[19] Patents Registry
The Hong Kong Special Administrative Region
香港特別行政區
專利註冊處

[11] 1175552 B
CN 102841780 B

[12] STANDARD PATENT SPECIFICATION
標準專利說明書

[21] Application No. 申請編號
13102774.3

[22] Date of filing 提交日期
06.03.2013

[51] Int.Cl.[8] G06F

[54] METHOD AND DEVICE FOR CREATING AND INVOKING COMMON COMPONENTS 一種創建並調用通用組件的方法及設備

[43] Date of publication of application 申請發表日期
05.07.2013

[45] Publication of the grant of the patent 批予專利的發表日期
08.07.2016

CN Application No. & Date 中國專利申請編號及日期
CN 201110170947.3 23.06.2011

CN Publication No. & Date 中國專利申請發表編號及日期
CN 102841780 26.12.2012

Date of Grant in Designated Patent Office 指定專利當局批予專利日期
14.10.2015

[73] Proprietor 專利所有人
Alibaba Group Holding Limited
Fourth Floor, One Capital Place
P. O. Box 847, George Town
Grand Cayman
CAYMAN ISLANDS
阿里巴巴集團控股有限公司
開曼群島
大開曼島
資本大廈一座四層 847 號郵箱

[72] Inventor 發明人
TANG, Hongbing 唐紅兵

[74] Agent and / or address for service 代理人及/或送達地址
香港集佳知識產權代理有限公司
香港九龍旺角洗衣街 39-55 號
金雞廣場 12 層 1201 號

(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102841780 B
(45)授权公告日 2015.10.14

(21)申请号 201110170947.3

(22)申请日 2011.06.23

(73)专利权人 阿里巴巴集团控股有限公司
地址 英属开曼群岛大开曼岛资本大厦一座四层847号邮箱

(72)发明人 唐红兵

(74)专利代理机构 北京同达信恒知识产权代理有限公司 11291
代理人 黄志华

(51)Int.Cl.
G06F 9/44(2006.01)

(56)对比文件
CN 101359999 A,2009.02.04,
CN 101739254 A,2010.06.16,
US 7624397 B1,2009.11.24,

审查员 李敏

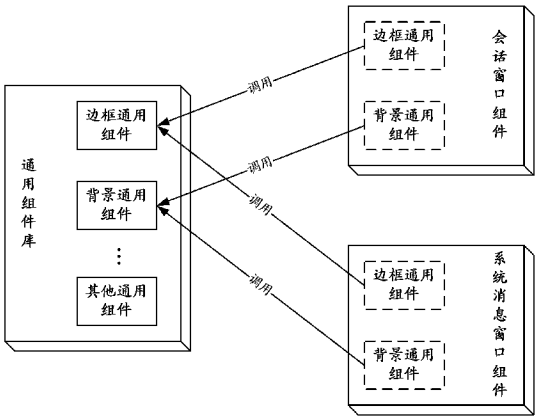
权利要求书2页 说明书10页 附图6页

(54)发明名称

一种创建并调用通用组件的方法及设备

(57)摘要

本申请公开了一种创建并调用通用组件的方法及设备，主要内容包括：在软件初始化过程中完成通用组件的统一创建以及统一存储在通用组件库的过程，在其他普通组件需要调用通用组件时，无需分别创建通用组件，而只需要直接调用已创建的通用组件即可，由于在本申请实施例的方案中，通用组件只需创建一次，减少了通用组件的创建数量，因此，可有效减少通用组件创建过程中占用的系统资源，且提高其他普通组件的运行速度。



CN 102841780 B

1. 一种创建通用组件的方法，其特征在于，应用于具有封装后的组件的软件系统中，包括：

在软件初始化时，生成通用组件库；

根据配置文件中记录的通用组件的属性信息，创建通用组件；

将创建的通用组件以及该通用组件的标识存储至所述通用组件库中；

根据配置文件中记录的通用组件的属性信息，创建通用组件，具体包括：

按照设定的实例创建方式，创建通用组件的实例，并根据配置文件中记录的通用组件的属性信息，为创建的实例设置属性。

2. 如权利要求 1 所述的方法，其特征在于，所述属性信息包括以下信息中的一种或多种组合：

通用组件的类型、通用组件的名称和运行通用组件时所需数据的存储地址。

3. 如权利要求 1 所述的方法，其特征在于，将创建的通用组件以及通用组件的标识存储至所述通用组件库中，具体包括：

确定指向创建的实例在通用组件库中存储位置的内容指针，以及为创建的实例分配标识，并建立实例的标识与内容指针之间的对应关系；

将实例的标识与内容指针之间的对应关系存储至所述通用组件库中。

4. 一种调用通用组件的方法，其特征在于，包括：

确定待调用的通用组件的标识；

根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件，所述通用组件库是利用权利要求 1 中的方法创建的；

调用查找出的通用组件。

5. 如权利要求 4 所述的方法，其特征在于，确定待调用的通用组件的标识之后，且从通用组件库中查找所述标识所表示的通用组件之前，所述方法还包括：

判断所述标识的内容是否为空；

在所述标识的内容不为空时，进一步确定所述标识内容的字符串长度在设定范围值内。

6. 如权利要求 4 所述的方法，其特征在于，根据确定的所述标识，从通用组件库中查找所述标识所表示的通用组件，具体包括：

根据通用组件的实例的标识与内容指针之间的对应关系，确定待调用的通用组件的实例的标识对应的内容指针；

将确定的内容指针所指向的实例作为查找出的通用组件的实例。

7. 如权利要求 6 所述的方法，其特征在于，调用查找出的通用组件，具体包括：

调用通用组件的实例入口，并运行所述实例。

8. 一种创建通用组件的设备，其特征在于，应用于具有封装后的组件的软件系统中，包括：

库创建模块，用于在软件初始化时，生成通用组件库；

组件创建模块，用于根据配置文件中记录的通用组件的属性信息，创建通用组件；

存储模块，用于将创建的通用组件以及该通用组件的标识存储至所述通用组件库中；

所述组件创建模块包括：实例创建子模块和属性设置子模块，其中：实例创建子模块

用于按照设定的实例创建方式，创建通用组件的实例；属性设置子模块用于根据配置文件中记录的通用组件的属性信息，为创建的实例设置属性。

9. 一种调用通用组件的设备，其特征在于，包括：

标识确定模块，用于确定待调用的通用组件的标识；

查找模块，用于根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件，其中，存储的通用组件库是利用权利要求 8 所述的创建通用组件的设备创建的；

