[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G05B 19/042 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200580014538.7

[43] 公开日 2007 年 4 月 18 日

[11] 公开号 CN 1950767A

[22] 申请日 2005.5.4
[21] 申请号 200580014538.7
[30] 优先权
[32] 2004. 5. 4 [33] US [31] 60/567,980
[86] 国际申请 PCT/US2005/015941 2005.5.4
[87] 国际公布 WO2005/109129 英 2005.11.17
[85] 进入国家阶段日期 2006.11.6
[71] 申请人 费舍－柔斯芒特系统股份有限公司
地址 美国德克萨斯州
[72] 发明人 马克·J·尼克松 坦尼森·郝
弗朗西斯·德卡兹曼
理查德·罗德里格兹
赖安·瓦尔德拉玛
迈克尔·J·卢卡斯
肯·J·贝欧格特
斯蒂芬·吉尔伯特
[74] 专利代理机构 北京德琦知识产权代理有限公司
代理人 周艳玲 朱登河

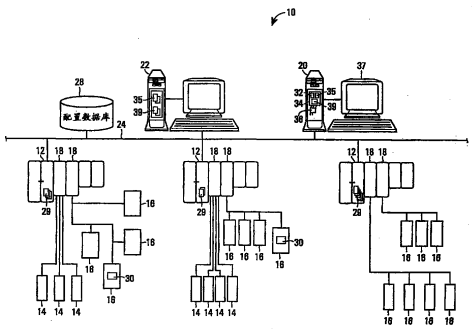
权利要求书 4 页 说明书 62 页 附图 14 页

[54] 发明名称

加工厂用户界面中基于标记语言的动态过程图形

[57] 摘要

一种用于加工厂的用户界面系统，包括图形显示编辑器，其用于配置过程图形显示，该过程图形显示具有代表加工厂的加工厂元素的图形显示元素。该过程图形显示通过用说明性语言表示的配置信息被指定。图形呈现引擎用于从所述配置信息得出的指令，在运行时期间生成过程图形显示的描绘。该过程图形显示的配置信息可以作为一对象被存储，比如，该对象可以包括第一部分和第二部分，分别用来定义图形参数和标识数据源。图形参数可以被集中用于定义加工厂元素的图形描绘，为此，可以按照该说明性语言的格式来表示。该数据源可以为数据指定一位置或路径，该数据用于表现出将要通过所述图形描绘来显示的加工厂元素的在线操作。



1、一种存储在计算机可读介质中的对象实体，其与用于加工厂的用户界面系统一起使用，所述对象实体包括:

第一部分，其通过该用户界面为所述加工厂的加工厂元素的描绘定义图形；以及

第二部分，其为数据标识一数据源，所述数据表现出通过所述描绘将被显示的加工厂元素的在线操作；

其中所述第一部分以说明性格式表述。

2、根据权利要求 1 所述的对象实体，其中，所述第一部分定义在呈现所述描绘时使用的形状对象的实例。

3、根据权利要求 1 所述的对象实体，其中，所述第一部分定义在呈现所述描绘时使用的复合形状对象的实例。

4、根据权利要求 1 所述的对象实体，其中，所述说明性格式基于可扩展标记语言。

5、根据权利要求 1 所述的对象实体，其中，所述说明性格式包括用于定义所述图形的脚本的矢量图形格式。

6、根据权利要求 1 所述的对象实体，其中，所述第一部分进一步定义数据转换参数，以指定表现出加工厂元素的在线操作的所述数据的图形描绘。

7、根据权利要求 1 所述的对象实体，进一步包括第三部分，其定义用来实现以仿真所述加工厂元素的在线操作的方法。

8、根据权利要求 7 所述的对象实体，其中，所述第三部分以所述说明性格式来表述。

9、根据权利要求 1 所述的对象实体，其中，所述第二部分以所述说明性格式来表述。

10、根据权利要求 1 所述的对象实体，其中，所述图形包括动画元素，所述动画元素具有表现出所述加工厂元素的在线操作的动画。

11、一种用于加工厂的用户界面系统，包括：

计算机可读介质；

图形显示编辑器，其用于配置过程图形显示，所述过程图形显示具有代表所述加工厂的加工厂元素的图形显示元素，其中，用于由所述图形显示编辑器生成的过程图形显示的配置信息按照说明性语言存储于该计算机可读介质中；以及

图形呈现引擎，其基于从所述配置信息获得的指令，在运行时期间生成该过程图形显示的描绘。

12、根据权利要求 11 所述的用户界面系统，其中，所述说明性语言定义用于表达所述配置信息的可扩展格式。

13、根据权利要求 11 所述的用户界面系统，其中，所述配置信息基于所述说明性语言按照对象模型框架来存储。

14、根据权利要求 13 所述的用户界面系统，其中，所述对象模型框架定义原始形状对象，所述原始形状对象可被所述图形显示编辑器提供，以便配置所述过程图形显示使之包括从所述原始形状对象构造的附加的图形显示元素。

15、根据权利要求 13 所述的用户界面系统，其中，所述对象模型框架定义复合对象，所述复合对象可被所述图形显示编辑器提供，以便配置所述过程图形显示使之包括从所述复合对象构造的附加的图形显示元素。

16、根据权利要求 13 所述的用户界面系统，其中，所述图形显示编辑器包括图形编辑工具，其用于从之前构造的存储于所述计算机可读介质中的过程模型对象创建所述复合对象。

17、根据权利要求 16 所述的用户界面系统，其中，所述图形编辑工具通过所述对象模型框架来定义。

18、根据权利要求 11 所述的用户界面系统，其中，所述说明性语言为可扩展标记语言。

19、根据权利要求 11 所述的用户界面系统，其中，所述说明性语言定义用于描述所述配置信息的基于 XML 格式。

20、根据权利要求 11 所述的用户界面系统，进一步包括一转换引擎，其基于所述配置信息的图形相关信息，按照另一说明性语言生成所述命令。

21、根据权利要求 20 所述的用户界面系统，其中，所述另一说明性语言按照矢量图形格式表述所述图形相关语言。

22、根据权利要求 20 所述的用户界面系统，其中，所述转换引擎进一步生成另外的命令，所述另外的命令为所述图形显示元素指定一数据转换例程。

23、根据权利要求 20 所述的用户界面系统，其中，所述转换引擎进一步根据所述过程图形显示的所述配置信息生成一数据源参考文件，所述数据源参考文件为连同图形显示元素一起显示的数据标识一数据源。

24、一种配置用于加工厂的用户界面系统的方法，包括:

使用定义复合图形元素的对象在将要通过该用户界面描绘的各个过程图形显示中创建该复合图形元素的多个实例;

在所述用户界面系统的计算机可读介质中存储用于定义该复合图形元素的所述多个实例的数据；和

通过修改所述对象来自动更新该复合图形元素的所述多个实例。

25、根据权利要求 24 所述的方法，其中，所述对象包括用基于 XML 图形语言来表述的定义。

26、一种存储于计算机可读介质中的对象实体，其与用于加工厂的用户界面系统一起使用，所述对象实体包括:

图形部分，其通过所述用户界面为该加工厂的加工厂元素的描绘定义图形;

参数部分，用于标识所述图形的可配置方面；以及

导航部分，用于为将要与所述图形一起显示的内容标识数据源;

其中，所述图形部分、参数部分以及导航部分被分散地存储于所述计算机可读介质中。

27、根据权利要求 26 所述的对象实体，其中，所述计算机可读介质包括多个存储器存储设备，以使所述图形部分、参数部分以及导航部分不存储在单个存储器存储设备中。

28、根据权利要求26所述的对象实体，其中，所述图形部分包括基于XML图形语言形式的描述。

加工厂用户界面中基于标记语言的动态过程图形

相关申请

本申请是2004年5月4日递交的顺序号为60/567,980、题为“用于表示、监测和与过程控制系统交互的图形用户界面”的美国临时专利申请的正式递交申请，并出于优先权的目的要求其权益，在此本申请明确地将其全部内容合并作为参考。本申请还与2003年7月21日递交的、题为“加工厂中图形显示元素、过程模块和控制模块的集成”、并在2004年8月5日以美国出版号2004/0153804出版的、顺序号为10/625,481的美国专利申请有关，该申请为2002年10月22日递交的、在2004年4月22日以美国出版号2004/0075689出版的、题为“加工厂中的智能过程模块和对象”的、顺序号为10/278,469的美国专利申请的部分继续申请，在此明确地将这两个公开的全部内容合并作为参考。本申请还与2003年2月18日递交的、题为“加工厂配置系统中的模块类对象”、并在2004年10月7日以美国出版号2004/0199925出版的、顺序号为10/368,151的美国专利申请有关，在此明确地将其全部内容合并作为参考。本申请还和与本申请在同一天递交的正在作为国际（PCT）申请的下列专利申请有关，在此明确地将这些专利申请全部内容合并作为参考：“过程环境中的关联图形显示”（代理备案号06005/41111）；“用于过程控制系统的用户可配置警报和警报趋势”（代理备案No.06005/41112）；“加工厂中过程模块和专家系统的集成”（代理备案 No.06005/41113）；“集成环境中具有定制的过程图形显示层的加工厂用户界面系统”（代理备案No.06005/41114）；“过程环境中的脚本图形”（代理备案No.06005/41115）；“过程配置和控制环境中的图形集成”（代理备案No.06005/41116）；“过程环境中具有多种显像的图形元素”（代理备案No.06005/41117）；“加工厂中用于配置图形显示元素和过程模块的系统”（代

理备案 No.06005/41118）；“用于统一的过程控制系统界面的图形显示配置框架”（代理备案 No.06005/41124）；“用于修改过程控制数据的方法和装置”（代理备案 No.06005/591622 和 No.20040/59-11622）；“用于访问过程控制数据的方法和装置”（代理备案 No.06005/591623 和 No.20040/59-11623）；“用于过程控制系统的集成图形运行时界面”（代理备案 No.06005/591628 和 No.20040/59-11628）；“面向服务的过程控制系统架构”（代理备案 No.06005/591629 和 No.20040/59-11629）。

技术领域

本发明总的来说涉及加工厂，更具体地说，涉及具有动态、集成过程图形的加工厂用户界面。

背景技术

分布式过程控制系统，比如用在化学、石油或其他过程中的分布式过程控制系统，通常包括一个或更多过程控制器，这些过程控制器通过模拟、数字或模拟数字混合总线，连接到一个或更多现场设备上。所述现场设备可以是阀门、阀门定位器、开关和变送器（例如，温度、压力、液位和流速传感器），它们位于过程环境内，并执行比如开/关阀门、测量过程参数等功能。智能现场设备，例如符合公知的 Fieldbus（现场总线）协议（比如 FOUNDATION™Fieldbus 协议）的现场设备，也可以执行控制计算、报警功能以及其他通常在控制器中实现的控制功能。过程控制器通常也位于工厂环境内，可以接收表示由现场设备所做的过程测量结果和/或关于现场设备的信息的信号，并可以执行一个控制器应用程序，例如该应用程序运行不同的控制模块，这些控制模块可以基于所收到的信息做出过程控制决策、生成控制信号，以及与在比如 HART、Fieldbus 现场设备之类的现场设备中正在执行的控制模块或者块协同工作。控制器中的控制模块通过通信线路将控制信号发送到所述现场设备，从而控制该过程的操作。

来自现场设备和控制器的信息通常可通过数据总线应用于一个或以上其它位于控制室或其他远离吵杂的工厂环境的硬件设备，例如操作员工作站、个人计算机、数据历史记录站、报告发生器、集中数据库等。在这些硬件设备上运行应用程序，这些应用程序比如使操作员能够执行与过程有关的功能，例如改变过程控制例程的设置、修改控制器或现场设备中控制模块的操作、查看过程的当前状态、查看由现场设备和控制器产生的警报、仿真过程的操作以培训人员或测试过程控制软件、维护和更新系统配置数据库等。

例如，由爱默生过程管理出售的 DeltaV™ 控制系统包含多种应用程序，这些应用程序存储在加工厂内位于多个地方的不同设备中，并由加工厂内位于多个地方的不同设备来执行。配置应用程序存在于一个或更多操作员工作站中，并使得用户可以创建或更改过程控制模块，并通过一个数据总线将这些过程控制模块下载到专用分布式控制器。通常，这些控制模块由通信上互连的功能块组成，这些功能块是面向对象编程协议中的对象，这些对象基于输入执行控制方案内的功能，并且将输出结果提供给控制方案内的其他功能块。配置应用程序还允许设计人员创建或改变操作员界面，这些操作员界面由一个浏览应用程序所使用，来向操作员显示数据并使得操作员能够在过程控制例程内更改设置，例如设置点。每个专用控制器以及某些情况下的现场设备，存储并执行一个控制器应用程序，该应用程序运行所分配和下载的控制模块来实现实际的过程控制功能。可以在一个或更多操作员工作站上运行的浏览应用程序，可以通过数据总线接收来自控制器应用程序的数据，并将所接收数据显示给过程控制系统设计者、操作员或者使用用户界面的用户，还可以提供任意多个不同视图，例如操作员视图、工程师视图、技术员视图等等。数据历史应用程序通常存储于一个数据历史设备中并由该设备来执行，该数据历史设备收集和存储通过数据总线得到的部分或全部数据，同时可以在一个与数据总线连接的计算机上运行一个配置数据库应用程序，来存储当前过程控制例程配置和与之相关联的数据。作为选择，配置数据库也可以与配置应用程序放置在同一个工作站中。

鉴于用在过程控制环境中的控制和支持应用程序的数量和种类都有所增加，为了使用户能够有效地配置和使用这些应用程序，已经有不同种类的图形显示应用程序问世。例如，为了使配置工程师能够通过图形来创建控制程序，且这些程序能够下载到加工厂内的控制设备上，图形显示应用程序已经被用来支持控制配置应用程序。另外，图形显示应用程序也已经被用来达到以下目的，例如，使操作人员能够查看加工厂当前的运行情况或者加工厂的各个区域，使维护人员能够查看加工厂内硬件设备的状态，能够启动加工厂的仿真，等等。然而在过去，这些图形显示应用程序是作为其所关联具体应用程序一部分被创建的，或者是为了支持其所关联具体应用程序而创建的，因此一般在使用性方面具有局限性，局限于其被创建时的具体过程功能。例如，很难或几乎不可能将一个被创建用于支持控制或其他操作员的图形程序应用于维护、配置或者仿真功能。

举一个具体的例子，一些过程控制配置应用程序目前包含一个模板对象库，例如功能块模板对象，或者在某些情况下的控制模块模板对象，这些对象用于为加工厂创建一个控制策略。模板对象具有默认的属性、设置和与其关联的方法，而且使用图形配置应用程序的工程师可以选择这些模板对象，然后实质上将所选模板对象的复制件放置在配置屏幕上，以开发控制模块。在选择和放置模板对象于配置屏幕的过程中，工程师将这些对象的输入和输出相互连接，并更改这些对象的参数、名字、标签和其他属性，以创建一个在加工厂中具有特定用途的特定控制模块。在创建一个或更多这样的控制模块后，工程师接着可以实例化该控制模块，并将该控制模块下载到合适的一个或更多控制器以及现场设备上，以便在加工厂运行期间执行。

其后，工程师可以使用一个不同的图形显示创建应用程序，通过在显示创建应用程序中选择和建立显示对象，为在加工厂内的操作员、维护人员等创建一个或更多显示。这些显示通常是在一个或更多工作站基于泛系统实现的，并能够为操作员或维护人员提供预配置显示关于控制系统或工厂内设备的运行状态。这些显示的形式通常包括：报警显示，所述报警显示接收并显

示控制器或加工厂内设备所生成的警报；控制显示，指示控制器或加工厂内其它设备的运行状态；维护显示，指示加工厂内设备的功能状态，等等。然而，这些显示一般会预先配置，从而以已知的方式来显示从过程控制模块或加工厂内设备所接收到的信息或数据。在有些已知的系统中，通过使用对象来创建显示，对象具有于物理或逻辑元素相关联的图形，并与所述物理或逻辑元素之间存在通信连接，从而可以接收关于该物理或逻辑元素的数据。对象可以根据所接收的数据改变显示屏幕上的图形，例如所接收的数据表明有一个储罐半满，表明流量传感器所测量的流量，等等。但是，用于配置、操作员控制、维护和仿真活动的图形显示一般都是使用不同的图形编辑器彼此单独创建的。

鉴于此，类似于控制配置应用程序，显示创建应用程序可以具有模板图形显示项，比如储罐、阀门、传感器、滑杆之类的操作员控制按钮、开关等等，这些项可以位于任何期望配置的屏幕上来创建操作员显示、维护显示以及其他类似显示。当把个别图形项放在屏幕上时，这些项可能在屏幕上相互连接，其连接方式可以为用户提供加工厂内部工作的一些信息或显示。但是，为了将图形显示做成动画，显示创建者必须通过在图形项和加工厂内的相关数据源之间指定通信链接，来手工将每个图形项与加工厂内生成的数据结合在一起，所述加工厂内生成的数据比如是传感器测量到的数据或者阀门定位器指示的数据，等等。这样的处理过程非常冗长耗时，而且可能会充满错误。

虽然控制配置应用程序中的控制模板对象和显示创建应用程序中的显示项比较方便，因为它们可以被复制和用于创建多种不同的控制模块和图形显示，但是经常需要为加工厂内不同设备创建大量同样的控制模块和图形显示。例如，很多情况下，中型到大型加工厂都有同样的或相似的设备，这些设备可以用同样的基本通用控制模块和显示来控制和查看。然而，为了创建这些大量的控制模块和显示，可以创建一个通用控制模块或显示模块，然后为每个可以使用所述通用控制模块或显示模块的不同设备都复制所述通用控制模块或显示模块。当然，在复制之后，必须在配置应用程序中手工更改

每个新的控制或显示模块，从而指定某个设备附着于哪个新的控制或显示模块，然后将所有这些控制和限制模块必须实例化并下载到过程控制系统。

不幸的是，上述控制模块和显示项并非在任何形式下都是模块化的。因此，在复制操作后，必须使用合适的配置应用程序来手工逐个更改所述控制模块和显示，从而指定工厂内的设备与哪个控制模块和显示相关联。对于一个工厂，如果它具有同一类型设备的很多拷贝（例如被复制的设备），这个过程将会冗长耗时，并充满人为操作错误。而且，这些不同的控制模块和显示一旦被编程，这些控制模块和显示将互不察觉。因此，为了在创建控制模块时进行某种改变，工程师或操作员必须手工对不同复制设备的每个不同控制模块进行同样的改变，这种操作同样也是冗长耗时的。同样的问题也存在于为工厂内不同套的复制设备所创建的图形视图。换句话说，一旦创建了一个具体控制模块或者一个具体的图形视图（分别建立或者从一个模板对象进行复制），然后将所建控制模块或图形视图连接到工厂内的一套特定设备，所述控制模块或图形视图是系统内的独立实体或对象，即使系统内的其他控制模块或图形视图与所述控制模块或图形视图相同或相似。结果，适用于每个特定类型的控制模块和图形显示的变化，对于这些模块和显示来说必须是单独进行的。当图形显示是为工厂内不同功能背景中，比如对于控制查看、维护查看和仿真功能的相同设备创建时，这个问题甚至更加明显。在这种情况下，就要单独创建图形视图，而这些视图相互间并不知道对方的存在。

因此，虽然图形视图已经被提供给和关联到不同应用程序，其中所述应用程序用于加工厂内进行的不同常规活动，但是这些图形显示和关联的图形显示编辑器一般被添加到功能级的应用程序，而所述图形显示和编辑器正是用于支持这些应用程序而创建的。结果，图形编辑器在一定程度上只是使用户能够建立一些视图，这些视图能够支持特定应用程序所需的特定功能。以前的加工厂没有提供这样的图形显示编辑器，该编辑器用于或者支持在工厂配置和支持范围内各种活动的图形需要。从而，举例来说，一个用于支持或者启动控制配置活动的图形显示编辑器只能使用户创建控制程序，而不支持

操作员或者维护显示的需求或功能。同样地，用于创建操作员视图、维护视图等（其中所述视图在工厂操作过程中被提供给控制操作员或者维护技师）的图形显示编辑器不支持与配置活动、仿真活动等关联的功能。由于需要在加工厂的各个功能级，例如在控制配置、维护支持、控制操作员支持和仿真支持功能级，支持图形显示，由各种编辑器所创建的不同显示最终模拟和描述工厂内的相同部件，结果导致各种不同的工作人员在加工厂的图形显示方面加倍付出努力。所述加倍努力不仅表现在需要建立描述不同用途的相同过程元素的不同图形显示，而且表现在将用于不同显示应用程序的图形元素连接到加工厂内所关联的实际硬件或软件单元。

由于之后已经提供了对于各种加工厂活动的图形支持，并且作为实际执行活动的一部分，图形支持没有被集成在工厂环境中来允许创建通用图形并在工厂的各种不同功能级处进行使用。这种图形非集成引发了针对不同功能或者不同应用的不同功能的图形，从而导致用户侧的疑惑，因为用户可能偶尔需要不时地查看与工厂内不同操作或功能相关联的不同显示，而用户一般只熟悉一种特定类型的图形显示。同样，正如上面所提到的，对工厂内各种不同功能级图形显示的支持，会导致图形支持的加倍，无论是在创建显示方面，还是在将显示内的元素适当地连接到工厂内实际硬件和软件单元方面。

另外，差错检测和其他编程对于检测与不同控制器中运行的控制环路相关的条件、错误、警报等、以及单个设备内的问题都很有用。传统上来说，这种差错检测在加工厂的不同功能级处执行，并被显示在为那些不同功能活动的图形显示上。因此，一直都很难对过程控制系统编程来识别系统级的状况和错误，其中这些状况和错误必须通过分析广泛分布在加工厂内的不同设备的数据才能检测到，而且，将这些类型的错误显示在操作员显示上更加困难，因为还没有建立将这种系统级的状况信息显示或呈现给操作员或维护人员的操作员显示。而且，也很难用显示内不同元素的这些交替信息源或数据源将操作员显示内的对象做成动画。

发明内容

根据本发明的一个方面，公开了用于加工厂的用户界面系统。所述用户界面系统包括：计算机可读介质和用于配置过程图形显示的图形显示编辑器，所述过程图形显示具有代表加工厂中加工厂元素的图形显示元素。用于由图形显示编辑器生成的过程图形显示的配置信息以一种说明性语言（declarative language）存储在计算机可读介质中。所述用户界面系统进一步包括图形呈现引擎，该图形呈现引擎基于从配置信息得到的命令生成运行时期间过程图形显示的描绘。

所述说明性语言可以定义用来表达配置信息的可扩展格式。作为可替换或者附加地，该配置信息可以基于该说明性语言依照对象模型框架来存储。所述对象模型框架可以定义原始形状对象，所述原始形状对象可由该图形显示编辑器提供，以便配置该过程图形显示使其包括由原始形状对象构造的附加的图形显示元素。所述对象模型框架还可以或可替换地定义复合对象，所述复合对象可由图形显示编辑器提供，以便配置该过程图形显示使其包括由复合对象构造的附加的图形显示元素。图形显示编辑器可以包括图形编辑工具，其用来根据之前构造的存储在计算机可读介质中的过程模型对象创建所述复合对象。所述图形编辑工具也可以通过对象模型框架来定义。

在一些实施例中，该说明性语言是一种可扩展标记（markup）语言。所述说明性语言也可以或者可替换地定义用于描述所述配置信息的基于 XML 格式。

用户界面系统还可以包括转换引擎、程序或其它工具，用于基于配置信息中图形相关信息按照另一说明性语言生成所述命令。该另一说明性语言，在某些情况下，可以按照矢量图形格式表述（set forth）该图形相关信息。该转换引擎也可以或者可替换地生成另外的命令，其用来为图形显示元素指定一数据转换例程。该转换引擎还可以从用于过程图形显示的配置信息生成一数据源参考文件，所述数据源参考文件为将要与图形显示元素一起显示的

数据标识一数据源。

根据本发明的另一个方面，公开了存储在计算机可读介质中的对象实体，其与用于加工厂中的用户界面系统一起使用。所述对象实体包括两个部分。第一个部分通过该用户界面为加工厂中的加工厂元素的描绘定义图形，而第二部分为数据定义数据源，所述数据表现出通过所述描述将被显示的加工厂元素的在线操作。所述图形参数用一种说明性格式表述。

在一些实施例中，第一部分定义在呈现所述描绘时使用的形状对象的实例。作为替换或者附加地，第一部分还可以定义在呈现所述描绘时使用的复合形状对象的实例。在上述任一种情况下，所述说明性格式可以基于可扩展标记性语言。此外，该说明性格式还可以包括用于表达所述图形的矢量图形格式。

该第一部分还可以进一步定义数据转换参数，所述数据转换参数用来指定表现出加工厂元素的在线操作的数据的图形描绘。所述对象实体还可以包括第三部分，其定义用来实现以仿真加工厂元素的在线操作的方法。该对象实体还可以包括第三部分，其中定义用来实现以仿真加工厂元素的在线操作的方法。所述第三部分和第二部分都可以用说明性格式表述，所述图形可以包括动画元素，动画元素包括表现出加工厂元素的在线操作的动画。

附图说明

图 1 是一个位于加工厂内部的分布式过程控制网络的框图，包括一个用于实现显示例程的操作员工作站，其使用智能过程对象创建过程模块和图形显示以对加工厂的操作进行仿真。

