(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106293680 A
(43)申请公布日 2017.01.04

(21)申请号 201510317830.1

(22)申请日 2015.06.11

(71)申请人 中兴通讯股份有限公司
地址 518057 广东省深圳市南山区高新技术产业园科技南路中兴通讯大厦

(72)发明人 周雅琴

(74)专利代理机构 深圳鼎合诚知识产权代理有限公司 44281
代理人 薛祥辉 李发兵

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

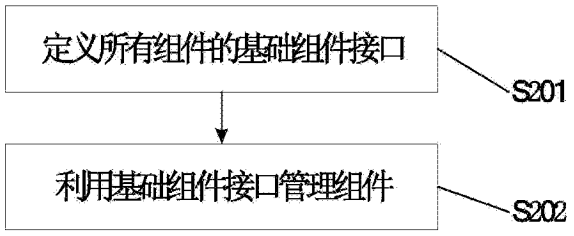
权利要求书1页 说明书5页 附图2页

(54)发明名称

一种组件管理方法及装置

(57)摘要

本发明提供了一种组件管理方法及装置，以提高组件封装性；该方法包括：定义所有组件的基础组件接口；利用基础组件接口管理组件。通过本发明的实施，提供统一的基础组件接口，统一对组件内不同类型组件元素的注册/调用等进行管理，跨组件使用在目标接口中定义了的成员函数时，无需关注该函数所在组件是否加载，即可保证对装置代码的正常维护，组件内部函数实现修改在组件内部完成，在输入输出参数无变化时对组件对外接口的定义无影响，对组件间函数调用也无影响，这样就保证了单个组件内部变更的封装性。



CN 106293680 A

1.一种组件管理方法,其特征在于,包括:

定义所有组件的基础组件接口;

利用所述基础组件接口管理所述组件。

2.如权利要求1所述的组件管理方法,其特征在于,利用所述基础组件接口管理所述组件包括:建立组件树,控制所述组件利用所述基础组件接口注册到所述组件树。

3.如权利要求2所述的组件管理方法,其特征在于,所述组件包括LIB组件及Module组件,所述基础组件接口包括组件注册接口CO_Create及组件加载接口CO_LoadModules;控制组件注册到所述组件树包括:控制LIB组件利用CO_Create注册之后,利用CO_LoadModules完成该LIB级组件下module级组件表中所有module组件的注册,和/或,控制Module组件利用CO_Create注册。

4.如权利要求2所述的组件管理方法,其特征在于,所述基础组件接口包括用于完成组件对外服务接口调用的调用接口CO_CALL;利用所述基础组件接口管理所述组件包括还包括:通过调用接口CO_CALL,根据组件标识在组件树内获取组件节点,获取所述组件节点对应实例数据中的成员接口,根据接口标识查找接口实例,调用所述接口实例中函数标识对应的函数。

5.如权利要求1至4任一项所述的组件管理方法,其特征在于,还包括:利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

6.一种组件管理装置,其特征在于,包括:

设置模块,用于定义所有组件的基础组件接口;

管理模块,用于利用所述基础组件接口管理所述组件。

7.如权利要求6所述的组件管理装置,其特征在于,所述管理模块用于建立组件树,控制所述组件利用所述基础组件接口注册到所述组件树。

8.如权利要求7所述的组件管理装置,其特征在于,所述组件包括LIB组件及Module组件,所述基础组件接口包括组件注册接口CO_Create及组件加载接口CO_LoadModules;所述管理模块具体用于控制LIB组件利用CO_Create注册之后,利用CO_LoadModules完成该LIB级组件下module级组件表中所有module组件的注册,和/或,控制Module组件利用CO_Create注册。

9.如权利要求7所述的组件管理装置,其特征在于,所述基础组件接口包括用于完成组件对外服务接口调用的调用接口CO_CALL;所述管理模块还用于通过调用接口CO_CALL,根据组件标识在组件树内获取组件节点,获取所述组件节点对应实例数据中的成员接口,根据接口标识查找接口实例,调用所述接口实例中函数标识对应的函数。

10.如权利要求6至9任一项所述的组件管理装置,其特征在于,还包括保护模块,用于利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