调用模块，用于调用查找出的通用组件。

一种创建并调用通用组件的方法及设备

技术领域

[0001] 本申请涉及计算机技术领域，尤其涉及一种创建通用组件并对创建的通用组件进行调用的方法及设备。

背景技术

[0002] 随着计算机软件技术的不断发展，软件越来越朝着大而复杂的方向发展，为了对这种大而复杂的软件进行有效运用，可以将这些软件划分成一系列可先行实现、易于开发、理解和调整的组件。划分后的每个组件功能确定、单独设计、分别编码，最后将多个组件组装在一起，完成软件的运行。

[0003] 在软件开发过程中，为了提高软件代码的可复用性，提高开发效率，可将常用的界面元素封装成一组相对独立的组件，由这些组件构成图形用户界面 (Graphical User Interface，GUI) 库或 GUI 平台。GUI 平台中可存在几十种甚至上百种可用的组件，每种组件能够实现界面上的一种可见或不可见的功能。

[0004] 在 GUI 平台内的组件可以以树形结构存在，任意一个组件都可以将其他若干个组件作为自身的子组件，构成组件之间的嵌套关系。如图 1 所示，为 GUI 平台内组件之间的嵌套关系示意图，从图 1 中可以看出，组件 1 内嵌套了组件 2 和组件 3，而组件 2 内又进一步嵌套了组件 4。作为子组件的组件 4 可以被组件 2 所调用，同样的，作为子组件的组件 2 和组件 3 又可以被组件 1 所调用。

[0005] 而实际上，作为被嵌套的某些组件的功能具有通用性，即这些组件经常被嵌套至多种组件中，并且嵌套在每种组件中时，这些组件的属性都相同，则这些具有通用功能的组件称之为通用组件 (Common Component)。

[0006] 例如，即时通信软件中的边框组件可以被各种窗口类组件所嵌套，用于绘制窗口类组件的边框，背景组件也可以被各种窗口类组件所嵌套，用于绘制窗口类组件的背景，因此，边框组件和背景组件可称之为通用组件。

[0007] 假设窗口类组件有会话窗口组件和系统消息窗口组件，按照目前的组件嵌套模式，如图 2 所示，会话窗口组件要通过创建边框组件和背景组件来实现对边框组件和背景组件的嵌套，进而获得边框组件和背景组件的功能；系统消息窗口组件同样也要通过创建边框组件和背景组件来实现对边框组件和背景组件的嵌套，进而获得边框组件和背景组件的功能。

[0008] 从图 2 所示的嵌套结构可以看出，对于通用组件而言，当通用组件被多个其他组件嵌套时，即使这些通用组件被嵌套在其他组件中的属性相同，其他组件也必须分别嵌套通用组件，并只能对自身嵌套的通用组件进行调用，来达到运行通用组件功能的目的。在这种嵌套方式下，系统需要为每个其他组件创建通用组件分配用于创建的系统资源，即使相同的通用组件被多个其他组件创建时，系统也要为每次创建过程分配系统资源，导致内存资源的大量浪费；同时，每个组件在运行通用组件之前，都要创建该通用组件，由于每次的创建过程会占用一定的时长，因此，组件的正常运行速度也受到极大的影响。

发明内容

[0009] 本申请的目的在于：提供一种创建并调用通用组件的方法及设备，用以解决现有技术中存在的由于嵌套创建通用组件，导致内存资源的大量浪费和组件运行速度低的问题。

[0010] 一种创建通用组件的方法，应用于具有封装后的组件的软件系统中，包括：

[0011] 在软件初始化时，生成通用组件库；

[0012] 根据配置文件中记录的通用组件的属性信息，创建通用组件；

[0013] 将创建的通用组件以及该通用组件的标识存储至所述通用组件库中。

[0014] 一种调用通用组件的方法，包括：

[0015] 确定待调用的通用组件的标识；

[0016] 根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件；

[0017] 调用查找出的通用组件。

[0018] 一种创建通用组件的设备，应用于具有封装后的组件的软件系统中，包括：

[0019] 库创建模块，用于在软件初始化时，生成通用组件库；

[0020] 组件创建模块，用于根据配置文件中记录的通用组件的属性信息，创建通用组件；

[0021] 存储模块，用于将创建的通用组件以及该通用组件的标识存储至所述通用组件库中。

[0022] 一种调用通用组件的设备，包括：

[0023] 标识确定模块，用于确定待调用的通用组件的标识；

[0024] 查找模块，用于根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件；

[0025] 调用模块，用于调用查找出的通用组件。

[0026] 本申请有益效果如下：

[0027] 本申请实施例提供的方案，在软件初始化过程中，完成通用组件的统一创建，并统一存储在通用组件库的过程，在其他普通组件需要调用通用组件时，无需分别创建通用组件，而只需要直接调用已创建的通用组件即可。在本申请实施例的方案中，通用组件只需创建一次，这减少了通用组件的创建数量，可有效减少通用组件创建过程中占用的系统资源，且提高其他普通组件的运行速度。

附图说明

[0028] 图 1 为背景技术中组件之间的嵌套关系示意图；

[0029] 图 2 为背景技术中组件之间的嵌套关系示意图；

[0030] 图 3 为本申请实施例一中创建通用组件的方法步骤示意图；

[0031] 图 4 为本申请实施例二中调用通用组件的方法步骤示意图；

[0032] 图 5 为本申请实施例三中调用通用组件的方法步骤示意图；

[0033] 图 6 为本申请实施例三中普通组件与通用组件之间的调用关系示意图；

[0034] 图 7 为本申请实施例四中创建通用组件的设备结构示意图；

[0035] 图 8 为本申请实施例五中调用通用组件的设备结构示意图。

具体实施方式

[0036] 为实现本申请目的，本申请实施例提出一种新的针对通用组件的创建以及调用的方案，在软件的初始化阶段，创建通用组件并将创建成功的通用组件以及其标识存储在通用组件库中，当普通组件需要使用通用组件的功能时，无需再分别嵌套该通用组件，而是直接调用通用组件库中的通用组件即可。通过本申请实施例的方案，软件程序中的各普通组件无需分别创建通用组件，而只要需要调用通用组件库中的通用组件就能够获得通用组件的功能，由于只需要在软件的初始化阶段创建一次通用组件，普通组件就可以在任何时候自由地调用通用组件，减少了通用组件的创建次数，可有效减少通用组件创建时对系统内存资源的占用，同时，对于普通组件而言，不需要各自创建通用组件，而只需要调用通用组件，极大地提高了组件的运行速度，进而提高了软件的运行速度。

