[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
*G06F 9/44* (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200580014527.9

[43] 公开日 2007 年 5 月 9 日

[11] 公开号 CN 1961288A

[22] 申请日 2005.5.4
[21] 申请号 200580014527.9
[30] 优先权
[32] 2004. 5. 4 [33] US [31] 60/567,980
[86] 国际申请 PCT/US2005/015585 2005.5.4
[87] 国际公布 WO2005/109250 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 斯蒂芬·吉尔伯特
斯蒂芬·G·汉莫克 周 玲
迈克尔·J·卢卡斯
马克·J·尼克松
[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 9 页 说明书 41 页 附图 27 页

[54] 发明名称

# 用于存取过程控制数据的方法和设备

[57] 摘要

用于访问过程控制数据的方法、装置和制品，包括：装载客户对象，和从该客户对象向被配置为与服务器通信的实对象传送数据访问请求。 该实对象然后根据该数据访问请求向该服务器传送查询，并且响应于该查询从该服务器获得过程控制数据。然后将该过程控制数据从与服务器模式关联的第一数据布局映射到与客户模式关联的第二数据布局。然后将所映射的过程控制数据传送给应用程序。

1、一种用于访问过程控制数据的方法，包括:

装载客户对象;

从该客户对象向被配置成与服务器通信的实对象传送数据访问请求;

根据该数据访问请求从该实对象向该服务器传送查询;

响应于该查询，从该服务器获得过程控制数据;

将该过程控制数据从与服务器模式关联的第一数据布局映射到与客户模式关联的第二数据布局; 和

将所映射的过程控制数据传送给应用程序。

2、根据权利要求 1 所述的方法，其中该过程控制数据以可扩展标记语言形式从该服务器获得。

3、根据权利要求 1 所述的方法，其中该过程控制数据是存储的过程控制数据或实时过程控制数据中的至少之一。

4、根据权利要求 1 所述的方法，其中该客户对象通过应用程序装载。

5、根据权利要求 1 所述的方法，其中将该过程控制数据从该第一数据布局映射到该第二数据布局的步骤包括: 将与该客户模式关联的多个客户角色元素映射到与该服务器模式关联的服务器角色元素。

6、根据权利要求 1 所述的方法，其中将该过程控制数据从该第一数据布局映射到该第二数据布局的步骤包括: 将与该客户模式关联的客户对象映射到与该服务器模式关联的多个服务器对象。

7、根据权利要求 1 所述的方法，其中将该过程控制数据从该第一数据布局映射到该第二数据布局的步骤包括: 将与该客户模式关联的客户角色元素映射到与该服务器模式关联的多个服务器角色元素。

8、根据权利要求 1 所述的方法，其中将该过程控制数据从该第一数据布局映射到该第二数据布局的步骤包括: 将客户角色元素和客户对象插入到该客户模式，其中该服务器模式不包括对应于该客户角色元素和该客户对象

的服务器角色元素和服务器对象。

9、根据权利要求 1 所述的方法，其中将该过程控制数据从该第一数据布局映射到该第二数据布局的步骤包括以命令方式实现客户角色元素。

10、根据权利要求 1 所述的方法，进一步包括在将该映射的过程控制数据传送给该应用程序之后卸载该客户对象。

11、根据权利要求 10 所述的方法，其中卸载该客户对象的步骤包括指定该客户对象未被使用。

12、一种用于访问过程控制数据的装置，包括：

处理器系统；和

与该处理器系统通信连接的存储器，该存储器包括存储的指令，其使得该处理器系统能够：

装载客户对象；

从该客户对象向被配置为与服务器通信的实对象传送数据访问请求；

根据该数据访问请求从该实对象向该服务器传送查询；

响应于该查询，从该服务器获得过程控制数据；

将该过程控制数据从与服务器模式关联的第一数据布局映射到与客户模式关联的第二数据布局；和

将所映射的过程控制数据传送给应用程序。

13、根据权利要求 12 所述的装置，其中该过程控制数据以可扩展标记语言形式从该服务器获得。

14、根据权利要求 12 所述的装置，其中该过程控制数据是存储的过程控制数据或实时过程控制数据中的至少之一。

15、根据权利要求 12 所述的装置，其中该客户对象通过应用程序装载。

16、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够通过将与该客户模式关联的多个客户角色元素映射到与该服务器模式关联的服务器角色元素，来将该过程控制数据从该第一数据布局映射到该第二数

据布局。

17、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够通过将与该客户模式关联的客户对象映射到与该服务器模式关联的多个服务器对象，来将该过程控制数据从该第一数据布局映射到该第二数据布局。

18、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够通过将与该客户模式关联的客户角色元素映射到与该服务器模式关联的多个服务器角色元素，来将该过程控制数据从该第一数据布局映射到该第二数据布局。

19、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够通过将客户角色元素和客户对象插入到该客户模式中来将该过程控制数据从该第一数据布局映射到该第二数据布局，其中该服务器模式不包括对应于该客户角色元素和该客户对象的服务器角色元素和服务器对象。

20、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够通过以命令方式实现客户角色元素来将该过程控制数据从该第一数据布局映射到该第二数据布局。

21、根据权利要求 12 所述的装置，其中该指令使得该处理器系统能够在将该映射的过程控制数据传送给该应用程序之后卸载该客户对象。

22、根据权利要求 21 所述的装置，其中该指令使得该处理器系统能够通过指定该客户对象未被使用来卸载该客户对象。

23、一种机器可访问介质，其上存储有指令，当执行该指令时使得机器：

装载客户对象；

从该客户对象向被配置为与服务器通信的实对象传送数据访问请求；

根据该数据访问请求从该实对象向该服务器传送查询；

响应于该查询，从该服务器获得过程控制数据；

将该过程控制数据从与服务器模式关联的第一数据布局映射到与客户模式关联的第二数据布局；和

将所映射的过程控制数据传送给应用程序。

24、根据权利要求 23 所述的机器可访问介质，其中该过程控制数据以可扩展标记语言形式从该服务器获得。

25、根据权利要求 23 所述的机器可访问介质，其中该过程控制数据是存储的过程控制数据或实时过程控制数据中的至少之一。

26、根据权利要求 23 所述的机器可访问介质，其中该客户对象通过应用程序装载。

27、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器通过将与该客户模式关联的多个客户角色元素映射到与该服务器模式关联的服务器角色元素，来将该过程控制数据从该第一数据布局映射到该第二数据布局。

28、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器通过将与该客户模式关联的客户对象映射到与该服务器模式关联的多个服务器对象，来将该过程控制数据从该第一数据布局映射到该第二数据布局。

29、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器通过将与该客户模式关联的客户角色元素映射到与该服务器模式关联的多个服务器角色元素，来将该过程控制数据从该第一数据布局映射到该第二数据布局。

30、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器通过将客户角色元素和客户对象插入到该客户模式中来将该过程控制数据从该第一数据布局映射到该第二数据布局，其中该服务器模式不包括对应于该客户角色元素和该客户对象的服务器角色元素和服务器对象。

31、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器通过以命令方式实现客户角色元素来将该过程控制数据从该第一数据布局映射到该第二数据布局。

32、根据权利要求 23 所述的机器可访问介质，其中当执行该指令时，使得该机器在将该映射的过程控制数据传送给该应用程序之后卸载该客户

对象。

33、根据权利要求 32 所述的机器可访问介质，其中当执行该指令时，使得该机器通过指定该客户对象未被使用来卸载该客户对象。

34、一种用于访问过程控制数据的方法，包括:

响应于用户界面请求装载第一和第二客户对象，其中该第一和第二客户对象与访问基于客户模式组织的过程控制数据关联;

装载与该第一和第二客户对象关联的实对象，该实对象被配置用来获得基于服务器模式组织的过程控制数据;

将过程控制数据从该服务器模式组织映射到该客户模式组织，并将该过程控制数据传送到该第一和第二客户对象；和

通过与该第一客户对象关联的第一用户界面和与该第二客户对象关联的第二用户界面获得该过程控制数据。

35、根据权利要求 34 所述的方法，其中基于至少一个掩码将该过程控制数据从该服务器模式组织映射到该客户模式组织。

36、根据权利要求 34 所述的方法，进一步包括在该第一用户界面结束使用该第一客户对象之后将该第一客户对象指定为未活动状态。

37、根据权利要求 36 所述的方法，进一步包括卸载该第一客户对象句柄。

38、根据权利要求 34 所述的方法，进一步包括响应于更新通知，更新与该第一用户界面或该第二用户界面关联的过程控制数据。

39、一种用于访问过程控制数据的装置，包括:

处理器系统；和

与该处理器系统通信地连接的存储器，该存储器包括存储的指令，所述指令使得该处理器系统能够:

响应于用户界面请求装载第一和第二客户对象，其中该第一和第二客户对象与访问基于客户模式组织的过程控制数据关联;

装载与该第一和第二客户对象关联的实对象，该实对象被配置用来

获得基于服务器模式组织的过程控制数据；

将过程控制数据从该服务器模式组织映射到该客户模式组织，并将该过程控制数据传送给该第一和第二客户对象；和

通过与该第一客户对象关联的第一用户界面和与该第二客户对象关联的第二用户界面获得该过程控制数据。

40、根据权利要求 39 所述的装置，其中该指令使得该处理器系统能够基于至少一个掩码将该过程控制数据从该服务器模式组织映射到该客户模式组织。

41、根据权利要求 39 所述的装置，其中该指令使得该处理器系统能够在该第一用户界面结束使用该第一客户对象之后将该第一客户对象指定为未活动状态。

42、根据权利要求 41 所述的装置，其中该指令使得该处理器系统能够卸载该第一客户对象句柄。

43、根据权利要求 39 所述的装置，其中该指令使得该处理器系统能够响应于更新通知，更新与该第一用户界面或该第二用户界面关联的过程控制数据。

44、一种机器可访问介质，其上存储有指令，当执行所述指令时使得机器：

响应于用户界面请求装载第一和第二客户对象，其中该第一和第二客户对象与访问基于客户模式组织的过程控制数据关联；

装载与该第一和第二客户对象关联的实对象，该实对象被配置用来获得基于服务器模式组织的过程控制数据；

将过程控制数据从该服务器模式组织映射到该客户模式组织，并将该过程控制数据传送给该第一和第二客户对象；和

通过与该第一客户对象关联的第一用户界面和与该第二客户对象关联的第二用户界面获得该过程控制数据。

45、根据权利要求 44 所述的机器可访问介质，其中当执行该指令时使

得该机器基于至少一个掩码将该过程控制数据从该服务器模式组织映射到该客户模式组织。

46、根据权利要求 44 所述的机器可访问介质，其中当执行该指令时使得该机器在该第一用户界面结束使用该第一客户对象之后将该第一客户对象指定为未活动状态。

47、根据权利要求 46 所述的机器可访问介质，其中当执行该指令时使得该机器卸载该第一客户对象句柄。

48、根据权利要求 44 所述的机器可访问介质，其中当执行该指令时使得该机器响应于更新通知，更新与该第一用户界面或该第二用户界面关联的过程控制数据。

49、一种用于访问过程控制数据的系统，包括:

预先产生的部分类，其包括与访问过程控制数据关联的预先产生的类元素;

用户产生的部分类，其与该预先产生的部分类关联，并且具有能通过该预先产生的类元素来访问过程控制数据的用户定义的类元素;

用户界面，其被配置用来基于该预先产生的部分类和该用户产生的部分类来实例化客户对象，并且被配置用来根据该预先产生的类元素和用户定义的类元素访问过程控制数据；和

客户模型，其被配置用来装载对象句柄和与该客户对象关联的实对象，并且在该客户对象与服务器之间通过该对象句柄和该实对象传送过程控制数据。

50、根据权利要求 49 所述的系统，其中该预先产生的部分类和该用户产生的部分类形成完整类。

51、根据权利要求 49 所述的系统，其中该预先产生的部分类和该用户产生的部分类的该预先产生的类元素和用户产生的类元素通过继承和聚集进行共用。

52、根据权利要求 49 所述的系统，其中该预先产生的部分类和该用户

产生的部分类被配置用来跨名字空间共用该预先产生的类元素和用户定义的类元素。

53、根据权利要求 49 所述的系统，其中该用户界面包括 I/O 控件，其被配置成被数据绑定到与所述预先产生的类元素或用户定义的类元素关联的属性。

54、根据权利要求 49 所述的系统，其中该客户模型被配置用来将与修改的过程控制数据关联的更新信息传送给客户应用程序。

55、根据权利要求 54 所述的系统，其中该用户界面被配置用来确定该更新信息中所指定的任何修改的过程控制数据是否与该客户对象关联。

56、根据权利要求 54 所述的系统，其中如果任何修改的过程控制数据与该客户对象关联，并且如果该用户界面需要使用该客户对象，那么该用户界面被配置用来将该客户句柄标记为已使用过。

57、根据权利要求 49 所述的系统，其中该客户模型被配置用来基于该实对象与多个服务器通信。

58、根据权利要求 57 所述的系统，其中所述多个服务器包括与存储的过程控制数据关联的数据库服务器或者与实时过程控制数据关联的运行期服务器。

59、根据权利要求 49 所述的系统，其中该实对象被配置用来获得基于服务器模式层次结构组织的过程控制数据。

60、根据权利要求 49 所述的系统，其中该客户对象被配置用来获得基于客户模式层次结构组织的过程控制数据。

61、根据权利要求 49 所述的系统，其中该客户模型被配置用来将该过程控制数据从服务器模式层次结构映射到客户模式层次结构。

62、一种用于修改过程控制数据的方法，包括:

获得与修改的过程控制数据相关联的更新通知事件;

根据该更新通知事件识别与该修改的过程控制数据相关联的对象;

检索与该对象相关联的修改的过程控制数据;

确定该修改的过程控制数据是否不同于所装载的与该对象关联的过程控制数据；和

如果该修改的过程控制数据不同于所装载的过程控制数据，那么就根据该修改的过程控制数据来更新与客户应用程序关联的值。

# 用于存取过程控制数据的方法和设备

相关申请

本申请是2004年5月4日申请的、标题为“用于再现、监控过程控制系统并与之交互的图形用户界面”、序列号为60/567,980的美国临时申请的正规申请，并且要求其优先权，并且本申请在此将其全部内容引作参考。本申请也与2003年7月21日申请的、标题为“图形显示元件、过程模块和控制模块在加工厂中的集成”、序列号为10/625,481的美国专利申请相关，并且其公开文本是2004年8月5日的美国专利公开US2004/0153804,其又是2002年10月22日申请的、标题为“加工厂中的智能过程模块和对象”、序列号为10/278,469的美国专利申请的部分后续申请，并且后者的公开文本是2004年4月22日的美国专利公开US2004/0075689,其全部内容都在此引作参考。本申请也与2003年2月18日申请的、标题为“加工厂配置系统中的模块类对象”、序列号为10/368,151的美国专利申请相关，并且其公开文本是2004年10月7日的美国专利公开US2004/0199925，其全部内容在此引作参考。本申请也涉及下列专利申请，它们是在与本申请的申请日期相同申请的国际申请（PCT），并且其全部内容在此引作参考：“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报警和报警趋势”（代理备案号：06005/41112）；“加工厂的过程模块和专家系统集成”（代理备案号：06005/41113）；“在集成化环境中具有用户化过程图形层的加工厂用户界面系统”（代理备案号：06005/41114）；“过程环境中的脚本化图形”（代理备案号：06005/41115）；“过程配置和过程环境中的图形集成”（代理备案号：06005/41116）；“过程环境中具有多个可视功能的图形元素”（代理备案号：06005/41117）；“在加工厂中配置图形显示元件和过程模块的系统”（代理备案号：06005/41118）；“用于统一的过程控制系统界面的图

形显示配置框架”（代理备案号：06005/41124）；“加工厂用户界面中的基于标记语言的动态过程图形”（代理备案号：06005/41127）；“用于修改过程控制数据的方法和装置”（代理备案号：06005/591622 和 20040/59-11622）；“用于过程控制系统的集成化图形运行期界面”（代理备案号：06005/591628 和 20040/59-11628）；以及“用于过程控制系统的面向服务的架构（Service-Hyphen Oriented Architecture for Process Control Systems）”（代理备案号：06005/591629 和 20040/59-11629）。

技术领域

本发明公开总的来说涉及一种过程控制系统，更具体而言，涉及用于访问过程控制数据的过程控制装置和方法。

背景技术

诸如在化学、石油或其它过程中所使用的那些过程控制系统典型地包括一个或多个集中过程控制器，其通过模拟、数字、或者模拟/数字组合总线与至少一个主机或操作员工作站以及一个或多个现场设备通信地连接。该现场设备例如可以是阀门、阀门定位器、开关以及变送器（例如温度、压力和流速传感器），其在该过程中执行诸如开启或关闭阀门以及测量过程参数等功能。该过程控制器接收由现场设备所作出的表示过程测量值的信号以及与现场设备相关的其它信息，并使用该信息来实现控制例程，然后产生控制信号，其通过总线或其它通信线路发送到现场设备以控制该过程的操作。来自该现场设备和该控制器的信息可以由该操作员工作站所执行的一个或多个应用程序使用，以使得操作者能够执行想要的关于该过程的功能，诸如查看该过程的当前状态、修改该过程操作等。

在设计阶段和系统操作过程中，系统工程师必须经常访问过程控制数据，以查看、监视、增加、更新、修改该过程控制数据等。例如，过程控制系统典型地使用配置应用程序来进行配置，其使得工程师、操作者、用户等

能够定义过程控制系统内的每个现场设备对于特定的过程（例如特定的化学生产过程）应该如何操作。当将现场设备加到特定的过程时，或者当每次对该过程进行改变时，工程师可以产生新的控制程序或新的配置数据，或者可以更新或修改现存的控制程序。每个过程可以使用大量的现场设备、控制器和/或其它控制设备，于是控制程序可以包括大量过程控制数据。某些已知过程控制系统提供有编辑器或过程控制数据查看器，其使得用户在操作期间能够监视过程，和/或查看、创建、和/或更新控制程序。已知的过程控制数据编辑器和查看器典型地将用户限制在由过程控制软件开发者所提供的特征。例如，过程控制软件开发者可以调查其客户，以确定理想的用户界面控件的类型和数据访问功能。然后，使在发布过程控制软件时对客户有用的该用户界面和数据访问特征符合其它客户的一般要求，以包括这些特征。

根据客户而定制的过程控制软件通常相对较贵并且项目复杂。具体地说，如果客户需要特定的或定制的用户界面或数据访问特征，该客户需要理解并修改原始的过程控制软件源代码。在这种情况下，过程控制软件售主必须向希望定制他们的软件的每一客户提供许多资源（例如软件开发者、系统工程师、源代码等）。另外，该软件售主在将源代码提供给客户之前可能需要客户购买源代码许可证或开发许可证。资源和/或许可证对于软件售主和/或客户通常都比较贵。而且，通过发布某些源代码，如果该源代码包括商业秘密、机密或其它有竞争优势的编码技术，售主存在风险。

## 发明内容

本文公开了用于访问过程控制系统数据的示例方法和系统。根据一个例子，用于访问过程控制数据的方法包括：装载客户对象，和从该客户对象向被配置为与服务器通信的实对象（real object）传送数据访问请求。该实对象然后根据该数据访问请求向该服务器传送查询，并且响应于该查询而从该服务器获得过程控制数据。然后将该过程控制数据从与服务器模式（schema）关联的第一数据布局映射到与客户模式关联的第二数据布局。然后将所映射

的过程控制数据传送给应用程序。

根据另一例子，用于访问过程控制数据的另一方法包括：响应于用户界面请求装载第一和第二客户对象。该第一和第二客户对象与访问基于客户模式组织的过程控制数据关联。然后装载与该第一和第二客户对象关联的实对象。该第一和第二实对象被配置用来获得基于服务器模式组织的过程控制数据。然后将该过程控制数据从该服务器模式组织映射到该客户模式组织，并将其传送给第一和第二客户对象。然后通过与该第一客户对象关联的第一用户界面和与该第二客户对象关联的第二用户界面获得该过程控制数据。

根据另一例子，用于访问过程控制数据的系统包括：预先产生的部分类和用户产生的部分类。该预先产生的部分类包括与访问过程控制数据关联的预先产生的类元素。该用户产生的部分类与该预先产生的部分类关联，并且包括用户定义的类元素，所述用户定义的类元素可以通过该预先产生的类元素访问过程控制数据。该系统还包括用户界面，其被配置用来基于该预先产生的部分类和该用户产生的部分类来实例化客户对象。该用户界面还被配置用来根据该预先产生的类元素和用户定义的类元素访问过程控制数据。该系统还包括客户模型，其被配置用来装载对象句柄和与该客户对象关联的实对象，并且在该客户对象与服务器之间通过该对象句柄和该实对象传送过程控制数据。

附图说明

图 1 所述的框图为示例的客户/服务器构架，其包括与过程控制系统机器通信地连接的客户机。

图 2 详细描述了图 1 的该客户模型和用户界面的功能框图。

图 3 和 4 描述了示例代码配置，其可以用来通过继承在用户产生的与预先产生的部分类之间共用代码。

图 5 描述了另一个示例代码配置，其可以用来通过聚集在用户产生的与预先产生的部分类之间共用代码。

图 6 描述了预先产生的部分类与具有两个客户模式的客户应用程序的实对象之间的关系。

图 7 和 8 描述了在运行阶段期间，在图 1 和 2 的用户界面与客户模型之间形成的数据路径。

图 9 的方框图描述了图 1 和 2 的用户界面与客户对象之间的数据绑定。

图 10 描述了定义示例服务器模式的示例服务器模式 XML 源代码。

图 11 为该过程控制系统数据库服务器响应于由该客户模型所提交的查询而返回给该客户模型的示例 XML 源代码。

图 12 描述了可以用来将过程控制数据从服务器模式映射到客户模式的示例客户模式 XML 源代码。

图 13 描述了表示对象和包含在其中的角色的示例用户界面。

图 14 的详细框图描述了客户模式与服务器模式之间的映射配置，以产生图 13 的该示例用户界面。

图 15 描述了可以用来产生从图 14 的服务器模式层次结构到客户模式层次结构的映射的示例 XML 源代码。

图 16 描述了示例用户界面，其表示包含功能框和两个属性的复合功能框“PT_COMP”。

图 17 的详细框图描述了服务器模式层次结构与客户模式层次结构之间的映射配置，其将单个客户角色映射到多个服务器角色，以产生图 16 的示例用户界面。

图 18 描述了可以用来产生从图 17 的服务器模式层次结构到客户模式层次结构的映射的示例 XML 源代码。

图 19 描述了表示特定工厂区域内多个不同控制设备的示例用户界面。

图 20 的详细框图描述了服务器模式层次结构与客户模式层次结构之间的映射配置，其将多个客户角色映射到单个服务器角色，以产生图 19 的示例用户界面。

图 21 描述了可以用来产生从图 20 的服务器模式层次结构到客户模式层

次结构的角色映射的示例 XML 源代码。

图 22 描述的示例用户界面可以用来选择性地显示与控制设备关联的项。

图 23 的详细框图描述了服务器模式层次结构与客户模式层次结构之间的映射配置，其将客户对象映射到服务器对象的子集，以产生图 22 的示例用户界面。

图 24 描述了可以用来产生从图 23 的服务器模式层次结构到客户模式层次结构的对象映射的示例 XML 源代码。

图 25 描述的示例用户界面可以用来将附加项插入到控制设备视图中，即使该附加项并不是用于该控制设备的服务器模式的一部分。

图 26 的详细方框图描述了服务器模式层与客户模式层之间的映射配置，其将客户对象插入到该客户模式层。

图 27 描述了用于将客户对象插入到图 26 的该客户模式层的示例 XML 源代码。

图 28 描述的示例用户界面可以用来显示可以通过命令获得的实时过程控制数据的项。

图 29 的详细方框图描述了作为命令实施客户角色的映射配置。

图 30 描述了可以用来作为命令在图 29 的该客户模式层中实施客户角色的示例 XML 源代码。

图 31A 和 31B 所述的流程图是可以用来在运行阶段期间提供通过客户对象对实对象进行客户应用程序访问的示例方法。

图 32 是可以用来更新客户对象中的修改过程控制数据的示例方法。

图 33 是示例处理器系统的方框图，其可以用来实施此处所述的该示例装置、方法和制品。

## 具体实施方式

虽然下面所公开的示例系统包括在硬件上执行的软件和/或固件以及其

它组件，但是应该注意到，这些系统仅仅只是说明性的，并且不应该作为限制考虑。例如，可以考虑将任何或所有这些硬件、软件和固件组件专门通过硬件、专门通过软件、或通过硬件和软件的组合来实施。相应地，虽然下面描述了示例系统，但是本领域的普通技术人员可以容易地认识到，所提供的这些示例并不是实施这些系统的唯一方式。

相比于将终端用户限制为预定义的特征和用于访问过程控制数据和与过程控制数据交互的功能这种已知系统，这里所描述的该示例装置、方法和制品可以使用可定制的数据访问工具来访问过程控制系统服务器中的过程控制数据，其使得终端用户能够定制客户应用程序访问、表示以及显示该过程控制数据的方式。过程控制数据典型地包括与控制系统、过程、材料流和成分、控制系统装备、现场设备、以及用来操作、维护、和诊断整个系统的任何工作显示关联的任何数据或信息。过程控制系统服务器典型地位于加工厂中，并且用来存储过程控制数据。为了自动化、管理和配置过程控制系统，企业典型地使用运行于过程控制系统服务器上的过程控制系统软件（即过程控制系统应用程序），并且根据用户定义的过程控制数据管理与该过程控制系统关联的每个操作。用户（例如系统工程师）可以使用客户应用程序与过程控制数据进行交互（例如管理、查看、修改、配置等），其中客户应用程序与过程控制系统应用程序交换命令、请求和过程控制数据。该客户应用程序典型地可以安装于或运行于与该过程控制系统服务器也连接的网络连接的任何工作站（即任何计算机终端）上。