一种组件管理方法及装置

技术领域

[0001] 本发明涉及代码模块管理领域，尤其涉及一种组件管理方法及装置。

背景技术

[0002] 当前通讯领域面对项目多、软件功能模块多的现状，如何在最大化实现代码共享的同时保证模块自身的封装性、可复用性及对外独立交付需求是降低项目管理成本和提高代码质量所需解决的问题。

[0003] 现有技术采用的方法包括：1) 定义顶层初始化登记表、模块登记表和模块族登记表的全局变量数组；2) 遍历顶层初始化登记表，完成顶层初始化；3) 完成调度初始化：根据模块族登记表完成模块公共消息处理任务初始化，根据模块登记表完成模块自处理任务初始化；4) 由模块公共处理任务和模块自处理任务完成消息管理。该方法因为组件内部实现对外提供全局变量数据太多，降低了组件的封装性。

[0004] 因此，如何提供一种可提高组件封装性的组件管理方法，是本领域技术人员亟待解决的技术问题。

发明内容

[0005] 本发明提供了一种组件管理方法及装置，以提高组件封装性。

[0006] 本发明提供了一种组件管理方法，其包括：定义所有组件的基础组件接口；利用基础组件接口管理组件。

[0007] 进一步的，利用基础组件接口管理组件包括：建立组件树，控制组件利用基础组件接口注册到组件树。

[0008] 进一步的，组件包括LIB组件及Module组件，基础组件接口包括组件注册接口CO_Create及组件加载接口CO_LoadModules；控制组件注册到组件树包括：控制LIB组件利用CO_Create注册之后，利用CO_LoadModules完成该LIB级组件下module级组件表中所有module组件的注册，和/或，控制Module组件利用CO_Create注册。

[0009] 进一步的，基础组件接口包括用于完成组件对外服务接口调用的调用接口CO_CALL；利用基础组件接口管理组件包括还包括：通过调用接口CO_CALL，根据组件标识在组件树内获取组件节点，获取组件节点对应实例数据中的成员接口，根据接口标识查找接口实例，调用接口实例中函数标识对应的函数。

[0010] 进一步的，还包括：利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

[0011] 本发明提供了一种组件管理装置，其包括：设置模块，用于定义所有组件的基础组件接口；管理模块，用于利用基础组件接口管理组件。

[0012] 进一步的，管理模块用于建立组件树，控制组件利用基础组件接口注册到组件树。

[0013] 进一步的，组件包括LIB组件及Module组件，基础组件接口包括组件注册接口CO_Create及组件加载接口CO_LoadModules；管理模块具体用于控制LIB组件利用CO_Create

注册之后，利用 CO_LoadModules 完成该 LIB 级组件下 module 级组件表中所有 module 组件的注册，和 / 或，控制 Module 组件利用 CO_Create 注册。

[0014] 进一步的，基础组件接口包括用于完成组件对外服务接口调用的调用接口 CO_CALL ；管理模块还用于通过调用接口 CO_CALL，根据组件标识在组件树内获取组件节点，获取组件节点对应实例数据中的成员接口，根据接口标识查找接口实例，调用接口实例中函数标识对应的函数。

[0015] 进一步的，还包括保护模块，用于利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

[0016] 本发明的有益效果 ：

[0017] 本发明提供了一种组件管理方法，提供统一的基础组件接口，统一对组件内不同类型组件元素的注册 / 调用等进行管理，跨组件使用在目标接口中定义了的成员函数时，无需关注该函数所在组件是否加载，即可保证对装置代码的正常维护，组件内部函数实现修改在组件内部完成，在输入输出参数无变化时对组件对外接口的定义无影响，对组件间函数调用也无影响，这样就保证了单个组件内部变更的封装性。