[0037] 下面结合说明书附图对本申请实施例的方案进行详细描述。

[0038] 需要说明的是，为了与创建以及被调用的通用组件相区别，将本申请各实施例中涉及的软件程序中需要调用通用组件的其他组件称之为普通组件。

[0039] 实施例一

[0040] 如图 3 所示，为本申请实施例一中创建通用组件的方法步骤示意图，所述方法包括以下步骤：

[0041] 步骤 101：在软件开始初始化时，生成通用组件库。

[0042] 本申请实施例的方案是，在软件初始化时开始执行创建通用组件的操作，并在初始化结束之前完成通用组件的创建过程。这样做的目的是：在软件正常运行之前已执行完毕创建通用组件的操作，可使在软件正常运行时，软件中的普通组件根据需求直接调用已创建完毕的通用组件。若是在软件初始化结束时还没有完成创建通用组件的操作，甚至在软件中的普通组件需要调用通用组件时才创建的话，由于创建通用组件需要占用一定的时长，因此，这种做法会造成普通组件调用通用组件的过程有较大时延。

[0043] 在执行步骤 101 时，当软件被初始化启动后，就为该软件生成一个通用组件库。

[0044] 以所述软件是即时通信软件为例，本步骤 101 的具体实现过程为：

[0045] 首先，实时检测所述即时通信软件是否被初始化启动，即该即时通信软件是否在所安装的设备中启动。

[0046] 然后，当检测出所述即时通信软件被初始化启动时，对即时通信软件所在的设备的内存资源进行扫描，并分配一定量的内存资源作为通用组件库的内存资源，所述分配的内存资源用于在通用组件库中存储后续所创建的通用组件。

[0047] 在为即时通信软件的通用组件库分配内存资源，为通用组件库的读写操作分配进程，以及记录通用组件库的入口地址和其他用于对通用组件库中的内容进行操作的信息后，所述通用组件库就生成完毕。

[0048] 为了提高通用组件库的访问速度，本实施例中生成的通用组件库中可采用映射 (map) 或哈希映射 (hashmap) 的数据结构。

[0049] 步骤 102：根据配置文件中记录的通用组件的属性信息，创建通用组件。

[0050] 在软件的开发过程中，已经将一些常用的元素封装成相对独立的组件，这些已封装的相对独立的组件中包括通用组件，因此，在软件开发完成时，通用组件的属性信息已记录在配置文件中，该配置文件中记录了能够用于创造通用组件的属性信息，且该配置文件存储在软件的文本存储区域中。

[0051] 所述属性信息包括但不限于以下信息中的一种或多种的组合：

[0052] 1、通用组件的类型，表示通用组件是什么功能的组件。

[0053] 例如：若通用组件的类型是 Background，表示该通用组件是用于生成背景的组件；若通用组件的类型是 Border，表示该通用组件是用于生成边框的组件。

[0054] 2、通用组件的名称。

[0055] 仍以一个 Background 类型的通用组件为例，其名称可以是在软件开发过程中所定义的名称。

[0056] 3、运行通用组件时所需数据的存储地址，用于表示通用组件在被调用运行时，所需调用的其他数据的存储地址。

[0057] 例如，一个 Background 类型的通用组件被调用以生成背景时，同时需要调用各背景图片，因此，在 Background 类型的通用组件的属性信息中需要记录各背景图片的存储地址，以便于在生成背景时可快速调用显示各项背景图片。

[0058] 在步骤 102 的执行过程中，由于配置文件中记录了多个通用组件的属性信息，每一组属性信息表示一个通用组件的属性，因此，当正确读取出配置文件中记录的每组属性信息后，可以使用该属性信息分别创建多个通用组件，具体的创建过程为：

[0059] 第一步：按照设定的实例创建方式，创建通用组件的实例。

[0060] 在本实施例的方案中，每个通用组件的实例创建过程相同，具体过程包括：

[0061] 根据待创建的通用组件的类型查找出相应的类工厂 (class factory)，并通过调用所述类工厂的创建实例 (CreateInstance) 成员函数完成对同类型通用组件的创建，并返回创建后的通用组件的实例。

[0062] 所述类工厂的实质是一个组件对象模型 (COM) 中的对象，类工厂中定义了一个 IClassFactory 接口，通过该 IClassFactory 接口中的 CreateInstance 成员函数，完成对通用组件的实例化。

[0063] 另外，根据 COM 规范的定义，组件对象类型 (COM Class) 和类工厂是配对出现的，也就是说，若要对某种类型通用组件实例化，就要通过该类型的通用组件对应的类工厂来实现。

[0064] 第二步：根据配置文件中记录的通用组件的属性信息，为创建的实例设置属性，得到创建完成的通用组件。

[0065] 所述为实例设置属性是指：通过调用组件的初始化 (Initialize) 算法，把一组属性信息传入实例，实现将属性应用于通用组件实例的目的。

[0066] 下面以 Background 类型的通用组件为例，来说明使用属性信息创建通用组件的过程：

[0067] 假设在配置文件中有两个 Backgound 通用组件，这两个通用组件的属性信息不同，第一个 Background 通用组件加载一个图片来填充背景，第二个 Background 通用组件使用白色来填充背景。

[0068] 配置文件中记录的这两个 Backgound 通用组件的属性信息可以如下所示：

[0069]

```
<Components>
  <Component>
    <Class>Background</Class>      //第一个通用组件的类型是Backgound
    <ID>Background_ID_1</ID>   //第一个 Backgound 通用组件名称为Background_ID_1
    <Style>Image</Style>          //第一个 Backgound 通用组件的填充方式是图像填充
    <Image>back.bmp</Image>   //第一个 Backgound 通用组件中图像填充用于填充背景
    <FillType>Tile</FillType>    //第一个 Backgound 通用组件的图像填充方式是平铺
  </Component>
  <Component>
    <Class>Background</Class>   //第二个通用组件的类型是 Backgound
    <ID>Background_ID_2</ID>   //第二个 Backgound 通用组件名称为Background_ID_2
    <Style>FillColor</Style>      //第二个 Backgound 通用组件是颜色填充
```

[0070]

```
    <Color>0xFFFFFF</Color>   //第二个 Backgound 通用组件的填充的颜色是白色
  </Component>
</Components>
```

[0071] 在创建过程中，当读取配置文件中的上述第一个通用组件的属性信息后，确定该通用组件的类型是 Background，则查找出对应的类工厂 BackgroundFactory。

[0072] 然后调用 BackgroundFactory 的 CreateInstance 成员函数完成第一个 Backgound 通用组件的创建，假设返回的第一个 Backgound 通用组件的实例保存在 spBackground1 中。

[0073] 接着，设置第一个 Backgound 通用组件的属性，调用通用组件的 Initialize 算法，将第一个 Backgound 通用组件的属性信息传入第一个 Backgound 通用组件的实例中。具体的传入方式是：

[0074] 运行 Initialize 算法，依次读取出第一个 Backgound 通用组件的每一个属性信息，针对每次读取的属性信息，调用相应的属性设置函数进行传入。

