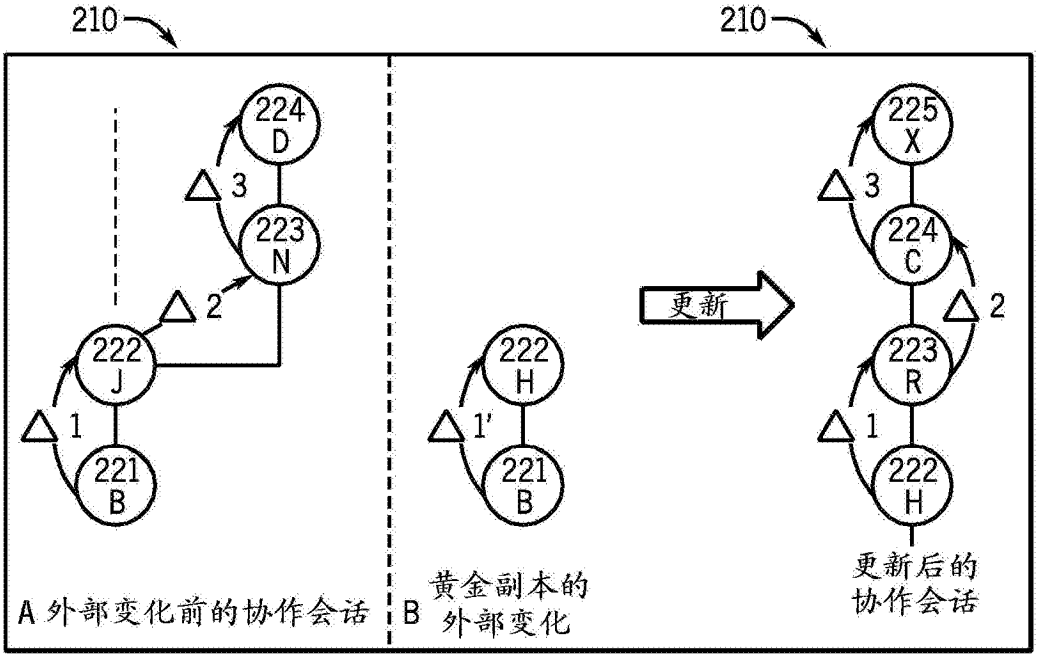

210
210
224
D
3
223
N
2
222
J
1
221
B
222
H
1'
221
B
更新
225
X
3
224
C
2
223
R
1
222
H
A 外部变化前的协作会话
B 黄金副本的
外部变化
更新后的
协作会话

图8

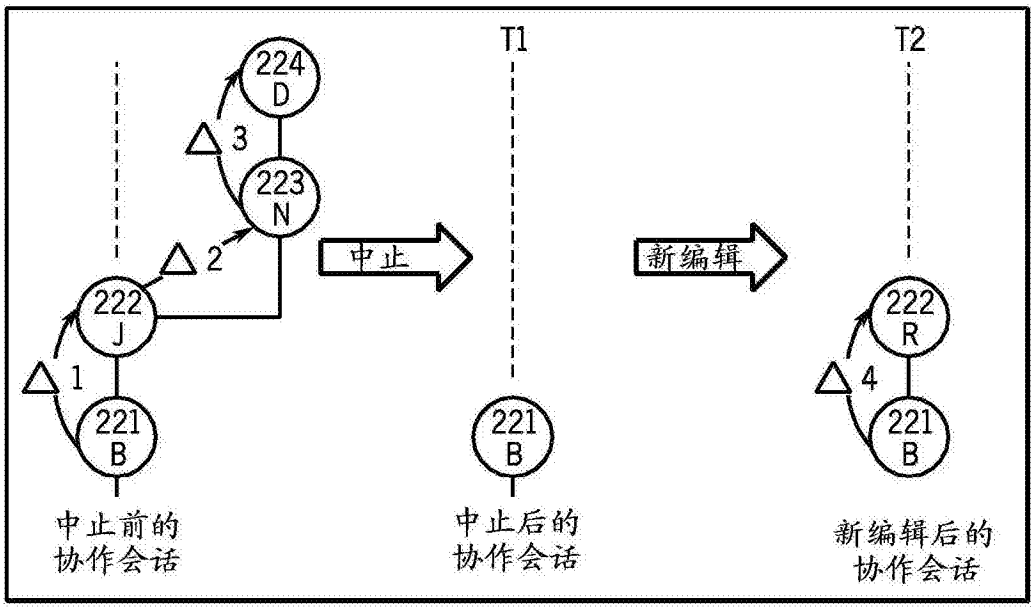

T1
T2
224
D
3
223
N
2
222
J
1
221
B
中止
新编辑
222
R
4
221
B
221
B
中止前的
协作会话
中止后的
协作会话
新编辑后的
协作会话