附图说明

[0018] 图 1 为本发明第一实施例提供的组件管理装置的结构示意图 ；

[0019] 图 2 为本发明第二实施例提供的组件管理方法的流程图 ；

[0020] 图 3 为本发明第三实施例提供的组件管理方法的流程图。

具体实施方式

[0021] 现通过具体实施方式结合附图的方式对本发明做出进一步的诠释说明。

[0022] 第一实施例 ：

[0023] 图 1 为本发明第一实施例提供的组件管理装置的结构示意图，由图 1 可知，在本实施例中，本发明提供的组件管理装置 1 包括 ：

[0024] 设置模块 11，用于定义所有组件的基础组件接口 ；

[0025] 管理模块 12，用于利用基础组件接口管理组件。

[0026] 在一些实施例中，上述实施例中的管理模块 12 用于建立组件树，控制组件利用基础组件接口注册到组件树。

[0027] 在一些实施例中，上述实施例中的组件包括 LIB 组件及 Module 组件，基础组件接口包括组件注册接口 CO_Create 及组件加载接口 CO_LoadModules ；管理模块 12 具体用于控制 LIB 组件利用 CO_Create 注册之后，利用 CO_LoadModules 完成该 LIB 级组件下 module 级组件表中所有 module 组件的注册，和 / 或，控制 Module 组件利用 CO_Create 注册。

[0028] 在一些实施例中，上述实施例中的基础组件接口包括用于完成组件对外服务接口调用的调用接口 CO_CALL ；管理模块 12 还用于通过调用接口 CO_CALL，根据组件标识在组件树内获取组件节点，获取组件节点对应实例数据中的成员接口，根据接口标识查找接口实例，调用接口实例中函数标识对应的函数。

[0029] 在一些实施例中，如图 1 所示，上述实施例中的组件管理装置 1 还包括保护模块 13，用于利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

[0030] 第二实施例：

[0031] 图2为本发明第二实施例提供的组件管理方法的流程图，由图2可知，在本实施例中，本发明提供的组件管理方法包括以下步骤：

[0032] S201：定义所有组件的基础组件接口；

[0033] S202：利用基础组件接口管理组件。

[0034] 在一些实施例中，上述实施例中的利用基础组件接口管理组件包括：建立组件树，控制组件利用基础组件接口注册到组件树。

[0035] 在一些实施例中，上述实施例中的组件包括LIB组件及Module组件，基础组件接口包括组件注册接口CO_Create及组件加载接口CO_LoadModules；控制组件注册到组件树包括：控制LIB组件利用CO_Create注册之后，利用CO_LoadModules完成该LIB级组件下module级组件表中所有module组件的注册，和/或，控制Module组件利用CO_Create注册。

[0036] 在一些实施例中，上述实施例中的基础组件接口包括用于完成组件对外服务接口调用的调用接口CO_CALL；利用基础组件接口管理组件包括还包括：通过调用接口CO_CALL，根据组件标识在组件树内获取组件节点，获取组件节点对应实例数据中的成员接口，根据接口标识查找接口实例，调用接口实例中函数标识对应的函数。

[0037] 在一些实施例中，上述实施例中的组件管理方法还包括：利用符号检查工具根据编译规则对组件编译进行符号隐藏和生成库属性划分。

[0038] 现结合具体应用场景对本发明做进一步的诠释说明。

[0039] 第三实施例：

[0040] 图3为本发明第三实施例提供的组件管理方法的流程图，由图3可知，在本实施例中，本发明提供的组件管理方法包括以下步骤：

[0041] S301：定义组件类型、基础组件接口、基础函数；

[0042] 组件管理装置定义并提供一套基础组件接口，装置实现应使用组件管理装置提供的基础组件接口完成所需功能，以屏蔽组件内部实现差异

[0043] 组件管理装置按照独立交付的思想将装置中组件及软件接口表达分为如下层次：软件包(LIB)、软件模块(module)、接口(interface)和函数(function)。装置约定组件包含LIB和module两种类；interface定义为组件对外发布的服务接口，每个组件内部可定义多个interface，其按照子功能类别进行划分；一个interface内部可以包含多个function，function是实现组件服务的最基层单元。组件管理装置为组件内部interface、function实现定义了统一规范，组件内部需按此统一规范进行定义，以达到在interface接口层上保证组件的独立交付的目的。

[0044] 组件管理装置将组件逻辑标识定义为PROG_ID，interface逻辑标识定义为INTERFACE，function逻辑标识定义为FUNCTION。组件管理装置定义了一组表现形式为CO_CALL的调用方法来支持对服务接口的访问，CO_CALL使用PROG_ID、INTERFACE、FUNCTION来寻址组件function实际地址，寻址成功的产生实际调用行为，寻址失败的返回错误码。基于此，组件是否“编译、注册、初始化”仅影响组件间函数调用的实际效果，但不影响版本的编译、生成和运行。