[0075] 例如，读取的属性信息是第一个 Backgound 通用组件的填充方式 (Style)，则调用 SetStyle 函数，将第一个 Backgound 通用组件的填充方式“图像填充 Image”传入实例；再例如，读取的属性信息是第一个 Backgound 通用组件的图像填充方式 (FillType)，则调用 SetFillType 函数，将第一个 Backgound 通用组件的图像填充方式“平铺 Tile”传入实例。

[0076] 在上述针对第一个通用组件创建完成后，继续读取第二个通用组件的属性信息，针对第二个通用组件的创建方式与针对第一个通用组件的创建方式相同，但是，在将第二个 Backgound 通用组件的属性信息传入第二个 Backgound 通用组件的实例中时，由于第一个通用组件和第二个通用组件的属性信息的内容不同，因此，最终创建出的两个 Backgound 通用组件能够实现不同外观效果的背景。

[0077] 步骤 103：将创建的通用组件以及通用组件的标识存储至所述通用组件库中。

[0078] 从步骤 102 的实现方式中可以看出，本申请实施例在创建实例时，是以某一类型的通用组件对应的类工厂来创建的，创建的同一类型的通用组件可以有多个实例，因此，为了详细区分各通用组件，当创建了实例后，就可以为每个创建的实例分配标识 (ID)。

[0079] 本申请实施例的方案中就是针对通用组件的实例来分配 ID 的，这样做的目的是：通过相同的类工厂创建的同类型通用组件，这些通用组件的类型虽然相同但其属性信息不完全相同，产生的行为结果也不同，因此，如果概括性地针对同类组件分配 ID 的话，可能会导致后续的调用过程出现错误，但使用本实施例中针对实例分配 ID 的方式，就可以避免对不同实例的通用组件进行调研，保证通用组件调用的精确度。

[0080] 具体地，本步骤 103 可以以分布式 Key-Value（键 - 值对）的存储方式进行存储，实现方式如下：

[0081] 第一步：确定指向创建的实例在通用组件库的存储位置的内容指针。

[0082] 以创建 Background 类型的通用组件（称之为通用组件 _1）为例，当在步骤 102 中根据通用组件 _1 的属性信息创建实例（称之为实例 _1）后，将该实例 _1 存储在通用组件库中，并定义指向实例 _1 存储位置的内容指针（称之为内容指针 _1）。

[0083] 第二步：为创建的通用组件的实例分配标识。

[0084] 仍以创建 Background 类型的通用组件 _1 为例，当在步骤 102 中根据通用组件 _1 的属性信息创建实例 _1 后，为该实例 _1 分配一个唯一表示该实例 _1 的 ID，例如，分配给该实例 _1 的标识为“ID_COMMON_BKGND1”。

[0085] 上述第一步和第二步的实现顺序不固定，也可以先执行第二步后执行第一步，也可以第一步和第二步并列执行。

[0086] 第三步：建立通用组件的实例的标识与内容指针之间的对应关系。

[0087] 在第一步中为通用组件 _1 的实例 _1 分配 ID，且在第二步中定义了指向实例 _1 在通用组件库中的存储位置的内容指针 _1 后，可以建立 ID 与内容指针 _1 之间的对应关系，所述对应关系中的标识所表示的实例与对应的内容指针指向的实例相同。

[0088] 第四步：将实例的标识与内容指针之间的对应关系存储至所述通用组件库中。

[0089] 此时，在通用组件库中不仅存储了通用组件的内容，还同时以 key-value 分布式存储系统的形式将标识与内容指针之间的对应关系存储在通用组件库中。

[0090] 本申请实施例一的步骤 102 和步骤 103 中，依次对将配置文件中记录的各通用组件的属性信息进行读取并操作，分别创建对应的通用组件并存储在通用组件库中，直至将配置文件中记录各属性信息对应的通用组件全部创建并存储。

[0091] 通过本申请实施例一的方案，在软件程序初始化过程中完成通用组件的统一创建以及统一存储的过程，与现有的由普通组件各自创建通用组件的方式相比，可有效减少创建通用组件的次数，进而减少通用组件的创建过程所占用的内存数量，提高普通组件的运行速度；同时，本实施例在软件初始化过程中完成通用组件的创建操作，使得普通组件可根据需求直接调用已创建完毕的通用组件，而无需再等待通用组件的创建过程，并且本实施例中通过创建实例的方式来创建通用组件，并为实例分配唯一的标识，即使通用组件库中存储了同类型的多个通用组件，也可以根据实例的 ID 查询出特定属性的通用组件，避免同类型的多个通用组件的混淆，保证了通用组件高精确度的调用。

[0092] 实施例二

[0093] 当采用实施例一的方案创建了通用组件后，就可以进一步对通用组件库中的通用组件进行调用，如图 4 所示，为本申请实施例二中调用通用组件的方法步骤示意图，所述方法包括以下步骤：

[0094] 步骤 201：确定待调用的通用组件的标识。

[0095] 在本实施例的方案中，普通组件内存储了可调用的通用组件的标识，当普通组件需要调用通用组件时，读取自身存储的通用组件的标识来进行调用操作。

[0096] 步骤 202：根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件。

[0097] 根据通用组件库中存储的通用组件的形式不同，本步骤的具体实现方式也有所不同。假设实施例一中，通用组件库中存储的是通用组件的实例，以及实例的标识与内容指针之间的对应关系，则在本步骤中，从通用组件库中查找所述标识所表示的通用组件的具体实现过程为：

[0098] 第一步：根据通用组件的实例的标识与内容指针之间的对应关系，确定待调用的通用组件的实例的标识对应的内容指针。

[0099] 步骤 201 中确定的待调用的通用组件的标识是能够唯一表示该通用组件的标识，若在实施例一种通用组件库中存储的是通用组件的实例的标识，则在步骤 201 中确定的通用组件的标识即为该通用组件的实例的标识。

[0100] 在确定待调用的通用组件的标识后，就可从通用组件库中查询与该标识对应的内容指针。

[0101] 第二步：将确定的内容指针所指向的实例作为查找出的通用组件的实例。

[0102] 由于通用组件是由实例创建的，因此，在查找出通用组件的实例后，也就是查找出了已创建的通用组件。

[0103] 步骤 203：调用查找出的通用组件。

[0104] 若在步骤 202 中查找出的是通用组件的实例，则在本步骤中通过调用通用组件的

实例入口，并运行所述实例，以达到调用、运行通用组件的目的。

[0105] 通过本申请实施例二的方案，普通组件在需要运行通用组件、获得通用组件的功能时，无需分别创建通用组件，而是直接调用通用组件库中已创建的通用组件，去除了普通组件分别创建通用组件时的时间消耗以及内存资源消耗，提高了组件运行效率；且本申请实施例的方案中，通过通用组件库内针对存储的各实例的固定 ID 来进行通用组件的访问，无需由系统动态为实例分配 ID，在保证通用组件正确调用的基础上，减少资源的占用。