图 2 是一套应用程序和其他实体的逻辑框图，包括存储在图 1 所示的操作员工作站的智能过程对象和过程模块，可以用于在加工厂中实现增强的功能。

图 3 是对配置屏幕的简化描述，所述的配置屏幕被配置工程师使用存储在对象库中的智能过程对象来创建过程图形显示或过程模块。

图 4 详细描述了一个示例性的过程图形显示，其包括对加工厂内的流和连接元素的描述，这一描述通过互连数个智能过程对象中的图形显示元素来创建。

图 5 是在现存的过程控制网络中创建和实现使用智能过程对象的过程模块的方法的逻辑框图。

图 6－11 是对多个示例性的过程图形显示的某些部分的描述，这些过程图形显示使用基于标记语言的对象模型所配置和创建。

图 12－14 是类图中提出的对象模型框架各部分的简化描述，用来表明对象模型框架中类之间的关系。

图 15 是示例性的过程图形显示中已定义部件的示意图，包括以说明性语言提出的呈现定义。

图 16 是一个示例性的复合形状类中已定义部件的示意图，这些复合形状类可以用于创建和配置过程图形显示。

图 17 是数据库存储实例图的示意图，所述数据库存储实例图使用数个有已定义参数的复合形状来表明示例性过程图形显示之间的关系。

图 18 是根据结合图 15 和图 16 描述的配置环境元素，并结合运行期过程图形显示的生成而创建的组件的示意图。

图 19 是与运行期过程图形显示的生成相关创建的组件、文件和其他项目的进一步的示意图。

图 20 是用于示例性数据源的对象模型框架的数据源对象模型部分的示意图。

具体实施方式

这里公开的加工厂用户界面系统通常被配置和构成为以一致且熟悉的形式和以可扩展且灵活的方式向用户呈现支持加工厂的操作和维护的信息和内容。以一种可配置的、广泛的并且个性化的访问方式来创建、配置、存储和处理该通过用户界面呈现的内容和信息。为此，并且为了克服过去的加

工厂用户界面的上述缺点和局限，以一种灵活且可扩展格式提出、配置和定义该用户界面及其图形结构（比如，代表加工厂元素的图形显示元素以及菜单、板、面板和其他的用户界面结构）、到其他的文件或数据源的链路，以及其他的嵌入式形状或复合形状。所述格式可以以一种说明性语言或标记性语言来定义，这里的说明性语言或标记性语言指的是 PGXML(过程图形可扩展标记语言)。如这里进一步描述的，PGXML 可以基于工业上标准的标记语言 XML。通过 PGXML 对用户界面的综合定义可以支持高级图形的输出。为此，本文将要描述的 PGXML 脚本可以被转换成矢量图形格式，比如微软的 XAML 或者可伸缩的矢量图形（SVG）。接着，该 PGXML 脚本中的非图形相关元素可以用来创建数据结构、命令或其它的指令以及文件，以支持其它的通过用户界面的 PGXML 描述定义的其它功能实体。于是，该 PGXML 脚本一般提出了即将在一显示中描述的图形显示元素的图形结构的内部定义，以及这些元素的其它方面，以便支持生成将要被下载到工作站或其它显示设备上的过程图形运行期文件，其中所述显示将被描绘在上述工作站或其他的显示设备上。

用户界面的主要目的是为加工厂提供系统级的虚拟视图。为此，这里描述一种模块化的面向对象的方法来定义用户界面，该用户界面可以提供在线和离线状况下的加工厂的虚拟视图，并能实现系统级的交互和控制。一般来讲，图形对象可以包括用于规定数据源的属性，如下面描述的，其中这些数据源是加工厂中过程值的数据源，是过程控制系统中过程控制值的数据源，并且是其它系统或源中的值和数据的数据源。这些属性以及根据其生成的文件和其它的数据，相应地可以用来为由图形对象提供的过程图形建立数据通信链路。

以下对于加工厂用户界面方案的多个实施例的公开内容从向其提供用户界面的示例性加工厂的上下文描述开始。下面一部分是对智能过程对象的类型的描述，这些智能过程对象规定图形、仿真和其它功能，以支持加工厂在线运行和仿真运行的描绘。然后描述用于这些智能过程对象的对象模型框

架和说明性语言。

现在参见图 1，详细地图示了示例的加工厂 10，在该加工厂中，智能过程对象被用来形成过程图形显示和过程模块，这两者均可以与控制模块集成，以在工厂环境中提供增强的控制和仿真功能。具体地说，加工厂 10 使用有一个或一个以上控制器 12 的分布过程控制系统，每个控制器都通过输入输出（I/O）设备或卡 18 连接到一个或一个以上现场设备 14 和 16，例如这些输入输出设备或卡 18 可以是 Fieldbus 接口、Profibus 接口、HART 接口、标准的 4－20ma 接口，等等。控制器 12 还通过数据总线 24 连接到一个或一个以上主机或操作员工作站 20 和 22，数据总线 24 比如可以是以太网链路。数据库 28 可以连接到数据总线 24，并且作为一个数据历史记录器运行，该数据历史记录器用来收集和存储参数、状态和其它与控制器和工厂 10 内的现场设备相关联的数据，和/或作为配置数据库运行，当工厂 10 内的过程控制系统的当前配置被下载到或存储到控制器 12 和现场设备 14 和 16 中时，该配置数据库以存储当前配置。虽然控制器 12、I/O 卡 18 和现场设备 14 和 16 一般都位于和分布在有时恶劣的工厂环境中，但操作员工作站 20 和 22 以及数据库 28 通常位于控制室或其它不是很恶劣的环境中，以便容易控制人员或维护人员接近。

众所周知，对于每个控制器 12，举例来说它可以是爱默生过程管理所出售的 DeltaV^{TM} 控制器，它存储和执行控制器应用程序，所述控制器应用程序使用任意数量的不同的且独立执行的控制模块或块 29 来实现控制策略。每个控制模块 29 可以由普遍所称的的功能块组成，其中每个功能块是整个控制例程的一部分或子例程，并且与其它功能块协同操作（通过被称作通信链路）来实现加工厂 10 中的过程控制回路。众所周知，功能块可以是面向对象编程协议中的对象，通常执行如下功能之一：输入功能，比如与变送器、传感器或其他过程参数测量设备相关联的功能；控制功能，比如与执行 PID、模糊逻辑等等控制的控制例程相关联的功能，或者输出功能，该输出功能控制一些设备的操作，比如说阀门的操作，以执行加工厂 10 中的一

些物理功能。当然，存在混杂的或其他类型的复杂功能块，比如模型预测控制器（MPC）、优化器等。虽然 Fieldbus 协议和 DeltaV 系统协议使用在面向对象编程协议中设计和实现的控制模块和功能块，但控制模块也可以使用任何所需的控制程序设计方案设计，例如，顺序功能块、梯形逻辑，等等，并且控制模块不限于用功能块或其它特殊编程技巧来设计和实现。

在图 1 所示的工厂 10 中，连接到控制器 12 的现场设备 14 和 16 可以是标准的 4－20ma 设备，也可以是包括处理器和存储器的智能现场设备，比如 HART、Profibus、或 FOUNDATION™ Fieldbus 现场设备，也可以是任何其它想要类型的设备。上述现场设备中的一些现场设备，比如说 Fieldbus 现场设备（在图 1 中以数字 16 标识），可以存储和执行与控制器 12 中实现的控制策略相关联的模块或子模块，比如功能块。如图 1 中所描述的被设置在两个不同的 Fieldbus 现场设备 16 中的功能块 30，可以与控制器 12 中的控制模块 29 协同执行，以实现过程控制，这是公知的。当然，现场设备 14 和 16 可以是任何类型的设备，比如传感器、阀门、变送器、定位器，等等，并且 I/O 设备 18 可以任何类型的符合任何需要的通信或控制协议，比如 HART、Fieldbus、Profibus 协议的 I/O 设备。

在图 1 的加工厂 10 中，工作站 20 包括一套操作员界面应用程序和其它的数据结构 32，这些操作员界面应用程序和数据结构 32 可以被任何授权用户（这里有时是指配置工程师，有时是指操作员，但也存在其他类型的用户）访问，以查看连接在加工厂 10 内的设备或单元等，并向其提供相关功能。这套操作员界面应用程序 32 存储在工作站 20 的存储器 34 中，并且在这套应用程序 32 中的每个应用程序或实体适于在与工作站 20 相关联的处理器 36 上执行。虽然整套应用程序 32 被描述为存储于工作站 20 中，但一些应用程序或其它实体也可以在其它工作站或与工厂 10 相关联或其中的计算机设备中存储和执行。此外，这套应用程序可以向与工作站 20 相关联的显示屏幕 37 提供的显示输出，或者向任何需要的显示屏幕或显示设备提供的显示输出，这些显示屏幕或显示设备包括手持设备、笔记本电脑、其它的工作

站、打印机，等等。同样地，在这套应用程序 32 中的应用程序可以分解开并在两台或更多的计算机或机器上执行，并且可以被配置为彼此间协同操作。

一般来说，这套应用程序 32 提供或者能够创建和使用三种不同类型的实体，这些实体的操作可以被集成在一起，以在加工厂 10 中提供增强的控制、仿真和显示功能。更具体而言，这套应用程序 32 可以被用来创建和实现过程图形显示 35（一般提供关于加工厂一部分的操作员显示）、过程模块 39（一般提供加工厂一部分的仿真）和过程控制模块，比如控制模块 29，其一般提供或执行过程的在线控制。过程控制模块 29 一般是本领域公知的，并且过程控制模块 29 可以包括任意类型的控制模块，比如功能块控制模型，等等。下面将会更详细描述的过程图形显示元素 35 一般是由操作员、工程师或其它显示经常使用的元素，用于向用户，比如向操作员提供关于加工厂及其中的元素的操作、配置和安排的信息。过程模块 39 一般与过程图形显示元素 35 紧密相关联，并且可以被用来仿真加工厂的操作或仿真加工厂内一些相连的不同元素的操作，所述仿真是以在过程图形显示 35 中描绘的方式进行的。过程图形显示 35 和过程模块 39 被图示为在工作站 20 和 22 中存储和执行，但是过程图形显示 35 和过程模块 39 也可以被下载到并在任何与加工厂 10 相关联的其它计算机，比如笔记本电脑、手持设备等上执行。

图 2 图示了在工作站 20 的这套应用程序 32 中的一些应用程序和数据结构或其它实体。具体是，这套应用程序 32 包括控制模块、过程模块和图形显示的配置应用程序 38，该配置应用程序 38 被配置工程师用来创建控制模块、过程模块（也被称为工艺流程模块）以及相关联的图形显示。虽然控制模块配置应用程序 38 可以是任何标准的或已知的配置模块应用程序，但过程模块和图形显示配置应用程序可以使用一个或一个以上智能过程对象来创建过程模块和图形显示，其性质将在下面更详细描述。更进一步，虽然过程模块和过程图形配置应用程序 38 被分开显示，但一个配置应用程序可以创建这两种类型的元素。

智能过程对象 42 的库 40 包括示例和模板智能过程对象 42，这些智能过程对象 42 可以被配置应用程序 38 访问、复制和使用，以创建过程模块 39 和图形显示 35。如将会理解的，配置应用程序 38 可以被用来创建一个或一个以上过程模块 39，其中每个过程模块 39 由一个或一个以上智能过程对象 42 组成或创建，并且可以包括一个或一个以上保存在过程模块存储器 46 中的工艺流程或仿真算法 45。另外，配置应用程序 38 可以被用来创建一个或一个以上图形显示 35，其中每个图形显示 35 由一个或一个以上智能过程对象 42 组成或创建，并且可以包括任意数量的连接在一起的显示元素。图 2 以一种扩展的形式图示了一个图形显示 35b，该图形显示包括对一套过程元素，比如通过管道、导管、电缆、传送带等连接元素互相连接的阀门、储箱、传感器和流量变送器的描述。

执行引擎 48 在运行时操作或实现每个图形显示 35 和过程模块 39，以为操作员创建一个或一个以上由图形显示 35 定义的过程显示，并实现与过程模块 39 相关联的仿真功能。执行引擎 48 可以使用规则数据库 50，该规则数据库 50 定义将在过程模块 39 上实现的逻辑为一个整体，特别是这些逻辑也可以在上述模块中的智能过程对象中所实现。执行引擎 48 还可以使用连接矩阵 52，所述连接矩阵 52 定义工厂 10 中并且也是位于过程模块 39 中的过程元素间的连接，以实现过程模块 39 的功能。

图 2 更详细地图示了一个智能过程对象 42e。虽然智能过程对象 42e 被图示为作为一个模板智能过程对象，但将会理解，其它智能过程对象一般也会包括与所描述的智能过程对象 42e 相同的或相似的元素、特征、参数，等等，并且这些元素、特征和参数的细节或值可以根据智能过程对象的特点和使用在每个智能过程对象之间改变或有所不同。此外，虽然智能过程对象 42e 可能是面向对象编程环境中的对象，并且因此而包括相关联的数据存储器、输入输出和方法，但该智能处理对象也可以通过任何其它预期的编程范例或协议来创建或实现。

如将会明白的，智能过程对象 42e 在实例化之前是与某一个特定类型的

实体，比如图 1 中加工厂 10 中的物理或逻辑实体相关联的对象。但是，智能过程对象 42e 在复制和实例化之后，它可能会被关联到加工厂中的特定实体上。在任何情况下，智能过程对象 42e 包括用于存储数据的数据存储器 53，所存储的数据是从与智能过程对象 42e 相关联的逻辑实体上接收的或属于上述逻辑实体的数据。数据存储器 53 一般包括数据存储器 53a，其用来存储与智能过程对象 42e 所附属的实体相关的一般信息或永久信息，，比如制造商、修订版、名称、类型，等等。数据存储器 53b 可以存储变量或变化的数据，比如智能过程对象 42e 所附属的实体的参数数据、状态数据、输入和输出数据、成本数据和其它数据，包括与该实体过去存在时相关的数据，或与该实体现在位于加工厂 10 内时相关的数据。当然，智能过程对象 42e 可以基于周期或非周期被配置或编程，以接收上述数据（比如成本数据），所述的接收操作可以通过需要的通信链路从实体本身接收，也可以通过以太网总线 24 或其它任何需要的方式从历史记录器 28 中接收。数据存储器 53c 可以存储智能过程对象 42e 所附属的实体的图形表示，并用于通过操作员界面，比如图 1 中与工作站 20 相关联的屏幕 37，向操作员提供实际的显示。当然，所述图形表示可以包括关于实体信息的占位符（在数据存储器 53c 中以下划线标识），该信息比如由数据存储器 52b 中存储的关于实体的参数或其它变量数据所定义的信息。上述参数数据可以在图形表示在显示设备 37 上作为一个图形显示 35 的一部分被呈现给操作员时在图形占位符中显示。上述图形表示（以及智能过程对象 42e）可以包括预先定义的连接点（在数据存储器 53c 中用 X 标识），这些预先定义的连接点能使操作人员或配置工程师将上游或下游的元素附接到过程元素中，就像图形表示所描绘的那样。当然，这些连接点还能使智能过程对象 42e 感知到与配置在过程模块中的智能对象相连的元素，并且这些连接点还可以规定必须使用的连接元素的类型，比如管道、导管等，和与上述元素相关联的流，等等。

智能过程对象 42e 还可以包括一个或一个以上输入 54 和输出 56，使其能与使用智能过程对象 42 的过程模块之中或之外的其它智能过程对象相通

信。从输入 54 和输出 56 到另一智能过程对象的连接可以由配置工程师在对过程模块的配置期间进行配置，这种配置可以通过简单地将其它智能过程对象连接到上述输入和输出，或通过指定智能过程对象之间将发生的特定通信来进行。这些输入和输出中的一部分可以被定义为连接到一些智能过程对象，上述智能过程对象可以被连接到对智能过程对象预先定义的连接点上。这些输入 54 和输出 56 还可以由规则数据库 50 中的一套规则和连接矩阵 52 来确定或定义，其中连接矩阵 52 定义工厂 10 中不同设备或实体间的连接。输入 54 和输出 56 包括与之相关联的数据存储器或缓冲器，一般而言，他们用来提供从其它智能过程对象到智能过程对象 42e 的数据通信，或者用来提供存储在智能过程对象 42e 中或由智能过程对象 42e 生成的数据到其它智能过程对象的数据通信。这些输入和输出可以被用来在智能过程对象 42e 和过程控制系统中的其它对象之间提供通信，其它对象比如是控制器 12、现场设备 14 和 16 等等中的控制模块。

如图 2 中所示，智能过程对象 42e 还包括方法存储器 58，其用来存储零个、一个或多个方法 60（在图 2 中图示为方法 60a、60b、60c），上述方法可能是在使用了智能过程对象 42e 的过程模块的执行期间由智能过程对象 42e 实现的算法。一般来说，存储在方法存储器 58 中的方法 60 将会使用存储在数据存储部分 53a 和 53b 中的数据，以及从其它智能过程对象中获得的数据，或从其它数据源，比如配置数据库或历史纪录器 28 中获得的数据，通过输入 54 和输出 56 确定加工厂 10 中的相关信息或工厂 10 中的实体。比如方法 60 可以确定与由智能过程对象 42e 定义的实体相关联的差的或坏的操作条件，与加工厂 10 中的这些或其它实体相关联的错误，等等。可以基于智能过程对象的类型或类预先配置或提供方法 60，并且一般将在运行期间每次执行该执行引擎 48 内的智能过程对象 42e 时执行方法 60。一些在智能过程对象，比如智能过程对象 42e 中提供的示例方法 60，包括检测泄漏、死区、停滞时间、移动、变化、状况监控、计算成本或其它与实体相关的状况。

方法 60 也可以用来帮助就流过与智能过程对象相关联的过程实体的材料，对过程实体的操作进行仿真。于是，可以用方法 60 计算质量平衡、能量平衡、流量、温度、成分、蒸汽状态以及其它与工厂 10 中材料相关联的系统级或流级的参数，以仿真元素的操作，从而根据所提供的输入计算预期的输出。当然，这些仅是能存储在智能过程对象 42e 中并由其运行的少部分方法，还有其他可以使用的方法，并且这些可以使用的方法是由被呈现实体的类型，实体在加工厂中相连和使用的方式以及其它因素决定的。需要注意的是，虽然智能过程对象 42e 可以存储并执行能对系统级状况、错误等进行检测的方法，但这些方法还可以被用来确定设备、逻辑元素，比如过程控制模块和环路，以及其他非系统级的实体的其它信息。如果需要，方法 60 可以用任何需要的编程语言，比如 C,C++,C#等等编程或提供，或者可以被引用或定义规则数据库 50 中的可应用规则，其中上述可应用规则应该在智能过程对象 42e 执行期间运行。

如果需要，每个智能过程对象可以包括可应用算法库或可应用方法库，上述可应用算法库或可应用方法库可以用来定义连接在过程模块中的智能过程对象的仿真行为。图 2 以一种下拉菜单 61 的方式描述了智能过程对象 42e 中的这种库，并且相似的菜单可以与各个其他智能过程对象相关联。配置工程师可以经过比如下拉菜单 61 通过选择仿真算法（被称为方法 1、方法 2，等等）库中的一个算法，在智能过程对象置于过程模块 39 中时定义该智能过程对象的仿真行为。通过这种方式，配置工程师可以根据使用智能过程对象进行建模的过程的类型或特点为该智能过程对象定义不同的仿真行为。

如果需要，配置工程师可以选择提供自身保存的算法或其它用户提供的算法，以定义由智能过程块定义的过程元素的仿真行为。当智能过程对象被放置于过程模块 39 中或被过程模块 39 使用时，这种用户定义的算法（图示为在下拉菜单 61 中的“用户定义”条目）可以用来被提供到或存储于该智能过程对象中。上述功能实体使得仿真行为可以通过用户进行定制，因此可

以提供更好并且更准确的仿真。如果需要，下面将进行更详细的描述，智能过程对象 42 或每个过程模块 39 可以包括操作员可操作开关（比如电动开关或标记），操作员可操作开关使得智能过程对象中的仿真算法不能使用，从而使得过程模块的行为可以被高保真度的仿真包或程序确定，比如由 HYSYS 提供的仿真包。在这种情况下，智能过程对象或过程模块从高保真度仿真中获取仿真参数，并且不使用智能过程对象自身的仿真算法。

在执行引擎 48 执行图形显示 35 或过程模块 39 的过程中，引擎 48 实现由输入 54 和输出 56 定义的到图形显示 35 中的每个智能过程对象或过程模块 39 中的通信，引擎 48 还可以为上述每个对象实现方法 60，以执行方法 60 提供的功能函数。如上所述，方法 60 的功能函数可以位于智能过程对象的程序中，或由规则数据库 50 中的一套规则定义，引擎 48 根据智能过程对象的类型、类、标识、标签名等等执行上述规则，以实现由这些规则定义的函数。

需要注意的是，智能过程对象 42e 的实例在与其相关联的过程模块的上下文中有标签或唯一的名字，并且上述标签或唯一的名字可以用来提供与智能过程对象 42e 之间的通信，并可以由执行引擎 48 在运行时引用。过程模块的标签在控制系统配置中应该是唯一的。标签转换使得过程模块 39 中的元素可以被过程图形显示 35、过程模块 39 以及控制模块 29 中的其它模块中的元素所引用。更进一步，智能过程对象 42e 中的参数可以是简单的知道与其关联的预期单位和属性的参数，比如简单值、结构参数或智能参数。智能参数可以由过程规则引擎或执行引擎 48 解释和使用，以保证所有的信号以相同单位被发送或进行适当地转换。智能规则还可以为智能过程对象（或过程模块）开关警报组，从而为操作员创建智能告警策略和/或智能告警界面。更进一步，智能过程对象类可以与工厂 10 中的过程控制策略中的设备和模块类相关联，以在智能过程对象和需要被解释或访问的过程变量之间提供已知的链接。

当智能过程对象用于过程图形显示或过程模块中时，智能过程对象还可

以包括操作模式、状态和告警行为，使得这些智能对象可以在运行时被置于不同的模式，比如关模式、启动模式和正常模式，上述智能模型也可以基于其当前操作状态提供与对象相关联的状态，可以根据检测条件，比如超范围参数、界限参数、高可变性的参数等等提供告警。此外，上述状态可以反映在智能过程对象的连接中，上述状态也可以依赖或使用智能过程对象的连接，以便使呈现对象的图形显示元素的视图形成动画。上述状态可以通过智能过程对象反向计算，以影响前面过程或显示中已经出现的智能过程对象的状态。如下所述，智能过程对象还可以具有类/子类的层次结构，该层次结构使智能过程对象可以在类库中被分类，在复合结构中被收集在一起，等等。更进一步，智能过程对象可以使用其它元素的信息，比如使用控制模块和其它对象的信息以使得智能过程对象感知何时与其相关联的实体处于忙碌状态，或者使用比如通过工厂 10 中的批量控制过程获得的信息。

智能过程对象可以与任何所需的过程实体相关联，比如物理设备或逻辑实体，其中物理设备如泵、储罐、阀门等等，逻辑实体如过程区域、测量结果或执行器、控制策略等。在一些情况下，智能过程对象可以与连接器相关联，连接器比如管道、导管、线路、传送机或任何其它在过程中从一点到另一点与移动材料、电、气体等等的设备或实体。上述与连接器相关联的智能过程对象，有时也被称为智能链路或连接器元素，它们还被加上标签（即使实际的设备或连接器本身没有被加标签或不能在加工厂 10 中通信），并且这些与连接器相关联的智能过程对象一般用于表示过程中的其它元素之间的材料流。

典型地，智能链路包括定义的不同材料或现象（比如电）如何流过所述连接（比如蒸汽、电、水、污水，等等）的属性或参数。这些参数可以指明通过连接器的流的类型和特点（比如大体的速度、摩擦系数、湍流的和非湍流的流的类型、电磁，等等）以及通过连接器的流的方向或可能方向。智能链路可以包括程序或方法，它们使得智能链路连接到的源对象或目的对象的单位相匹配，并且如果上述源对象或目的对象的单元不匹配可以进行转换。

智能链路的方法可以使用模型或算法对通过连接器的流建立模型，以估计通过实际的连接器的流的速度或特点、物理连接的长度和大小、传输时延，等等。为智能过程对象存储的参数（比如摩擦参数）可以在上述方法中使用。因此，从本质上说，上述智能链路或连接器元素使得智能过程对象可以感知其它的上游和下游实体或对象。当然，例如，智能链路可以以任何所需的或方便的方式定义系统中其它对象之间的连接、流体的类型，比如液体、气体、电流等等，上述智能链路还可以以任何所需的或方便的方式定义材料流、流体流、电流的方向等等，以及该智能过程对象上游和下游实体的上游侧和下游侧。在一个实施例中，矩阵 52 可以在执行过程流模型之前创建，并且可以为智能链路定义工厂中不同设备之间的互相连接，从而定义不同智能过程对象之间的互相连接。事实上，执行引擎 48 可以使用矩阵 52 确定上游和下游的实体，从而定义智能过程对象之间的通信和与智能过程对象相关联的方法。更进一步，智能过程对象可以使用一套或更多套规则来与其它的智能过程对象交互，并从对方获得智能过程对象中的方法所需要的数据，以解决与输出连接相关联的智能对象的影响。