[0045] 组件管理装置为了从编译上对组件间函数调用方法进行限制，对组件编译进行

了符号隐藏和生成库属性划分，装置下的库被分为组件动态库、共享库、静态库，版本生成过程将由编译链接规则及额外增加的符号检查工具对组件内部函数进行保护，以推动使用组件 CO_CALL 方式进行组件间函数调用，而约束使用 extern 或 __attribute__((visibility("default"))) 属性进行函数声明和调用，因为使用 extern 或 __attribute__((visibility("default"))) 属性进行函数声明和调用在如上编译约束情况时，对组件函数使用有如下约束：对于使用了隐藏规则的静态库，其中引出的函数不能在除静态库及 app.bin 外使用；对使用了隐藏属性的共享库，其中未引出的函数接口不能在本组件外使用；对于使用了隐藏属性的动态库中函数接口不能在本组件外使用。

[0046] 组件管理装置为实现组件管理定义了一套命名以 CO_ 为前缀基础函数，如：CO_Create、CO_LoadLibs、CO_LoadModules、CO_LoadDLL、CO_CALL、CO_MsgDispatch、CO_QueryInterface 等，这组基础组件接口为组件统一管理提供支持。其中，CO_LoadLibs、CO_LoadModules、CO_LoadDLL 分别用于完成 LIB 组件、module 组件和动态库组件的加载；CO_Create 用于将组件挂接到组件树上；CO_CALL 用于完成组件对外服务接口调用；CO_MsgDispatch 用于实现组件内消息的转发；CO_QueryInterface 用于依据索引 PROG_ID、INTERFACE、FUNCTION 来完成对组件 interface 及 function 的查找，是其它基础函数实现的基础。

[0047] 组件管理装置定义的组件内部实现规范约定包括如下几点：为组件 interface 指定定义格式、为组件注册函数指定定义格式、约定将组件 interface 汇总到组件接口集注册表中。组件管理装置提供了三个标准接口：CO_IUNKNOWN、CO_IBASIC、CO_IMSG_MA，还约定了组件服务接口定义格式：CO_IXXX，组件内部将为 interface 创建 interface 实例，所有本组件接口实例需汇总到本组件接口集注册表中，每个组件有一个 libmain 函数，用于完成组件的注册，组件管理装置为 libmain 函数约定了定义格式。

[0048] S302：组件树初始化。

[0049] 组件树初始化包括组件内存申请和组件树创建，后续注册的组件将挂接在组件树结点上。装置启动时，会调用 Lib 初始化接口完成组件树的初始化。

[0050] S303：组件注册。

[0051] 组件通过组件管理装置提供的组件注册接口 CO_Create 将组件挂接到组件树中，组件注册后，组件管理装置通过基础接口 CO_CALL 就可以完成对组件服务接口的调用了。

[0052] 在实际应用中，组件注册包括组件注册 module 流程及组件注册接口 CO_Create 流程。

[0053] 具体的，组件注册 module 流程具体包括以下步骤：

[0054] 判断组件是否为 LIB 级组件；

[0055] 若是，则调用 CO_Create 完成 LIB 组件注册，之后，调用 CO_LoadModules 完成该 LIB 级组件下 module 级组件表中所有 module 组件的注册；

[0056] 若否，则调用 CO_Create 完成 Module 组件的注册。

[0057] 组件注册 module 流程实现将组件挂靠在组件树上，在此基础上进行组件注册接口 CO_Create 流程；具体的，组件注册接口 CO_Create 流程具体包括以下步骤：