[0106] 实施例三

[0107] 本申请实施例三通过一个具体的实例，来详细说明本申请实施例二的方案。

[0108] 假设：要调用通用组件的普通组件为会话窗口组件，会话窗口组件可调用的通用组件为背景通用组件和边框通用组件，背景通用组件的标识为 BackgroundID，用于指明会话窗口组件的背景由哪个背景通用组件来绘制；边框通用组件的标识为 BorderID，用于指明会话窗口组件的边框由哪个边框通用组件来绘制。这两个通用组件的标识记载在会话窗口组件内。

[0109] 如图 5 所示，为本实施例三种，会话窗口组件调用通用组件库中相应通用组件的步骤示意图，包括以下步骤：

[0110] 步骤 301：当会话窗口组件要开始绘制窗口时，确定用于绘制背景的背景通用组件的标识 BackgroundID，以及确定用于绘制边框的边框通用组件的标识 BorderID。

[0111] 步骤 302：会话窗口组件分别判断 BackgroundID 和 BorderID 是否有效，若是，则执行步骤 303；否则，若只有一个标识有效，则有效的标识继续执行步骤 303；若两个标识都无效，则停止调用通用组件的操作，利用会话窗口组件自身的绘制功能进行绘制。

[0112] 在本实施例的方案中，虽然会话窗口组件中存储了背景通用组件和边框通用组件的标识，但存储的通用组件的标识可能是无效的标识，为了保证会话窗口组件能够正确的调用通用组件，以及避免无效标识的失败调用额外占用系统资源的情况，本步骤中对用于查询通用组件的标识的有效性进行检查，只有有效的通用组件的标识才能进行后续的通用组件查找过程，对于无效的通用组件的标识不允许使用。

[0113] 以判断 BackgroundID 是否有效为例，其判断方式包括但不限于以下内容：

[0114] 第一步：预先设定有效的通用组件标识内容的字符串长度的范围值。

[0115] 在本申请各实施例的方案中，通用组件的标识（特殊地，通用组件实例的标识）内容是字符串，对于有效的标识，其内容中字符串的长度应该在一定的范围内，因此，可预先设定有效的标识内容的字符串长度的范围值。内容中的字符串长度在该范围值内的标识为有效标识，否则为无效标识。

[0116] 第二步：判断 BackgroundID 的内容是否为空，若是，则确定所述 BackgroundID 无效；否则，执行第三步。

[0117] 第三步：判断 BackgroundID 内容的字符串长度是否在设定的范围值内；若是，则确定所述 BackgroundID 有效；否则，确定所述 BackgroundID 无效。

[0118] 步骤 303：会话窗口组件通过有效的 BackgroundID 和 BorderID，从通用组件库中查找出对应的内容指针。

[0119] 在本实施例的方案中，会话窗口组件可以以同步方式，通过预设的函数查找多个内容指针并进行实例的调用。

[0120] 通过有效的 BackgroundID 查找出指向背景通用组件的实例的内容指针，通过有效的 BorderID 查找出指向边框通用组件的实例的内容指针。

[0121] 步骤 304：会话窗口组件利用查找出的内容指针，从通用组件库中调用指向的背景通用组件的实例和边框通用组件的实例。

[0122] 由于通过 BackgroundID 对应的内容指针可以确定背景通用组件的实例，因此，会话窗口组件调用并运行背景通用组件的实例，来绘制窗口的背景；同理，通过调用并运行边框通用组件的实例，来绘制窗口的边框。

[0123] 如图 6 所示，为通过本实施例三的方案，普通组件与通用组件之间的调用关系示意图，其中，左边实线的通用组件是创建并存储在通用组件库中的通用组件；右边虚线的通用组件是被普通组件调用的通用组件。除了会话窗口组件外，其他诸如系统消息窗口组件的普通组件需要调用背景通用组件和边框通用组件时，也可按照上述方式进行。

[0124] 实施例四

[0125] 本申请实施例四还提供一种与实施例一属于同一发明构思下的创建通用组件的设备，如图 7 所示，应用于具有封装后的组件的软件系统中，包括：库创建模块 11、组件创建模块 12 和存储模块 13，其中：库创建模块 11 用于在软件初始化时，生成通用组件库；组件创建模块 12 用于根据配置文件中记录的通用组件的属性信息，创建通用组件；存储模块 13 用于将创建的通用组件以及通用组件的标识存储至所述通用组件库中。

[0126] 所述组件创建模块 12 包括：实例创建子模块 21 和属性设置子模块 22，其中：实例创建子模块 21 用于按照设定的实例创建方式，创建通用组件的实例；属性设置子模块 22 用于根据配置文件中记录的通用组件的属性信息，为创建的实例设置属性。

[0127] 所述存储模块 13 包括：对应关系建立子模块 23 和执行子模块 24，其中：对应关系建立子模块 23 用于确定指向创建的实例在通用组件库中存储位置的内容指针，以及为创建的实例分配标识，并建立实例的标识与内容指针之间的对应关系；执行子模块 24 用于将实例的标识与内容指针之间的对应关系存储至所述通用组件库中。

[0128] 本实施例四中的创建通用组件的设备可以是能够执行实施例一各步骤的设备，其实现实施例一各步骤的逻辑部件不再一一赘述。

[0129] 本实施例四中的创建通用组件的设备可以是集成在软件中的客户端，也可以是独立于软件但能够调用软件中各项信息的客户端。

[0130] 实施例五

[0131] 本申请实施例五还提供一种与实施例二和实施例三属于同一发明构思下的调用通用组件的设备，如图 8 所示，包括：标识确定模块 31、查找模块 32 和调用模块 33，其中：标识确定模块 31 用于确定待调用的通用组件的标识；查找模块 32 用于根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件；调用模块 33 用于调用查找出的通用组件。

[0132] 还包括判定模块 34，用于判断标识确定模块 31 确定的所述标识的内容是否为空，在所述标识的内容不为空时，且确定所述标识内容的字符串长度在设定范围值内时，触发所述查找模块 32。

[0133] 所述查找模块 32 包括：指针确定子模块 41 和操作子模块 42，其中：指针确定子模块 41 用于根据通用组件的实例的标识与内容指针之间的对应关系，确定待调用的通用组

件的实例的标识对应的内容指针；操作子模块 42 用于将确定的内容指针所指向的实例作为查找出的通用组件的实例。

[0134] 所述调用模块 33，具体用于调用查找出的通用组件的实例入口，并运行所述实例。

[0135] 本实施例五中的调用通用组件的设备可以是能够执行实施例二和实施例三各步骤的设备，其实现实施例二和实施例三各步骤的逻辑部件不再一一赘述。

[0136] 本实施例五中的调用通用组件的设备可以集成在软件中的各普通组件中，当某一普通组件需要调用通用组件时，可以根据本地集成的调用通用组件的设备的功能完成调用；调用通用组件的设备也可以独立于软件的普通组件，当某一普通组件需要调用通用组件时，应首先运行调用通用组件的设备，通过该设备完成对通用组件的调用。