传统的客户应用程序不管是由终端用户还是由向终端用户提供过程控制软件的软件售主开发，通常都是在与该过程控制系统应用程序同时或与其结合在一起开发。传统的客户应用程序典型地提供固定集合的数据访问和数据处理功能，用户通过这些功能被限制于访问、表示和查看过程控制数据。定制该数据访问和数据处理功能通常是一个复杂并且昂贵的过程，因为其需要修改该过程控制系统应用软件、修改该客户应用软件、重新编译所有软件以及重新测试所有软件。

这里所描述的示例方法、装置和制品提供客户模型数据接口层（例如图1的客户模型116），其提供可定制的客户/服务器数据接口，由此客户应用程序（例如图1的该客户应用程序108）可以通过该数据接口与过程控制系统服务器（例如该过程控制系统数据库服务器112）交换数据。该客户模型116使得客户应用程序从该过程控制系统服务器112中提取出来，并且可以在不同的过程控制系统服务器之间移动。该客户模型116包括核心数据访问或数据交换功能，和一套基本的数据访问和数据处理功能，这些功能使得该客户应用程序108能够与该过程控制系统数据库服务器112通信和交互，以及与其交换过程控制数据。该客户模型116可以在软件开发工具包（SDK）中作为多个目标代码文件、头文件、源文件等提供给终端用户（例如消费者、系统工程师等）。终端用户然后可以基于该客户应用程序SDK开发客户应用程序，以与该过程控制系统服务器交互，从而查看、管理和配置过程控制数据。该终端用户可以在任何时候修改或增加数据访问功能，并且每次可以只重新编译该客户应用程序软件，而不需要修改该过程控制系统应用软件。

如下面将要更详细的描述的，使用与面向对象的编程语言关联的部分类（partial class）来实现该客户模型116。部分类用来将类的类型分离、划分或分割成可以驻留于两个或更多个文件中的两个或更多个部分。通过这种方式，程序员可以根据功能、使用频率或任何其它标准将长代码分割或断开成多个小片断或部分代码。如下所述，该部分类可以用来区分用户产生的代码中和预先产生的代码。相应的部分类可以驻留于目标代码和源代码的任何组合中。预先产生的代码包括用于客户模型116的预先产生的部分类，其在过程控制系统应用程序的初始开发被开发，并被编译以产生例如通过如上所述的客户应用程序SDK发送到终端用户的客户模型目标代码。用户产生的代码包括用户产生的部分类，其对应于该预先产生的部分类，并且被用来为随后开发（例如在销售之后）的客户应用程序定义定制功能。

该终端用户可以通过只使用该客户模型目标代码中所提供的那些功能来开发客户应用软件，或者该终端用户可以随后开发源代码，以定义附加的

用户定义的数据访问功能，从而按照该终端用户的需要来访问、表示、和/或显示过程控制数据。可以使用该客户模型目标代码的对应预先产生的部分类中所定义的任何资源、元素、或功能来在用户产生的部分类中开发该用户定义的数据访问功能。在编译时间期间，编译者扫描软件过程的每个文件（例如每个目标文件和源文件），并且交叉链接相应的部分类以形成定义该类的每个元素、功能、或方面的完整类（complete class）。在执行期间，该客户应用程序 108 将所组合的对应部分类识别为完整类，从而使得用户定义的数据访问功能能够与该客户模型 116 一起工作，就好像该用户定义的数据访问功能最初就是该客户模型目标代码的一部分。

该客户模型 116 也使得用户能够定义该过程控制数据布局，或者定义当从过程控制系统数据库（例如图 1 的该过程控制系统数据库 110）中检索出该过程控制数据时如何表示该过程控制数据。该过程控制系统数据库 110 使用表、列、记录、项、字段等组织过程控制数据。当该过程控制系统数据库服务器 112 从过程控制系统数据中检索过程控制数据时，该服务器根据服务器模式组织该过程控制数据。然而，客户应用程序通常需要使该过程控制数据进行不同组织、表示、或布局，从而该客户应用程序可以按照用户定义来显示该过程控制数据。为了便于通过客户应用程序访问和显示数据，终端用户可以在该客户应用程序的设计阶段为每一客户应用程序定义客户模式。在操作期间，该客户模型 116 可以从该过程控制系统数据库服务器 110 中获得按照服务器模式组织或设置的过程控制数据，并按照下面结合图 13 至 30 所述根据该用户定义的客户模式将该过程控制数据从该服务器模式组织重新设置成客户模式组织或设置。

现在详细参见图 1，客户/服务器架构 110 包括客户机 102 和过程控制系统机 104。该客户机 102 一般用来查看、修改、和管理与过程控制数据相关的任何过程控制数据。该客户机 102 可以使用计算机、工作站终端、便携计算机、膝上计算机、手持个人数字助理（PDA）、或任何其它适当的处理器系统来实现。该过程控制系统机 104 存储该过程控制数据并根据该过程控制

数据对该过程控制系统进行自动化和管理。该过程控制系统机 104 可以是工作站、主机、服务器、或与过程控制系统中的控制设备通信地连接的任何其它适当的处理器系统（例如图 33 的示例处理器系统 3310）。该过程控制系统机 104 被配置用来将该过程控制数据提供给该客户机 102（或被配置用来与该过程控制系统机 104 通信的任何其它客户机），并且被配置用来按照该客户机 102 和/或对应的过程控制系统中的控制设备的请求修改、增加、或更新过程控制数据。

该客户机 102 可以通过通信网络 106 与该过程控制系统机 104 通信地连接。该通信网络 106 例如可以是局域网（LAN）或广域网（WAN），并且可以使用任何适当的通信技术或者各种技术的组合来实现，诸如 Ethernet、IEEE 802.11、Bluetooth®、任何数字或模拟移动通信系统(即蜂窝通信系统)、数字用户线（DSL）、任何宽带通信系统等。

该客户机 102 包括让用户能够检索、查看、管理和存储过程控制数据的客户应用程序 108。用户可以在实现客户应用程序 108 的客户机 102 上安装机器可访问的指令，并随后使用客户应用程序 108 来访问所存储的过程控制数据和/或实时过程控制数据。所存储的过程控制数据可以包括与控制设备配置参数相关联的信息、以周期性间隔测得的过程控制数据值、历史测量值、或可以被存储用于以后检索的任何其它值。总的来说，实时过程控制数据包括不是存储、而是根据请求所产生或得出的任何过程控制数据。例如，实时过程控制数据可以包括响应于来自该客户应用程序 108 的请求而测得、获得、产生、或计算得到的过程控制数据值。

所存储的过程控制数据可以从过程控制系统机 104 中获得。过程控制系统机 104 包括被配置用来存储过程控制数据（例如所存储的过程控制数据）的过程控制系统数据库 110，和与该过程控制系统数据库 110 通信地连接的过程控制系统数据库服务器 112。该过程控制系统数据库服务器 112 被配置用来在该过程控制系统数据库 110 与客户应用程序 108 之间传送所存储的过程控制数据。

客户机 102 包括运行期（runtime）服务器 114，以提供实时过程控制数据。该运行期服务器 114 可以与该网络 106 通信地连接，并且被配置用来获得从该数据库服务器 112 和/或直接从过程控制系统中的控制设备获得过程控制数据。例如，运行期服务器 114 可以通过从该数据库服务器 112 请求过程控制数据，并且对所检索到的过程控制数据执行例如数学运算或任何其它操作，从而根据一个或多个所存储的过程控制数据得到实时过程控制数据。如果该客户应用程序 108 请求与控制设备关联的实时测得的过程控制数据值（例如温度值、压力值、流速值等），那么运行期服务器 114 可以通过过程控制系统机 104 和/或该网络 106 与该控制设备通信，以检索实时测得的该过程控制数据值。

该客户应用程序 108 包括该客户模型 116 和用户界面 118。该客户模型 116 与过程控制系统机 104 和运行期服务器 110 通信地连接，并使得该客户应用程序 108 能够与该过程控制系统机 104 和该运行期服务器 110 通信，以访问所存储的过程控制数据和实时过程控制数据。具体地说，该客户模型 116 提供可以被该客户应用程序 108 使用的数据访问功能，以访问和交换所存储的以及实时的过程控制数据。该数据访问功能包括一套基本的数据访问功能，并且也可以包括用户定义的数据访问功能。如下结合图 2 的详细所述，通过预先产生的部分类提供该套基本数据访问功能，并且通过与该预先产生的部分类对应的用户产生的部分类提供该用户定义的数据访问功能。

该用户界面 118 被配置用来产生多个基于图形和/或基于文本的用户界面屏幕，其可以用来对过程控制数据进行访问、查看、管理、修改、更新等。用户可以在设计阶段和/或在运行阶段指定由该用户界面 118 所要使用的显示布局或显示设置，以显示该过程控制数据。例如，所示的该用户界面 118 包括树视图界面 120 和内容视图界面 122。该树视图界面 120 可以用来以分级树结构来显示过程控制数据，该结构具有可以查看所选控制设备的更少或更多详情的扩展和折叠部分。该内容视图界面 122 可以用来显示层叠到过程控制系统图上的过程控制数据。例如，该内容视图界面 122 可以显示过程控

制系统图中的彼此通信地连接的多个控制设备，并且显示与相应控制设备相邻或其上的过程控制数据。通过这种方式，用户可以在整个过程控制系统的范围内查看过程控制数据。

客户模型 116 在该用户界面 118、运行期服务器 114 和过程控制系统数据库服务器 114 之间根据模式、查询和命令传送过程控制数据。模式用来定义特定的数据组织、布置、或应该如何表示过程控制数据的数据布局。例如，客户模式为该用户界面 118 定义特定的数据布置或用来表示过程控制数据的数据布局。该用户界面 118 也与多个客户模式关联，每个客户模式都用来布置、组织、或表示不同的过程控制数据。例如，一个客户模式可以用来表示泵控制设备数据，而另一客户模式可以用来表示多个控制设备共用的属性值，而还有的客户模式可以用来表示与特定工厂区域内的控制设备关联的过程控制数据。

服务器模式为过程控制系统数据库服务器 112 和运行期服务器 114 定义用来表示或布置过程控制数据的特定数据布置或数据布局。用于该过程控制系统数据库服务器 112 的服务器模式可以与用于运行期服务器 114 的服务器模式不同。然而，总的来说，服务器模式用来与客户模式不同地表示、组织、或布置数据。例如，服务器模式可以用来表示与过程控制系统关联的所有过程控制数据，而客户模式可以用来仅表示该过程控制数据的特定部分或片断。

该客户模型 116 被配置用来在服务器模式与客户模式之间转换或映射过程控制数据，如下面结合图 13 至 30 详述的那样。例如，响应于来自该用户界面 118 的数据请求查询，该客户模型 116 将该过程控制数据从服务器模式转换或映射到客户模式，以根据由该用户界面 118 所提供的该客户模式布置该处理数据。该客户模型 116 也可以响应于来自该客户界面 118 的更新查询，将修改或更新后的过程控制数据从客户模式转换或映射到服务器模式。

由该用户界面 118 和/或该客户模型 116 所产生的查询可以包括数据请求查询和更新查询。该数据请求查询用来检索特定的过程控制数据，该更新

查询用来修改或更新例如该过程控制系统数据库 110 中的过程控制数据。响应于接收到来自该用户界面 118 的查询，该客户模型 110 确定该查询是否与所存储的过程控制数据或者与实时过程控制数据关联，并相应地将该查询传送到该过程控制系统数据库服务器 112 或该运行期服务器 114。如果该查询包括与所存储的和实时的过程控制数据关联的部分，该客户模型 116 可以解析该查询，或将其分解成实时数据查询和存储数据查询，并分别将这些查询传送到该服务器 112 和 114。

该命令可以包括使得服务器 112 和 114 检索、修改和/或创建过程控制数据的机器可访问指令。例如，某些命令可以包括与根据这些查询和更新查询来访问（例如检索、修改或创建）该过程控制系统数据库 110 中的过程控制数据相关联的指令。另外，某些命令可以包括使得运行期服务器 114 从控制设备测量或获取过程控制数据，或者使得该运行期服务器 114 根据所存储的过程控制数据得出过程控制数据值（例如平均值、滤波值等）的指令。

图 2 是图 1 的客户模型 116 和用户界面 118 的详细功能框图。具体地说，图 2 描述了在运行阶段期间使用部分类来在该用户界面 118、该客户模型 116 以及该服务器 112 和 114 之间交换过程控制数据的方式。该用户界面 118 使用多个客户模式（例如图 14、17、20、23、26 和 29 的该多个客户模式分级层）来寻址不同的过程控制数据。例如，所示的该用户界面 118 包括第一客户模式对象模型 202a、第二客户模式对象模型 202b 和第三客户模式对象模型 202c，其中每个对象模型与各自的第一、第二和第三客户模式关联。用户界面 118 也包括用户 I/O 控件 204，其被配置用来显示用户界面（UI）视图（例如图 1 的树视图 120 和内容视图 122）中的过程控制数据，并且用来获得与检索、显示、和/或修改过程控制数据相关联的用户输入。该用户 I/O 控件 204 例如可以包括文本框、按钮、列表、数据域等，并且可以使用任何适当的控件框架来实现，例如可以包括 Microsoft® Avalon 控件框架。

每个客户模式对象模型 202a-c 包括一个或多个预先产生的部分类 206 和一个或多个用户产生的部分类 208，其能够对与各个第一、第二和第三客

户模式相关联的过程控制数据进行访问和处理。具体地说，该预先产生的部分类 206 包括预定义的类元素，而该用户产生的部分类 208 包括用户定义的类元素。类元素可以包括数据成员、存取器、方法或功能、实现、和/或本领域所熟知的任何其它类元素，每个类元素都可以被指定为私有的、受保护的或公有的。该预先产生的部分类 206 的类元素可以用来通过如下结合图 6 详述的实对象（例如下面所述的该实对象 216）与该过程控制系统数据库服务器 112（图 1 和 2）通信。该用户产生的部分类 208 的类元素可以被配置用来访问与相应的预先产生的部分类 206 的类元素关联的数据，并且可以被配置用来通过该预先产生的部分类 206 中的类元素与该过程控制系统数据库服务器 112 通信。如图 2 中所示，该用户产生的部分类 108 包括用户定义的功能 210。由对应的该部分类 206 和 208 形成的每个完整类可以与不同类型的过程控制数据关联。例如，示例的类可以包括与特定类型的控制设备关联的数据访问和处理功能，而另一示例的类可以包括与特定的加工厂区域或特定的过程控制子系统关联的数据访问和处理功能。还有的示例类可以包括与对过程控制数据进行数学和/或统计处理（例如平均、滤波等）关联的功能。

在开发阶段期间，对于每个客户模式，终端用户可以选择一个或多个预先产生的部分类来创建该每个预先产生的部分类 206。而且在开发阶段期间，该终端用户可以开发该用户定义的功能 210，并通过选择与该用户定义的功能 210 相关联的一个或多个用户产生的部分类来创建每个客户模式对象模型 202a-c 的用户产生的部分类 208。

为第一客户模式对象模型 202a 所选择的部分类 206 和 208 可以不同于为第二客户模式对象模型 202b 所选择的部分类 206 和 208。例如，为第一客户模式对象模型 202a 选择的部分类 206 和 208 可以用来访问与第一加工厂区域的控制设备关联的过程控制数据，而为第二客户模式对象模型 202b 选择的部分类 206 和 208 可以用来访问与第二加工厂区域的控制设备关联的过程控制数据。

在运行阶段期间，部分类 206 和 208 可以用来为每个客户模式对象模型 202a-c 产生多个客户对象 212。该客户对象 212 符合或对应于该客户应用程序（例如图 1 的客户应用程序 108）的数据布局或数据部置。每个客户对象 212 是由部分类 206 和 208 之一所定义的类的类型。可以产生该相同类的类型的两个或更多个客户对象 212，诸如两个类的类型泵的对象，分别用于该过程控制系统中不同的物理泵控制设备。另外，可以产生两个或更多个客户对象 212，以访问与该同一物理控制设备关联的过程控制数据。由两个或更多个该客户对象 212 对该同一物理控制设备的过程控制数据的访问通过实对象（例如该实对象 216）在该客户模型 116 中进行仲裁或处理，如下所述的那样。该客户对象 212 可以用来将该用户 I/O 控件 204 数据绑定到实时和存储的过程控制数据，如下结合图 9 详述的那样。通过将该用户 I/O 控件 204 数据绑定到该过程控制数据，该客户对象 212 可以响应于通过该用户 I/O 控件所提供的用户输入，产生数据请求查询和/或数据更新查询。如上所述，使用这些查询来检索或修改所存储的或实时的过程控制数据。该客户对象 212 也可以响应于指示例如该过程控制系统数据库 210 中的至少某些过程控制数据值已经改变或被修改的数据更新事件，来更新通过该用户 I/O 控件 204 显示的过程控制数据值。

该客户模型 116 包括多个第一对象句柄 214a、多个第二对象句柄 214b、和多个第三对象句柄 214c。如图 2 中所示，多个对象句柄 214a-c 中的每个对象句柄与客户模式对象模型 202a-c 中的相应对象模型关联。该客户模型 116 也包括多个实对象 216。该实对象 216 符合或对应于与图 1 的过程控制系统数据库服务器 114 和/或运行期服务器 114 关联的服务器模式的该数据布局或数据布置。该对象句柄 214a-c 是存储器中对应于实对象 216 的位置的地址参考或基本地址。在运行期间，当在存储器堆中创建并存储一个该实对象 216 时，在存储器栈上也创建并存储句柄 214a-c 中的一个对应句柄。通过该实对象的对象句柄（例如，句柄 214a-c 中的一个句柄）向该实对象发送数据访问请求、查询、更新查询、和/或过程控制数据，每个客户对象

212 与一个实对象 216 关联，并通过一个实对象 216 访问过程控制数据。该实对象 216 响应于从该客户对象 212 接收到的查询或更新查询来将查询和/或更新查询传送到该过程控制系统数据库服务器 112。该实对象 216 从该过程控制系统数据库服务器 112 获得基于服务器模式（例如，图 14、17、20、23、26 和 29 的服务器模式层次结构中的一个）而组织或布置的过程控制数据。该客户模型 116 然后将该过程控制数据从服务器模式组织映射、重新布置或转换到客户模式组织，如下结合图 13 至 29 中详述的那样。

在某些情况下，两个或更多个客户对象 212 对应于特定的一个实对象 216。例如，当从相同类型的类构建两个或更多个客户对象 212 来访问该相同的过程控制数据时，就为这两个或更多个客户对象 212 创建单个实对象 216。通过这种方式，一个实对象 216 可以仲裁由两个或更多个实对象 216 对该相同的过程控制数据所作出的数据访问请求。

图 3 和 4 描述了示例的代码配置，其可以用来通过继承在用户产生的部分类与预先产生的部分类之间共用代码。图 3 显示的预先产生的文件 302 的名称为“MODULE_GEN.CS”，第一用户产生的文件 304 的名称为“MODULE_MANUAL.CS”，第二用户产生的文件 306 的名称为“MODULE_BASE.CS”。该预先产生的文件 302 定义名称空间“DELTAV.CONFIG.EXPLORER.HIERARCHY”308 中的部分类，其包含类型“MODULE”310 的预先产生的公有部分类（即该预先产生的公有部分类“MODULE”310）。该公有部分类“MODULE”310 可以包括被配置用来与图 1 和 2 的该过程控制系统数据库服务器 112 通信的类元素。该公有部分类“MODULE”310 可以是图 2 的预先产生的部分类 206 的一部分。

该第一用户产生的文件 304 使用名称空间“DELTAV.CONFIG.EXPLORER”312，以及名称空间“DELTAV.CONFIG.EXPLORER.HIERARCHY”310 中的部分类定义。在名称空间“DELTAV.CONFIG.EXPLORER.HIERARCHY”310 中，该预先产生的公有部分类“MODULE”310 继承该用户产生的类“MODULE_BASE”314 的类元素。通过这种方式，该

类“MODULE”310和“MODULE_BASE”314可以共用或访问彼此的类元素。例如，在类“MODULE_BASE”314中定义的类元素可以通过类“MODULE”310中所定义的类元素与该过程控制系统数据库服务器112通信。类“MODULE_BASE”314在下面所描述的第二用户产生的文件306中定义，并且其可以是图2的用户产生的部分类208的一部分。

第二用户产生的文件306包括可以用来配置客户应用程序(例如图1的该客户应用程序108)的源代码，以在预先产生的部分类“MODULE”310与用户产生的部分类“MODULE_BASE”314之间共用源代码。在该第二用户产生的文件306中，该名称空间“DELTAV.CONFIG. EXPLORER”312包含用于用户产生的部分类“MODULE_BASE”314的特定应用方法的实现。该用户产生的部分类“MODULE_BASE”314的特定应用方法由第一用户产生的文件304中的用户产生的部分类“MODULE”310继承。

图4描述的示例运行配置用来在运行阶段期间在两个不同的名称空间之间共用图3的源代码。图4显示了名称空间为“DELTAV.CONFIG.EXPLORER”312的实例、名称空间为“DELTAV.CONFIG.EXPLORER.HIERARCHY”308的实例和名称空间为“DELTAV.CONFIG.EXPLORER.CONTENT”402的实例。在名称空间308或402中装载或实例化的部分类的类型“MODULE”310的客户对象可以使用在该预先产生的文件302或用户产生的文件306中定义的源代码或类元素。

名称空间“DELTAV.CONFIG.EXPLORER”312的实例包括用于该用户产生的部分类“MODULE_BASE”314的第二用户产生的文件306(图3)中所定义的类元素。名称空间“DELTAV.CONFIG.EXPLORER.HIERARCHY”308的实例包括类的类型“MODULE”(例如部分类的类型“MODULE”310)的第一“MODULE”客户对象404。名称空间“DELTAV.CONFIG. EXPLORER.CONTENT”402的实例包括第二“MODULE”客户对象406，其也是类类型“MODULE”(例如该部分类型“MODULE”310)。每个客户对象404和406包括用户产生的部分410(例如用户产生

的类元素）和预先产生的部分412（例如预先产生的类元素）。该用户产生的部分410包括在该第二用户产生的文件304的用户产生的类“MODEL_BASE”310中所定义的类元素。该预先产生的部分412包括在该预先产生的文件302的用户产生的部分类型类“MODULE”310中所定义的类元素。

图5描述了另一个示例代码配置，其可以用来通过聚集来在用户产生的与预先产生的部分类之间共用代码。预先产生的部分类代码502包括名称空间“DLETAV.CONFIG.EXPLORER.HIERARCHY”308，其包含预先产生的部分类“MODULE”310。用户产生的部分类代码504包括名称空间“DLETAV.CONFIG.EXPLORER.HIERARCHY”312，其包含用户产生的类“EXPLORER_MODULE”506。为了在该预先产生的部分类“MODULE”310与该用户产生的类“EXPLORER_MODULE”506之间共用源代码或类元素，该用户产生的类“EXPLORER_MODULE”506定义“MODULE”508（例如部分类类型“MODULE”310）类型的客户对象，其可以被该预先产生的部分类代码502中所定义的预先产生的部分类“MODULE”310的所有类元素使用。

图6描述了预先产生的部分类与具有两个客户模式的客户应用程序（例如图1的该客户应用程序108）的实对象之间的关系。第一客户模式602使用多个第一预先产生的部分类604，并且第二客户模式606使用多个第二预先产生的部分类608。该预先产生的部分类604和608基本上与图2的预先产生的部分类206类似或相同。该客户应用程序108使用两个客户模式602和606来寻址不同的需要或不同的过程控制数据。例如，该第一客户模式602与特定工厂区域下的模块关联，并且该第二客户模式606与特定子系统的报警关联。

该客户模型116包括与该第一客户模式602关联的多个第一对象句柄610和与该第二客户模式606关联的多个第二对象句柄612。该对象句柄610和612基本上与图2的对象句柄214a-c类似或相同。该客户模型116为每

个客户模式 602 和 606 建造或产生对象句柄的单独的组或树。每个预先产生的部分类 604 和 608 中预先产生的部分类的层次关系与对应的对象句柄 610 和 612 的层次关系相同。

该客户模型 116 还包括基本上与图 2 的实对象 216 类似或相同的多个实对象 614。该客户应用程序 108 通过与预先产生的部分类 604 和 608 关联的客户对象（例如图 2 的客户对象 212 和图 7 的 710a-d）与实对象 614 通信。具体地说，与该部分类 604 和 608 关联的客户对象通过该对象句柄 610 和 612 访问该实对象 614。

当该客户应用程序 108 从第一客户模式 602 发送数据访问请求（例如查询）到客户模型 116 时，该客户模型 116 装载与特定过程控制数据（例如特定的工厂区域）关联的实对象（例如某些实对象 614），其中该数据访问请求与上述特定过程控制数据关联。当该客户应用程序 108 从该第二客户模式 606 发送数据访问请求（例如查询）时，该客户模型 116 装载所有报警属性或与该特定子系统关联的一个实对象 614 的属性，如图 6 中所示，其中该数据访问请求与上述特定过程控制数据关联。

该客户用于 108 控制该实对象 614 的寿命或装载该实对象 614 的时间。例如，如果该客户应用程序 108 指示某些实对象 614 应该保持被装载，那么这些对象就不会被卸载。典型地，通过垃圾收集例程卸载诸如该实对象 614 之类的对象，其中垃圾收集例程周期性地检测已经被指定为可垃圾收集或可卸载的实对象。例如，如果该客户应用程序 118 确定应该卸载某些实对象 614，那么那些实对象就被指定可垃圾收集。在这种情况下，随后的垃圾收集扫描就会卸载被标记为可垃圾收集的或者没有被标记为当前使用的这些实对象 614。

