(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 101563671 B
(45)授权公告日 2013.10.30

(21)申请号 200780045489.2
(22)申请日 2007.10.12
(30)优先权数据
11/549,824 2006.10.16 US
(85)PCT申请进入国家阶段日
2009.06.08
(86)PCT申请的申请数据
PCT/US2007/081246 2007.10.12
(87)PCT申请的公布数据
WO2008/048896 EN 2008.04.24
(73)专利权人 因文西斯系统公司
地址 美国马萨诸塞州
(72)发明人 R·C·莫迪 K·卡萨简 D·特拉小 J·R·安德森
(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 李向英
(51)Int.Cl.
G06F 9/44(2006.01)
G05B 15/00(2006.01)

(56)对比文件
US 6957439 B1,2005.10.18,第2栏第65-67行、第3栏第5段、第6栏第3段、第6栏第55-60行、第8栏第1段.
US 6957439 B1,2005.10.18,第2栏第65-67行、第3栏第5段、第6栏第3段、第6栏第55-60行、第8栏第1段.
US 2006/0069946 A1,2006.03.30,第[0004]段.
US 2003/0023333 A1,2003.01.30,全文.
US 2006/0199572 A1,2006.09.07,全文.
US 2006/0059478 A1,2006.03.16,全文.

审查员 张涛

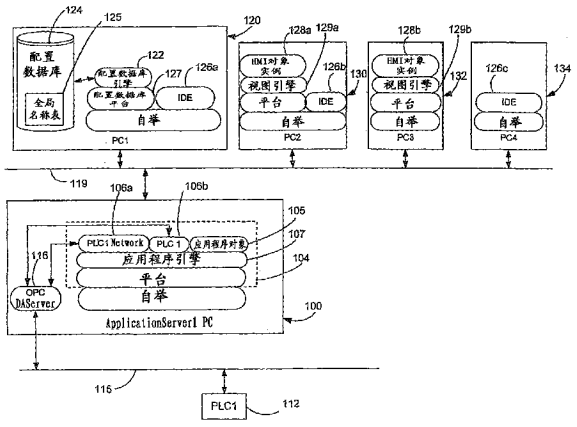
权利要求书2页 说明书26页 附图7页

(54)发明名称

桥接过程自动化和信息管理环境中的人机界面技术

(57)摘要

在对象管理、图形技术,以及HMI应用程序的命名空间处理程序中,实现工业控制和自动化人机界面(HMI)技术迁移方案。对于HMI图形,支持第二技术的新的特点,而同时保留了实现第一技术的系统的功能,包括将以第二技术开发和/或管理的图形导出到第一技术的能力。提供了功能的组合,以提供第一和第二HMI两种图形技术:命名空间集成、图形再现集成,以及HMI应用程序管理集成。



CN 101563671 B

1. 一种用于管理工业控制和自动化的人机界面 (HMI) 应用以便第一 HMI 技术和第二 HMI 技术共存于集成的应用开发和执行环境中的系统，所述系统包括 ：

导入装置，用于将根据第一 HMI 技术定义的基于第一 HMI 技术的 HMI 应用结合到有助于以所述第二 HMI 技术管理所述基于第一 HMI 技术的 HMI 应用的形式中 ；

图形再现装置，其基于第一 HMI 技术并且包括便于呈现根据第二 HMI 技术定义的图形的另外的组件 ；

命名空间管理装置，其支持由第一 HMI 技术和第二 HMI 技术使用的不同的命名约定 ；和

配置数据库装置，用于维护列表，所述列表标识每一个 HMI 应用程序模板和在其内嵌入了每一个符号模板的符号模板，其中所述列表有助于将对符号模板的改变传播到所有 HMI 应用程序和在其内嵌入了改变的符号模板的符号模板。

2. 如权利要求 1 的系统，其中所述第一 HMI 技术是 INTOUCHHMI 技术。

3. 如权利要求 1 的系统，其中所述导入装置通过将在第一 HMI 技术中定义的 HMI 图形封装在由第二 HMI 技术支持的 HMI 对象模板内，将基于第一 HMI 技术的 HMI 应用结合到有助于以所述第二 HMI 技术进行管理的形式中。

4. 如权利要求 3 的系统，其中所述 HMI 对象模板是包裹对象模板。

5. 如权利要求 4 的系统，其中所述包裹对象模板包括便于由视图引擎托管相应的包裹对象实例的属性。

6. 如权利要求 1 的系统，其中所述图形再现装置的另外的组件支持呈现 GDI+ 图形。

7. 如权利要求 1 的系统，其中第一 HMI 技术利用基于本地标签的命名空间引用。

8. 如权利要求 7 的系统，其中给从第一 HMI 技术迁移到第二 HMI 技术的引用分配标识所述引用起源自第一 HMI 技术的前缀。

9. 如权利要求 1 的系统，其中第二 HMI 技术利用全局配置数据库命名空间引用。

10. 如权利要求 1 的系统，还包括用于将在第二 HMI 技术中开发的图形转换为第一 HMI 技术中的图形的导出工具。

11. 一种由计算机系统执行的用于管理工业控制和自动化的人机界面 (HMI) 应用以便第一 HMI 技术和第二 HMI 技术共存于集成的应用开发和执行环境中的方法，所述方法包括 ：

由导入装置将根据第一 HMI 技术定义的基于第一 HMI 技术的 HMI 应用结合到有助于以所述第二 HMI 技术管理所述基于第一 HMI 技术的 HMI 应用的形式中 ；

由基于第一 HMI 技术并且包括另外的组件的图形再现装置呈现根据第二 HMI 技术定义的图形 ；

由命名空间管理装置支持由第一 HMI 技术和第二 HMI 技术使用的不同的命名约定 ；和

在配置数据库装置中维护列表，所述列表标识每一个 HMI 应用程序模板和在其内嵌入了每一个符号模板的符号模板，其中所述列表有助于将对符号模板的改变传播到所有 HMI 应用程序和在其内嵌入了改变的符号模板的符号模板。

12. 如权利要求 11 的方法，其中所述结合步骤包括通过将在第一 HMI 技术中定义的 HMI 图形封装在由第二 HMI 技术支持的 HMI 对象模板内，将基于第一 HMI 技术的 HMI 应用修改为便于以第二 HMI 技术进行管理的形式。

13. 如权利要求 12 的方法，其中所述 HMI 对象模板是包裹对象模板。

14. 如权利要求 13 的方法，其中所述包裹对象模板包括便于由视图引擎托管相应的包裹对象实例的属性。

15. 如权利要求 11 的方法，还包括由导出工具将在第二 HMI 技术中开发的图形转换为第一 HMI 技术中的图形的步骤。

桥接过程自动化和信息管理环境中的人机界面技术

技术领域

[0001] 本发明一般涉及联网的计算机化工业控制和自动化系统领域。更具体来说，本发明涉及管理级别的控制和制造信息系统。这样的系统一般在过程控制系统中的管理控制层的上游执行，以为较低级别的控制元件，例如，可编程逻辑控制器或分布式控制系统 (DCS) 提供指导。还可以使用这样的系统获取和管理涉及这样的过程以及它们的关联的输出的历史信息。

背景技术

[0002] 工业界越来越依赖于高度自动化的数据收集和控制系统，以确保工艺过程是有效地、安全地并可靠地运行，同时又能降低它们的总的生产成本。数据获取开始时，许多传感器测量工艺过程的各个方面，并将它们的测量值报告回数据收集和控制系统。这样的测量结果以各式各样的形式存在。例如，由传感器 / 记录器产生的测量值包括 :温度、压力、pH、材料的质量 / 容积流量、经过特定机器 / 过程的物品的统计，在运输线中等待的包装的带标记的库存，周期完成等等。先进的过程管理和控制软件检查与工业过程关联的传入的数据，产生状态报告和操作摘要，在很多情况下，通过向驱动装置 / 控制器发送修改工业过程的至少一部分的操作的事件 / 操作员指令来做出响应。由传感器产生的数据还允许操作员执行许多监督任务，包括 :响应变化的外部条件（包括原材料的成本），修整进程（例如，指定新设置点），检测效率低的 / 非最佳工作状态和 / 或即将发生的设备故障，采取补救措施，如根据需要，让设备开始运行和停止运行。

[0003] 典型的工业过程非常复杂，接收的信息量比任何人以信息的原始形式可能消化的要多得多。例如，未听说过用成千上万的传感器（模拟 / 数字）和控制元件（例如，阀动器、马达等等）监视 / 控制工厂内的多级进程的各个方面。传感器具有各种类型，并报告进程的各种特征。它们的输出在测量值的含义、为每一个测量值发送的数据量，以及测量的频率方面是不同的。至于后者，为了准确，并迅速响应，这些传感器 / 控制元件中的某些每秒钟进行一次或多次测量。当乘以成千上万的传感器 / 控制元件时，大量的定期读数导致这么多的数据流入控制和制造信息管理系统，需要先进的数据管理和进程可视化技术 / 应用程序。

[0004] 当今，存在高度先进的人机界面 / 进程可视化系统，它们链接到诸如上文所描述的传感器和控制器之类的数据源。这样的系统获取并消化（例如，过滤）上文所描述的过程数据。被消化的过程数据又驱动可视化应用程序，再现 / 呈现过程的图形视图，以便由操作员进行观察。这样的系统的一个示例是著名的 Wonderware **IN-TOUCH®**人机界面 (HMI) 软件系统，用于可视化并控制各式各样的工业过程和制造信息。**IN-TOUCH®** HMI 过程可视化应用程序包括特定过程的一组图形视图以及其物理输出。每一个视图又包括一个或多个图形元件。图形元件潜在地“形成动画”，它们的显示状态响应关联的 / 链接的数据源，随着时间的推移而改变。例如，提炼过程的视图潜在地包括油罐图形元件。油罐图形元件具有可视指示器，显示油罐内包含的液体的液面，图形元件的液面指示器响应由

罐内液面传感器提供的表示油罐内的液面的数据流，上升和下降。由数据流内的不断地变化的过程数据值驱动的动画图形图像（油罐液面指示器只是一个示例）对于观察者来说，理解起来比数字流容易得多。由 HMI 应用程序所提供的图形图像还用于描述并有助于修改当前的过程设定值。因此，诸如 IN-TOUCH 之类的过程可视化系统，成为监督过程控制和制造信息系统的基本组件。

[0005] InTouch HMI 使用户能快速而轻松地开发他们的过程的自定义图形视图。用户可以利用 WONDERWARE 的 WindowMaker 图形视图编辑程序中的各种工具开发图形，包括：标准图形组件、显示、动画、位图图像、ActiveX 控件、包含成千上万的预先配置的工业图像的图形库、SmartSymbol 技术、标记定义、I/O 配置、绑定、脚本、报警和历史配置。

[0006] 通常，用户使用 InTouch 来开发监视控制和数据收集系统应用程序和 HMI 应用程序。用户使用 InTouch 开发他们的自定义应用程序，通过 HMI 软件连接到诸如 PLC 之类的工厂设备的源，实时地可视化工厂数据和状态。为开发 InTouch 应用程序，用户需要定义到 PLC 的实时数据连接，标记数据库、图形开发、图形动画和报警定义。

[0007] HMI 应用程序几十年来用于监督控制、面板和控件。使用 HMI 应用程序开发实用程序来开发 HMI 应用程序，开发实用程序允许用户为他们自己的特定需求 / 应用程序创建特定配置（这里称为 HMI 应用程序）。因此，由软件供应商设计和开发 HMI 开发实用软件。此后，最终用户使用 HMI 开发实用程序来再现潜在大量的 HMI 应用程序，包括针对特定过程自动化和制造信息安装的特殊化需求的视图和关联的功能。

[0008] 尽管创新和提供新的工艺产品十分重要，但是，提供从现有的技术到新的产品的迁移路径也同样重要。HMI 技术以及它们所在的系统也在不断地发展。典型的制造自动化 HMI 应用程序定义包括许多配置的元件，包括：显示、标记、I/O 绑定、PLC 连接、动画、脚本、报警和事件，历史配置。因此，HMI 技术的发展给使用现有的技术创建了大量的 HMI 应用程序的用户带来了潜在的问题。

[0009] 为鼓励用户采用较新的技术，HMI 实用程序开发人员提供了迁移路径，允许用户在采用较新的平台的系统中利用他们的以前创建的 HMI 应用程序。这样的开发人员的一般的方法是提供一些工具，用于从现有的 HMI 应用程序提取现有的 HMI 应用程序的配置信息，并导入将在新的工艺平台上运行的 HMI 应用程序。

发明内容

[0010] 鉴于需要为过程自动化和制造信息系统提供第一和第二 HMI 技术之间的有效的迁移路径，发明了提供用于在两种技术之间迁移的共存方法的方法和基础架构。对于 HMI 图形，支持第二技术的新的特点，而同时保留了实现第一技术的系统的功能，包括将以第二技术开发和 / 或管理的图形导出到第一技术的能力。

[0011] 根据示范性实施例，提供了功能的组合，以接纳第一和第二 HMI 两种图形技术：命名空间集成、图形再现集成，以及 HMI 应用程序管理集成。关于图形，导入工具接收根据第一技术定义的图形，并支持转换为第二技术。此外，还通过添加根据第二技术定义的用于处理图形的组件，增强根据第一技术的用于显示图形的编辑器和查看器。所产生的图形编辑器和查看器支持在同一视图内显示根据第一和第二技术两种技术定义的图形。

[0012] 关于 HMI 应用程序管理，根据第一技术定义的 HMI 应用程序封装在由第二技术支

持的 HMI 对象模板内。实现了第二技术的系统使用其工具通过它们的 HMI 应用程序包裹对象操纵 / 管理根据第一技术定义的 HMI 应用程序。

[0013] 关于命名空间集成，与第一技术关联的标记被映射到 HMI 技术命名空间（例如，向旧名称添加前缀，以标识第一技术），而同时保留根据第一技术定义的原始应用程序中提供的地址信息。

附图说明

[0014] 尽管所附的权利要求详细地阐述了本发明的特点，但是，通过下面的结合附图对本发明的详细描述，可以更好地理解本发明，以及其目标和优点，其中：

[0015] 图 1 是描述了示范性监督过程控制网络的示意图，包括多层监督过程控制和制造信息应用程序，包括一组个人计算机，其中具有视图引擎和关联的人机接口 (HMI) 应用程序对象；

[0016] 图 2 描述了在平台上托管应用程序的多层对象托管布局和实现了本发明的示范性系统内的引擎；

[0017] 图 3 描述了视图引擎对象自定义基本要素的属性的示范性集合；

[0018] 图 4 描述了 HMI 对象自定义基本要素的属性的示范性集合；

[0019] 图 5a 概述了 HMI 应用程序对象模板 / 实例和可嵌入的符号模板之间的一组关系；

[0020] 图 5b 描述了与符号模板关联的级的序列，包括向在其中嵌入更改的符号模板的 HMI 应用程序模板传播对符号模板的更改；

[0021] 图 6 概述了潜在地在 HMI 应用程序模板上执行的一组功能；

[0022] 图 7 概要描述了实现了上一代和下一代 HMI 技术的迁移方案；

[0023] 图 8 说明性地描述了迁移方案的导入 / 导出功能，因为它涉及配置数据库和关联的导入 / 导出功能；

[0024] 图 9 包括同时支持以前的和以后的 HMI 技术的说明性 HMI 应用程序编辑器视图；

[0025] 图 10a 和 10b 说明性地描述了相对于 HMI 技术的旧的和新的版本的命名空间处理方案；以及

[0026] 图 11 包括一组符号以及它们的关联的迁移的名称的说明性示例。

具体实施方式

[0027] 下面的描述基于本发明的实施例，不应该理解为对这里没有显式描述的备选实施例作出限制。例如，本发明可以包括在监督过程控制和制造信息应用程序开发和运行时环境内，其中，单个数据源（例如，工艺设备和关联的逻辑）通过应用程序对象来表示。在 Resnick 等人在 2002 年 6 月 24 日提出的标题为“SUPERVISORYPROCESS CONTROL AND MANUFACTURING INFORMATIONSYSTEM APPLICATION HAVING A LAYEREDARCHITECTURE”的美国申请系列号 10/179,668 中详细地描述了这样的系统的示例，在此引用了该申请的全部内容作为参考，包括其中标识的 / 包含的任何引用的内容和原理。然而，如那些精通相关技术的人所理解的，鉴于所说明的示范性实施例，本发明潜在地适用于各种备选监督过程控制和制造信息应用程序开发和运行时环境。

