[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/45

[12] 发明专利申请公开说明书

[21] 申请号 03808883.5

[43] 公开日 2005 年 7 月 27 日

[11] 公开号 CN 1647042A

[22] 申请日 2003.2.14 [21] 申请号 03808883.5

[30] 优先权

[32] 2002. 2.22 [33] US [31] 60/359,409

[32] 2002. 8.30 [33] US [31] 10/233,118

[86] 国际申请 PCT/US2003/004723 2003.2.14

[87] 国际公布 WO2003/073275 英 2003.9.4

[85] 进入国家阶段日期 2004.10.20

[71] 申请人 BEA 系统公司

地址 美国加利福尼亚州

[72] 发明人 K·W·马文 D·包三世

R·A·查维兹

[74] 专利代理机构 中国专利代理(香港)有限公司

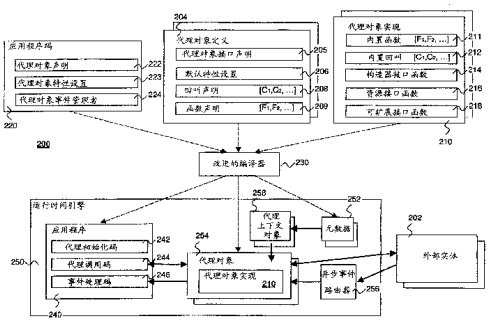
代理人 吴立明 王 勇

权利要求书 10 页 说明书 23 页 附图 16 页

[54] 发明名称 定制软件抽象的方法

[57] 摘要

通过本发明的一个代理对象(图 2 的 254)的实现抽象一个实体，包括在运行时供一个软件应用程序以程序设置的方式与该实体(202)互动的函数，以及可选的用于支持说明性代理对象定义类(204)的开发和定制的编译时和运行时实现类。 在一个实施例中，提供用于辅助实现类的创建的构造器(214)和资源接口(216)。 在一个实施例中，也提供方便代理对象(254)的实现的扩展的扩展接口(218)。 包括代理对象函数(209)的程序设置方式的使用的软件应用程序，被编译成具有代理对象和元数据文件的目标码。 编译后的目标码被用运行引擎(250)执行，运行引擎包括用于方便逐个方式的互动的代理上下文对象(258)和用于传送实体(202)的异步事件的异步事件路由器(256)。



ISSN1008-4274

1．一个计算环境，包含：

一个定义为方便一个软件应用程序与一个实体互动而要被生成的一个代理对象的代理对象定义，该代理对象定义标识该代理对象的至少一个实现，该代理对象的实现具有一个或多个用于方便该应用程序以程序设置的方式与该实体互动的内置函数；和

该代理对象的实现。

2．权利要求1的计算环境，其中，该代理对象定义进一步包括一个说明该代理对象定义以直接和间接方式中被选择的一种方式扩展一个标记接口的说明。

3．权利要求1的计算环境，其中，该代理对象定义进一步包括一个说明一个用于方便在运行期间由该实体生成的一个或多个异步事件的处理的回叫的说明。

4．权利要求1的计算环境，其中，

该代理对象实现支持一个或多个可设置的特性；和

该代理对象定义包括一个或多个限定对该一个或多个可设置特性的设置的设置详细说明。

5．权利要求1的计算环境，其中，该代理对象实现进一步实现一个构造器接口，用于在该软件应用程序的编译期间辅助至少确认为与该实体互动而由该软件应用程序对该代理对象实现的内置函数的使用。

6．权利要求1的计算环境，其中，该代理对象实现进一步实现一个资源接口，用于为要被生成的代理对象配备获得和释放运行时资源的能力。

7．权利要求1的计算环境，其中，该代理对象实现进一步实现一个可扩展接口，用于使该代理对象实现能被扩展。

8．一种编译器，包含：

一个用于分析一个源形式的软件应用程序的分析器，包括分析逻辑，用于辨别和理解一个定义为方便一个软件应用程序与一个实体互动而要被生成的一个代理对象的代理对象定义，该代理对象定义标识该代理对象的至少一个实现，该代理对象的实现具有一个或多个用于

方便该应用程序以程序设置的方式与该实体互动的内置函数；和

一个与该分析器相连的生成器，用于至少部分地根据所述分析的结果、该代理对象定义以及该代理对象的实现，生成可执行形式的该软件应用程序和该代理对象。

9．权利要求 8 的编译器，其中：

该分析逻辑包括用于辨别和理解该代理对象实现的一个或多个可设置特性的设置详细说明的逻辑；和

该生成器包括用于生成包括所述的该代理对象实现的一个或多个可设置特性的设置详细说明的一组元数据的生成逻辑。

10．权利要求 8 的编译器，其中，该生成器包括用于随同该软件应用程序的可执行形式包括用于在该软件应用程序的执行期间实例化该代理对象的代理实例化码的生成逻辑。

11．权利要求 10 的编译器，其中，该代理初始化码被配备得向一个变量分配该代理对象的一个实例，并为接收该实体的异步事件的通知登记该代理对象。

12．权利要求 10 的编译器，其中，该生成逻辑被设计得随同该软件应用程序的可执行形式包括该代理实例化码，其方式要确保在该代理对象的实现的函数由该软件应用程序为与该实体互动而作的任何程序设置方式的使用之前，该代理实例化码被执行。

13．权利要求 8 的编译器，其中，该生成器包括用于随同该软件应用程序的可执行形式包括用于接收该代理对象的实现的内置函数的程序设置方式的调用的一个或多个接口函数的生成逻辑。

14．权利要求 13 的编译器，其中，该生成逻辑被配备得为该代理对象的实现的每个内置函数生成该代理对象的一个接口函数，其中每个接口函数调用对应的内置函数来与该实体互动。

15．权利要求 8 的编译器，其中，该生成逻辑被配备得为该代理对象的实现的每个每个增加的非内置函数生成该代理对象的一个接口函数，其中每个接口函数调用一个特殊的调用函数，以为对应的增加的非内置函数完成与该实体的互动。

16．权利要求 15 的编译器，其中，该特殊的调用函数被配备得访问该代理对象的、包括描述该增加的非内置函数的配对(companion)元数据，用于理解要为之完成该特殊调用函数的、该软件应用程序与该

实体之间的互动。

17. 权利要求 8 的编译器，其中，该生成器被配备得随同所生成的代理对象包括一个特殊的调用函数，以为该代理对象说明的一个实现的增加的非内置函数完成与该实体的互动。

18. 权利要求 8 的编译器，其中，该生成器被配备得进一步生成该代理对象的、描述该代理对象的选定的方面的配对元数据。

19. 权利要求 8 的编译器，其中，该生成器被配备得进一步根据该代理对象生成一个代理对象工厂，以方便软件应用程序与该实体之间的 n 路互动。

20. 权利要求 8 的编译器，其中，该生成器包括生成逻辑，用于随同所生成的代理对象包括一个或多个用于从该实体的一个异步事件路有器接收异步事件的报告的接口回叫函数。

21. 权利要求 20 的编译器，其中，该生成逻辑被配备得为该代理对象的一个实现的每个内置回叫生成一个代理对象接口回叫函数，其中每个接口回叫函数为被报告的异步事件要被对应的内置回叫处理而调用对应的内置回叫函数。

22. 权利要求 20 的编译器，其中，该生成逻辑被配备得为该代理对象说明的一个实现的每个增加的非内置回叫生成一个代理对象接口回叫函数，其中每个接口回叫函数为被报告的异步事件要被软件应用程序的一个对应的事件处理程序处理而调用该件应用程序的该对应的事件处理程序。

23. 一种计算环境，包含：

一个编译器，用于将一个软件应用程序从源形式编译成可执行的形式，包括生成一个配对的代理对象，用于方便该软件应用程序与一个实体互动，源形式的软件应用程序包括程序化表达的、采用该代理对象的一个实现的函数与该实体的互动；和

一个运行引擎，用于方便该可执行形式的软件应用程序以及该代理对象的执行，该运行引擎包括一个异步事件路由器，用于通过该代理对象向该实体的软件应用程序报告异步事件。

24. 权利要求 23 的计算环境，其中，该编译器被配备得能分析源形式的该软件应用程序，包括分析逻辑，用于辨别和理解至少从以下选择的一个：一个该代理对象的使用的说明、该代理对象的一个实现

的一个特性的一个设置详细说明、以及一个限定用于为该实体而被报告的事件的事件处理程序的处理程序详细说明。

25．权利要求 23 的计算环境，其中，该编译器被配备得至少部分地根据分析源形式的软件应用程序的结果、代理对象的定义和代理对象的实现，生成可执行形式的软件应用程序和配对的代理对象。

26．权利要求 23 的计算环境，其中，该编译器包括用于随同该软件应用程序的可执行形式包括用于在该软件应用程序的执行期间实例化该代理对象的代理实例化码的生成逻辑。

27．权利要求 23 的计算环境，其中，该编译器包括用于随同被生成的代理对象包括用于接收该代理对象的实现的内置函数的程序设置方式的调用的一个或多个接口函数的生成逻辑。

28．权利要求 23 的计算环境，其中，该编译器被配备得随同所生成的代理对象包括一个特殊的调用函数，以通过该代理对象的实现的增加的非内置函数完成与该实体的互动。

29．权利要求 23 的计算环境，其中，该编译器被配备得进一步生成该代理对象的、描述该代理对象的选定的方面的配对元数据。

30．权利要求 23 的计算环境，其中，该编译器被配备得进一步根据该代理对象成一个代理对象工厂，以方便软件应用程序与该实体之间的 n 路互动。

31．权利要求 23 的计算环境，其中，该编译器包括生成逻辑，用于随同所生成的代理对象包括一个或多个用于从该实体的一个异步事件路有器接收异步事件的报告的接口回叫函数。

32．一种运行环境，包含：

一个加载器，用于加载一个可执行形式的软件应用程序供执行，该软件应用程序包括为方便一个代理对象的初始化而被包括的代理初始化码，该代理对象的实现包括该软件应用程序以程序设置的方式用来与一个实体互动的函数；和

一个异步事件路由器，用于通过该代理对象向该软件应用程序报告该实体的异步事件。

33．权利要求 32 的运行环境，其中，该代理对象包括一个或多个用于以程序设置的方式接收该代理对象的实现的函数的调用的接口函数。

34．权利要求 32 的运行环境，其中，该代理对象的实现包括一个特殊的调用函数，用于通过该代理对象的实现的增加的非内置函数完成与该实体的互动。

35．权利要求 32 的运行环境，其中，该代理对象有一个描述该代理对象的选定的方面的配对元数据，包括该代理对象的实现的设置和可配置特性。

36．权利要求 32 的运行环境，其中，该代理对象有一个根据该代理对象的代理对象工厂，用于方便该软件应用程序与该实体之间的 n 路互动。

37．权利要求 32 的运行环境，其中，该异步事件路由器被配备得收听由该实体报告的异步事件。

38．权利要求 37 的运行环境，其中，该异步事件路由器被配备得将所检测到的由该实体报告的异步事件报告给该代理对象的接口回叫函数中被选择的接口回叫函数。

39．权利要求 38 的运行环境，其中，接口回叫函数中被选择的接口回叫函数包含该代理对象的一个默认的实现的一个内置回叫的一个接口回叫函数，其中该接口回叫函数为被报告的异步事件要被一个对应的内置回叫处理而调用该对应的内置回叫函数。

40．权利要求 38 的运行环境，其中，接口回叫函数中被选择的接口回叫函数包含该代理对象的该实现的一个增加的非内置回叫的一个接口回叫函数，其中该接口回叫函数为被报告的异步事件要被该软件应用程序的一个对应的事件处理程序处理而调用该该软件应用程序的该对应的事件处理程序。

41．在一个计算环境中的一种方法，包含：

定义一个为了方便一个软件应用程序与一个实体互动而要被生成的代理对象，包括该代理对象的至少一个实现的标识，该实现具有一个或多个用于方便该软件应用程序以程序设置的方式与该实体互动的函数；和

提供该代理对象的该实现。

42．权利要求 41 的方法，其中所述定义进一步包括以一个标记接口的直接扩展和间接扩展中的选定的一个说明该代理对象的定义。

43．权利要求 41 的方法，其中所述定义进一步包括说明一个用于

在运行期间处理由该实体生成的一个或多个异步事件的回叫。

44．权利要求 41 的方法，其中

被提供的代理对象实现支持一个或多个可设置特性；和

所述定义进一步包括说明一个或多个用于限定该一个或多个可设置特性的一个或多个设置的设置详细说明。

45．权利要求 41 的方法，其中所述提供包含一个实现一个构造器接口的代理对象实现，该构造器接口用于在该软件应用程序的编译期间辅助至少确认为与该实体互动而由该软件应用程序对该代理对象实现的内置函数的使用。

46．权利要求 41 的方法，其中所述提供包含一个实现一个资源接口的代理对象实现，该资源接口用于为要被生成的代理对象配备获得和释放运行时资源的能力。

47．权利要求 41 的方法，其中所述提供包含一个实现一个可扩展接口的代理对象实现，该可扩展接口用于使该代理对象实现能被扩展。

48．一种软件编译方法，包含：

分析一个源形式的软件应用程序，包括辨别和理解一个定义为方便一个软件应用程序与一个实体互动而要被生成的一个代理对象的代理对象定义，该代理对象定义标识该代理对象的至少一个实现，该代理对象的实现具有一个或多个用于方便该应用程序以程序设置的方式与该实体互动的内置函数；和

至少部分地根据所述分析的结果、该代理对象定义以及该代理对象的实现，生成可执行形式的该软件应用程序和该代理对象。

49．权利要求 48 的软件编译方法，其中

所述分析包含辨别和理解该代理对象实现的一个或多个可设置特性的设置详细说明；和

所述生成包含生成包括所述的该代理对象实现的一个或多个可设置特性的设置详细说明的一组元数据。

50．权利要求 48 的软件编译方法，其中，该所述生成包含随同该软件应用程序的可执行形式包括用于在该软件应用程序的执行期间实例化该代理对象的代理实例化码。

51．权利要求 50 的软件编译方法，其中，该代理初始化码被配备得向一个变量分配该代理对象的一个实例，并为接收该实体的异步事

件的通知登记该代理对象。

52．权利要求 50 的软件编译方法，其中，所述生成进一步包含随同该软件应用程序的可执行形式包括该代理实例化码，其方式要确保在该代理对象的实现的函数由该软件应用程序为与该实体互动而作的任何程序设置方式的使用之前，该代理实例化码被执行。