图9

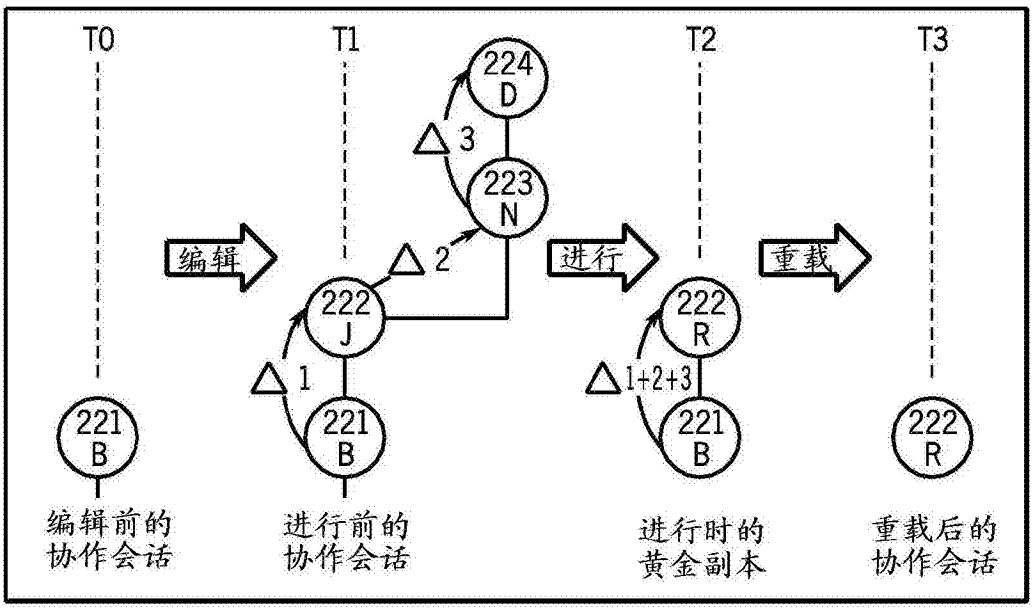

T0
T1
T2
T3
224
D
3
223
N
编辑
2
进行
重载
222
J
1
222
R
1+2+3
221
B
221
B
222
R
编辑前的
协作会话
进行前的
协作会话
进行时的
黄金副本
重载后的
协作会话