[0028] 这里的说明主要针对用于集中地管理潜在地包括运行以前定义的 HMI 应用程序

的单独的实例的许多联网的HMI节点的监督过程控制和制造信息应用程序环境内的HMI应用程序（例如，INTOUCH应用程序）的基础架构和相关的方法。说明书包括对封装在可重复使用的HMI应用程序模板内的HMI应用程序的描述。此后，从HMI应用程序模板实例化HMI应用程序对象，并安装在指定的联网的HMI节点中。

[0029] 这里所说明的集中地管理HMI应用程序的第二方面涉及将构成HMI应用程序模板的一部分图形的符号的更改传播到一组HMI应用程序对象模板中。例如，在HMI应用程序的外面全局地定义了符号模板。符号图形通过对集中地管理的符号模板的引用，被集成到中HMI应用程序模板中。使用符号模板来定义HMI应用程序的符号图形，有助于将对符号模板的改变（使用如前所述的交叉引用列表）传播到所有子符号模板以及通过引用包括改变的原始和派生的子符号模板的所有HMI应用程序模板。下面将参考图5进一步描述这样的关系和传播路径。

[0030] 这里所说明的集中地管理HMI应用程序的第三方面涉及维护并通过IDE 126用图形方式呈现配置数据库124的内容的各种视图（例如，部署、派生、模型等等）中的HMI对象的状态。当前状态的示例包括：签入/签出、部署/撤销部署，以及改变。这些示范性状态中的每一个都可使用户对于HMI应用程序的分布式实例作决定。

[0031] 所说明的集中管理安排的再一个方面是用户从远程部署的配置工具（如集成的开发环境（IDE）功能）对现有的HMI应用程序定义（模板）进行编辑的能力。

[0032] 请参看图1，示意图描述了组件在示范性分布式/联网监督过程控制环境内托管/分层关系。在示范性网络中，多个计算硬件节点（PC100，120，130，132，134）中的每一个都运行自举软件，该软件作为随后加载的平台对象的宿主和这里称为IDE功能的开发工具操作。此后，平台对象实例安装在PC上。在每一个PC上只能安装一个平台对象。平台对象托管了服务，并提供给随后安装的引擎对象。引擎对象又潜在地作为随后安装的HMI、设备集成和应用程序对象的宿主来操作。引擎对象的区别在于不同的服务/宿主功能-如此，它们托管的对象的类型也不同。例如，视图引擎托管了HMI对象实例，而应用程序引擎托管了设备集成对象和应用程序对象。下面进一步描述上文所提及的各种对象。

[0033] 继续参考图1，多个PC 120、130和134运行集成的设计和开发工具（IDE 126a-c）。开发人员使用IDE 126来配置监督过程控制和制造信息系统的组件（包括应用程序对象），并将它们部署到连接到工程网络119的指定的PC节点。IDE 126是一个实用程序（潜在地包括多个组件），用来定义、创建过程控制和制造信息应用程序（包括应用程序对象和引擎），并部署到各种平台/引擎中，包括，例如，应用程序服务器PC 100。监督过程控制和制造信息应用程序的开发人员，通过IDE 126，执行各式各样的应用程序设计功能，包括：导入新对象和模板类型，从现有模板配置新的模板，定义新的应用程序对象，将应用程序对象部署到宿主应用程序引擎（例如，应用程序服务器PC 100上的AppEngine1）。IDE 126也是定义HMI模板（包括以前开发的HMI应用程序）的地方，所产生的HMI对象被实例化并部署到具有以前安装的视图引擎（例如，视图引擎129a和129b）的目标PC。

[0034] IDE 126副本对存储在配置数据库124（例如，星系数据库）中的一组对象模板进行操作，其中，定义的对象模板的名称被保留在全局名称表125中。全局名称表125有助于将与位置无关的对象名称绑定到位置派生的句柄，有助于在图1中所描述的系统内的对象之间路由消息。对于配置的应用程序组件，配置数据库124存储对象数据以及与配置的对

象关联的任何代码或文档。配置数据库 124 存储了图 1 中所描述的基本对象模板和各种对象的派生模板（例如，应用程序引擎、应用程序对象、视图引擎和 HMI 对象）。下面参考图 5 描述了示范性可视化 HMI 应用程序对象派生和实例创建方案。在示范性实施例中，配置数据库 124 包括 MICROSOFT SQL 服务器。

[0035] 通过配置数据库引擎 122 访问配置数据库 124，也称为星系储存库的内容。配置数据库引擎 122 通过 IDE 126 副本，通过配置数据库 124 中的每一个定义的对象的用图形方式可呈现的签入 / 签出状态描述符，支持远程多用户访问。配置数据库引擎 122 还支持将对象和软件从集中源部署到系统上的其他节点。

[0036] 在说明性实施例中，配置数据库引擎 122 由配置数据库平台 127 托管。配置数据库平台 127 一般与安装在系统中的 PC 上的其他平台相同。然而，给配置数据库平台 127 分配了系统内的唯一状态（以及对应的名称），作为与单一活动配置数据库 124 关联的平台。如此，所说明的系统包括单一集中地管理的配置数据库。在备选实施例中，在系统中的多个节点上维护了数据库 124 的内容的多个副本（例如，数据库 124 的内容的只读或备份副本）。在说明性实施例中，配置数据库平台 127 和托管的配置数据库引擎 122 执行下列特殊功能：数据 / 软件分发，维护全局名称表 125、解析（使用名称表 125）全局唯一位置 - 独立引用字符串到位置派生的句柄（用于消息交换），管理安全性 / 对多用户环境中的限制的访问，版本控制，集中许可证管理和导入 / 导出对象模板和实例。

[0037] IDE 126 支持涉及配置数据库 124 的各种配置操作。例如，工程师利用 IDE 126 将新对象模板导入到配置数据库 124 中（通过配置数据库引擎 122），配置新对象模板，并将对象部署到工程网络 119 上的指定的 PC 上。如上文所指出的，驻留在不同的网络节点上的 IDE 126 的多个副本能够访问和编辑对象定义，包括 HMI 应用程序定义和潜在地包括到 HMI 应用程序定义中的符号定义（模板）。

[0038] 在说明性示例中，在多个硬件节点 (PC130 和 PC132) 上部署多个 HMI 对象实例 128a-b。多个 HMI 对象实例 128a-b，正如以下通过参考图 4 进一步描述的，基于通过设备集成和驻留在工厂地板网络 115 上的设备 / 控制器的应用程序对象获取的信息，提供代表进程 / 工厂或其部分的当前状态的图形视图 / 窗口。单一视图引擎托管了多个不同的 HMI 对象实例，它们对应于由例如连接的现场设备或 PLC（例如，PLC 112）所提供的信息驱动的各种配置的进程 / 工厂视图。在示范性实施例中，HMI 对象实例 128a-b 由多层监督过程控制和制造信息系统体系结构中的视图引擎 129a-b（下面参考图 3 进行描述）托管。尽管对于图 1 中的每一个视图引擎只显示了一个 HMI 对象实例，但是，每一个视图引擎都能够同时托管多个 HMI 对象实例。

[0039] HMI 对象实例 128 和对应的视图引擎 129 之间的托管的关系促进了对视图引擎 129 支持的某些服务的访问。例如，视图引擎 129 独立地支持更新托管的 HMI 对象实例 128（当对应的模板更新时，自动地进行改变传播）。此外，视图引擎 129 还高速缓存（在关联的网络节点上）与 HMI 对象实例 128 关联的显示。

[0040] 请参看工程网络 119 上的应用程序服务器 1PC 100，在说明性实施例中，数据源是，例如，以应用程序对象 105 的形式呈现的。应用程序对象 105 执行各种功能，包括，代表工艺设备的状态和关联的应用程序逻辑。应用程序对象在位于所显示的分布式分层的监督过程控制和制造应用程序体系结构的应用程序级别时，执行各种监视 / 控制功能中的

任何一个。位于应用程序级别也位于层次结构中的设备集成对象 106a 和 106b，代表诸如 PLC(PLC1) 之类的工厂地板网络上的数据源，智能现场设备，以及关联的 I/O 网络（例如，PLC1 网络）。

[0041] 应用程序对象和设备集成对象彼此在本地（在单一个人计算机内）以及通过与连接到工程网络 119 的个人计算机上托管的对象的非本地通信，彼此进行通信。

[0042] 例如，应用程序对象 105，例如，在由配置数据库 124（例如，WONDERWARE 星系储存库）内维护的全局名称表 125 内进行标识，配置数据库 124 的内容对开发人员可用（通过，例如，IDE 126a-c 和 HMI 对象实例 128a-b，它们，例如，包括 INTOUCH 应用程序以及它们的关联的显示）。如此，根据本发明的实施例，以 INTOUCH 应用程序的形式存在的工厂 / 进程的动态图形视图，最初被使用，例如，WINDOWMAKER 实用程序创建。此后，整个 INTOUCH 应用程序被包括到 HMI 对象模板中，包括用于这里所描述的多级别的应用程序执行环境的必需的组件。在配置数据库 124 中存储 / 维护 / 管理所产生的 HMI 对象模板。此后，基本模板的随后派生的版本是作为子模板来维护的，与父 HMI 对象模板保持了继承关系。原始的和派生的模板可以通过 IDE 126 分发到包含以前安装的视图引擎（例如，视图引擎 129a）的网络 119 上的适当的节点。

[0043] 继续参考图 1，应用程序服务器 1 PC 100 执行包括第一部分 104 的多层监督过程控制和制造信息应用程序。应用程序部分 104 包括应用程序对象 105 以及设备集成对象 PLC1Network 106a 和 PLC1 106b。PLC1Network 106a 设备集成对象用于配置数据访问服务器（例如，OPC DAServer 116）。PLC1 106b 设备集成对象，作为 OPC 客户端操作，访问 OPC DAServer 116 的缓冲区内的数据位置。数据访问服务器 116 和设备集成对象协作地从工厂地板网络 115 上的诸如 PLC（例如，PLC1 112）之类的外部过程控制组件或其他现场设备（未描述）导入并缓冲数据。应用程序引擎 107 托管了应用程序对象 105 以及设备集成对象 106a 和 106b。应用程序引擎 107，作为宿主，管理被托管的应用程序以及设备集成对象的定期 / 事件驱动的执行。下面参考图 2，描述 PC 100 上的分层的托管布局的如前所述的组件。

[0044] 在说明性示例中，通过数据访问服务器 116 提交数据请求，以从 PLC1 112 检索数据。此后，检索到的数据供 HMI 对象实例 128a 和 128b 使用，以驱动代表，例如，工厂地板设备的状态的图形显示。数据访问服务器 116 的数据缓冲区被在个人计算机 100 上执行的各种应用程序级别的对象（例如，应用程序对象 105、PLC1Network106a、PLC1 106b 等等）访问（直接 / 间接地）。应用程序对象的示例代表数据源和逻辑，包括，例如，分离的设备、模拟装置，现场引用、事件 / 触发器、生产事件等等。在示范性实施例中，由应用程序级别的对象 105、106a 和 106b 获得的 / 所提供的信息存储在运行时（历史）进程信息数据库中（未显示）。此后，由 HMI 对象实例 128a-b 获取数据，以驱动动画的进程图形的显示状态。

[0045] 数据访问服务器 116 是，例如，OPC 服务器。然而，所属领域的技术人员将轻松地认识到潜在地由数据访问服务器 116 执行的各种自定义和标准化数据格式 / 协议。此外，示范性应用程序级别的设备集成对象 106a 和 106b，通过与数据访问服务器 116 的连接，代表 PLC 网络和 PLC 本身的操作。然而，由应用程序引擎 107 托管的应用程序级别的对象（例如，设备集成和应用程序对象）包括在监督过程控制和制造信息应用程序的上下文中执行所需的监督控制和数据获取 / 集成功能的可执行的对象的类的几乎无限的谱。

[0046] 监督过程管理和制造信息系统通过各种通信信道，潜在地与各种进程 / 工厂信息源集成。包括多层应用程序（包括部分 104）的示范性系统可通信地连接到 PLC1 112。而 PLC1 又通过工厂地板 115 接收工厂设备状态信息。在特定实施例中，PLC 112 包括 PC 100 所连接到的以太网 LAN 上的节点。在其他实施例中，PLC 112 直接链接到 PC 100 上的物理通信端口。在其他备选实施例中，PC 100 从现场 I/O 模块接收数据，而现场 I/O 模块，例如，从在分布式管理控制系统中操作的现场设备接收模拟数据。

[0047] 值得注意的是，图 1 中和上文中所描述的系统只是监督过程控制和制造信息系统的包括多层体系结构的系统。进一步需要注意的是，图 1 是作为在安装的包括软件和物理计算硬件的组件之间的托管和 / 或包含相互关系的逻辑视图来呈现的。这里所说明的系统几乎适用于任何网络拓扑。例如，本发明适用于其中配置实用程序和监督过程控制可视化应用程序在链接到被控进程的单一计算机系统上运行的系统。

[0048] 请参看图 2，类图形描述了分层的软件的分层的托管布局，包括计算机可执行的指令，这些指令与执行监督过程控制和制造信息应用程序的至少一部分的计算机（例如，PC 100）关联。计算机在层次结构的最低级别执行诸如 MICROSOFT WINDOWS 之类的操作系统 200。操作系统 200 托管了自举对象 202。自举对象 202 被加载到计算机上，并与由操作系统 200 执行的起动过程相关联地激活。作为平台类对象 204 的宿主，自举对象 202 必须在平台类对象 204 的启动操作之前激活。自举对象 202 启动和停止平台类对象 204。自举对象 202 还提供由平台类对象 204 使用的服务，以启动和停止由平台类对象 204 托管的一个或多个引擎对象 206。

[0049] 平台类对象 204 是一个或多个引擎对象 206 的宿主。在本发明的实施例中，对于一个或多个引擎对象 206，平台类对象 204 代表执行特定操作系统的计算机。平台类对象 204 维护了部署在平台类对象 204 上的引擎对象 206 的列表，启动和停止引擎对象 206，如果引擎对象 206 崩溃，还重新启动它们。平台类对象 204 监视引擎对象 206 的运行状态，并向客户端发布状态信息。平台类对象 204 包括系统管理控制台诊断实用程序，用于在执行平台类对象 204 的计算机系统上执行诊断和管理任务。平台类对象 204 还向分布式报警子系统提供报警。

[0050] 引擎对象 206 托管了一组应用程序对象 210，这些对象实现与应用程序关联的监督过程控制和 / 或制造信息获取功能。引擎对象 206 启动所有应用程序对象 210。引擎对象 206 还借助于调度器对象 208，预定应用程序对象 210 相对于彼此的执行。引擎对象 206 向调度器对象 208 注册应用程序对象 210，以便执行。调度器对象 208 基于由对应的一个引擎对象 206 指定的配置，相对于其他应用程序对象，执行应用程序对象。引擎对象 206 监视应用程序对象 210 的操作，并将不正常工作的那些置于隔离状态。引擎对象 206 通过将由自动化对象作出的对运行时应用程序的改变保存 / 恢复到配置文件，支持检验点。引擎对象 206 维护了一个名称约束服务，该服务将属性引用（例如，tank1.value.pv）绑定到适当的一个应用程序对象 210。引擎对象 206 对于托管的设备集成对象，执行类似的功能。

[0051] 引擎对象 206 最终控制如何执行关联的应用程序对象 210。然而，一旦引擎对象 206 确定了应用程序对象 210 的执行时间表，则由调度器 208 控制它们的执行的实时调度。调度器 208 支持包含方法 RegisterAutomationObject() 和 UnregisterAutomationObject() 的界面，这些方法可使引擎对象 206 向调度器 208 的预定

操作的列表中添加 / 从中删除特定的应用程序对象。

[0052] 应用程序对象 210 包括各式各样的对象，这些对象执行业务逻辑，在，例如，工业过程控制系统的环境中，促进执行特定过程控制操作（例如，打开泵，驱动阀门），和 / 或信息收集 / 管理功能（例如，基于接收到的现场设备输出信号值，产生报警）。过程控制（自动化）应用程序对象的示例包括模拟输入、分离的器件，以及 PID 循环对象。一类应用程序对象 210 通过设备集成对象（例如，OPC DAServer118），对由诸如 PLC 之类的过程控制系统提供的数据进行处理。还由引擎对象托管的设备集成对象的功能是在过程控制 / 制造信息源和监督过程控制和制造信息应用程序之间提供桥路 / 数据路径。