如果需要，智能过程对象 42e 还可以包括到关键文档的热链路，比如 URL，其中关键文档可以可应用于该类对象，或者可以是智能过程对象 42e 所属的设备的实例（取决于临界状态和应用程序）所特有的。上述文档可以由销售商提供也可以是用户特定的。上述文档的一些例子包括配置、启动和关闭程序以及运行和维护文档。如果需要，操作员可以单击操作员显示上显示的对象，以为对象或相关设备提供具体（如果有的话）的实例和一般文档。并且，操作人员也可以增加/删除/改变独立于系统软件的文档，比如：维护请求、关于操作问题的记录等等。此外，这些热链路可以由用户配置或者由用户更改，从而在操作员界面上提供增加到对象知识链接的能力，以提供与对象相关链的合适信息的快速导航，并且提供针对特定的对象类型甚或特定的对象实例，向用户增加具体的工作指令的功能。

虽然上述过程模块和过程对象被描述为通过不同智能过程对象的相互

连接共同创建，但他们也可以被单独创建。比如，过程图形可以使用智能过程对象创建，当创建完成时，用于上述过程图形的过程模块可以基于在图形显示中的图形元素及其相互连接而生成。作为一种可替换的方式，过程模块可以首先使用智能过程对象被创建，并且一旦被创建，用于该过程模块的图形显示会被配置应用程序 38 使用智能过程对象中用于创建该过程模块的图形显示元素自动生成。更进一步，过程模块和图形显示可以被单独创建，并且在过程模块和图形显示两个实体中的单独元素，可以通过彼此引用（比如：使用图形显示和过程模块中元素的标签属性）被手工配置从而相互联系。通过这种机制，一智能过程对象可以被多个显示所引用。在任何情况下，一旦过程图形显示和相关联的过程模块被创建，他们可以独立运行或单独运行，尽管他们通常根据需要来回交互参数和信息。

一般说来，可以在配置应用程序中提供一套预先定义的图形元素来使用户能够创建反映加工厂状态的操作员显示或图形显示。这些图形元素被设计用来动态显示在线的并与控制系统相连接的测量装置和执行器。另外，可以使用过程模块中提供的在线过程仿真计算反映过程操作的非测量参数，并且以相关联图形显示的组成部分的形式来显示这些非测量参数。

另外，在为了工程或培训仿真目的而使用的离线环境中，由过程模块提供的过程仿真可以用于代替图形元素中的过程测量值，并且可以用在相关联的控制模块中。由相关联的过程模块计算得出的上述过程测量值，可以基于过程图形中图示的手动干扰值和执行器的位置和状态而计算得出。以这种方式，在线或控制情况和离线或仿真情况都可以使用过程图形显示和控制模块。并且，虽然在很多情况下，图形元素的静态部分看起来类似于包含在已知图形库中的三维组件，但下面将会针对图形元素的若干可能类型和实例，描述上述图形元素的独一无二的特征或属性、用上述元素显示的信息，以及上述元素到控制系统 I/O 和过程仿真模块的链接。

一般来讲，与智能过程对象相关联的过程模块中的图形元素和仿真算法，属于若干种不同类型的过程元素中的一种类型，所述过程元素包括流元

素、过程连接元素、执行器元素、处理元素、测量元素和估计属性元素。流元素一般定义加工厂中的材料流，并且可以被公开在图形显示中，以示出成分、密度、温度、流量、压力、重量和/或任何定义材料流的其它参数。流元素可以在过程模块的输入处定义，并被提供通向过程模块内的元素，从而使得通过过程模型的材料流可以被建立模型并在图形显示中得到描述。类似地，可以在过程模块输出处或过程模块的末端图示流元素，从而可以在图形显示中图示由图形显示描绘的加工厂部分的材料输出。流元素也可以用来定义不同的图形显示（以及相关联过程模块）之间如何相互连接。例如，一个过程模块的输出流可以作为另一个过程模块的输入流，并且一个过程模块的输出流可以提供另一个过程模块的输入流所使用的数值。流可以包括以下四部分：名字（比如：pH 流），方向（比如：流输入），测量（比如：流量、温度、压力），和成分（比如：氮、氨，等等）。但是，如果需要的话，流可以有其他部分或其它参数。

过程连接元素定义工厂中材料，比如固体材料、液体、蒸汽和气体材料从一个设备传递到或运送到另一个设备的方式。为了清楚地图示通过过程的材料流，可以使用三种不同类型的过程连接方式，包括：管道（piping）、导管（duct）、传送带。当然，也可以使用其它的连接元素，比如在电学－化工过程中传送电力流的电缆，等等。管道一般用来图示（和仿真）工厂中的液体流和高压蒸汽流或高压气体流。导管一般用来图示（和仿真）工厂中的低压气体流。传送带一般用来图示（和仿真）处理元素之间固体材料的运动。结果，每个过程连接元素都定义了连接的类型，比如：管道连接、导管连接或传送带连接，上述连接用于在设备的输入或输出处提供材料。

如果需要，可以由上游输入确定由连接传送的材料的属性。上述信息加入定义所述连接是否完整的连接状态变量可以获得图形显示上连接元素的属性。连接元素可以在处理元素输出、执行器元素输出或流元素输出处开始。以相似的方式，连接元素可以在处理元素输入、执行器元素输入或流元素输入处终结。

当光标位于图形显示中连接元素的上方时，连接元素的属性可以自动地显示出来。并且，通过将测量元素或估计属性元素（在下面定义）置于连接元素上，可以长期显示与连接元素相关联的属性。如果需要，可以通过在元素输出（比如：流输出、处理元素输出或执行器元素输出）的上方按住鼠标左键，同时按下鼠标的按钮，将光标定位在元素输入上方，来创建连接元素。对于即将成功创建的连接，上游和下游元素的输入输出类型（管道、导管或传送带）必须匹配。上述连接会自动呈现上游元素的类型。

如果需要，可以将管道元素显示或描绘在过程图形显示中作为管道连接，可以将导管元素（比如：空气或气体）显示作为导管，并且可以将传送带元素显示作为传送带。管道、导管和传送带元素连接可以在处理元素之间被自动形成路径，并且箭头可以在这些元素描绘之外显示出来，以表示流的方向。如果上游输出对于两个连接是公用的，那么在管道、导管或传送带上可以包括一个“T”元素。类似地，“T”元素可以用来结合多个输出。可以改变传送带元素的颜色或其它图形属性，从而指示传送带元素的状态，比如运行/停止、流动/不流动、塞住状态，等等。一般而言，沿传送带传送的材料流由连接到传送带的马达驱动器确定。于是，马达驱动执行器（即，将在下文详细描述的执行器元素）可以连接到传送带。另外，测量元素（在下文描述）可以被连接到管道、导管和传送带元素，以便可以显示与管道、导管或传送带元素相关联的测量结果，比如传送带的速度，或者管道或导管中材料流的速度，位于传送带、管道或导管之上或之中的材料的属性，比如湿度或重量。并且，被显示的元素属性可以被添加到位于管道、导管或传送带之上或之中的材料的未被测量的显示属性中，比如材料的成分。

如果需要，每个管道、导管和传送带连接元素可以以图形的方式和动态的方式反映出失去的连接（比如通过改变颜色），以及所选择的属性（压力、温度、长度等等）超出配置限制之外的情况（通过改变颜色）。此外，由相关联过程模块计算得到的参数可以显示在图形中。例如：不管连接状态是好是坏，由上游连接提供的属性，关于连接元素的一个或更多个所选择的参数

的界限，等等，这些可以在图形显示中展现出来，从而为操作员提供关于连接元素或由连接元素传送的流的信息。

一般而言，执行器元素是执行与流相关的传动功能的元素，并且执行器元素可以位于不同的连接元素之间或位于处理元素和连接元素之间。执行器元素的例子包括调节阀（带有执行器）、开关阀（带有执行器）、泵（带有马达）、送风机（带有马达）、引风机（带有马达）、喷射器（带有开关阀）、阻尼器（带有驱动器）、给料机（带有变速马达）、传送带马达驱动器（其可以连接到传送器元素），等等。

阀门元素的图形描绘可以动态地反映暗含的阀门位置(比如通过动画)、阀门故障（比如通过改变颜色）、阀门的全开/全闭位置（比如通过改变颜色）以及控制上述阀门的相关联控制块的 AO、DO、DC、设置点、PV、OUT、模式，等等（比如通过字符串或其它指示信息）。与阀门元素相关联的仿真元素(用于过程模块中)可以带有计算与阀门执行器相关的参数的仿真算法，这些参数比如：排出压力、质量流量、液体温度、液体成分、入口压力和出口压力。如果需要的话，这些仿真的参数或计算得到的参数可以在过程图形中显示。然而，用户或配置工程师必须经常配置到在与阀门相关联的控制模块中的 AO、DO 或 DC 块的引用，以及阀门类型（比如线型、快速开启型、等百分比型、阀门定尺寸等等)和阀门从开启到关闭所需的冲程时间。当然，用于仿真阀门在流过阀门的材料流上的操作的仿真算法，可以取决于阀门的类型和尺寸信息。

泵元素的图形描绘可以动态地反映马达的状态（比如通过改变颜色）、相关的 DO 或 DC 功能块模式和设置点（比如使用字符串）、马达速度（如果使用了变速驱动器）、AO 设置点、PV,OUT 模式（如果使用了变速驱动器）以及其它所需的参数。同样，用于该元素的过程仿真（用于过程模块中）可以确定或计算一些参数，比如排出压力、液体成分、液体温度和质量流量，上述参数可以在图形显示中呈现。用户可能需要依据泵的类型定义泵的曲线。但是，用户也可以配置到与马达启动/停止相关联的 DO 或 DC 块的引用，

到用于变速驱动器（如果使用）相关联的 AO 功能块的引用，以及用于定义泵的操作的泵曲线（比如：压力对流量）。

通风或引风机执行器元素的图形描绘可以具有动态反映马达的状态，DO 或 DC 功能块模式和设置点，马达速度（如果使用了变速驱动器），AO 设置点，PV、OUT、DO 或 DC 功能块模式（如果使用了变速驱动器）以及其它所需的参数的描述，上述内容都会呈现在图形显示中。用于该元素的过程仿真元素（用于过程模块中）可以确定或计算一些参数，比如：排出压力、气体成分、气体温度和气体质量流量，上述参数可以呈现在图形显示中。用户可以为马达的启动/停止配置到相关联的 DC 块的引用，为变速驱动器（如果使用了的话）到配置 AO 块的引用，以及为定义风机的仿真操作配置风机曲线（压力对流量）。

在一些情况下，某种特殊类型的执行器可能会用于一种具体的连接类型中，比如管道、导管或传送带。下表为一些典型的执行器元素定义了一些示例性连接限制。

	管道	导管	传送带
调节阀	X		
开关阀	X		
泵	X		
喷射器	X		
压力通风机		X	
引风机		X	
阻尼器驱动器		X	
给料器	X		X
马达驱动器			X

过程元素包括可以以某种方式处理工厂中材料或流的工厂设备。一般而

言，所有到和处理元素的输入和从处理元素的输出将会通过连接元素完成。标准的处理元素包括储罐（垂直的和水平的）、加热器、静态混合器、反应器、混合器、空气加热器以及其他能够执行某种简单处理活动或标准处理活动的元素。对于标准的处理元素，用户可以规定输入到或输出到元素的数量，以及物理设备的属性比如尺寸、容量等等。可以设置这些标准的处理元素的仿真算法和静态表示，使得他们不能由用户修改但可以像前面所述的在配置期间可选。当然，如果需要，其它典型的和更复杂的工厂设备（比如蒸馏塔、蒸发器、分离器、锅炉等等）可以实现作为定制的处理元素。可以修改上述静态表示，输入输出的数量和定制的处理元素的仿真算法以满足用户界面的需求。一旦定义了定制的处理元素，上述内容可以被作为组合或模板保存，并且组合或模板可以在其它处理元素的创建过程中作为起始点被重复使用或使用。

储罐标准处理元素（不管是垂直的还是水平的）可以基于到该储罐的管道连接进行配置，并且储罐元素可以动态反映储罐的液位（比如使用动态动画），以及储罐的状态是空还是满（比如通过改变颜色）。用于储罐的过程模块仿真可以通过图形显示计算和呈现一些参数，比如：出口温度、出口成分、液体温度和储罐的仿真液位。但是，为了将储罐连接到系统上，用户或配置工程师可能需要配置输入连接和输出连接的数量，以及到储罐的完整连接和配置储罐属性，比如尺寸（例如：直径和高度），等等。

加热器处理元素可以动态地通过图形显示计算和反映热量传送系数（比如：通过改变颜色）、出口产品温度、入口产品温度、出口压力（假定呈固定的下降趋势），等等。用户或配置工程师可能需要配置到加热器的完整连接、加热器表面区域和在加热器清洁时的传热系数。

当然，其它的一些处理元素比如静态混合器、反应器、混合器、空气加热器、热交换器等等，可以具有适用于这些设备类型的显示和仿真能力。非标准的处理元素比如蒸馏塔、蒸发器、分离器、锅炉等等，可以使用定制的处理元素用图形表示，在上述定制的处理元素中，与容器相关联的仿真，如

果标准选择中没有的话，则可以是用户定义的。这些元素中的处理可能作为与该容器相关的每一个输入和输出的阶跃响应模型来描述和定义。输入可以是气体流或液体流。可选地，用户可以定义用于描述处理元素的输入输出之间的关系的等式，并且这些等式可以保存在使用该元素执行仿真的过程模块中。如果需要，可以提供给用户一些简单的静态图形表示，以帮助用户快速创建与定制的处理元素相关联的静态图形。如果使用了这些简单的图形，那么用户可能只需要规定所需的输入和输出连接数量和定制的处理元素所支持的连接类型（比如管道、导管或传送带）。作为响应，会显示图形项，并且图形项会立刻用于操作员图形的创建中。如果需要，在用户选择了规定仿真算法为阶跃响应时，与处理元素的每个输入和输出关联的增益以及任何动态特性可以被设定。如果用户选择了定制的算法，那将会为用户提供表达式编辑器以定义仿真算法。可以基于选择的方法分别计算定制的处理元素输出的属性。此外，用户可以引用在单独的软件集合中定义的一个或更多个算法。

另外，可以提供一些预定义的组合或模板以用于创建定制的处理元素。这些模板可以包括，如有定制算法的锅炉模板，其中该定制算法可以计算出口气体氧气，出口气体一氧化碳、生成的蒸汽，锅炉包液位和锅炉通风。这种模板可以基于单个的燃料输入。然而，通过修改这些模板，可以利用多种燃料来仿真锅炉。其它预定义的模板可以包括专用的容器旋风分离器模板，上述容器旋风分离器模板可以与喷雾干燥器定制处理元素共同使用，并且可以包括一个阶跃响应模型来对分离器的操作进行建模。同样，塔模板、喷雾干燥剂和蒸发器可以使用阶跃响应模型定义预期的过程响应。在蒸发器中，可以基于能量输入和输入流的浓度计算输出流的浓度和蒸汽释放量。多个蒸发器元素可以一起与热量交换器和喷雾器元素相连接，从而创建一个多效蒸发器。类似地，专用的容器堆定制的模板处理元素可以与锅炉处理元素一起使用。在这种情况下，如果需要，可以将入口的属性不经任何修改贯通于该堆，或者使入口的属性反映堆中执行的排出物减少量。

可以用来创建过程图形显示和过程模块的其他类型的元素包括测量元

素和属性元素。其中，测量元素包括可以用于在图形显示中访问与物理变送器相关联的测量值的变送器元素，以及开关元素。一般而言，变送器元素可以动态反映控制块中相关联的 AI 功能块的模式、AI 功能块的不好的或不确定的状态，还可以动态反映与实际的变送器（传感器）相关联的测量值和测量单位或与实际的变送器相关联的其他数据。在离线模式（或仿真模式）中，变送器元素可以用于访问和显示由过程模块提供的仿真值，而不是用来访问或显示与 AI 或 PCI 块相关联的值；或者，变送器元素可以用于向控制模块内相关联的 AI 块提供测量值，以作为仿真控制例程中要使用的测量值。上述变送器元素可以被添加到连接元素或处理元素上，并且，当这种变送器元素被添加到显示上时，用户一般需要标识正在提供测量的控制器模式中相关联的 AI、PCI 或 DI 模块。在线模式下，可以紧挨着该测量元素显示该测量值。在离线模式（或仿真模式）下，可以自动显示测量结果的仿真值（由相应的过程模块产生）。在线操作时，在测量失误时，用户可以选择切换控制和显示到仿真值。

开关元素可以动态地反映不利的或不确定的状态、相关联 DI 的模式（比如手动或 OS）以及开关的离散值（开、关，等等）。在离线仿真模式下，用户可以通过选择仿真值或手动值和状态，以及通过手工输入开关的值和状态，从而使用开关显示元素访问和改变图形显示和控制模块中的开关参数。然而，一般而言，用户可以通过提供到控制模式中相关联 DI 块引用、到启动开关的元素属性的引用以及提供与开关状态改变相关的界限和死区，来配置开关元素。

估计属性元素一般显示由过程模块确定的系统的估计属性，并且可以被添加到连接元素或处理元素中以显示连接元素或过程元素的任何属性。当估计属性元素置于连接元素中或置于设备中时，用户可以浏览和选择将要显示的属性。于是，通过物理测量不能获得的仿真属性可以通过使用估计属性元素来显示。上述估计过程元素可以动态反映好/不好的连接、估计的一个或多个属性值以及超过相关界限或变化的属性。用户一般必须配置对将要显示

属性的引用，并且若该属性超过了界限，则用户还需要为该元素配置界限和颜色改变。

将会理解，通过将变送器元素和估计属性元素添加到处理元素、执行器元素和连接元素中，在线操作时或离线仿真时将引用与上述加工厂元素的输入和输出相关联的属性。上述属性在过程图形显示中也可以是可视的。

一般而言，操作员可以运行或执行配置应用程序 38 以创建一个或多个过程模块 39 或图形显示，从而在工厂 10 的操作期间实现或在仿真环境中实现。在一个实施例中，配置应用程序 38 向配置工程师呈现如图 3 所示的配置显示。如图 3 所示，配置显示 64 包括库或模板区 65 和配置区 66。其中，模板区 65 包括对几组模板智能过程对象 67 的描述，上述模板智能过程对象 67 可以包含图 2 的智能过程对象 42，也可以是上述的任何一种连接元素、测量元素、流元素、处理元素和估计属性元素。如果需要，还可以提供仅有图形定义的非智能元素 68。实质上，模板 67 和 68 都是可以被拖放于配置区 66 中的类属对象，这些类属对象可以被托放于配置区 66 上，从而创建过程模块或图形显示（或为二者）内的智能过程对象的实例。图中显示了部分完成的过程图形显示 35c，其包括一个阀门、两个储罐、两个泵、一个流量变送器和与流路连接器相连的两个传感器，其中流路连接器可以是上面描述的智能链路或连接器元素，并且可以提供流量输出。需要注意的是，图形显示 35c 可以由智能过程对象和非智能元素共同组成。

在创建图形显示时，比如创建图形显示 35c（或过程模块）时，配置工程师可以选择模板区 65 中的智能过程对象 67 和元素 68，将二者拖动到配置区 66 上，并将它们放置于其中想要的位置。一般而言，配置工程师将会选择和拖动一个或多个用于描述设备的智能设备过程对象 67a 或非智能元素 68 到配置区 66 上。随后，配置工程师会将配置部件 66 中的智能设备过程对象与智能连接器过程对象 67b 相连，并放置输入和输出流 67c 于显示中。此外，非智能元素也可以添加到显示中。在上述过程中，配置工程师可以使用弹出式属性菜单等改变每个智能过程对象的属性，特别是，配置工程师可

以改变与上述智能过程对象相关联的方法、参数、标签、名称、热链接、模式、类、输入和输出，等等。当过程或配置工程师利用每个所需的元素创建了过程模块之后，典型的是，创建的过程模块可以表示过程配置、区域等等，配置工程师可以定义与该过程模块相关联的规则或功能实体。上述规则可以是执行规则，比如与系统级方法像质量平衡和流量计算的性能相关的执行规则。当过程显示在线时，过程工程师或操作员也可以决定增加有用的趋势线和面板。在创建了图形显示 35c 之后，配置工程师可以在存储器中保存上述显示，并且可以在保存的同时或保存之后，以执行引擎 48 提供图形显示的方式将上述显示实例化并下载到执行引擎 48 上。当然，配置工程师可以以相同或相似的方式创建过程模块，尽管过程模块元素可以用不同的图形描绘，而这是与过程图形显示元素相反的。此外，在各个级别正在工厂中运行时，操作员可以选择打开不同级别的细节。比如，一个级别细节可以显示每个连接处的组成。

如上所述，过程图形或过程模块可以具有特定标签。例如，图形显示或过程模块内的智能过程对象元素可以被提供一标签，该标签包括别名，该别名能够由例如执行引擎 48 在运行时基于诸如过程控制系统内的某个设备或所选择的路由之类的其它因素进行填充的或选择的。在美国专利专利 No. 6,385,496 中详细论述了过程控制系统中别名的使用和间接引用，上述专利的受让人与本发明的受让人是同一个人，并且特此专门将其内容并入本发明作为参考。上述任一种技术可以为这里描述的智能过程对象的标签提供和解析别名。通过使用别名或类似物，相同的过程模块可以包括或者用于支持几套设备的不同视图，等等。

配置应用程序 38 可以用来以多层方式设计图 3 中的显示 64，使得比如可以用选项卡（视图 1、视图 2 和视图 3）来访问和创建过程模块或图形显示中的不同视图。上述选项卡也可以在配置环境中以用来访问和创建不同的视图，并且这些选项卡在运行时（比如，当执行引擎 48 为用户生成过程图形显示时）可以或不可以用来进行上述视图之间的切换。在上述任何一种情

况下，不同的视图可以选择性地被提供，以支持与加工厂相关联的不同用户。

一般而言，当配置工程师创建了过程模块或图形显示时，配置应用程序 38 会自动在数据库中存储智能过程对象及其之间的连接。然后，数据库可以使用一个或更多个相同的智能过程对象创建其他的过程模块或图形显示。这样，当创建其他的显示时，配置工程师可以简单引用已被创建并存储于数据库中的智能过程对象，以及与之一并存储的任何方法等等，从而把智能过程对象置于另一显示中。利用这种方式，随着过程模块和图形显示的创建，数据库的内容可以增加，并且可以在任何时间，使用该数据库，以便通过使用已经存在于过程流数据库内的智能过程对象来创建并执行其它显示和模块。通过使用上述数据库，每个位于数据库中的智能过程对象可以支持或用于过程模块中，并且可以在多个图形显示中进行引用。可以理解的是，通过为上述模块创建显示，然后规定用于过程模块或与过程模块相关联的流算法，可以构建过程模块。当然，单独的过程模块可以分散到不同的计算机，并由不同的计算机执行，并且过程模块彼此之间可以在相同计算机上或不同计算机上以通信方式互连以便彼此协同操作。当上述操作完成之后，可以从外部引用输入输出流以使其与过程模块绑在一起。

如上所述，作为过程模块或图形显示创建的一部分，配置工程师可以附上或提供过程模块的仿真算法。可以预先配置上述仿真算法以计算或确定某些过程属性或系统级属性，比如有关由过程模块描述或建模的过程的质量平衡计算、流量计算、效率计算、经济计算，等等。结果，过程模块本身可以具有模式、状态和告警行为，上述属性可以被分配到工作站，并且可以作为显示下载的一部分下载。如果需要，执行引擎 48 可以执行仿真算法，以便使用过程模块的智能过程对象提供的数据进行与过程仿真相关的质量平衡、热量平衡、流量路由、流量效率、流量优化以及经济计算，或进行其他需要的计算。更进一步，上述仿真算法可以访问来自控制策略，即关联以及下载到控制器、现场设备等的控制模块的参数，反过来，上述仿真算法可以为上述控制模块提供数据或信息。

可以理解的是，需要执行引擎 48 能使过程算法执行所有过程对象的合并，和在所有显示上配置过的链接。于是，不管相关的图形显示是否被加载，即提供当前正显示的信息给用户，仿真算法（位于过程模块之中）一般都会执行。当然，在整个过程 10 中或过程 10 定义的子集中，仿真算法可以被交叉校验。也可以理解，在任何特殊过程模块的执行过程中，执行引擎 48 可以在操作员界面上为操作员提供显示，该操作员界面基于与过程模块相关的图形显示描述过程模块内的互连对象或实体。该显示的参数、图形等等，将会被过程模块中智能元素的配置或相互连接确定。此外，在上述显示或其它显示上提供的告警或其他信息可以由智能过程对象中的方法和与特定过程模块相关联的仿真算法定义和生成。如果需要，执行引擎 48 可以将过程模块的显示提供到多于一个操作员界面，或者被配置或设置为不提供显示，即使执行引擎 48 继续执行过程流模块并因此执行与其相关联的方法、告警行为、流量算法，等等。

图 4 描述了可以使用上述元素和配置应用程序创建的示例性的过程图形显示 100。详细的，图形显示 100 描述了可以由水、酸和盐基生产白醋的加工厂的一部分。如图 4 所示，过程图形显示 100 在输入端包含 4 个流元素 102，用于定义输入盐基、酸、水和冷却水的流。盐基输入流 102 通过管道连接元素 104 并被传送到阀门 103 形式的执行器。阀门 106 的输出端通过管道连接元素 104 连接到混和器 108 的第一个输入端。根据类似的方式，酸输入流 102 首先连接到变送器元素 110，然后连接到更远些的与混合器 108 相连的阀门 112。酸输入流 102 和变送器 110，变送器 110 和阀门 112 以及阀门 112 和混合器 108 都通过管道连接元素 114 互连。