[0058] 在组件树上查找组件标识 PROG_ID 对应的节点；

[0059] 若查找到节点，则在组件树新增组件节点，并依次将组件模块 CO_IUNKNOWN 实例、

组件版本信息、组件消息处理表信息、组件内进程表信息记入组件节点，完成组件注册接口CO_Create的流程；

[0060] 若没有查找到节点，则流程结束。

[0061] S304：函数调用。

[0062] 在实际应用中，调用组件实际是调用组件内的函数，在上述步骤的基础上，本发明的函数调用流程包括以下步骤：

[0063] 根据组件标识PROG_ID在组件树内查找到对应的组件节点；

[0064] 获取该组件节点中记录的CO_IUNKNOWN实例数据；

[0065] 调用该组件CO_IUNKNOWN实例数据中QueryInterface成员函数，查找接口标识INTERFACE对应的interface实例；

[0066] 在该interface实例中查找函数标识FUNCTION对应的function函数；

[0067] 调用function函数。

[0068] 针对任一个组件，组件管理装置还需要定义CO_IUNKNOWN、CO_IBASIC、CO_IMSG_MAP这三个标准interface的实例，组件管理装置根据实际需求按组件管理装置规范格式CO_IXXX定义组件服务interface及其实例。组件管理装置按照规范结合组件实际需求为interface实例创建function函数，并将其登记到对应interface中对应位置（不需要情况可不创建function，登记位置填写NULL）。将本组件中所有interface实例，汇总到本组件的接口注册表中，创建组件libmain函数。将非动态加载组件的libmain函数挂接到装置Lib注册表中或者Lib注册表中某Lib下的module注册表中；将动态加载的组件信息添加到动态加载Lib配置表中。在此基础上，装置就通过CO_CALL来完成对本组件内function的调用了。

[0069] 综上可知，通过本发明的实施，至少存在以下有益效果：

[0070] 采用本发明组件管理方法，跨组件使用在interface中定义了的成员函数时，无需关注该函数所在组件是否加载，即可保证对装置代码的正常维护，组件内部function实现修改在组件内部完成，在输入输出参数无变化时对组件对外interface的定义无影响，对组件间function调用也无影响，这样就保证了单个组件内部变更的封装性。

[0071] 进一步的，在调用组件内函数时，根据组件标识、接口标识及函数标识进行查找，不涉及组件内数据安全性，这样，组件内的函数可以自行更新，提高了组件封装性。

[0072] 进一步的，另外当两个不同项目拥有不同实现同一命名的库时，两者可使用统一的对外interface结合一定编译规则来选择所编译的模块，以实现同含义function在组件使用中的可复用性，以屏蔽两组件内部function实现的差异。同时使用组件框架所提供的组件动态库加载函数可支持组件的独立开发和运行加载，为新增组件动态库的独立交付提供支持。

[0073] 以上仅是本发明的具体实施方式而已，并非对本发明做任何形式上的限制，凡是依据本发明的技术实质对以上实施方式所做的任意简单修改、等同变化、结合或修饰，均仍属于本发明技术方案的保护范围。