[0053] 在一个示范性实施例中，应用程序对象 210 包括被引擎对象 206 和调度器 208 访问的应用程序接口。引擎对象 206 访问应用程序对象接口，以初始化应用程序对象，启动应用程序对象，以及关闭应用程序对象。调度器 208 使用应用程序对象接口来启动对应的应用程序对象的预定的执行。

[0054] 已经描述了示范性多层，分层次地排列的监督过程控制和制造信息应用程序中的自举、平台、引擎和应用程序对象之间的关系，值得注意的是，对于构成 HMI 应用程序的多层体系结构（参见，例如，图 1 中的 PC2 130 上的 HMI 应用程序分层体系结构）的对象，也存在类似的关系。

[0055] 请参看图 3，对于视图引擎对象自定义基本要素，标识了示范性属性集合，视图引擎对象自定义基本要素增强了基本引擎的功能，以促进托管已经部署到 PC（例如，PC 130）的一组可用的 HMI 对象实例中的指定的一个。在 Resnick 等人在 2002 年 6 月 24 日提出的标题为“SUPERVISORY PROCESS CONTROL ANDMANUFACTURING INFORMATION SYSTEM APPLICATIONHAVING A LAYERED ARCHITECTURE”的美国申请系列号 10/179,668 中详细地描述了基本引擎基本要素的内容 / 功能，在此引用了该申请的全部内容作为参考。视图引擎对象支持诸如部署、撤销部署、启动和关闭之类的基本引擎功能。视图引擎对象还支持下面进一步描述的可视化应用程序特定的功能。在说明性实施例中，视图引擎对象是只托管了 HMI 对象实例的特殊化引擎对象类型 - 而不是能够托管各种应用程序级别的对象（包括设备集成对象和应用程序对象）的应用程序引擎。

[0056] 视图引擎（例如，视图引擎 129a）托管并安排指定的 HMI 对象实例的执行。视图引擎基于当前占有的视图引擎运行时状态，对于托管的 HMI 对象实例，支持一组运行时操作。当视图引擎处于启动状态时，托管的 HMI 对象 :从检查点初始化，由视图引擎启动，向 Message Exchange（或其他合适的对象之间的数据通信服务）进行注册，根据由与视图引擎关联的调度器发出的命令而执行。当视图引擎进入扫描启动或扫描关闭状态时，托管的 HMI 对象接收视图引擎的新的扫描状态的通知。此外，当视图引擎进入关闭状态时，托管的 HMI 对象被它们的宿主引擎关闭。

[0057] 在示范性实施例中，视图引擎管理部署到它的 HMI 对象实例的列表。然而，视图引擎不负责调用脚本的执行或读取并写入与 HMI 对象实例关联的相关过程数据。相反，执行脚本以及管理数据预订的过程被委托给包括到（嵌入 / 封装在）对应的 HMI 对象实例的 HMI（例如，INTOUCH）应用程序中。如此，在说明性实施例中，不能在图 1 中所描述的多层托管体系结构内执行的以别的方式独立的 HMI 应用程序，被包括到 HMI 包裹对象内，以提供这样的功能。如此，独立的旧式 HMI（INTOUCH）应用程序可以无缝地包括到实现了上文参考图

1 和 2 描述的分层的基于对象的体系结构的系统中。

[0058] 如上文所指出的，视图引擎的自定义基本要素包括一组涉及托管 HMI 应用程序对象的属性。图 3 中标识的属性的集合（下面描述）只是示范性的，随着本发明的备选实施例的不同而不同。

[0059] 在说明性实施例中，值得注意的是，利用一组数据点，这里称为“属性”，定义了对象（例如，平台、引擎、应用程序对象等等）。每一个属性又潜在地包括配置和运行时处理程序，基于属性的当前指定的值，对对象进行处理。在示范性实施例中，处理程序是被触发的事件，具有自定义编码的功能。配置设置处理程序是当使用配置客户端（如 IDE）设置属性时触发的事件，运行时设置处理程序在运行时客户端（如 INTOUCH）设置属性的值时被触发。

[0060] 当指定的 HMI 对象模板被指定部署到视图引擎时，_CreateViewApp 属性 300 创建新的 HMI 对象实例。对新的 HMI 对象实例的引用将被添加到由视图引擎进行管理的部署的 HMI 对象的列表中。

[0061] _DeleteViewApp 属性 302 从视图引擎上目前部署的一组 HMI 对象中删除以前部署的 HMI 对象。从视图引擎上的部署的 HMI 对象的列表中删除对应的对 HMI 对象的引用。

[0062] _StartHostedObjects 属性 308 开始在视图引擎上运行所有部署的 HMI 对象。HMI 对象的初始状态基于从检查点永久存储器中提取的值。

[0063] _StopHostedObjects 属性 310 开始关闭当前由视图引擎托管的所有 HMI 对象实例。

[0064] 请参看图 4，将讨论 HMI 应用程序对象的自定义基本要素的示范性属性的集合。HMI 应用程序对象执行与提供分布式监督过程控制和制造信息应用程序的图形视图部分关联的功能。在上文所描述的分层的运行时环境中的宿主视图引擎上执行的 HMI 应用程序对象，管理包括的 HMI(INTOUCH) 应用程序（该应用程序又提供工厂 / 进程的动态图形视图）的签入 / 签出、编辑、部署，以及运行时属性监视。HMI 应用程序的图形状态由实时数据驱动，而实时数据，例如，由工厂设备传感器、监视器，以及控制器提供的。这样的信息是通过设备集成和在应用程序引擎（参考图 1 所描述的）上执行的应用程序对象从工厂地板网络中提取的。HMI 对象也支持应用程序服务器托管的应用程序级别的对象（通过它们，动态过程数据被传递到其中包括的 HMI 应用程序）上的引用标记 (Message Exchange)。

[0065] 在说明性示例中，执行脚本并管理数据预订的 HMI（例如，INTOUCH) 应用程序被包括到（嵌入 / 封装在）对应的 HMI 应用程序对象模板和实例中。如此，在说明性实施例中，不能在图 1 中所描述的多层托管体系结构内执行的以别的方式独立的 HMI 应用程序，被包括到 HMI 应用程序包裹对象内，该对象有助于将 HMI 应用程序集成在采用了如前所述的托管的分层的运行时环境的系统内（管理，运行等等）。如此，独立的旧式 HMI(INTOUCH) 应用程序可以无缝地包括到实现了上文参考图 1 和 2 描述的分层的基于对象的体系结构的系统中。

[0066] 如前所述的 HMI 包裹对象包括自定义基本要素，包括一组属性，这些属性涉及 HMI 应用程序在由视图引擎支持的托管环境内的执行。图 4 中标识的属性的集合（下面描述）只是示范性的，随着本发明的备选实施例的不同而不同。

[0067] _VisualElementReferenceList 属性 400 包含分配给 HMI 应用程序对象的所有可

视元素（例如，符号）的列表。

[0068] _VisualElementReferenceStatusList 属性 402 指定分配给 HMI 应用程序对象的每一个符号的当前状态。状态可以用来表达 HMI 应用程序对象内包含的符号的各种状态，包括，例如，显示符号何时从 HMI 应用程序对象中删除。

[0069] 当与 HMI 应用程序对象关联的 HMI 应用程序文件正在与配置数据库 124 同步时，DeploymentInProgress 属性 404 被设置为真。

[0070] _UndeployNotify 属性 406 指定是否可以撤销部署 HMI 应用程序对象。

[0071] _StartSyncronization 属性 408 被设置为真，以通知 HMI 应用程序对象。它应该开始将与 HMI 应用程序对象关联的应用程序的 HMI 应用程序文件传输其中部署了 HMI 应用程序对象的节点。

[0072] _SyncStatus 属性 410 指出 HMI 应用程序传输到其中部署了关联的 HMI 应用程序的节点的状态。

[0073] _NameSpace 属性 412 包含关于作为与 HMI 应用程序对象关联的 HMI 应用程序的一部分的参数标记的信息。_NameSpace 属性 412 用于支持在属性浏览器内浏览 HMI 应用程序的标记。

[0074] 就在关联的 HMI 应用程序编辑器关闭之前，写入 _ShutdownNotify 属性 414，确保正在进行中的异步方法在编辑器进程被允许关闭之前完成。

[0075] 当 HMI 应用程序编辑器启动时，写入 _BeginDBMonitoring 属性 416，以确保当编辑会话开始时，正确地加载并验证 HMI 应用程序对象。

[0076] LastModified 属性 418 指定 HMI 应用程序的版本号被修改的最后一次的时间。

[0077] HMI 应用程序对象，例如，表现了随后的描述中概述的运行时行为。当执行 HMI 应用程序对象时（在宿主视图引擎的指示下），包括到 HMI 应用程序对象的逻辑判断包括在 HMI 应用程序对象内的 HMI 应用程序是否需要从配置数据库 124 传输。如果需要启动传输，那么，在通过视图引擎对 HMI 对象进行下一次扫描时开始传输。

[0078] 在启动 HMI 应用程序对象之后的任何时间都可以进行同步。HMI 应用程序对象启动 HMI 应用程序与源应用程序的同步。如果待办的同步操作完成，那么，HMI 对象会设置配置数据库 124 内的属性，以指出传输已经完成。根据本发明的实施例，同步应用程序可以包括更新封装的 HMI 应用程序或在配置数据库 124 内已经更新的包括到 HMI 应用程序中的单个符号对象。在更新 HMI 应用程序的情况下，只从配置数据库 124 中传输配置数据库 124 内的不同于当前在具有包括了 HMI 应用程序的 HMI 应用程序对象实例的节点上的文件的应用程序文件。

[0079] 请参看图 5，该图描述了示范性可视化 HMI 应用程序对象派生和实例创建方案，该方案有助于潜在地分发给网络上的许多节点的 HMI 应用程序对象实例的集中管理。这样的集中管理包括响应对关联的 HMI 应用程序模板的配置的改变，更新以前部署的 HMI 应用程序对象，包括其中包括的符号对象。HMI 应用程序和符号模板的集合存储在，例如，诸如配置数据库 124 之类的集中式配置数据库中。

[0080] 在说明性实施例中，基本 HMI 应用程序对象模板 500 提供了一个框架，从该框架派生了一组派生的 HMI 应用程序对象模板 512a-n，并存储在数据库 124 内。基本 HMI 应用程序对象模板 500 提供了基本可执行代码和数据，用于管理与 HMI 对象实例关联的（封装在

HMI 对象实例内)HMI 应用程序。从基本 HMI 应用程序对象模板 500 派生出来的应用程序对象模板 502a-n 与特定 HMI 应用程序（例如，INTOUCH 应用程序）关联。HMI 应用程序被封装在 HMI 应用程序对象模板内，这些 HMI 应用程序对象模板提供包括多个 HMI 节点的系统内的每一个 HMI 应用程序的可重复使用的副本。在特定示范性实施例中，派生的 HMI 应用程序对象模板 502a-n 中的每一个都与使用独立于 IDE 配置环境执行的 HMI 应用程序编辑器实用程序定义的特定 INTOUCH 应用程序关联。

[0081] HMI 应用程序模板的开发和它们的管理（包括创建和部署实例）是通过潜在地驻留在网络的多个节点（例如，参见，图 1）上的 IDE 组件来处理的。因此，在说明性实施例中，以各种视图（例如，派生）列举了 HMI 对象模板的图形界面可视地显示了每一个对象模板的状态（例如，签入 / 签出 - 用于编辑，部署 / 撤销部署，已改变（在编辑之后））通过提供可视的状态指示，可使开发人员，使用 IDE，快速地确定环境中的特定 HMI 应用程序模板的状态，多个用户可以访问这样的模板，以便审阅、编辑和部署。

[0082] 将 HMI 应用程序封装在 HMI 应用程序模板内，有助于利用由 IDE 126 支持的各种开发视图。视图包括，例如，模型视图（代表工厂地板 / 进程的物理布局），部署视图（网络上的位置和托管的关系），派生视图（代表分层的父 - 子对象模板关系）。在 Resnick 等人在 2002 年 6 月 24 日提出的标题为“SUPERVISORYPROCESS CONTROL AND MANUFACTURING INFORMATIONSYSTEM APPLICATION HAVING A LAYEREDARCHITECTURE”的美国申请系列号 10/179,668 中详细地描述了由 IDE 126 支持的这样的视图，在此引用了该申请的全部内容作为参考，包括其中标识的 / 包含的任何引用的内容和原理。

[0083] 在示范性实施例中，从派生的应用程序对象模板 502（例如，HMI 应用程序对象模板 502a）创建 HMI 应用程序对象实例 504（例如，HMI 应用程序对象实例 504a-m），并部署到指定的视图引擎中。在示范性实施例中，开发人员定义 HMI 应用程序对象模板 502a($UserDefinedInTouchApp1)，然后调用部署实用程序，以创建 HMI 应用程序对象的 m 个实例，并部署到包括潜在地需要 HMI 应用程序的多个监测台的网络上的 m 个节点。

[0084] 说明性实施例还支持独立开发 / 编辑符号（作为符号模板），此后包括到 HMI 应用程序对象模板中。基本符号对象模板 510($Symbol) 提供了一个框架，从该框架定义一组派生的符号对象模板 512a-x，并存储在数据库 124 内。基本符号对象模板 510 为通过引用嵌入在特定应用程序对象模板 502（例如，HMI 应用程序对象模板 502n）内的符号提供基本可执行代码和数据。

[0085] 值得注意的是，尽管图 5a 描述了符号的独立模板，但是，系统支持独立符号模板，由其他对象模板（例如，应用程序对象模板）托管的符号模板，以及由对象实例托管的符号。

[0086] 在说明性示例中，符号模板本身是从基本符号模板 510 派生出来的其他符号的容器对象模板。参考图 5，定义的符号对象模板，如符号模板 512x，可嵌入在另一个符号模板（例如，符号模板 512a）内。符号模板 512（例如，符号模板 512a）也可通过引用嵌入在 HMI 应用程序模板 502（例如，HMI 应用程序模板 502n）中。在 HMI 应用程序对象实例之前，使用 HMI 应用程序模板内的到嵌入的符号模板的引用。此外，配置数据库 124 中还维护了列表，这些列表标识每一个 HMI 应用程序模板和在其内嵌入了每一个符号模板的符号模板。这样的列表有助于将对符号模板的改变传播到所有 HMI 应用程序和在其内嵌入了改变的符号

模板的符号模板。

[0087] 在示范性实施例中，更新机制使用级联更新机制来更新所有受影响的符号和在其内嵌入了改变的模板的 HMI 应用程序模板。如此，通知对符号模板的改变被传播到直接嵌入了符号模板的第一组模板。此后，那些模板被嵌入在其他模板内，或具有子派生的模板，改变通知和更新机制传播到那些受影响的模板。

[0088] 在示范性实施例中，符号模板嵌入在 HMI 应用程序内。在 IDE 可以访问的配置数据库 124 内维护了符号模板和 HMI 应用程序模板，它们具有关联的状态（例如，签入 / 签出、改变等等），有助于多个用户之间的协调编辑，以及对改变的符号模板所在的模板的改变。

[0089] 请参看图 5b，呈现了一组级，这些级概述了符号模板的寿命内的各种兴趣点。最初，在阶段 520，用户从基本符号模板 510 派生符号模板 512x，并将符号模板 512x 添加到由配置数据库 124 维护的图形工具箱中。

[0090] 此后，在阶段 525，当编辑 HMI 应用程序时，从配置数据库 124 中维护的一组对象模板中选择符号模板 512x，并使用与配置数据库 124 关联的浏览器工具列出。要么直接从图形工具箱选择符号模板，要么间接地选择符号模板 512x 所关联到的对象（例如，应用程序对象）。

[0091] 当符号模板 512x 被嵌入到 HMI 应用程序中时，只有对符号模板的引用被持续。当加载 / 部署 HMI 应用程序时，从配置数据库 124 中检索符号图形定义。插入到部署的 HMI 应用程序的版本是其他用户的最后“签入”的版本，或请求定义的副本的当前用户的最后保存的版本。如上文所指出的，参考图 4，HMI 应用程序模板维护了其 _VisualElementReferenceList 属性 400 中的所有嵌入的符号的列表。_VisualElementReferenceList 属性 400 供系统使用，用于传播、部署及其他目的。