容易看出，混和器 108 的输出端通过管道和两个变送器 124 和 126 与热交换器 122 相连。冷却水输入流 102 通过阀门 128 被输入到热交换器 122，并且通过阀门 130 从热交换器中流出，以生成返回的水流元素 131。同样的，热交换器 122 的输出通过变送器元素 132 和阀门 134 输送，从而提供输出的酸流元素 136。尽管不是在所有的情况都需要，图形显示中的元素都通过管

道连接元素相互连接。

可以理解，显示框 140，可以作为显示元素自身的属性被生成，也可以变送器的形式作为单独的元素，估计属性元素或参考控制模块中功能块的元素。在图形显示 100 中描述了显示盒，以指明其相关参数，比如与不同元素相关的过程变量（PV）值，设置点（SP）值，输出值等等。额外的，如果用户将要把光标置于一些元素之上，显示 100 将会图示说明与参考元素相关的其它值。例如，将光标置于其中一个流元素之上（比如酸输出流 136），可能使得图形指明过程中该点处酸流的成分、压力、温度、密度、流速等等。显然，在图形显示 100 中显示的数值和参数可以从过程控制系统（比如从控制系统中的 AI 块）中的实际参考的变送器中传送出来，或者从仿真元素功能特性的过程模块仿真元素中传送出来。图 4 的图形显示 100 可以在过程运转期间为用户提供，并且上述过程可以制造白醋，或对即将用到的过程实施仿真操作，例如，执行设计操作或操作员训练行为。

这里描述的智能过程对象、图形显示元素和过程模块的功能可以在操作工作站 20 中起作用，并且不需要下载到工厂 10 中的控制器、现场设备，并在控制器和现场设备中配置，这就使得实现、观察和改变上述功能更为方便。进一步，与置于加工厂、控制器中相比，上述功能使得更易做出系统级的决定，因为属于系统级设备的信息一般而言对于操作工作站 20 是可用的，并且在特殊情况下对于执行引擎 48 也是可用的，尽管对于加工厂 10 中的每一个控制器和现场设备而言，上述信息并不总是可用的。然而，当这样做有益时，一些与过程模块相关的逻辑，比如图元，可以被嵌入到加工厂中的装置、设备和控制器中。可以使用智能过程对象创建完整的过程控制模块和图形显示，比如，使得执行引擎 48 可以自动检测泄露和生成带有少量用户配置行为的智能告警，从而计算和追踪工厂 10 中的流量平衡、质量平衡和损失，为工厂 10 提供更高级别的诊断并在工程设计和操作训练时仿真工厂的操作。

图 5 显示了将执行引擎 48 和过程模块和使用的图形显示，集成到具有

分布式控制策略的加工厂中一种可能的方式。如图 5 所示，由过程模块创建的或与过程模块相关的显示类定义 220，在执行引擎执行时提供显示给操作员，并且显示类定义 220 被提供给控制配置数据库和工程工具 222，所述控制配置数据库和工程工具 222 在控制策略文件中以需要的方式对这些显示类定义进行使用和组织。过程算法 224 可以在运行时间之前与这些显示类定义 220 相关联，接着显示类定义及其流算法被实例化并提供给图形显示/过程模块运行时环境 226（该环境可以通过在一个或者多个工作站中的一个或者多个执行引擎的形式来实现）。图形显示/过程模块运行时环境 226 使用下载脚本解析器 228 在运行时解析代码（即，执行及时的对象代码转化），并使用基于规则的执行引擎 230 执行流算法或其它规则，其中所述的其它规则基于为显示类提供的程序，或者基于与显示类约定的程序。在这个过程中，图形显示/过程模块运行时环境 226 可以与控制模块运行时环境 232 通讯，以提供数据或者信息到控制模块运行时环境 232，或者访问来自提供数据或者信息到控制模块运行时环境 232 的数据或其它信息，所述控制模块运行时环境 232 可在与该过程相关的控制器和现场设备中执行。当然，图形显示/过程模块运行时环境 226 与控制模块运行时环境 232 可以通过任何需要的或者是任何预先配置的通信网络来进行通信，例如图 1 所示的以太网总线 24。更进一步，其他的将这里描述的图形显示、过程模块、过程控制集成在一个标准控制系统或加工厂中的方法，也可以被使用。

如上所述，显示于图 4 中的示例性过程图形显示中的智能过程对象具有图形元素和仿真元素，并被用来描述工厂中创建的加工厂元素的在线操作和仿真操作。用于支持智能过程对象的图形元素和仿真元素创建的对象类模型框架和说明性格式会结合一个或更多个实施例进行更详细地描述。一般而言，每个智能过程对象代表一个物理设备或加工厂元素。为此，每个智能过程对象都定义图形显示元素，而上述图形显示元素可以用在描述该元素的过程图形显示中。每个智能过程对象可以定义或建立加工厂元素的仿真模型，所述仿真模型可以包括一个或多个算法、方法或其他用于提供加工厂元素离

线仿真的动行。在特定的实施例中，智能过程对象可以包括：将向操作员显示的图形显示元素的呈现定义；数据存储，其用于存储关于位置（比如，路径）的参数的、指示加工厂元素在线操作的数据的处理；方法、算法或其它在图形显示元素通过用户界面显示时作为已发生事件的结果而被执行的步骤。上述由智能过程对象调用的方法、算法或其他处理可以通过执行引擎实现。由上述实现操作生成的内容和信息可以通过过程图形显示呈现，所述内容和信息还可以由执行引擎生成，或由为了使过程图形显示呈现于一个或多个显示设备上而提供的一个或多个呈现引擎（比如：矢量图形呈现引擎）分别生成。

在具体地描述所述对象和说明性格式支持过程图形显示及其图形显示元素的定义和处理的方式之前，需要注意的是，关于智能过程对象和过程图形显示以及它们与过程模块的使用的进一步信息可以在下面的三个美国专利申请中找到：（i）于 2002 年 10 月 22 日递交的，名称为“加工厂中的智能过程模块和对象”的序列号为 10/278,469 美国专利申请，（ii）2003 年 7 月 21 日递交的，名称为“加工厂中图形显示元素、过程模块和控制模块的集成”的序列号为 10/625,381 的美国专利申请和（iii）2004 年 12 月 16 日递交的，名称为“使用智能连接元素的加工厂仿真”的序列号为 11/014,307 的美国专利申请，本发明特此专门将上述美国专利申请公开的全部内容并入本文作为参考。比如，本发明参考了序列号为 11/014，307 的美国专利文件中的图 5 到图 8，以及相应的从[0075]段到[0107]段的原文内容，其中图 5 显示了嵌套的过程图形显示，图 6 显示了与过程图形显示相关的过程模块的关联，图 7A 和图 7B 过程图形显示、过程模块和控制模块相互的通信连接以及它们的集成，图 8 显示了提供先进的控制和仿真功能的并且相互连接的过程模块和控制模块。例如，与使用显示编辑器的过程图形显示配置相关的进一步信息，可以提供给配置应用程序 38 并作为其一部分，可以在共同转让的、上述指明的国际申请文件以及也向美国临时申请文件要求了优先权，名称为“过程配置和控制环境中的图形集成”的专利文件中找到，上述申请文

件公开的全部内容并入本发明作为参考。

现在结合特定实施例描述有关在公开的加工厂用户界面方案的配置环境中配置和定义智能过程对象的方式，以及对其进行装配、编译或进行其他处理以用于运行时环境中的方式。上述实施例使用了说明性语言或标记语言以支持上述智能过程对象及其相关联的过程图形的配置、生成和操作。一般而言，由标记语言建立的格式和以及采用这种格式表述的脚本，可以用来定义和支持具有模块化的动态过程图形的加工厂用户界面（以及用户界面系统）。上述模块化的动态过程图形被数据绑定或链接到加工厂设备、过程控制元素以及它们所属的其它系统或设备上。进一步的结果是，所获得的用户界面是可扩展的并且非常灵活，因而使用户能够根据加工厂状况、设备、控制例程等创建或修改用户界面。

就像下面描述的，以标记语言表述的脚本旨在用来生成这里描述的显示的动态图形和显示元素。除了能够支持过程图形各自的结构，以标记语言格式表述的信息也可以定义该显示中描述的元素的功能性（比如：操作或者行为），包括与数据转换相关的动作、事件处理以及与上述元素相关联的智能过程对象支持的其它动作。更一般地说，以标记语言表述的信息可以包括建立过程图形显示的任何功能和操作、任何图形显示元素以及用户界面自身的任何方面和功能性的脚本或代码。

按照下面描述的对象模型、架构或框架，标记语言定义用户界面的这些图形和其他方面。还描述在准备比如运行时环境过程中，对按照对象模型和标记语言表述的信息进行处理（比如，被转换）的方式，包括比如下载脚本解析器 228 或其他作为脚本转换或处理引擎的实体所进行的任何处理步骤。

一般而言，说明性语言或标记语言用来支持生成这里显示和描述的、在配置环境和运行时环境中使用的加工厂用户界面，从而运行和维护加工厂。基于标记语言呈现各种上下文的信息，使得信息可以以一种一致的、熟悉的、用户可访问到的方式呈现和存储，而不管用户在何处想要使用这些信息。更进一步的结果，用户界面以一种灵活和可扩展的格式描述，该用户界面包括

其菜单结构、到其他文件的链接、到外部信息的链接、嵌入的和可替换的形状，等等。下文结合一些实施例描述建立上述格式的配置和运行时环境的内部标记语言，PGXML。虽然基于工业标准可扩展标记语言（XML），但可替换地，PGXML 也可以基于任何一种其他的标记语言或说明性语言。此外，如下文将更描述的，使用 PGXML 阐明的过程图形信息可能会随后在初始化、编译、下载、呈现或其他处理步骤时被转换。上述处理可以包括从用 PGXML 描述的特定信息转换到其他说明性格式，比如微软的用于用户界面的内部格式、XAML（可扩展的应用程序标记语言）、或者开放源格式、SVG（可缩放的矢量图形）。由被转换到的上述矢量图形格式（比如：XAML 或 SVG）的 PGXML 信息一般可以用于框架中，并且带有支持通过内部 PGXML 格式定义的动态图形和其他图形的完整性的扩展部分。下面将会描述示例性的过程图形框架，更一般地说，是使用 PGXML 和 XAML 脚本生成上述过程图形的过程图形架构。

基于 XML 的过程图形架构和框架可以使用一个或多个对象模型，包括，比如，由 PGXML 表示和定义的内部对象模块、由 XAML 表示和定义的对象模型，以便规定由用户界面使用和呈现的绘图形状和其他图形。XAML 格式规定了使用矢量图形命令的绘图形状。此外，就像使用微软 WinFX Avalon 工具的用户所知道的，XAML 脚本除了支持静态数据外还可以支持动态数据表示（比如，动画）。使用上述功能，过程图形元素的动态形状和属性可以与实时数值、历史数值、物理属性，比如质量-流量和成分，以及对于可变特性的其他过程或仿真数据相关联。从这种意义上说，这里描述的基于 XML 的图形是数据驱动的。

作为背景，XAML 是微软 WinFX Avalon 工具的图解形式的表示法，并且与客户端-服务器架构一起形成微软 Longhorn 操作系统版本的一个组成部分。本领域的技术人员都非常熟悉有关 XAML 和 Avalon 的对象模型和其他细节信息，关于它们的进一步信息可以在为基于 Avalon 软件的开发人员提供的微软因特网网站 msdn.microsoft.com/longhorn 找到。简而言之，XAML

支持包括直线和曲线、图像和文本的矢量图形形状。并且 XAML 支持包括矩形、圆形、椭圆和多边形的基本图像形状。如下面描述的，可以增加与加工厂和制造相关的另外的基本的、原始的或其他形状。使用 XAML 格式的用户界面可以实现像三维形态效果一样复杂的动画技术。如下所述，过程图形可以包含上述动画和其他复合效果。

XAML 格式的矢量图形的性质使得复合图形信息可以以基于文本命令的顺序存储，其中基于文本的命令可以画出不同的矢量图形形状。上述信息可以通过不同的方法进行转换，以显示特定应用程序的图形图像。由 XAML 格式提供的另一种功能是可缩放性。因为重画指令被传送到能够基于 XAML 命令实现的呈现程序中，而不是作为像素值以位图的形式传送，所以在过程图形中使用 XAML 格式允许改变或缩放图形显示元素的大小，而没有锯齿状边缘。运行时环境的执行引擎 48 可以为每个目标环境提供一个或多个呈现引擎，并且被提供的呈现引擎中至少有一个专门用于处理基于 XAML 的命令来生成这里描述的过程图形显示。

因为图像是基于矢量的，所以不管在何处提供显示，比如，在智能电话、手持设备，高端监控器还是其他显示设备上提供显示，都能生成高质量的图像。因此，过程图形，更一般而言，不管显示设备的特点如何，可能都会以一种一致的方式生成加工厂用户界面。因此，用户界面并不只存在于包括工作站的实现工具中，尽管上述工作站可能会包含在服务器、数据库或其他设备中。此外，因为使用标记语言，因而可以提供用户界面解决方案，于是，可以通过由浏览器或与该语言兼容的软件建立的因特网、网络（web）之类的通信连接来生成过程图形显示。在上述情况下，基于 XML 或 XAML 的脚本可以简单地下载到具有呈现引擎（与具有安装在其上的用户界面应用程序的整个副本不同）的远端设备上。换句话说，由过程图形显示提供的运行时环境成为具有集仿真和结合过程模块描述的其他应用功能为一体的因特网应用程序或网络应用程序。

使用下面描述的基于 XML 的对象模型和框架以及由他们产生的 XAML

脚本，过程图形显示中的每个元素都可以用动画的形式显示。接下来，Avalon框架支持用于创建高效图像的位图型滤波效应。滤波效应一般用来为配置工程师和用户提供向配置环境中的图形显示元素和文本直接加入不同效果的功能。例如，过程图形可以包括和支持下述效果，比如：调合效果、平铺效果、形状旋转或转换效果、变形、偏移、合并和特殊灯光效果。应该理解，上述或其他基于 XAML 或 Avalon 的效果可以与图中所示的示例性过程图形显示及其元素一起获得而使用。上述滤波效应可以单独应用，或与矢量图像一起使用，使得特殊的效果，比如鼠标移过（mouse-over）特性可以被分派到任何 XAML 图形对象中。

在将基于 PGXML 的描述转换为 XAML 的实施例中，由 XAML 提供的矢量图形和其他上述特征、优点和能力可以运行时在有动画或其他动态图形特征的过程图形显示及其图形显示元素中使用。

基于文本特性的标记语言，比如XML和XAML使得下列操作变得方便，配置、运行时和其他用户界面应用程序 32 通过下载的脚本存储和传送过程图形和其他数据。因为过程图形元素可以以文本的形式存储，因此用户界面应用程序 32 一般可以包括查询功能，以支持在图像内对文本的搜索。

这里公开的基于 XML 的对象模型，以及因而建立的框架或架构，除了有先进的图形还有大量的优点。例如，使用基于 XML 的描述可以生成很小的文件，从而使得快速下载变为可能。另外，对象模型的可扩展特性可以生成重要的预先创建的图形显示元素库，并且预先生成的图形显示元素可以在随后补充。为此，该框架可以包括具有组、复合、类和模板的对象模块，更多的细节将在下文描述。下文还将描述，配置工程师或其他用户可以根据与装置、设备或其他实体相关的外部信息生成图形显示元素（比如：形状、复合、类和模板）。例如，用于上述实体的图形显示元素的信息，以及整个过程图形显示，可以由第三方提供者提供，比如 INtools。

这里公开的基于 XML 的架构也支持在运行时使用和显示数据，所述数据来源于分布式计算环境中若干不同数据源。例如，像上面提到的，通过过

程图形显示的数据可以在相关联的过程控制系统（比如：DeltaV 历史记录器）、基于 OPC 的系统、任何 XML 文件或任何其他的源或系统中生成。为此，过程图形架构包括摘录表或数据源层以建立数据到基于 XAML 图形的关系。例如，若用于过程图形元素的 XAML 脚本描述了以像素形式表示数据（比如，填充一个条形图）的图像，位于加工厂中的仪器将会提供以华氏度为单位的实时数据。像下面将被描述的，在呈现过程图形元素之前，可识别数据源，并且连同转换、缩放等需要的代码一起，建立和提供图形元素和数据之间的逻辑关系。以这种方式，所得到的过程图形用户界面将以一种容易理解和不会被误解的方式提供数据。

在上面描述的一套界面应用程序和数据结构 32 中实现和提供了基于 XML 的功能。具体而言，可以在包含图形显示配置应用程序 38 的配置环境中创建过程图形，并且图形显示配置应用程序 38 可以具有用于创建比如形状、复合、类、模板和动态演示之类的图形显示元素的若干操作模式，上述每种图形显示元素都会在下文详细描述。配置应用程序 38 一般提供图形编辑器，其具有若干集成工具（比如，GUI 和其他工具）用于执行数个任务，所述的任务包括，例如创建脚本、将图形绑定到数据源，等等。使用上述工具的步骤将在下文描述。在配置并转换到 XAML 格式后，运行时环境通过前面提到的执行引擎 48 进行基于 XAML 命令实现的操作。更具体而言，运行时环境可以包括提供用于执行和呈现过程图形显示及其元素的后端支持的界面应用程序，上述过程图形显示及其元素位于由运行时工作区定义的框架内。

现在参见图 6－11，其中用相同的附图标记来表示相同的元素，这里，结合示例性界面或环境 300 来描述过程图形显示的配置和图形显示元素的创建，该示例性界面或环境 300 可由配置应用程序 38 或通过其它能够呈现基于 XML 的过程图形的设备来生成。在此配置过程期间，所创建的过程图形显示及其各个图形显示元素在内部以 PGMXL 格式进行描述。以下进一步详述关于 PGXML 格式支持和定义这些显示和元素的方法。

界面 300 一般为配置工程师或其它用户提供图形编辑器，以便为设备操作、设计、维护、管理以及为设备定义的其它角色的运行时环境创建和定义过程图形。图 6－11 渐进地描述了使用界面 300 创建的过程图形显示，该界面 300 可以包含若干区域和面板，专用于定义过程图形显示及其图形显示元素的属性、参数或其它的方面。例如，界面 300 可以包含一个或多个组织面板 302、304，用来为图形显示提供上下文或总览信息。面板 302 可以显示过程图形显示的名称及其在设备内的相关位置（例如，其与其它过程图形显示的关系）。面板 302 还可以列出并排列已由工厂区域或根据任一其它预期标准生成的各种过程图形显示。面板 304 可以识别每个已被加入至工作区或界面 300 的画布区 306 中的图形显示元素的名称。如以上所述，每个元素的通用版本可以被选中并从选项板区 308 被拖拽至画布区 306。选项板区 306 可以包含若干对应于不同元素类型的子区域以表示不同的设备、仪器、或其它要由所创建的过程图形显示所表示的加工厂元素。例如，选项板区 306 可以包含执行器子区（图 6）、处理设备子区（图 7）、测量设备子区（图 8）和专用（或定制）子区（图 9）。编辑器界面 300 还包括一个或多个工具条 310、312，它们显示了多种用于图形选择、创建、编辑或其它处理的编辑工具。

当把图形显示元素从选项板区拖至画布区 306 中时，用户可以通过诸如在某一连接点按住鼠标左键再移动光标至目标连接物的方式将元素连接在一起。然后，所得到的连接可以被编辑器 300 在两个元素之间以自动选路的方式自动画出。或者，每当鼠标键在光标到达规定的连接点之前被释放时，可以自动产生连接的转弯或弯头。进一步，连接类型（如，管道、传送带或导管）可以通过定义连接点的性质而被自动设定。在图 6－9 所示的示例中，管道 314 被创建于泵 316 的下游连接点处。当某一连接支持多种连接类型时，接着，通过例如选择工具条图标 317 可以定义缺省的连接类型。

各个独立的位于画布区 306 的图形显示元素可通过在其图形表示的上面或附近点击鼠标而被选中。当该元素被选中时，接着与该显示图形元素相

关联的可配置参数列表可由参数面板 318 来显示。如图 6 所示，泵 316 的选择显示出元素的图形方面（aspect）被给定，如用来描述泵为开（“ON_CLR”）和关（“OFF_CLR”）的颜色。图 8 示出了由图形显示元素表示的变送器 320，该图形显示元素具有指定将一个或多个功能块参考作为数据源的信息，如针对 AI_REF 参数的 F1521/A13。这些参考参数之一被选中后，就可以提供允许用户定义到达期望数据源的路径的对话，比如，控制模块/块/参数。关于指定参数、参考参数、属性以及图形显示元素的其它方面的进一步信息将在下面阐述。

具体参照图 7，使用处理设备选项板可以将常用的处理元素，如储罐、搅拌器、热交换器添加至显示中。当处理元素，如储罐 322 被添加至某一显示时，其可被扩展到与该显示相称的大小。选择像储罐 322 这样的元素还通过面板 318 为该元素提供了指定输入和输出流的数量或连接点的数量的机会（参见，如，入口流（In Streams）、出口流(Out Streams)）。通过面板 318 中示出的对应参数，还可以修改连接点的位置（参见，如 In1 Position，Out1 Position）。

图 9 所示的定制设备选项板可以提供给不太常用的处理设备、流元素和到其它过程图形显示的参考（即，显示参考）。例如，可以标识出流的始发点和结束点。内部的流参考允许通过所示的面板 318 名命流的源。外部的流参考可以包括用于标识在另一显示中定义的相关流的参数。

图 9 还示出了专用图形显示元素的另一示例，即通用的或模板的显示参考 330，其位于选项板区 308 中，用来支持标识另一过程图形显示，该另一过程图形显示用于表示在当前显示之内且通过隐式操作而带有所有与其相关的功能和下层图形显示元素的被参考的显示。图 10 示出了，在显示参考模板 330 已经被选中并被拖至画布区以便在过程图形显示 350 之内放置和连接之后，一般用 350 指示的简化的过程图形显示的一部分。此时，配置工程师已经修改该显示参考模板 330 来标识蒸发器显示 352。为此，修改或指定面板 318 中的“DISPLAY_REF”参数，以便通过名称，在某些情况下通过路

径，来标识该过程图形显示。

图 11 提供了过程图形显示的定制化内容层的示例，具体地说，该过程图形显示的泵 360 具有估计属性元素 362（当用户具有该元素时示出），比如工程访问权限。如在同时提交的名为“针对具有集成的仿真功能的加工厂用户界面系统的定制化的过程图形显示层（Customized Process Graphic Display Layers For A Process Plant User Interface System Having Integrated Simulation）”的国际专利申请中更详细的描述，因此以其整体作为参考被特别结合进来，用户可以选择或被提供一个或多个与用户概述的特征相当的过程图形显示的内容层，其中该用户概述的特征可指定用户的访问权限。在图 11 的例子中，提供给用户对通过估计属性元素 362 计算泵 360 的排放压力的仿真信息的访问。

以上所述的图形显示元素（一般来说是智能过程对象）可以由配置工程师或其它用户在任何时候预先定义或创建。配置应用程序 38 创建或定义这些图形显示元素的方式利用了对象模型架构和各自的 PGXML 描述。每一元素可以具有一组对应于过程变量、常量或其它外部值的固有属性。该元素可定义若干可视化的表示，其可包括：动态行为，如颜色变化或动画。从过程控制系统接收的值的变化可以提示显示中的元素的动态行为来改变它们的显像。

如以上所述，当前提交的申请中，两种工具可以用来配置这些过程图形。第一种工具可以是用来配置过程图形元素的元素编辑器，从而将定制元素添加至若干预先配置的可被包含在加工厂用户界面系统中的过程图形元素。第二工具可以是用于从图形显示元素库建立过程图形显示的显示编辑器。

如以上所述，关于该两种编辑器或编辑工具的操作以及为元素定义、提供或分配属性和显像的方式进一步上述的同时提交的申请中详述。然而，在一个示例中，用户可以选择通过对话框和表达式编辑器为所选的动画加入动画和输入配置信息，以向某元素加入动态行为。为此，该表达式可以涉及该元素的固有属性。例如，为了当泵启动时指定泵的颜色从蓝变为绿，用户将

在指定采用何种颜色时应用具有涉及泵IsOn属性的表达式的颜色变化动画。

在用户可以画出原始图形形状（如以下进一步详述）的意义上，过程图形显示的创建与图形显示元素的创建相似。该配置工程师或其它用户还可从之前定义的复合形状库中选择复合形状。这些之前定义的形状的可用性可以显示于以上所述的面板302、304。

基于对象模型框架，以上提及的原始形状和复合形状分别基于形状类和复合形状类，以便通过类的实例来使它们能够被重用。这些对象类的使用为每个复合形状实例提供了方便的参数或其它更新，该更新通过对各自形状类的修改而自动产生。另外，使用前述编辑工具来定义附加复合形状类的能力为该模块化方案定义过程图形显示提供了灵活性。

形状对象类是对象模块的图形部分的基准或基础，各个形状对象类用于定义各自的原始矢量图形实体。一般来讲，根据所述对象模块架构、图形和其它用于用PGXML来表述的各个呈现定义中描述的对象的配置信息，各智能处理过程对象以及相应的各图形显示元素的图形描绘由一组图形对象或由其构成的复合对象组成。在此种方式下，这里描述和显示的过程图形对象可由原始图形构成，该原始图形由各自的排列为图形生成块的图形对象定义。如下所述，在为呈现图形做准备时，用PGXML描述的各图形对象随后被转换为对应的XAML图形对象（或其它图形格式，如SVG）。