[0137] 本领域内的技术人员应明白，本申请的实施例可提供为方法、系统、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器、CD-ROM、光学存储器等）上实施的计算机程序产品的形式。

[0138] 本申请是参照根据本申请实施例的方法、设备（系统）、和计算机程序产品的流程图和 / 或方框图来描述的。应理解可由计算机程序指令实现流程图和 / 或方框图中的每一流程和 / 或方框、以及流程图和 / 或方框图中的流程和 / 或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能的装置。

[0139] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能。

[0140] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能的步骤。

[0141] 尽管已描述了本申请的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例做出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本申请范围的所有变更和修改。

[0142] 显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。

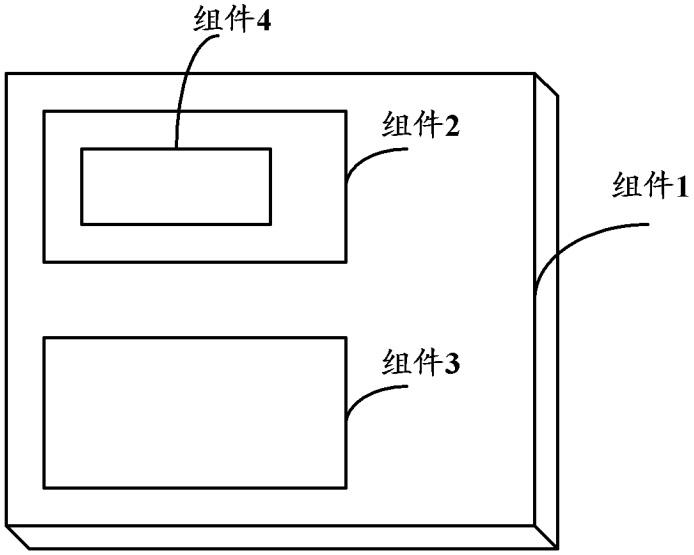


图 1

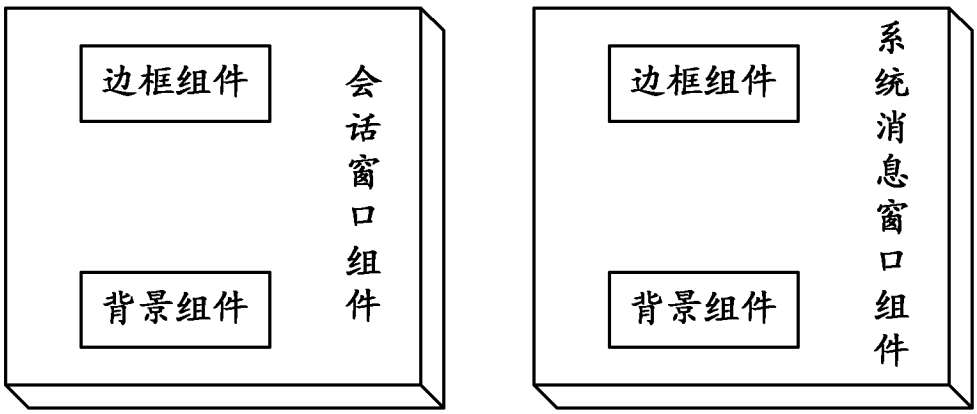


图 2

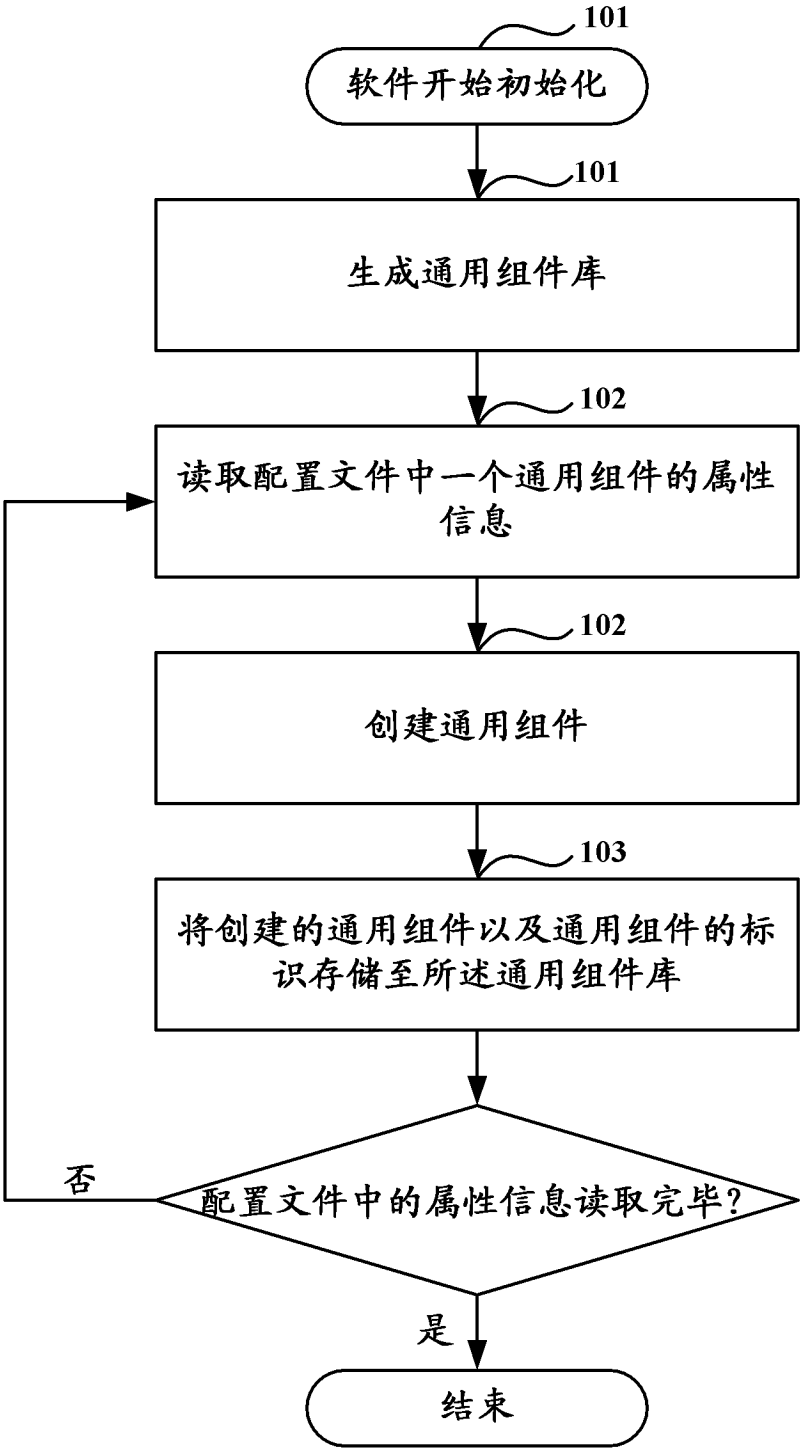

101
软件开始初始化
101
生成通用组件库
102
读取配置文件中一个通用组件的属性信息
102
创建通用组件
103
将创建的通用组件以及通用组件的标识存储至所述通用组件库
否
配置文件中的属性信息读取完毕?
是
结束