[0092] 在符号模板 512x 已经嵌入在 HMI 应用程序 (HMI 应用程序又封装在 HMI 应用程序模板内）内之后，在阶段 530，符号模板 512x 被编辑，以再现改变的符号模板 512x′ 。对符号模板 512x 执行的编辑操作的示例包括 :替换符号文本串（替代字符串）、替换符号数据引用（替代标记）、替换符号图形属性、应用动画、调整大小、移动、删除、剪切、复制、粘贴、复制、对准、分发、制作单元（作为单元的一部分添加），置后、置前等等。此后，改变的符号模板 512x′ 被签入到配置数据库 124 中。

[0093] 在示范性实施例中，IDE 支持交叉引用功能，对于每一个对象模板，该功能提供两组引用 -“谁引用了我”列表和“我引用谁”列表。“我引用谁”引用组标识了符号或 HMI 应用程序模板中的任何嵌入的符号。“谁引用了我”引用组显示了在其内嵌入了符号模板的任何 HMI 应用程序或其他符号模板。IDE 的此功能利用 HMI 模板上的 _VisualElementReferenceList 属性 400，以创建 / 更新 HMI 应用程序模板的交叉引用，例如，当在添加新的符号之后签入符号模板或 HMI 应用程序模板时。

[0094] 此后，在阶段 535，使用使用“谁引用了我”引用列表，将对符号模板的改变（通过潜在地级联的符号模板）传播到包含（或者直接或者通过在其内嵌入了符号的一个或多个其他符号模板）改变的符号模板的每一个 HMI 应用程序模板。在示范性实施例中，当改变的符号被“签入”到配置数据库 124 时，与配置数据库 124 关联的对象管理结构将受改变影响的任何部署的 HMI 应用程序对象实例标记为“待办的改变”。此后，使用远程重新部署机制来更新每一个受影响的实例。然而，只有部署的实例的改变的部分被传输到包含受影响

的 HMI 应用程序实例的运行时节点。

[0095] 将改变传播到 HMI 实例

[0096] 继续参考图 5，分别地定义的 HMI 应用程序对象模板 502（例如，HMI 应用程序模板 502n）和符号模板 512（例如，符号模板 512a）支持将对模板的改变传播到对应的 HMI 应用程序对象实例 504。如此，对嵌入到 HMI 应用程序对象模板的 HMI 应用程序对象模板 502 或符号模板 512 的任何改变都被传播到包含对改变的 HMI 应用程序 / 符号模板的引用的任何 HMI 应用程序对象实例。为促进这样的传播，数据库 124 维护了包含任何 HMI 应用程序 / 符号模板的所有对象实例的列表。如此，当特定 HMI 应用程序 / 符号模板改变时，由配置数据库引擎 122 通知托管了受改变影响的 HMI 应用程序对象的部署的实例的所有视图引擎。此后，改变的对象（或其改变的部分）的新的版本被重新部署到适当的视图引擎上，并重新启动。

[0097] HMI 应用程序在 IDE 环境内的集中管理

[0098] 下面概述了用于在包括 IDE 126 的上文所描述的环境中创建和维护 HMI 应用程序对象的示范性管理方案。在说明性示例中，HMI 应用程序是在 IDE 126 外面开发的。此后，HMI 应用程序通过在系统中的（参见例如，图 1）潜在地任何节点上运行的 IDE 126 的副本，被封装在从基本 HMI 应用程序模板 500 派生出来的 HMI 应用程序模板内。

[0099] 将 HMI 应用程序封装在 HMI 应用程序模板内并维护对 HMI 模板内的 HMI 应用程序的引用，有助于通过 IDE 对 HMI 应用程序的协调的编辑，IDE 支持从运行 IDE 126x 的副本的远程连接的 / 联网的节点，对数据库 124 内的对象进行编辑。此外，通过其 HMI 应用程序模板访问 HMI 应用程序，还有助于应用由上文所描述的配置数据库 124 以及其关联的平台 / 引擎功能支持的并行访问规则 / 状态基础架构（例如，签入 / 签出、部署的 / 未部署的，以及更改的）。

[0100] 在示范性实施例中，HMI 应用程序在 DDE 126 内被表示为应用程序对象模板的特定类型，这里称为 HMI 应用程序对象模板。HMI 应用程序对象模板包含对 HMI 应用程序的引用和有关 HMI 应用程序的行为的特定信息，但是，HMI 应用程序对象模板不将 HMI 应用程序的数据存储在配置数据库 124 内。相反，HMI 应用程序数据保留在由 HMI 应用程序所定义的标准格式的与模板关联的文件储存库目录中（如此，保持了源 HMI 应用程序的格式）。由于存在将 HMI 应用程序对象模板与 HMI 应用程序关联的影响，模板在用户可以做什么和不可以做什么方面存在限制。从 HMI 应用程序对象模板创建的实例也是同样的道理。用户不能改变基本 HMI 应用程序对象模板 500 的任何 HMI 特定的属性。基本模板上的所有其他属性与通过 IDE 126 提供的其他对象模板遵循相同的规则。用户从基本 HMI 应用程序对象模板 500 派生 HMI 对象模板（例如，HMI 模板 502a）派生 HMI 对象模板（例如，HMI 模板 502a），以设置 HMI 应用程序特定的属性。基本模板 500 不支持 HMI 应用程序对象实例（例如，应用程序实例 504a）的直接创建。派生的 HMI 应用程序对象模板以及它们的对象实例由两个单独的数据定义构成：上文所描述的系统框架内的对象定义，以及 HMI 应用程序。

[0101] 下面将参考图 6，描述 HMI 应用程序模板的寿命管理过程中支持的一组函数。派生函数 600 允许用户定义与特定 HMI 应用程序关联的 HMI 应用程序对象模板（封装）。通过派生函数 600，用户将 HMI 应用程序独立）与派生的模板关联。示范性实施例支持多种将 HMI 应用程序与派生模板关联的方式。用于将 HMI 应用程序与 HMI 应用程序对象模板关联

的由 IDE 126 支持的操作的两个示例包括 :创建和导入。这些操作只对 HMI 模板可用,不能对 HMI 应用程序对象模板的实例执行。与 HMI 应用程序模板 / 对象相比,与 HMI 模板关联的 HMI 应用程序不存储在数据库 124 中。相反,HMI 应用程序存储在文件储存库下的目录中(在图 1 中未显示)。此外,使用能够独立于 IDE 126 操作的 HMI 应用程序开发工具(例如,WINDOWMAKER),单独地编辑 HMI 应用程序,并处理其内容。

[0102] 当用户启动 HMI 应用程序开发工具时,提示用户创建新的 HMI 应用程序或导入现有的应用程序。将现有的应用程序导入到 HMI 应用程序对象模板的过程涉及指定现有的 HMI 应用程序在文件系统目录内的位置。这里引用的“导入”操作,在实践中,是复制和关联操作。如此,当用户导 HMI 应用程序,以便创建派生的 HMI 应用程序对象模板时,HMI 应用程序对象模板接收指定的 HMI 应用程序的全部内容的副本,包括子目录,然后存储在与 IDE 126 关联的文件储存库中。一旦在 HMI 应用程序对象模板和 HMI 应用程序之间创建了关联,关联就是永久的,不能被改变。创建与不同 HMI 应用程序的新的关联的过程需要从配置数据库 124 中删除 HMI 应用程序模板以及模板的所有部署的实例。在特定实施例中,对导入 HMI 应用程序的过程施加了某些限制。例如,对于目前与另一个 HMI 应用程序对象模板关联的应用程序、已经与 HMI 应用程序对象模板一起部署的应用程序,以及已经从 HMI 应用程序对象模板导出的应用程序,不允许导入操作。

[0103] 删除操作 602 允许用户通过 IDE 126 从配置数据库 124 删除 HMI 应用程序对象模板。当用户删除 HMI 应用程序对象模板时,完全删除模板和与该模板关联的 HMI 应用程序目录。删除 HMI 模板的过程服从与其他人同时使用模板或者从该模板创建的 HMI 对象实例关联的规则。复制的(源)HMI 应用程序本身不受模板的删除的影响。

[0104] 对于 HMI 应用程序模板或其实例,支持重命名操作 604。重命名 HMI 应用程序对象实例不影响关联的 HMI 应用程序。

[0105] 对于 HMI 应用程序对象模板和实例,支持导出 HMI 模板操作 606。当导出 HMI 应用程序对象模板以便导入到另一个配置的系统(这里被称为“星系”)中时,创建封装文件,包括 HMI 应用程序对象模板以及其关联的 HMI 应用程序的所有必需的数据和文件。在示范性实施例中,符号不包括在封装内。然而,在备选实施例中,关联的 HMI 应用程序内嵌入的任何符号还包括在导出封装中。

[0106] 对于 HMI 应用程序模板内包含的封装的 HMI 应用程序,支持导出 HMI 应用程序操作 607。从存储在配置数据库 124 中的 HMI 模板导出 HMI 应用程序的过程在其以前的独立环境中再现 HMI 应用程序。只希望向独立 HMI 应用程序添加新的 HMI 技术图形的用户可以利用通过派生函数 600 导入独立 HMI 应用程序来做到这一点,但是,不能够利用部署功能(需要 ARCHESTRA 基础架构)。为了将 HMI 应用程序移动到目标机器,用户调用当管理 HMI 应用程序模板时可用的导出操作。当调用导出操作时,提示用户输入目标目录路径。一旦用户执行了此操作并确认了该操作,整个封装的 HMI 应用程序都被置于提供的路径中,包括 :所有 HMI 应用程序窗口、标记名称词典、前代符号、前代局部化数据,以及嵌入的新技术图形。使用被添加的组件增强的以前的 HMI 技术 (INTOUCH) 的查看器实用程序,处理任何新技术 (ARCHESTRA) 图形,以便接纳新的图形技术,以及嵌入的新技术图形数据。如前所述的导入 / 导出序列允许用户包括新技术图形,无需完全迁移到新的 HMI 技术的平台。

[0107] 现在可以在通过添加的组件增强的编辑器中打开导出的 HMI 应用程序,以允许在

断开连接的 / 独立的 HMI 应用程序中在现场预先形成编辑。增强的编辑器允许对以前的和新的 HMI 技术图形进行编辑。对新技术图形的编辑的程度是由添加的组件所提供的增强功能确定的，包括，例如 :调整大小、删除、配置动画、移动、复制，以及剪贴板操作（剪切、复制，以及粘贴）。

[0108] 对于 HMI 应用程序对象模板和实例，支持导入操作 608。当导入 HMI 应用程序对象模板时，模板容器特定的文件和数据被导入到配置数据库 124 中。从用于导入 HMI 应用程序模板的封装文件中提取 HMI 应用程序，并复制到文件储存库中。如果导入的 HMI 应用程序对象用现有的关联的 HMI 应用程序覆盖现有的 HMI 应用程序对象，以前的现有的 HMI 应用程序的所有版本的所有数据都被删除。

[0109] 对于 HMI 应用程序对象模板，支持备份操作 610 和恢复操作 612。当备份包含完全配置的 HMI 应用程序对象模板的系统时，所有关联的 HMI 应用程序数据都包括在备份文件中。备份文件的随后恢复将关联的 HMI 应用程序对象模板数据置于恢复的系统的文件储存库中。

[0110] 支持版本管理 614，以便在配置数据库 124 内维护 HMI 应用程序对象的多个以前的版本。对于非 HMI 对象模板，所有对象配置数据都存储在配置数据库 124 中。然而，在示范性实施例中，HMI 应用程序对象模板的 HMI 应用程序部分存储在配置数据库 124 的外面（然而，模板容器数据存储在数据库 124 中）。存储在数据库 124 内的对象模板的多个版本包括 :签入、签出，以及部署的版本。关联的 HMI 应用程序的对应的版本存储在配置数据库 124 的外面，存储在文件储存库中。

[0111] HMI 应用程序对象模板的版本管理对于签入、签出，以及部署的版本，表现了下面的行为。模板的签入版本代表关联的 HMI 应用程序的最当前的配置版本。在签出 HMI 应用程序对象模板的任何时间，签入的版本都被用作用户进行编辑的起点。在部署实例的任何时间，签入的版本是发送到指定的目的地平台的版本。在签入已签出的 HMI 应用程序对象模板的任何时间，将模板的已签出的版本复制到签入版本。用户从不直接对 HMI 应用程序对象模板的签入版本进行编辑。

[0112] 下面的点描述了 HMI 应用程序对象模板的签出的版本行为。HMI 应用程序对象模板的签出版本代表用户将其签出的用户进行改变的 HMI 应用程序模板的副本。在用户签出 HMI 应用程序对象模板的任何时间，签出的版本是当前签入的版本（在用户进行任何改变之前）的副本。当用户签入 HMI 应用程序对象模板时，签入的版本被用签出的版本覆盖。用户直接对 HMI 应用程序对象模板的签出版本进行编辑。HMI 应用程序对象实例始终被锁在模板里。对于 HMI 应用程序实例，没有签出的状态。对签出的 HMI 应用程序执行“撤消签出”操作，导致当前的签出的版本被丢弃，当前签入的版本用于随后的签出和编辑操作。

[0113] 下面的点描述了 HMI 应用程序对象模板的部署的版本。HMI 应用程序对象模板和关联的 HMI 应用程序的部署的版本代表当前在目标平台上发现的版本。当部署 HMI 应用程序对象模板时，HMI 应用程序的关联的签入的版本被复制到指定的目标平台，当前部署的版本用数据库 124 中的签入的版本覆盖。没有给用户提供对部署的版本的直接编辑访问权限。HMI 应用程序对象模板没有被部署，没有部署的版本。与 HMI 应用程序对象模板关联的 HMI 应用程序的部署的版本只能包含对 HMI 应用程序成功地运行不可缺少的信息。代表备份或只配置文件的任何文件不应该包括在 HMI 应用程序的部署的副本中。这将最小化在部

署过程中必须传输到目标 PC 的数据量。

[0114] 现在注意配置 HMI 应用程序对象模板的过程，包括在单独的 HMI 应用程序设计工具（例如，WONDERWARE′ S WindowMakerHMI 应用程序编辑器）中开发的嵌入的 HMI 应用程序。下面将描述 DDE 126（安装在潜在地远离包含数据库 124 的节点的多个节点）和用于配置 HMI 应用程序对象模板的 HMI 应用程序编辑器（例如，WindowMaker）的组合功能。

[0115] IDE 126 支持 HMI 应用程序对象模板对象上的下列操作 / 工作流。用户最初启动 HMI 应用程序编辑器，以编辑与 HMI 应用程序对象模板关联的 HMI 应用程序。例如，HMI 应用程序编辑器与 DDE 126 分开在单独的进程上运行。然而，在示范性实施例中，当关闭 IDE 126 时，如果 HMI 应用程序编辑器打开，则提示用户保存在 HMI 应用程序编辑器中进行的任何更改。IDE 126 只有在关闭 HMI 应用程序编辑器之后才关闭。在包括安全登录的实施例中，HMI 应用程序编辑器在变为另一个登录用户之前关闭。防止编辑 HMI 应用程序对象模板，而同时关联的 HMI 应用程序正在被编辑。

[0116] 如上面所描述的图 5 中所指出的，潜在地定义了多个 HMI 应用程序对象模板，并存储在数据库 124 中。此外，IDE 126x 的多个副本能够同时在与包含数据库 124 的节点的相同或不同（远程）节点上操作。IDE 126 使用对象模板编辑会话管理来跟踪具有通过 HMI 编辑器编辑的 HMI 应用程序的 HMI 应用程序模板。如此，在说明性实施例中，HMI 应用程序编辑器（例如，WindowMaker）将不会打开特定 HMI 应用程序对象模板，以便在某些情况下进行编辑，如 :HMI 应用程序对象模板被签出到另一个模板，以派生层次结构定义了选定的 HMI 应用程序模板内封装的 HMI 应用程序，但不在其中正在启动 HMI 应用程序的同一个实例或模板内。然而，在这样的情况下，HMI 编辑器将被允许以只读模式打开。

[0117] 下面将根据示范性实施例描述配置 HMI 应用程序对象模板的 HMI 应用程序节点属性。例如，HMI 应用程序节点属性，适用于执行 HMI 应用程序的整个机器，因此，不能从 HMI 应用程序对象模板的 IDE 126 进行编辑。相反，从特定节点上的 HMI 应用程序管理器对 HMI 应用程序节点信息进行管理。

[0118] 下面将针对示范性实施例描述配置 HMI 应用程序对象模板的 HMI 应用程序编辑器的行为。HMI 应用程序对象模板具有两组配置数据 :(1)HMI 应用程序对象模板属性，以及 (2) 关联的 HMI 应用程序数据。使用 HMI 应用程序编辑器（例如，WindowMaker）来配置 HMI 应用程序数据，并持续到配置的系统（星系）的文件储存库中的位置中的文件。在配置之前，HMI 应用程序对象模板与 HMI 应用程序关联，在 HMI 编辑器打开 HMI 应用程序模板（以及其关联的 HMI 应用程序）。HMI 编辑器，例如，除支持编辑 HMI 应用程序之外，还支持编辑 HMI 应用程序对象属性（如模板的描述）。