53．权利要求 48 的软件编译方法，其中，所述生成包含随同该软件应用程序的可执行形式包括用于接收该代理对象实现的内置函数的程序设置方式的调用的一个或多个接口函数。

54．权利要求 53 的软件编译方法，其中，所述生成包含为该代理对象的实现的每个内置函数生成该代理对象的一个接口函数，其中每个接口函数调用对应的内置函数来与该实体互动。

55．权利要求 48 的软件编译方法，其中，所述生成包含为该代理对象的实现的每个每个增加的非内置函数生成该代理对象的一个接口函数，其中每个接口函数调用一个特殊的调用函数，以为对应的增加的非内置函数完成与该实体的互动。

56．权利要求 55 的软件编译方法，其中，该特殊的调用函数被配备得访问该代理对象的、包括描述该增加的非内置函数的配对元数据，用于理解要为之完成该特殊调用函数的、该软件应用程序与该实体之间的互动。

57．权利要求 48 的软件编译方法，其中，所述生成包含随同所生成的代理对象包括一个特殊的调用函数，以为该代理对象说明的一个实现的增加的非内置函数完成与该实体的互动。

58．权利要求 48 的软件编译方法，其中，所述生成包含生成该代理对象的、描述该代理对象的选定的方面的配对元数据。

59．权利要求 48 的软件编译方法，其中，所述生成包含根据该代理对象生成一个代理对象工厂，以方便软件应用程序与该实体之间的 n 路互动。

60．权利要求 48 的软件编译方法，其中，所述生成包含随同所生成的代理对象包括一个或多个用于从该实体的一个异步事件路有器接收异步事件的报告的接口回叫函数。

61．权利要求 60 的软件编译方法，其中，所述生成包含为该代理对象的一个实现的每个内置回叫生成一个代理对象接口回叫函数，其

中每个接口回叫函数为被报告的异步事件要被对应的内置回叫处理而调用对应的内置回叫函数。

62．权利要求60的软件编译方法，其中，所述生成包含为该代理对象说明的一个实现的每个增加的非内置回叫生成一个代理对象接口回叫函数，其中每个接口回叫函数为被报告的异步事件要被软件应用程序的一个对应的事件处理程序处理而调用该件应用程序的该对应的事件处理程序。

63．一种计算方法，包含：

将一个软件应用程序从源形式编译成可执行的形式，包括生成一个配对的代理对象，用于方便该软件应用程序与一个实体互动，源形式的软件应用程序包括程序化表达的、采用该代理对象的一个实现的函数与该实体的互动；和

方便该可执行形式的软件应用程序以及该代理对象的执行，包括通过该代理对象收听并向该软件应用程序传送由该实体报告的异步事件。

64．权利要求63的计算方法，其中，所述编译包含分析源形式的该软件应用程序，包括辨别和理解至少从以下选择的一个：一个该代理对象的使用的说明、该代理对象的一个实现的一个特性的一个设置详细说明、以及一个限定用于为该实体而被报告的事件的事件处理程序的处理程序详细说明。

65．权利要求63的计算方法，其中，所述编译包含至少部分地根据分析源形式的软件应用程序的结果、代理对象的定义和代理对象的实现，生成可执行形式的软件应用程序和配对的代理对象。

66．权利要求63的计算方法，其中，所述编译包含随同该软件应用程序的可执行形式包括用于在该软件应用程序的执行期间实例化该代理对象的代理实例化码。

67．权利要求63的计算方法，其中，所述编译包含随同被生成的代理对象包括用于接收该代理对象的实现的内置函数的程序设置方式的调用的一个或多个接口函数。

68．权利要求63的计算方法，其中，所述编译包含随同所生成的代理对象包括一个特殊的调用函数，用于通过该代理对象的实现的增加的非内置函数完成与该实体的互动。

69．权利要求63的计算方法，其中，所述编译包含生成该代理对象的、描述该代理对象的选定的方面的配对元数据。

70．权利要求63的计算方法，其中，所述编译包含根据该代理对象成一个代理对象工厂，用于方便软件应用程序与该实体之间的 n 路互动。

71．权利要求63的计算方法，其中，所述编译包含随同所生成的代理对象包括一个或多个用于从该实体的一个异步事件路有器接收异步事件的报告的接口回叫函数。

72．一种执行方法，包含：

加载一个可执行形式的软件应用程序供执行，该软件应用程序包括为方便一个代理对象的初始化而被包括的代理初始化码，该代理对象的实现包括该软件应用程序以程序设置的方式用来与一个实体互动的函数；和

通过该代理对象向该软件应用程序报告该实体的异步事件。

73．权利要求72的运行方法，其中，该方法进一步包含可执行形式的软件应用程序的代理调用码，其调用该代理对象的一个或多个接口函数，用于方便该代理对象的实现的函数的程序设置方式的调用。

74．权利要求72的运行方法，其中，该方法进一步包含可执行形式的软件应用程序的代理调用码，其调用该代理对象的一个特殊的调用函数，用于用该代理对象实现的增加的非内置函数完成与该实体的互动。

75．权利要求72的运行方法，其中，该方法进一步包含该代理对象访问描述该代理对象的选定的方面的配对元数据。

76．权利要求72的运行方法，其中，该方法进一步包含根据该代理对象生成一个代理对象工厂，用于方便该软件应用程序与该实体之间的n路互动。

77．权利要求76的运行方法，其中，该方法进一步包含一个异步事件路由器收听由该实体报告的异步事件。

78．权利要求77的运行方法，其中，该方法进一步包含该异步事件路由器所检测到的由该实体报告的异步事件报告给该代理对象的接口回叫函数中被选择的接口回叫函数。

79．权利要求78的运行方法，其中，该方法进一步包含该代理对

象的对应该代理对象实现的一个内置回叫的一个接口回叫函数，其中该接口回叫函数为被报告的异步事件要被一个对应的内置回叫处理而调用该对应的内置回叫函数。

80．权利要求 79 的运行方法，其中，该方法进一步包含一个增加的非内置回叫对应的一个接口回叫函数，其中该接口回叫函数为被报告的异步事件要被该软件应用程序的一个对应的事件处理程序处理而调用该该软件应用程序的该对应的事件处理程序。

定制软件抽象的方法

相关发明

本发明涉及临时申请号 60/359,409 的美国专利申请“A UNIFIED FRAMEWORK FOR INTERACTING WITH EXTERNAL ENTITIES FROM A PROGRAMMING LANGUAGE AND EXAMPLE APPLICATIONS”（申请日 2002 年 2 月 22 日），并要求其优先权，特此全文引用其详细说明作为参考。

发明领域

本发明涉及数据处理领域。更具体来说，本发明涉及软件互动(interaction)方法。

发明背景

在开发现代软件应用程序的过程中，开发者经常遇到与外观、感觉和性能不像熟悉的内部程序设计语言对象的外部实体。这些外部实体包括数据库、传统（legacy）系统、web 服务、非本机（non-native）软件组件、以及(例如用于控制它们的设置的)物理对象。.

一般来说，程序员必须学习用于与这些实体的每一个互动的新范例(paradigms)、技巧和技术。此外，程序员必须开发和获得可能是巨大部分的软件才能处理每一种外部实体的错综复杂。例如，编写与外部网络服务(web service)互动的软件，可能要求开发者掌握几种技术，包括可扩展标记语言(XML)、XML 模式语言、XML 协议(XP，也叫 SOAP)和网络服务描述语言(WSDL)。

应用程序(applications)与之互动的每种外部实体，都需要开发者的一个不同的技巧、知识和软件集合。例如，与数据库互动需要有与外部网络服务互动完全不同的技巧、知识和软件集合。学习和实现用于与各种外部实体互动的技术所带来的额外负担增加开发软件应用程序所需的时间、知识、技巧以及终归的金钱。

所需要的是一种提供一致的对外部实体的使用(access)、重新使用开发者已有的通用软件概念的知识、并且最小化与每种外部实体互动所需的专业知识的简单软件抽象(abstraction)。这些软件抽象不但应当容易使用，而且也应当容易定制得适合特定用途，而无需多少或者不需要软件开发。软件抽象应当使得容易同时与外部实体的几个实

例互动(例如用一个网络服务的几个实例同时为几个顾客进行信用审查。也应当简化对由一个外部实体的一个或多个实例生成的异步事件(例如数据库触发器(trigger))的处理。

例如考察图 1 的计算环境。所示的计算环境例子 100 包括服务器 102、115、 120 和 125，以及通过网络组织 101 通信地连接的客户机 112。

服务器 125 例如提供电子商务应用程序 130，各种客户装置的用户—例如客户 112，可以用它来购买各种东西。电子商务应用程序 130 可以包括多个网页(web pages)131，网页具有诸如商品说明、评论和价格信息的内容以及一个或多个函数(functions)132。

作为补充，服务器 102 例如提供（例如在客户装置的用户与它们的网页/函数 131/132 互动时）由诸如电子商务应用程序 130 的电子商务应用程序使用的购物车服务 104。因此，电子商务应用程序的开发者不需要开发他们自己的“购物车”函数，因此可以转而他们的努力专注于网页 131 内容上。

购物车服务 104 例如可以包括一个“向购物车中加一个东西”函数、一个“从购物车中去掉一个东西”函数、以及“结帐”函数。响应用户例如对网页 131 上显示的一个图形按钮的选择，函数 132 之一可以导致一个或多个对相应购物车函数的请求被生成并被发送到购物车服务 104，供在服务器 102 上处理。

这些请求的处理，可能进而要求购物车函数例如与服务器 115 和 120 的服务 118 和 128 互动。服务 118 和 128 的例子是信用授权、存货或产品位置确认、出货/发货时间安排、等等。

因此，即使在这个有限的例子中，电子商务应用程序 130 的开发者也要装备应用程序 130 才能与外部的“购物车服务”104 互动，而“购物车服务”104 则要装备服务 104 才能与诸如信用授权、存货或产品位置确认、出货/发货时间安排等等的外部实体互动。

除了为便于这些互动所需的通用网络服务软件(例如 XML, SOAP, WSDL) 、每种网络服务 (例如购物车、信用授权、运送)都要求编写专业软件来与它的特定功能部件(features)互动。例如，必须编写软件，以创建“向购物车添加东西”XML 消息并将它发送到与正确的购物篮相关联的 URL 地址。此外，必须编写软件，以把由信用审查和运送服

务返回的消息变换成可以向终端用户表示的形式。

如所属技术领域的熟练人员能理解的那样，通常，该例子要求与每个外部服务的几个实例同时发生的和异步的互动。例如，在任何时刻，一个电子商务应用程序 130 都可能正在接待(hosting)许多用户，每个用户有单独的购物车，内含不同的东西。购物车服务 104 进而可能正在为不同电子商务应用程序的不同用户的许多购物车处理购物车互动。同样，一个信用授权服务可能正在同时处理许多电子商务应用程序中出现的多个结账的授权请求。对这些交易的每一个的最终信用核准和运送安排的通知，可能在顾客已经完成他们的订单之后异步地进行(例如通过电子邮件)。很清楚，将与关系每个顾客的购物车、信用授权、运送细节和通知进行关联并与其他顾客的这些内容分开是很关键的，即使在同时处理几个顾客请求时也如此。处理所要求的关联和异步事件处理所需的软件可能变得相当复杂。

即使在这个相对简单的例子中，显然应用程序开发者一般也必须获得新的专业技巧、知识和软件才能与外部实体互动。他们必须经常创建用于与每个外部实体的特定形式互动的专用软件。他们也必须应对同时地和异步地与一个给定类型的实体的几个实例互动的复杂性。因此，需要一种提供一致的对外部实体的使用、简化为特定类型的外部实体创建专用软件抽象、并且方便同时地和异步地与外部实体的多个实例互动的简单软件抽象。

附图说明

现在将通过附图中所示的示例性实施例非限定地说明本发明。附图中相似的标注代表相似的元件，其中：

图 1 表示一例现有技术的计算环境；

图 2 表示按照一个实施例的本发明的总体；

图 3 表示按照一个实施例的为外部实体指定代理对象的方法；

图 4 表示按照一个实施例的、通过与一个标记代理-对象接口(marker proxy-object interface)的直接或间接的关联而限定一个软件对象是一个外部实体的代理对象的方法；

图 5 表示一例外部定时器(timer)的代理对象定义的详细说明(specification)例子；

图 6 更详细地表示按照一个实施例的、图 2 的代理对象实现；

图 7 表示按照一个实施例的一例限定（specify）可用于定制一个代理对象的行为（behavior）的元数据特性（meta-data properties）的语法的 XML 文档；

图 8 表示按照一个实施例的、本发明的包括使用外部实体的软件抽象的应用程序开发方法；

图 9a-9c 表示说明(declare)一个代理对象、设定它的特性和处理它的异步事件的详细说明的例子；

图 10a 表示一个用于一个外部实体的、通过定制其特性的默认值而扩展图 5 的代理对象定义的代理对象定义的详细说明的例子；

图 10b 表示一个通过说明一个新函数和相关联的缺生特性设定而扩展一个现有代理对象定义的代理对象定义的详细说明的例子；

图 11a-11b 表示按照本发明一个实施例的、图 2 的增强的编译器的相关方面的操作流程；

图 12 更详细地表示按照一个实施例的、图 2 的代理对象；

图 13a 表示按照本发明一个实施例的、图 2 的运行环境(runtime environment)的相关方面的操作流程；

图 13b 表示一例执行流程；

图 14a-c 表示用于说明一个代理对象工厂、用一个代理对象工厂创建代理对象以及处理与生成的代理对象相关联的异步事件的详细说明；

图 15 表示按照本发明一个实施例的一例适合用来实践本发明的计算机系统。

具体实施方式

本发明包括用于简化作为用于与来自一个软件应用程序内的外部实体互动的软件抽象的代理对象的开发、定制和使用的方法和设备。

在以下的说明中，将说明本发明的各个方面。然而，所属技术领域的熟练人员将明白，只以本发明的一些方面或所有方面都能实现本发明。为了方便解释，陈述了具体的数字、材料和配置，以便于透彻地理解本发明。然而，对于所属技术领域的熟练人员来说，即使没有这些具体细节显然也可以实现本发明。在其它实例中，为了不妨碍对本发明的说明，省略或简化了众所周知的功能部件。