图 7 和 8 描述了在运行阶段期间，在图 1 和 2 的该用户界面 108 与该客户模型 116 之间形成的数据路径。特别地，图 7 描述了该客户对象 212 与用于第一用户界面（UI）视图 702 和第二 UI 视图 704 的实对象 216 之间的数据路径。在图 7 中，该 UI 视图 702 和 704 共用一个共同的客户模式。图 8

描述了客户对象 212 与实对象 216 之间的数据路径，其中第一 UI 视图 702 和第二 UI 视图 704 分别具有其自己的客户模式。UI 视图 702 和 704 可以基本上与图 1 的树视图 120 和/或内容视图 122 类似或相同。

如图 7 和 8 中所示，第一和第二 UI 视图 702 和 704 属于客户应用对象 706 或与其关联。每个 UI 视图 702 和 704 通过一个或更多个实对象 216 访问过程控制数据。所示实对象 216 具有父实对象“R”708a，其装载“ModuleLibrary”实角色 708b。诸如“ModuleLibrary”实角色 708b 之类的角色暴露于与该角色关联的多个过程控制数据，或提供对与该角色关联的多个过程控制数据的访问。例如，角色可以提供对与特定工厂区域相关联的，或者与特定类型的控制设备相关联的过程控制数据的访问。该“ModuleLibrary”实角色 708b 响应于来自 UI 视图 702 和 704 的过程控制数据请求，装载第一实对象“R1”708c 和第二实对象“R2”708d。例如，第一实对象 708c 可以与第一控制设备关联，而第二实对象 708d 可以与第二控制设备关联。

为了通过该实对象 216 访问过程控制数据，该 UI 视图 702 和 704 装载该客户对象 212，以与该实对象 216 通信。例如，如图 7 中所示，为了访问与该“ModuleLibrary”实角色 708b 关联的过程控制数据，该第一 UI 视图 702 装载父客户对象“O”710a。该父客户对象“O”710a 引用父对象句柄“H”712a，该父客户对象 710a 使用父对象句柄“H”712a，以通过第零个掩码“M”704a 访问该父实对象“R”708a。诸如第零个掩码“M”714a 之类的掩码被配置用来将过程控制数据从服务器模式的数据布局或布置翻译或映射到客户模式的数据布局或布置。例如，父实对象 708a 可以是提供对多个过程控制数据文件库的访问的库对象。如果父客户对象 710a 被配置用来只访问与该父实对象 708a 相关联的多个库对象（例如，该“ModuleLibrary”实角色 708b）的子集，那么掩码 714a 将库访问请求从父客户对象 710a 翻译或映射到父实对象 708a 中对应的库。例如，掩码可以包括布置在布局中的多个指针，其对应于客户模式，并且指向或者引用实对象中的过程控制数据。

所示该数据路径为多个活动和未活动的数据路径。实线表示活动的或者UI视图702或704使用的以访问特定对象、句柄或掩码的数据路径。虚线表示未活动的或者UI视图702或704都不再使用的数据路径。例如，起始第一UI视图702使得父客户对象“O”710a实例化或装载第一子客户对象“O1”710b。第一UI视图702可以使用第一子客户对象“O1”710b，以通过第一实对象708c来访问过程控制数据。该第一UI视图702通过第一数据路径716访问该第一子客户对象710b，其起始是表示活动的或“在使用中”的数据路径的实线。该第一子客户对象710b引用第一句柄“H1”712b，并通过第一掩码“M1”714b访问第一实对象708c。

另外，由于第一和第二UI视图702和704共用公用的客户模式，该第二UI视图704也可以通过该第一子客户对象710b访问该第一实对象708c。如图7中所示，该第二UI视图704通过第二数据路径718访问该第一子客户对象710b。当该第一UI视图702结束使用该第一子客户对象710b时，该第一数据路径716变成图7中通过虚线所示的未活动。如果该客户应用程序706没有其它UI视图需要访问该第一子客户对象706b，那么客户对象710b变为可以进行垃圾收集(例如可卸载)。然而，因为该第二UI视图704继续访问该第一子客户对象710b，该数据路径718保持为图7中用实线表示的活动状态，并且该客户对象710b并不变为可用于垃圾收集。从该第一子客户对象710b到该第一实对象708c的数据路径720、722和724也保持为用实线所示的活动状态。

当客户对象不再被客户应用程序的任何用户界面使用时，与该客户对象关联的所有数据路径变为未活动，并且该客户对象和对应的句柄和掩码都被标记为准备用于垃圾收集，并且随后由该客户模型116卸载。另外，如果被该客户对象访问的实对象不再被其它任何客户对象访问时，那么该实对象也被标记为准备用于垃圾收集，并且随后由客户模型116卸载。例如，开始为该第一UI视图702初始化或装载以访问该第二实对象708d的第二子客户对象“O2”710c，当其不再被任何UI视图702或704使用时随后变为未活动

状态并被标记为准备用于垃圾收集。第二对象句柄 714c 和第二掩码 714c 也变成被标记为用于垃圾收集。在这种情况下，与该第二子客户对象 710c 关联的所有数据路径都变为图 7 中用虚线所示的未活动状态。虽然该客户模型 116 卸载第二子客户对象 710c、第二对象句柄 712c、以及第二掩码 714c，但是如果另一个子客户对象仍然还在使用或者还请求访问该第二实对象 708d 时，该客户模型 116 可以不卸载该第二实对象 708d。

在特定客户对象的实例被卸载之后，可以为随后任何访问与该客户对象关联的过程控制数据装载该客户对象的另一个实例。例如，如图 7 中所示，在卸载该第二子客户对象 710c 之后（例如在垃圾收集期间被卸载），该第一 UI 视图 702 可以使得该父对象 710a 装载或实例化该第二子客户对象“O2” 710d 的第二个实例，以访问第二实对象 708d。

在图 8 中，每个 UI 视图 702 和 704 都有其自己单独的客户模式，并且使用其自己的子客户对象来访问该实对象 216。例如，第二 UI 视图 704 实例化或装载客户对象 “O1” 802，其与被第一 UI 视图 702 所使用的所有客户对象都分离。下面结合图 31A 和 31B 所述的该示例方法可以用来在 UI 视图 702 和 704 与图 7 和 8 中所示的实对象 216 之间形成通信路径。

图 9 的框图描述了在图 1 和 2 的该用户界面 118 与基本上与图 2 的一个客户对象 212 类似或相同的客户对象 902 之间的数据绑定。图 9 中该用户界面 118 举例来说具有基本上与图 2 的该用户 I/O 控件 204 相同或类似的第一、第二和第三用户 I/O 控件 904a、904b 和 904c。例如，该用户 I/O 控件 904a-c 可以是文本框、列表框、数据域、复选框等。该客户对象 902 包括用户产生的类元素部分 906 和预先产生的类元素部分 908。该用户产生的元素部分 906 在用户产生的部分类（例如图 2 的一个用户产生的部分类 208）中定义，并且该预先产生的类元素部分 908 的类元素在预先产生的部分类（例如图 2 的一个预先产生的部分类 206）中定义。根据该用户产生的和预先产生的部分 906 和 908 的类元素创建下面所述的多个属性 910a－910e。

该用户产生的部分 907 包括 PROPERTY_A 元素 910a 和 PROPERTY_B

元素 910b。该预先产生的部分 908 包括 PROPERTY_C 元素 910c、PROPERTY_D 元素 910d 和 PROPERTY_E 元素 910e。使用属性元素 910a-c 来从该过程控制系统数据库 110 中检索所存储的过程控制数据。PROPERTY_A 元素 910a 与 PROPERTY_C 元素 910c 关联，并且基于该 PROPERTY_C 元素 910c 得出值。该客户对象 902 包括如图 9 中所示的私有哈希表 912，其包括用户产生的元素（例如属性元素 910a 和 910b）与预先产生的元素（例如属性元素 910c-e）之间的该映射或关联。可以通过对于对象 912 私有的类型“HashTable”的变量或指针来引用或访问该私有哈希表 912。在该私有哈希表 912 的行 914 中所示的为 PROPERTY_A 元素 910a 与 PROPERTY_C 元素 910c 之间的关联。

该用户 I/O 控件 904a-c 分别被数据绑定到 PROPERTY_A 元素 910a、PROPERTY_D 元素 910d 和 PROPERTY_E 元素 910e。通过这种方式来数据绑定该 I/O 控件 904a-c 就使得每次当任何这些数据值被更新时，该客户应用程序 108 在该 I/O 控件 904a-c 与它们各自的元素 910a、910c 和 910e 之间传送数据值。例如，如下结合图 32 更加详细所述的，该过程控制系统数据库服务器 112 和客户模型 116 可以交换关于过程控制系统数据库服务器 112 中已经被修改的过程控制数据的更新信息（例如“UpdateNotification”事件和可疑或脏对象列表）。通过这种方式，客户对象 902 可以确定任何修改的过程控制数据是否与任何其属性元素 910c-d 关联，其中属性元素 910c-d 用来访问所存储的过程控制数据，如果是，那么就更新过程控制数据值被修改的对应的每个属性元素 910c-d 的值。该客户对象 902 然后可以使用该私有哈希表 912 来更新使用或与过程控制数据值被修改的对应的任何属性元素 910c-d 关联的任何用户产生的属性（例如该属性 910a 和 910b）的值。

为了更新该数据绑定的用户 I/O 控件 904a-c 的值，该客户模型 116 解析通过更新通知事件所接收到的更新信息，以产生用于其对应的过程控制数据值已经被修改的任何用户产生的或预先产生的属性的“PropertyChanged”事件。该“PropertyChanged”事件然后使得该数据绑定的用户 I/O 控件 904a-c

从各自的属性 910a-c 中获得修改的过程控制数据，并更新该用户 I/O 控件 904a-c 的值。

该客户对象 902 也可以使用该更新信息来产生脏哈希表 916。该脏哈希表 916 通过类型“HashTable”的变量或指针来引用，并且对于对象 902 是私有的。该脏哈希表 916 用来存储客户应用程序（例如图 1 的客户应用程序 108）的脏的或可疑的对象句柄。在该更新信息中提供该脏对象句柄，并且该脏对象句柄指示某些客户对象的过程控制数据已经处于过程控制系统数据库 110 中。该客户应用程序 108 可以使用“PopulateDirtyHashTable”函数来产生具有该脏对象句柄的脏哈希表 916。例如，该“PopulateDirtyHashTable”函数首先接收来自该更新通知事件的更新信息，然后解析该更新信息和产生具有脏句柄的该脏哈希表 916。

图 10 描述了定义示例服务器模式的示例服务器模式 XML 源代码 1000。该服务器模式 XML 源代码 1000 存储在该过程控制系统数据库服务器 112 中，并且定义用来通过该过程控制系统数据库服务器 112 表示过程控制数据的布置或布局。该服务器模式 XML 源代码 1000 包含类类型定义和枚举定义。每个类类型具有名称，并包含多个属性、角色和命令。每个属性具有名称和数据类型。

角色具有名称和其包含的类类型。角色中所包含的用于项目的基本类型在角色元素中声明。特定类的子类型被嵌套在该角色元素中。每个所包含的类类型被标记其是否可以通过命令由客户应用程序（例如图 1 的该客户应用程序 108）创建。类类型定义还包含可以通过例如从该客户应用程序 108 获得的命令脚本来执行的命令。每个命令具有名称并定义其参数和返回类型。

如图 10 中所示，该服务器模式 XML 源代码 1000 指定的类型名称为“ControlModule”（线 1002）。该类型名称“ControlModule”（线 1002）包含名称为“detailDisplayName”的数据类型为字符串的属性（线 1004）和名称为“Blocks”的目的类型“BlockBase”的角色（线 1006）。该角色名称“Blcoks”（线 1006）包含可以通过该客户应用程序 108 创建的名称为

“ModelUsage”的类型。该类型名称“ModelUsage”（线 1008）包含当客户应用程序 108 装载或实例化对象类型“ModelUsage”时创建的多个创建参数（线 1010 和 1012）。该类型名称“ControlModule”（线 1002）还包含名称为“renameTo”的返回类型为空的命令（线 1014）。该命令名称“renameTo”（线 1014）包含名称为“newName”的数据类型为字符串的参数（线 1016）。

该服务器模式 XML 源代码 1000 也指定一枚举定义。具体地说，该服务器模式 XML 源代码 1000 包括名称为“DbeAttributeType”的枚举（线 1018）。该枚举名称“DbeAttributeType”（线 1018）包含包括名称为“Float”的条目（线 1020）和名称为“FloatWithStatus”的条目（线 1022）的多个项。

可以在属性的数据类型字段中引用枚举。例如，类型名称“Attribute”（线 1024）包含名称为“attributeType”的属性，其数据类型为“DbeAttributeType”（线 1026），该数据类型对应于枚举名称“DbeAttributeType”（线 1018）。

图 11 为由过程控制系统数据库服务器 112 响应于由客户模型 116 提交的查询而返回给客户模型 116 的示例 XML 源代码 1000。具体地说，响应于该查询 Site.PlantAreas[name='AREA_A'](index).Modules(na me)，返回该示例 XML 源代码 1000。如图 11 中所示，该 XML 源代码 1000 中的结果包括名称为“PlantAreas”的 ModelRole（线 1102）。该 ModelRole 名称“PlantAreas”（线 1102）包含名称为“AREA_A”的 PlantArea（线 1104）。该 PlantArea 名称“AREA_A”（线 1104）包含被设置为零的属性索引（1106）和名称为“Models”的 ModelRole（线 1108）。该 ModelRole 名称“Models”（线 1108）包含多个模块和每个模块的对应属性。例如，ModelRole 名称“Models”（线 1108）包含名称为“EM1”的 ModuleInstanceBase（线 1110），其包含名称为“EM1”的属性（线 1112）。

图 12 描述了可以用来将过程控制数据从服务器模式（例如图 10 的服务器模式 XML 源代码 1000 或下面结合图 14、17、20、23、26 和 29 所述的服务器模式层次中的一个）映射到客户模式（例如下面结合图 14、17、20、

23、26 和 29 所述的客户模式层次中的一个）的示例客户模式 XML 源代码 1200。客户模式按照基本上与服务器模式定义类型相似的方式来定义属性、角色和命令的类型。对于每个类型、属性、和角色，客户模式 XML 源代码（例如客户模式 XML 源代码 1200）指示了到服务器模式的映射，从而客户模式 116（图 1）可以在客户模式与服务器模式之间重新布置或映射过程控制数据。

如图 12 中所示，客户模式中名称为“Module”的类型被映射到服务器模式中名称为“Module”的类型（线 1202）。在这种情况下，客户模式中名称为“Module”的该类型在服务器模式中名称也为“Module”。然而，客户模式中的类型名称可以映射到服务器模式中具有不同名称的类型名称。例如，类型名称“ModuleOne”可以映射到类型名称“Module Two”。

属性元素（线 1204）可以包含一个或多个属性元素。每个属性元素定义该客户类型的一个或多个客户属性，并且包含每个客户属性的名称、该客户属性与其关联的该域、以及到服务器模式的映射。如图 12 中所示，属性名称“desc”与该数据库域关联（例如该属性对应于存储在该过程控制系统数据库 110 中的存储过程控制数据），并且被映射到服务器模式中名称为“description”（线 1206）的属性。属性名称“rtValue”与运行域关联（例如该属性对应于可以从图 1 的运行期服务器 114 中获得的实时过程控制数据），并被映射到服务器模式中名称为“ST”（线 1208）的属性。该属性映射与该包含的类型有关。例如，如果包含的类型为“Module”，其位于包含的类型“Site”内，那么产生用来检索用于 MOD_X 的“desc”属性的查询为 Site.Modules[name='MOD_X'](description)。

角色元素（线 1210）包含一个或多个 Role 元素。每个 Role 元素定义关联客户类型的客户角色，并包含该客户角色的名称、该客户角色关联的域、用来得到该客户角色的映射、以及该客户角色的对象类型。如图 12 中所示，该角色元素（线 1210）包含与该数据库域关联的名称为“attributes”的角色（例如该角色对应于存储在该过程控制系统数据库 110 中的存储过程控制

数据），并且被映射到服务器模式中名称为“Attributes”的角色（线 1212）。该属性映射与该包含的类型有关。例如，如果包含的类型为“Module”，其位于该包含的类型“Site”内，那么产生用来检索用于 MOD_X 的“attributes”角色的查询为 Site.Modules[name ='MOD_X'].Attributes。

图 13 至 29 描述了示例用户界面、模式映射配置、和示例 XML 源代码，XML 源代码可以用来将诸如对象和角色的类元素从驻留于服务器（例如图 1 的该过程控制系统数据库服务器 112 或该运行期服务器 114）上的服务器模式映射到驻留于客户应用程序（例如图 1 的该客户应用程序 108）中的客户模式。下面描述的该映射配置可以通过客户模型 116 来实现，以从过程控制系统服务器中提取客户应用程序。通过将类元素通过该客户模型 116 从服务器模式映射到客户模式，可以配置客户应用程序来与多个服务器通信地连接并与其一起工作。进一步，下面描述的基于使用模式映射配置的客户模型 116 对客户应用程序进行开发可以使得分别具有特定过程控制数据需要的多个应用程序能够与过程控制系统服务器通信地连接并与其一起工作。

图 13 描述了表示对象和包含在其中的角色的示例用户界面 1300。该示例用户界面 1300 所示的单元模块“AREA_A”1302 作为一个完整对象。然而，在过程控制系统数据库 110（图 10）中，作为两个分开的对象存储该单元模块“AREA_A”1302。第一对象包含该单元模块“AREA_A”1302 的行为，第二对象包含定义可以如何在单元模块“AREA_A”1302 中嵌套其它模块的模块容器行为。

图 14 的详细框图描述了客户模式与服务器模式之间的映射配置 1400，以产生图 13 的示例用户界面 1300。可以通过客户模型 116（图 1）来实现映射配置 1400。具体地说，图 14 显示了服务器模式层次 1402、客户模式层次 1404、和一对一映射配置形式的客户模型层次 1406。该服务器模式层次 1402 指示了存储在过程控制系统数据库 110（图 1）中的“Unit”1408 类型的服务器对象与 “UnitModule”1410 类型的服务器对象之间的区别。该客户模型层次 1406 的层次符合该服务器模式层次 1402 的层次，或基本上与其

相似或与其相同。客户模式层次 1404 通过使用将指针提供到该客户模型层 1406 中的掩码 1410a、1410b 和 1410c 来仅表示服务器模式层 1402 中所关心的那些对象和角色。

服务器模型层 1402 包括类型“AREA”的服务器对象 1412，该服务器对象包含类型“Units”的服务器角色 1414。该服务器角色“Units”1414 指向服务器对象“Unit”1408，其包含类型“UnitModule”的服务器角色 1418 和类型“Modules”的服务器角色 1420。服务器角色“UnitModule”1418 指向该服务器对象“UnitModule”1410，并且该服务器角色“Modules”1420 指向类型“Module”的服务器角色 1422。

客户模式层 1404 包括类型“AREA”的客户对象 1424，该客户对象包含类型“UnitModules”的客户角色 1426。该客户角色“UnitModules”1426 指向类型“UnitModules”的客户对象 1428，其包含类型“Modules”的客户角色 1430。客户角色“Modules”1430 指向类型“Module”的客户对象。该客户对象和客户角色通过下面结合图 15 所描述的客户模型层次 1406 的实对象和角色以及掩码 1410a-c 被映射到该服务器对象和服务器角色。

客户模型层次 1406 包括多个实对象和实角色，其设置基本上与该服务器模式层次 1402 相似或相同。客户模型层次 1406 包括对应于该服务器对象“AREA”1412 的类型“AREA_A”的实对象 1434。该实对象“AREA_A”1434 包含对应于该服务器角色“Units”1414 的类型“Units”的实角色 1436。该实角色“Units”1436 指向对应于该服务器对象“Unit”1408 的类型“ABC”的实对象 1438。该实对象“ABC”1438 包含对应于该服务器角色“UnitModule”1418 的类型“UnitModule”的实角色 1440，和对应于该服务器角色“Module”1420 的类型“Modules”的实角色 1442。该实角色“UnitModule”1440 指向对应于该服务器对象“UnitModule”1410 的实对象“ABC”1444。该实“Module”1442 指向对应于该服务器对象“Module”1442 的实对象“SSS”1446。

图 15 描述了可以用来产生从图 14 的服务器模式层次 1402 到客户模式

层次 1404 的映射的示例 XML 源代码 1500。该 XML 源代码 1500 显示了该客户对象“AREA” 1424（图 14）被映射到该服务器对象“AREA” 1412（图 14）（线 1502）。为了将该客户对象“AREA”1424 映射到该服务器对象“AREA” 1412，该客户模型 116 通过该掩码“AREA_A” 1410a 将该实对象“AREA_A” 1434 映射到该客户对象“AREA” 1424。通过这种方式，该客户对象“AREA” 1424 通过该实对象“AREA_A” 1434 访问该服务器对象“AREA” 1412，就好像是该客户对象“AREA” 1424 在直接访问该服务器对象“AREA” 1412。

客户对象“AREA” 1424 包含客户角色“UnitModules” 1426（图 14），其映射到该服务器角色“Units” 1414（图 14）并且指向该客户对象“UnitModules” 1428（图 14）（线 1504）。该客户模式层次 1404 中的客户对象可以具有与该服务器模式层次 1402 中的服务器对象相同的名称，即使该客户对象并不映射到该相同名称的服务器对象。例如，该客户对象“UnitModules” 1428 被映射到该服务器对象“Unit” 1408（图 14）（线 1506）。在这种情况下，客户模型 116 将客户对象“UnitModules” 1428 通过该掩码“ABC” 1410b 映射到实对象“ABC” 1438。

客户对象“UnitModules” 1428 包含名称为“scanRate” 的属性，其映射到该服务器对象“UnitModules” 1410 的“scanRate” 属性（即‘UnitModule（scanRate）’）（线 1508）。在这种情况下，该客户模型 116 将实对象“ABC” 1444 的“scanRate” 通过该掩码“ABC” 1410b 映射到客户对象“UnitModule” 1428 的“scanRate” 属性。通过将实对象“ABC” 1444 的“scanRate” 属性通过该掩码“ABC” 1410b 映射到客户对象“UnitModule” 1428 的“scanRate” 属性，该客户模型 116 从该服务器对象“Unit” 1408 遍历该服务器角色“Unitmodule”1418（图 14），以将该客户对象“UnitModule”1428 的“scanRate” 属性映射到该服务器对象“UnitModules” 1410 的“scanRate” 属性。

对象“UnitModules” 1426 包含客户角色“Models” 1430（图 14），其映射到服务器角色“Modules” 1420（图 14）并且指向该客户对象“Modules” 1432（图 14）（线 1510）。

图16描述的示例用户界面1600表示包含功能框和两个属性的复合功能框“PT_COMP”1602。如该用户界面1600中所示，该复合功能框“PT_COMP”1602包括包含功能框“CALC1”1606、属性“ABS_PRESS_CF”1608、和属性“ABS_TEMP_CF”1610的项的统一列表1604。如果用户选择显示对象的统一列表，该客户模型116可以被配置用来将客户角色映射到两个或多个服务器角色。如下结合图17所述，在服务器模式（例如图17的该服务器模式层次1702）中，对应于该功能框“CALC1”1606的对象关联于与对应于属性“ABS_PRESS_CF”1608和属性“ABS_TEMP_CF”1610的对象不同的服务器角色。然而，在客户模式（例如图17的客户模式层次1704）中，对应于该功能框“CALC1”1606、属性“ABS_PRESS_CF”1608和属性“ABS_TEMP_CF”1610的对象被表示作为该同一客户角色的一部分。

图17的详细框图描述了服务器模式层次1702与客户模式层次1704之间的映射配置1700，其将单个客户角色映射到多个服务器角色，以产生图16的该示例用户界面1600。在该映射配置中所描述的映射可以由客户模型116(图1)执行。映射配置1700通过客户模型层次1706和多个掩码1708a-d将该客户模式层次1704映射到该服务器模式层次1702。该服务器模式层次1702包括类型“Composite Function Block”的服务器对象1710，其包含两个服务器角色：类型“Blocks”的服务器角色1712和类型“Attributes”的服务器角色1714。该服务器角色“Blocks”1712指向类型“BlockUsage”的客户对象1716，并且该服务器角色“Attributes”1714指向类型“AttributeDefinitions”的客户对象1718。如图所示，该客户模型层次1706符合该服务器模式层次1702。

该客户模式层1704包括类型“Composite”的客户对象1720，其包含类型“Children”的客户角色1722。该客户角色“Children”1722指向类型“Blocks”的客户对象1724和类型“Attributes”的客户对象1726。

图18描述了可以用来产生从图17的服务器模式层次1702到客户模式层1704的映射的示例XML源代码1800。该XML源代码1800将客户对象

“Composite”1720(图17)映射到该服务器对象“Composite Function Block”1710(图17)(线1802)。包含在该客户对象“Composite”1720(线1804)内的该客户角色“Children”1722映射到两个服务器角色。特别地，该客户角色“Children”1722映射到该服务器角色“Blocks”1712(线1806)，并映射到该服务器角色“Attributes”1714(线1808)。为了映射到该服务器角色“Blocks”1712，该客户角色“Children”1722指向该客户对象“Block”1724(线1806)。为了映射到该服务器角色“Attributes”1714，该客户角色“Children”1722指向该客户对象“Attribute”1726(线1808)。该客户角色“Block”1724映射到该服务器对象“BlockUsage”1716(线1810)，并且该客户角色“Attribute”1726映射到该服务器对象“AttributeDefinition”1718(线1812)。