图10

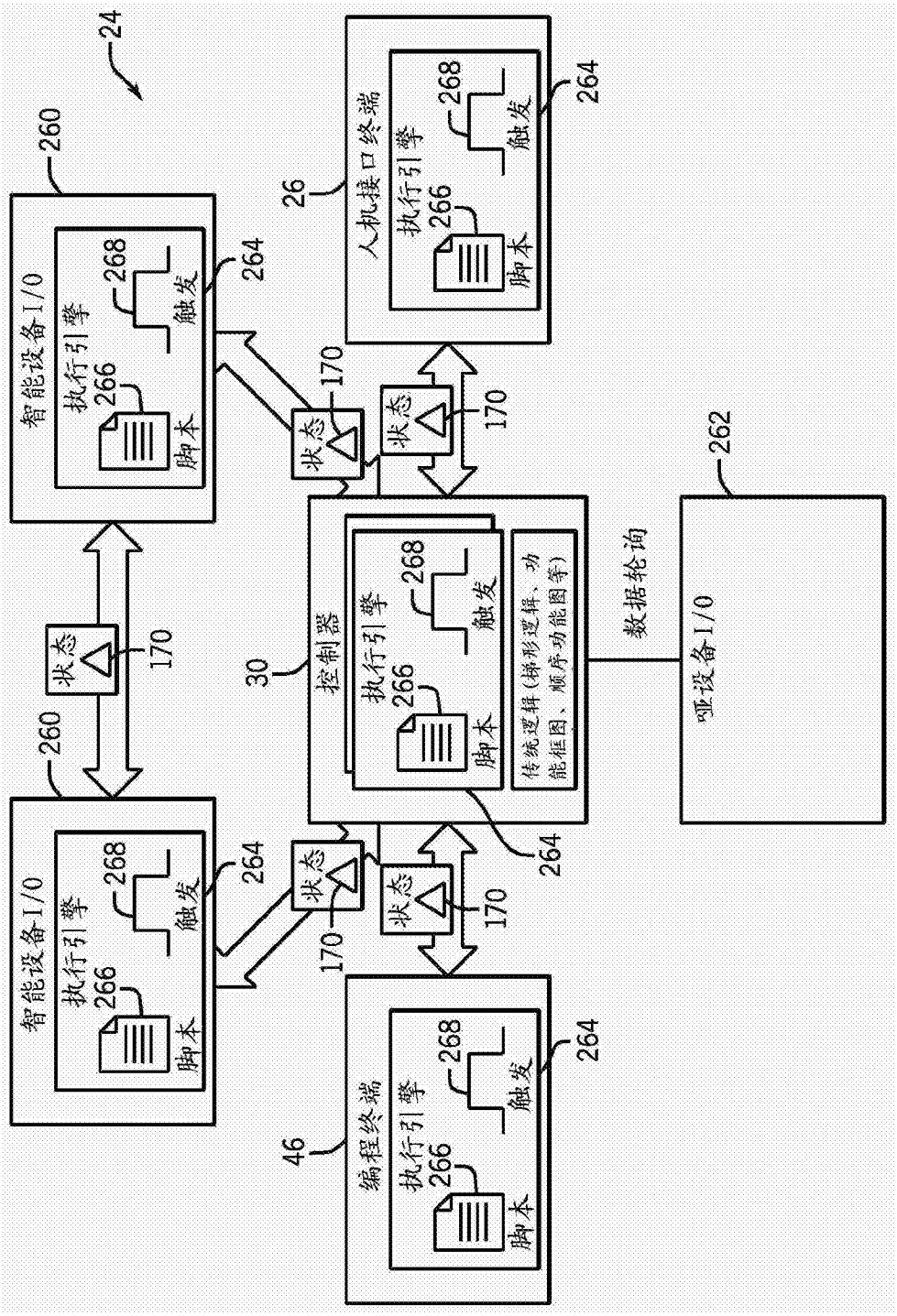

24
260
智能设备I/O
执行引擎
268
触发
266
脚本
264
26
人机接口终端
执行引擎
268
触发
266
脚本
264
状态
170
状态
170
262
30
控制器
执行引擎
268
触发
266
脚本
传统逻辑(梯形逻辑、功能框图、顺序功能图等)
264
数据轮询
哑设备I/O
状态
170
260
智能设备I/O
执行引擎
268
触发
266
脚本
264
状态
170
状态
170
46
编程终端
执行引擎
268
触发
266
脚本
264