术语

说明书的各部分将以所属技术领域的熟练人员向该领域其他熟练人员传达他们工作的实质所通常采用的方式一致的方式，用数据处理术语进行表达，这些术语诸如有数据、选择、检索、生成、等等。正如所属技术领域的熟练人员所明白的那样，这些数量采取电、磁或光信号的形式，能被存储、传送、组合和以其它方式通过处理器和其子系统的电和/光部件被操纵。

说明书有的部分将采用各种缩略语，包括—但不限于：

URL(同一资源定位器)

XML(可扩展标记语言)

本申请(包括权利要求书)中所使用的术语“外部实体”，不仅指外部软件实体，也指硬件实体。“外部”是从与该实体互动的软件应用程序的角度来看的。

段落标题、说明和实施例的次序

段落标题仅仅用来改善可读性，不应认为它们限制或缩小本发明的范围。

各种操作将依次以分立的步骤的形式、以最有助于理解本发明的方式进行说明。然而不应认为说明的顺序意味着这些操作必然是与顺序相关的。特别地，这些操作并不需要按所陈述的顺序执行。

短语“在一个实施例中”被再三使用。该短语一般不是指相同的实施例，不过它可以指相同的实施例。术语“包含”、“有”和“包括”是同义词，除非上下文中另外指出。

概述

图 2 表示按照本发明一个实施例的、本发明的概述。如图所示，为了简化开发与外部实体 202 互动的软件应用程序 240，本发明提供便于为外部实体 202 提供代理对象 254 的方法和装置，使得软件应用程序 240 可以用软件开发者熟悉的通用程序设计概念以程序设置的方式(programmatically)与外部实体 202 互动。

更具体来说，开发者可以为外部实体 202 创建一个代理对象定义 204。该开发者可以是外部实体 202 的开发者，第三方开发者、甚至是应用程序 240 的开发者。

代理对象定义 204 包括确定根据对与外部实体互动的定义应被生成的一个代理对象 254 的接口说明(interface declaration)205。另

外的代理对象定义 204 包括用于定义代理对象 254 的默认行为和默认实现(implementation)210 的默认特性设置 206、用于处理来自外部实体 202 的回叫说明(callback declarations)208 和用于启动与外部实体 202 互动的函数说明(function declarations)209。

在一个实施例中，一个或多个代理对象实现类(implementation classes)210 包括一个运行时(run-time)实现类。在另一个实施例中，该一个或多个代理对象实现类 210 进一步包括一个编译时(compile-time)实现类。在另一个实施例中，实现类 210 进一步包括一个设计时(design-time)实现类。

运行时实现类为在代理对象说明 204 中所说明的并由软件应用程序码 220 使用的函数提供运行时实现，用于以程序设置的方式与外部实体 202 互动。运行时实现类可以提供一个或多个用于启动与外部实体 202 互动的内置(built-in)函数 211 和一个或多个用于处理由外部实体 202 生成的外部事件的内置回叫 212。

可选的编译时实现类，提供编译时确认实现(validation implementation)，以在编译期间辅助编译器 230 确认由代理对象定义 204 和由应用程序码 220 对函数和特性设置的使用。

可选的设计时实现类提供用于辅助代理对象定义 204 和应用程序码 220 的开发者的设计时实现类。它辅助开发者以程序设置的方式扩展和使用由运行时实现执行的用于与外部实体 202 互动的特性和函数。这种设计时实现的一个例子包括—但不限于——一个图解魔板，它引导开发者在给定外部网络服务的 WSDL 说明的条件下创建该网络服务的代理对象定义。另一个例子是提供对应于代理对象定义 204 的函数的用法的图标，在为一个应用程序码 220 被选择时，它把对应的函数调用插入到该应用程序码 220 中。

对于所示的实施例来说，代理对象实现 210 可以实现一个或多个接口 214-218。特别地，对于该实施例来说，代理对象实现 210 可以实现构造器接口(builder interface) 214、资源接口(resource interface)216 和可扩展接口(extensible interface)218。

构造器接口 214 可以由代理对象实现 210 的编译时组件实现，以辅助编译器 230 确认由代理对象实现 210 实现的特性和函数的使用。资源接口 216 可以由代理对象实现 210 的运行时组件实现，以获得和

释放为代理对象实现所需要的关键资源，诸如数据库和文件柄(handles)。可扩展接口 214 可以由代理对象实现 210 的运行时组件实现，以使代理对象定义 204 能说明未内置到代理对象实现 210 的新函数。

仍然参看图 2，一旦创建了代理对象定义 204 和代理对象实现 210，应用程序码 220 的开发者就可以配备应用程序 240，以通过包括代理对象说明 222 和调用在作为结果的代理对象上说明的函数 209，启动与外部实体 202 的互动。应用程序码也可以包括定制代理对象的行为的特性设置 223，或者包括处理由外部实体 202 生成的异步事件的事件处理程序(handlers)224。

按照本发明配备的软件应用程序码 220、代理对象定义 204 和代理对象实现 210，被用增强的编译器 230 编译成应用程序 240、代理对象 254 和元数据 252。

把编译器 230 加强，以认识代理对象定义 204，生成使用代理对象实现 210 的相关联的代理对象 254，用于方便在运行时与软件实体 202 的互动。编译器也生成为每个代理对象说明 222 创建一个代理对象的代理初始化码 242，将该代理对象分配给被说明的变量，并向异步事件路由器 256 登记该代理对象，以接收由相关联的外部实体 202 生成的合适事件。进一步，编译器 230 被增强，以收集和输出描述特性对象定义 204 的接口、函数、回叫和特性设置的元数据 252，在运行时供对应的代理对象 254 使用。

仍然参看图 2，被编译的目标码在运行期间的执行，在运行引擎(run-time engine)250 的控制之下。运行引擎 250 特别包括代理上下文对象(proxy context objects)258，为每个代理对象调用创建代理上下文对象的一个实例，用于与外部实体 202 的一个实例互动和维持该特定互动的状态信息。对于该实施例来说，互动上下文(interaction context)258 包括许多方法，代理对象实现 210 可以通过这些方法获得关于某个互动的信息。

对于该实施例来说，如前所述，代理对象定义 204 可以说明一个或多个用于处理由对应的外部实体 202 生成的异步事件的回叫函数 208。作为补充，运行引擎 250 包括异步事件路由器 256，用于收听、接收和把由外部实体 202 生成的异步事件传送(routing)到适当的代理

对象 254，供由应用程序 240 的事件处理码 246 处理。异步事件路由器 256 所听的位置，由代理初始化码 242 根据代理对象实现 210 和相关联的特性设置 204 和 223 限定。

开发者可以用上述的机制创建应用程序码 220，以通过调用在代理对象 222 上说明的函数、设定代理对象特性 223 和定义事件处理程序 224，与外部实体 202 互动。以这种方式与外部实体互动，非常类似于与其它软件对象的互动，不要求开发者学习过多的范例、技巧和/或技术。此外，开发者甚至用新的函数和回叫、不必限定这些新函数和回叫的实现，就可以创建新的代理对象定义 204。所得到的代理对象 254 与运行引擎 250 协作，处理多个与外部实体 202 同时发生的和异步的互动。

在各种实施例中，外部实体 202 可以是个 web 服务、数据库或传统系统、以及物理对象。

可选的设计时实现类的提供，不是实践本发明的一个根本方面。此外，这是在所属技术领域的那些熟练人员的能力范围之内的，因此将不作进一步的描述。下面将依次地进一步说明本发明的其它方面。

代理对象定义

图 3 进一步详细地表示按照本发明一个实施的、开发者或设计时工具为开发本发明的代理对象定义 204 所可能采取的操作。如图所示以及前文所示的那样，创建代理对象定义 204 所要采取的行动之一，是限定一个代理对象接口说明 205，框 302。

在一个实施例中，这是通过说明代理对象定义 204 扩展一个特殊的“代理对象”标记接口(图 4 的 402)而完成的。如图 4 所示，标记接口 402 的扩展可以是直接的—如在代理对象定义 404a-404b 的情形中的那样，或者可以是间接的—如在代理对象定义 404c-404i 的情形中的那样。在编译时，增强的编译器 230 将通过寻找扩展标记接口 402 的接口来确定代理对象定义 204，并将为每个这种接口生成代理对象。如果标记接口 402 的扩展是间接的，代理对象定义 204 将继承它扩展的其它代理对象定义的函数、特性和回叫(例如，代理对象定义 404i 将继承由代理对象 404c 和 404a 所定义的函数、特性和回叫)。

回过来参看图 3，如图所示以及前文所示的那样，创建代理对象定义 204 所要采取的另一个行动是为该代理对象定义指定默认特性设

置，框 303。这些设置将在运行时被代理对象实现 210 用来决定代理对象 254 的行为。

此外，程序员或设计时工具可以任选地指定代理对象定义 204 的函数说明 209，框 304。应用程序码 220 可以用所说明的函数以程序设置的方式与外部实体 202 互动。函数说明 209 可以对应于代理对象实现 210 的内置函数 211；或者，如果代理对象实现 210 实现可扩展接口 214，则函数说明 209 可以导入未被代理对象实现 210 明确提供的新函数。

此外，程序员或设计时工具可以任选地限定代表可能是由外部实体 202 在运行时生成的异步事件的回叫函数说明 208。回叫函数说明 208 可能对应于代理对象实现 210 的内置回叫函数 212，在这种情况下，代理对象 254 将把由外部实体 202 生成的对应的异步事件传送（route）到代理对象实现 210 供处理（后者进而把它们传送到应用程序 240 的事件处理码 246）。当回叫函数说明 208 不对应于代理对象实现 210 的内置回叫函数 212 时，代理对象 254 将把由外部实体 202 生成的对应的异步事件直接传送到应用程序 240 的事件处理码 246。

此外，开发者或设计时工具可以限定代理对象定义 204 的实现类，包括运行时实现类、以及—可选地—编译时实现类和/或设计时实现类，块 306。如果代理对象定义 204 扩展另一个限定实现类的代理对象定义，则不必限定实现类。在这种情况下，实现类详细说明是从扩展的代理对象定义继承的。

在一个实施例中，用特性设置构造实现类的详细说明。在一个实施例中，特性设置是以注解（annotation）形式限定的，即以传统上被认为是源文件的注解的形式限定的。

图 5 表示外部定时器实体的一例代理对象定义。所属技术领域的熟练人员将认识这是一个熟悉的 Java 接口定义，它扩展 502 行上的一个称作 com.bea.jws.ProxyObject 的现有接口并包括 510-516 行上的一些特殊的 JavaDoc 注释(comments)。通过限定 Timer (定时器)接口扩展本发明的"ProxyObject"(代理对象)标记接口，Timer 接口被标识为本发明的一个代理对象定义。在这种情况下，Timer 接口直接地扩展 ProxyObject 标记接口；然而，也有可能如图 4 中所示地间接地扩展 ProxyObject 标记接口。

此外，Timer 接口被限定有一个 setTimeoutln(int milliseconds) 函数 504a、一个 setTimeoutAt(java.util.Date date) 函数 504b、等等，供应用程序码 220 在 n 个消逝的时间单元后或者在特定的时刻设定一个“警报”(alarm)。

此外，Timer 接口还包括一个回叫函数 504c，用于在定时器于所请求的时间到点时，处理由外部实体 202 生成的警报事件，例如把警报事件异步地传送到应用程序 240。在一个实施例中，回叫说明是在名为"Callback"(回叫)的嵌套接口中定义的函数，如图 5 中所示。

运行时、编译时和设计时实现类分别被指定为 "com.bea.jws.private.TimerImpl"512 "com.bea.jws.private.TimerValidator" 514 和 "com.bea.jws.private.TimerDesigner" 516。详细说明是用特性设置构造的。在一个实施例中，特性设置在注释段中以注解的形式被限定。如所属技术领域的熟练人员将认识到的那样，本例中的特性设置是用特殊的 Javadoc 注解 @implementation(实现) 510 限定的。

代理对象实现

除了采用可扩展接口、资源接口和/或构造器接口 214-218、使用代理上下文对象 258 以及实现符合预期的执行范例的装置外，无论运行时、编译时还是设计时的每个实现类的核心构造(core constitution)都是依赖于应用程序的。就是说，它们依赖于外部实体 202 的行为和所提供的服务，以及函数的性质。

然而，如前文指出的那样，运行时实现类被指望或者直接地通过内置函数 211 或者间接地通过可扩展接口 218 的"invoke"(调用) 函数在本发明的执行上下文中实现代理对象定义 204 的函数。

图 6 更详细地表示按照一个实施例的代理对象实现 210。如图所示，对于该实施例来说，代理对象实现 210 包括内置函数 211、内置回叫函数 212、构造器接口 214、资源接口 216 和扩展接口 218。

如前文所述的那样，构造器接口 214 在被一个编译时实现类实现时，辅助编译器 230 确认由代理对象实现 204 定义并由应用程序码 220 使用的特性受到代理对象实现 210 的支持。此外，构造器接口可以被一个集成的开发环境用来帮助开发者明白可以在哪里以及如何使用特性。

资源接口 216 在被一个运行时对象实现类实现时，辅助该运行时对象实现类获得和释放资源，诸如数据库连结和文件柄。

可扩展接口 218 在被一个运行时对象实现类实现时，使代理对象定义 204 能说明不被代理对象实现 210 直接支持的新函数，而不定义这些函数如何被实现。

对于所示的实施例来说，构造器接口 218 特别包括一个 Get Property Syntax（得到特性语法）函数 602、 Validate Class Properties(确认类特性)函数 604 和 Validate Field Properties(确认域特性)函数 606。正如这些函数的名字所提示的那样，这些函数被调用时，返回代理对象的一个有效特性语法并确认代理对象的类和域级特性。

在一个实施例中，当 Get Property Syntax 函数 602 被调用时，返回一个标识一个文件的 URL，该文件是由编译时实现类的开发者提供的，以一个 XML 文件的形式标识有效特性语法。

这样一个 XML 文件的片断的例子在图 7 中表示。如图所示，这种片断可以规定特性的名 702a 或 702b、特性的属性(attributes)704a、704b 或 704c—包括是否它们是必需的、属性值的数据类型 706 以及它们的默认值 708—如果适用的话。