图19描述了表示特定工厂区域“AREA_A”1902内多个不同控制设备的示例用户界面1900。具体地说，该工厂区域“AREA_A”1902包括“LIC-549”控制设备1904、“PLM1”控制设备1906、和“EM1”控制设备1908。如下结合图20和21所述，该控制设备1904、1906和1908在客户模式(例如图20的该客户模式层次2004)中表示为映射到服务器模式(例如图20的该服务器模式层次2002)中的单个服务器角色的三个独立的客户角色。

图20的详细框图描述了服务器模式层次2002与客户模式层次2004之间的映射配置2000，其将单个客户角色映射到多个客户角色，以产生图19的示例用户界面1900。该映射配置2000通过客户模型层次2006和多个掩码2008a-d将该客户模式层次2004映射到该服务器模式层次2002。该服务器模式层次2002包括类型“Area”的服务器对象2010，其包含类型“Modules”的服务器角色2012。该服务器角色“Modules”2012指向类型“Phase Logic Module”的服务器对象2014、类型“Equipment Module”的服务器对象2016、和类型“ControlModule”的服务器对象2018。该客户模型层次2006包括类型“PLM1”的实对象2020、类型“EM1”的实对象2022、和类型“LIC-459”的实对象2024，其分别对应于服务器对象“Phase Logic Module”2014、服

务器对象“Equipment Module”2016、以及服务器对象“Control Module”2018。该实对象“PLM1”2020、实对象“EM1”2022、以及实对象“LIC-459”也分别对应于该用户界面 1900 中所示的该“LIC-549”控制设备 1904、“PLM1”控制设备 1906、和“EM1”控制设备 1908。

该客户模式层次 2004 包括类型“Area”的客户对象 2026，其指向类型“ControlModules”的客户角色 2028、类型“PhaseLogicModules”的客户角色 2030、和类型“EquipmentModules”的客户角色 2032。该客户角色“ControlModules”2028 指向类型“Control Module”的客户对象 2034。该客户角色“PhaseLogicModules”2030 指向类型“Phase Logic Module”的客户对象 2036。该客户角色“EquipmentModules”2032 指向类型“Equipment Module”的客户对象 2038。

图 21 描述了可以用来产生从图 20 的该服务器模式层次 2002 到客户模式层次 2004 的角色映射的示例 XML 源代码 2100。如该 XML 源代码 2100 中所示，该客户角色“ControlModules”2028 被映射到该服务器角色“Modules”2006（线 2102）。而且，该客户角色“PhaseLogicModules”2030 被映射到该服务器角色“Modules”2006（线 2104）。另外，该客户角色“EquipmentModules”2032 被映射到该服务器角色“Modules”2006（线 2106）。

图 22 描述的示例用户界面 2200 可以用来选择性地显示与控制设备关联的项。例如，该用户界面 2200 示出了“CH01”控制设备 2202 并仅显示了一个属性，即属性“CTLR1C01CH01”2004，其该“CH01”控制设备 2202 关联，尽管该“CH01”控制设备 2202 与服务器模式（例如图 23 的服务器模式层次 2302）中的至少两个属性关联。如下进一步详述的，客户模式（例如图 23 的客户模式层次 2304）可以被映射到服务器模式（例如服务器模式层次 2302）中的服务器对象的子集，使得客户应用程序（例如图 1 的客户应用程序 108）仅显示如用户界面 2200 中所示对象的该子集。

图 23 的详细框图描述了服务器模式层次 2302 与客户模式层次 2304 之间的映射配置 2300，其将客户对象映射到服务器对象的子集，以产生图 22

的该示例用户界面 2200。客户模式层次 2306 包括类型“CTLR1C01”的实对象 2308，其对应于图 22 的属性“CTLR1C01CH01”2204。该服务器模式层次 2302 包括类型“Device”的服务器对象 2310 和类型“DST”的服务器对象 2312。该客户模式层次 2304 包括类型“DST”的客户对象 2314。如图 24 的该示例 XML 源代码 2400 中所示，该客户对象“DST”2314 被映射到该服务器对象“DST”2312（线 2402）。该客户对象“DST”2314 通过该实对象“CTLR1C01CH01”2308 和掩码“CTLR1C01CH01”2316 被映射到该服务器对象“DST”2312。

图 25 描述的示例用户界面 2500 可以用来将附加项插入到控制设备视图中，即使该附加项并不是用于该控制设备的服务器模式的一部分。该用户界面 2500 包括“CTLR1”控制设备 2502，其所示具有“Assigned Modules”项 2504 和“TIC-205”项 2506。如下面结合图 26 所述的，服务器模式（例如图 26 的服务器模式 2602）并不包括对应于该“Assigned Modules”项 2504 的服务器对象。相反，对应于该“Assigned Modules”项 2504 的客户对象被插入到客户模式（例如图 26 的该客户模式 2604）中。

图 26 的详细框图描述了服务器模式层次 2602 与客户模式层次 2604 之间的映射配置 2600，其将客户对象插入到该客户模式层次 2604 中。服务器模式层次 2602 和客户模式层次 2604 中的每个都包括单个对象。具体而言，服务器模式层次 2602 包括类型“Module”的服务器对象 2608，并且客户模型层次 2606 包括对应于该服务器对象“Module”2608 和该“TIC-205”项 2506（图 25）的类型“TIC-201”的实对象 2610。该客户模式层次 2604 包括对应于该服务器对象 2608 和客户模型对象 2610 的客户对象“Module”2612。另外，客户应用程序（例如图 1 的客户应用程序 108）已经插入类型“AssignedModules”的客户对象 2614。

图 27 描述了用于将客户对象“AssignedModules”2614（图 26）插入到客户模式层次 2604（图 26）的示例 XML 源代码 2700。具体地说，为了创建该客户对象“Assigned Modules”2614，该示例 XML 源代码 2700 规定：

类型“AssignedModules”的客户角色2616（图26）不具有服务器模式映射（例如 Role name=”AssignedModules” mapping=“ ”），并且该客户角色“AssignedModules”2616 指向该客户对象“AssignedModules”2614（线2702）。为了将该客户对象“AssignedModules”2614 插入到该客户模式层次2604，该示例XML源代码2700规定：该客户对象“Assigned Modules”2614假装映射到类型“Control”的服务器对象2618（例如该服务器模式层次2602的父对象）。该客户对象“AssignedModules”2614包含类型“Modules”的客户角色2620（图26），其映射到类型“AssignedModules”的服务器角色2622（图26）（线2706）。

图28描述的示例用户界面2800可以用来显示可以通过命令为其获得实时过程控制数据的项。该用户界面2800包括现场总线端口“P01”2802，其包含可以从其中通过该运行期服务器 114 获得实时过程控制数据的控制设备（图1）。具体地说，该现场总线端口“P01”2802包含“Decommissioned Fieldbus Devices”2804，其包括控制设备“D1”2806。该控制设备“D1”2806在客户模式（例如图29的该客户模式层2902）中作为如下结合图29所述的命令来实现。

图29的详细框图描述了作为命令实现的客户角色的映射配置。客户模式层次2902包括作为命令实现的类型“Devices”的客户角色2904。该客户角色“Devices”2904可以用来从图1的运行期服务器114中获得实时过程控制数据。图30的示例XML源代码3000规定：类型“DecommissionedDevices”的客户对象2906（图29）（图30的线30）包含该客户角色“Devices”2904（图29）。该示例XML源代码3000还规定：使用存储在文件“commands.dll”（例如assembly=”commands.dll”）中的命令指令来实现该客户角色“Devices”2904，并且规定实现客户角色“Devices”2904的指定命令是“GetDecommissionedDevices”（线3004）。该客户角色“Devices”2904指向类型“FieldbusDevices”的客户对象2908（图29）（图30的线3004），其并不映射到服务器模式层次2901（图29）（图30的线

3006）。

图 31A、31B 和 32 所述的流程图表示用于实现图 1 和 2 的示例客户应用程序 108、示例客户模型 116、和示例用户界面 118 的示例机器可读并可执行的指令。在这些例子中，该机器可读指令包括用于由处理器执行的程序，诸如图 33 的该示例处理器系统 3310 中所示的处理器 3312。该程序可以实施在存储于有形介质上的软件中，其中有形介质诸如 CD-ROM、软盘、硬盘驱动、数字万能盘（DVD）、或与处理器 3312 相关联的存储器，和/或该程序可以以已知的方式实施于固件或专用硬件中。例如，图 1 和 2 的示例客户应用程序 108、示例客户模型 116 和示例用户界面 118 内的任何或所有结构都可以通过软件、硬件和/或固件实现。进一步，虽然参照图 31A、31B 和 32 中所述的流程图描述了该示例程序，但是本领域的普通技术人员会容易认识到，可替换地可以使用许多其它方法来实现图 1 和 2 的示例客户应用程序 108、示例客户模型 116、和示例用户界面 118。例如，这些方框的执行顺序可以改变，和/或可以改变、消除或组合某些所述方框。

图 31A 和 31B 所述的流程图是可以用来在运行阶段期间提供通过客户对象（例如图 2、7 和 8 的客户对象 212）对实对象（例如图 2、7 和 8 的实对象 216）进行客户应用程序（例如图 1 的客户应用程序 108）访问的示例方法。起始，该客户应用程序 108 引用第一和第二 UI 视图（例如图 7 的第一和第二 UI 视图 702 和 704）（方框 3102）。该第一 UI 视图 702 然后请求通过父客户对象（例如图 7 的父客户对象“O” 710a）以及对应的父对象句柄和掩码（例如图 7 的父对象句柄“H” 712a 和第零个掩码“M” 714a）访问父实对象（例如图 7 的父实对象“R” 708a）（方框 3104）。例如，该第一 UI 视图 702 可以发送调用来使用该父客户对象“O” 710a。该父客户对象“O” 710a 然后可以通过引用该父对象句柄“H” 712a 进行响应，该父对象句柄又引用该第零个掩码“M” 714a。该第零个掩码“M” 714a 然后引用该父实对象“R” 708a，并且在该第一 UI 视图 702 与该父实对象“R” 708a 之间建立通信路径，如图 7 中所示。

该第一 UI 视图 702 然后使得该父实对象“R” 708a 装载实角色（例如图 7 的“ModuleLibrary” 实角色 708b）（方框 3106）。例如，该第一 UI 视图 702 调用该父客户对象“O” 710a 上的装载角色并传送该角色名称“ModuleLibrary”。该父客户对象“O” 710a 然后从该父对象句柄“H” 712a 中提取该实角色，并且该父对象句柄“H” 712a 使得该第零个掩码“M” 714a 装载该“ModuleLibrary” 实角色 708b。

该父实对象“R” 708a 然后装载该“ModuleLibrary” 实角色 708b 下的第一和第二实对象 708c 和 708d（方框 3108）。该第零个掩码“M” 714a 然后为第一和第二实对象 708c 和 708d 创建掩码（例如图 7 的第一和第二掩码 714b 和 714c），并将这些掩码返回到该父对象句柄“H” 712a（方框 3110）。该父对象句柄“H” 712a 然后为该第一和第二掩码 714b 和 714c 创建对象句柄（例如图 7 的第一和第二对象句柄 712b 和 712c），并将对象句柄返回到该父客户对象“O” 710a（方框 3112）。

该父客户对象“O”710a 然后实例化第一和第二子客户对象 710b 和 710c（图 7），并将每个子客户对象 710a-b 引用到对应的对象句柄（方框 3114）。具体地说，该第一子客户对象 710b 被引用到该第一对象句柄 712b，并且该第二子客户对象 710c 被引用到该第二对象句柄 712c。该第一 UI 视图 702 然后引用该第一和第二子客户对象 710b 和 710c（方框 3116），这就形成了如图 7 中所示第一 UI 视图 702 与第一和第二实对象 708c 和 708d 之间的通信路径。

如图 31B 中所示，该第二 UI 视图然后请求访问该第一实对象 708c（方框 3118）。该客户模型 116 然后确定该第一和第二视图 702 和 704 是否共用客户模式（例如图 14、17、20、23、26 和 29 中的客户模式层次中的一个）（方框 3120）。如果该 UI 视图 702 和 704 的确共用客户模式，那么该第一 UI 视图 702 将第一子客户对象 710b 传送到第二 UI 视图 704（方框 3122）。在这种情况下，如图 7 中所示，在该第二 UI 视图 704 与该第一子客户对象 710b 之间建立数据路径，并且该第一 UI 视图 702 与该第一子客户对象 710b

之间的数据路径未被激活，因为该第一 UI 视图 702 不再引用该第一子客户对象 710b。

如果该第一和第二 UI 视图 702 和 704 并不共用客户模式，那么该第一 UI 视图 702 将该第二 UI 视图 704 传送到对应于该第一子客户对象 710b 的服务器路径（方框 3124）。该第二 UI 视图 704 然后将该服务器路径传送到该客户模型 116，并发送请求给该客户模型 116 来查找该服务器路径上的项（方框 31260）。该客户模式 116 然后创建第三掩码 804（图 8）和第三对象句柄 806（图 8），其引用该第一实对象 708c（方框 3128）。该第二 UI 视图 704 然后从该客户模型 116 接收该第三对象句柄 804（图 8）（方框 3130）。该第二 UI 视图 704 然后实例化该第三子客户对象 804（方框 3132）。该第三子客户对象 804 然后引用该第三对象句柄 804（方框 3134）。然后在该第二 UI 视图 704 与该第一实对象 708c 之间形成通信路径，如图 8 中所示。当该第一 UI 视图 702 结束访问该第一实对象 708c 时，该客户模型指示该第一子客户对象 710b、第一对象句柄 712b 和第一掩码 714b 不再被使用或未活动，并准备用于垃圾收集（方框 3136）。

在该客户模型 116 指示哪些客户对象、句柄和掩码准备用于垃圾收集之后，或者在方框 3122 该第一 UI 视图 702 将该第一子客户对象 710b 传送到该第二 UI 视图 704 之后，该客户模型 116 确定该第一 UI 视图 702 是否需要再次访问该第一实对象 708c（方框 3138）。如果该第一 UI 视图 702 的确需要再次访问该第一实对象 708c，那么就将控制传送回到方框 3104（图 31A），并且重复上面结合方框 3104 至 3116 所述的操作，以在该第一 UI 视图 702 与该第一实对象 708c 之间建立通信路径。在这种情况下，如图 7 中所示，通过该第一子客户对象“O1’”的第二实例 710e、第一对象句柄“H1’”的第二实例 712d、和第一掩码“M1’”的第二实例 714d 建立从该第一 UI 视图 702 到第一实对象 708c 之间的通信路径。如果该客户模型 116 在方框 3138 确定该第一 UI 视图 702 并不需要再次访问该第一实对象 708c，那么该过程就结束。

图 32 是可以用来更新客户对象（例如图 9 中的客户对象 902）中的修改的过程控制数据的示例方法。初始，该客户模型 116（图 1）获得更新通知事件（方框 3202）。该客户模型 116 从该过程控制系统数据库服务器 112 和/或该过程控制系统数据库 110 中接收该更新通知事件。例如，当修改了该过程控制系统数据库 110 中的存储过程控制数据时，该过程控制系统数据库 110 发布更新通知事件（例如“UpdateNotification”事件）和可疑实对象或脏实对象（例如图 2 和图 7 中与该数据库 110 中被修改的过程控制数据相关联的某些实对象 216）的列表。响应于该“UpdateNotification”事件，该客户模型 116 识别一个或多个可疑客户对象（例如图 2 和图 7 中的某些客户对象 212），其对应于通过该“UpdateNotification”事件提供给该客户模型 116 的可疑实对象（方框 3204）。该客户模型 116 然后例如通过“QuestionableUpdateNotification”事件将可疑客户对象的列表传送到客户应用程序（例如图 1 的客户应用程序 108）（方框 3206）。该可疑客户对象的列表可以包括每个可疑客户对象的句柄或标识。

该客户应用程序 108 然后从该客户模型 116 接收该可疑客户对象的列表，并根据该可疑客户对象的列表产生脏哈希表（例如图 9 的该脏哈希表 916）（方框 3208）。例如，该客户应用程序 108 的预先产生的部分类（例如图 2 的预先产生的部分类 206）可以包括用来监视该“QuestionableUpdateNotification”事件和可疑客户对象的列表的方法或事件。该客户应用程序 108 然后通知任何可视 UI 视图（例如图 7 的 UI 视图 702 和 704 以及图 9 的 UI 视图 118）：已经接收到更新通知事件（方框 3210）。例如，为了减少更新用户可以看到的过程控制数据所需要的时间量，该客户应用程序 108 可以不告知最小化在用户界面显示器上的任何 UI 视图。该可视 UI 视图然后根据该脏哈希表 916 确定它们这些客户对象是否有脏的（方框 3212）。例如，该可视 UI 视图的客户对象确定任何他们这些句柄或标识符是否对应于该脏哈希表 916 中的可疑客户对象的句柄或标识符。

该可视 UI 视图然后标志出处于该脏哈希表 916 中的任何使用的对象句

柄（方框 3124）。例如，如果该 UI 视图 118 仍然在使用该客户对象 902，并且对应于该客户对象 902 的对象句柄在该脏哈希表 916 中，那么该对象 902 标志出该脏哈希表 916 中其对应的对象句柄，以指示该客户对象 902 仍然“在使用”，并从而不准备用于垃圾收集。该脏哈希表 916 中任何没有被标志的对象句柄都被标定或指定为未活动或未被使用，并可以用于垃圾收集。该客户模型 116 然后删除或卸载所有未活动或未被使用的脏对象句柄（方框 3216）。例如，该客户模型 116 可以使用垃圾收集例程来删除或卸载该脏哈希表 916 中所列出的、但是没有被标志为在使用或活动的所有脏对象句柄。

该客户模型 116 然后为该脏哈希表 916 中被标志为在使用的脏对象句柄检索更新的或修改的过程控制数据（方框 3218）。例如，该客户模型 116 产生对应于“在使用”的脏对象的过程控制数据的查询，并将该查询传送到过程控制系统数据库服务器 112（图 1）。该过程控制系统数据库服务器 112 然后从该过程控制系统数据库 110 检索该修改的过程控制数据，并将该修改的过程控制数据返回到该客户模型 116。

该客户模型 116 然后将该更新的或修改的过程控制数据从服务器模式转换到客户模式（方框 3220）。例如，该客户模型 116 可以使用上面结合图 14、17、20、23、26 和 29 所述的任何映射配置来将该修改的过程控制数据从该服务器模式（例如图 14、17、20、23、26 和 29 的服务器模式层次中的一个）转换或映射到客户模式（例如图 14、17、20、23、26 和 29 的客户模式层次中的一个）。

该客户模型 116 然后确定哪些当前为该可疑客户对象所装载的过程控制数据值不同于所接收到的修改的过程控制数据，并只将不同的修改的过程控制数据（例如实际上已经改变的数据）传送到该客户应用程序（方框 3222）。通过这种方式，该客户模型 116 并不必将标志为修改的、但是实际上与当前装载在客户应用程序 108 中的过程控制数据没有变化的任何过程控制数据传送到客户应用程序 108。该客户应用程序 108 然后将更新通知发送到与已

使用的脏对象句柄关联的、并且从客户模式 116 接收到修改的过程控制数据的客户对象（方框 3224）。例如，该客户应用程序 108 可以发送通知到该客户对象 902，并传送与任何 PROPERTY_C 元素 910c、PROPERTY_D 元素 910d、以及 PROPERTY_E 元素 910e（图 9）关联的修改的过程控制数据。每个客户对象然后根据所接收到的修改的过程控制数据更新其数据（方框 3226）。例如，该客户对象 902 可以根据所接收到的修改的过程控制数据更新元素 910c、910d 和 910e 的值。

图 33 是示例处理器系统的框图，其可以用来实施此处所述的示例装置、方法和制品。如图 33 中所示，该处理器系统 3310 包括与互连总线 3314 连接的处理器 3312。该处理器 3312 包括寄存器组或寄存器空间 3316，其在图 33 中所示为与芯片一体，但是其可替换地可以全部或部分不在芯片上，并通过专用电子连接和/或通过互连总线 3314 与该处理器 3312 直接连接。该处理器 3312 可以是任何适当的处理器、处理单元或微处理器。虽然在图 33 中没有示出，但是该系统 3310 可以是多处理器系统，并于是可以包括一个或多个附加处理器，其与该处理器 3312 相似或相同，并与该互连总线 3314 通信地连接。

图 33 的处理器 3312 与芯片组 3318 连接，其包括存储器控制器 3320 和输入/输出（I/O）控制器 3322。如所熟知的，芯片组典型地提供 I/O 和存储器管理功能以及可以被一个或多个与该芯片组 3318 连接的处理器访问或使用的多个通用和/或专用的寄存器、定时器等。该存储器控制器 3320 执行使得处理器 3312（或如果有多个处理器就是多个处理器）能够访问系统存储器 3324 和大容量存储器 3325 的功能。

该系统存储器 3324 可以包括任何所想要类型的易失性和/或非易失性存储器，诸如静态随机存取存储器（SRAM）、动态随机存取存储器（DRAM）、快速存储器、只读存储器（ROM）等。该大容量存储器 3325 可以包括任何所想要类型的大容量存储设备，包括硬盘驱动器、光盘驱动器、磁带存储设备等。

该 I/O 控制器 3322 执行使得处理器 3312 能够通过 I/O 总线 3332 与外围输入/输出（I/O）设备 3326 和 3328 以及网络接口 3330 通信的功能。该 I/O 设备 3326 和 3328 可以是任何想要类型的 I/O 设备，诸如键盘、视频显示器和监视器、鼠标等。该网络接口 3330 例如可以是使得该处理器系统 3310 能够与另一个处理器系统通信的以太网设备、异步传输模式（ATM）设备、802.11 设备、DSL 调制解调器、线缆调制解调器、蜂窝调制解调器等。

虽然在图 33 中该存储器控制器 3320 和 I/O 控制器 3322 所示为该芯片组 3318 内的不同功能块，但是由这些功能块所执行的功能可以集成在单个半导体电路内，或者可以使用两个或更多个分开的集成电路来实现。

虽然这里已经描述了某些方法、装置以及制品，但是本专利的覆盖范围并不限于此。相反，本专利范围覆盖落入所附权利要求书字面意思或等同物的范围内的所有方法、装置和制品。

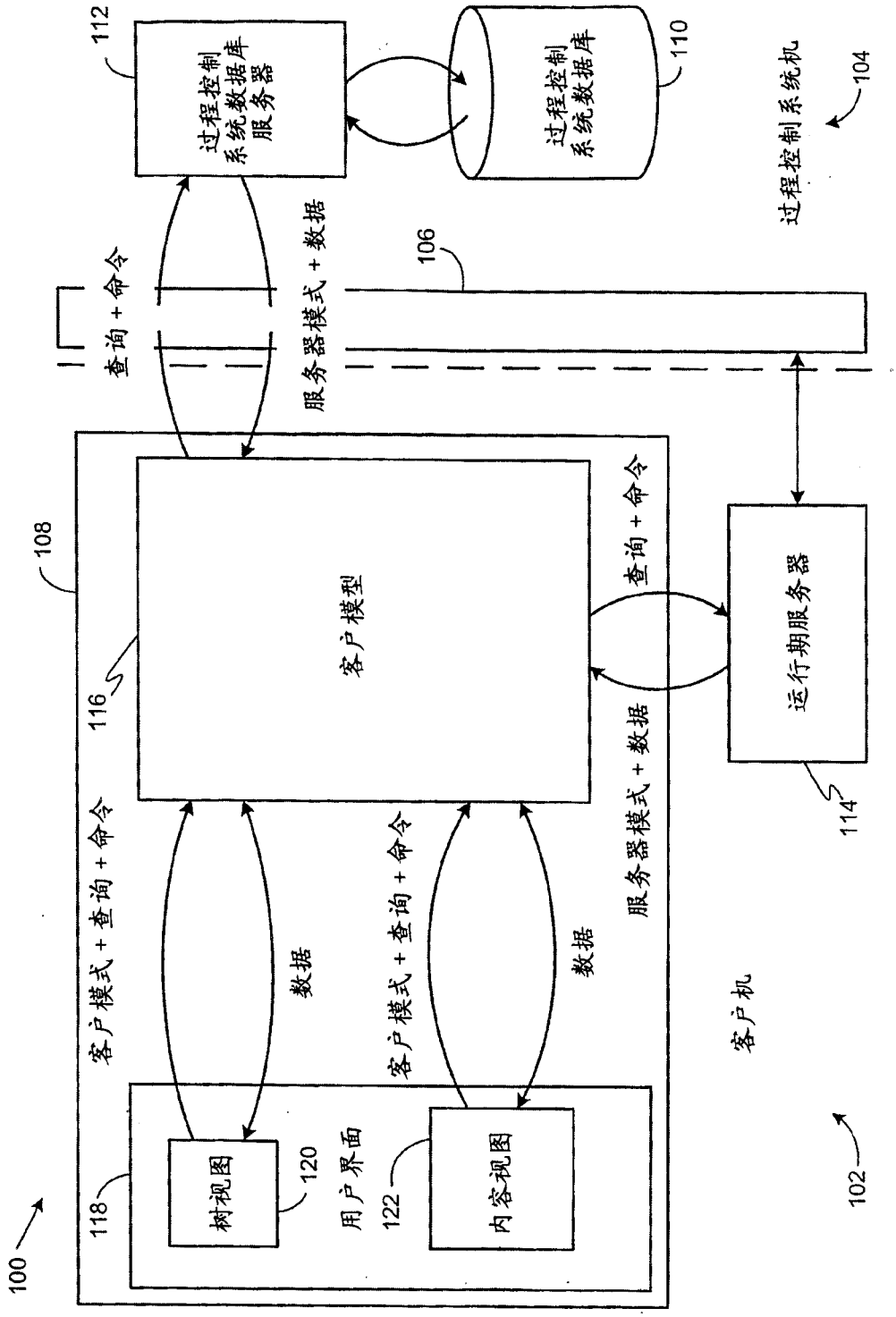
100
102
104
106
108
110
112
114
116
118
120
122
过程控制系统数据库服务器
过程控制系统数据库
过程控制系统机
查询 + 命令
服务器模式 + 数据
客户模型
运行期服务器
客户模式 + 查询 + 命令
数据
客户机
用户界面
树视图
内容视图

