(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 103870260 B
(45)授权公告日 2019.01.08

(21)申请号 201210544215.0

(22)申请日 2012.12.14

(65)同一申请的已公布的文献号
申请公布号 CN 103870260 A

(43)申请公布日 2014.06.18

(73)专利权人 腾讯科技(深圳)有限公司
地址 518044 广东省深圳市福田区振兴路赛格科技园2栋东403室

(72)发明人 蔡宇轩

(74)专利代理机构 北京康信知识产权代理有限责任公司 11240
代理人 赵囡囡

(51)Int.Cl.
G06F 8/36(2018.01)
G06F 8/41(2018.01)

(56)对比文件
CN 102810070 A,2012.12.05,
US 2003005110 A1,2003.01.02,
CN 101650648 A,2010.02.17,

审查员 杨昕瑞

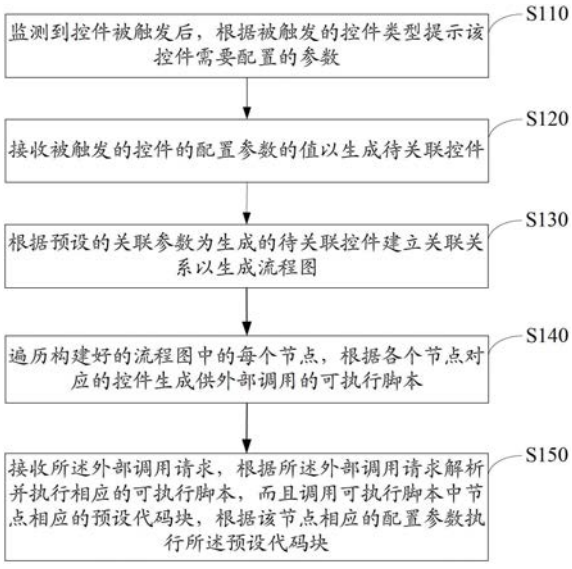
权利要求书2页 说明书12页 附图3页

(54)发明名称

业务接口开发的方法及系统

(57)摘要

本发明公开一种业务接口开发的方法及系统，该方法包括：监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数；接收被触发的控件的配置参数的值以生成待关联控件；根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图；遍历所述流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本；接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块。本发明使得业务逻辑可以以流程图的形式可视化地展示，不但简化了业务开发的过程，也降低了业务开发人员的开发难度。



CN 103870260 B

1.一种业务接口的开发的方法,其特征在于,包括以下步骤:

监测到控件被触发后,根据被触发的控件类型在用户端提示该控件需要配置的参数,其中,被触发的控件为所述业务接口的开发系统提供的用于在所述用户端绘制流程图的控件;

接收被触发的控件的配置参数的值以生成待关联控件;

根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图;

遍历所述流程图中的每个节点,根据各个节点对应的控件生成供外部调用的可执行脚本;

接收所述外部调用请求,根据所述外部调用请求解析并执行相应的可执行脚本,而且调用可执行脚本中节点相应的预设代码块,根据该节点相应的配置参数执行所述预设代码块。

2.根据权利要求1所述的业务接口的开发的方法,其特征在于,所述遍历所述流程图中的每个节点,根据各个节点对应的控件生成供外部调用的可执行脚本包括:

遍历所述流程图中的所有节点,获取每个节点的节点标识及节点类型;

根据节点类型的不同,获取该节点相应的节点信息;

根据节点类型对应的预设代码模板,将各个节点的节点标识及节点信息赋值到所述代码模板中以生成可执行脚本。

3.根据权利要求1所述的业务接口开发的方法,其特征在于,所述遍历所述流程图中的每个节点,根据各个节点对应的控件生成供外部调用的可执行脚本之后还包括:

接收调试请求,解析所述调试请求相应的可执行脚本,执行该脚本并在执行成功后的脚本行或脚本行对应的流程图节点添加执行标识。

4.根据权利要求1-3任一项所述的业务接口开发的方法,其特征在于,所述根据外部调用请求解析并执行相应的可执行脚本包括:

接收外部调用请求,并根据所述外部调用请求创建调用栈和输入参数栈;

根据外部调用请求中的根节点名,获取所述根节点名对应的可执行脚本;

利用调用栈和输入参数栈执行所述可执行脚本。

5.根据权利要求1-3任一项所述的业务接口开发的方法,其特征在于,还包括:

在可执行脚本执行期间,将所有调用的输入和调用流程图所执行过的节点的标识按照执行顺序生成日志。

6.一种业务接口开发的系统,其特征在于,包括:

流程图配置模块,用于监测到控件被触发后,根据被触发的控件类型在用户端提示该控件需要配置的参数,其中,被触发的控件为所述业务接口的开发系统提供的用于在所述用户端绘制流程图的控件;接收被触发的控件的配置参数的值以生成待关联控件;根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图;

流程图解析模块,用于遍历所述流程图中的每个节点,根据各个节点对应的控件生成供外部调用的可执行脚本;

脚本执行模块,用于接收所述外部调用请求,根据所述外部调用请求解析并执行相应的可执行脚本,而且调用可执行脚本中节点相应的预设代码块,根据该节点相应的配置参数执行所述预设代码块。