各形状类可以具有固定的属性或参数集合。改变形状类实例的属性值，即形状用法（usage），可使得形状的图形描绘发生变化。属性可以为诸如大小、位置、颜色、线条粗细或透明度指定数值。可选择地，或者作为补充地，形状类的某些属性或参数可以改变形状用法的行为。示例性行为包括：激活或禁止数据条目，以及隐藏形状或其任一部分。参数或属性值可以被赋予动态值或固定值。对于动态值而言，数值格式程序对象（以下进一步所述）被创建来建立和维护通过定义引用和/或路径串确定数值的一般方式。形状的属性或参数可以因此从其它形状用法的属性中获取。该形状可以进一步定义与将要根据这些数值执行的计算或格式转换相关的动态行为。

形状类也可以定义触发执行脚本命令（也可以结合形状类来表述），或者，在某些情况下，用户界面中被全局定义的命令（比如，用来显示用户界面菜单的右击选择）的事件。事件可以涉及形状的选择或其它用户动作，或者，作为替换，涉及经过编程的对属性值变化的反应。

像图形生成块一样，形状类可以为过程图形显示以及复合形状类形成基础，或者基本构造。更具体来说，形状对象可以在显示或复合形状内使用，它的一个实例可以作为图形元素被实例化。在对象模型框架中定义的形状对象包括如下一些原始形状：矩形、圆形、椭圆、线（单线和多线）、多边形，以及路径。另外的形状对象的类型包括：容器（container）形状，比如符号。符号可用于在过程形状层次结构中表示点或图标，该过程形状层次结构可用于描绘不同图形显示元素和过程图形显示之间的层次关系。例如，配置和运行时环境可以为用户提供在面板302或304中使用这些符号来显示该层次结构的选择。这些符号接下来可以为该层次结构中节点的特性（例如显示、显示元素、元素属性等等）提供方便的可视化指示。例如，显示的形状图标可以包括工厂的微缩描绘，而用于元素的属性或特性的形状图标可以为小圆圈。用于容器（比如组和复合）的形状图标可以是一文件夹，而该层次结构中的一组项目可以显示为一对重叠的正方形。其它可用形状图标包括用来表示位图的带有画笔的通用图片，以及带有被覆盖链接的通用显示，其用来表示与外部形状的链接，比如文档。另外的形状对象的类型包括文本形状的类型，比如文本和文本路径（即可以理解运行时路径以在运行时环境中建立数据位置的图形元素）。更进一步，形状对象可以是参考形状，比如到图像的链接或参考、HTTP或其它链接、或许多用户界面形状中的任何一个，如按钮、复选框、滑块等等。

一个或多个属性可以为每个形状对象或形状对象组赋值。示例性的属性包括工具提示（即当鼠标指针或其它选择机构悬浮于形状上时出现的、与形状对象相关的小方框，其用来为形状对象提供描述或其它信息），上下文或其它菜单（比如，下拉菜单，可以从下拉菜单中启动帮助文件），拖动/放

下行为定义（比如，关于标签分配的详细内容），以及缩放和跨度定义。

形状对象、形状属性和形状对象组也可以具有与之相关联的行为。比如，形状对象可以具有基于动作或提醒的行为来响应从数据源收到的数据（如下所述）。条件行为（比如 if-then-else，switch 等等）和其它行为也可以通过定制脚本来定义。这些行为可以以与形状对象关联表述的脚本形式通过代码来定义。并且可以使用任何期望的编程语言，比如 C#。接着可以在运行时执行通过脚本表述的指令来更新形状对象或复合形状对象的属性或参数，因而可以修改过程图形显示的外观。

除了增加属性或行为，配置应用程序 38 一般为配置工程师或其它用户提供若干执行针对形状对象或形状组的动作的能力。例如，配置程序 38 可以支持图形用户界面环境一般提供的生成、复制/粘贴、移动、替换、缩放和其它功能。其它静态形状，或成组的形状，也可以在预知下层 XAML 框架能力的情况下被转换为动态形状。例如，用户可以定义一传送带组，将该组转换为一个动态组，公开或分配与传送带的动画相关的参数，并增加脚本来定义传送带的移动。配置程序 38 也可以支持增加收听者行为，其用来注册事件并为事件（如按下按钮）的发生指定动作。另外的行为是装载 URL 或 XAML 脚本、或任何一般的诸如打印和视图改变（如向前移或向后移，向前翻或向后翻等等）之类的用户界面命令，该行为可以结合形状对象或形状组一起进行。

可以用配置应用程序38提供的选择工具或其它工具产生或者关联形状组对象。选择工具也可以用于比如定义那些要设为动态的形状，或者更一般地用于选择一个或者更多个形状来分配任何其它功能性。结合将对象绑定至输入或输出数据源，选择工具可以用于比如公开或查看形状对象的属性。关于数据绑定的信息在下面进一步。

形状对象可以被组合成定制的或配置的成组形状或形状组对象，并可以作为库的一部分保存下来，以便以后单独地或作为复合形状的一部分用于另一过程图形显示中。在这种情况下，以前创建的形状组或复合可以被选中并

从用户界面的选项板区（如前述的库或模板区 65 或选项板区 308）拖入画布区（如配置区 65 或区 306），该画布区可以作为创建新的过程图形显示或图形显示元素（如新的形状组或复合结构）的工作区。

上述形状对象为形状类的实例化，这些形状类形成对象模型框架的基础。当将形状对象组合成复合形状时，也就定义了对应的复合形状类。示例性的用户界面系统可以包括具有任意数量（如 100）的预定义的复合形状类的库，以支持特定设备的建模。已公开的用户界面系统具有的灵活的、模块式特征支持对库进行修改，比如像预期的一样定义新的复合形状类。用户界面系统的模块化特点允许这些形状类和复合形状类应用和重用于生成许多用于需要综合表示加工厂的显示（如 1000）。配置工程师或系统设计者对复合形状类的重用可以方便地由此以嵌套布置的方式参与构造和生成多个更复杂的复合形状，因而产生另一个复合形状类以将来用于任何数量的显示的实例化。正如这里描述的，随后的对复合形状类库的参数、属性、行为等等的修改自动更新所有的具有被修改过的类的图形显示元素（如实例）的显示。为此，每个复合形状类在它的 PGXML 描述中或通过一个相关联的列表、表或其它标识，可以具有被标识类的每个实例，以便自动对改变进行传播，并同时对复合形状类对象进行保存、存储或其它记录。

现在参看图 12，用户界面系统使用的对象模型框架也可以定义若干被用于配置和运行时环境的其它方面的类。这些类比如可以用于为显示及其元素定义过程图形显示、复合形状和数据源的各方面。具体来说，图 12 仅示出了专用于指定数据转换方法的对象模型框架的小的、示例性部分。如前所述，智能过程对象可以用多种方式动态地显示数据的值，包括，比如，一个棒图或颜色的改变。在配置图形显示元素时，无论是特定的实例或是模板，值转换属性都可以被添加到其上。在这个示例中，数据可以根据五种不同的转换方法中的一种来进行转换，这些转换方法由相应的子类（ValueConverterBase 类 400、ColorLookup 类 402、RangeConverter 类 404、FormatterBase 类 406、CustomConverter 类 408 以及 BooleanConverter 类 410）。每个各自的类可以

有如果在实例化时指定的参数或属性(参看 RangeConverter 类 404 的 InRange 和 OutRange 参数)。ColorLookup 类 402 可以与智能过程对象行为相对应，该智能过程对象行为使得对象的颜色能根据当前的数据值改变。例如，储罐可以用由颜色描述的液体来指示液位正常（如绿色），高于危险值（如红）或低于不正常值（如黄色）。可以通过属于 ColorLookup 类 402 的子类，即 ColorLookupEntry 类 412，来表示不同的液位和其它需要实现这些颜色改变的信息说明。

继续参考图 12，FormatterBase 类 406 有若干子类，它们用来指定满足不同数据类型需要的不同类型的可能的数据格式化方法。示例包括 NumberFormatter 类 414，DateTimeFormatter 类 416 以及 GeneralFormatter 类 418，与此相关的是，把数据转换成用于显示的字符串，如块 419 所注释的。如图 12 所示的类图通过连接两个类的线可以指示类之间的关系，如 CustomConverter 类 408 和 DeltaV.ProcessGraphics.Base.Scripts.ScriptBase 类 420 之间的关系，它可以为用户提供机会去指定定制脚本或其它命令来实现定制的指令，在这里该指令用于格式化数据。

图 12 也在块 422 中注释了上述实例化的对象的定义可以存储在动态链接库（DLL）文件中，创建该 DLL 文件为将过程图形显示下载到用于实现运行时环境的工作站做准备。当过程图形显示被呈现时，DLL 文件可以表述格式化的定义和其它要执行的方法。为此，DLL 文件可以包括由比如 C#代码编译的指令。而 C#或其它代码可以在过程图形信息被转换为 XAML 脚本的时候下生成。

图 13 示出了对象模型框架的另一部分，该对象模型框架与指向可以为图形显示元素指定的示例性行为或动作的类图相对应。每个可用的行为或动作通过脚本指令来指定，并且对象模型提供了以面向对象的方式而不是作为未经组织的特定实例代码的块来指定指令的框架。以这种方式，指令可以被方便地重用或定制化。一个通过这些类的对象定义的示例性行为可以涉及用户按下或启动图形显示元素的按钮，比如提供具有开/关转换的开关选项的面

板图形。面板元素可以接着使用 IScript 类 430 来一般地标识该行为，接下来用一个具有 ScriptArg 类 434 和 ScriptAttribute 类 435 类的 ScriptBase 子类 432 来分别标识若干变元或属性以用于要定义的脚本指令中。反过来，ScriptBase 子类 432 有若干子类 436-439，其指定若干预定义的动作（例如示出—显示，如面板盘），以及子类 440，其指定定制脚本。指定预定义的动作的一个或更多个子类可以有一个或更多个相应的子类，用来定义参数或其它要执行的动作的方面（参看比如 GraphicParameter 子类 442）。

图 14 也示出了示例性对象模型框架的另一部分。这部分被侧重于定义串行化对象，在一些实施方式中，串行化对象可以被用于支持从配置环境到运行时环境的移动。具体来说，许多类被定义来指定由用户界面系统执行的动作来使其它定义过程图形显示的对象串行化（或者在类图的其它部分是串并转换）。实现时，这些类可以形成配置应用程序 38 的一部分，用于将在编辑器中创建的图形结构的表示转换为 PGXML 脚本（串行化），以及形成为转换引擎、程序或可以处理 PGXML 脚本（串并转换）的其它工具的一部分。在任何一情况下，图 14 的示例性框架和对象仅表示一种机制或技术，其用来产生或以另外的方式处理 PGXML 脚本。更一般地说，例如，产生的 PGXML 脚本在配置过程期间被转换为一个或多个文件，以便在运行时期间生成（呈现）显示时使用，或者，可替换地，用于产生其它显示呈现期间使用的文件（例如编译的文件）。关于转换过程和此处产生的文件的细节在下面进一步详述，但一般说来，PGXML 提供中间的、初始的（或基本的）格式来支持使用若干不同的呈现引擎和图形对象框架中的任何一个。

被串行化的对象指定或包括基于 XML 的内容，该内容用来描述过程图形显示的各方面和他们的组成图形显示元素。换句话说，可以按照图 14 所示的类指定的处理步骤来解析通过 PGXML 语言描述的显示和显示元素的定义。

在图 14 的示例性类图中，指定了两种不同类型的串行器功能类：PGLibSerializer 类 450 和 PGSerializer 类 452，其中每个都是 SerializerBase

类 454 的子类，该 SerializerBase 类 454 一般地标识要实现的串行化操作的类型（即串行化（Serialize）、串并转换（Deserialize）、OnError）。也包括 PGParser 类 456 来定义实现 XML 脚本片断或其它保存为 XML 脚本的脚本片断的串行转换的对象。

根据 PGLibSerializer452 类采取的串行化处理步骤进一步既可以通过在 XMLBlobHelper 类 458 中建立的属性来指定，也可以通过 DisplaySerializer 子类 460 和 CompositeShapeSerializer 子类 462 来指定。

图 14 中指定的类和子类仅代表处理步骤的子集，这些子集可以用来为下载和执行运行时环境下的显示做准备。而且，应当理解的是，通过图 14 指定的对象模型和处理技术实质上是示例，并且 PGXML 脚本可以用本领域的技术人员所知道的其它处理基于 XML 的描述和脚本的方式进行处理。

现在参看图 15，在配置环境中创建的过程图形显示 500 提供了为在运行时环境中执行而被转换或处理之前的部件的示例性视图。配置之后，显示 500 包括呈现定义 502，该呈现定义 502 包括要呈现的图形的 PGXML 描述。由于这部分 PGXML 脚本不涉及显示 500 的功能性，所以呈现定义 502 与显示 500 中对配置数据库 28 来说未知的部分相对应。许多定义过程图形显示或其元素的数据可以不透明地存储于配置数据库 28 内。在这种情况下，数据的特征和细节不为配置数据库 28 所知或者其索引在数据所属的图形显示元素的通用标识之外。这样的数据通常可以从元素定义的剩余部分中用一种离散或者分离的方式来定义图形显示元素的图形部分。

显示 500 的其它部分一般存储在配置数据库 28 中，以便支持在完成配置后通过用户执行的重命名步骤来进行跟踪。具体来说，这部分可以包括属性参考表 504，它将呈现定义 502 中的属性参考和用法属性或参数联系起来。图 15 的一般示例用于具有占位符###.##的参考，这里用法参数被标识为 DVSYS:LIC-101/PID1/PV.CV。参考表 504 除了图 15 中的动态属性外，还可以进一步包括对脚本和事件的说明。如下面更加详细描述的，接下来将用法的属性或参数与数据绑定表或列表 506 中的数据源联系起来。可以通过数据

路径或位置信息，如 FIC-109/PID2/OUT.CV 来指定数据源，在本示例中，该信息指定要从过程控制系统接收的信息的数据位置（虽然可以使用网络 10 内外的其它数据源）。数据绑定表 506 可以为显示 500 提供所有数据源参考（data source reference）的列表。数据源可以通过别名来标识，这些别名由配置应用程序 38 来解析。如上所述，显示 500 可以进一步包括一个或多个数值格式程序对象（未示出），每个具有标记化表达式的数值格式程序对象包含与其它对象的关联和用于格式化要显示内容的属性参考。该显示也可以包括数据源参考以及对用法的复合形状类的参考，其中该数据源参考与数据库对象建立另外的联系。

显示 500 是简化的显示，这种情况下，一般若干显示的不同元素将要求专用的属性参考表和数据源绑定列表。为了开始图解通过组合的方法定义显示来解决潜在复杂性的问题，图 16 示出了可以用在更典型的显示中的示例性复合形状类 510 的部件。当然复合形状类 510 的可以用于产生更复杂的复合形状类，它的用法接下来可以参与创建一显示。如上面所描述以及图 16 示意性所示，复合形状类 510 的组件可以由形状用法、Line1、Rect1 以及 Rect2 产生。复合形状类 510 包括呈现定义 512，它分别从属用于线和矩形形状用法的形状类 514 和 516。一般来说，形状类 514 和 516 以及复合形状类 510，可以有许多属性和事件。当属性为动态时，用于数据的支持动态图形的参考路径可以通过与形状的属性的关联被解析，通过别名被解析，或通过应用脚本被解析。无论哪种情况，内部用法的属性可以被作为参数公开，这些参数由配置应用程序中显示编辑器的用户来配置（例如参看图 6-9 的面板 318），以便属性可以在复合形状类被用于一显示或另一复合形状类时被配置。

和显示 500 类似，复合形状类 510 可以被分成呈现定义 512 和一个或更多个参考表 518-520。虽然在呈现定义 512 中表述的细节对配置数据库 28 来说可以是未知的，但被跟踪的项可以被拖出定义复合形状类的 PGXML 脚本，以成为一个参考表 518-520 中的条目。参考表中的每个条目可以在配置

数据库 38 中有一个相关联的数值格式程序条目。当复合形状用法由复合形状类产生时，数值格式程序可以包含对属性和参数的赋值，这些属性或参数可以包含特定的数据源参考或留作以后绑定使用。

如上所述，复合形状类的复合形状类用法的名称可以与该类相关联地存储，这样对类的修改被转给或传播给每个用法。类似地，图形参数用法可以与图形参数的定义相关联地存储。数值格式程序也可以包括用于参数或属性的脚本和事件。这些数据项可以存储在参考表或其它类似参考表 518-520 的表中。

复合形状类 510 也可以包括表 522，表 522 定义由比如形状类用法的属性产生的参数。具体来说，表 522 与别名（即 STUFF）相关联来解析所有参考它的动态属性。两个其它参数也可以定义为对类公开，即 ALARM 和 FLOW_RATE，其中 ALARM 被绑定至用法 LINE1 的 BGCOLOR 属性，FLOW_RATE 被绑定至用法 RECT1 和 RECT2 的宽度属性。当复合形状被置于一显示中时，产生复合形状类的用法和实例的方式可以和在如图 16 中所示的结合复合形状类 510 产生形状类用法有非常类似。配置工程师接下来可以为表 512 中标识的参数提供固定的或动态的数值。

如上简要提到的，复合形状类也可以包含链接的或嵌入的其它复合形状类的用法。这种嵌套的复合形状可用在几个形状被组合在一起形成类属复合形状类的情况下。所述用法的属性或者复合形状类用法的图形参数可以部分地与指定路径中的一个或多个别名绑定。结果，为每个唯一的别名创建新的参数，因而允许用户创建一个专用的但又可重用的复合形状类，当该复合形状类被置于一显示中时，它需要最小的配置。以这种方式，更多的复杂的复合形状类可以从较简单的复合形状类创建。作为例子，可以为限位指示器创建复合形状类，该指示器可以涉及对三角对象的描绘，该三角对象有一个角，该角被定位用来指向并因此标识棒图的水平。因此，一个或多个限位指示器复合形状类的实例可因此用于液位指示器复合形状类来标识储罐中的各种液位。

如果不是因为每个对象类都有自动更新每个各自的用法和实例的能力，复合形状类的嵌套一般将是麻烦的。也就是说，如上所述，任何对对象类，如复合图形类或显示类的配置的改变，会自动地通过用户界面系统进行传播，使得类的每个实例能够反映配置的改变。

上述对象类的灵活性可以与由运行时环境并且在某种期望的情况下由配置环境所提供的配置控制的方面（aspect）相结合。如下面进一步结合某些实施例描述的，一般可以限制用户在运行时环境中执行对复合形状类的改变。这种对类的控制通过仅允许用户通过改变用法的参数的值或定义对类的特定用法的外表进行修改来实现。这样，用户仅可以被允许为当前显示的特定的复合形状类用法或实例而不是为下层的图形结构改变数据源。

图 17 为数据库存储实例图的图解，一般地以 550 表示，该图解标识了名为 Display1 的示例性过程图形显示和它的附属对象之间的关系，用于所有这些对象的数据可以存储在配置数据库 28 中。初始情况下，存在用于该显示本身的对象 552，并且该对象可以被存储在配置数据库 28 中。对象 552 可以支持显示类的创建，如果需要的话，它可以用于创建更复杂的显示。该显示包含和使用泵的复合形状类的元素用法或实例 554。元素用法 554 被称为 Pump1 并且可以进一步由元素定义 556 定义，也可以分别用于电动泵和手压泵的显像 558 和 560。以这种方法，显示 552 的实例可以根据需要由配置工程师描述泵的两个可用的显像中的一个。泵复合形状 556 有两个内部属性，该属性可用作要配置的参数，即 IsOn 参数对象类或定义 562 以及 Speed 参数对象类或定义 564。用法 554 利用相应参数定义 562 和 564 的实例或用法 566 或 568 来支持泵用法 554 的配置。为进一步通过显示 552 来支持泵用法 554 的配置，显示 552 包括两个属性 570 和 572，它们分别与泵参数用法 566 或 568 相链接。在这个特殊的例子中，显示 552 的这些属性 570 和 572 已经由比如配置工程师通过面板 318 来指定。具体地说，显示属性 570 已经指定 IsOn 参数绑定至具有路径或位置“Mod1/...”（方便起见其全名被截短）的数据源 574。同样，显示属性 572 已经指定 Speed 参数绑定至具有路径或

位置“Mod1/...”（方便起见其全名被截短）的数据源 576。以这种方式，泵的固有属性被绑定至控制运行时变量，其值将在显示 552 的运行时环境的生成中显示。

如前面的示例所示，参数可以为一显示组成用户指定的配置项，该显示可以在配置显示实例期间，或者甚至在运行时期间例如在可以改变参数来修改仿真的离线情况下，在上述编辑器中的一个中得到。可以替换的是，图形方面和图形显示元素的其它属性可以在配置环境中指定作为对象类（如复合形状类）的属性，这样可以完整地定义对象，而无需为参数作任何进一步的说明来支持例如在运行时期间的动态行为。例如，可以通过类的内部属性可以完全地建立用于静态元素的复合形状类的属性，如有静态颜色的静态形状。然而，当这样的属性不是静态时，图形显示元素的 PGXML 定义通过编辑器例如在配置期间定义对将在以后建立的参数的参考。这样，对象类属性可以使图形显示参数支持高级的动态图形。

图 18 简化描述了根据某些实施方式，在配置环境期间产生的配置信息在在运行时环境中执行的准备过程中被处理的方式。一般说来，若干集合可以从显示实例或用法中创建，其中的配置信息可以被表述为 PGXML 呈现定义。从每个显示的呈现定义，可以产生显示图形类集合 600。集合 600 一般包括并定义图形方面（aspect）和非图形方面。例如在一个使用微软 Avalon 矢量图形的实施方式中，呈现定义的图形方面被转换为 XAML 脚本，而与例如数据转换和事件处理相关的非图形方面可以用代码隐藏指令（如 C#码）来表述。这些部件接着可以被编译为显示图形类集合 600。其它实施方式可以将 PGXML 定义转换为其它图形格式，如 SVG。

如上所述，显示可以包含复合形状类和形状类的实例。虽然这些用法可以被转换为 Avalon 控件并且具有它们自己的集合（如下所述），但绑定至过程控制数据的显示、形状和复合图形用法的属性可以链接或绑定至由单独产生的集合 602 定义的属性，集合 602 可以被称为数据源绑定类。在某些实施方式中，显示图形类集合 600 和数据源绑定类集合 602 可以合并为单个的

集合。数据源绑定类集合 602 可以包含对所有由显示实例暗指的数据源参考（如数据源 603 图示）产生的绑定的说明，以及所有与显示类相对应的链接。在这种情景下，集合 602 和数据源 603 包括所有的用于该显示及其组成复合形状和其它形状的数据源参考。

用于显示的复合形状的每个 PGXML 呈现定义可以被处理为分离的集合 604 和 606。在某些实施方式中，被链接或通过别的方式被关联的复合形状类的实例可以组成其专用于如集合 604 的单个集合。类似地，可通过将所有嵌套的复合形状类处理成单个独立的集合（如集合 606），来减少产生的集合总数。根据复合形状涉及的嵌套特点，显示集合 600 和集合 606 可以进行组合以便优化。

用于显示的形状类被处理成各自的集合，如形状集合 608，与生成的显示和复合形状类集合分离并在其外部。在预先定义形状类以避免不得不生成和下载它们的情况下，形状类集合可以预先生成。也就是说，当形状被预先安装在系统中时，它们各自的集合可以从具有配置应用程序的工作站装载、或者从配置数据库 28 下载至每个工作站或将生成具有这些形状的显示的描绘的其它显示设备中。虽然某些实施方式可以提供灵活地增加（即生成）新的形状类，但是当其它用于一显示的集合被下载时，所得到的形状集合可以被增加，因此避免在操作员工作站或其它显示设备上进行重新安装。形状集合 608 可以从各自的呈现定义中生成，这些呈现定义由 PGXML 表述，或可替换地以例如 C#来表述的代码指令来表述。

图 19 示出了涉及将 PGXML 配置信息转换成控件集合（如那些结合图 18 讨论的集合）的处理步骤，以及将这样的集合转换成用于部署运行时实现的文件的编译和其它处理。这样的处理一般在元素或显示配置完成之后进行，并且得到的数据被存储在配置数据库 28 中。一般说来，如图 19 所示的处理和部署步骤可以由转换引擎或其它工具来执行，其中，其它工具为由一个或多个计算机、处理器或其它在网络 10 中能够完成这样的处理的设备来实现。在一个实施方式中，由转换引擎或工具采取的处理步骤主要由 Pro+

工作站执行，其可以但并不是必要地为配置应用程序 38 提供用户界面。然而，图 19 的示例性实施方式提供了一种可能的两个涉及的工作站之间的职责分工，其中分隔线 610 一般指明与配置工作站（如用于和 Emerson 过程管理的 DeltaV 安装程序一起使用的 Pro+工作站）和操作员站相关联的各自的任务和文件位置。

如图 19 所示的过程图形部署策略步骤基于 Microsoft Avalon (XAML)来实现，虽然，在其它实施方式中，其它图形配置也可以使用。例如，为了获得 XAML 的可伸缩的矢量图形优点，可使用 SVG 格式，在这种情况下，由于 SVG 脚本被处理和执行的方式，使得部署步骤将和图 19 所示的类似但并不相同。