对于该片断例来说，它规定"@sql" 特性只被允许在代理对象定义函数 208 的前面，并且在此需要有该特性。@sql 特性可以有 statement(语句)、 maxcount(最大计数)、以及 returnType(返回类型)属性。Statement 属性是必需的。除非另外规定，否则所有属性都必须被赋值。Maxcount 和 returntype 是任选的。Maxcount 取整数值，默认值是 infinity(无穷大)。除非另外规定，否则属性(诸如 Statement 和 returntype) 取串值，默认值是空串。允许@pool 注解在代理对象说明 222、代理对象定义函数 209 以及代理对象定义 204 的前面，并且在所有这些位置都是任选的。最后，@pool annotation 可以有一个名字属性，它是必须出现的并且由一个串值。

在替代实施例中，可以以其它格式或者用其它数据组织技术提供和/或返回信息。

Get Property Syntax 函数 602、Validate Class Properties 函数 604 和 Validate Field Properties 函数 606 的实现，在所属技

术领域的熟练人员的能力范围内，因此将不作进一步的说明。

实现构造器接口 218 使编译时实现类能用这些函数来提供期望的语法以及为编译器验证元数据。

回过来参看图 6，对于所示实施例来说，资源接口 216 包括一个 Acquire Resource(获得资源)函数 612 和 Release Resource (释放资源)函数 614。正如这些函数的名字所提示的那样，函数 612 使代理对象实现 210 能在运行时创建一个代理的每个新实例之前获得该实现所需要的系统资源，诸如数据库连结和文件柄，函数 614 使代理对象实现 210 能在运行时破坏一个代理对象的每个实例之后释放资源。类似地，Acquire Resource 函数 612 和 Release Resource 函数 614 的实现，在所属技术领域的熟练人员的能力范围内，因此将不作进一步的说明。

仍然参看图6,对于所示实施例来说,扩展接口214包括一个Invoke Object(调用对象)函数616. Invoke object 函数 616 被设计得处理由代理对象定义 204 所说明的定制方法的调用。因此，代理对象定义 204 可以说明未被代理对象实现 210 的内置函数 211 明确地实现的新函数 209。在运行期间，当应用程序码 220 调用新函数 209 时，代理对象 254 将把它们打送到代理对象实现 210 的 invoke 函数 616。代理对象实现 210 的 invoke 函数 616 可以通过代理上下文对象 258 访问与代理对象调用 244 有关的名字、自变量(arguments)、返回类型、特性和其它元数据，以确定该 invoke 操作所需的语义(semantics)。该访问可以用例如与代理上下文对象 258 相关联的方法进行。

类似地，Invoke Object 函数 616 的实现，在所属技术领域的熟练人员的能力范围内，因此将不作进一步的说明。

开发应用程序

图 8 表示按照一个实施例的、本发明的包括使用外部实体的软件抽象的应用程序开发方法。如图所示，在框 801，首先创建一个代理对象实现 210，其可选地包括内置函数、内置回叫、构造器接口实现、资源接口实现和/或可扩展接口实现。

然后在框 802，创建一个代理对象定义 204，其直接地或者通过另一个代理对象定义间接地扩展 marker(标记) ProxyObject 接口。如果代理对象定义直接地扩展 marker ProxyObject 接口，它例如用一

个"implementation"（实现）属性规定相关联的代理对象实现 210。一个间接地扩展 markerProxyObject 接口的代理对象定义，也可以规定一个相关联的实现，取代与它的基类(base class)相关联的实现。代理对象定义也可以规定新的默认特性质，并且如果实现 210 是可扩展的，可以规定新的函数和回叫。代理对象定义可以由应用程序 220 的开发者、代理对象实现 210 或者另一个独立第三方构造。如前文所述的那样，如果相关联的实现 210 实现可扩展的借口 214，则代理对象定义 204 是可扩展的。下文将参照图 10a-10b 描述扩展例子。

在框 804，应用程序 220 的开发者将一个或多个代理对象说明 222 插入引用代理对象定义 204 的应用程序码 220 中。正如前文指出的那样，代理对象定义 204 可以是例如由外部实体 202 的软件抽象的开发者提供的基(base)代理对象定义 204，或者可以使代理对象定义 204 的一个定制版本。下文将参照图 9a 描述一例说明。

在框 806，应用程序 220 的开发者限定代理对象定义 204 的特性中适合的特性的值。在一个实施例中，该详细说明以源文件的一个注释段内的注解为形式。下文将参照图 9b 描述一例详细说明。

插入代理对象说明 222 后，并且对于任何适合的代理代理对象说明、特性值，在框 808，应用程序 220 可以用由代理对象定义 204 定义的、并由实现 210 直接用内置函数 211 或者间接通过可扩展接口 218 实现的函数，以程序设置的方式与外部实体 202 互动。

如前文指出的那样， 应用程序 220 的开发者也可以为由外部实体 202 的软件抽象的异步事件生成函数生成和发送的异步事件指定一个处理程序。下文将参照图 9c 描述一例详细说明。

定制代理对象特性

图 10a 表示一个通过限定一个新的接口说明 1002 和新的默认特性设置 1004 而扩展图 5 中所示的 Timer(定时器)接口例的简单代理对象定义 204。图 10a 的 StandardTimer 代理对象定义继承由图 5 中的代理对象定义所定义的所有函数和特性，但是把@Timer 的"timeoutln"属性的默认设置改变为 30 秒。因此，如果 30 秒是可接受的，使用 StandardTimer 的应用 220 将不必规定@Timer 的"timeoutln"属性。

所属技术领域的熟练人员当然将认识到，上述例子故意简单化，以方便解释和易于理解。实践中，本发明的代理对象定义可以更广泛

地定制默认特性设置。特别地，代理对象定义也可以定制与特性对象函数和回叫相关联的特性。此外，特性对象定义可以被连续地定制多次，就是说，一个被定制的对象定义，本身可以被进一步定制。

定制代理对象接口

当代理对象实现 210 实现可扩展接口 214 时，也可能通过增加新的函数说明 209 和回叫说明 208 定制相关联的代理对象定义 204 的接口。图 10b 表示一例通过说明一个名为 getEmployeeData 的新函数定制 com.bea.jws.Database 代理对象定义的、名为 EmployeeDB 的代理对象定义 1020。1022 行的接口说明 205，说明 EmployeeDB 接口扩展 com.bea.jws.Database 接口，后者又扩展把 EmployeeDB 接口标识为本发明的代理对象定义的 com.bea.jws.ProxyObject 接口(未予示出)。这样，EmployeeData 将继承在 Database(数据库)代理对象定义中说明的所有特性设置、函数和回叫，以及它扩展的所有代理对象定义。

1028 行是一个把函数 getEmployeeData 添加到从 Database(数据库)代理对象定义继承的现有函数列表的函数说明。这个函数可以被应用程序 240 在运行时调用，以与由代理对象 1020 描述的外部的 Employee（职员）数据库互动。然而要注意的是，代理对象定义或代理对象实现哪一个都不特别地限定 getEmployeeData 函数。关于在运行时究竟如何处理对由代理对象说明 204 说明的函数 209 的调用的细节，在下文中进一步说明。

1026 行是描述 getEmployeeData 函数所需的语法的特性设置，1024 行定义由 getEmployeeData 函数返回的 EmployeeRecord 数据结构。所有接口说明 205、特性设置 206、回叫说明 208、函数说明 209 和相关联的定义（例如 EmployeeRecord 数据结构），都被编译器 230 存储在元数据 252 中，并可通过代理上下文对象 258 的在运行时用于代理对象 254。这个元数据辅助代理对象 254 和代理对象实现 210 提供由代理对象定义 204 所说明的函数 208 和回叫 209 的实现。

使用代理对象

图 9a 表示一例如在应用程序码 220 中可能找到的代理对象说明 222。902 行说明一个名为 theTimer 的新代理对象，它实现图 5 中的 com.bea.jws.Timer 代理对象定义。

图 9b 表示一个几乎相同的一例代理对象说明，它带有被设定为 30 秒的@Timer 特性的 timeoutln 属性，904。在这个例子中，timeoutln 特性的值是以一个注解段中的 Javadoc 注释的形式规定的。应用程序码 220 调用关于这个对象的函数，以与相关联的外部定时器实体互动。此外，应用程序码 220 的开发者可以为外部实体 202 生成的异步事件指定处理程序。

图 9c 表示这样一例处理异步超时(timeout)事件通知的异步事件处理程序 906。在这个例子中，处理程序被编写成应用程序码 220 中一个特别命名的函数。函数名的构成方式是，把要被处理的异步事件名(即"ontimeout")附接到相关联的代理对象的名(即"theTimer")。如在下文将看到的那样，在运行时，代理对象 254 将把异步事件传递到应用程序 240 中的适当的时间处理码 246。

编译时

图 11a-11b 表示按照本发明一个实施例的、编译器 230 的有关方面的操作流程。首先如图 11a 所示，在框 1102，编译器 230 分析应用程序码 220 的源语句，以确定源语句中存在的语言元素。特别地，编译器 230 通过寻找为实现起源于代理对象标记接口 402 而说明的对象而确定是否在应用程序码 220 中包含本发明的任何代理对象说明，框 1104。

如果找不到本发明的代理对象说明，就像编译现有技术中的其它软件实体那样编译应用程序码 220，框 1106。这个编译的准确的性质与语言和编译器实现有关。

如果找到至少一个本发明的代理对象说明，编译器 230 就收集为描述本发明的代理对象说明所需的元数据，框 1108。

在一个实施例中，元数据收集操作包括从应用程序码 220 中识别和抽取特性设置 223，以及从所有相关联的代理对象定义 204 中—包括从在代理对象说明 222 中所标识的代理对象定义所源自的代理对象定义中—识别和抽取默认特性设置 204。此外，元数据收集还包括从所有相关联的代理对象定义 204 中识别和抽取被说明的接口 205 的名和签名，以及代理对象实现 210 的内置函数 211 和内置回叫 212 的名和签名。

在一个实施例中，特性设定是用应用程序码 220 和代理对象定义

204 的源文件的注解段中的 Javadoc 注释形式限定的。编译器 230 包括一个负责分析应用程序码 220 和代理对象定义 204 的源文件的注解段中的特性处理器（未予示出）。

在一个实施例中，编译器 230 的特性处理器也要进行与编译时实现类的协商，以确认特性设置和相关联的特性被代理对象实现 210 实现和允许。在一个实施例中，该协商是通过构造器接口 218 的函数实现的。

收集好为描述本发明的代理对象说明所需的源数据后，编译器 230 输出一个或多个含有所收集的元数据的元数据文件 252，框 1110，供由对应的代理对象 254 在运行期间使用。

然后，编译器 230 为与代理对象说明 222 相关联的(即被代理对象说明 222 引用的)每个代理对象定义 204 生成一个代理对象 254，以方便应用程序 240 与外部实体 202 之间的互动。这个过程在下文参照图 11b 作更详细地说明。

此外，编译器 230 为每个代理对象说明 222 生成代理初始化码 242，框 1114。在运行时，代理初始化码 242 的每个实例创建一个实现在相关联的代理对象说明 222 中所标识的接口的代理对象，把该代理对象分配到在相关联的代理对象说明 222 中所标识的代理对象变量，并向异步事件路由器 256 登记该代理对象，以从相关联的外部实体 202 接收所有异步事件。

然后，编译器 230 如现有技术中的那样编译其余的应用程序码 220，插入代理初始化码 242，以在相关联的代理调用码 244 和事件处理码 246 之前运行，框 1106。编译的方式与语言和编译器实现有关。

此外，属性处理在本领域的技术人员能力之内，不需要进一步描述。

图 12 更详细地表示由编译器 230 生成的代理对象 254。代理对象 254 包括由代理对象说明 204 所说明的、在图 12 中用黑圈表示的函数接口 1222-1224 和回叫接口 1226-1228。 此外，代理对象 254 还包括代理对象实现 210—包括在图 12 中用白圈表示的内置函数 211 和内置回叫 212。如果代理对象实现扩展的接口 216，则代理对象实现也包括用于处理对没有对应的内置函数 211 的函数接口 1224 的调用的 invoke 函数 616。

此外，代理对象 254 和代理对象实现 210 能通过描述相关联的代理对象定义 204（包括接口说明、特性设置、回叫说明和函数说明）和特性设置 223 的代理对象上下文 258，访问元数据 252。这个元数据可以在运行时被用来确定所需的对没有对应的内置函数 211 的函数接口 1224 的调用的语法。在一个实施例中，通过调用由运行引擎 250 提供的全局函数 getProxyContext( )，获得一个对代理对象上下文 258 的引用。在运行时，getProxyContext( )函数将按下文进一步说明的那样，返回与当前代理对象调用相关联的代理对象实例。

如前文所述的那样，在各种实施例中，代理对象上下文 258 包括各种用于方便“上下文”信息的访问的方法。在一个实施例中，这些方法包括一个用于取得元数据的 getMetaData( ) 方法，一个用于取得特定特性值的 getattribute( ) 方法。元数据例如可以包括与代理对象函数和回叫相关联的方法、自变量、域和/或注释。

在一个实施例中，代理对象上下文 258 也包括一个用于方便获得代理对象实例的独有 ID 的 getInstanceID( )，以及一个用于向应用程序 240 发送异步事件的 sendEvent( )。在一个实施例中，sendEvent( )通过把事件的名附接在代理对象说明中指定的代理对象变量，确定要调用的合适的事件处理程序 246。它从元数据 252 中抽取事件名和代理对象变量名。这些方法的实现，在所属技术领域的熟练人员的能力范围内，因此将不作进一步说明。在可替代的实施例中，可以用更多或更少的与代理对象上下文 258 相关联的方法实践本发明。

如前文所述的那样，在框 1112，编译器 230 生成代理对象 254，更具体来说，用从应用程序码 220、代理对象定义 204 和代理对象实现 210 中收集的对象，生成代理对象 254。如图 11b 中所示，它为在代理对象实现 210 中有一个对应内置函数 211 的代理对象定义 204 的每个函数说明 209，生成一个代理对象函数 1222，框 1122。代理对象函数 1222 的每个实现，简单地调用代理对象实现 210 中的内置函数 211 传入 provide（提供）参数并返回结果。

如果代理对象实现 210 实现可扩展接口 214，编译器 230 也为在代理对象实现 210 中没有一个对应内置函数 211 的代理对象定义 204 中的每个函数说明 209，生成代理对象函数 1224，框 1124。代理对象函数 1224 的每个实现，调用 “invoke” 函数 616 传送所提供的参数