图 3

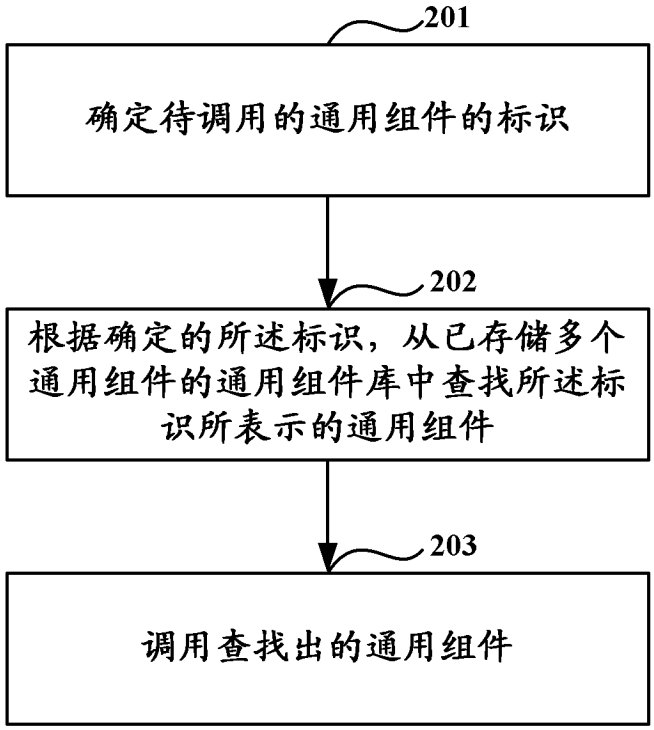

201
确定待调用的通用组件的标识
202
根据确定的所述标识，从已存储多个通用组件的通用组件库中查找所述标识所表示的通用组件
203
调用查找出的通用组件

图 4

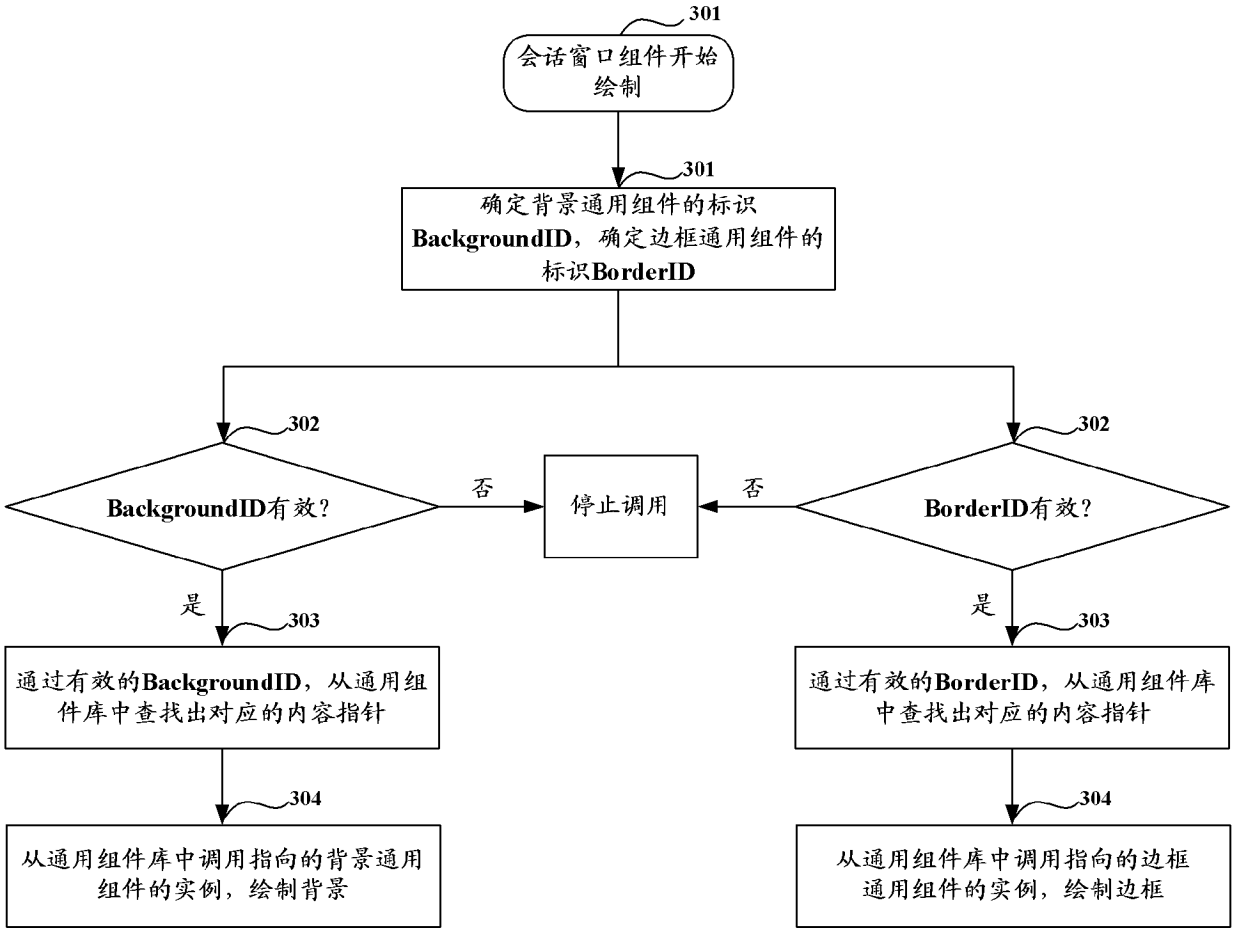

301
会话窗口组件开始绘制
301
确定背景通用组件的标识BackgroundID，确定边框通用组件的标识BorderID
302
BackgroundID有效？
否
停止调用
否
302
BorderID有效？
是
303
通过有效的BackgroundID，从通用组件库中查找出对应的内容指针
是
303
通过有效的BorderID，从通用组件库中查找出对应的内容指针
304
从通用组件库中调用指向的背景通用组件的实例，绘制背景
304
从通用组件库中调用指向的边框通用组件的实例，绘制边框