[0119] 从 IDE 126 的观点来看，HMI 应用程序编辑器是 HMI 应用程序对象的对象编辑器。但是，HMI 应用程序编辑器不是常规对象编辑器，因为其主要功能是定义 / 配置封装在 HMI 应用程序对象模板 / 实例内的 HMI 应用程序。例如，HMI 应用程序编辑器包括下面的功能。HMI 应用程序编辑器没有“保存并关闭”命令。用户关闭编辑器，提示用户保存任何未完成的编辑。当由系统隐式签出 HMI 应用程序对象模板时，“保持签出”选项被设置为假。如果显式签出 HMI 应用程序对象模板，那么，选项被设置为真。当关闭 HMI 应用程序编辑器时，无论是“保存并关闭”或通过“关闭”，“保持签出”选项判断是否执行隐式签入。只有在选项被设置为假的情况下才进行隐式签入。

[0120] 在一个示范性实施例中，下面的行为适用于关闭 HMI 应用程序编辑器。如果“保持签出”选项被设置为假，对于 HMI 应用程序对象模板，执行隐式签入。如果“保持签出”选项被设置为假，并且什么也没有改变，执行隐式撤消签出。

[0121] 在说明性实施例中，HMI 应用程序编辑器还访问 / 编辑 HMI 应用程序对象的属性。通过 HMI 应用程序编辑器，编辑 HMI 应用程序对象的对象特定的数据（而不是 HMI 应用程序数据）。HMI 应用程序编辑器提供配置 HMI 应用程序对象的属性的用户界面（例如，属性列表、描述，以及当前值）。

[0122] 下面概述了支持与 HMI 应用程序对象模板关联的图形的定义的功能。在示范性实施例中，所有支持的图形都被部署到目标节点中，作为部署的 HMI 应用程序对象的一部分。当对象、图形及其他支持组件被作为 HMI 应用程序对象的一部分部署到目标节点上时，它们代表在进行部署时配置数据库 124（以及 HMI 应用程序的文件储存库）的快照。在示范性实施例中，数据库 124 和文件储存库的内容被允许改变（在部署潜在地受这样的改变影响的对象之后）。

[0123] 在示范性实施例中，使用引用列表来确保部署的 HMI 应用程序对象的所有所需的图形都被复制到目标节点中。支持两种引用列表 :隐式的和显式的。对于隐式引用，当一个符号被嵌入到另一个图形中或者在动画中使用窗口时，在组件中更新内部引用列表，确保当部署组件时，包括所有所需的支持图形。这在这里被称为“隐式引用”。例如，自动地创建隐式引用列表（无需用户干预）。由于 HMI 应用程序对象模板 / 实例中的每一个定义的图形视图和嵌入的符号都包括隐式引用列表，当引用具有其自己的引用的视图或符号时，对传播和部署有级联影响。

[0124] 在不自动地生成隐式引用的情况下，使用显式引用列表。在某些情况下，系统不能确定 HMI 应用程序对象 / 模板的图形组件的图形视图 / 符号的一组引用。例如，基于在运行时确定的信息调用动画的按钮上的脚本不会导致任何隐式引用被生成。由于与 HMI 应用程序对象关联的视图的运行时显示仅仅基于当前所部署的，系统不能加载请求的窗口，除非它已经在某些其他动画中被隐式引用。

[0125] 配置和访问封装了 HMI 应用程序的 HMI 应用程序对象模板 / 实例的配置的再一个方面是通过 IDE 126 查看与封装的 HMI 应用程序关联的标记。在示范性实施例中，IDE 126 内的属性浏览器支持浏览与 HMI 应用程序对象模板 / 实例关联的（封装在 HMI 应用程序对象模板 / 实例内）的 HMI 应用程序的标记。浏览器还支持浏览属于 HMI 应用程序对象模板 / 实例本身的命名空间的属性。

[0126] 当通过 IDE 126 的属性浏览器实用程序选择 HMI 应用程序对象实例时，生成列表控件，包括 HMI 应用程序标记名称列和数据类型列。标记名称列，例如，包含 HMI 应用程序标记的名称。列表控件将提供对应于 HMI 应用程序对象模板 / 实例上的属性的任何条目的属性名称。数据类型列指定列表中的条目的 HMI 应用程序标记的数据类型。

[0127] 浏览器包括刷新功能，有助于同步改变的 HMI 应用程序对象模板的部署的实例。如果 HMI 应用程序模板被签出，以便通过 IDE126 进行编辑，用户更新封装在 HMI 应用程序模板内的 HMI 应用程序的标记 / 向其中添加新的标记，那么，用户可以通过属性浏览器浏览 HMI 应用程序对象的属性，一旦用户保存了封装的 HMI 应用程序，就可以看到更改。此外，没有 HMI 应用程序对象被签出的用户将看到与 HMI 应用程序对象模板的当前签入的版本关联

的标记。此外，当标记数据库被刷新时，IDE 126 的属性浏览器将显示对与封装的 HMI 应用程序关联的 HMI 应用程序标记数据库的任何更改。对标记数据库的可能的更改是通过 HMI 应用程序编辑器手动或通过标记的整体导入，向标记数据库添加标记以及删除标记编辑现有的标记以及改变数据类型或者名称，所产生的。如此，通过将 HMI 应用程序封装在配置数据库 126 内维护的 HMI 应用程序对象模板内，在一个示范性实施例中，允许通过在系统（例如，参见图 1）中的潜在的许多节点中的任何一个节点上执行的 IDE 126x 复制，管理与 HMI 应用程序关联的标记。

[0128] 已经描述了支持 HMI 应用程序的封装和集中管理的系统的涉及配置的方面，下面将讨论这样的系统的部署和运行时行为。关于将 HMI 应用程序对象实例部署到诸如图 1 中所描述的网络上的节点，在将任何 HMI 应用程序对象部署到节点之前，部署视图引擎 129。单一平台能够同时托管多个视图引擎 129，而多个 HMI 应用程序对象 128 潜在地分配给单一视图引擎 129。

[0129] 在部署 HMI 应用程序对象实例（以及任何嵌入的符号对象实例）过程中，部署的 HMI 应用程序对象和封装的 HMI 应用程序在目标节点上所需的所有数据和文件根据需要被复制到目标节点。只有那些丢失的或自从最后一次部署以来改变的文件才被复制到目标节点。HMI 应用程序对象实例的部署操作利用组件的签入版本。

[0130] 部署 HMI 应用程序对象实例的过程包括部署容器 HMI 应用程序对象实例和定义封装的 HMI 应用程序的数据。例如，HMI 应用程序数据包括与 HMI 应用程序关联的文件储存库目录中的文件和文件夹。如果 HMI 应用程序对象是以前部署的，则必须假定以前部署的应用程序当前正在使用。如下面所描述的，基于 HMI 对象的“改变模式”，用户具有处理以前部署到目标节点的 HMI 对象的多种选项。“忽略改变”改变模式，用于手动管理使用标记和脚本函数的 HMI 应用程序中的更改，以实现自定义解决方案。当有新的应用程序可用时，名称为 $ApplicationChanged 的离散的（布尔）HMI 应用程序系统标记被设置为真。下面的脚本函数用于接受新的应用程序：

[0131] 1.RestartWindowViewer()- 使与封装的 HMI 应用程序关联的查看器立即关闭，然后自动地重新启动。在封装的 HMI 应用程序重新启动时，部署到节点的 HMI 应用程序的最新的版本将被加载，这还将与 HMI 应用程序关联的 $ApplicationChanged 标记设置为假。如果 HMI 应用程序查看器关闭并重新打开，而没有使用 RestartWindowViewer() 函数，那么，以前使用的应用程序将重新加载，较新的应用程序将不会加载（$ApplicationChanged 系统标记将仍为真）。RestartWindowViewer() 脚本函数对于这里所描述的所有改变模式，将如这里所描述的那样起作用。

[0132] 2.ReloadWmdowViewer()- 使与封装的 HMI 应用程序关联的查看器加载已经部署到节点的应用程序的最新的版本。ReloadWindowView() 函数与 RestartWindowViewer() 函数的不同之处在于，如果应用程序改变是若不完全重新启动就不能加载的那种，则它只重新启动 HMI 应用程序查看器。ReloadWindowViewer() 脚本函数对于这里所描述的所有改变模式，将如这里所描述的那样起作用。

[0133] 利用“重新启动查看器”改变模式，每当将新的 HMI 应用程序版本部署到目标节点时，使 HMI 应用程序查看器自动地重新启动。在重新启动时，部署到节点的最新的 HMI 应用程序版本将会加载到 HMI 应用程序查看器中。

[0134] 利用“提示用户重新启动查看器”改变模式，当新的 HMI 应用程序部署到目标节点时，会提示用户指出用户是否希望重新启动 HMI 应用程序查看器，并接受新的 HMI 应用程序。如果用户选择不重新启动 HMI 应用程序查看器，则在提醒周期到期之后再发出提醒。在重新启动时，最新部署到该节点的 IIMI 应用程序将加载到 WindowViewer 中。

[0135] 利用“将改变加载到查看器”改变模式，使 HMI 应用程序查看器将最新部署 HMI 应用程序加载到节点中，无需重新启动查看器。关联的配置设置确定如何处理需要重新启动的改变：

[0136] 1.“提示用户重新启动”- 使查看器提示用户是否重新启动查看器以接受新的应用程序。如果用户选择不重新启动查看器，则在提醒间隔过期之后，再次提醒他。在重新启动时，最新部署到该节点的 HMI 应用程序将加载到查看器中。

[0137] 2.“自动地重新启动”- 使查看器自动地重新启动，以应用改变。如果查看器不需要重新启动即可应用改变，那么，新的应用程序将被加载，对运行中的进程没有中断。在重新启动查看器时，最新部署到该节点的 HMI 应用程序将加载到 WindowViewer 中。

[0138] “提示用户将改变加载到查看器中”改变模式导致 HMI 应用程序查看器通知用户，有 HMI 应用程序的新的版本可用。如果用户选择不接受改变的 HMI 应用程序，则在提醒间隔过期之后，再次提醒他。如果 HMI 应用程序查看器需要重新启动，以应用改变，那么，会将此情况通知给操作员，当接受时，查看器将会自动地重新启动。如果不需要重新启动以可应用改变，那么，当用户接受新的应用程序时，它将被加载，对运行中的进程没有中断。

[0139] 现在将讲述部署封装上文讨论的 HMI 应用程序的 HMI 应用程序对象实例的过程。对于在引擎上执行的所有对象，部署 HMI 对象实例的过程使用由系统支持的标准部署机制。HMI 应用程序对象的所有实例数据与 HMI 应用程序对象的实例一起同步地部署。如果 HMI 对象实例已经部署，在该对象上调用了“部署改变”操作，那么，将进行检查，以判断 HMI 容器对象本身是否具有需要它被部署的任何改变。如果它没有任何这样的改变（即，所有改变都在封装的 HMI 应用程序内），那么，HMI 对象将不会被撤销部署，然后，再部署，只提供改变的 HMI 应用程序。

[0140] 在示范性实施例中，通过 IDE 126 撤销部署 HMI 应用程序对象实例或其他合适的配置实用程序，会删除 HMI 包裹对象以及关联的 / 封装的 HMI 应用程序。如果 HMI 应用程序当前正在被正在运行的 HMI 应用程序（例如，INTOUCH）查看器使用，那么，撤销部署操作失败（在节点上没有执行）。一旦撤销部署操作成功地完成，HMI 应用程序将不能运行。实际文件被保留，直到平台被撤销部署（由于部署设计），但是，HMI 应用程序被阻止运行。

[0141] 请参看图 7，示意图描述了 HMI 系统 700，它支持用于实现制造自动化的第一（例如，上一代）HMI 技术 702 和第二（例如，下一代）HMI 技术 704。根据本发明的实施例，这样的系统支持各种迁移方案，包括：在两个技术空间之间导入 / 导出 HMI 应用程序，从第一 HMI 技术 702 和第二 HMI 技术 704 运行包括图形的应用程序。下面将详细地讨论包括 / 接纳第一和第二 HMI 技术的示范性迁移方案的功能。

[0142] 上面的描述触及了以唯一方式支持两个不同的 HMI 应用程序管理和运行时环境的系统的许多方面。系统在两个共存的 HMI 技术之间提供了有效的迁移路径，从用户的观点来看，几乎是无缝的。下面将描述包括多个方面的 HMI 技术的桥接技术 / 方案，该方案允许根据以前的 HMI 技术开发的过程自动化 HMI 应用程序在支持新的和老的技术的环境内被

访问、编辑和增强，并执行－甚至在同一 HMI 应用程序视图内，在特定示范性实施例中，集成的方面包括：命名空间、图形呈现 (rendering)，以及两种技术的运行时集成。

[0143] 在示范性实施例中，两个技术包括 INTOUCH，这是基于标记的 HMI 技术，以及 ARCHESTRA，这是基于对象的技术。总之，如下面所描述的，这两种技术之间的迁移路径包括：将 INTOUCH 应用程序封装在 HMI 应用程序对象模板包裹内，以有助于 INTOUCH 应用程序在 ARCHESTRA 基础架构内的管理，修改 INTOUCH 标记，以便可以在包括全局名称约定的环境内使用，以及，增强 INTOUCH 窗口编辑器和显示软件的功能，以支持 ARCHESTRA 图形使用的增强的图形功能。

[0144] 在特定说明性实施例中，INTOUCH 将其窗口配置数据存储在 .win 文件中。为了托管新技术符号，INTOUCH 中的 .win 文件格式被扩展，以支持 ARCHESTRA（下一代）符号定义。当最初为 INTOUCH 应用程序开发的编辑器 / 查看器遇到 ARCHESTRA 输入时，编辑器 / 查看器经过抽象层访问 ARCHESTRA 系统。抽象层通过将 INTOUCH “设备上下文”传递到基于 ARCHESTRA HMI 技术的符号，来管理这两种技术之间的交互，符号再将其本身再现到设备上下文中。新的 HMI 技术使用 GDI+ 的先进的图形功能，GDI+ 能够处理本地设备上下文。包括在上一代编辑器 / 查看器中的桥接功能在新的符号技术中使用带有 GDI+rending 代码的旧软件所创建的现有的设备上下文。

[0145] 如此，部分地，通过在现有技术内嵌入新技术，以在以前存在的系统内支持新技术的新的功能，支持迁移路径。所有数据连接、变量定义，以及工厂数据都被保留，并被无缝地映射到较新的功能，在 INTOUCH/ARCHESTRA 迁移的情况下，以前的技术基于 C/C-H-/MFC 代码实现方式，而新技术使用 .NET 技术。在示范性实施例中，通过修改当前利用 MFC/C/C++ 的 HMI 应用程序编辑器 / 查看器，支持 .NET 集成。

[0146] 请参看图 8，迁移方案包括将根据上一代技术定义的 HMI 应用程序 802 导入到新的 HMI 技术的功能（通过将上一代技术封装在基于新的 HMI 技术的应用程序模板 802′ 内）。此后，所产生的 HMI 应用程序模板 802′ 保留在与新的 HMI 技术关联的配置数据库 124 中。与 HMI 应用程序 802 关联的图形定义也导入到图形工具箱 804 中。

[0147] 在示范性实施例中，在用户通过，例如，IDE 126，选择“导入现有应用程序”操作之后，用户将具有下面的选择：手动键入现有应用程序的路径，浏览到现有应用程序的路径，并搜索包括上一代 HMI 应用程序 802 的现有的应用程序。如果用户选择搜索应用程序 802，那么，提示用户输入或浏览到该搜索的起始目录。一旦执行了搜索，则搜索实用程序会呈现满足搜索项的应用程序的列表，对于满足搜索参数的每一个查找到的 HMI 应用程序，将呈现下面的信息：应用程序名称、应用程序描述，以及应用程序路径。