如上所述，过程图形显示和其图形显示元素（即形状、复合形状和附属显示）利用配置应用程序 38 和具体来说一个或多个图形编辑器或编辑工具来生成和配置。显示和元素通过上述对象类来描述，这样每个元素被各自的元素定义 612 和元素用法描述 614 来定义，并且每个显示由各自的显示定义 616 来定义。每个定义或描述可以由 PGXML 脚本（特别是定义 612 和 616）来表述，并因而包括要转换为 XAML 图形格式的呈现定义。为此，每个元素定义 612 被转换为 XAML 文件 618（如 control.xaml），该 XAML 文件 618 列出 Avalon 控件，所述 Avalon 控件用于呈现由元素类定义的图形结构。类似地，一个或多个控件集合文件 620 被从元素用法定义 614 中得出或生成。应该理解的是，数据绑定或其它与元素用法描述一起被存储的信息可以在参考表或列表中列出，如上所述，这样的信息也可以或可替换地用 PGXML 描述来表述。

继续参考图 19，显示定义 616 的图形部分也可以转换成 XAML 脚本 622，其可以在分离的文件（如 display.xaml）中列出。显示定义 616 的其它部分可以被处理来生成一个或多个附加的包含涉及实现显示的动态方面的指令的代码隐藏文件 624，包括指定基础数据源和产生这些方面（aspect）的事件。具体来说，代码隐藏文件 624 可以包括标识每个用在一显示中的参

考的定制数据源列表。该被参考元素的列表可以用于决定下载哪个用于该显示的控件集合。

接着可以编译 Avalon 控件和其它 XAML 脚本文件，如显示文件 622，来支持运行时环境。在这个实施方式中，编译产生一个或多个动态链接库（DLL）文件，该 DLL 可以列出呈现以及其它在运行时被执行的指令。第一 DLL 文件 626 可以列出用于呈现图形显示元素的 XAML 控件的命令，而第二 DLL 文件 628 可以主要包括用于描述该显示的命令。在其它实施方式中，由于呈现图形指令的方式，上述编译步骤不是必须的。

控件和显示集合的编译版本可以接下来被下载或以另外的方式传输至任何数量的工作站或其它将要呈现过程图形显示的显示设备。在这个示例性的实施方式中，文件 626 和 628 被下载至存储器 630，比如它可以是操作员站的一部分或被许多工作站共享。当过程图形显示被呈现时，过程图形运行时应用程序 632 利用计算机可读介质生成运行时工作区 634，在该运行时工作区中可以分别实现 DLL 文件 626 和 628 的运行时实例 636 和 638 的执行和处理，从而呈现过程图形显示。具体来说，运行时工作区 634 可以包括一个或多个用于在 Avalon 框架中来执行应用程序的应用程序域 640 和 642。DLL 文件 636 和 638 接下来被装载入应用域 640 中的一个中，来生成被配置为示出该显示的用户界面窗格。为此，访问和激活显示数据源文件 644，并利用从标识在文件 644 中的源收集到的数据来呈现该显示。数据源文件 644 为定制数据适配器，它将 Avalon 图形与后端数据源如过程控制系统有效连接。

一般说来，如果显示或复合形状将包含关于一个或多个属性的动态行为，那么这些属性必须被绑定至来自控制系统或其它源的数据。这个数据链接在这里称为数据绑定。数据绑定可以有两个部件：数据源的标识以及数据源参考。在简单情况下，绑定值被完全限定，即绑定值解析为固码在比如用于定义类的 PGXML 脚本中的特定数据源。数据源标识可以包括路径名称，比如“DVRT:LIC-101/PID1/PV.CV.”。当显示被呈现或运行时，由用于显示的

数据绑定通过格式化和其它与绑定参数一起被指定的细节来检索数据。以这种方式，当值改变时，显示将被改变，从而建立该显示的或该显示元素的动态行为，并提供被呈现的显示和数据源之间通信的连续方式（即通信链接）。数据源的例子包括过程控制系统运行时系统（如 DeltaV 运行时，DvRt）、过程控制历史记录器、过程控制系统事件或报警、过程控制系统诊断数据和其它外部系统，如 Ovation、Provox（POC）、OPC 和 AMS。

图 20 为对象模型框架的示例性数据源对象模型部分 650 的示意性表示，用以支持上述数据源功能性。一般来说，如上所述，为每个过程图形显示生成定制数据源。在包含泵的显示中，该泵具有由颜色改变来反映的 IsOn 参数，配置工程师或其它用户可以配置泵的 IsOn 属性来绑定至数据源中路径上的项目上，如过程控制系统的以“MOD1\...”开头的路径。当 XAML 或其它过程图形呈现脚本在下载到操作员站或其它运行时环境显示设备之前产生时，泵实例可以接下来包含参考中间变量名的绑定语句。定制数据源也可以被生成（或修改）来包括具有相同名称的属性。为此，如图 20 所示，数据源对象模型 650 可以包括一般用于几个注册过的数据源的数据源管理对象 652。在这个简化的示例中，仅单个的 DvRt 数据源对象 654 被示出，它可以像数据源一样描述过程控制系统的使用。DvRt 数据源 654 和其它控制系统数据源被组织为 IDataSource 类 656 的子类，它可以为数据源路径表达式指定寄存器（或内存存储位置）信息。根据 IDataSource 类 656 组织的激活的数据源反过来是，根据专用于管理根据 IDataSourceManager 类 652 管理的数据源的控制系统部分的 DvDataSource 管理器 658。

用于所述显示的定制数据源已经通过 IDataSourceBase 类 660 得出，该 IDataSourceBase 类 660 通过通知类 662 和注册类 664 中的对象访问过 DvRtDataSource 类 654。DataSourceBase 类 660 的对象可以利用由显示得出的参考来生成。IBindingObject 类 664 的绑定对象 662 被添加到用于显示中的每个参考的数据源中。IPropertyChange 类 666 可以应用于处理显示的定制数据源。

利用定制数据源，从 PGXML 脚本中得出的 Avalon（或其它图形）控件可以数据绑定从属属性到数据源中。这样做时，从属属性的值由数据源在运行时提供，并且对基础数值的改变接下来反映在从属属性中。如果用户提供新的值，该值就被自动地推送到数据源中。

用于从数据源检索数据值的路径在这里可以被称为数据源参考。数据源参考语法在比如具有不同控制系统或其它类型的数据源的不同的实施方式之间是不同的。在某些情况下，数据源参考可以有一个属性来指示格式或语法的类型，比如字母数字或浮点型。顺便说一句，著名的用于商用系统的数据源参考的例子包括 LIC-101/PID1/PV.CV(DeltaV) 和 POINT/PARAMETER.FIELD(Provox)。如上所述，数据源和数据源参考的结合允许运行时工作区获得用于在显示中驱动动态行为的数据值。

当属性被绑定至过程控制数据值时，“Value Formatter”对象可以在配置期间被生成。数值格式程序对象可以指定绑定的细节（如用法和属性，或被绑定的参数）以及数据源和数据源参考。可替换地，或者说可补充地，数值格式程序可以包括另外的数据，如格式化、数据转换（查找表、线性化、范围或脚本化功能）、纠错处理、刷新率和最小值改变。

在某些实施方式中，一个或多个查找表可以存储在配置数据库 28 以辅助数据绑定。例如，数值格式程序对象可以与配置期间的查找表相关联。该查找表可以被下载至工作台，并且可以提供用于全局变量和查找表的数据源扩展。这样的查找表可以在显示之间被共享，并被与所述显示独立地下载。

如上所述，通过 PGXML 描述来表述和定义的参数和其它信息的数据绑定，支持高级的和动态的图形，这些图形具有的动画以一种灵活的、可扩展的和可配置的方式绑定于或受控制于图形所属的加工厂数据。使用图形类和复合形状类以模块化的可扩展的方式来创建过程图形显示，可以提供给配置工程师进一步的灵活性。利用显示中与库对象（如复合形状类）相关联或链接的这些复合形状的实例，其中这些复合形状是从库对象中被实例化的，通过单独对库类的改变来自动传播更新。这些只是被基于 XML 的过程图形描

述和参数化这样的描述（即图形结构定义和图形参数的分离）支持的一些有效之处。

如上所述，每个过程图形显示以及其中包含的图形显示元素的创建都记录在以第一说明性格式表述的各自的文本描述中。每种描述中的脚本命令提供有效的非存储强化机制来定义所述显示，尽管有图形呈现的复杂性。说明性格式以及因此的脚本命令，可以基于若干不同标记语言中的任何一种。具体来说，可以依赖基于 XML 的标记语言为每个显示和显示元素列出呈现定义或 XML 二进制大对象（blob）。为了支持高级的图形，如动画，标记语言也可以按照矢量图形方案来定义这样的图形。

如上面进一步的描述，过程图形的动态特性被设计为反映当在线或仿真条件改变时过程的参数的当前值。为此，过程图形可以链接至反映这种改变的数据的源。每个基于 XML 的描述可以因此包括一个或多个数据源参考，它们一般为根据数据修改的每个相应的动态图形参数（如罐内变化的颜色）标识数据源位置。数据源位置也可以通过编辑器保留开放给后来的在配置期间由编辑器做成的指定，这样脚本标识别名或占位符来参考后来指定的数据源或路径信息。因为数据源信息和过程图形显示的其它特征（如处理事件的行为）通过基于 XML 的描述来指定，所以该基于 XML 的语言可以称为 PGXML 或过程图形 XML。

完成关于定义过程图形显示及其组成元素的配置和设计工作之后，配置工程师或其它用户可以选择处理 PGXML 描述从而为下载过程图形给操作员站或其它用户显示设备做准备。一般来说，过程图形显示和图形显示元素的每个 PGXML 描述经过处理来产生：（i）与要使用的图形呈现引擎相兼容的矢量图形格式中的脚本；（ii）具有指定一显示的数据源参考和任何其它非图形功能性（例如行为）的指令的代码。脚本的矢量图形格式也可以是说明性或基于 XML 的语言。在使用微软 Avalon 用户界面架构的实施方式中，矢量图形脚本可以用微软 XAML(可扩展应用程序标记语言)来表述。其它实施方式可以使用其它格式，如开放源格式 SVG（Scalable Vector Graphics，

可缩放矢量图形)。之所以可能使用任何有效的矢量图形架构是因为 PGXML 描述用直接用于呈现图形结构的格式或对象模型之外的分离格式来表示。代码可以用 C#或任何其它期望的合适的编程语言来表述。

在某些实现方式中，矢量图形脚本和相关联的代码接下来被合并，并在列出用于操作员站或其它用户显示图形设备的可执行命令的文件中被编译。为此，可以为每个过程图形显示和图形显示元素创建各自的动态链接库（DLL）文件。在任何情况下，矢量图形脚本的这样的编译和相关代码可以在下载之前被执行，以便最小化网络数据传输要求。

前述对象模型框架的示例性部分只表示大量的类中的一些类，这些类被定义为支持过程图形和其它用户界面系统的功能性。

在执行时，任何本发明描述的软件可以被存储在任何计算机可读存储器中，比如在磁盘、光盘，或其它存储介质上，在计算机或处理器的 RAM 或 ROM 里，等等。同样，这些软件可以由用户、加工厂或其它操作员站使用任何公知的或期望的传送方法来传送，比如在一个计算机可读磁盘或其它可便携式计算机存储机制或通过通信通道，如电话线、互连网、万维网、任何其它局域网或广域网等等（这种传送被视为与通过便携式存储介质提供这样的软件一样或可替换）。此外，这种软件可以被不经过调制或加密而被直接提供，或者，可以在通过通信通道传输之前，用任何合适的调制载波和/或加密技术进行调制或加密。

以上根据特定实施例对本发明进行了描述，旨在用于说明而非限定本发明。本领域普通技术人员能理解的是，在不背离本发明的保护范围的情况下，可以对本发明公开的实施方式做出任何修改、补充或删除。

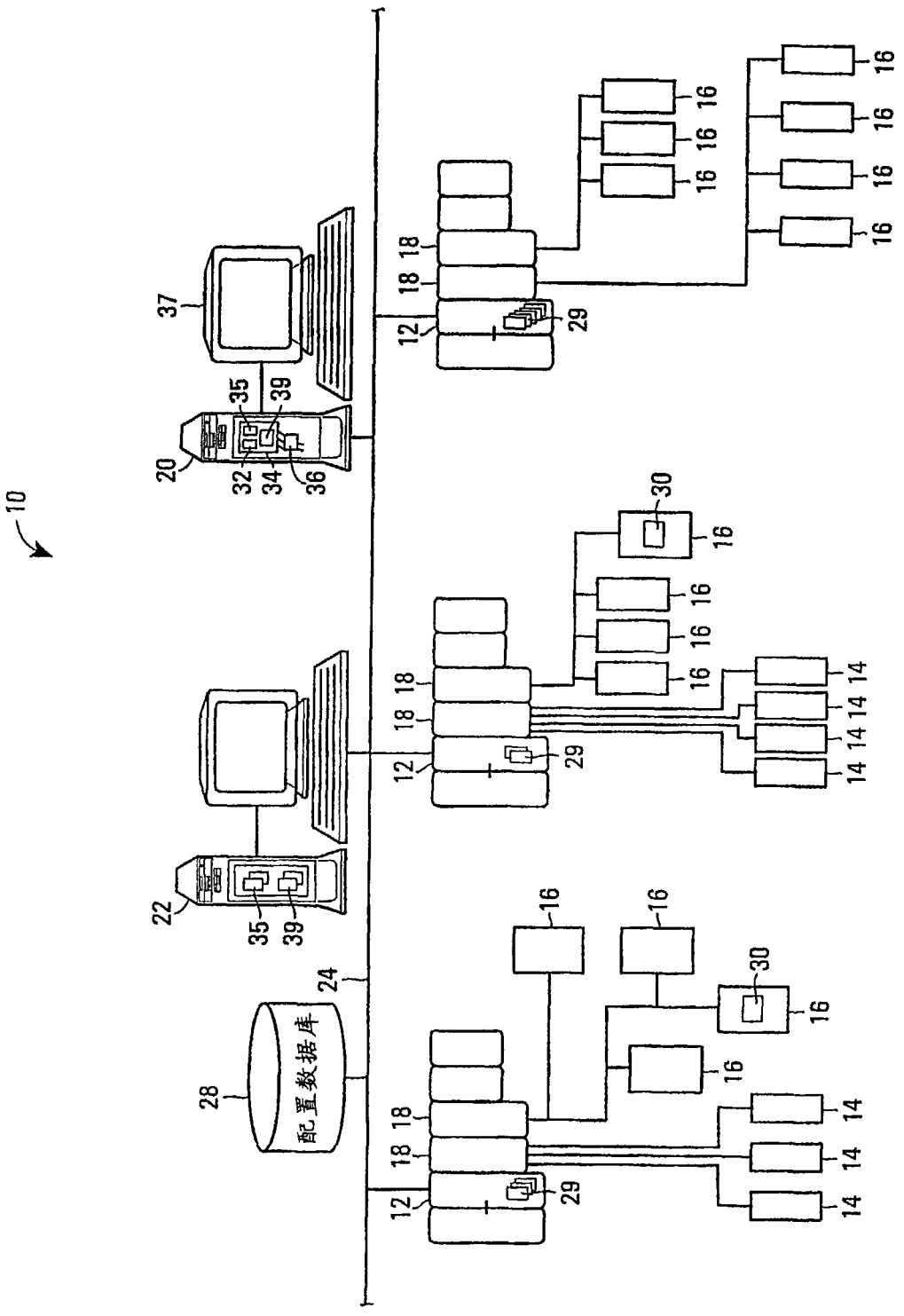


图 1

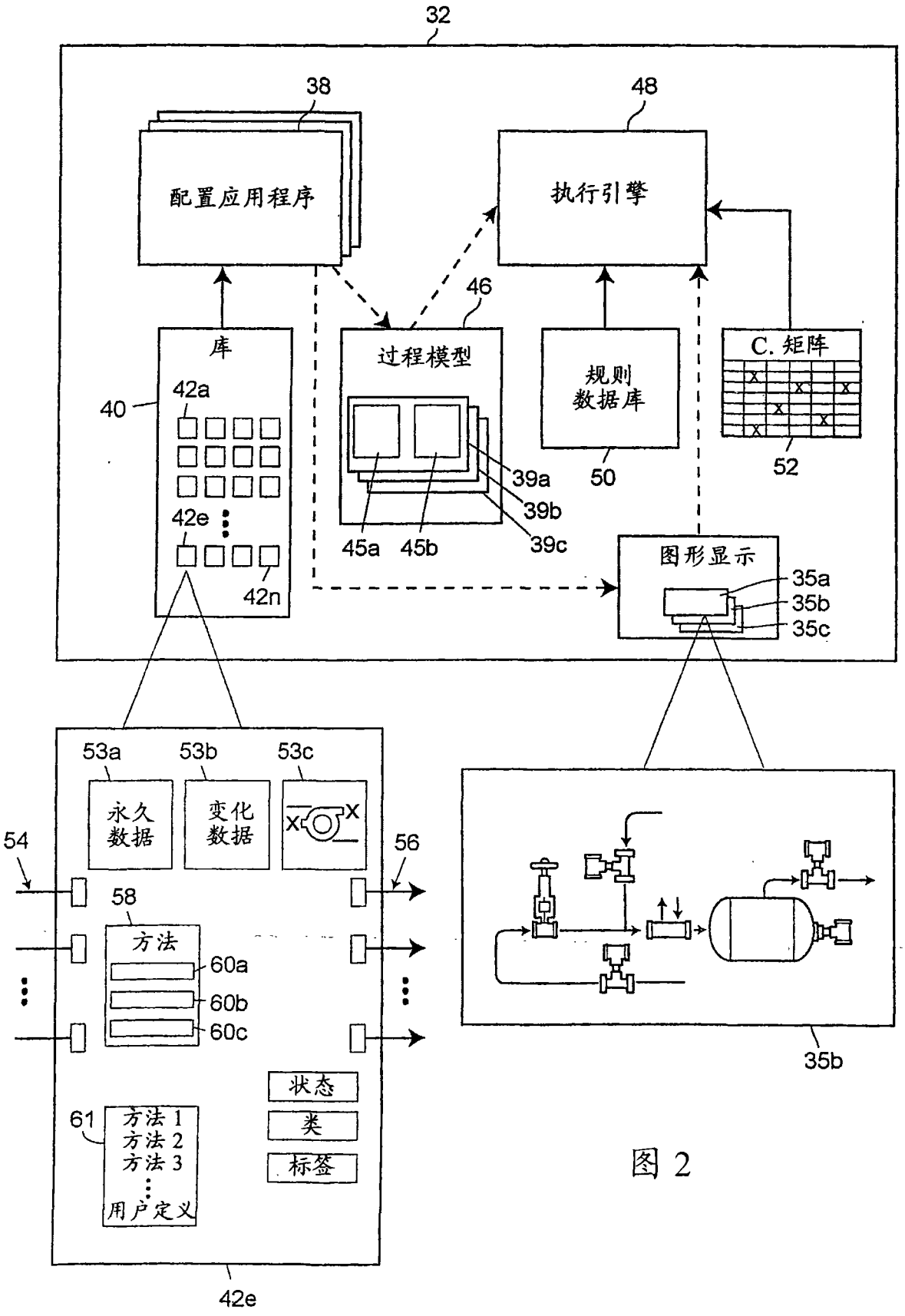

32
38
配置应用程序
48
执行引擎
46
过程模型
库
40
42a
42e
42n
39a
39b
39c
45a
45b
规则
数据库
50
C. 矩阵
52
图形显示
35a
35b
35c
53a
53b
53c
永久
数据
变化
数据
54
56
58
方法
60a
60b
60c
状态
类
标签
61
方法 1
方法 2
方法 3
用户定义
42e
35b

图 2

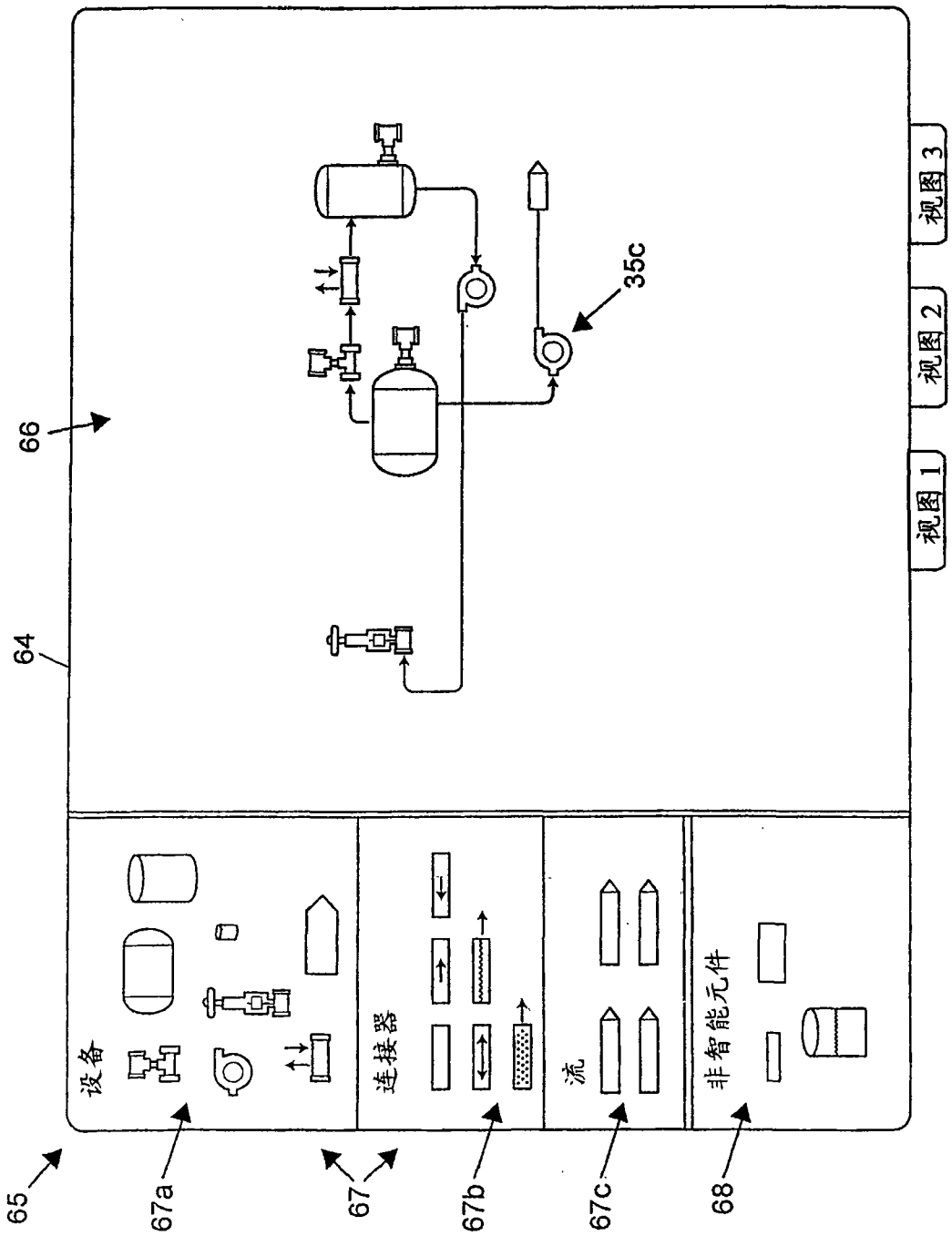


图 3

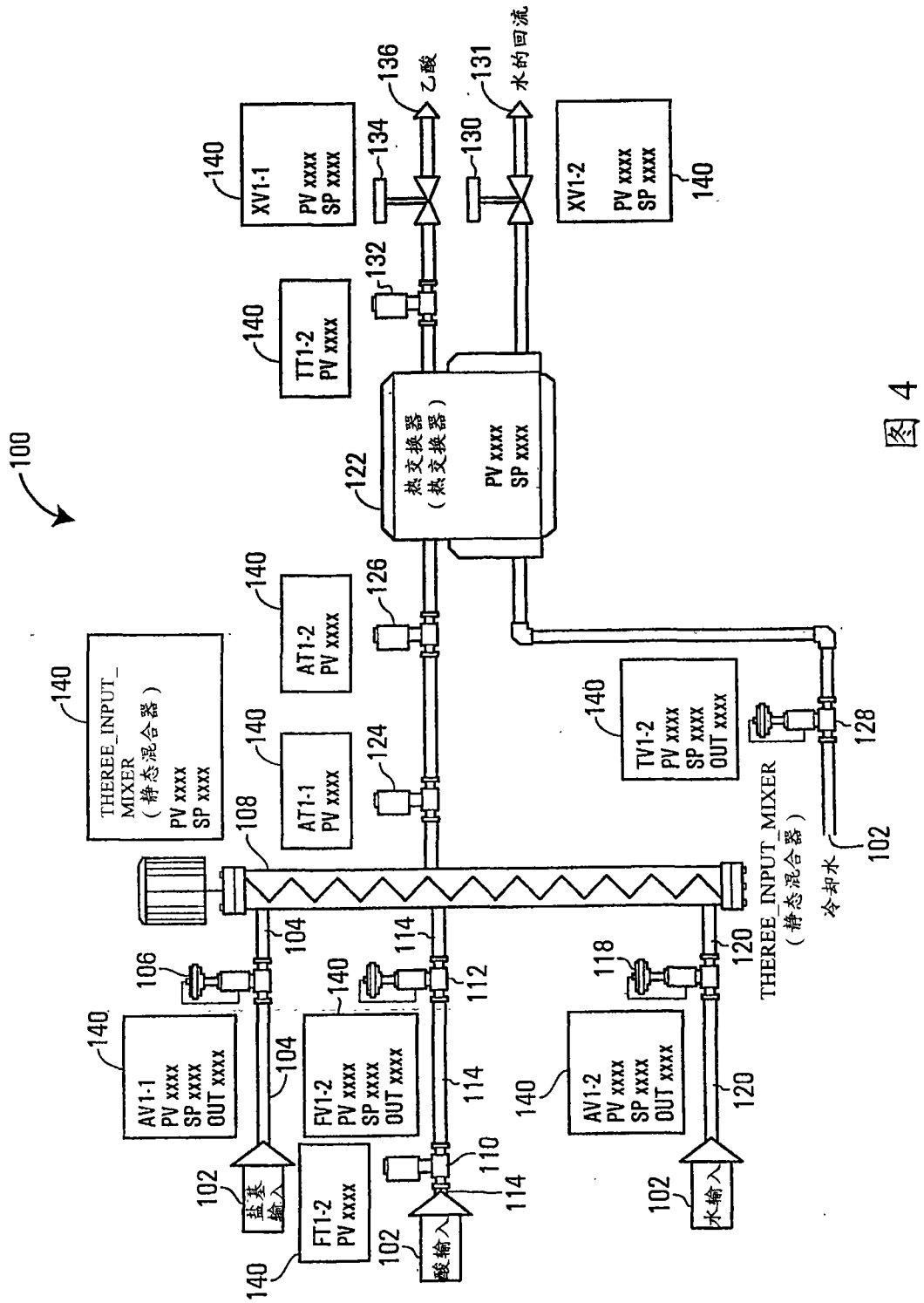

100
140
XV1-1
PV xxxx
SP xxxx
136
乙酸
134
131
水的回流
130
XV1-2
PV xxxx
SP xxxx
140
132
TT1-2
PV xxxx
140
热交换器
（热交换器）
PV xxxx
SP xxxx
122
140
AT1-2
PV xxxx
126
140
THEREE_INPUT_
MIXER
（静态混合器）
PV xxxx
SP xxxx
140
AT1-1
PV xxxx
124
108
140
TV1-2
PV xxxx
SP xxxx
OUT xxxx
128
102
冷却水
THEREE_INPUT_MIXER
（静态混合器）
104
114
120
106
112
118
140
AV1-1
PV xxxx
SP xxxx
OUT xxxx
140
FV1-2
PV xxxx
SP xxxx
OUT xxxx
140
AV1-2
PV xxxx
SP xxxx
OUT xxxx
104
114
120
102
盐基输入
FT1-2
PV xxxx
140
110
114
102
酸输入
102
水输入