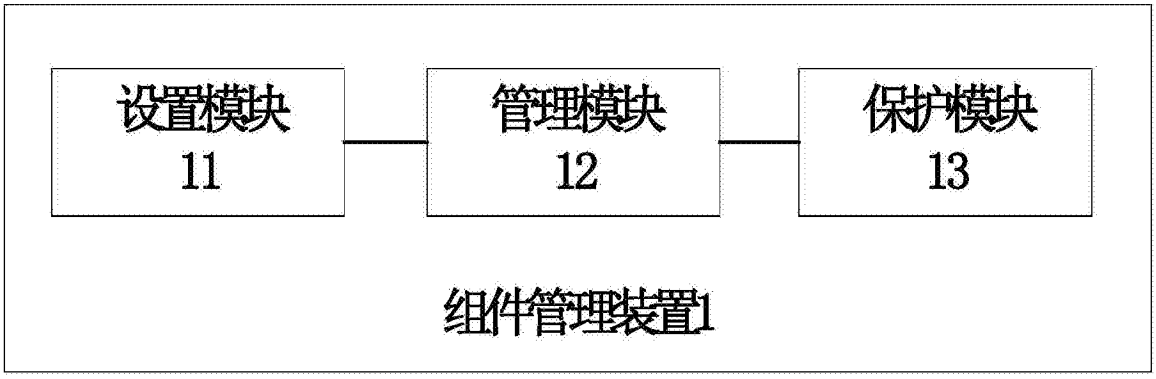

设置模块
11
管理模块
12
保护模块
13
组件管理装置1

图 1

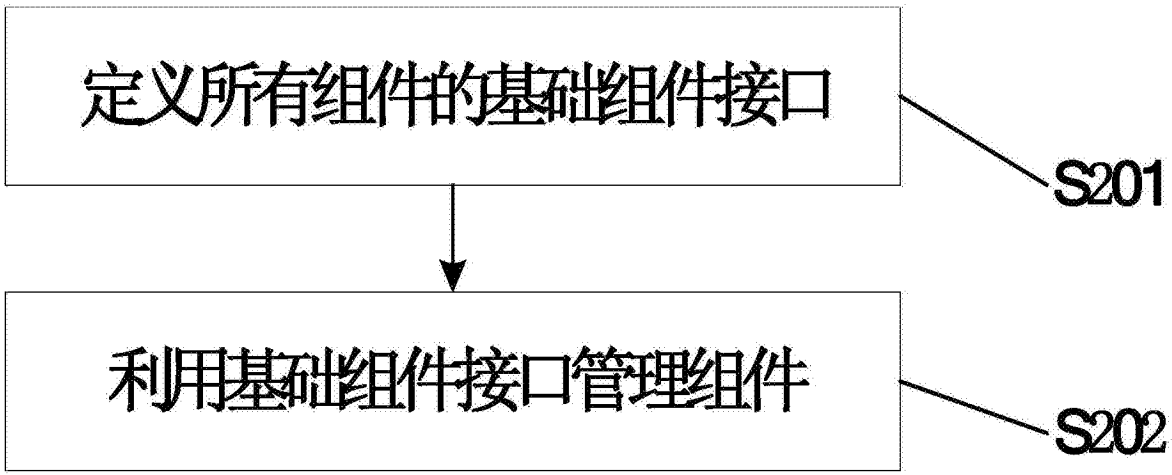

定义所有组件的基础组件接口
S201
利用基础组件接口管理组件
S202

图 2

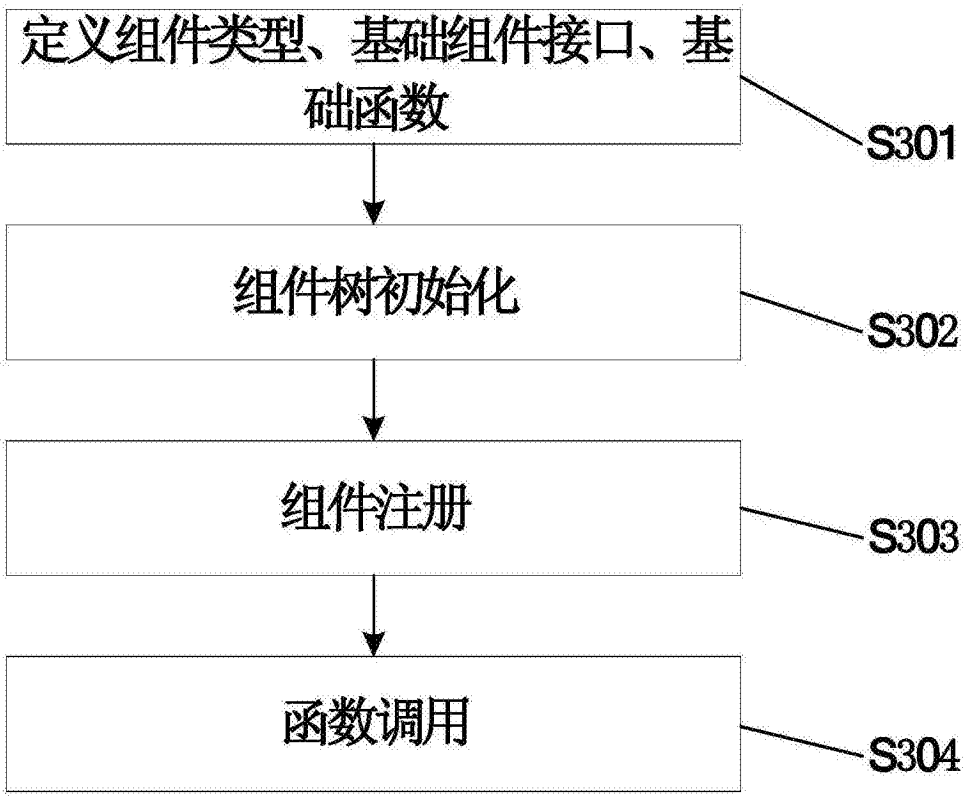


图 3