图11

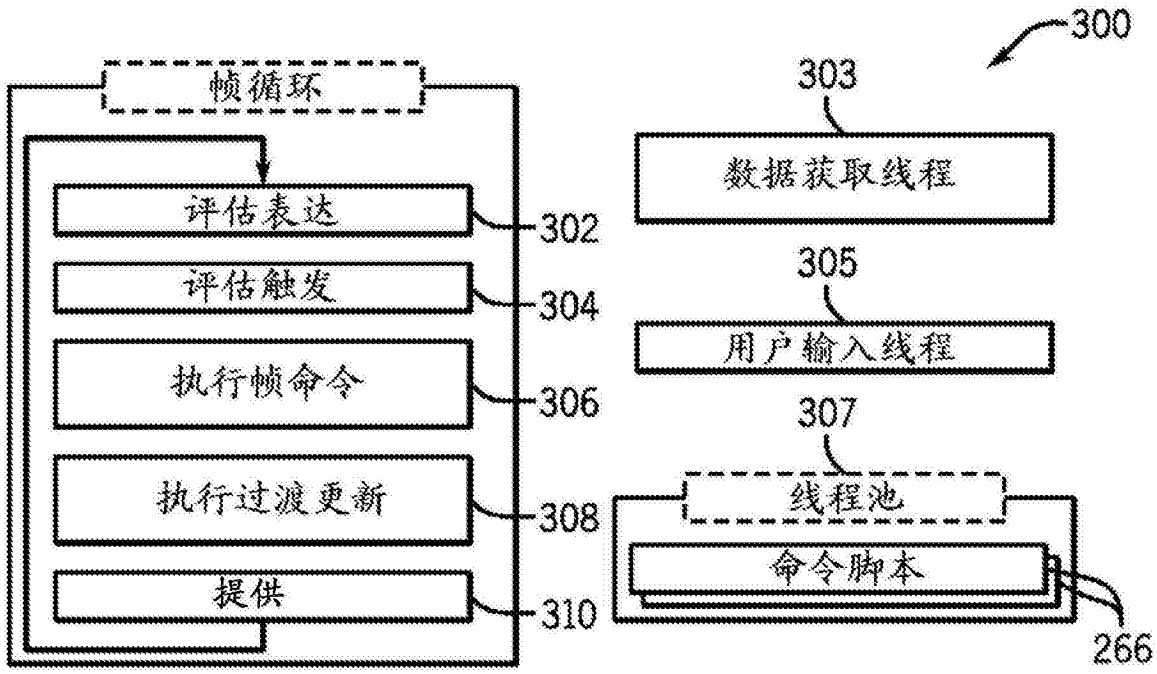

300
帧循环
303
数据获取线程
评估表达
302
评估触发
304
305
用户输入线程
执行帧命令
306
307
执行过渡更新
308
线程池
命令脚本
提供
310
266

图12

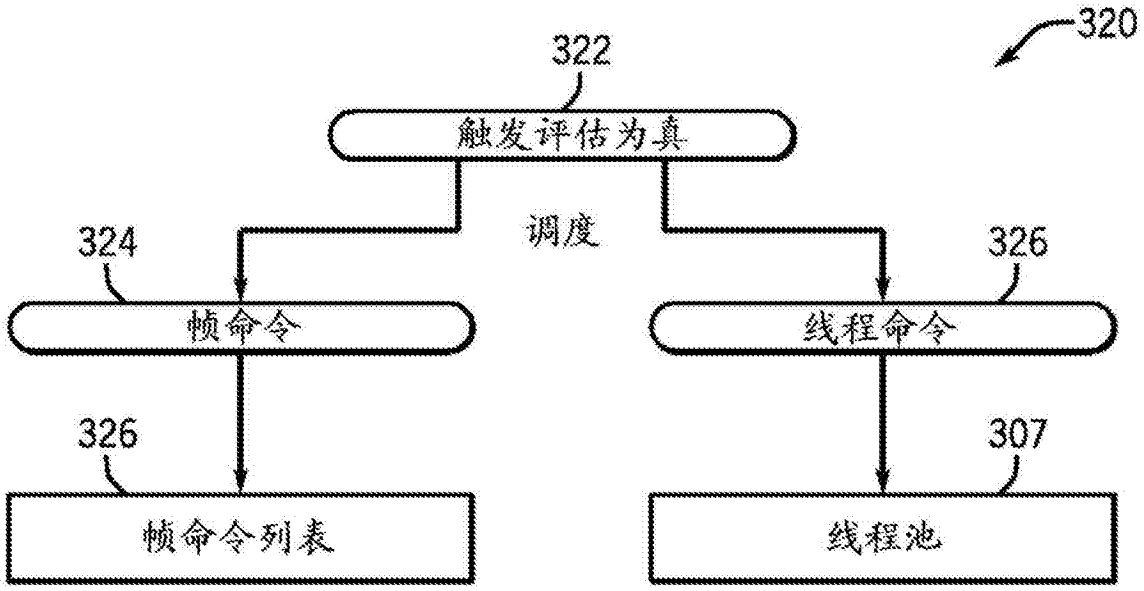

320
322
触发评估为真
调度
324
326
帧命令
线程命令
326
307
帧命令列表
线程池

图13