的列表，并返回结果。

类似地，编译器 230 为在代理对象实现 210 中有一个对应的内置回叫 212 的代理对象定义 204 的每个回叫说明 208，生成代理对象回叫函数 1226，框 1126。回叫函数 1226 的每个实现，简单地调用对应的内置回叫 212 传送所提供的参数并返回任何结果。

此外，对于在代理对象实现 210 中没有对应的内置回叫 212 的代理对象定义 204 的每个回叫说明 208，编译器 230 确定是否在应用程序 240 中存在一个用于处理该回叫(call back)的适当的事件处理程序 246，框 1128。如果存在适当的事件处理程序 246，编译器 230 生成一个代理回叫函数 1228，该代理回叫函数调用该适当的事件处理程序 246 传入(passing in)所提供的参数并返回由事件处理程序生成的任何结果，框 1128。如果不存在适当的事件处理程序，编译器 230 生成一个错误，框 1128。在一个实施例中，编译器 230 通过在应用程序 240 中搜索一个具有一个特殊名的函数而标识(identifies)该适当的事件处理程序并确定它的存在，该特殊名是通过把相关联的事件的名附接到在代理对象说明中限定的相关联的代理对象变量而构成的。该适当的事件和代理对象变量的名，是从元数据 252 中抽取的。

运行时

图 13a 表示按照本发明一个实施例的运行引擎(runtime engine)250 的相关操作流程。当运行引擎 250 最初被实例化时，它初始化运行环境，特别是包括异步事件路由器 256 的实例的创建，框 1302。在一个实施例中，异步事件路由器 256 是一个服务器部件，它收听使用各种网络协议的消息并把它们转发到已经（例如根据消息地址或内容）为具有匹配的特征的事件登记的客户机(clients)。在一个实施例中，异步事件路由器 256 是一个 Java 小应用程序，它收听使用因特网协议—诸如 HTTP—的 XML 消息。在一个实施例中，事件路由器 256 收听使用排队协议—诸如 JMS—的消息。

运行引擎 250 的另外的非实质性的详细资料可以在共同待审定美国专利申请"ANNOTATION BASED DEVELOPMENT PLATFORM FOR ASYNCHRONOUS WEB SERVICES"（申请号 10/082,807，申请日 2002 年 2 月 22 日）中找到。该申请与本申请具有至少部分共同的申请人，特此完整引用其详细说明作为参考。

运行环境一初始化，运行引擎 250 就等待要求执行应用程序的请求，框 1304。在框 1306，运行引擎 250 被请求执行加载应用程序 240（或者，如果该应用程序因为更早的执行请求已经在以前被加载，则创建该应用程序的一个新实例）。在加载和/或创建应用程序 220 的实例后，运行引擎 250“执行”应用程序 220，或者更具体来说，把执行控制转移到应用程序 220。

图 13b 表示按照本发明一个实施例的一个典型的执行流程。如由编译器 230 所指明的那样，如果应用程序 240 包括代理初始化码 242 等等，代理初始化码 242 在代理调用码 244 和事件处理码 246 之前执行。

如前文所述的那样，代理初始化码 242 为每个代理对象说明 222 实例化一个代理对象，把该代理对象分配给在代理对象说明 222 中指定的相关联的变量，框 1312。然后，代理初始化码 242 登记在相关联的代理对象定义 204 中说明的并由代理对象 254 实现的所有回叫函数 208，以异步事件路由器 256 作为来自外部实体 202 的异步事件的处理程序，框 1314。

之后，代理引擎 250 继续执行应用程序 240。在执行过程中，如果应用程序 240 有与外部实体互动的需要，它用在代理对象说明 222 中说明的相关联的变量调用代理对象函数 1222-1224，框 1318。如由编译器 230 所指明的那样，代理对象函数 1222-1224 用一个函数调用 ID 生成一个与被调用函数相关联的代理上下文 258 的实例。在一个实施例中，为每个函数调用创建一个单独的线程，并把线程 ID 用作函数调用 ID。

函数 2222 进一步调用代理对象实现 210 的相关联的内置函数 211，框 1318。内置函数 211 因每个代理对象实现 210 而异，在很大程度上与相关联的外部实体 202 的性质有关。如果提供的话，代理对象函数 1224 调用代理对象实现 210 的"invoke"函数 616。

在一个实施例中，内置函数通过因特网或讯息传递协议(messaging protocols)向外部实体 202 发送消息，然后等待响应。在一个实施例中，如果接收到一个响应，内置函数 211 返回一个代表性的结果，后者又被代理对象函数 1222 返回给应用程序 240 内的代理调用码 244。在一个实施例中，内置函数包括为方便由外部实体 202 生成的回叫事

件的生成和传送而发送给外部实体 202 的消息中的一个回叫位置和代理对象实例。

内部函数 211 和可扩展接口 218 的“invoke”函数 616 二者都可以含有对当前代理上下文 258 的一个引用，用于通过调用由运行引擎 250 提供的全局 getProxyContext( ) 函数而访问元数据 252。getProxyContext( ) 函数根据与当前函数调用相关联的调用 ID 寻找并返回时当的内容对象。在一个实施例中，为每个函数 ID 创建一个单独的线程，并且当前调用 ID 与当前线程 ID 相同。

像内置函数 211 一样，invoke 函数 616 因每个代理对象实现 210 而异，在很大程度上与相关联的外部实体 202 的性质有关。在一个实施例中，invoke 函数 616 通过代理上下文对象 258 访问元数据 252，已确定所需的代理对象函数 1224 的语法，然后向外部实体 202 发送消息，可选地等待一个响应，并把一个代表性的结果返回给代理对象函数 1224，代理对象函数 1224 进而把该结果返回给应用程序 240 中的代理调用码 244。在一个实施例中，invoke 函数 616 包括为方便由外部实体 202 生成的回叫事件的生成和传送而发送给外部实体 202 的消息中的一个回叫位置和代理对象实例。

外部实体 240 一接收到来自应用程序的请求，就以独立于应用程序的方式处理该请求，并可选地记录由该请求提供的一个回叫地址和实例标识符。外部实体 202 可以生成可由异步事件处理程序 256 检测到的异步事件，并可以指定所记录的回叫地址和实例标识符，以方便事件的处理。在一个实施例中，外部实体 202 以消息的形式向异步事件路由器 256 提供异步通知。

在框 1320，当异步事件路由器 256 检测到来自外部实体 202 的事件时，它检查它的登记的处理程序的列表，并调用指定代理对象的指定回叫函数 1226-1228 传送该事件的一个代表作为一个参数集合。在一个实施例中，异步事件路由器 256 用被提供的回叫位置来标识哪个登记的处理程序和回叫函数应当处理该事件。在一个实施例中，异步事件路由器 256 用被提供的实例标识符来确定所标识的处理程序的哪个实例应当接收该回叫。

如由编译器 230 所指明的那样，在框 1324，代理对象回叫 1226 调用代理对象实现 210 的相关联的内置回叫 212 传送任何所提供的参

数，框 1332。内置回叫 212 因每个代理对象实现 210 而异，在很大程度上与相关联的外部实体 202 的性质有关。

在一个实施例中，内置回叫 212 可以调用应用程序 240 中的一个适当的事件处理程序 246 传送被提供的参数，并可选地等待一个响应，框 1322。

内置回叫 212 一旦收到一个对外部实体 202 的事件的响应，就把任何被返回的结果返回给代理对象回叫函数 1226，代理对象回叫函数 1226 把它返回给异步事件路由器 252，异步事件路由器 252 把结果提供给外部实体 202，框 1324。在一个实施例中，该结果以一个代表性消息的形式被返回到外部实体。

也如编译器所指明的那样，代理对象回叫 1228 没有对应的内置函数 212，因此，被直接转发到适当的事件处理程序 246，而任何对应的结果则被可选地通过代理回叫函数 1228 和异步事件路由器 256 返回到外部实体 202，框 1324。

在一个实施例中，适当的事件处理程序 246 被标识为在应用程序中定义的特别命名的函数。在一个实施例中，通过把对应于回叫说明 208 的代理回叫函数 1226-1228 的名附接到在应用程序马 220 中的代理对象说明 222 中说明的代理对象的名，确定这个命名规则(naming convention)。

管理 N 路(N-Way)关系

对于有些应用程序来说，有管理与一个外部实体的 n 路的互动的需要。就是说， 一个单一的应用程序 240 可能需要同时与外部实体 202 的多个实体互动。根据运行时数据，所需的实例的个数可能不同；因此也许在编写应用程序码 220 时不可能确定将需要多少个代理对象实例。例如，某应用程序实例可能需要分解一个购买订单的行项目(line items)，并为每个行项目与外部实体的一个不同实例进行并发的谈话(conversation)。

在各种实施例中，为了满足这种需要，应用程序开发者可以在代理对象说明 222 中规定一个代理对象工厂（factory）而不是规定单一的代理对象。对于这些实施例来说，编译器 230 自动地为每个代理对象 254 生成一个“工厂类”(factory class)。例如，为一个名为 MyService 的代理对象， 自动地生成一个名为 MyServiceFactory 的工厂类(未

予单独地示出)。图 14a 表示在一个实施例中的一例对应于图 5 中所示的"Timer"代理对象定义的代理对象工厂说明。

在这些实施例的一些或全部中，自动生成的代理对象工厂可以包括一个用来使应用程序 240 能控制对新的代理对象实例的创建的 create( )函数和一个用来使应用程序 240 能控制对以前创建的代理对象实例的破坏的 destroy( )函数。 这样，应用程序 240 在运行时就可以创建要多少有多少的代理对象的实例。图 14b 表示应用程序码 202 如何能用一个实施例中的 creates(创建)函数生成"Timer"代理对象的一个新实例并用生成的代理对象与相关联的外部实体互动。

每个自动生成的代理对象工厂都可以被软件应用程序用来与一个 n 路互动中的对应的外部实体互动，其方式犹如前文参照图 9a-9c 所述的单一互动的情形中的一样。代理对象工厂的行为犹如注释（即使用说明）就在由代理对象工厂创建的实例的前面一样。

为了方便正确的异步事件传送，应用程序码 220 的开发者用代理对象工厂的名而不是代理对象变量名来命名相关联的事件处理程序 224。此外，开发者还限定一个“proxy object instance”(代理对象实例)为每个事件处理程序 224 的一个预定的参数，例如第一个参数。代理对象 254 将为每个回叫事件提供适当的代理对象实例，这样，应用程序 240 就可以确定外部实体 202 的哪个实例生成了事件，并用被提供的代理对象实例与它互动。图 14c 表示一个实施例中的一个事件处理程序 224，这是为处理由在图 14a 中说明的名为"manyTimers" 的代理对象工厂生成的 Timer 代理对象中的异步事件而开发的。如图所示，在调用时，第一个字变量"t"将引用与生成该事件的外部实体的实例相关联的 Timer 代理对象的特定实例。

计算机系统举例

图 15 表示按照一个实施例的一例适合用于实施本发明的计算机系统。视各种元件的大小、功能或能力，示例计算机系统 1500 可以被用来在运行期间寄放外部实体 202 的软件抽象、和/或外部实体 202 的软件抽象的实现。示例计算机系统 1500 也可以被用作开发以程序设置的方式与外部实体 202 互动的应用程序 220—包括编译该程序的、或者在运行期间执行应用程序 240 的主机系统。

如图所示，计算机系统 1500 包括一个或多个处理器 1502 和系统

内存 1504。此外，计算机系统 1500 还包括海量存储装置 1506（诸如磁盘、硬驱、CDROM 等等）、输入/输出装置 1508（诸如键盘、光标控制器等等）和通信接口 1510（诸如网络接口卡、调制解调器等等）。这些元件通过代表一个或多个总线的系统总线 1512 互相连接。如果有多个总线，这多个总线由一个或多个桥接器(未予示出)桥接。

这些元件的每一个执行它在现有技术中已知的传统功能。

特别地，系统内存 1504 和海量存储器 1506 存储实现本发明的的各个方面的程序指令的工作副本和永久副本，即外部实体 202 的抽象、软件抽象的实现 210、应用程序 220、编译器 230、和/或运行引擎 250。程序指令的永久副本，可以通过例如发布介质(未予示出)或通过通信接口 1510（来自(未予示出的)发布服务器），在工厂或者在现场被装入海量存储器 1506 中。

这些元件 1502-1512 的构造是已知的，因此将不作进一步说明。

结语

因此，从以上说明中可见，已经说明了一种用于简化用于与一个软件应用程序内的外部实体的互动的软件抽象的开发、定制和使用的新颖方法和设备。本发明的优点是辅助软件开发者开发与各种其它外部实体互动的软件，而不必需要软件开发者学习大量的范例或获得大量的新技术。

尽管以上述的实施例说明了本发明， 所属技术领域的熟练人员将认识到，本发明并不限于所描述的实施例。在本发明的精神和范围内实践本发明可以有修改或改变。因此应当认为详细说明对本发明是示例性的而非限制性的。