图 4

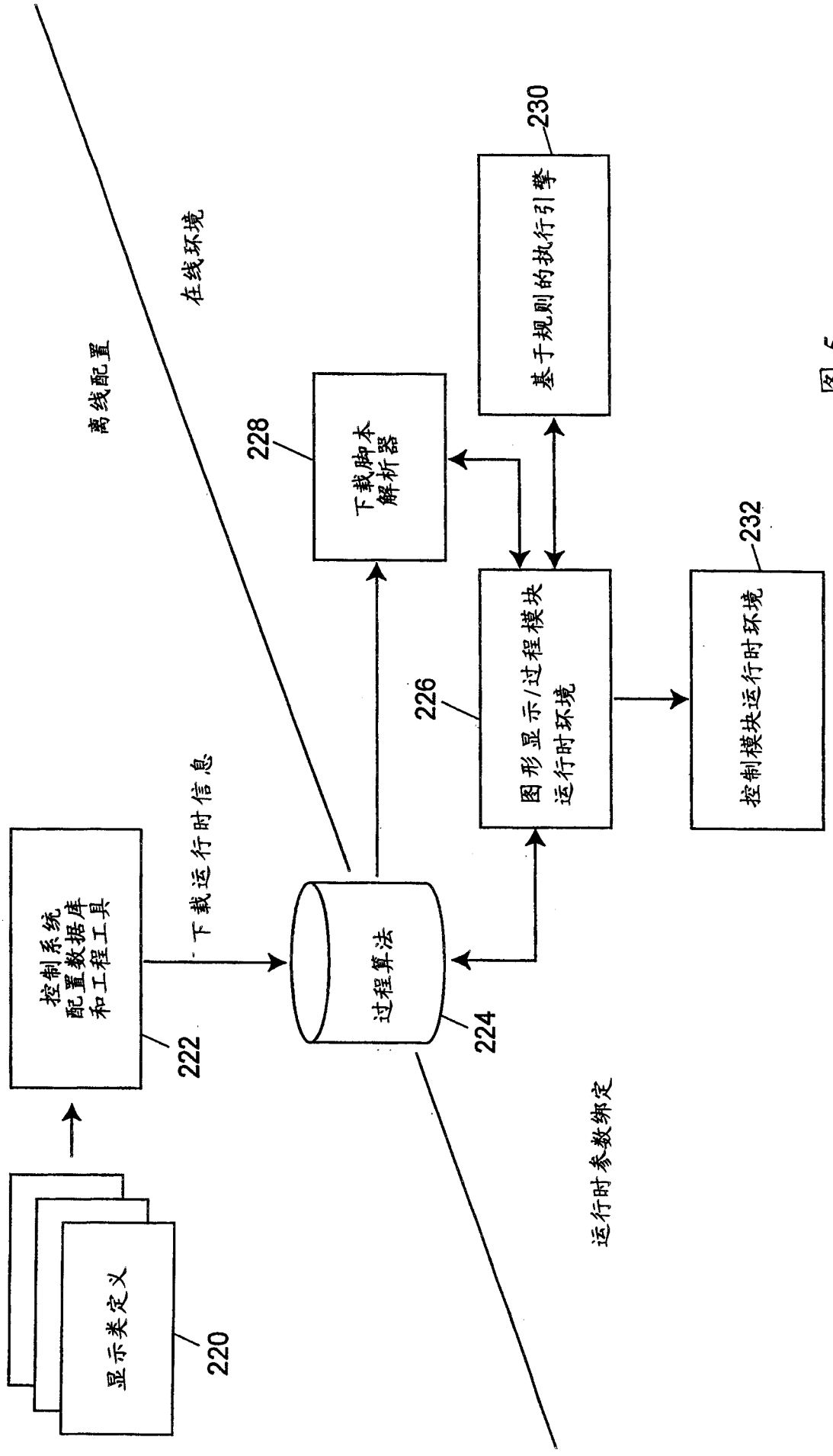

离线配置
在线环境
显示类定义
220
控制系统
配置数据库
和工程工具
222
下载运行时信息
过程算法
224
下载脚本
解析器
228
基于规则的执行引擎
230
图形显示/过程模块
运行时环境
226
控制模块运行时环境
232
运行时参数绑定

图 5

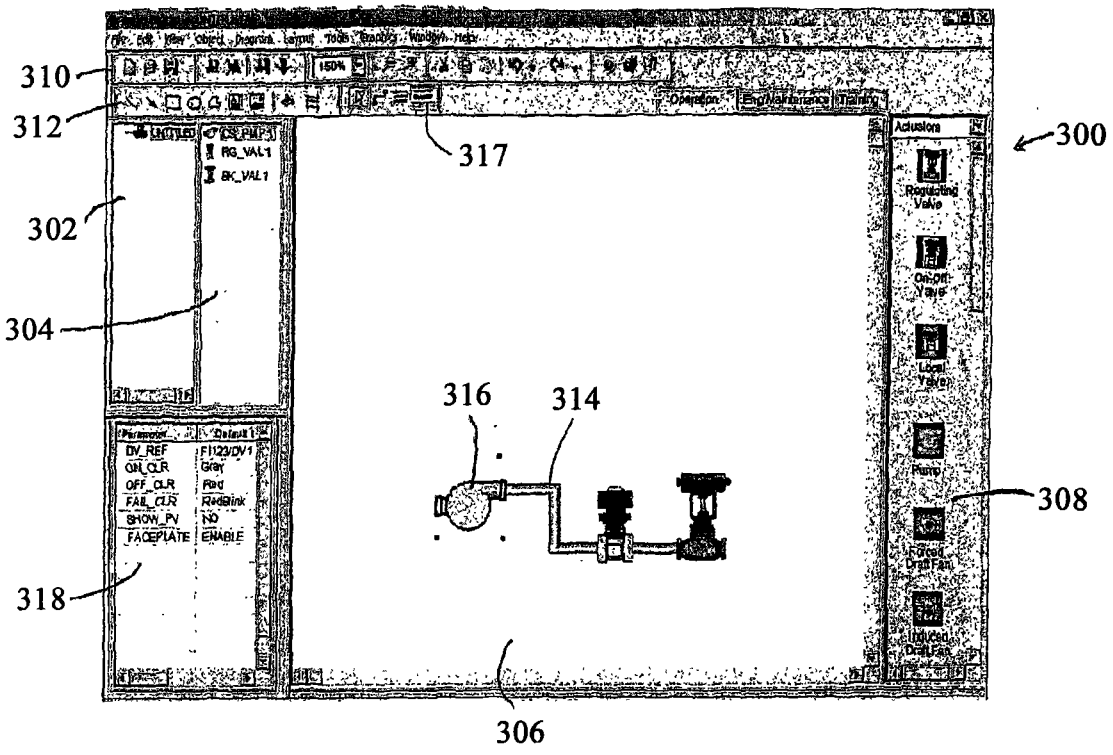

310
312
317
300
302
304
316
314
308
318
306

图 6

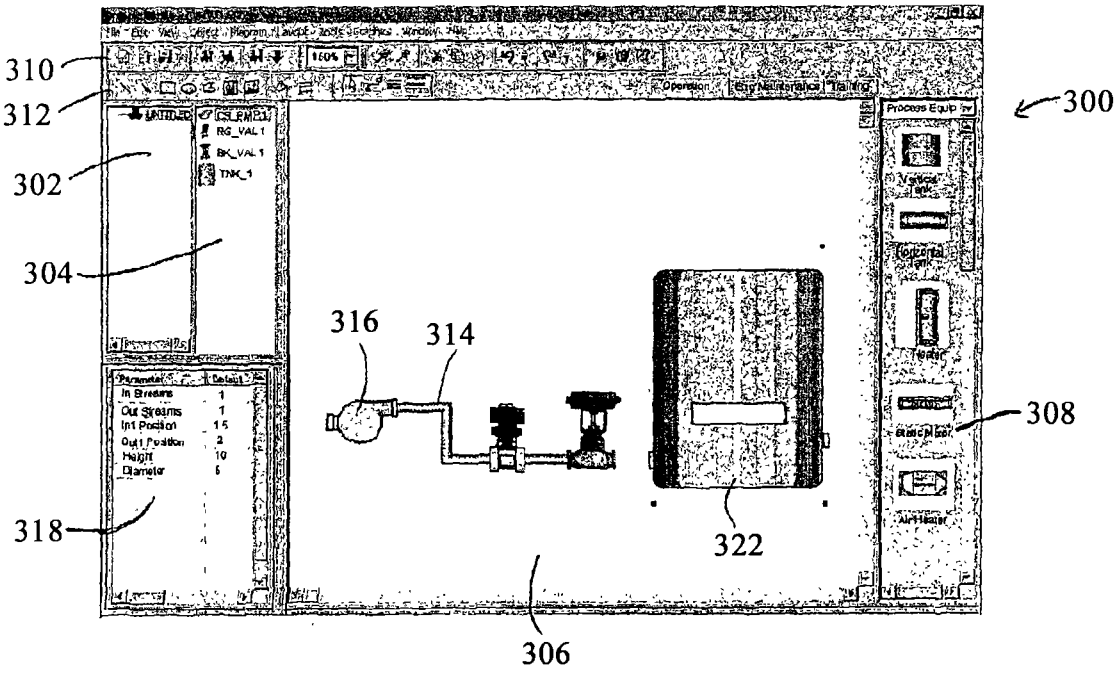

310
312
300
302
304
316
314
308
322
318
306

图 7

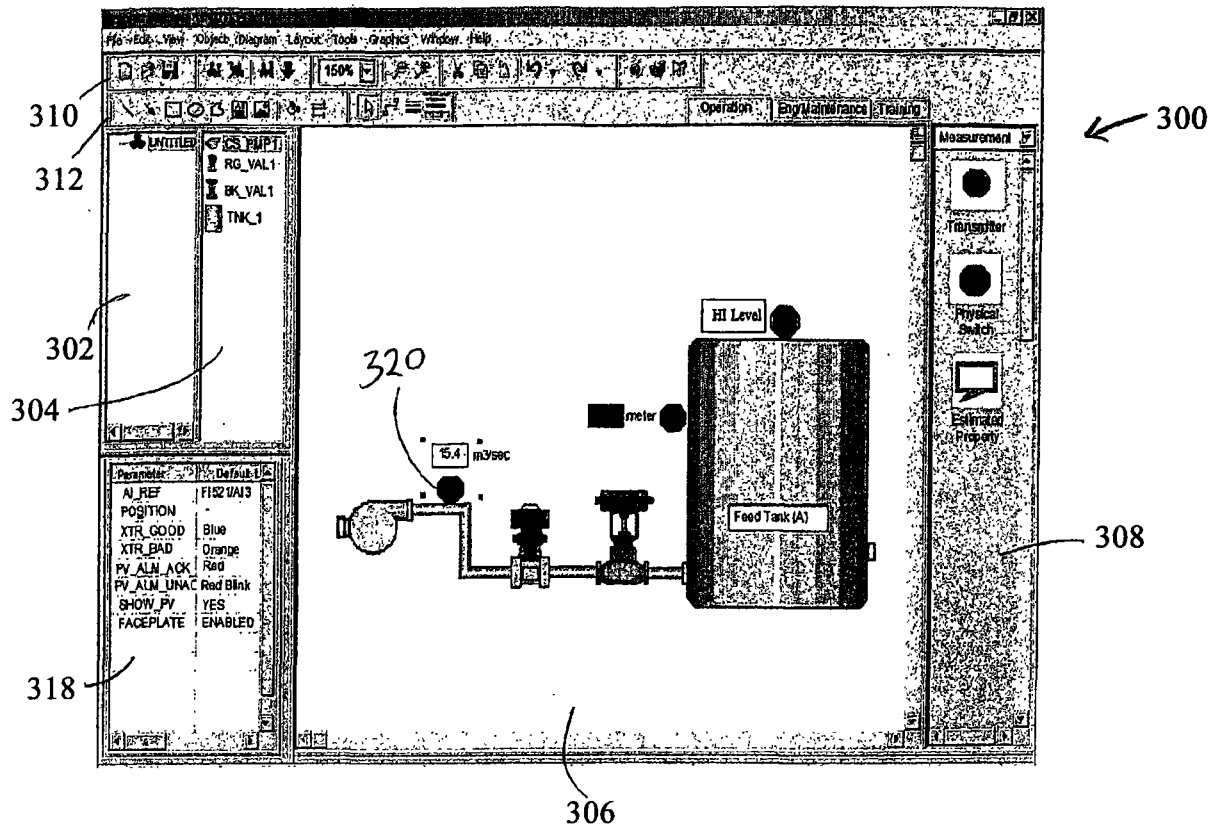

310
312
302
304
318
320
300
308
306
HI Level
meter
15.4 m3/sec
Feed Tank (A)
Measurement
Transmitter
Physical Switch
Estimated Property
RG_VAL1
BK_VAL1
TNK_1
Operation
Eng/Maintenance
Training

图 8

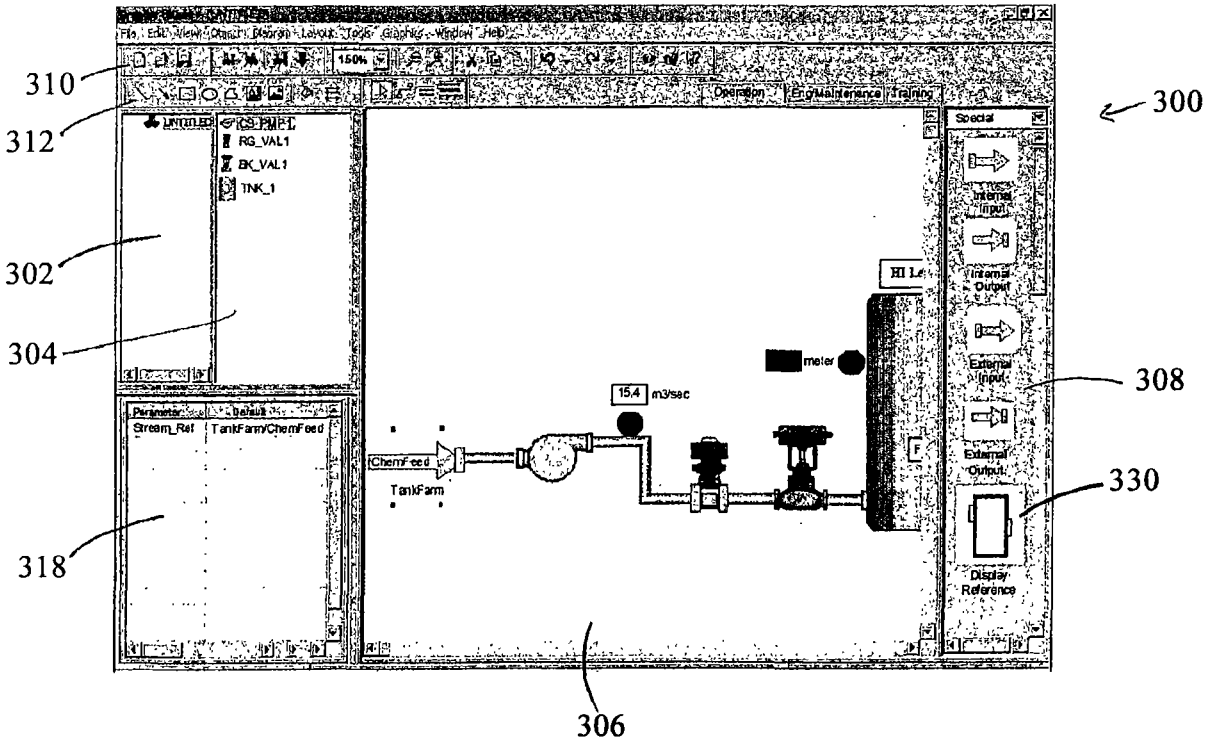

310
312
302
304
318
300
308
330
306
RG_VAL1
BK_VAL1
TNK_1
Operation
Eng/Maintenance
Training
Special
meter
15.4 m3/sec
ChemFeed
TankFarm
Internal Input
Internal Output
External Input
External Output
Display Reference

图 9

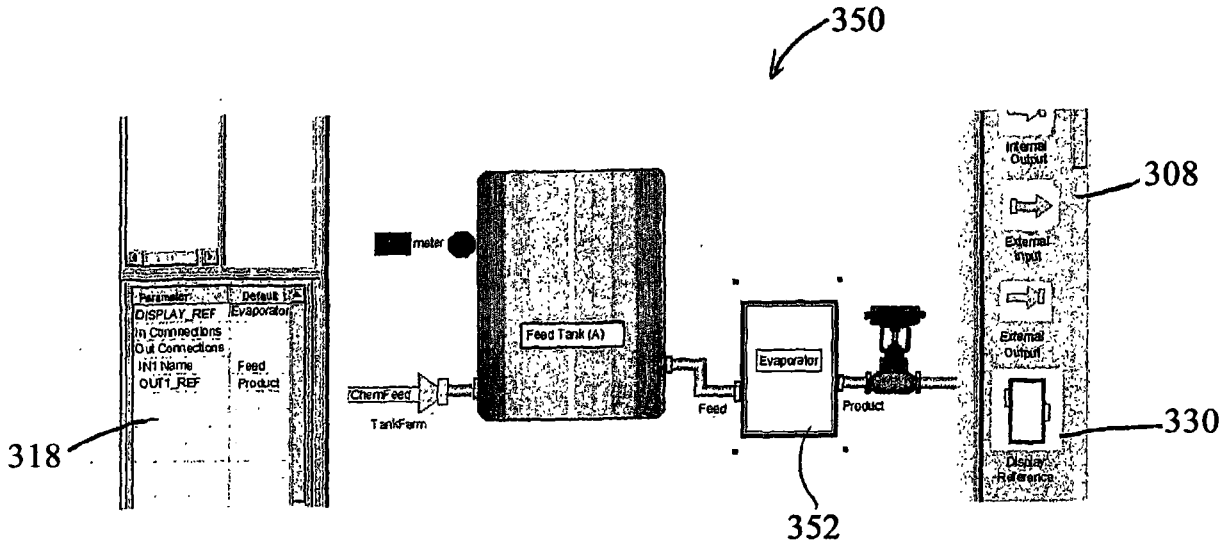

350
308
330
318
352
Parameter
Default
DISPLAY_REF
Evaporator
In Connections
Out Connections
IN1 Name
Feed
OUT1_REF
Product
meter
Feed Tank (A)
ChemFeed
TankFarm
Evaporator
Feed
Product
Internal Output
External Input
External Output
Display Reference

图 10

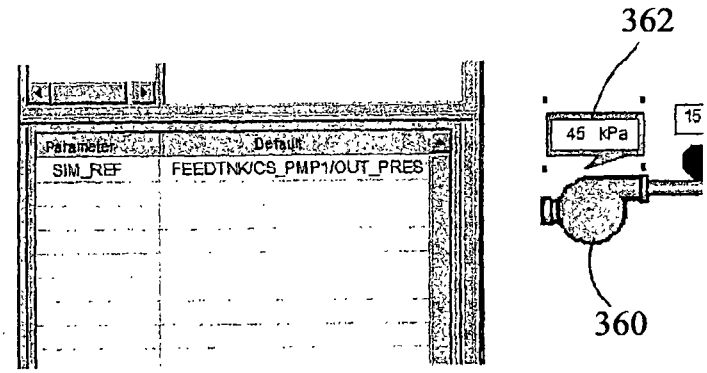

362
360
Parameter
Default
SIM_REF
FEEDTNK/CS_PMP1/OUT_PRES
45 kPa
15

图 11

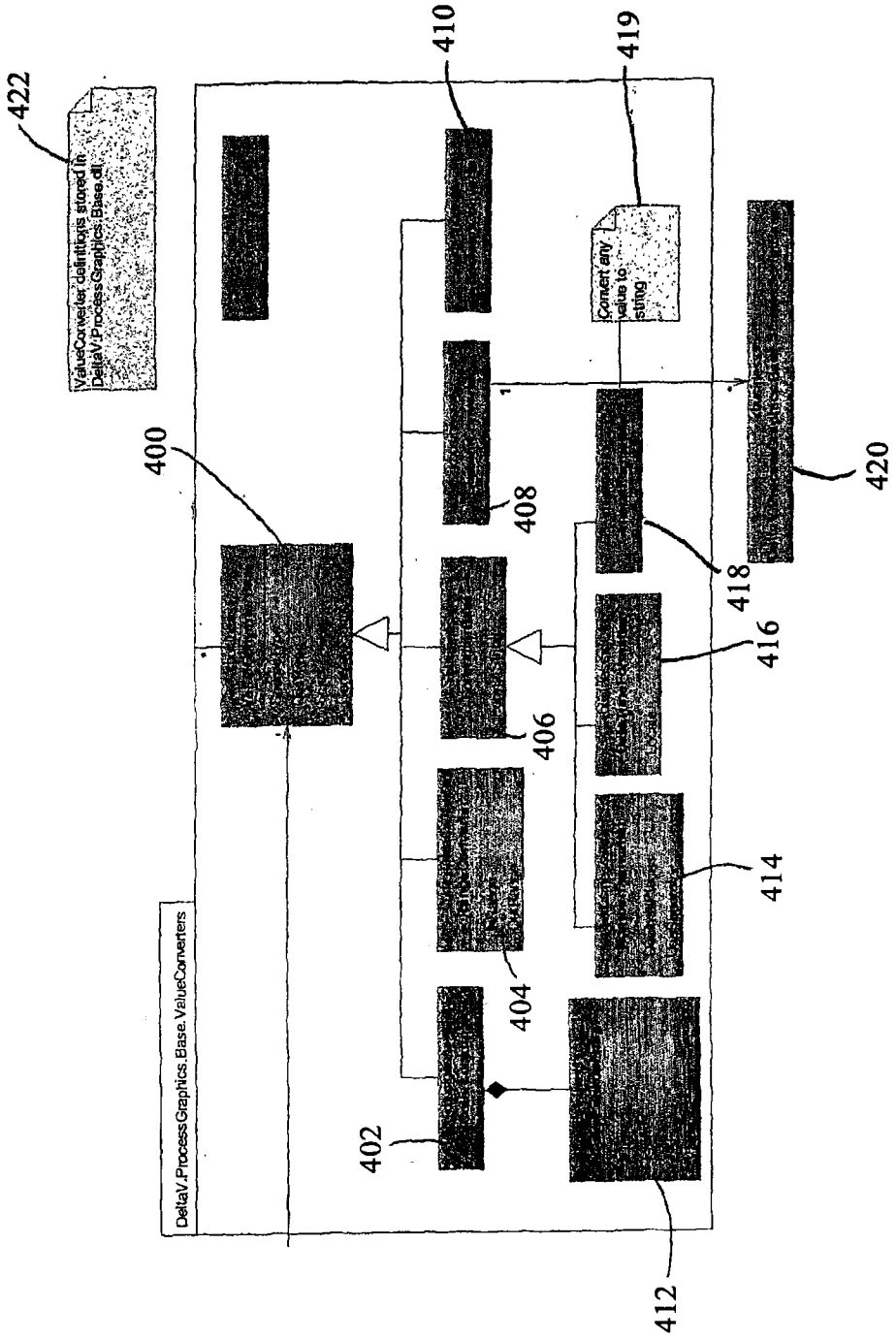

422
ValueConverter definitions stored in DeltaV.ProcessGraphics.Base.dll
410
419
Convert any value to string
400
408
418
420
416
406
414
404
402
412
DeltaV.ProcessGraphics.Base.ValueConverters