图 1

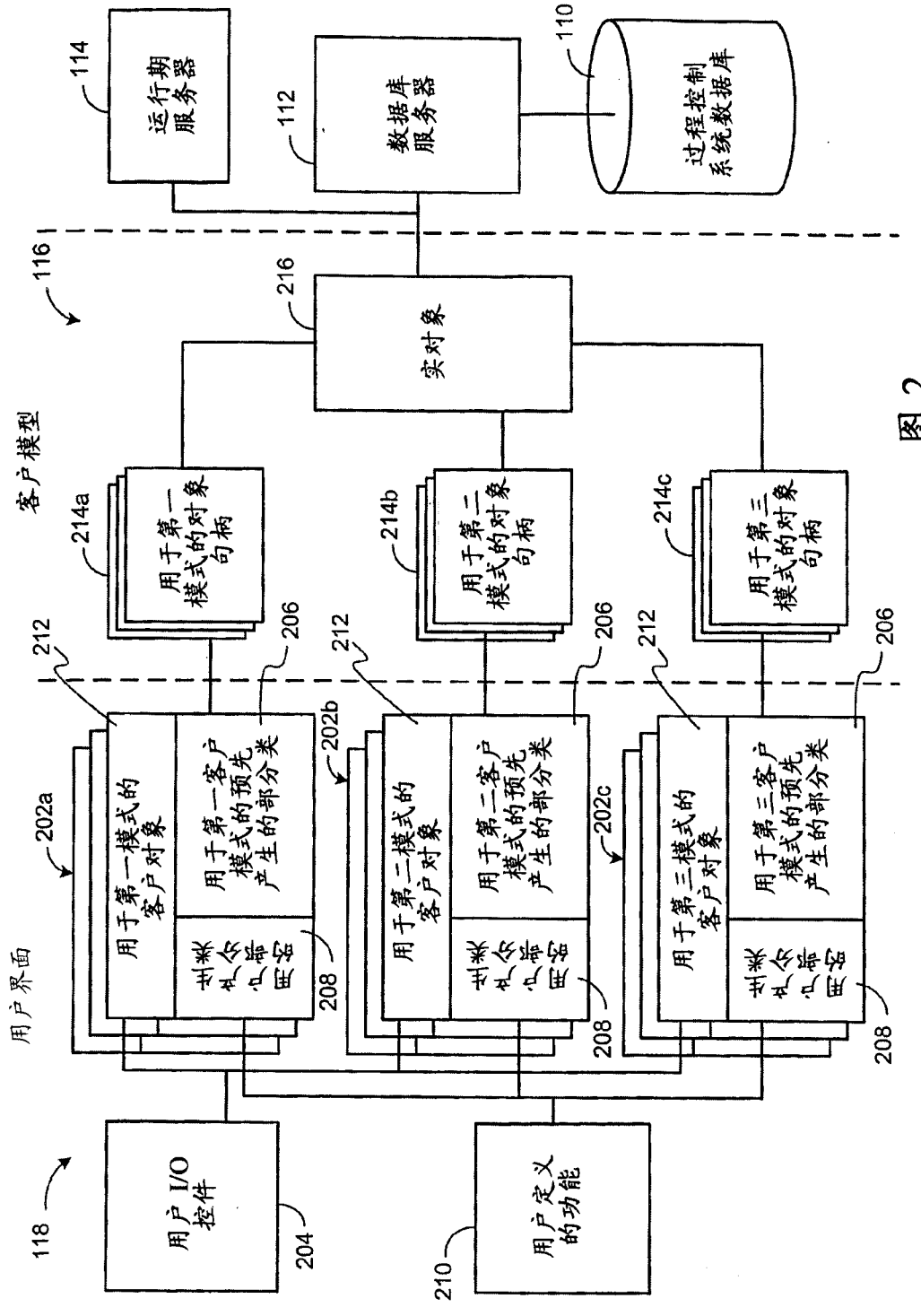
运行期服务器
114
数据库服务器
112
过程控制系统数据库
110
116
实对象
216
客户模型
用于第一模式的句柄
214a
用于第二模式的句柄
214b
用于第三模式的句柄
214c
212
206
用户界面
用于第一模式的客户对象
用于第一客户模式的预先产生的部分类
202a
202b
用于第二模式的客户对象
用于第二客户模式的预先产生的部分类
202c
用于第三模式的客户对象
用于第三客户模式的预先产生的部分类
208
118
用户 I/O 控件
204
用户定义的功能
210

图 2

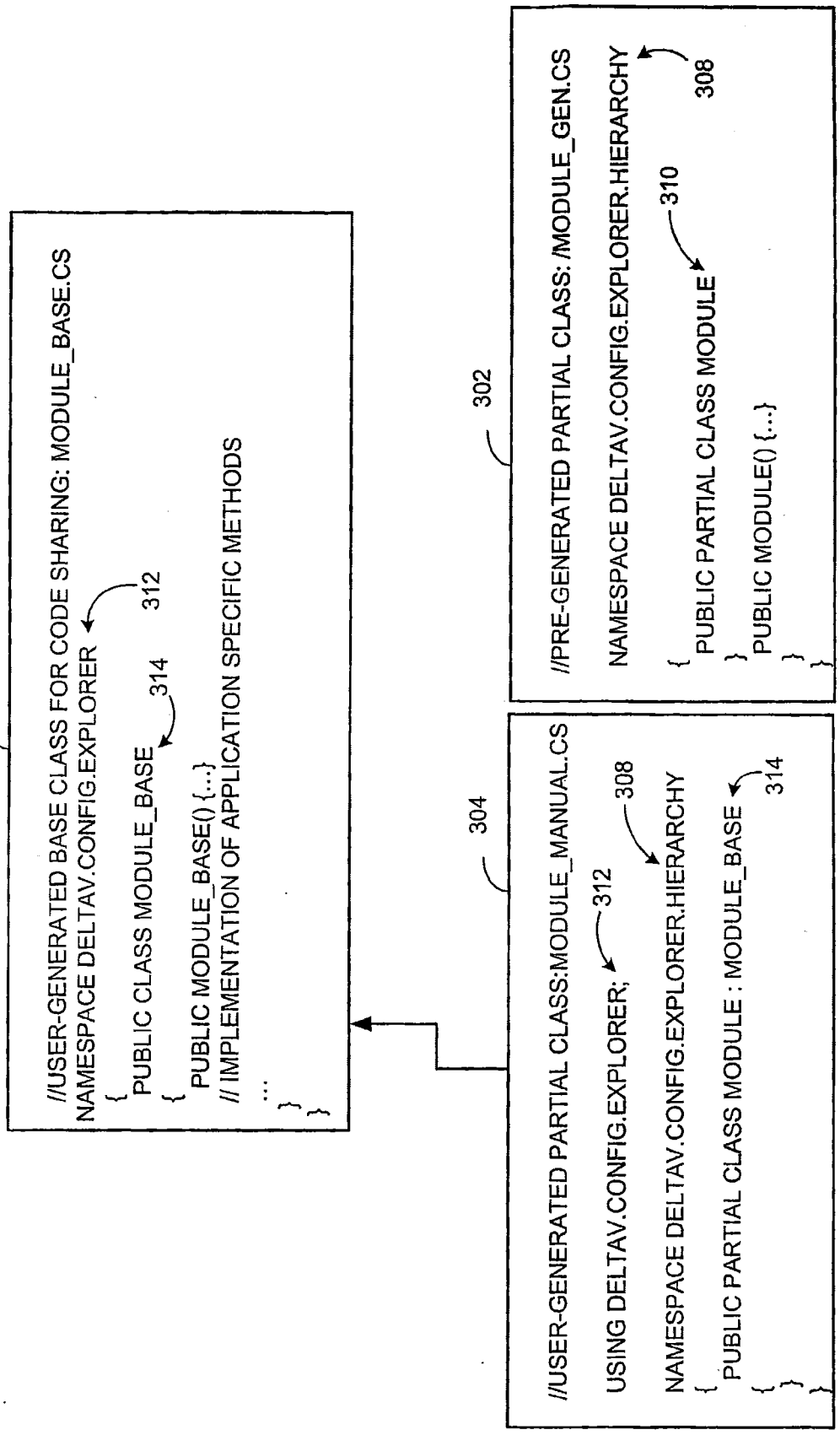
306
//USER-GENERATED BASE CLASS FOR CODE SHARING: MODULE_BASE.CS
NAMESPACE DELTAV.CONFIG.EXPLORER
{
PUBLIC CLASS MODULE_BASE
{
PUBLIC MODULE_BASE() {...}
// IMPLEMENTATION OF APPLICATION SPECIFIC METHODS
...
}
}
312
314
304
//USER-GENERATED PARTIAL CLASS:MODULE_MANUAL.CS
USING DELTAV.CONFIG.EXPLORER;
NAMESPACE DELTAV.CONFIG.EXPLORER.HIERARCHY
{
PUBLIC PARTIAL CLASS MODULE : MODULE_BASE
{
}
}
312
308
314
302
//PRE-GENERATED PARTIAL CLASS: /MODULE_GEN.CS
NAMESPACE DELTAV.CONFIG.EXPLORER.HIERARCHY
{
PUBLIC PARTIAL CLASS MODULE
{
PUBLIC MODULE() {...}
}
}
310
308

图 3

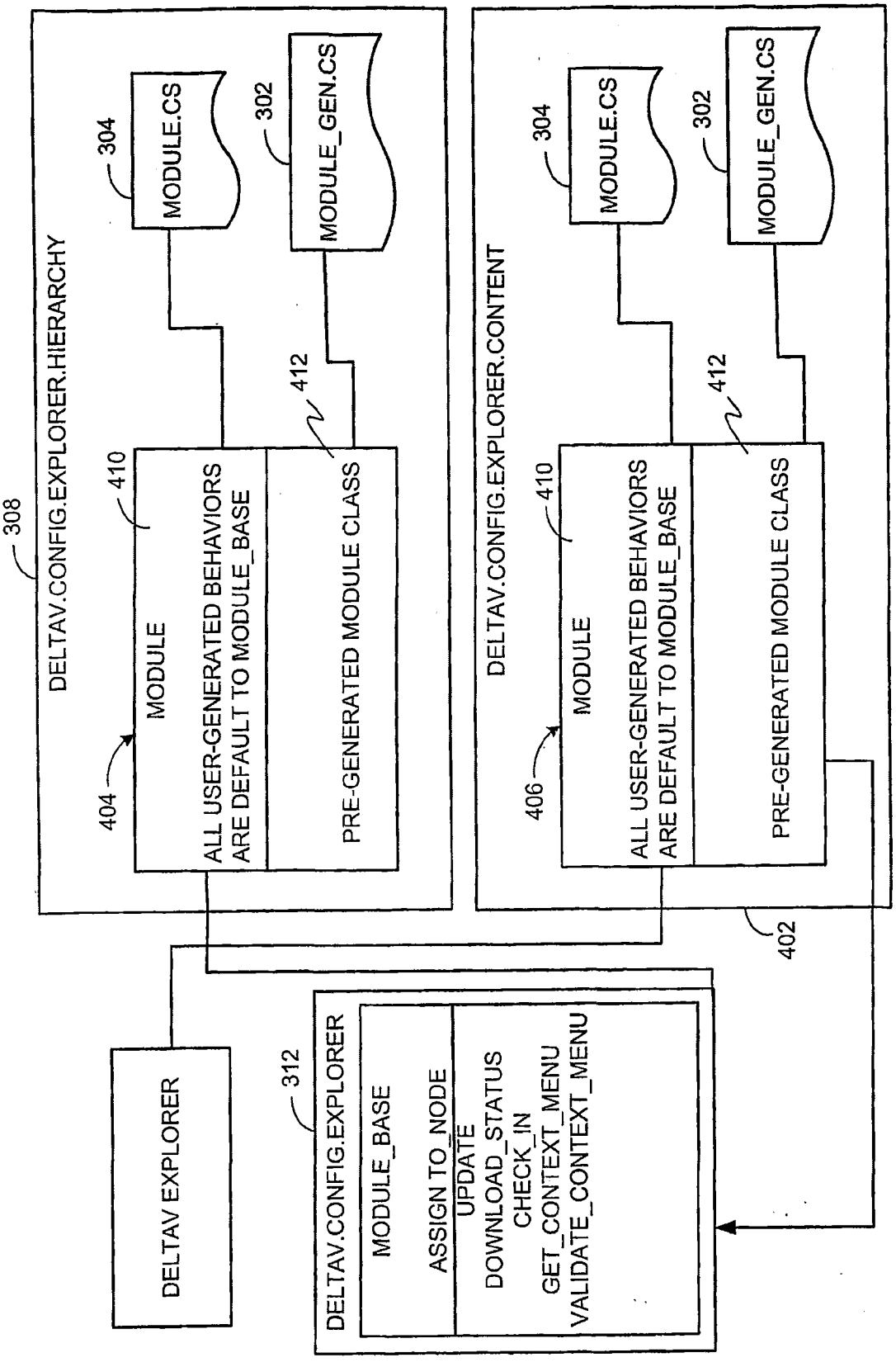
DELTAV EXPLORER
312
DELTAV.CONFIG.EXPLORER
MODULE_BASE
ASSIGN TO_NODE
UPDATE
DOWNLOAD_STATUS
CHECK_IN
GET_CONTEXT_MENU
VALIDATE_CONTEXT_MENU
402
308
DELTAV.CONFIG.EXPLORER.HIERARCHY
304
MODULE.CS
302
MODULE_GEN.CS
410
MODULE
ALL USER-GENERATED BEHAVIORS ARE DEFAULT TO MODULE_BASE
412
PRE-GENERATED MODULE CLASS
404
DELTAV.CONFIG.EXPLORER.CONTENT
304
MODULE.CS
302
MODULE_GEN.CS
410
MODULE
ALL USER-GENERATED BEHAVIORS ARE DEFAULT TO MODULE_BASE
412
PRE-GENERATED MODULE CLASS
406

图 4

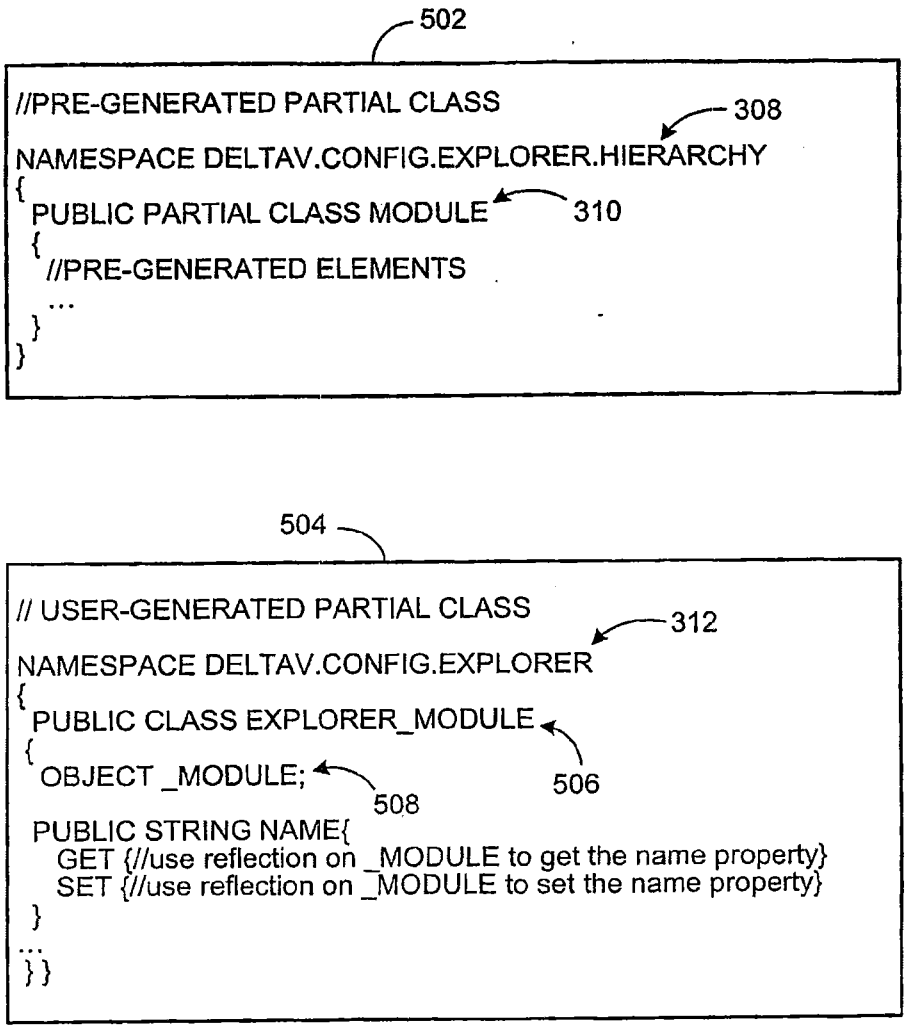
502
//PRE-GENERATED PARTIAL CLASS
308
NAMESPACE DELTAV.CONFIG.EXPLORER.HIERARCHY
{
PUBLIC PARTIAL CLASS MODULE
310
{
//PRE-GENERATED ELEMENTS
...
}
}
504
// USER-GENERATED PARTIAL CLASS
312
NAMESPACE DELTAV.CONFIG.EXPLORER
{
PUBLIC CLASS EXPLORER_MODULE
{
OBJECT _MODULE;
506
508
PUBLIC STRING NAME{
GET {//use reflection on _MODULE to get the name property}
SET {//use reflection on _MODULE to set the name property}
}
...
}}

图 5

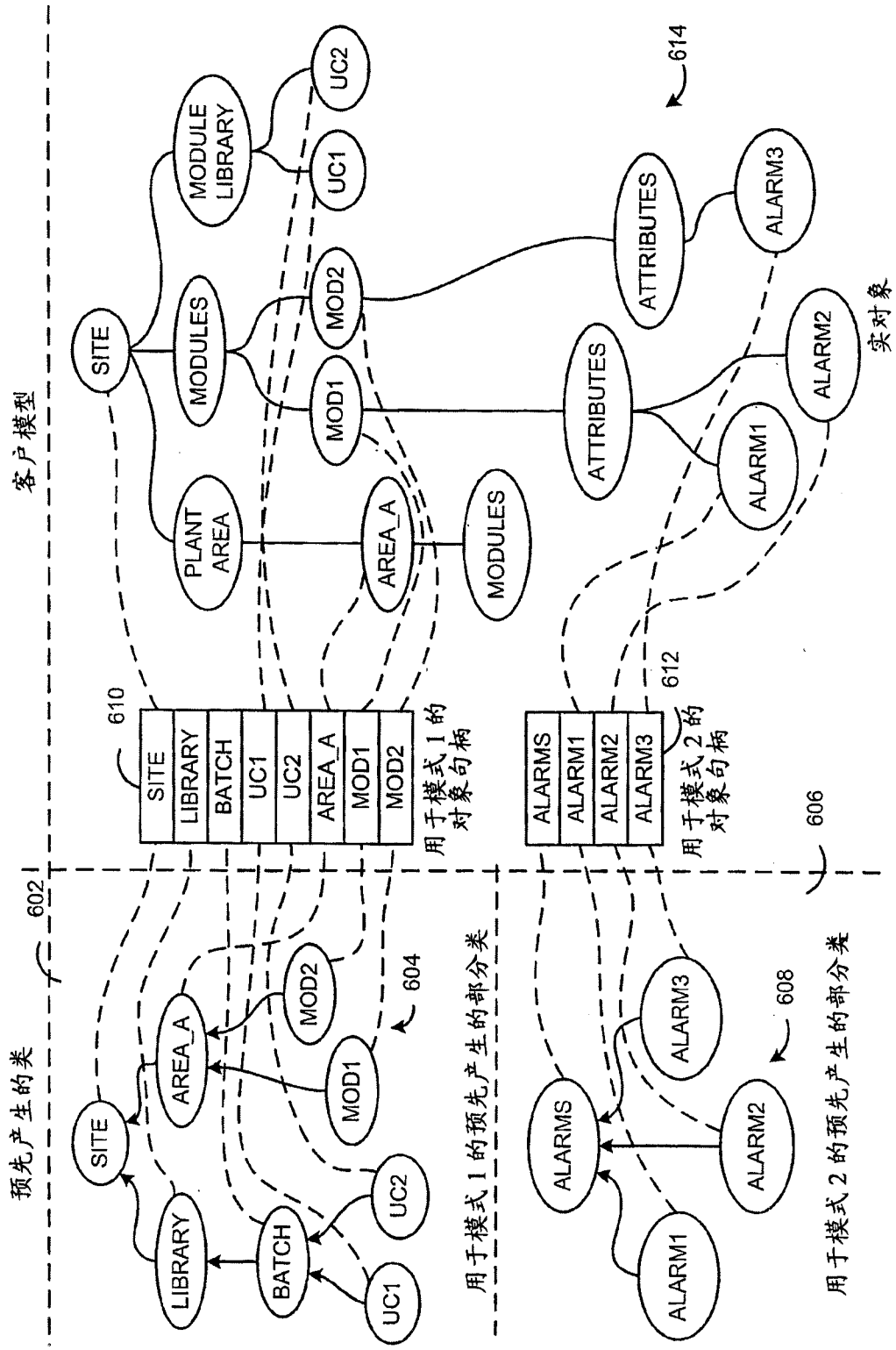
客户模型
预先产生的类
602
SITE
LIBRARY
BATCH
UC1
UC2
AREA_A
MOD1
MOD2
610
用于模式1的对象句柄
MODULE LIBRARY
MODULES
PLANT AREA
AREA_A
MODULES
ATTRIBUTES
ATTRIBUTES
614
ALARM3
ALARM2
ALARM1
实对象
ALARMS
ALARM1
ALARM2
ALARM3
612
用于模式2的对象句柄
606
604
608
用于模式1的预先产生的部分类
用于模式2的预先产生的部分类

图 6

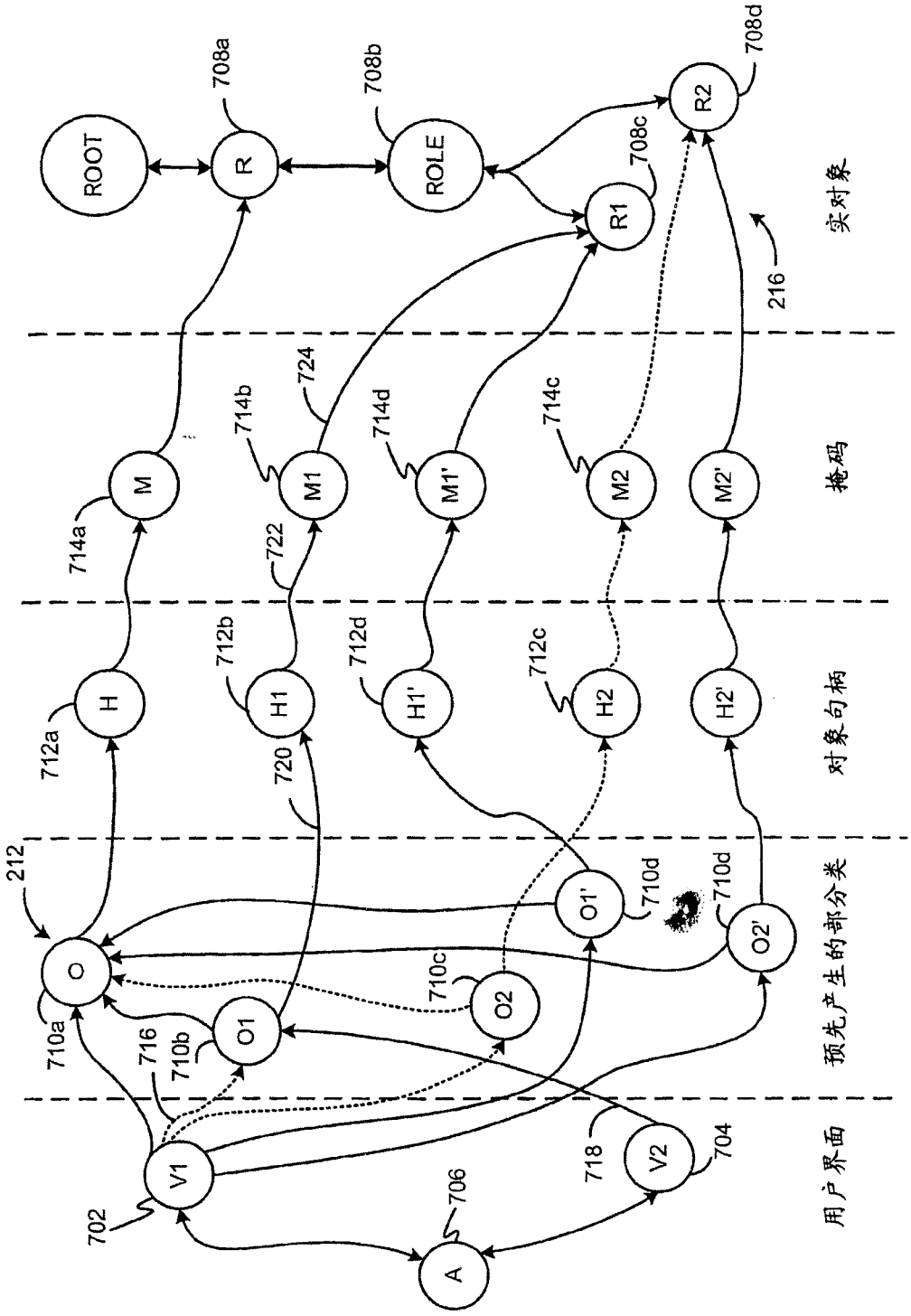
ROOT
R
ROLE
R1
R2
708a
708b
708c
708d
实对象
216
M
M1
M1'
M2
M2'
掩码
714a
714b
714d
714c
724
722
H
H1
H1'
H2
H2'
对象句柄
712a
712b
712d
712c
720
212
O
O1
O1'
O2
O2'
预先产生的部分类
710a
710b
710d
710c
716
V1
V2
用户界面
702
704
718
A
706