[0148] 值得注意的是，如前所述的 HMI 导入操作不是典型的导入操作。执行的以使 HMI 应用程序 802 采用新技术的导入操作以及其关联的管理和运行时环境是复制和关联操作（通过上文所描述的符合新技术的包裹对象 -HMI 应用程序模板中的封装）。当用户导入应用程序 802 时，导入操作将指定的应用程序 802 的全部内容（包含子目录）复制到配置数据库 124 中。通过封装 HMI 应用程序 802 以再现 HMI 应用程序模板 802′，可以在与新的 HMI 技术关联的 IDE 126 中全局地管理 HMI 应用程序 802′。一旦执行了此关联，HMI 应用程序模板 802′ 和 HMI 应用程序 802 之间的结合不能被中断。用户改变封装的 HMI 应用程序 802 的唯一选择是删除封装的 HMI 应用程序 802′ 以及所有的部署的实例，然后，重新创建 HMI

应用程序模板。

[0149] 在特定实施例中，HMI 应用程序 802 是 INTOUCH 应用程序，下一代 HMI 技术是 ARCHESTRA，包括基于对象的配置数据库 124 和 HMI 应用程序的管理结构。在说明性示例中，在配置数据库 124 内，INTOUCH 应用程序被表示为叫做 InTouchViewApp 的特定类型的模板。InTouchViewApp 模板（例如，802′）包含对原始 INTOUCH 应用程序（例如，HMI 应用程序 802）的引用，以及有关 INTOUCH 应用程序的行为的某些特定信息，但是，它没有在配置数据存储区内存储 INTOUCH 应用程序的数据。INTOUCH 应用程序数据以由 INTOUCH 所定义的标准格式保留在与模板关联的文件储存库目录中。

[0150] 在示范性实施例中，某些上一代 HMI 应用程序不被允许导入到新的 HMI 技术中。例如，可以作为 HMI 模板导入到配置数据库中的上一代 HMI 应用程序将被限制到尚未导出到 HMI 应用程序模板中的那些以及以前已经从 HMI 应用程序模板导出的那些。下面的应用程序类型将不被允许导入到 HMI 应用程序模板中：目前与 HMI 应用程序模板关联的应用程序，已经从 HMI 应用程序模板部署的应用程序，已经从 HMI 应用程序模板导出的应用程序。

[0151] 本系统也支持反向操作，从存储在配置数据库 124 中的 HMI 模板将 HMI 应用程序导出到上一代 HMI 技术。只希望向 HMI 应用程序 802 的独立版本添加新的 HMI 技术图形的用户可以这么做，但是，不能够利用部署功能（需要 ARCHESTRA 基础架构）。为了将 HMI 应用程序 802″ 移动到目标机器，用户调用当管理 HMI 应用程序模板 802′ 时可用的导出功能。当调用导出操作时，提示用户输入目标目录路径。一旦用户执行了此操作并确认了该操作，整个封装的 HMI 应用程序都被置于提供的路径中，包括：所有 HMI 应用程序窗口、标记名称词典、前代符号、前代局部化数据，以及嵌入的新技术图形。使用被添加的组件增强的以前的 HMI 技术的查看器实用程序，处理新技术图形，以便接纳新的图形技术，以及嵌入的新技术图形数据。如前所述的导入 / 导出序列允许用户包括新技术图形，无需完全迁移到新的 HMI 技术的平台。

[0152] 现在可以在通过添加的组件增强的编辑器中打开导出的 HMI 应用程序 802″，以允许在断开连接的 / 独立的 HMI 应用程序 802″ 中在现场预先形成编辑。编辑器允许对以前的和新的 HMI 技术图形进行编辑。对新技术图形的编辑的程度是由添加的组件所提供的增强功能确定的，包括，例如：调整大小、删除、配置动画、移动、复制，以及剪贴板操作（剪切、复制，以及粘贴）。

[0153] 在将 HMI 应用程序 802 封装在 HMI 包裹内以在 IDE 126 中再现 HMI 应用程序模板 802′ 然后嵌入新的 HMI 技术图形之后，模板 802′ 保留导入和导出窗口的能力。当导入 HMI 应用程序 802 时，包含在导入的窗口（而不是存储在配置数据库 124 中的）的所有以前的技术标记引用都具有置于引用的前面的占位符，以防止拆散以前的技术（例如，InTouch）标记引用。

[0154] 系统也支持将（下一代）HMI 应用程序模板导入到配置数据库 124 中。在导入操作过程中，获取封装的 HMI 应用程序，并存储在文件储存库中。导入新技术特定的文件和数据，并置于配置数据库 124 中。

[0155] 已经描述了示范性迁移方案的导入 / 导出功能，现在将讲述相对于潜在地包括根据新的和以前的 HMI 技术中的任何一种技术定义的图形的 HMI 应用程序的编辑器 / 查看器行为。在示范性实施例中，当在窗口编辑器中工作时，用户能够启动新的 HMI 技术符号编辑

器，以对嵌入的新的 HMI 技术图形符号进行编辑。用户还可以从窗口集合中对于引用的新技术 HMI 窗口，启动新技术编辑器。下面的讨论涉及 HMI 图形窗口查看器 / 编辑器工具，如图 9 中所描述的。

[0156] 如图 9 所描述的，增强了 HMI 应用程序编辑器 / 查看器，以支持根据以前的 HMI 技术和下一代 HMI 技术定义的图形。在特定说明性实施例中，INTOUCH 将其窗口配置数据存储在 .win 文件中。为了托管新技术图形符号，INTOUCH 中的 .win 文件格式被扩展，以支持 ARCHESTRA（下一代）符号定义。当最初为 INTOUCH 应用程序开发的编辑器 / 查看器遇到 ARCHESTRA 输入时，编辑器 / 查看器经过抽象层访问 ARCHESTRA 系统。抽象层通过将 INTOUCH “设备上下文”传递到基于 ARCHESTRA HMI 技术的符号，来管理这两种技术之间的交互，符号再将其本身再现到设备上下文中。新的 HMI 技术使用 GDI+ 的先进的图形功能，GDI+ 能够处理本地设备上下文。包括在上一代编辑器 / 查看器中的桥接功能在新的符号技术中使用带有 GDI+ 再现代码的旧软件所创建的现有的设备上下文。图 9 中描述了上文所描述的集成的结果，其中，在同一视图上产生旧的技术图形，如发光符号 900 和模拟式仪表 902，以及新的 HMI 技术图形，如模拟式仪表 904 和泵 906（包括与新技术关联的梯度图形）。

[0157] 在示范性实施例中，通过 IDE 126 启动查看器 / 编辑器实用程序。对于每一个嵌入的新技术图形，用户可以选择启动符号编辑器或者实例编辑器。基于符号的源，以不同的方式处理这些选项。下面描述了每一个选项。

[0158] 下面描述了当用户选择从编辑器来编辑符号时编辑器工具的行为。如果图形工具箱 804 符号没有签出到当前用户，则在启动编辑器之前在配置数据库 124 中执行隐式签出。

[0159] 下面描述了为在配置数据库 124 中的应用程序级别的对象（例如，应用程序对象或设备集成对象）定义的图形启动编辑器的过程。如果没有从数据库 124 中签出关联的对象模板，则应该在通过 IDE126 启动编辑器之前将它签出。使用 IDE 126，为基于用户选择而选择的模板或实例，启动对象编辑器。一旦打开了对象编辑器，则启动相关图形的图形编辑器，并使其进入焦点中。如果用户选择对实例进行编辑的选项，那么，关联的实例的对象编辑器将在父 IDE 实例中打开。

[0160] 下面描述了当用户从视图编辑器选择了对应用程序级别的对象的符号进行编辑时系统的行为。通过为直接在实例内定义的图形启动新技术编辑器，系统为关联的对象实例启动编辑器。一旦打开了对象编辑器，则启动相关图形的图形编辑器，并使其进入焦点中。如果没有签出关联的实例，则应该在启动编辑器之前将它签出。

[0161] 下面的讨论呈现了与将下一代功能包括到上一代 HMI 技术编辑器 / 查看器中关联的定义了用户功能的各种特点。

[0162] 在应用程序级别对象实例或图形工具箱 804 中定义的下一代符号能够包括到基于以前的 HMI 技术的 HMI 应用程序中。在说明性实施例中，以前的 HMI 技术编辑器的增强版本（例如，WONDERWARE′ s WindowMaker）使用在从 IDE 启动编辑器的过程中由 IDE 126 提供的连接信息，确定与下一代对象管理（配置数据库 124）和通信基础结构 (Message Exchange) 的连接。使用该连接来浏览配置数据库的内容，访问文件储存库，以修改上一代 HMI 应用程序文件，签入等等。

[0163] 包括到所说明的系统的有用的特点是支持将在下一代 HMI 技术中定义的符号（对象实例）嵌入到上一代 HMI 图形显示窗口中的能力。在示范性实施例中，从配置数据库 124

中的应用程序级别的对象（例如，应用程序对象）实例或图形工具箱 804 获取嵌入的符号。当符号被嵌入到上一代 HMI 应用程序窗口时，只有对符号的引用才持续在窗口中，而不是符号的实际定义。当加载窗口时，从配置数据库 124 中检索符号图形定义。加载的符号的版本其他用户的最后签入的版本或当前用户的最后保存的版本。HMI 应用程序模板在其引用列表 400 中维护所有嵌入的符号的列表。

[0164] 如上文所提及的，增强上一代 HMI 应用程序编辑器，对支持对与下一代 HMI 技术关联的配置数据库 124 中定义的对象的访问。在示范性实施例中，编辑器包括了由下一代 HMI 技术所提供的浏览器工具，以浏览配置数据库 124，以查找用于视图中的图形。在示范性实施例中，响应当前编辑的窗口上的“嵌入符号”，启动浏览器工具。

[0165] 类似地，增强的上一代 HMI 应用程序窗口编辑器支持在窗口内嵌入与应用程序级别的对象模板关联的符号。响应用户选择对象模板（而不是上文所描述的实例），系统隐式地创建选定的应用程序级别的对象的实例，然后，从新创建的应用程序级别的对象实例嵌入选定的符号。

[0166] 当用户从配置数据库中的对象模板选择符号时，与编辑器关联的自动创建功能会创建与符号关联的对象实例。与创建新对象实例相关联，提示用户，以提供对象实例的名称。此外，在这个意义上，模板包括嵌入的模板，还创建嵌入的模板的实例。

[0167] 一旦下一代 HMI 技术符号通过增强的 HMI 应用程序编辑器被嵌入到窗口，在符号上支持各种编辑操作，包括：替换符号文本串（替代字符串）、替换符号数据引用（替代标记）、替换符号图形属性、应用动画、调整大小、移动、删除、剪切、复制、粘贴、复制、对准、分发、制作单元（作为单元的一部分添加）等等。

[0168] 被替换的符号数据引用出现在由增强的上一代编辑器生成的交叉引用报表中。IDE 交叉引用功能覆盖了没有被替换的（并包含在配置数据库 124 中）引用。供符号使用的被替换的下一代符号数据引用将出现在由窗口和由标记名称（引用）中。

[0169] 对包括到由增强的上一代编辑器编辑的窗口的符号潜在地执行的另一个功能是能够改变符号与另一个对象的关联。嵌入的符号的关联的应用程序级别的对象可以更改为包含相同符号的另一个对象实例。此功能用于，例如，复制 / 粘贴嵌入的符号之后。配置数据库 124 浏览器用于浏览数据库 124，以便选择另一个同辈实例。浏览器被过滤，以只显示包含相同符号的实例。在第一示例中，符号的关联被更改为另一个现有的对象实例。在第二示例中，符号的关联链接到新对象实例。在该情况下，当改变对象关联时，编辑器自动创建新的同辈实例。在自动创建过程中，为给定实例创建新的自动化对象实例。将从指定的对象实例的父模板派生出新的实例，提示用户输入新的实例的名称。此外，如果给定模板具有任何包含的模板，那么，为所有包含的模板，创建实例。所有包含的模板都使用由系统生成的默认名称（通过 IDE 126 指定）。

[0170] 对增强的 HMI 应用程序编辑器编辑的窗口内的下一代符号的另一个支持功能被更改应用程序级别（例如，应用程序）对象的替代符号。配置数据库 124 浏览器的输出被过滤，只显示关联的自动化对象的符号。改变关联的符号的操作可适用于应用程序级别的对象的嵌入的符号。新选择的替代符号可以与当前嵌入的符号是不同大小。向用户提供了一个选项，要么使用新嵌入的符号的默认大小，要么保留窗口的大小。在更改为新的符号的过程中，以前对以前嵌入的符号执行的配置信息被保留。保留的配置信息包括，例如：符号

位置、动画链接，以及图形属性 - 以前的和当前符号之间的属性映射，被替换的文本串（用于匹配属性），被替换的数据引用（用于匹配属性）。

[0171] 由增强的以前的技术编辑器支持的对嵌入的下一代 HMI 技术符号的再一个功能是配置动画链接。可以对嵌入的下一代符号配置下面的示范性动画行为：位置、大小、禁用、闪烁、朝向，以及可见性。这里配置的动画整个地适用于嵌入的符号，配置的动画存储在 containing 窗口的配置定义内。

[0172] 由增强的编辑器支持的再一个功能是配置嵌入的符号的图形属性。当在上一代 HMI 窗口内嵌入包含图形属性的下一代符号时，编辑器支持与符号关联的图形属性的配置。编辑器提供了用户界面，该用户界面列举并允许替换嵌入符号内定义的图形属性的值。其配置被支持的图形属性包括，例如，（配置数据库 124 中管理的对象的）本地标记、远程标记，以及全局对象属性。

[0173] 下面将描述独立 INTOUCH 应用程序迁移到 ARCHESTRAHMI 配置和运行时环境的具体示例的功能。

[0174] INTOUCH 到 ARCHESTRA 功能映射：

[0175] INTOUCH 到 ARCHESTRA 迁移的目标是实现原始 INTOUCH 图形和在迁移过程中创建的新的 ARCHESTRA 图形之间的 100％功能等效。在 INTOUCH 和 ARCHESTRA 之间创建和维护图形的方式不同，但是，一旦迁移，表现方式相同。迁移不会在 ARCHESTRA 内复制 INTOUCH 用户体验。

[0176] 窗口和智能符号：

[0177] INTOUCH 具有两个较大的图形类型：窗口和智能符号。ARCHESTRA 支持这两者的迁移。对于 INTOUCH 窗口，INTOUCH 窗口是作为带有 .Win 扩展名的单个文件维护的。格式是由 INTOUCH 所定义的专有的二进制格式。ARCHESTRA 将读取本地窗口格式，并生成 ARCHESTRA 等效格式。一旦迁移进行完毕，本地 INTOUCH 格式将不能在 ARCHESTRA 内使用。对于 SmartSymbol，INTOUCH 智能符号是作为带有 .xml 扩展名的单个文件来维护的。格式是由 INTOUCH 定义的。ARCHESTRA 将读取本地智能符号格式，并生成 ARCHESTRA 等效格式。一旦迁移进行完毕，本地 INTOUCH 格式将不能在 ARCHESTRA 内使用。

[0178] 窗口和智能符号图形内容：

[0179] INTOUCH 内的窗口和智能符号由一个或多个图形元件构成。各种元件将按如下方式进行转换。

[0180] 图形基本要素：100％的基本图形基本要素将被迁移。没有功能被省略或改变。

[0181] 图形动画：100％的基本图形动画将被迁移。没有功能被省略或改变。

[0182] 脚本动画：在 INTOUCH 中可用的 100％的脚本事件在 ARCHESTRA 图形中都可用。置于 INTOUCH 中的那些事件中的任何脚本都被复制到 ARCHESTRA 图形中的对应的事件。并非所有的 INTOUCH 脚本功能都在 ARCHESTRA 图形中可用，并可能导致无效的脚本被迁移。包含带有无效内容或句法的脚本的 ARCHESTRA 图形，在用户编辑那些 ARCHESTRA 图形时，用户可以轻松地识别，通过查看图形命名空间树中的元件，对于单个 ARCHESTRA 图形，用户无需打开动画链接编辑器。那些带有配置错误的项目将清楚地通过图标重叠来标识。

[0183] 表达式句法：ARCHESTRA 图形和 INTOUCH 图形的表达式句法相同，无需任何修改，直接迁移。脚本功能可以用于表达式中。并非所有的 INTOUCH 脚本功能都在 ARCHESTRA 图

形中可用，并可能导致无效的表达式被迁移。包含带有无效内容的表达式的 ARCHESTRA 图形，在用户编辑那些 ARCHESTRA 图形时，用户可以轻松地识别，通过查看图形命名空间树中的元件，对于单个 ARCHESTRA 图形，用户无需打开动画链接编辑器。那些带有配置错误的项目将清楚地通过图标重叠来标识。