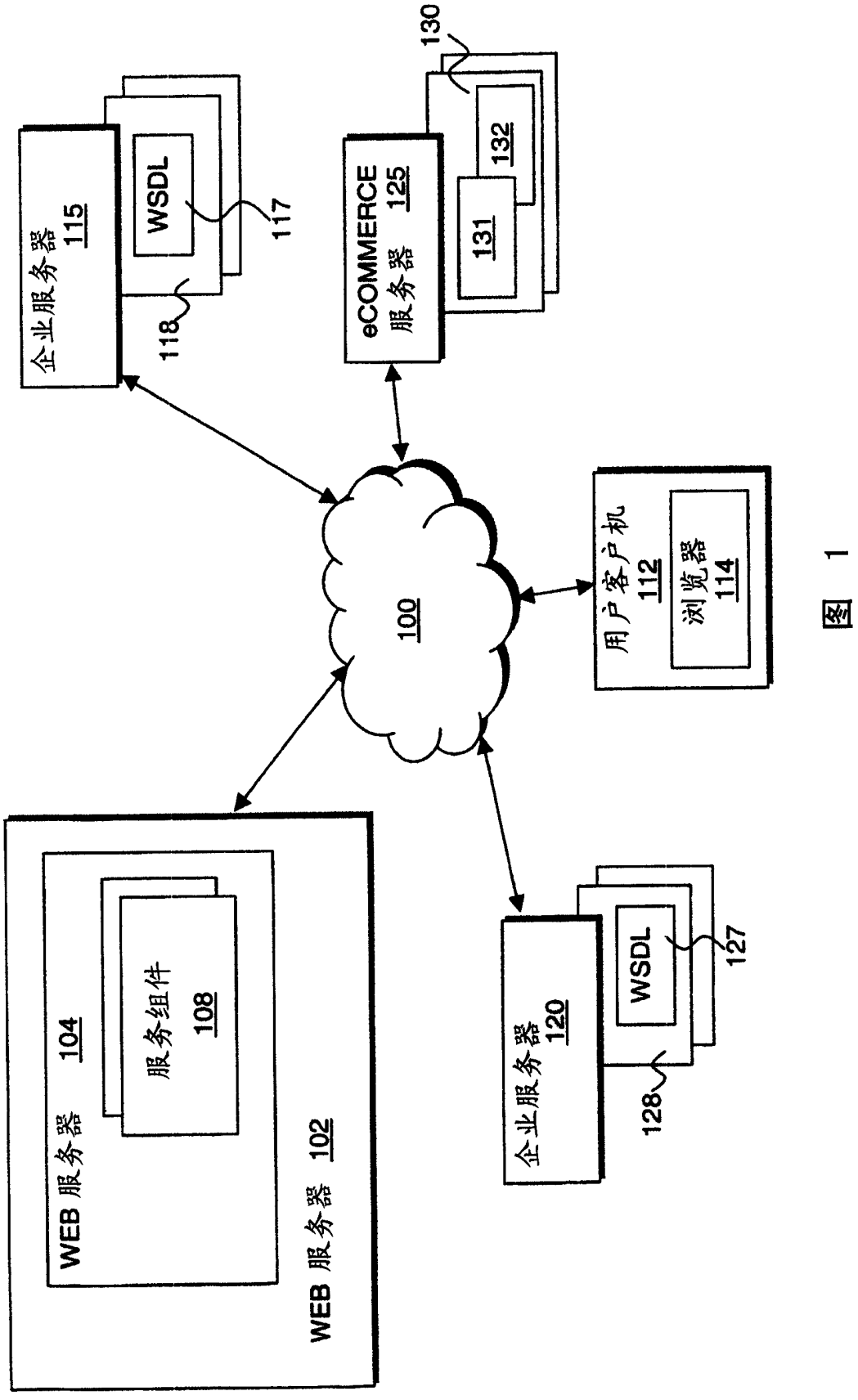

WEB 服务器 102
WEB 服务器 104
服务组件 108
企业服务器 115
WSDL
117
118
eCOMMERCE 服务器 125
130
131
132
100
用户客户机 112
浏览器 114
企业服务器 120
WSDL
127
128

图 1

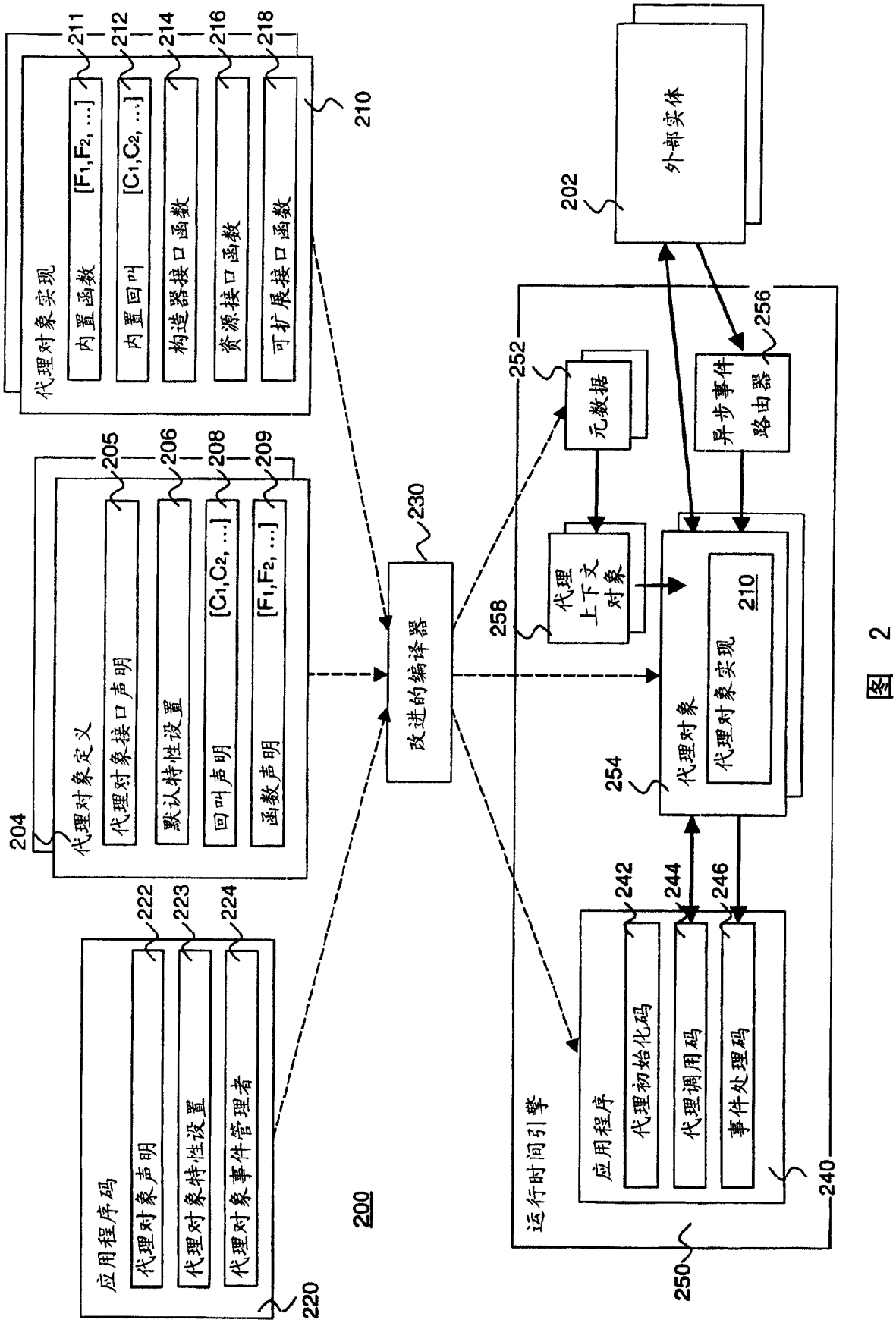

代理对象实现
内置函数 [F1,F2, ...]
内置回叫 [C1,C2, ...]
构造器接口函数
资源接口函数
可扩展接口函数
211
212
214
216
218
210
202
外部实体
代理对象定义
代理对象接口声明
默认特性设置
回叫声明 [C1,C2, ...]
函数声明 [F1,F2, ...]
205
206
208
209
204
改进的编译器
230
252
元数据
256
异步事件路由器
258
代理上下文对象
254
代理对象
代理对象实现
210
应用程序码
代理对象声明
代理对象特性设置
代理对象事件管理者
222
223
224
220
200
运行时间引擎
应用程序
代理初始化码
代理调用码
事件处理码
242
244
246
240
250

图 2

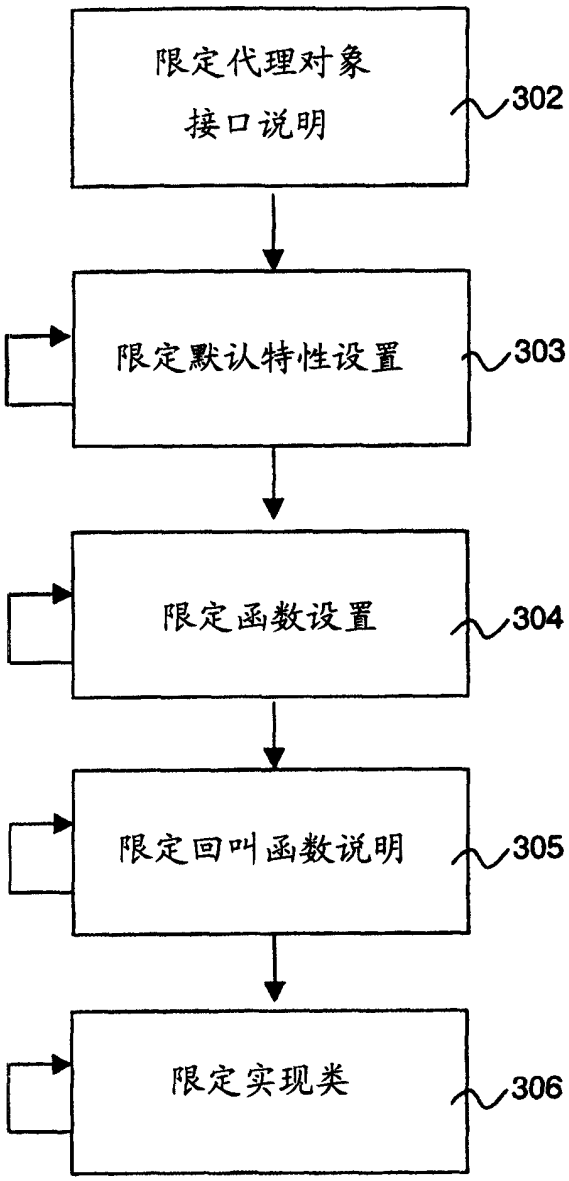

限定代理对象
接口说明
302
限定默认特性设置
303
限定函数设置
304
限定回叫函数说明
305
限定实现类
306

图 3

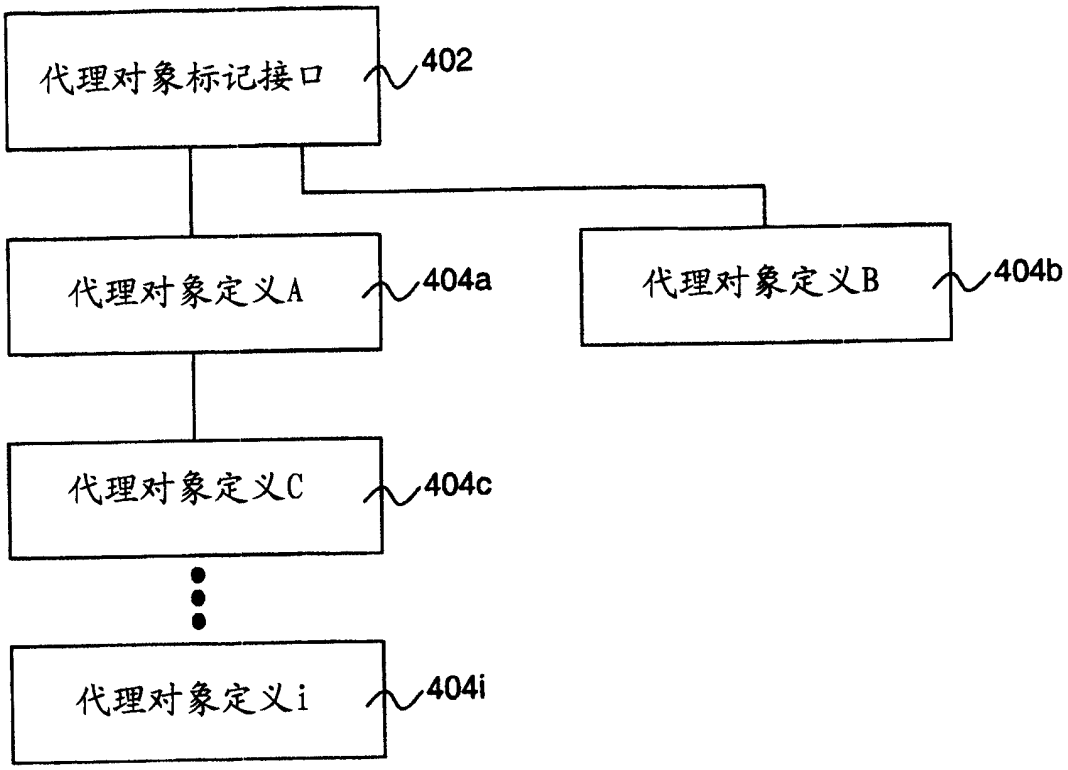

代理对象标记接口
402
代理对象定义A
404a
代理对象定义B
404b
代理对象定义C
404c
代理对象定义i
404i

图 4

图 5

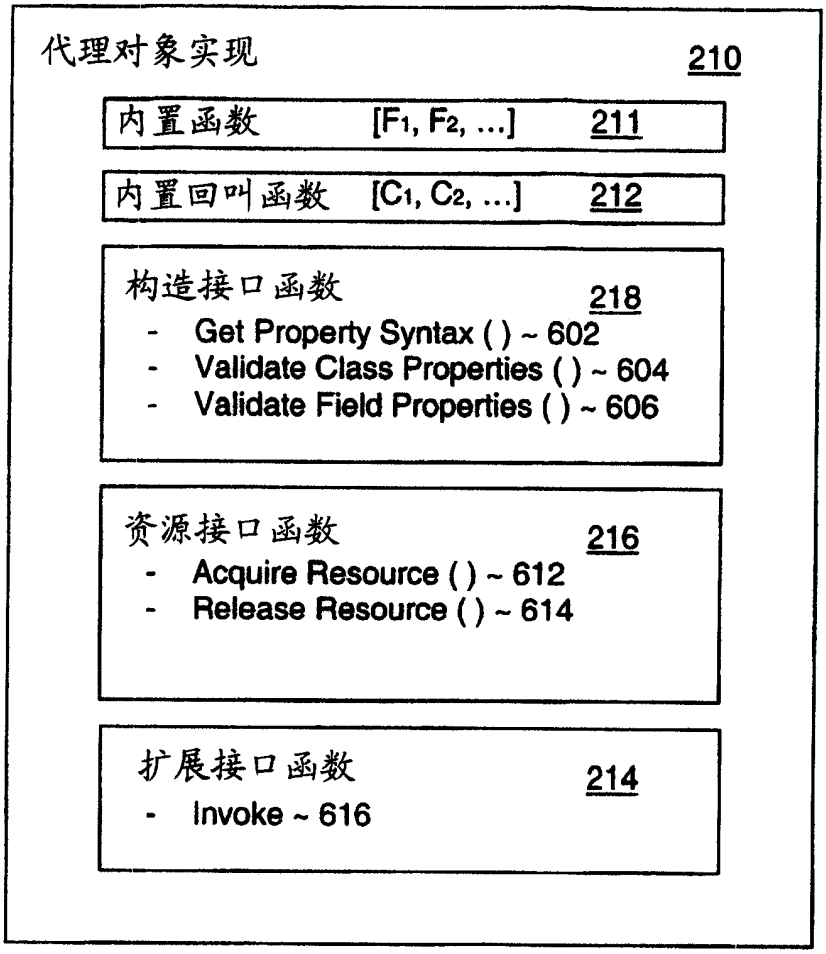

代理对象实现
210
内置函数 [F1, F2, ...] 211
内置回叫函数 [C1, C2, ...] 212
构造接口函数
218
- Get Property Syntax ( ) ~ 602
- Validate Class Properties ( ) ~ 604
- Validate Field Properties ( ) ~ 606
资源接口函数
216
- Acquire Resource ( ) ~ 612
- Release Resource ( ) ~ 614
扩展接口函数
214
- Invoke ~ 616