图 7

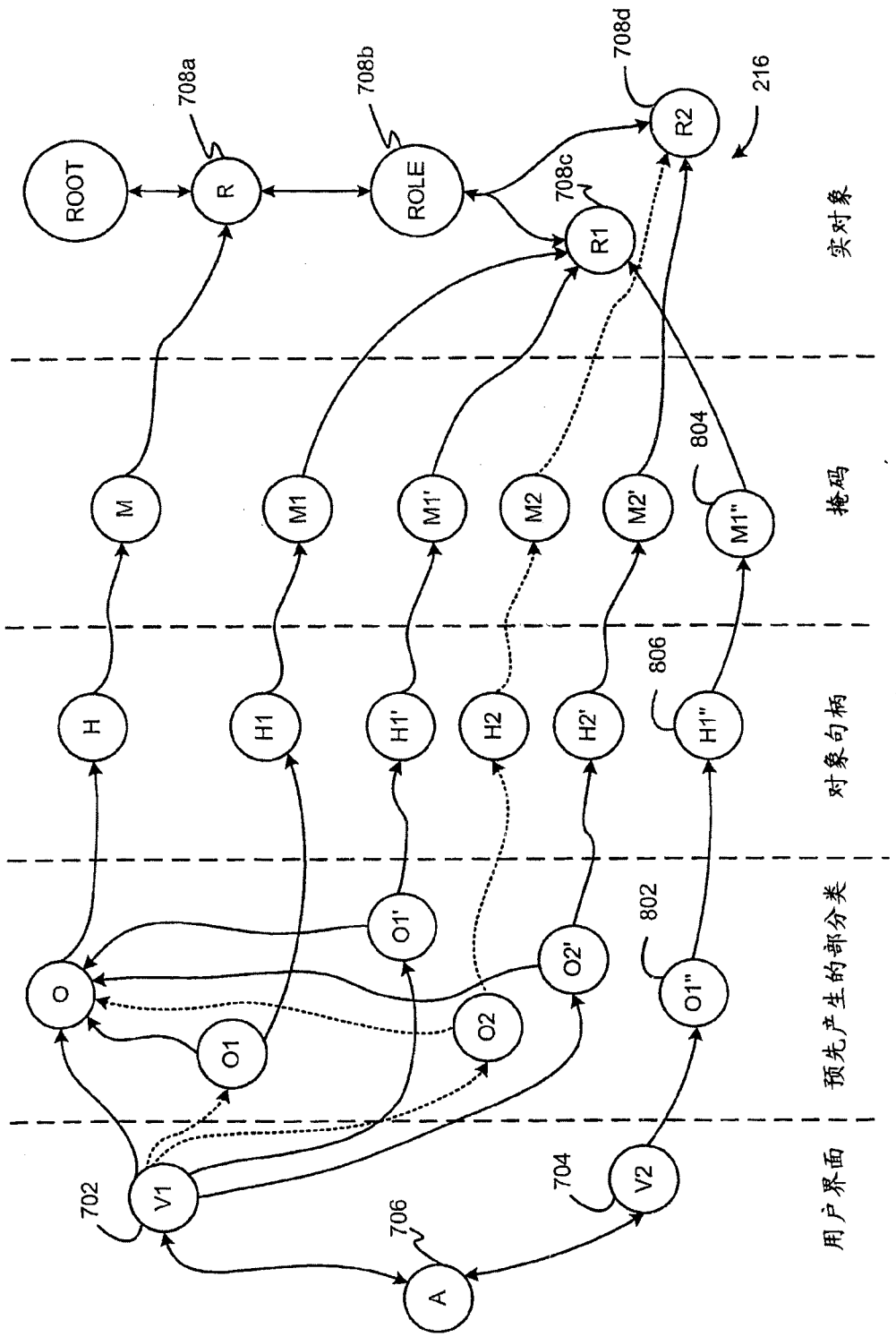
ROOT
R
ROLE
R2
R1
708a
708b
708c
708d
216
M
M1
M1'
M2
M2'
M1"
804
H
H1
H1'
H2
H2'
H1"
806
O
O1
O1'
O2
O2'
O1"
802
V1
V2
A
702
704
706
实对象
掩码
对象句柄
预先产生的部分类
用户界面

图 8

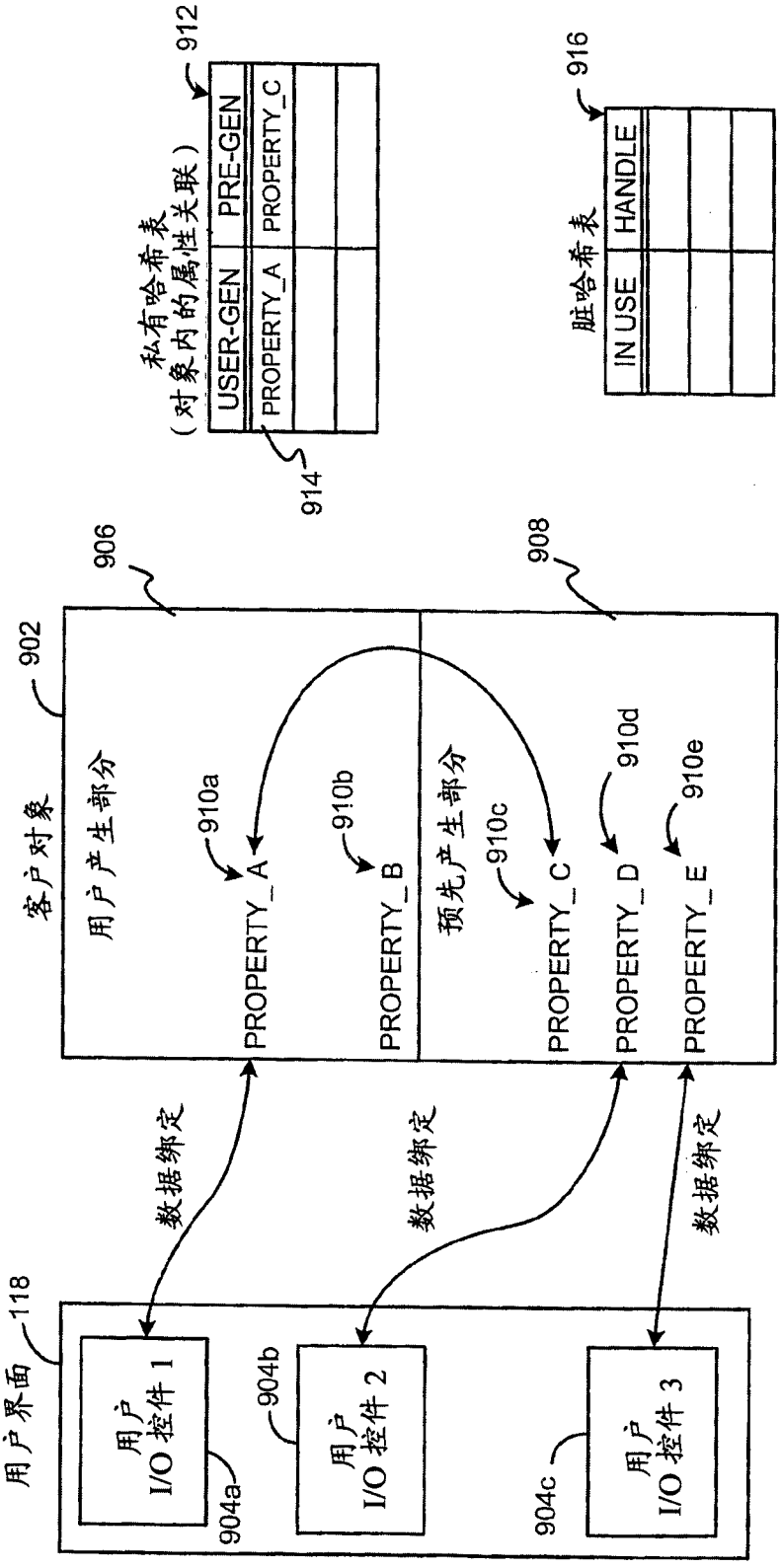
用户界面
118
用户 I/O 控件 1
904a
用户 I/O 控件 2
904b
用户 I/O 控件 3
904c
数据绑定
数据绑定
数据绑定
客户对象
902
用户产生部分
906
预先产生部分
908
PROPERTY_A
910a
PROPERTY_B
910b
PROPERTY_C
910c
PROPERTY_D
910d
PROPERTY_E
910e
私有哈希表
（对象内的属性关联）
912
USER-GEN
PRE-GEN
PROPERTY_A
PROPERTY_C
914
脏哈希表
916
IN USE
HANDLE

图 9

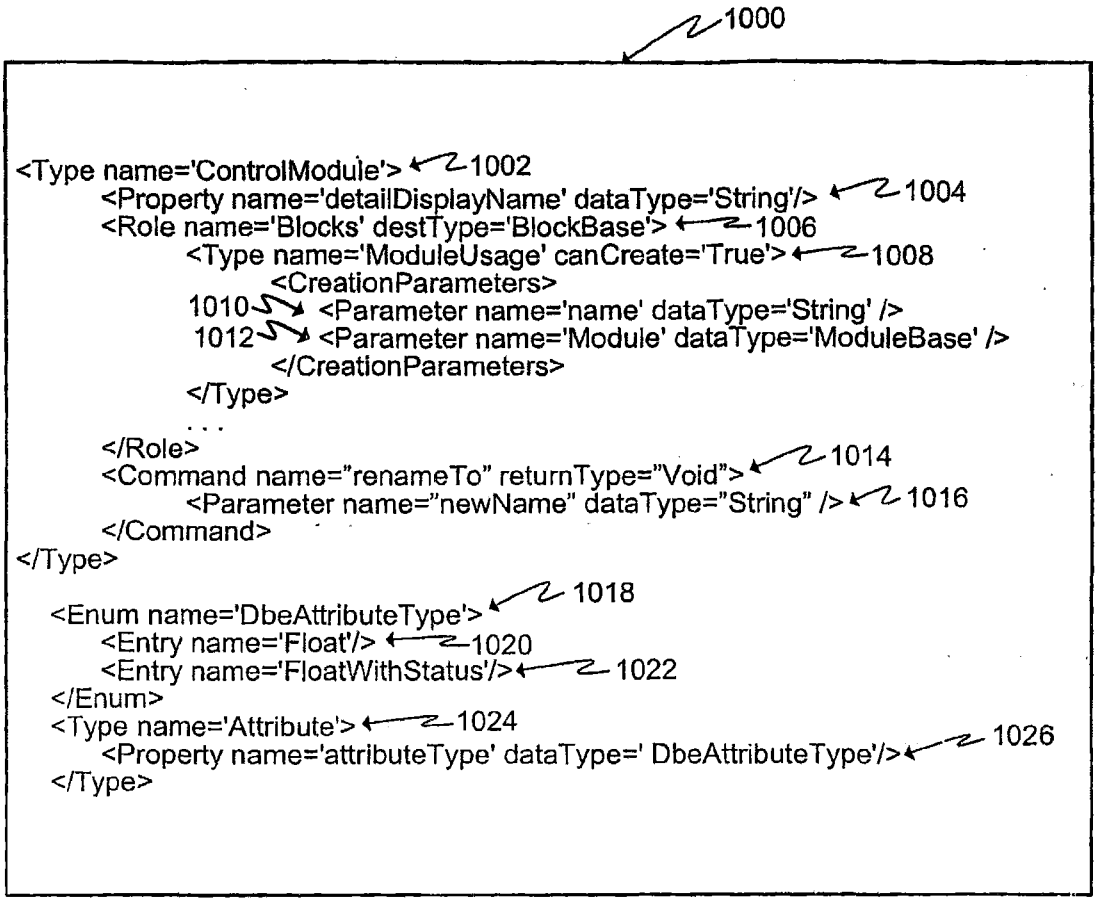
1000
<Type name='ControlModule'> 1002
<Property name='detailDisplayName' dataType='String'/> 1004
<Role name='Blocks' destType='BlockBase'> 1006
<Type name='ModuleUsage' canCreate='True'> 1008
<CreationParameters>
1010 <Parameter name='name' dataType='String' />
1012 <Parameter name='Module' dataType='ModuleBase' />
</CreationParameters>
</Type>
...
</Role>
<Command name="renameTo" returnType="Void"> 1014
<Parameter name="newName" dataType="String" /> 1016
</Command>
</Type>
<Enum name='DbeAttributeType'> 1018
<Entry name='Float'/> 1020
<Entry name='FloatWithStatus'/> 1022
</Enum>
<Type name='Attribute'> 1024
<Property name='attributeType' dataType=' DbeAttributeType'/> 1026
</Type>

图 10

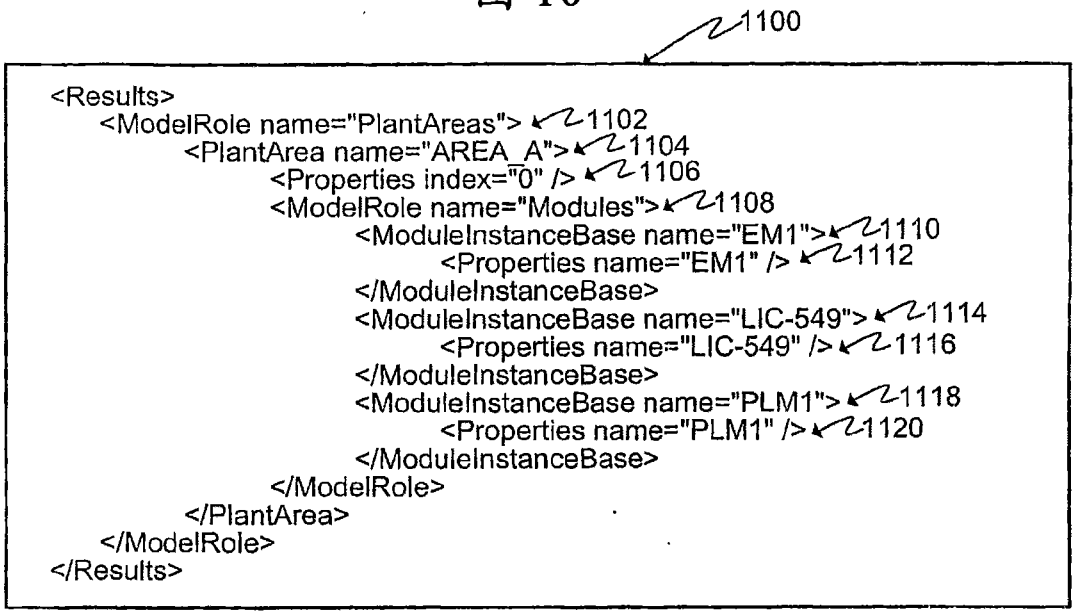
1100
<Results>
<ModelRole name="PlantAreas"> 1102
<PlantArea name="AREA_A"> 1104
<Properties index="0" /> 1106
<ModelRole name="Modules"> 1108
<ModuleInstanceBase name="EM1"> 1110
<Properties name="EM1" /> 1112
</ModuleInstanceBase>
<ModuleInstanceBase name="LIC-549"> 1114
<Properties name="LIC-549" /> 1116
</ModuleInstanceBase>
<ModuleInstanceBase name="PLM1"> 1118
<Properties name="PLM1" /> 1120
</ModuleInstanceBase>
</ModelRole>
</PlantArea>
</ModelRole>
</Results>

图 11

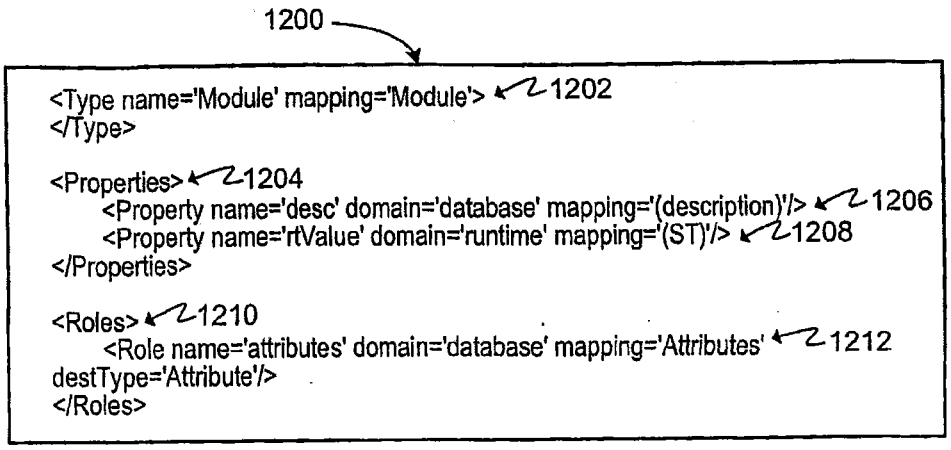
1200
<Type name='Module' mapping='Module'> 1202
</Type>
<Properties> 1204
<Property name='desc' domain='database' mapping='(description)'/> 1206
<Property name='rtValue' domain='runtime' mapping='(ST)'/> 1208
</Properties>
<Roles> 1210
<Role name='attributes' domain='database' mapping='Attributes' 1212
destType='Attribute'/>
</Roles>

图 12

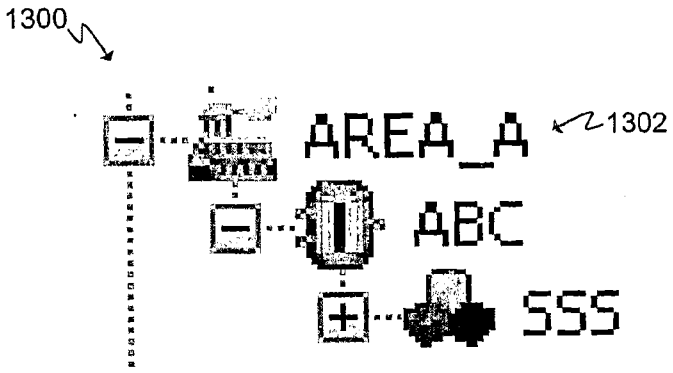
1300
AREA_A
1302
ABC
SSS

图 13

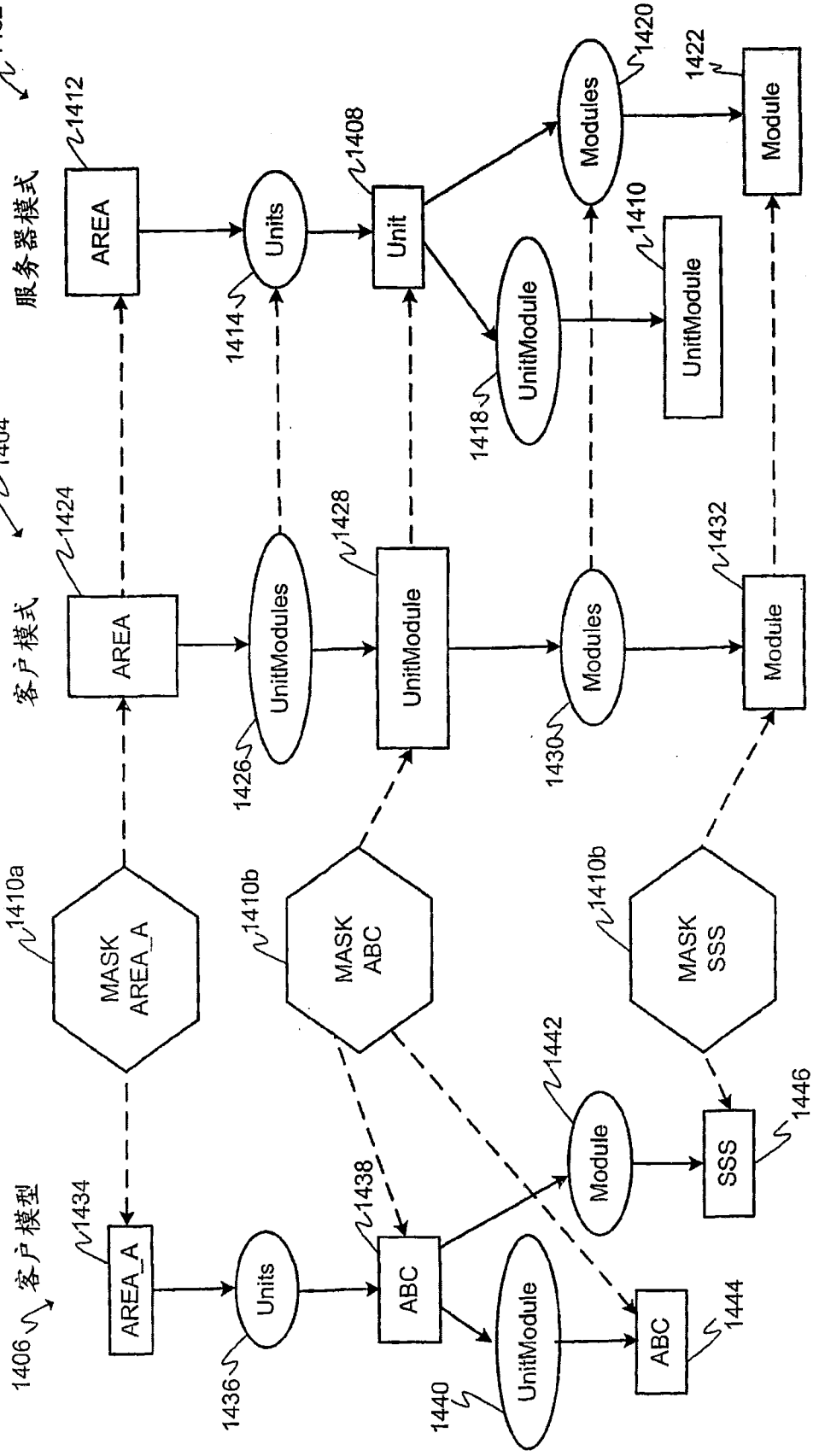
1402
服务器模式
AREA
1412
Units
1414
Unit
1408
Modules
1420
Module
1422
UnitModule
1418
UnitModule
1410
1404
客户模式
AREA
1424
UnitModules
1426
UnitModule
1428
Modules
1430
Module
1432
MASK
AREA_A
1410a
MASK
ABC
1410b
MASK
SSS
1410b
1406
客户模型
AREA_A
1434
Units
1436
ABC
1438
UnitModule
1440
Module
1442
SSS
1446
ABC
1444
1400

图 14

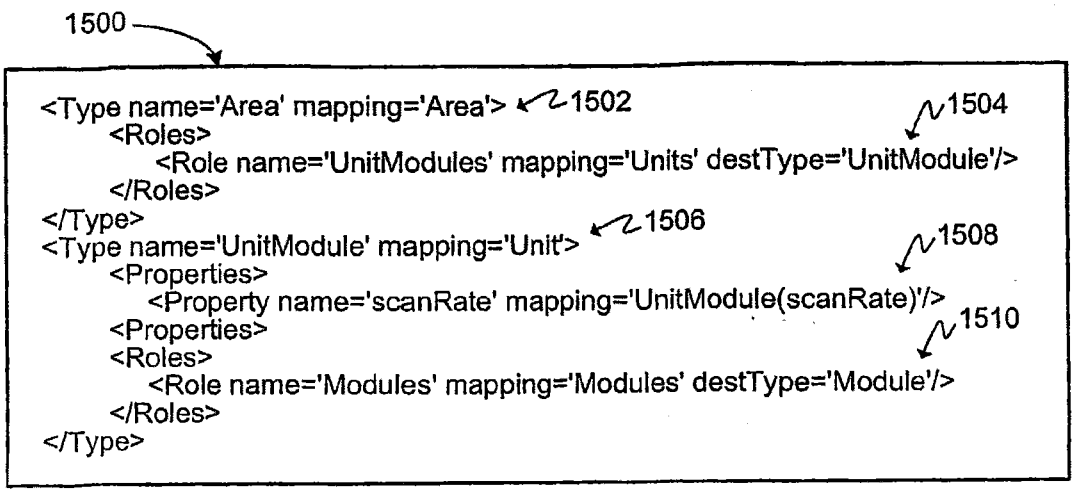
1500
<Type name='Area' mapping='Area'>
1502
1504
<Roles>
<Role name='UnitModules' mapping='Units' destType='UnitModule'/>
</Roles>
</Type>
<Type name='UnitModule' mapping='Unit'>
1506
1508
<Properties>
<Property name='scanRate' mapping='UnitModule(scanRate)'/>
<Properties>
1510
<Roles>
<Role name='Modules' mapping='Modules' destType='Module'/>
</Roles>
</Type>