[0184] 所说明的迁移方案的另一个方面涉及处理由上一代和下一代 HMI 技术使用的不同引用命名方案。在示范性实施例中，从上一代和下一代 HMI 技术支持所有引用类型。上一代 HMI 图形定义中的所有引用通过客户抽象层 (CAL) 在运行时进行处理。上一代引用是，例如，基于本地标记的。另一方面，下一代引用，是全局唯一对象属性，可以用来表示位于如配置数据库 124 所定义的系统中的任何地方的数据。在示范性实施例中，下一代命名方案用于向支持系统内的对象之间的通信的 Message Exchange(Mx) 功能标识数据。如此，在示范性实施例中，当上一代 HMI 图形被迁移到下一代图形技术时，引用句法也将被迁移，以匹配由 ARCHESTRA 图形预期的引用句法。下面将参考图 10a、10b，以及 11 讨论命名空间处理。

[0185] 关于以前的技术（例如，INTOUCH）引用的迁移，在示范性实施例中，所有对本地（例如，INTOUCH）标记名称词典的引用和没有引用全局配置数据库（例如，星系）访问名的远程引用都被视为 CAL 引用。当构建需要通过 CAL 传递到数据的下一代 HMI 技术图形时，在通知方案中，如果支持的话，用户通过将引用标识为 CAL 引用的关键字（例如，“InTouch”），给引用添加前缀。当 CAL 引用被迁移时，在迁移过程中，所需的关键字自动地添加到现有引用的前面。

[0186] CAL 引用的迁移示例：

[0187] 迁移之前的引用 迁移之后的引用

[0188] IntouchTagame.DotField Intouch:IntouchTagame.DotField

[0189] RemoteReferenceAccessName:Item Intouch:RemoteReferenceAccessName:Item

[0190] 关于新技术（例如，ARCHESTRA）引用的迁移，利用下一代命名方案来引用全局唯一地引用的数据的所有上一代 HMI 图形引用，都使用指向全局配置数据库（例如，Galaxy）访问名的远程引用。这样的图形不使用关键字（例如，Intouch）来访问全局可访问的数据（通过 Message Exchange）。如此，在特定示例中，当 INTOUCH 图形被迁移到 ARCHESTRA 时，使用“Galaxy”访问名的所有远程引用（根据需要，在以前的技术内）都具有从引用中剥离的“Galaxy”访问名，并将“InTouch”CAL 调用关键字加在引用的前头。远程引用的迁移示例：

[0191] 迁移之前的引用 迁移之后的引用

[0192] Galaxy:ObjectTagname.Property ObjectTagname.Property

[0193] Galaxy:ObjectHierarchicalName.Property ObjectHierarchicalName.Property

[0194] Galaxy:ObjectTagname.Property.#VString ObjectTagname.Property

[0195] 图 11 提供了一组图形示例，以及迁移的 HMI 应用程序的关联的标记，包括：InTouch 本地标记“Light1”、InTouch 引用“InTouch:$second”、InTouch 远程引用“Galaxy:tanklevel.pv”，以及全局唯一（在配置数据库 124 内）ARCHESTRA 引用“Tank001.pv”和“Pump001.speed”。

[0196] 如此，总之，上文所描述的系统提供了唯一的，基于共存的涉及第一和第二 HMI 技术的迁移路径，包括：集成的应用程序管理、集成的图形管理，以及集成的命名空间。此外，导入 / 导出功能还支持两个技术环境内的 HMI 应用程序的配置 / 执行。

[0197] 鉴于可以应用此说明的系统的原理的许多可能的实施例，应该认识到，这里参考图形所描述的实施例只是说明性的，不应该理解为对本发明的范围作出限制。例如，那些熟悉技术的人员将认识到，以软件的形式存在的所显示的实施例，以计算机可执行的指令的形式存在的存储在计算机可读取的介质上的某些元件，也可以以硬件实现，反之亦然，在不偏离本发明的精神的情况下，可以在布局和细节方面对所显示的实施例进行修改。因此，这里所描述的本发明预期，所有这样的实施例都可以归入下面的权利要求以及其等效内容的范围之内。

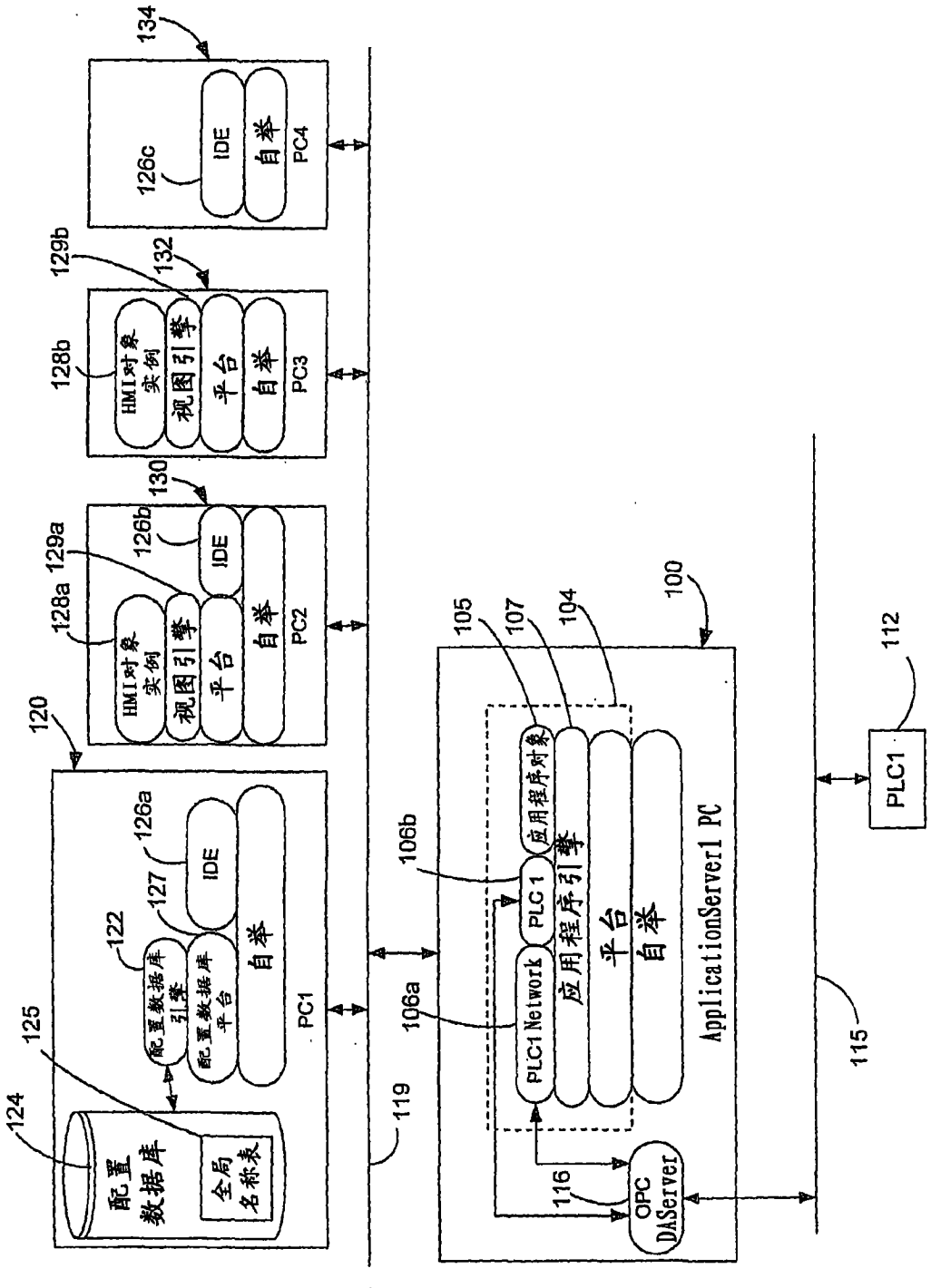

134
126c
IDE
自举
PC4
129b
132
128b
HMI对象实例
视图引擎
平台
自举
PC3
130
126b
IDE
129a
128a
HMI对象实例
视图引擎
平台
自举
PC2
120
126a
IDE
127
122
配置数据库引擎
配置数据库平台
自举
PC1
125
124
配置数据库
全局名称表
119
106b
106a
105
107
104
100
应用程序对象
PLC1
应用程序引擎
平台
自举
PLC1 Network
ApplicationServer1 PC
116
OPC DAServer
115
112
PLC1

图 1

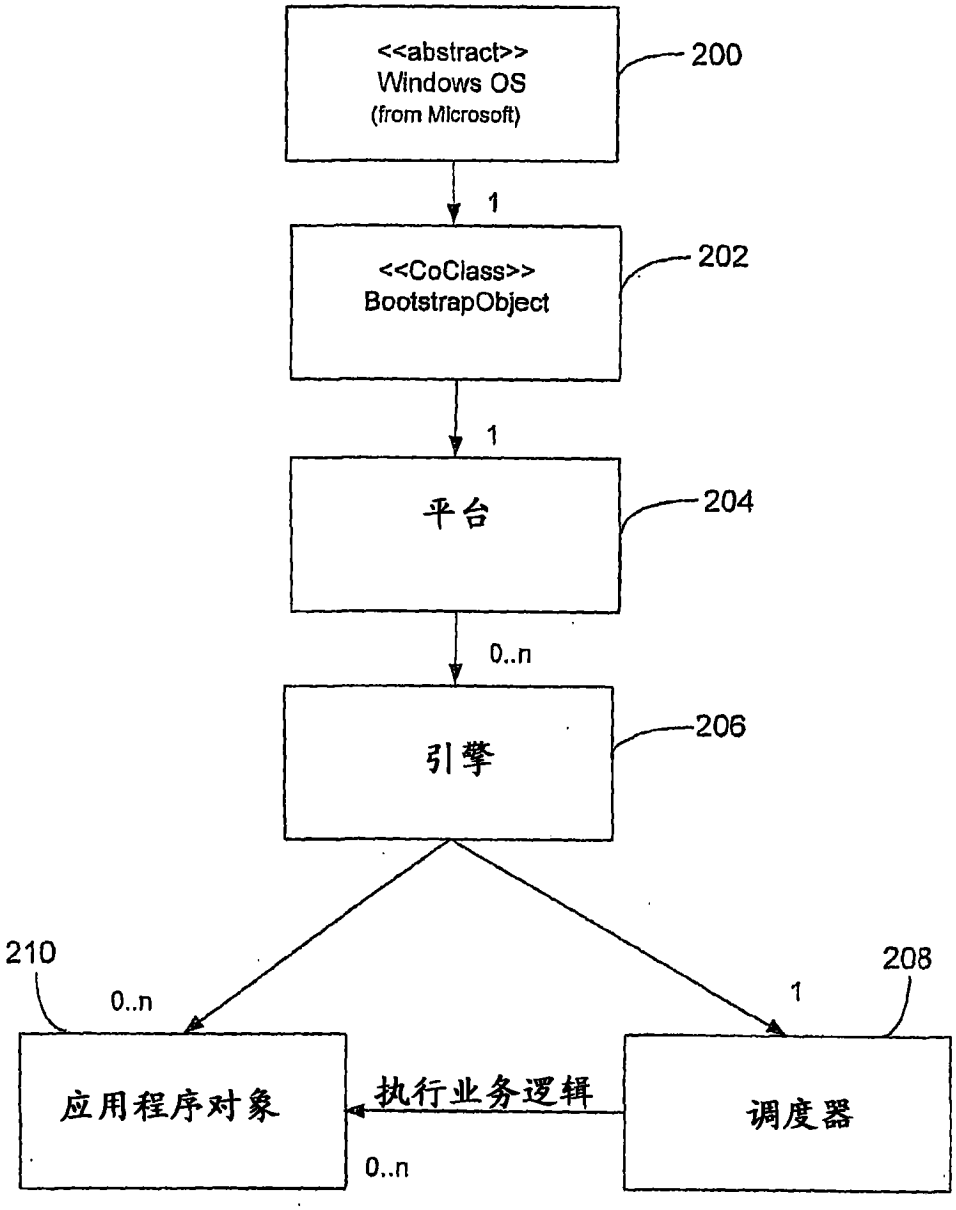

<<abstract>>
Windows OS
(from Microsoft)
200
1
<<CoClass>>
BootstrapObject
202
1
平台
204
0..n
引擎
206
210
0..n
208
1
应用程序对象
执行业务逻辑
调度器
0..n

图 2

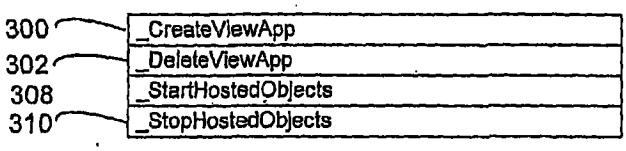

300
_CreateViewApp
302
_DeleteViewApp
308
_StartHostedObjects
310
_StopHostedObjects

图 3

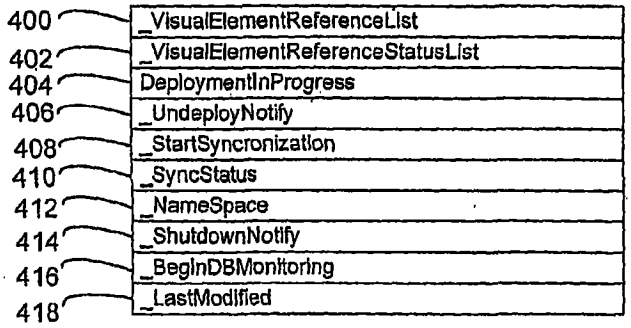

400
_VisualElementReferenceList
402
_VisualElementReferenceStatusList
404
DeploymentInProgress
406
_UndeployNotify
408
_StartSyncronization
410
_SyncStatus
412
_NameSpace
414
_ShutdownNotify
416
_BeginDBMonitoring
418
_LastModified

图 4

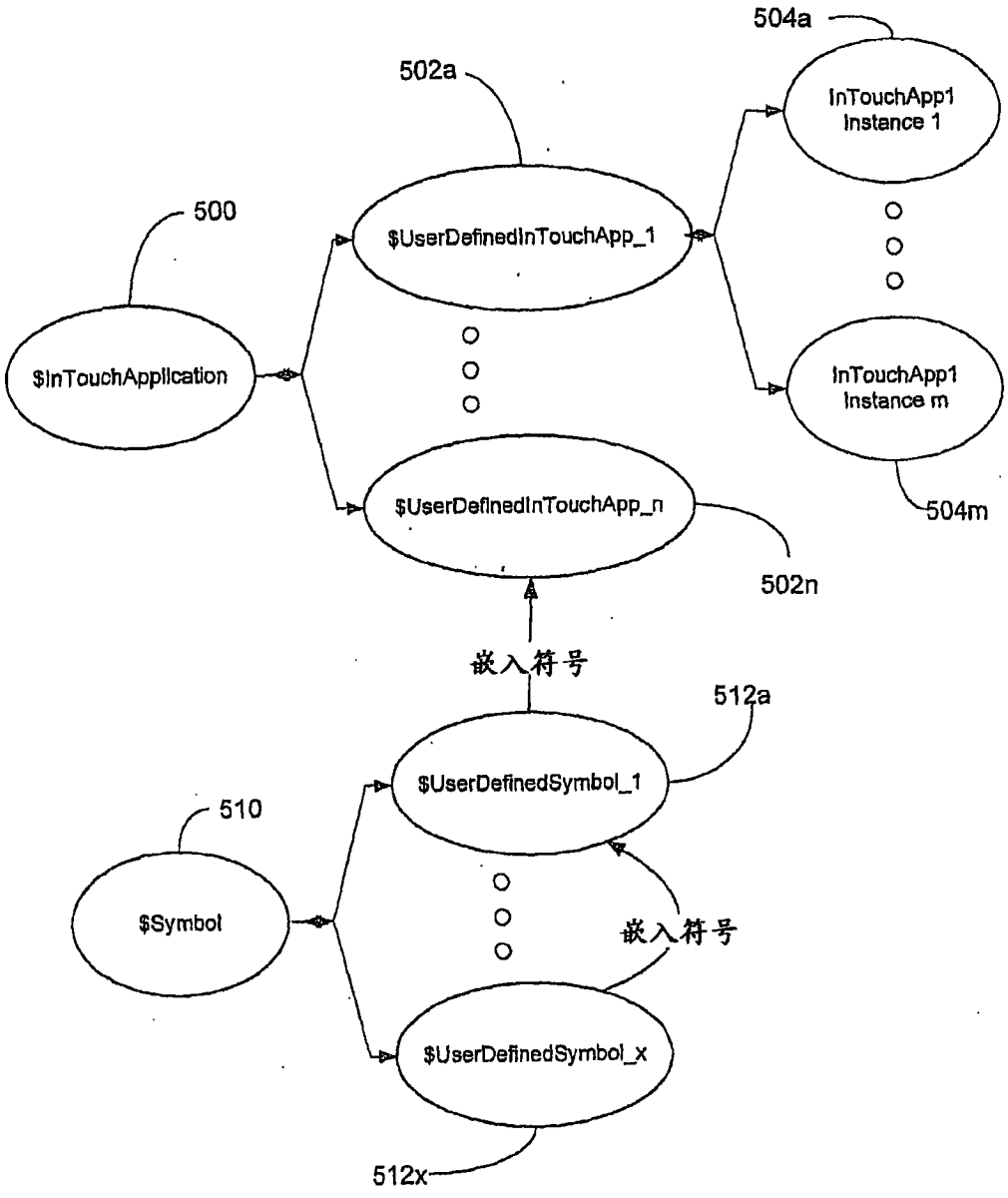

504a
502a
InTouchApp1
Instance 1
500
$UserDefinedInTouchApp_1
$InTouchApplication
InTouchApp1
Instance m
504m
$UserDefinedInTouchApp_n
502n
嵌入符号
512a
$UserDefinedSymbol_1
510
$Symbol
嵌入符号
$UserDefinedSymbol_x
512x