图 12

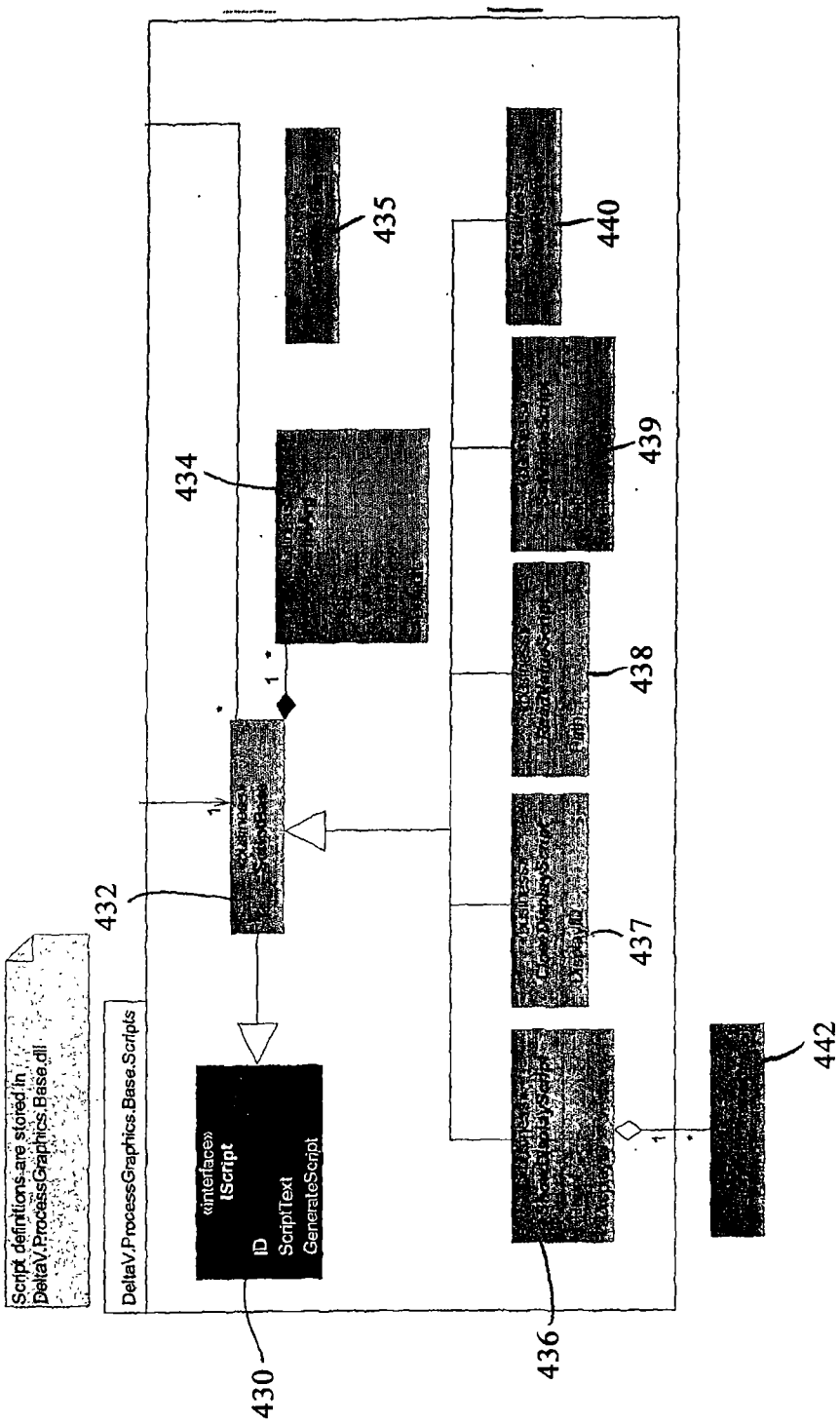

Script definitions are stored in DeltaV.ProcessGraphics.Base.dll
DeltaV.ProcessGraphics.Base.Scripts
«interface»
IScript
ID
ScriptText
GenerateScript
430
432
434
435
436
437
438
439
440
442

图 13

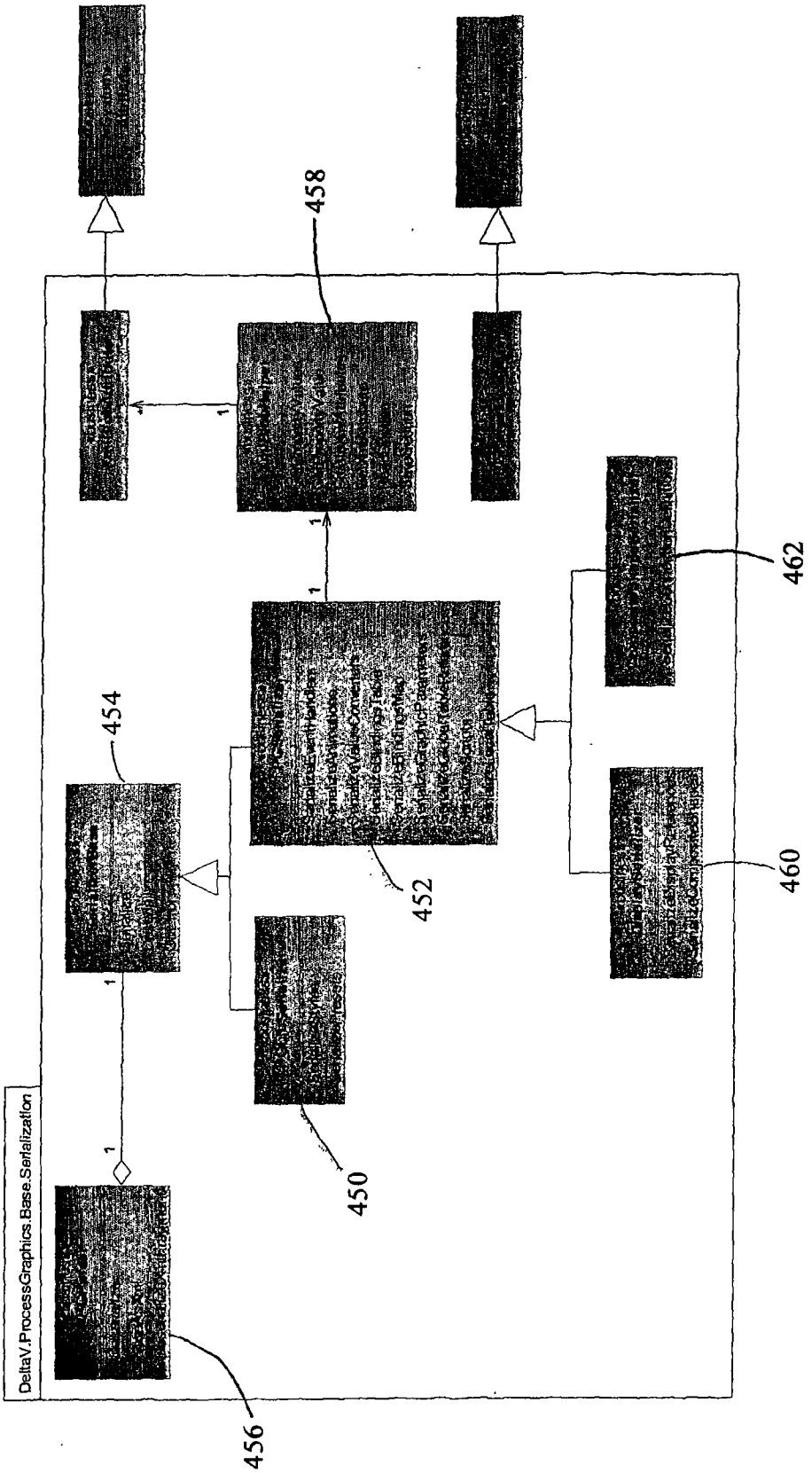

458
462
454
452
460
450
456
DeltaV.ProcessGraphics.Base.Serialization

图 14

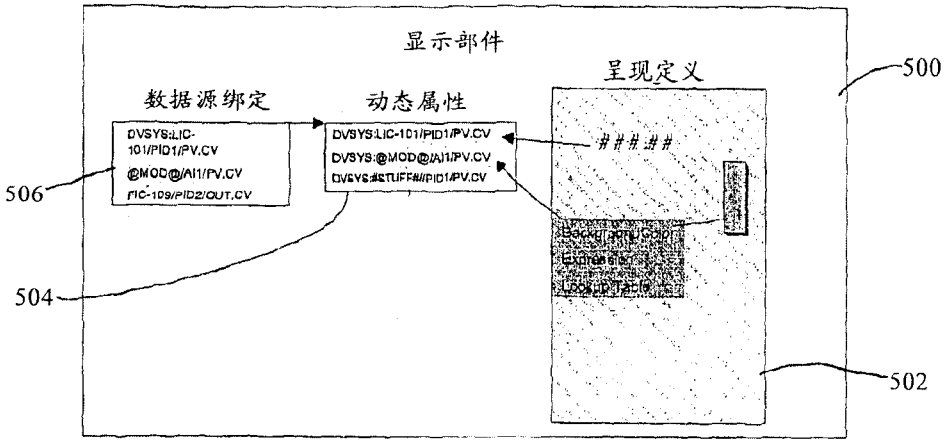

显示部件
呈现定义
500
数据源绑定
动态属性
DVSYS:LIC-101/PID1/PV.CV
@MOD@/AI1/PV.CV
FIC-109/PID2/OUT.CV
DVSYS:LIC-101/PID1/PV.CV
DVSYS:@MOD@/AI1/PV.CV
506
504
502

图 15

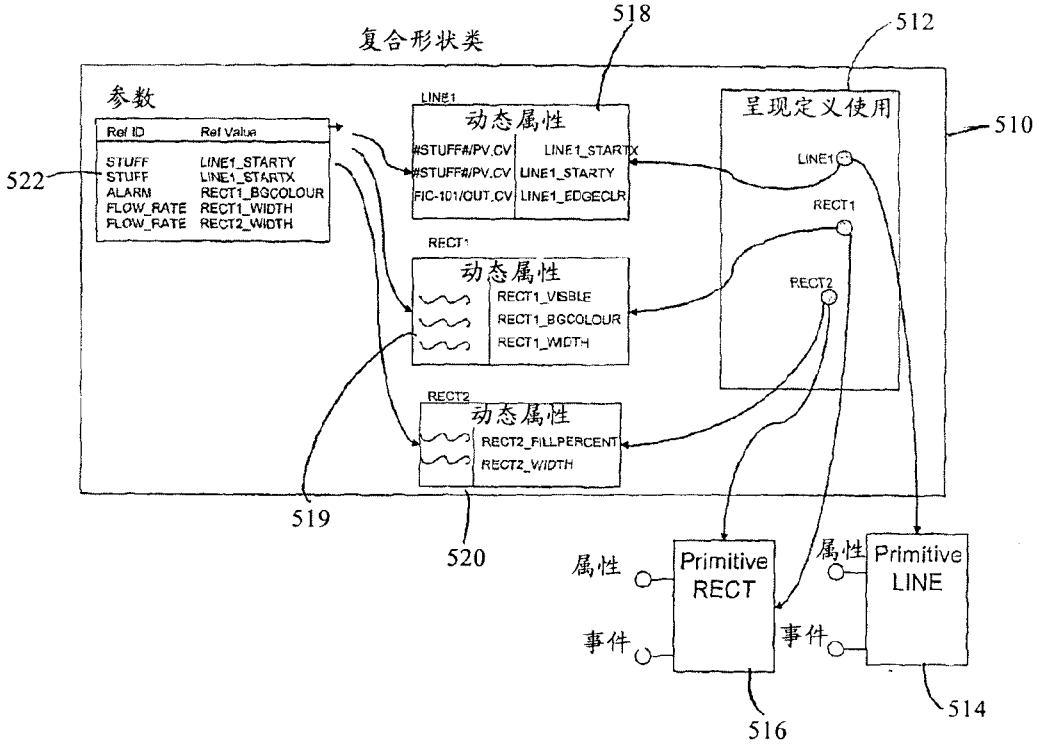

复合形状类
518
512
参数
Ref ID
Ref Value
STUFF LINE1_STARTY
STUFF LINE1_STARTX
ALARM RECT1_BGCOLOUR
FLOW_RATE RECT1_WIDTH
FLOW_RATE RECT2_WIDTH
522
LINE1
动态属性
#STUFF#/PV.CV LINE1_STARTX
#STUFF#/PV.CV LINE1_STARTY
FIC-101/OUT.CV LINE1_EDGECLR
RECT1
动态属性
RECT1_VISBLE
RECT1_BGCOLOUR
RECT1_WIDTH
RECT2
动态属性
RECT2_FILLPERCENT
RECT2_WIDTH
呈现定义使用
510
LINE1
RECT1
RECT2
519
520
属性
事件
Primitive RECT
属性
事件
Primitive LINE
516
514

图 16

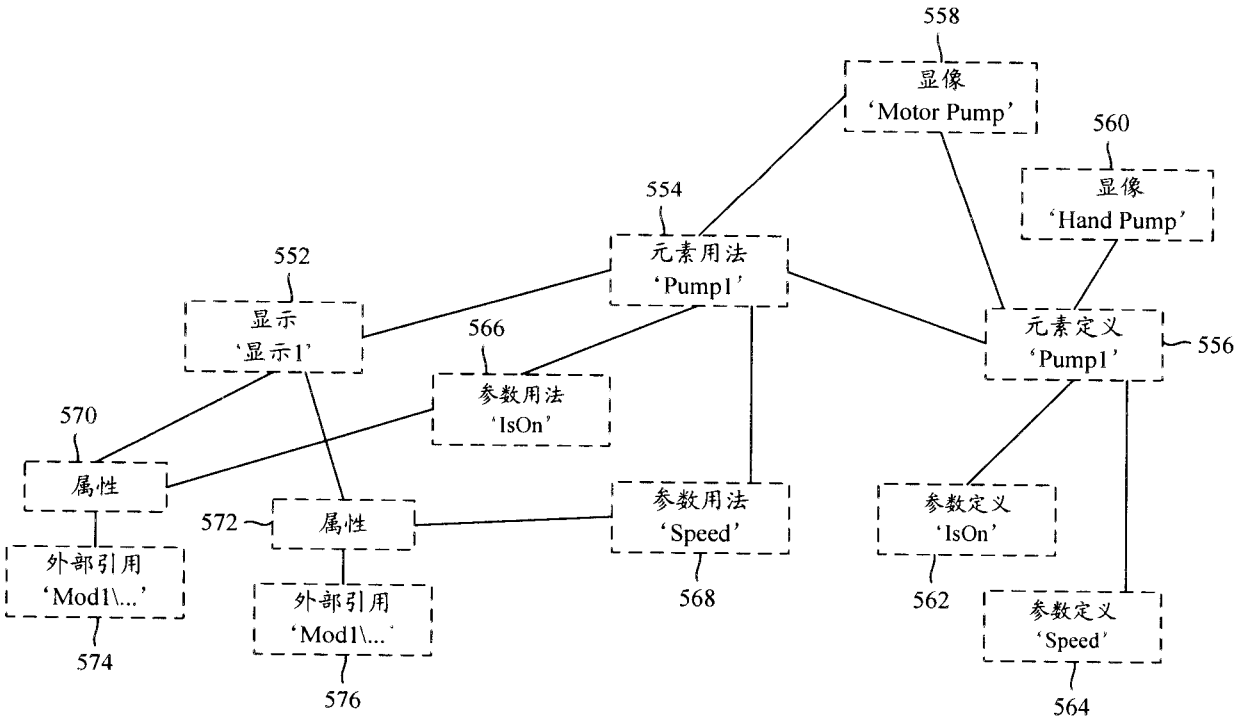

558
显像
'Motor Pump'
560
显像
'Hand Pump'
554
元素用法
'Pump1'
552
显示
'显示1'
566
参数用法
'IsOn'
元素定义
'Pump1'
556
570
属性
572
属性
参数用法
'Speed'
参数定义
'IsOn'
外部引用
'Mod1\...'
外部引用
'Mod1\...'
568
562
参数定义
'Speed'
574
576
564

图 17

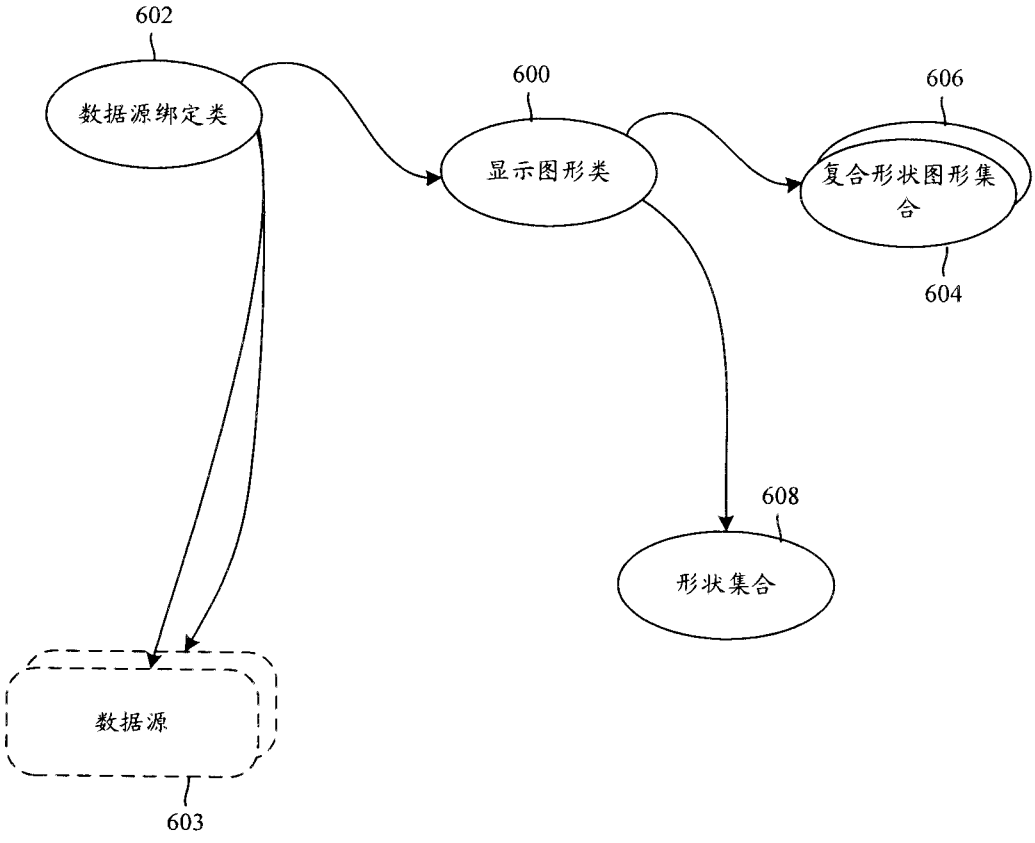

602
数据源绑定类
600
显示图形类
606
复合形状图形集合
604
608
形状集合
数据源
603

图 18

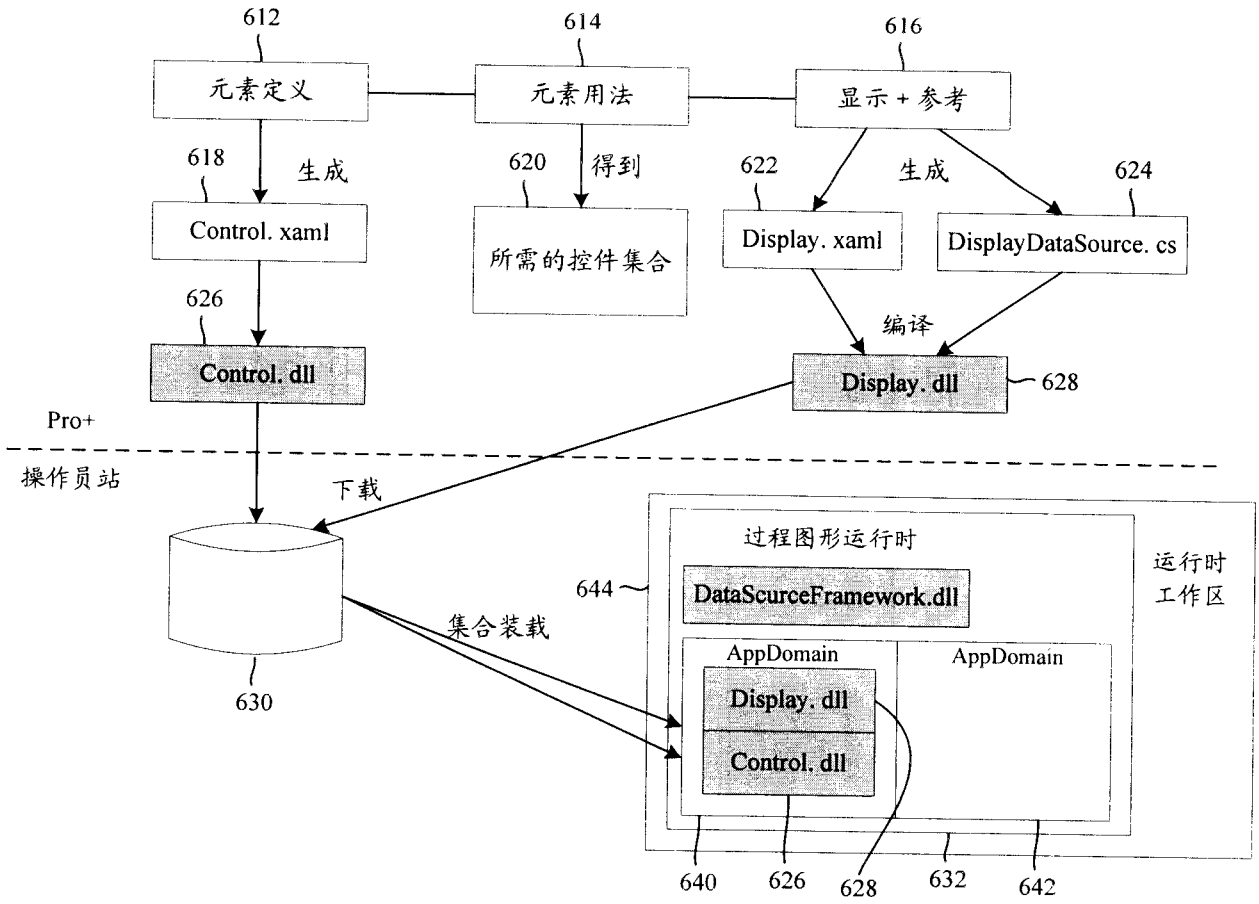

612
元素定义
614
元素用法
616
显示 + 参考
618
生成
620
得到
622
生成
624
Control. xaml
所需的控件集合
Display. xaml
DisplayDataSource. cs
626
编译
Control. dll
Display. dll
628
Pro+
操作员站
下载
过程图形运行时
运行时
工作区
644
DataScurceFramework.dll
集合装载
AppDomain
AppDomain
Display. dll
Control. dll
630
640
626
628
632
642

图 19

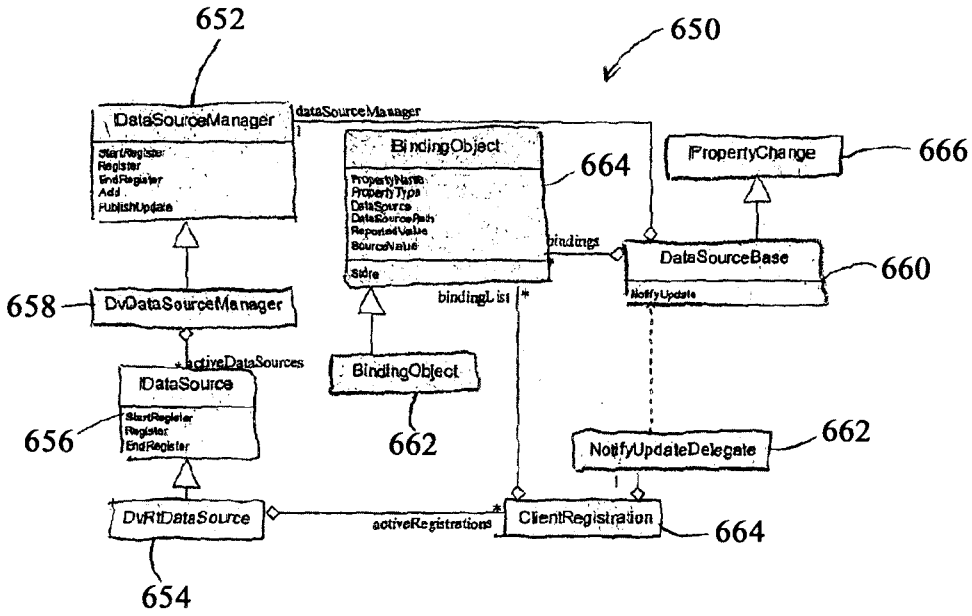

652
650
IDataSourceManager
dataSourceManager
BindingObject
664
IPropertyChange
666
bindings
DataSourceBase
660
658
DvDataSourceManager
bindingList
activeDataSources
IDataSource
BindingObject
656
662
NotifyUpdateDelegate
662
DvRtIDataSource
activeRegistrations
ClientRegistration
664
654

图 20