图 15

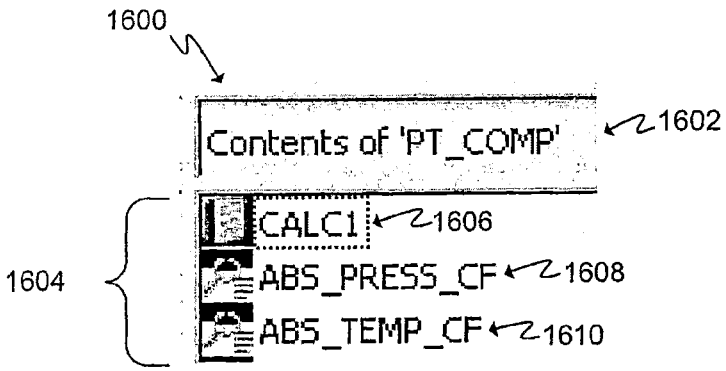
1600
Contents of 'PT_COMP'
1602
CALC1
1606
1604
ABS_PRESS_CF
1608
ABS_TEMP_CF
1610

图 16

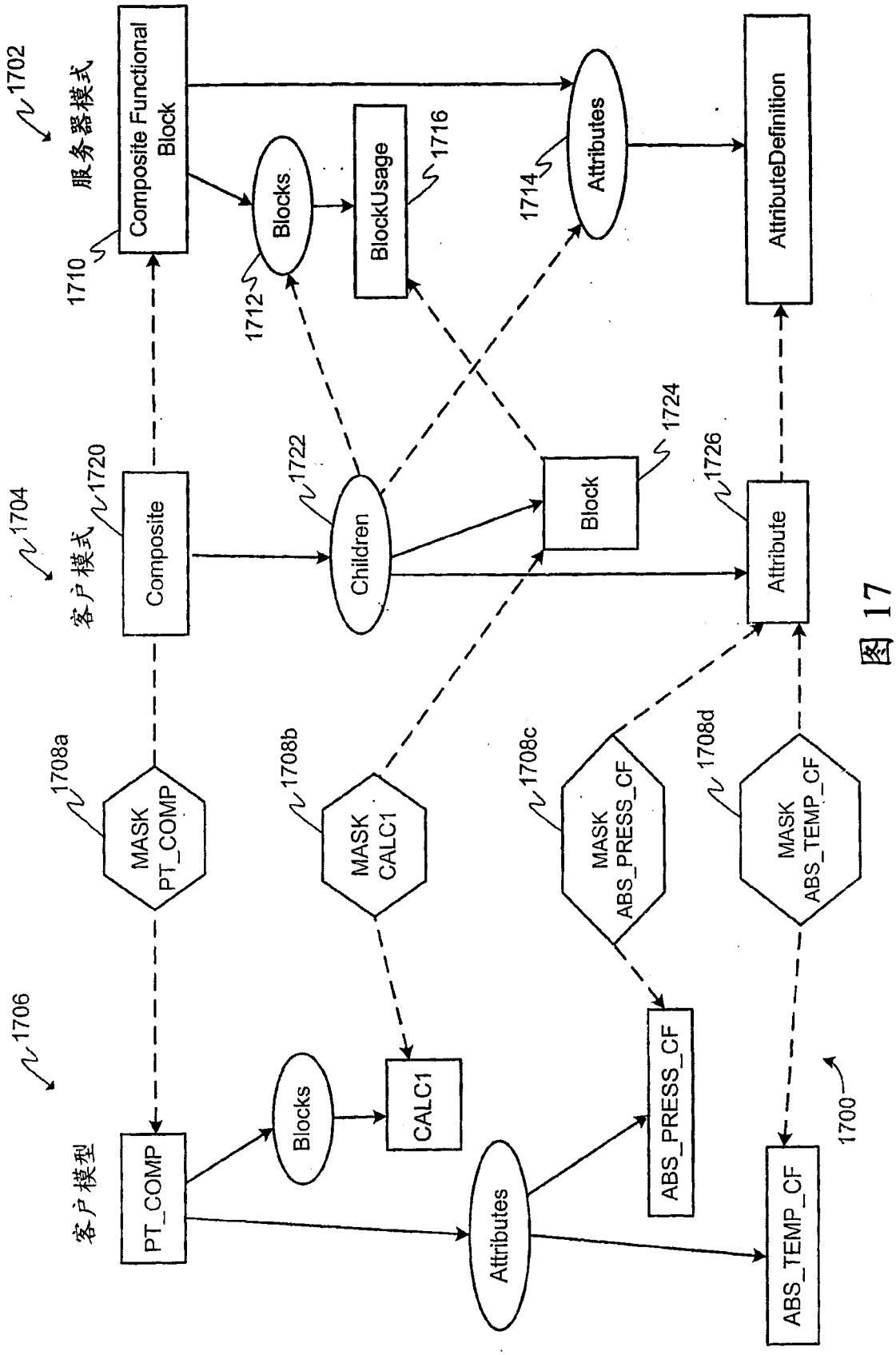
1702
服务器模式
Composite Functional Block
1710
Blocks
1712
BlockUsage
1716
Attributes
1714
AttributeDefinition
1704
客户模式
1720
Composite
Children
1722
Block
1724
Attribute
1726
1708a
MASK PT_COMP
1708b
MASK CALC1
1708c
MASK ABS_PRESS_CF
1708d
MASK ABS_TEMP_CF
1706
客户模型
PT_COMP
Blocks
CALC1
Attributes
ABS_PRESS_CF
ABS_TEMP_CF
1700

图 17

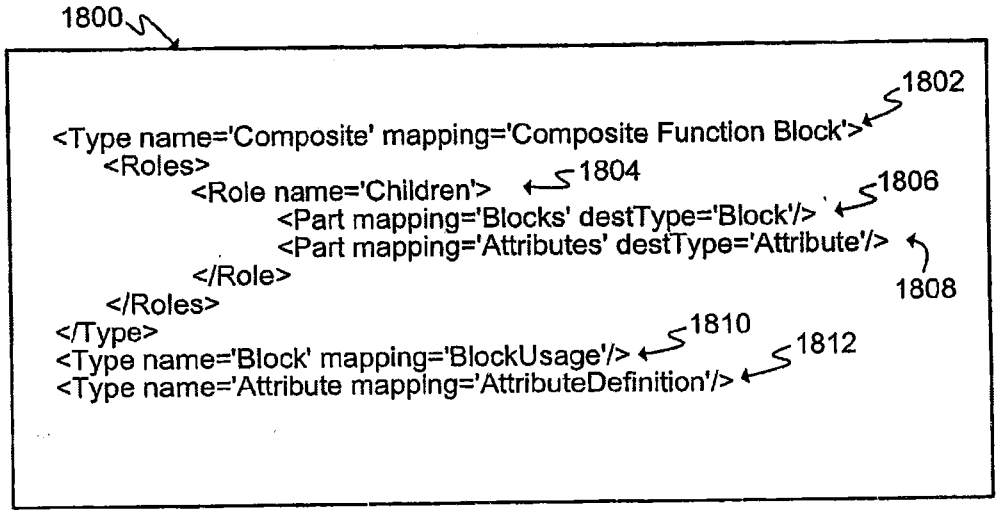
1800
1802
<Type name='Composite' mapping='Composite Function Block'>
<Roles>
<Role name='Children'>
1804
1806
<Part mapping='Blocks' destType='Block'/>
<Part mapping='Attributes' destType='Attribute'/>
1808
</Role>
</Roles>
</Type>
1810
<Type name='Block' mapping='BlockUsage'/>
1812
<Type name='Attribute mapping='AttributeDefinition'/>

图 18

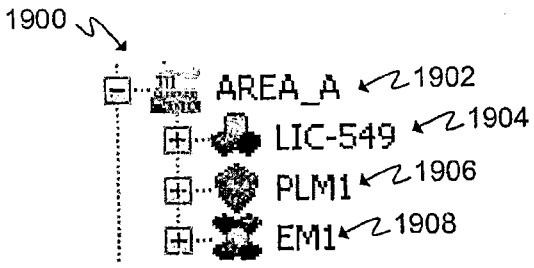
1900
AREA_A
1902
LIC-549
1904
PLM1
1906
EM1
1908

图 19

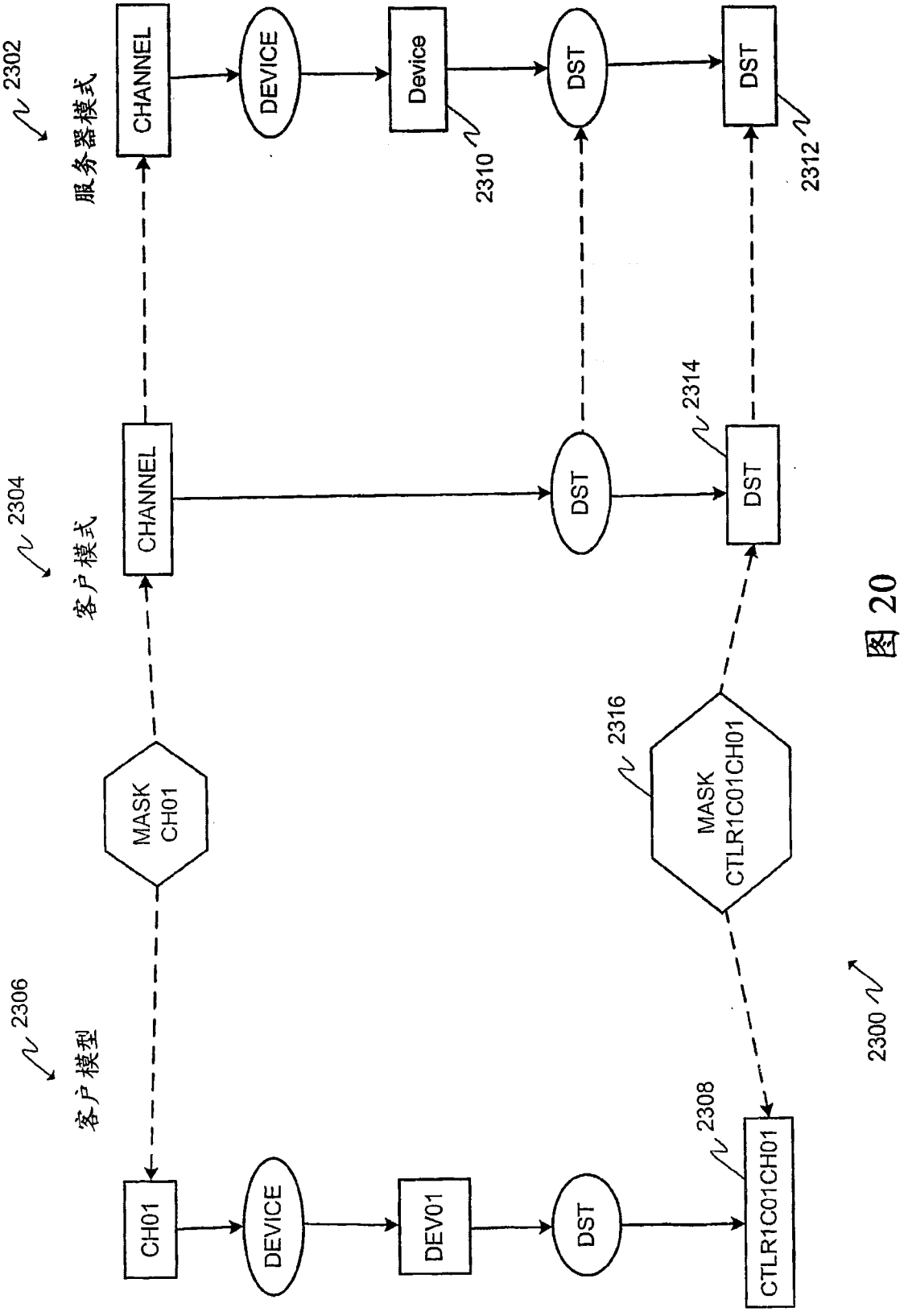
2302
服务器模式
CHANNEL
DEVICE
Device
DST
DST
2310
2312
2304
客户模式
CHANNEL
DST
DST
2314
MASK
CH01
2316
MASK
CTLR1C01CH01
2306
客户模型
CH01
DEVICE
DEV01
DST
2308
CTLR1C01CH01
2300

图 20

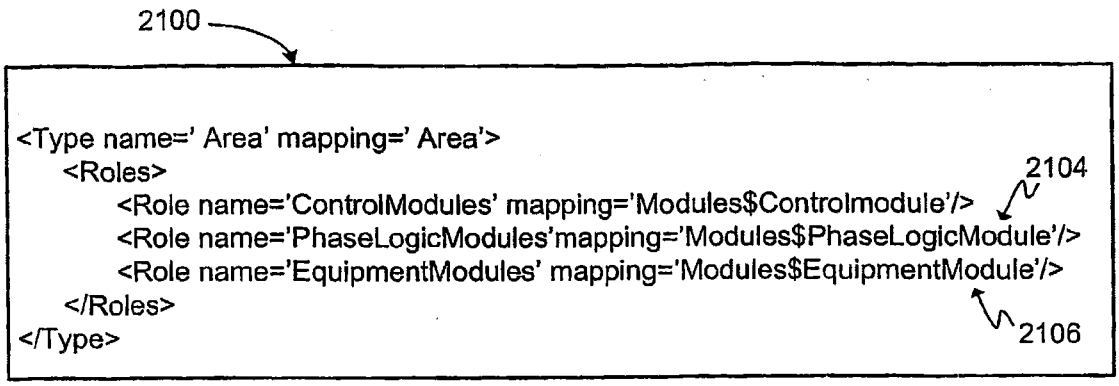
2100
<Type name=' Area' mapping=' Area'>
<Roles>
<Role name='ControlModules' mapping='Modules$Controlmodule'/>
<Role name='PhaseLogicModules'mapping='Modules$PhaseLogicModule'/>
<Role name='EquipmentModules' mapping='Modules$EquipmentModule'/>
</Roles>
</Type>
2104
2106

图 21


2200
2202
CH01
2204
CTLR1C01CH01

图 22

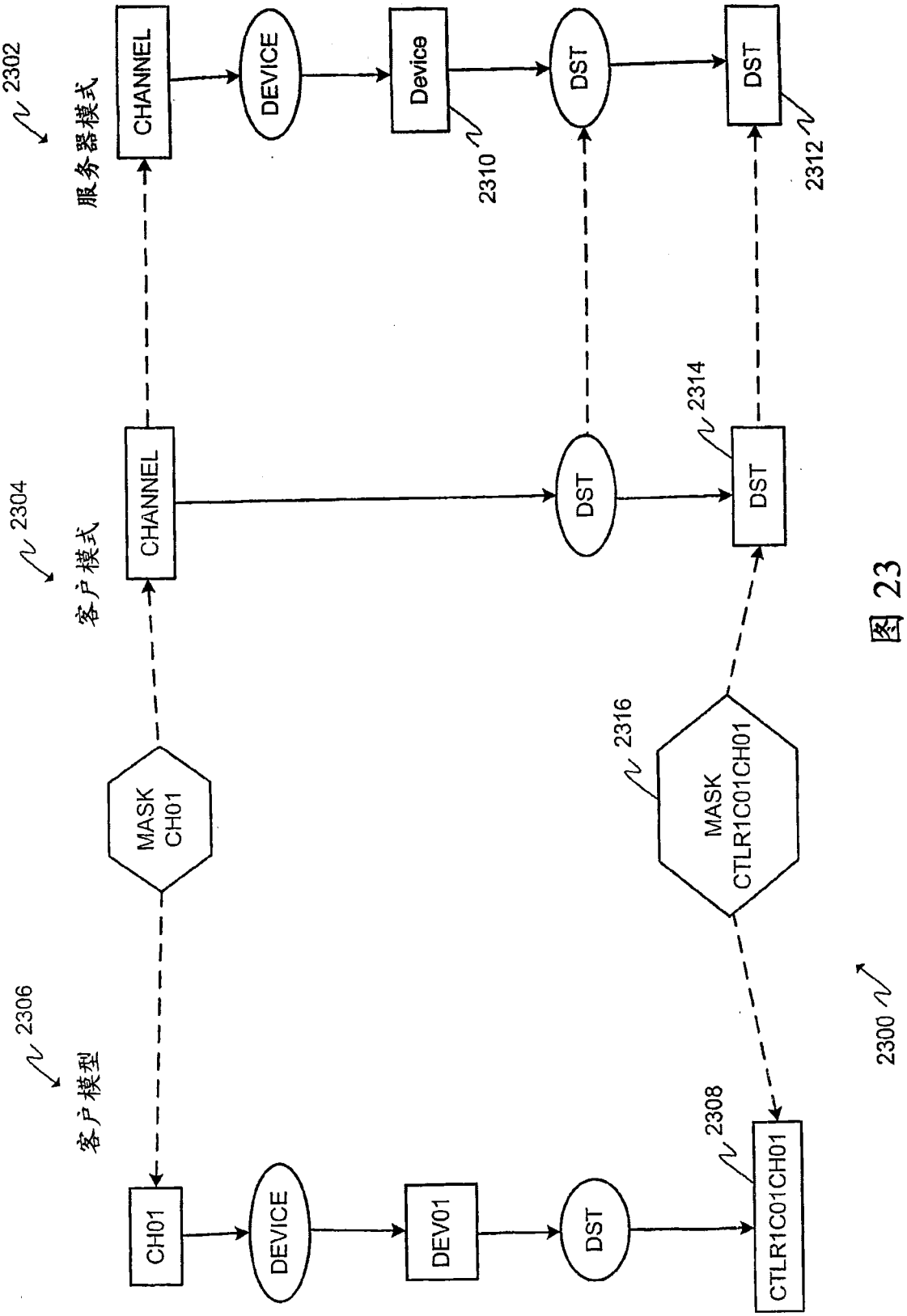
2302
服务器模式
CHANNEL
DEVICE
Device
DST
DST
2310
2312
2304
客户模式
CHANNEL
DST
DST
2314
MASK
CH01
2316
MASK
CTLR1C01CH01
2306
客户模型
CH01
DEVICE
DEV01
DST
CTLR1C01CH01
2308
2300

图 23

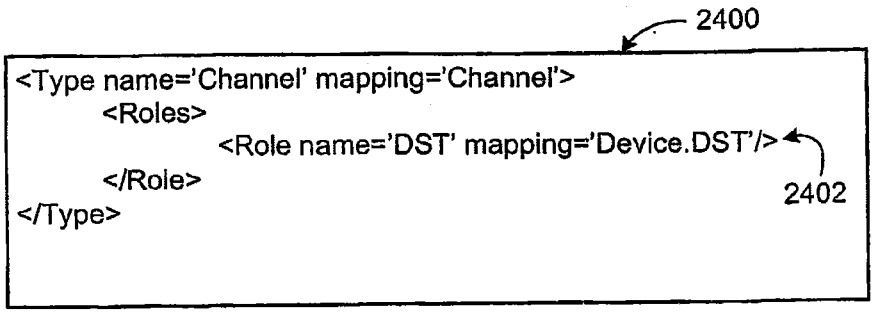
2400
<Type name='Channel' mapping='Channel'>
<Roles>
<Role name='DST' mapping='Device.DST'/>
</Role>
</Type>
2402

图 24

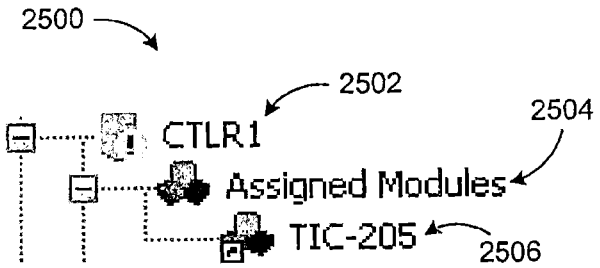
2500
2502
CTLR1
2504
Assigned Modules
TIC-205
2506

图 25

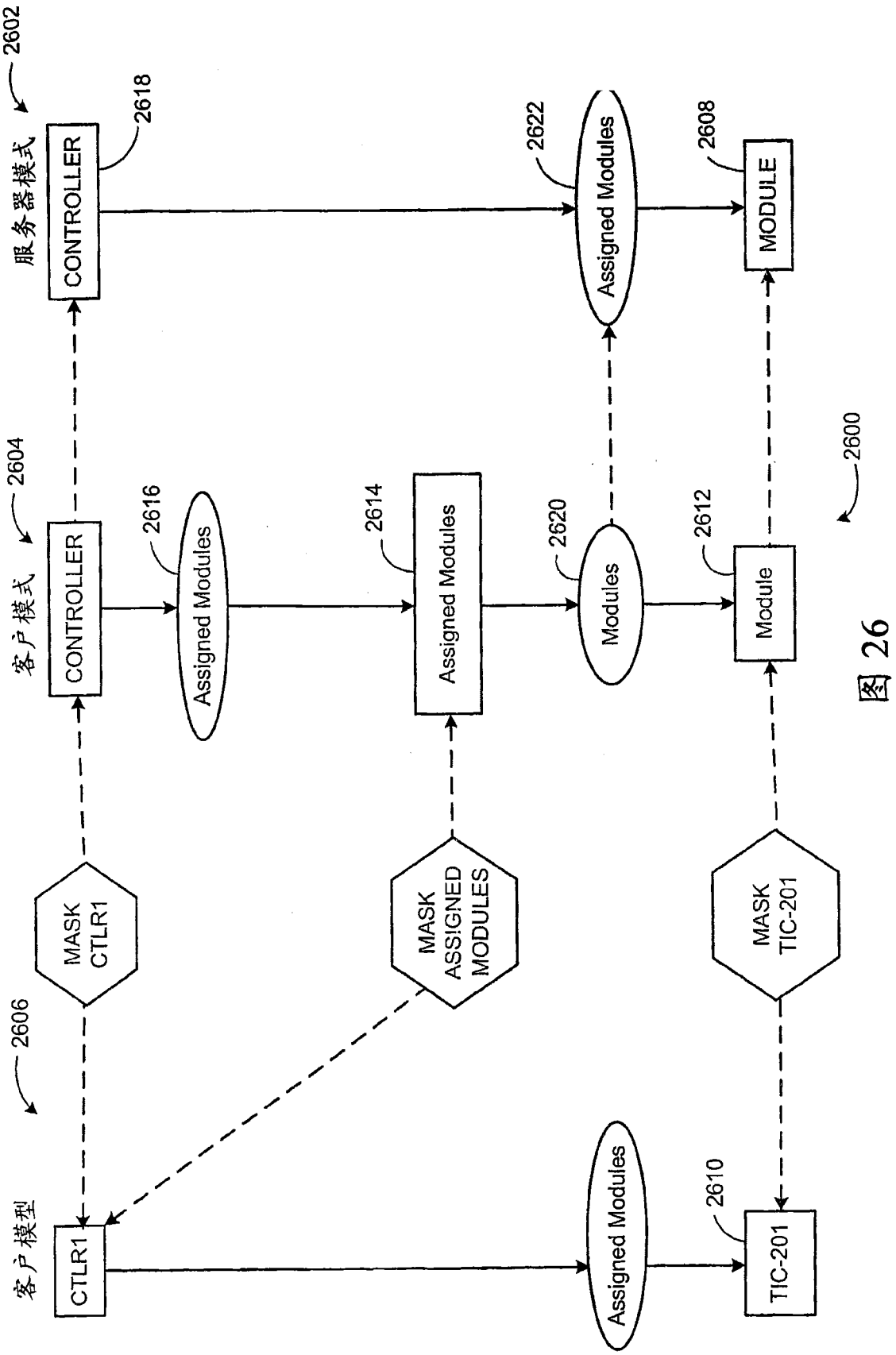
2602
服务器模式
CONTROLLER
2618
2622
Assigned Modules
2608
MODULE
2604
客户模式
CONTROLLER
2616
Assigned Modules
2614
Assigned Modules
2620
Modules
2612
Module
2600
MASK CTLR1
MASK ASSIGNED MODULES
MASK TIC-201
2606
客户模型
CTLR1
Assigned Modules
2610
TIC-201

图 26

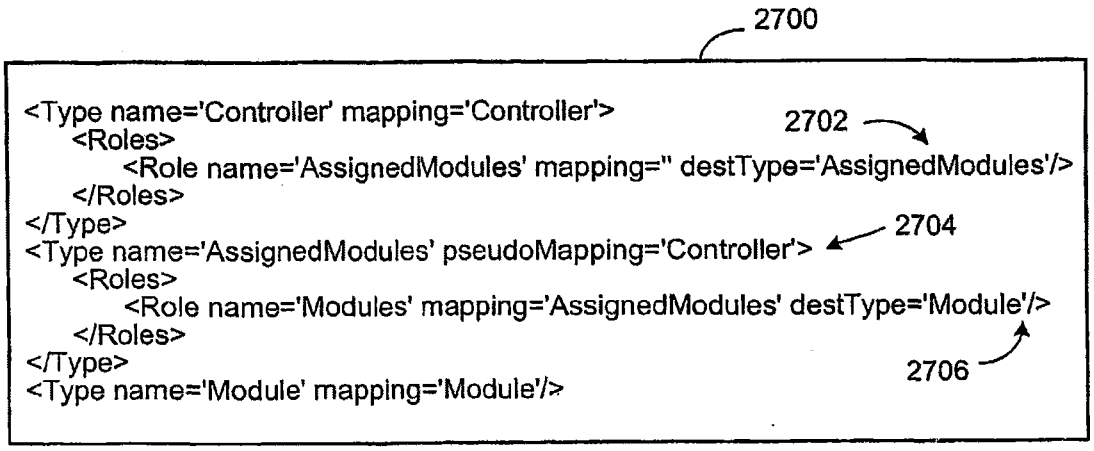
2700
<Type name='Controller' mapping='Controller'>
<Roles>
<Role name='AssignedModules' mapping='' destType='AssignedModules'/>
2702
</Roles>
</Type>
<Type name='AssignedModules' pseudoMapping='Controller'>
2704
<Roles>
<Role name='Modules' mapping='AssignedModules' destType='Module'/>
</Roles>
</Type>
2706
<Type name='Module' mapping='Module'/>