图 6

name="sql" onMethod="yes"> ~ 702a
name="statement" required="yes"/> ~ 704a
name="maxCount"> ~ 704b
integer/></type> ~ 706
infinity</default> ~ 708
name="returnType"/> ~704c
pool" onDecl="optional" onDef="optional" onFunc="optional"> ~ 702b
name="name" required="yes"/>

图 7

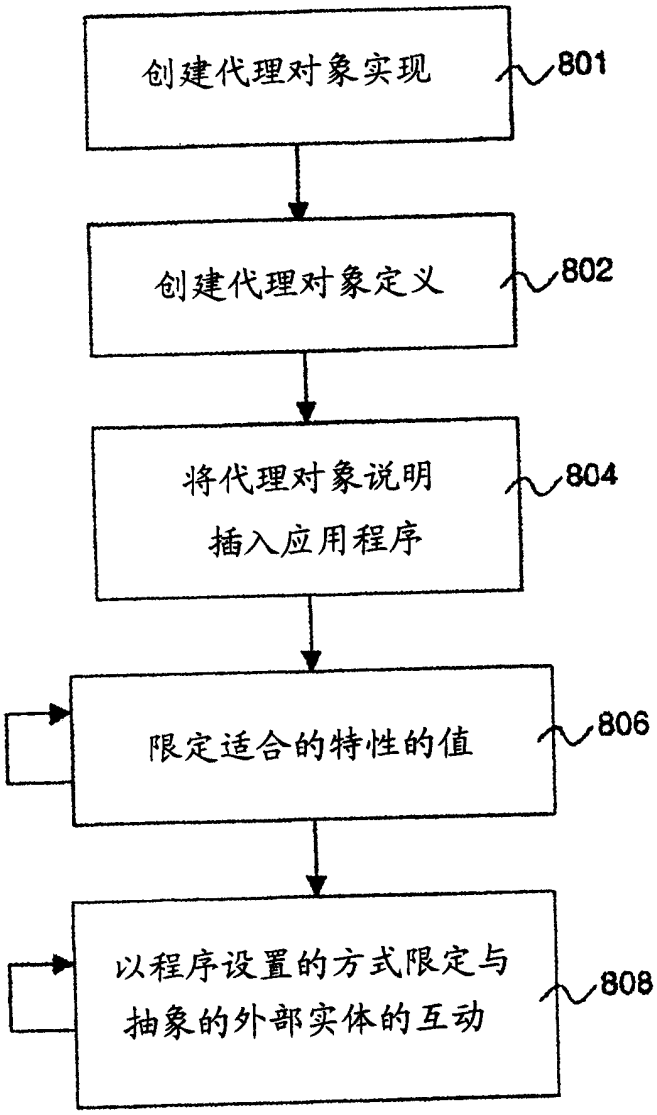

创建代理对象实现
801
创建代理对象定义
802
将代理对象说明
插入应用程序
804
限定适合的特性的值
806
以程序设置的方式限定与
抽象的外部实体的互动
808

图 8

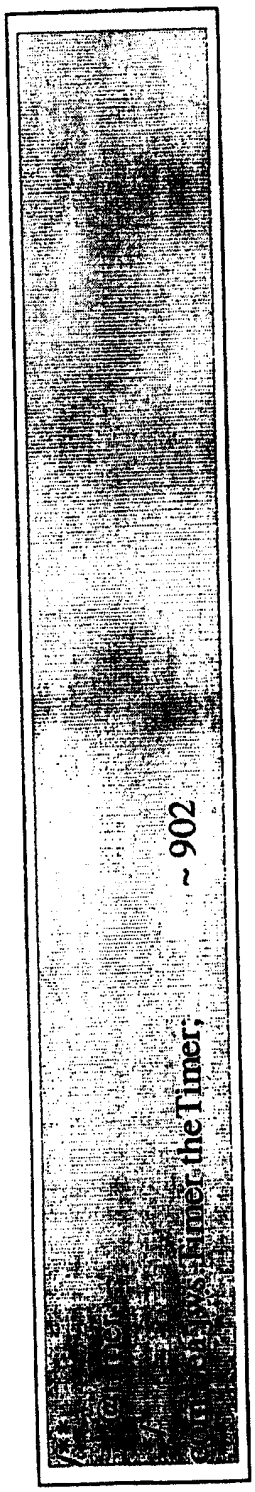


图 9a

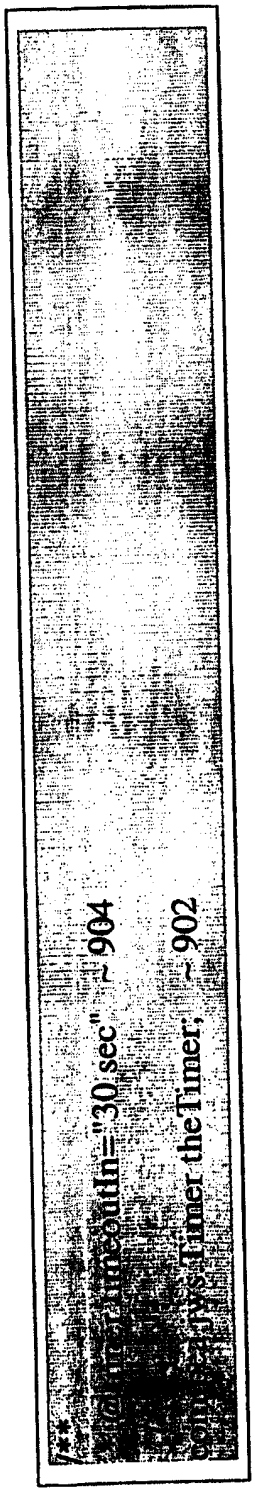


图 9b

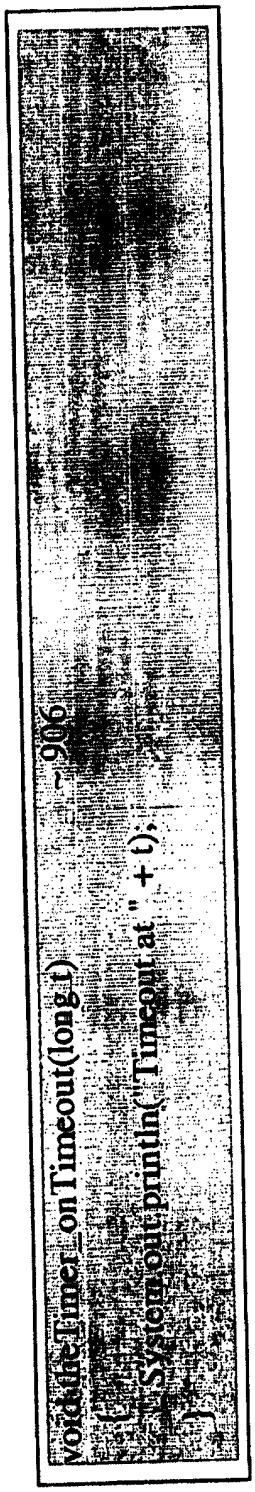


图 9c

StandardTimer.ctrl
timeoutIn="30 sec"
~1004
~1002
StandardTimer.extends com.bea.jws.Timer
图 10a
EmployeeDB.ctrl
1020
* EmployeeDB.ctrl
~1022
public interface EmployeeDB extends com.bea.jws.Database
~1024
public class EmployeeRecord
String name;
long salary;
* Port level resource attributes, such as the SQL associated
* with the port, parameter maps, etc.
@sql:statement="SELECT NAME, SALARY FROM EMPS WHERE ID = {id}"
~1026
EmployeeRecord getEmployeeData(int id) throws SQLException;
~1028
图 10b

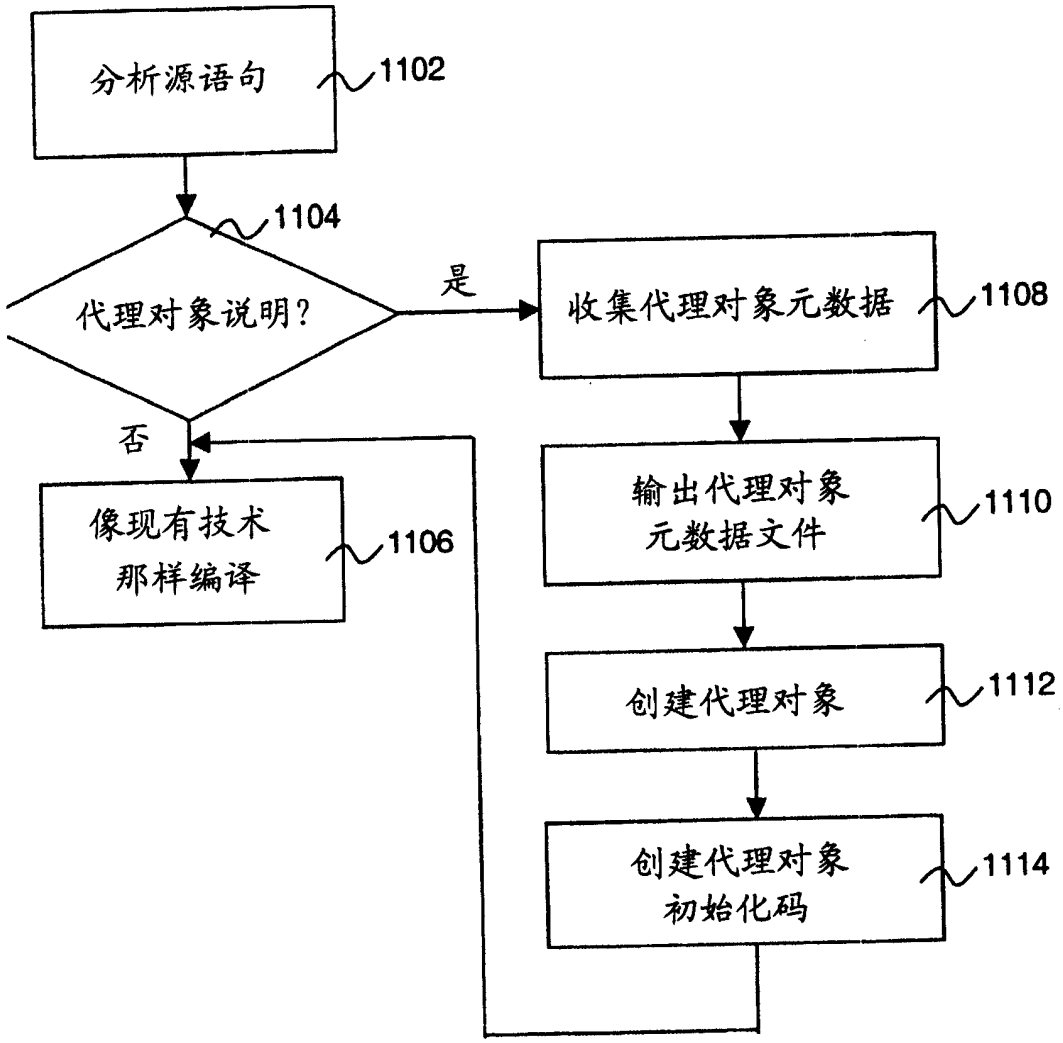

分析源语句
1102
1104
代理对象说明?
是
否
收集代理对象元数据
1108
输出代理对象
元数据文件
1110
像现有技术
那样编译
1106
创建代理对象
1112
创建代理对象
初始化码
1114

图 11a

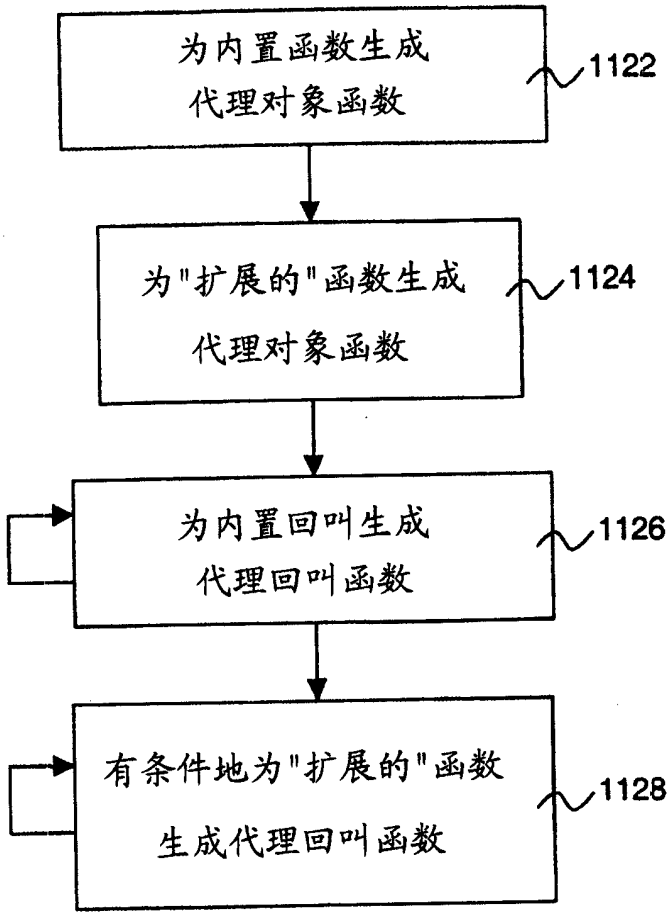

为内置函数生成
代理对象函数
1122
为"扩展的"函数生成
代理对象函数
1124
为内置回叫生成
代理回叫函数
1126
有条件地为"扩展的"函数
生成代理回叫函数
1128