图 5

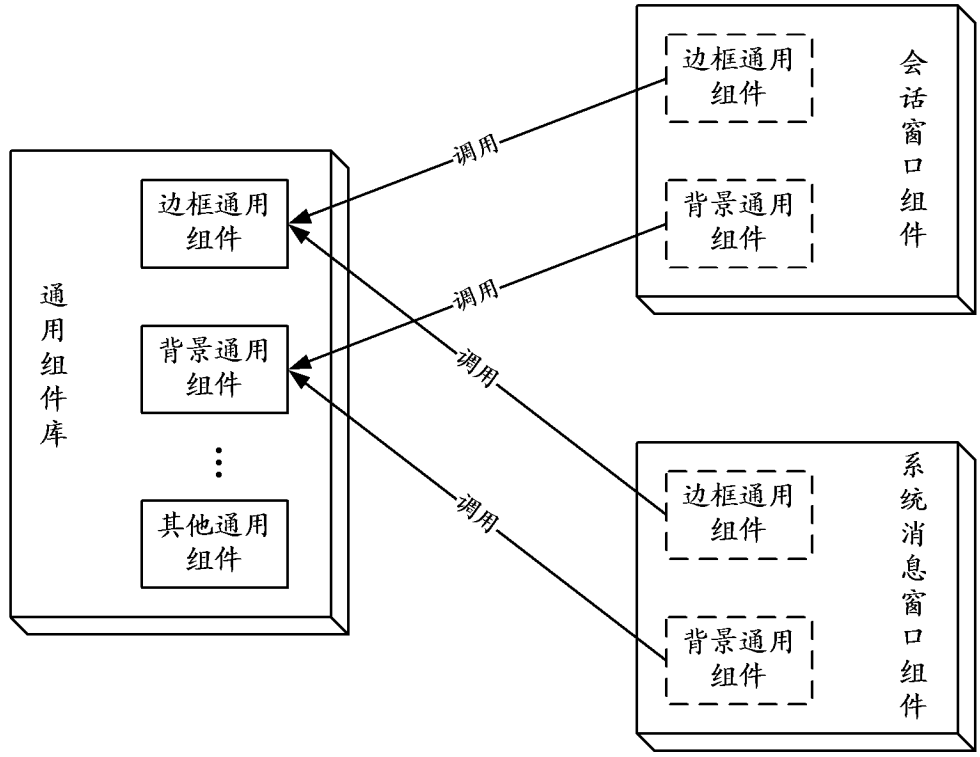

通用组件库
边框通用组件
背景通用组件
其他通用组件
调用
调用
调用
调用
会话窗口组件
边框通用组件
背景通用组件
系统消息窗口组件
边框通用组件
背景通用组件

图 6

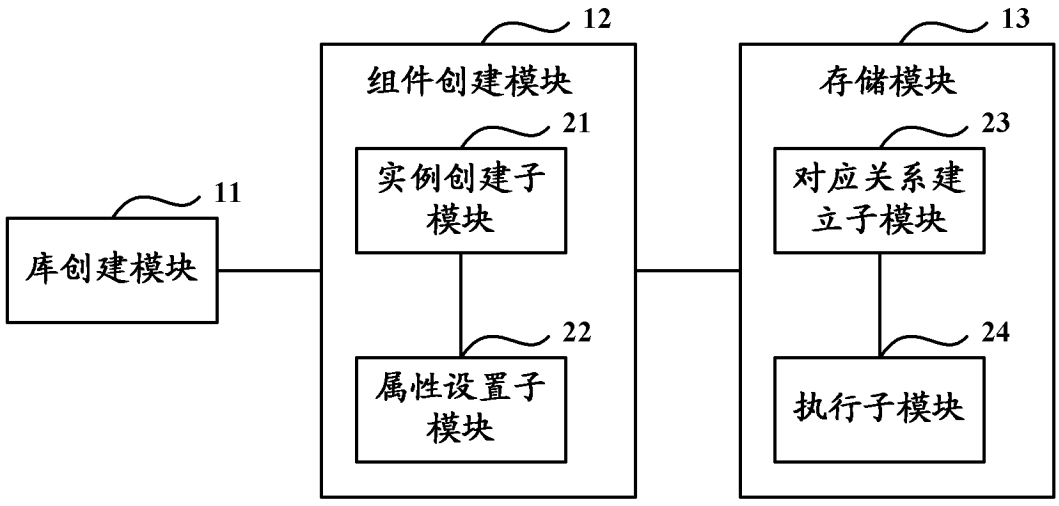

11
库创建模块
12
组件创建模块
21
实例创建子模块
22
属性设置子模块
13
存储模块
23
对应关系建立子模块
24
执行子模块

图 7

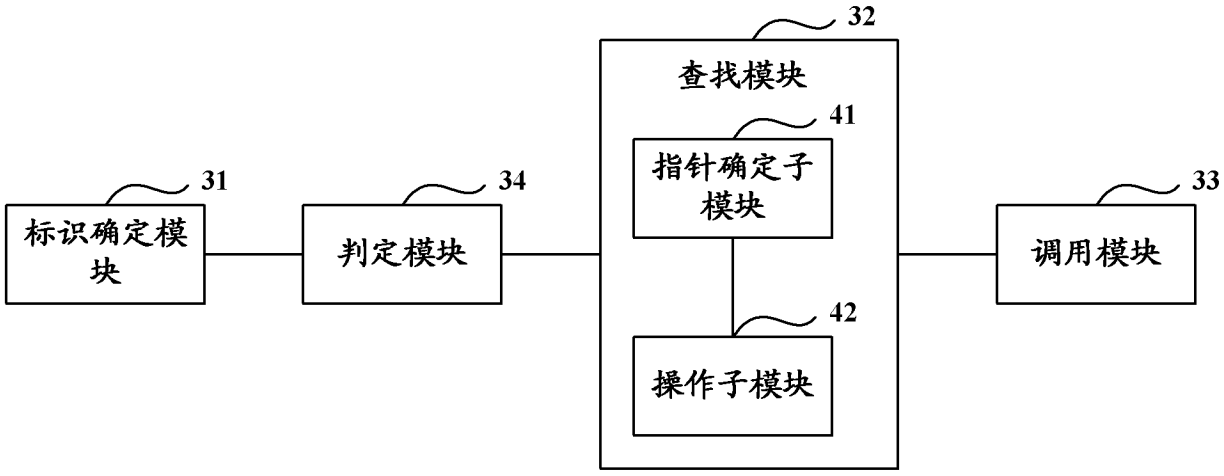


图 8