图 27

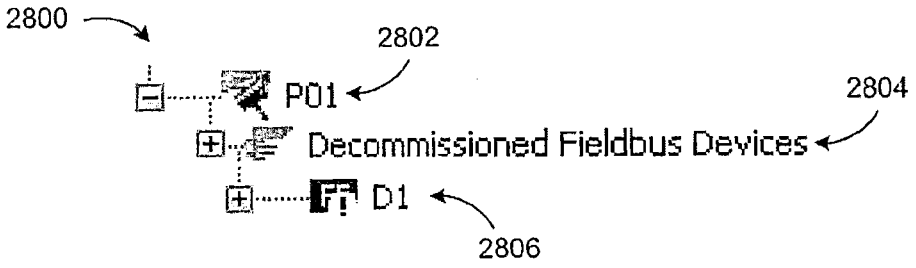
2800
2802
P01
2804
Decommissioned Fieldbus Devices
D1
2806

图 28

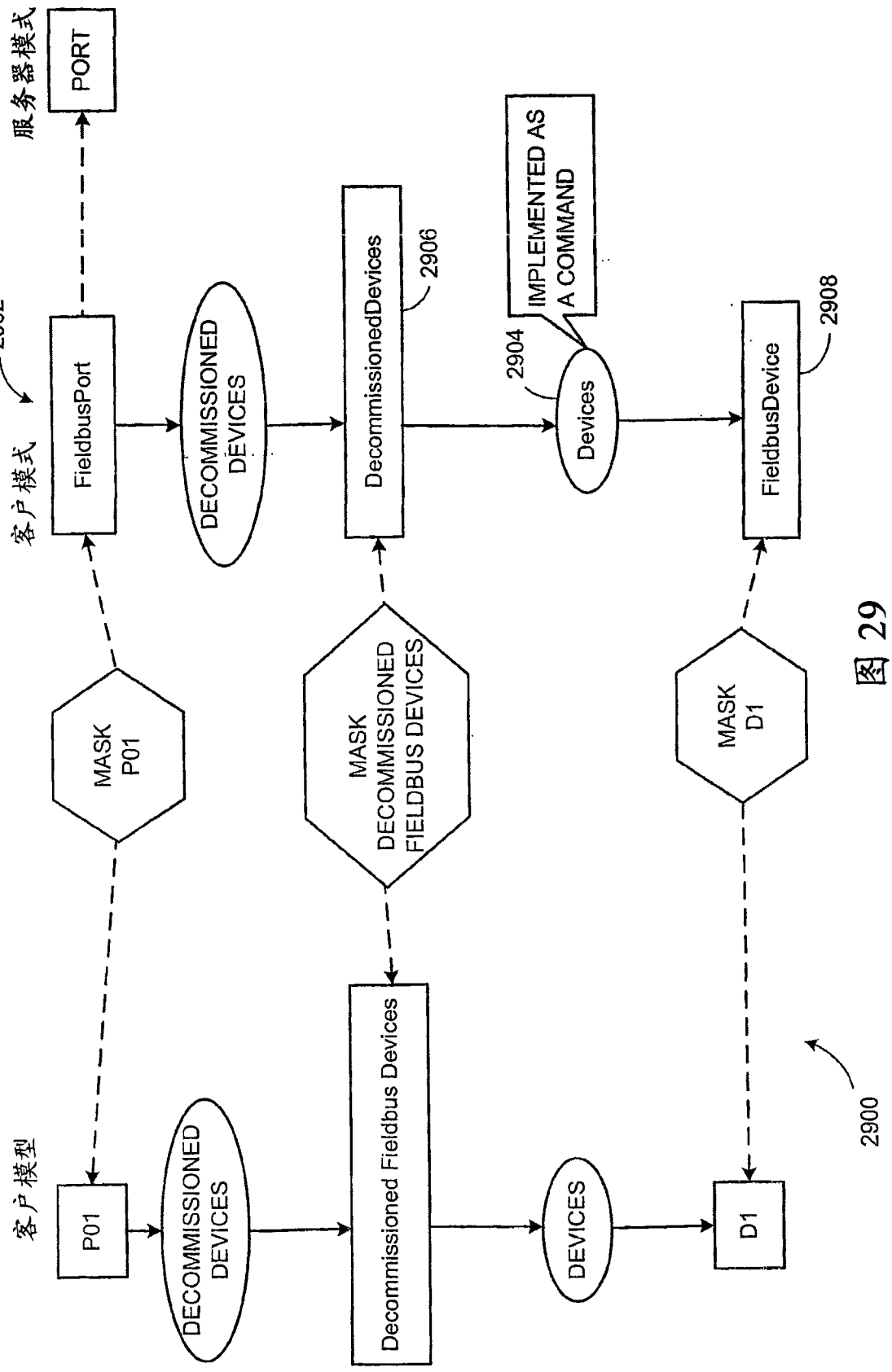
服务器模式
PORT
客户模式
2902
FieldbusPort
DECOMMISSIONED DEVICES
DecommissionedDevices
2906
IMPLEMENTED AS A COMMAND
2904
Devices
FieldbusDevice
2908
MASK P01
MASK DECOMMISSIONED FIELDBUS DEVICES
MASK D1
客户模型
P01
DECOMMISSIONED DEVICES
Decommissioned Fieldbus Devices
DEVICES
D1
2900

图 29

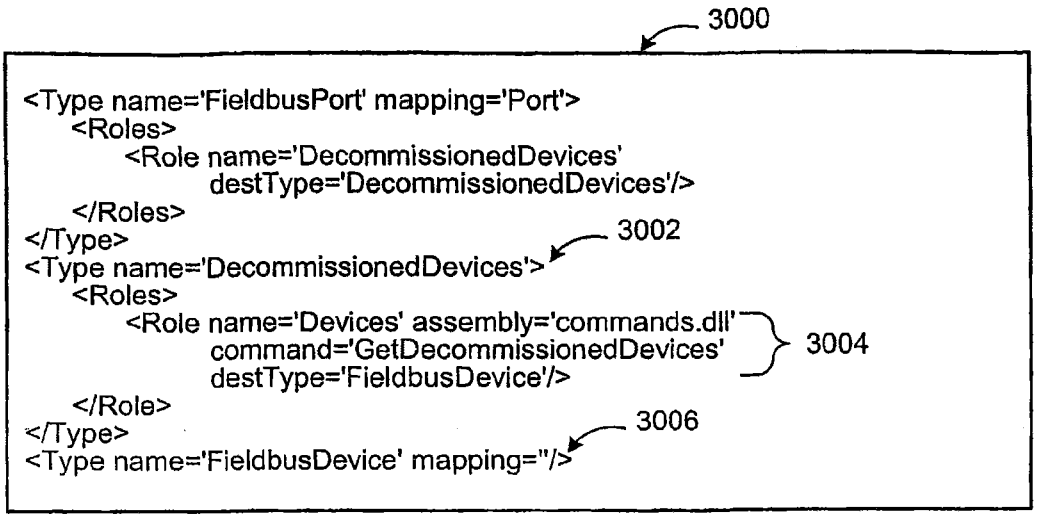
3000
<Type name='FieldbusPort' mapping='Port'>
<Roles>
<Role name='DecommissionedDevices'
destType='DecommissionedDevices'/>
</Roles>
</Type>
3002
<Type name='DecommissionedDevices'>
<Roles>
<Role name='Devices' assembly='commands.dll'
command='GetDecommissionedDevices'
destType='FieldbusDevice'/>
3004
</Role>
</Type>
3006
<Type name='FieldbusDevice' mapping="/>

图 30

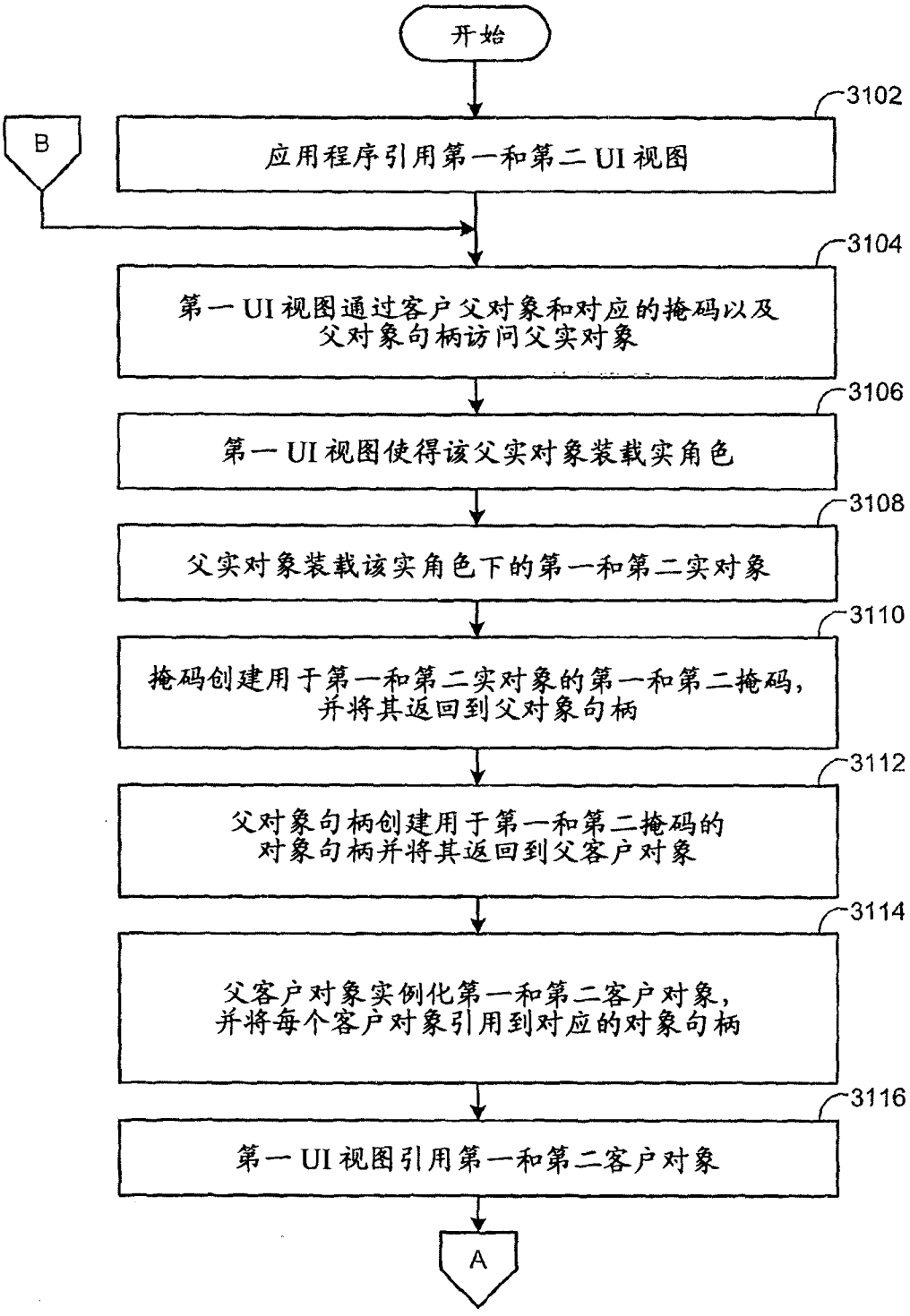
开始
B
3102
应用程序引用第一和第二 UI 视图
3104
第一 UI 视图通过客户父对象和对应的掩码以及父对象句柄访问父实对象
3106
第一 UI 视图使得该父实对象装载实角色
3108
父实对象装载该实角色下的第一和第二实对象
3110
掩码创建用于第一和第二实对象的第一和第二掩码，并将其返回到父对象句柄
3112
父对象句柄创建用于第一和第二掩码的对象句柄并将其返回到父客户对象
3114
父客户对象实例化第一和第二客户对象，并将每个客户对象引用到对应的对象句柄
3116
第一 UI 视图引用第一和第二客户对象
A

图 31A

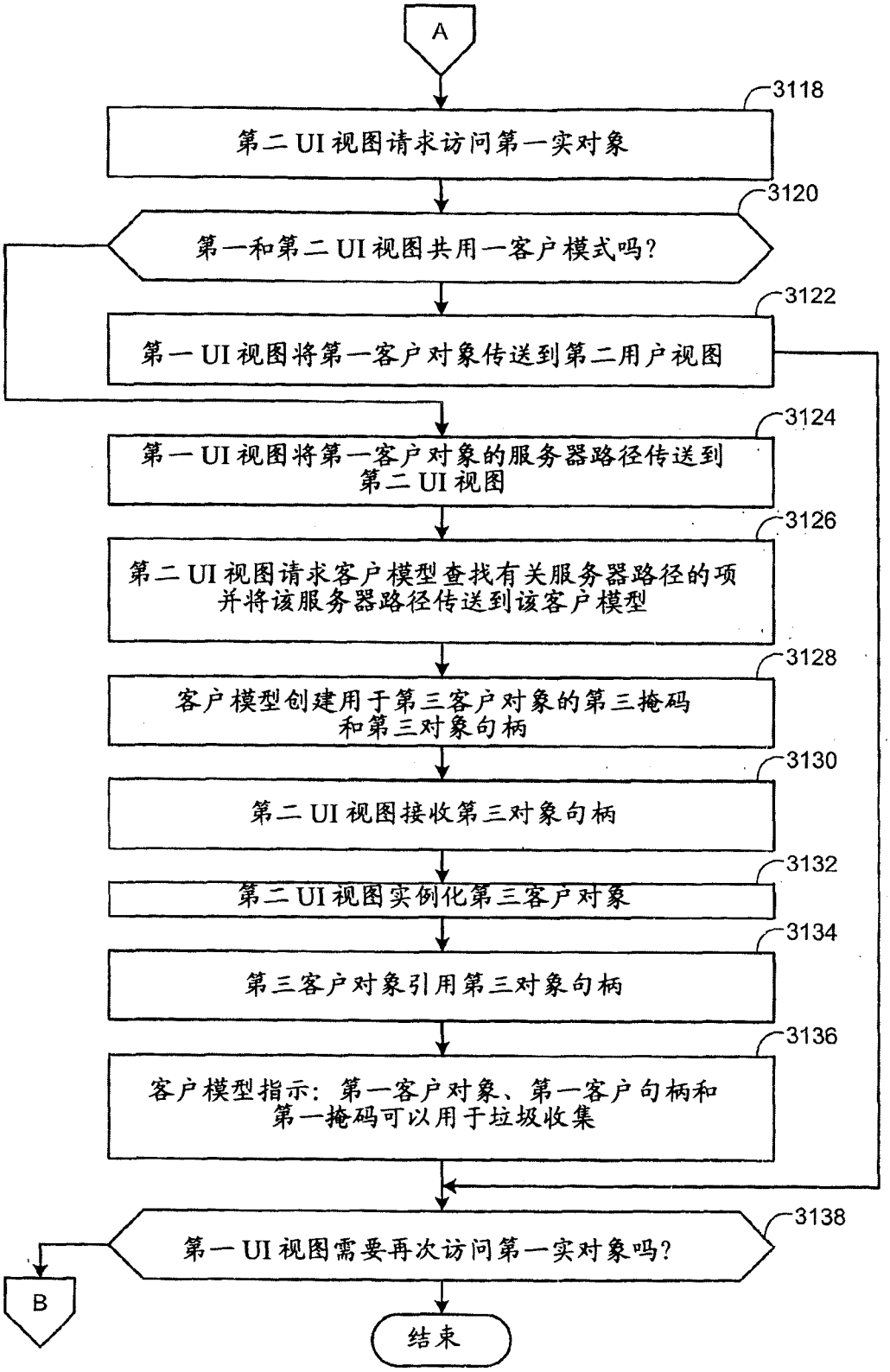
A
3118
第二UI视图请求访问第一实对象
3120
第一和第二UI视图共用一客户模式吗?
3122
第一UI视图将第一客户对象传送到第二用户视图
3124
第一UI视图将第一客户对象的服务器路径传送到第二UI视图
3126
第二UI视图请求客户模型查找有关服务器路径的项并将该服务器路径传送到该客户模型
3128
客户模型创建用于第三客户对象的第三掩码和第三对象句柄
3130
第二UI视图接收第三对象句柄
3132
第二UI视图实例化第三客户对象
3134
第三客户对象引用第三对象句柄
3136
客户模型指示:第一客户对象、第一客户句柄和第一掩码可以用于垃圾收集
3138
第一UI视图需要再次访问第一实对象吗?
B
结束

图 31B

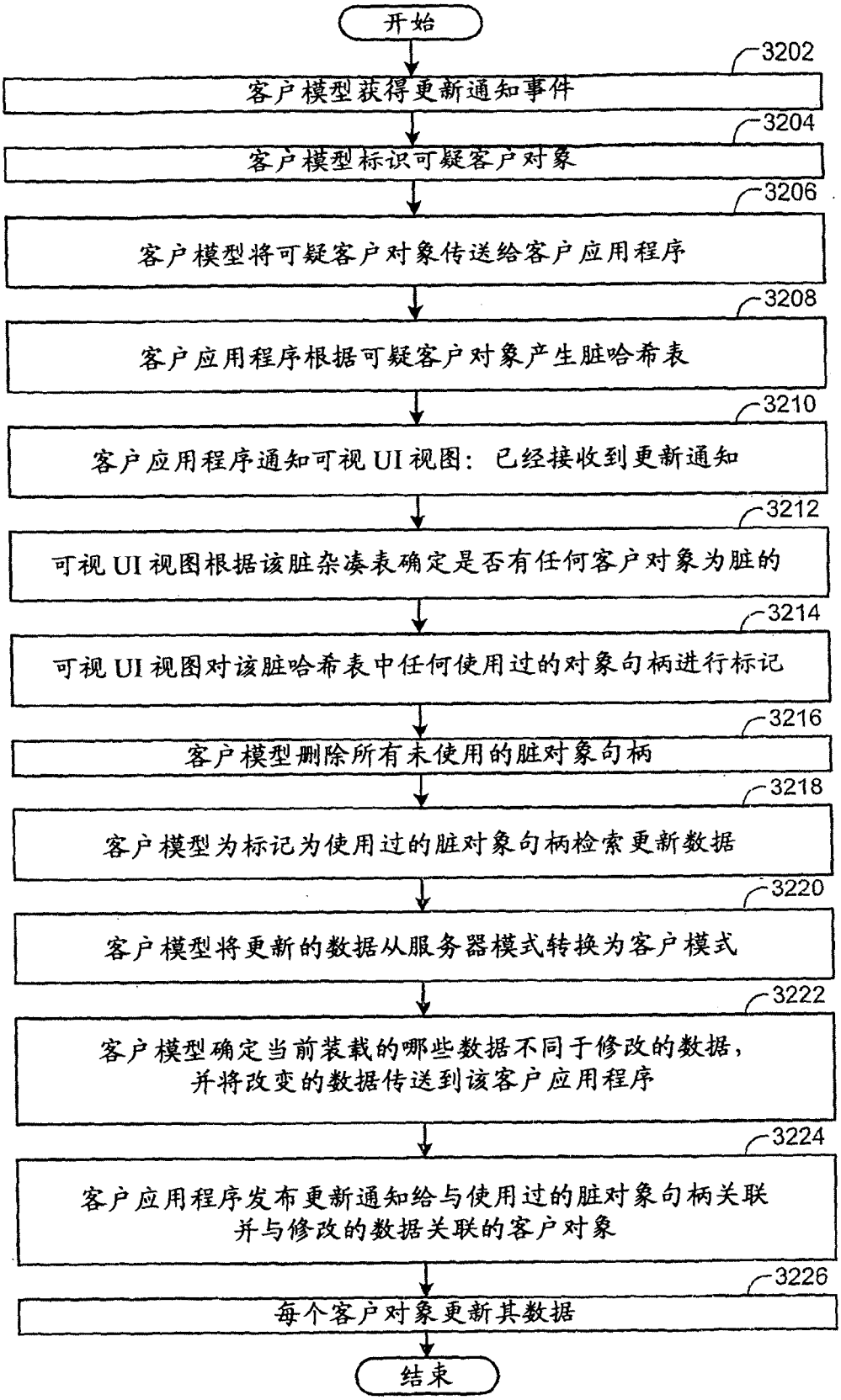
开始
3202
客户模型获得更新通知事件
3204
客户模型标识可疑客户对象
3206
客户模型将可疑客户对象传送给客户应用程序
3208
客户应用程序根据可疑客户对象产生脏哈希表
3210
客户应用程序通知可视 UI 视图：已经接收到更新通知
3212
可视 UI 视图根据该脏杂凑表确定是否有任何客户对象为脏的
3214
可视 UI 视图对该脏哈希表中任何使用过的对象句柄进行标记
3216
客户模型删除所有未使用的脏对象句柄
3218
客户模型为标记为使用过的脏对象句柄检索更新数据
3220
客户模型将更新的数据从服务器模式转换为客户模式
3222
客户模型确定当前装载的哪些数据不同于修改的数据，
并将改变的数据传送到该客户应用程序
3224
客户应用程序发布更新通知给与使用过的脏对象句柄关联
并与修改的数据关联的客户对象
3226
每个客户对象更新其数据
结束

图 32

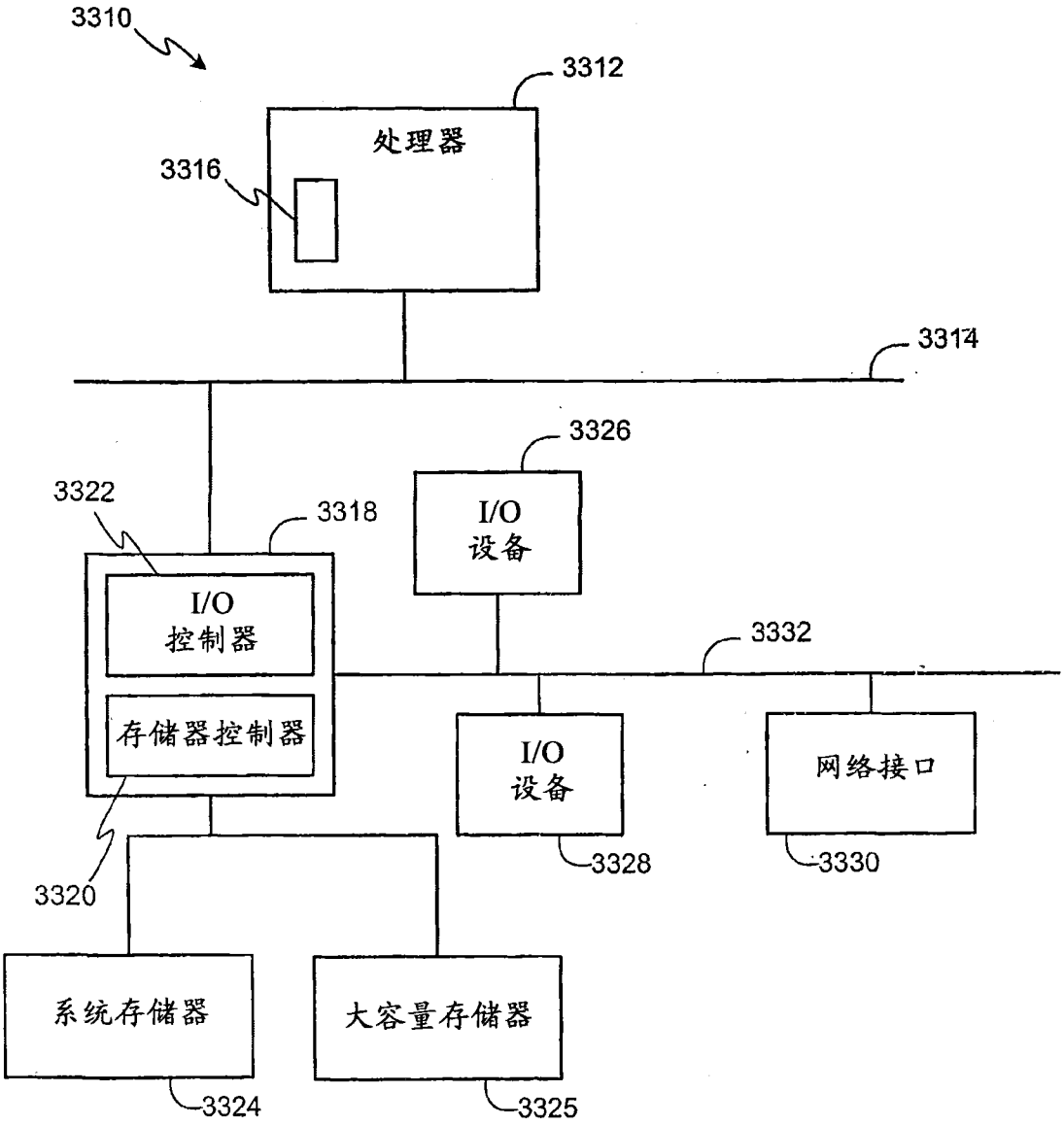
3310
3312
处理器
3316
3314
3326
I/O
设备
3322
3318
I/O
控制器
存储器控制器
3332
I/O
设备
网络接口
3328
3330
3320
系统存储器
大容量存储器
3324
3325

图 33