图 11b

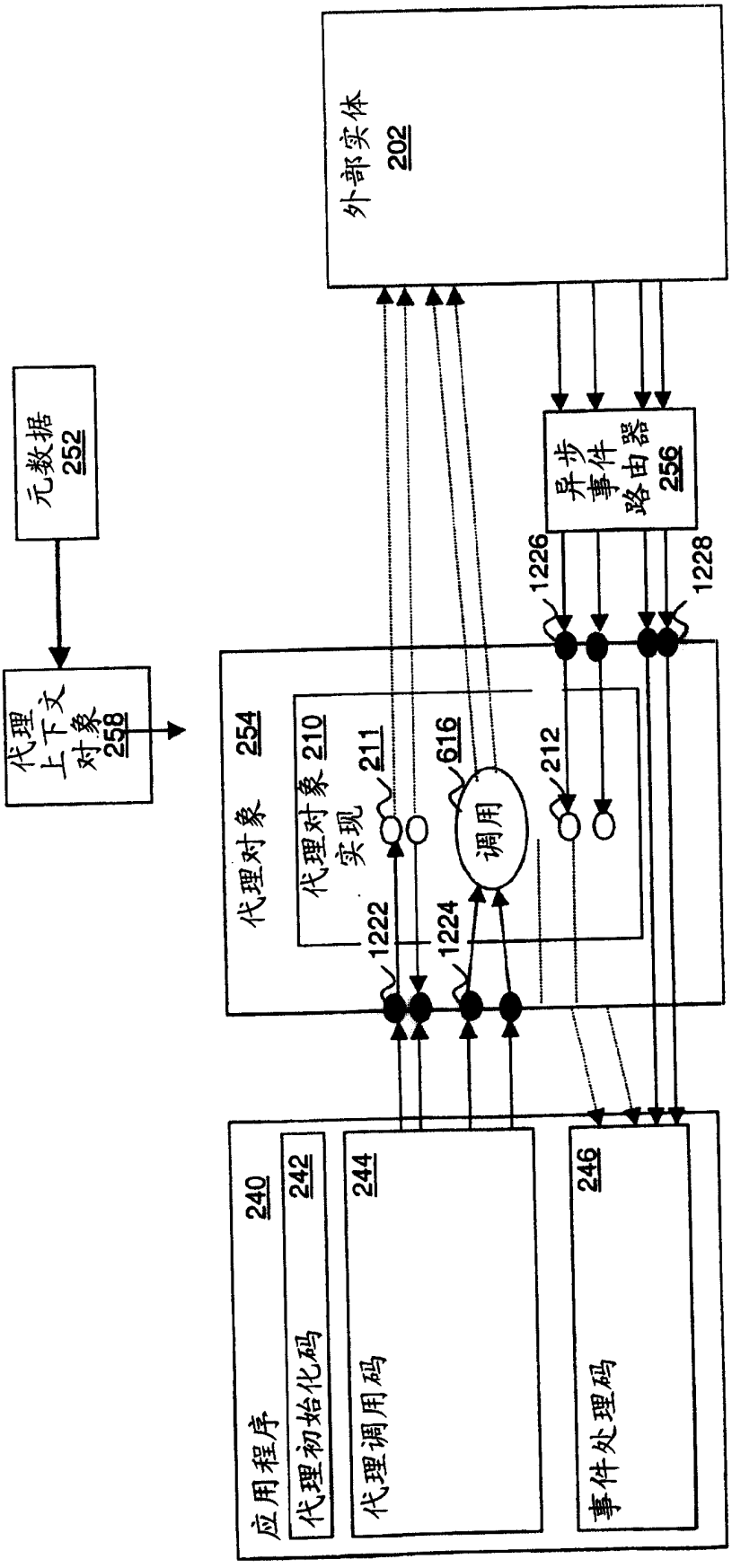

外部实体
202
元数据
252
代理
上下文
对象
258
异步
事件
路由器
256
1226
1228
代理对象 254
代理对象
实现 210
211
616
212
调用
1222
1224
应用程序 240
代理初始化码 242
代理调用码 244
事件处理码 246

图 12

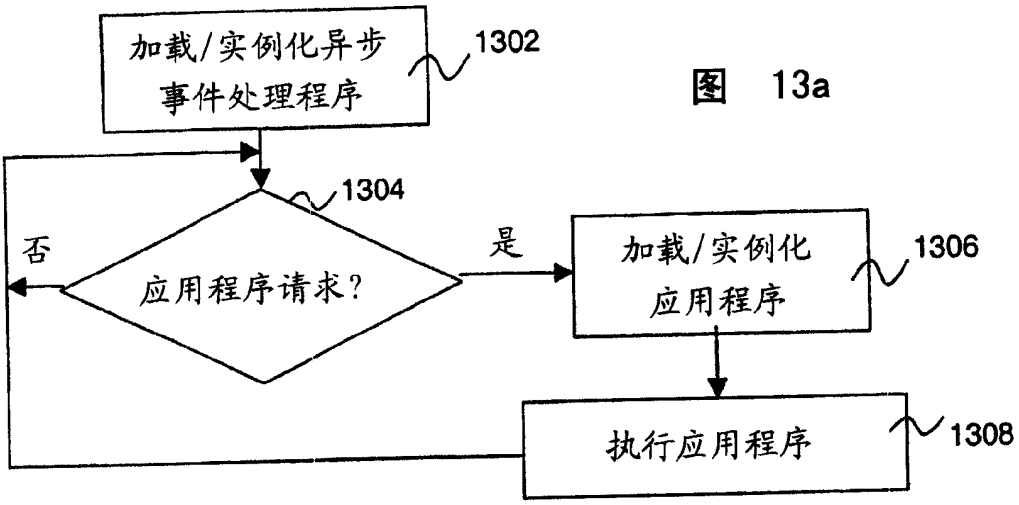

加载/实例化异步事件处理程序
1302
1304
否
应用程序请求?
是
加载/实例化应用程序
1306
执行应用程序
1308

图 13a

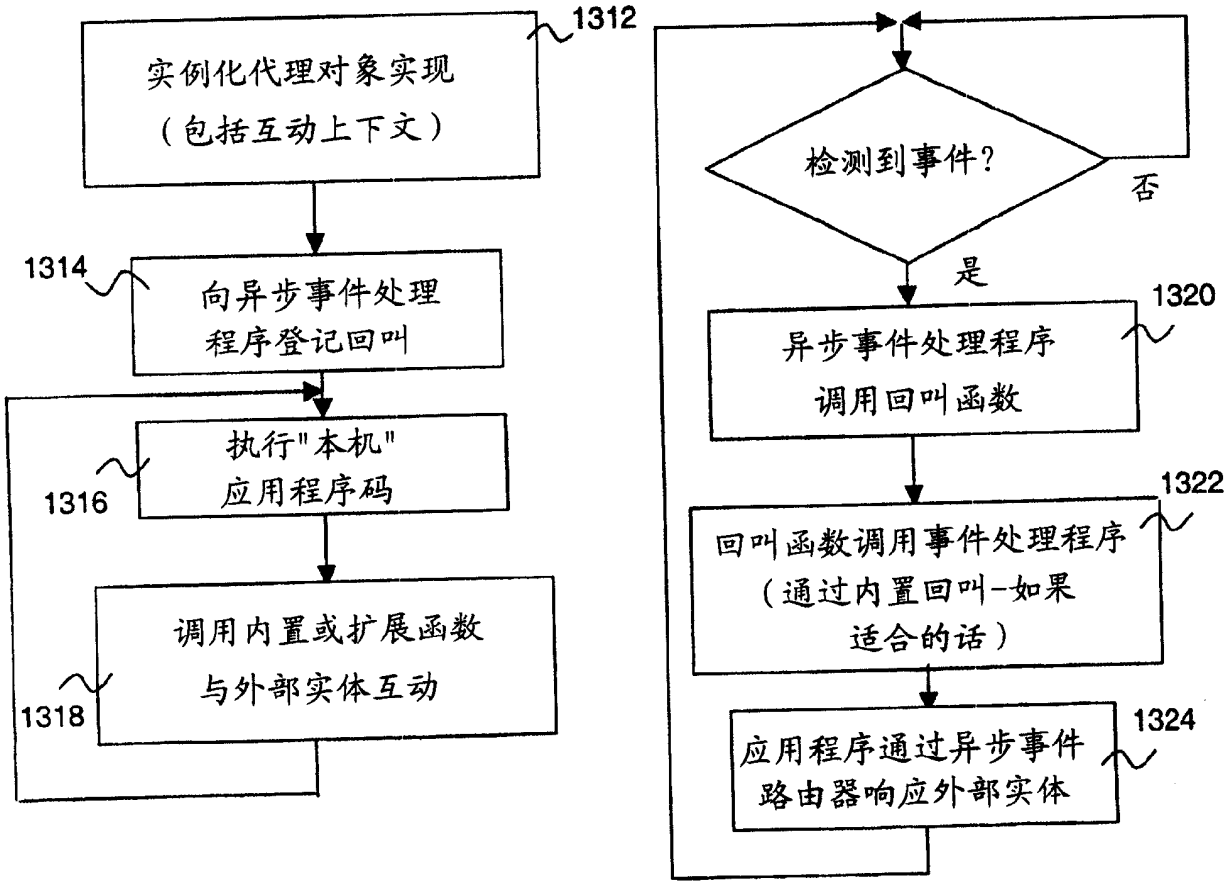

实例化代理对象实现（包括互动上下文）
1312
1314
向异步事件处理程序登记回叫
执行"本机"应用程序码
1316
调用内置或扩展函数与外部实体互动
1318
检测到事件?
否
是
异步事件处理程序调用回叫函数
1320
回叫函数调用事件处理程序（通过内置回叫-如果适合的话）
1322
应用程序通过异步事件路由器响应外部实体
1324

图 13b

```
/**
 * @jws:control
 * @jws:timer timeout="1h"
 */
TimerFactory manyTimers;
```

图 14a

```
Timer t = manyTimers.create();
// then you can just use the control, store it, or whatever.
t.start();

// For example, let's associate a name with the timer...
timerMap.put(t, "First Timer");
```

图 14b

```
void manyTimers_onTimeout(Timer t, long elapsed)
{
    // let's retrieve the remembered name associated with the timer
    String timerName = (String)timerMap.get(t);

    // and print it out
    System.out.println("Event received from " + timerName);
```

图 14c

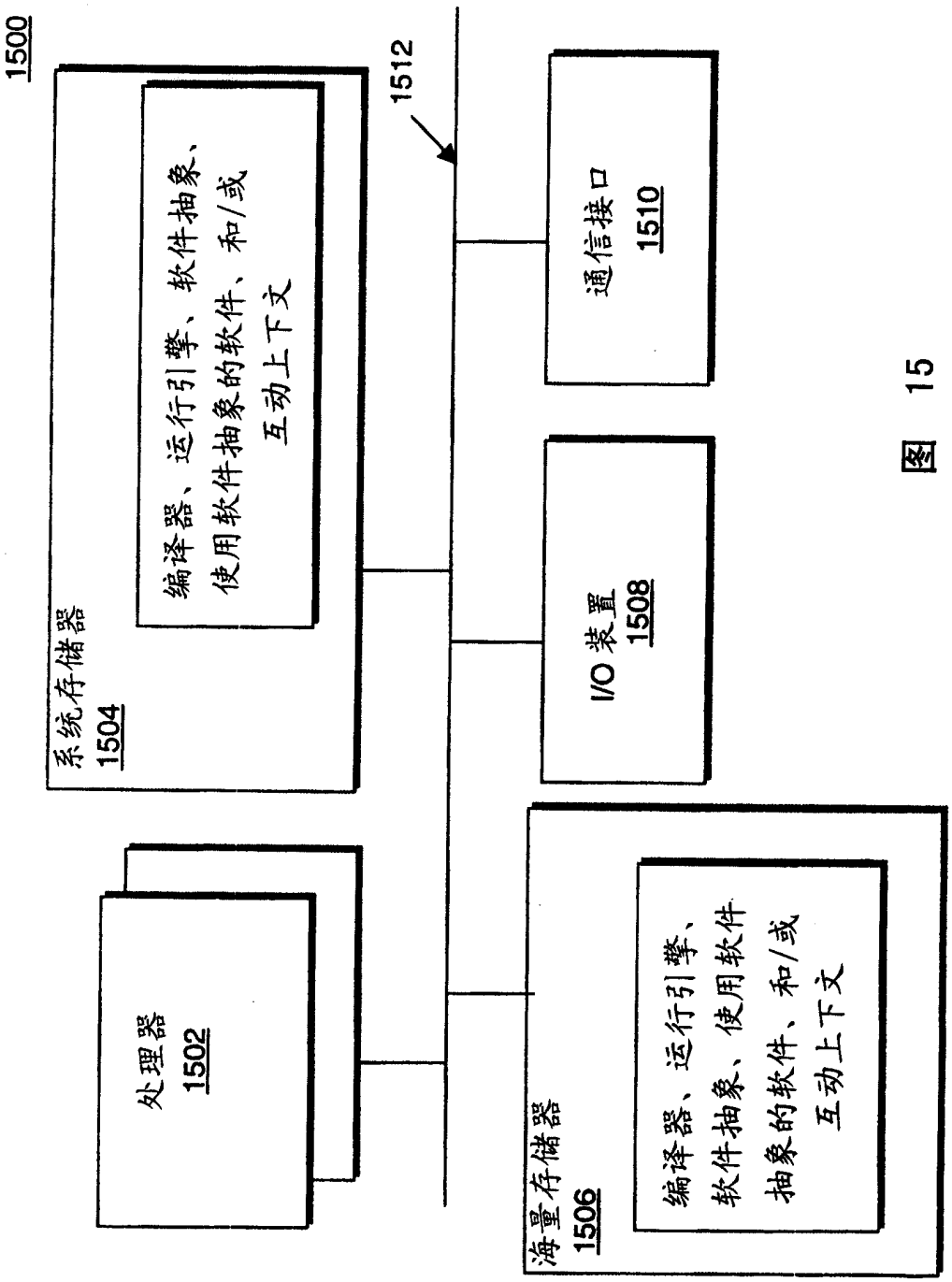

1500
系统存储器
1504
编译器、运行引擎、软件抽象、
使用软件抽象的软件、和/或
互动上下文
1512
通信接口
1510
I/O 装置
1508
处理器
1502
海量存储器
1506
编译器、运行引擎、
软件抽象、使用软件
抽象的软件、和/或
互动上下文

图 15