图 5a

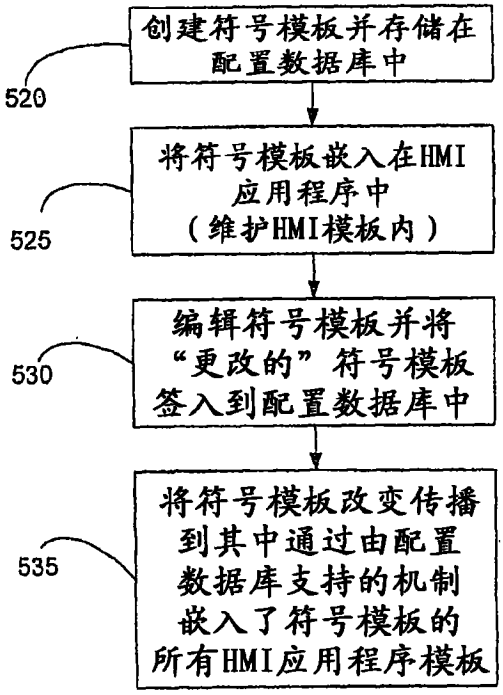

创建符号模板并存储在配置数据库中
520
将符号模板嵌入在HMI应用程序中（维护HMI模板内）
525
编辑符号模板并将“更改的”符号模板签入到配置数据库中
530
将符号模板改变传播到其中通过由配置数据库支持的机制嵌入了符号模板的所有HMI应用程序模板
535

图 5b

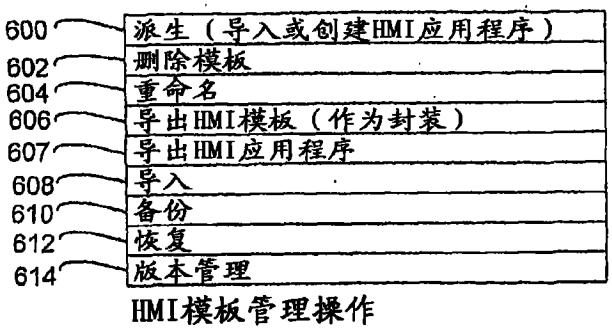

600 派生（导入或创建HMI应用程序）
602 删除模板
604 重命名
606 导出HMI模板（作为封装）
607 导出HMI应用程序
608 导入
610 备份
612 恢复
614 版本管理
HMI模板管理操作

图 6

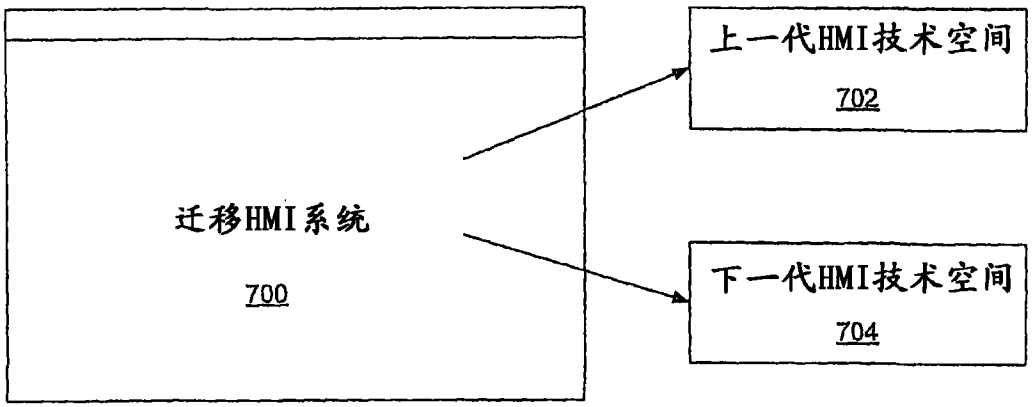

迁移HMI系统
700
上一代HMI技术空间
702
下一代HMI技术空间
704

图 7

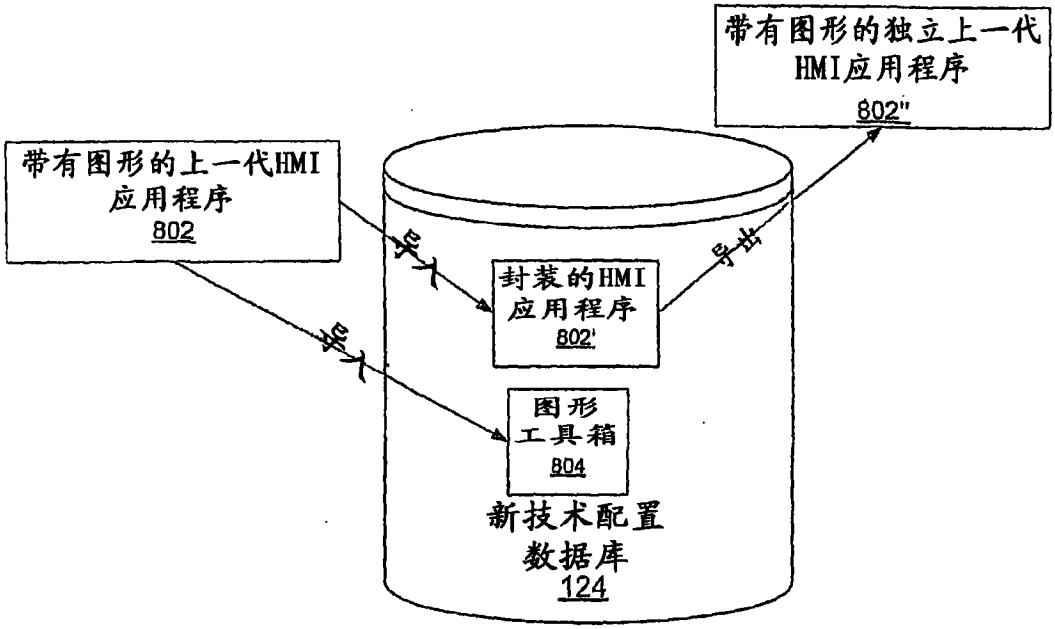

带有图形的独立上一代
HMI应用程序
802"
带有图形的上一代HMI
应用程序
802
导入
导出
封装的HMI
应用程序
802'
图形
工具箱
804
新技术配置
数据库
124

图 8

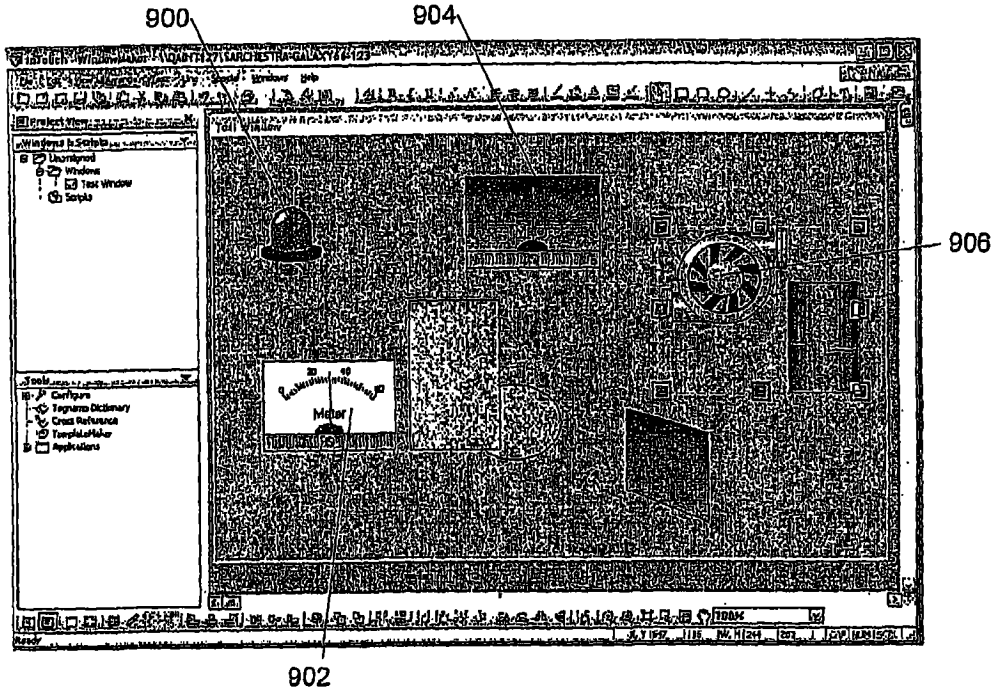

900
904
906
902

图 9

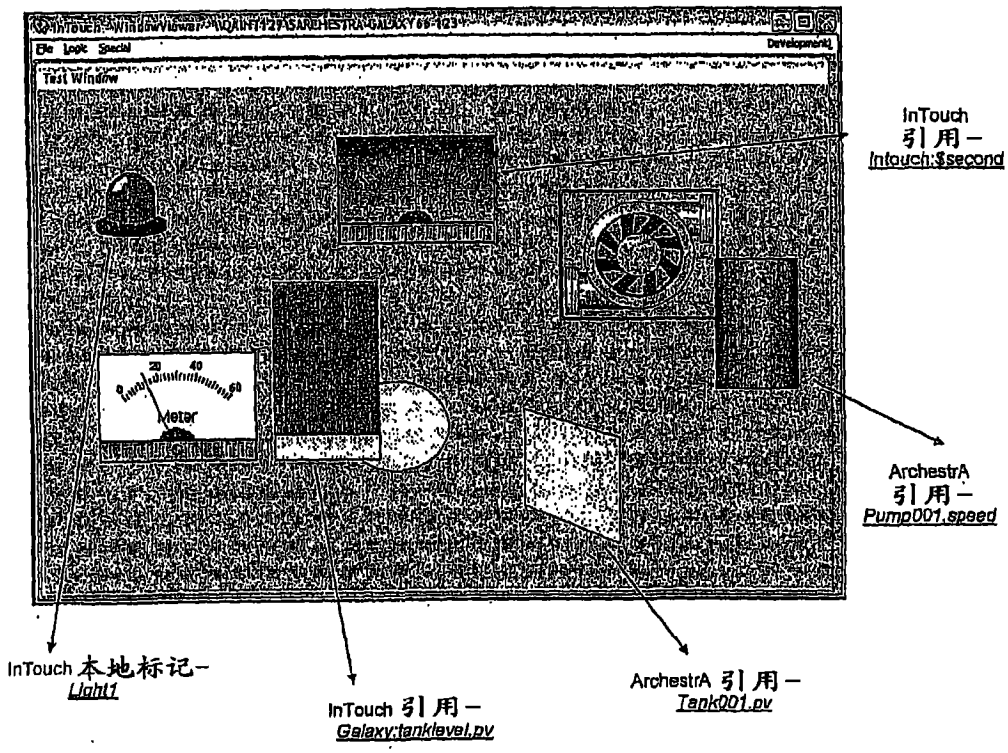

InTouch
引用－
Intouch:$second
ArchestrA
引用－
Pump001.speed
InTouch 本地标记－
Light1
InTouch 引用－
Galaxy:tanklevel.pv
ArchestrA 引用－
Tank001.pv

图 11

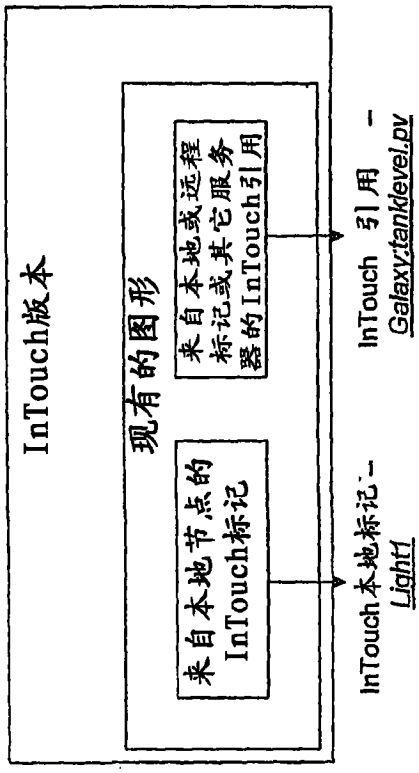

InTouch版本
现有的图形
来自本地节点的InTouch标记
来自本地或远程标记或其它服务器的InTouch引用
InTouch本地标记 – Light1
InTouch 引用 – Galaxy:tankdevel.pv

图 10a

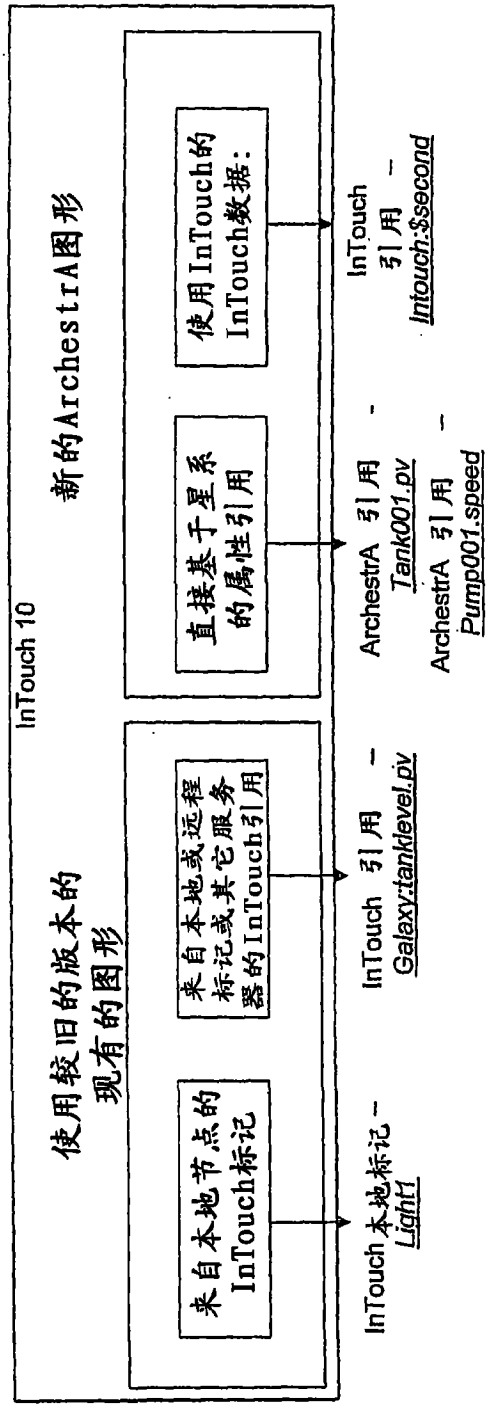

InTouch 10
使用较旧的版本的现有的图形
来自本地节点的InTouch标记
来自本地或远程标记或其它服务器的InTouch引用
新的ArchestrA图形
直接基于星系的属性引用
使用InTouch的InTouch数据:
InTouch本地标记 – Light1
InTouch 引用 – Galaxy:tankdevel.pv
ArchestrA 引用 – Tank001.pv
ArchestrA 引用 – Pump001.speed
InTouch 引用 – Intouch:$second

图 10b