7.根据权利要求6所述的业务接口开发的系统，其特征在于，所述流程图配置模块用于：

遍历所述流程图中的所有节点，获取每个节点的节点标识及配置参数的值；根据预设的代码模板，将各个节点的节点标识及配置参数的值赋值到所述代码模板中以生成可执行脚本。

8.根据权利要求6所述业务接口开发的系统，其特征在于，还包括：

调试模块，接收调试请求，解析所述调试请求相应的可执行脚本，执行该脚本并在执行成功后的脚本行或脚本行对应的流程图节点添加执行标识。

9.根据权利要求6-8任一项所述的业务接口开发的系统，其特征在于，所述脚本执行模块用于：

接收所述外部调用请求，并根据所述外部调用请求创建调用栈和输入参数栈；

根据外部调用请求中的根节点名，获取所述根节点名对应的可执行脚本；

利用调用栈和输入参数栈执行所述可执行脚本。

10.根据权利要求9所述的业务接口开发的系统，其特征在于，所述脚本执行模块还用于：

在可执行脚本执行期间，将所有调用的输入和调用流程图所执行过的节点的标识按照执行顺序生成日志。

业务接口开发的方法及系统

技术领域

[0001] 本发明涉及业务接口开发领域，尤其涉及一种业务接口开发的方法及系统。

背景技术

[0002] 目前很多业务开发面临这样的困难：产品开发周期短，产品上线后维护阶段还时常要修改需求，需求修改完后又得经过程序员修改代码、调试、测试几个阶段。使得很多程序员希望花更少的精力花在产品的重复类似代码的开发、调试、测试。

[0003] 在开发阶段，程序员需要花费比较多的时间在编译、调试、修改、再次编译、调试等等，从而浪费了不少开发时间，也很容易使得程序员因为经常重复编写修改类似的代码和等待编译而感到厌烦。

发明内容

[0004] 本发明的主要目的是提供一种业务接口开发的方法，旨在简单、快捷地进行业务开发。

[0005] 本发明提供了一种业务接口开发的方法，包括以下步骤：

[0006] 监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数；

[0007] 接收被触发的控件的配置参数的值以生成待关联控件；

[0008] 根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图；

[0009] 遍历所述流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本；

[0010] 接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块。

[0011] 本发明还提供了一种业务接口开发的系统，包括：

[0012] 流程图配置模块，用于监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数；接收被触发的控件的配置参数的值以生成待关联控件；根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图；

[0013] 流程图解析模块，用于遍历所述流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本；

[0014] 脚本执行模块，用于接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块。

[0015] 本发明通过调用预先设置的功能控件构建流程图，并为其配置相应的参数，并在构建好流程图后，对其解析并生成相应的可执行脚本；在执行脚本时，可以调用节点相应的预设代码块执行相应的功能，从而使得业务逻辑可以以流程图的形式可视化地展示，不但简化了业务开发的过程，也降低了业务开发人员的开发难度。

附图说明

[0016] 图1是本发明业务接口开发的方法较佳实施例的流程示意图；

[0017] 图2是本发明业务接口开发的方法中解析流程图，并生成可执行脚本的结构示意图；

[0018] 图3是本发明业务接口开发的方法中根据外部调用请求，执行相应的可执行脚本的流程示意图；

[0019] 图4是本发明业务接口开发的方法中以赠送礼品的业务为例所构建的流程图的结构示意图；

[0020] 图5是本发明业务接口开发的方法执行图4所示的流程图对应的可执行脚本的调用栈的结构示意图；

[0021] 图6是本发明业务接口开发的系统较佳实施例的结构示意图。

[0022] 本发明目的的实现、功能特点及优点将结合实施例，参照附图做进一步说明。

具体实施方式

[0023] 以下结合说明书附图及具体实施例进一步说明本发明的技术方案。应当理解，此处所描述的具体实施例仅仅用以解释本发明，并不用于限定本发明。

[0024] 本发明提供的业务接口开发的方法基于本发明设计的业务接口开发的系统而完成的，该业务接口开发的系统提供配置人员进行流程图构建的界面，待配置人员使用业务接口开发的系统所提供的功能控件构建流程图并对其进行参数配置后，业务接口开发的系统后台则将构建的流程图生成相应的可执行脚本，以供外部调用；而且业务接口开发的系统还提供功能控件相应的代码，以供可执行脚本执行时调用。

[0025] 为了方便后面描述，这里先对该业务接口开发的系统可供配置人员构建的流程图进行简单的描述。该构建的流程图主要由节点组成，而且节点根据不同的类型而分为以下几类：

[0026] 1．主流程根节点：供外部调用的入口，外部只能通过调用主流程根节点来执行流程图，这保证了逻辑的安全性；

[0027] 2．子流程根节点：子流程入口，子流程根节点都有对应的节点名，所有子流程的节点名不能重复，主流程可以通过指定子流程根节点名称来执行子流程，子流程也可以调用其它子流程；

[0028] 3．调用节点：调用节点会要求逻辑配置这指定子流程的根节点名和输入参数，如果一个流程图执行到调用节点时，他会转向指定的子流程，当子流程执行完毕后，返回继续执行主流程；

[0029] 4．逻辑节点：程序执行到逻辑节点时，会调用逻辑节点对应的功能组件代码，代码执行完后会返回一个执行状态(即返回值)，后台根据返回状态再决定接下来执行哪些节点；

[0030] 5．并发节点：当一个流程执行到并发节点时，会按随机顺序执行并发节点指向的所有节点。

[0031] 该业务接口开发的系统中，可以预先将常用的开发逻辑封装成功能控件，该功能

控件主要供配置人员调用，指定了调用的代码函数名和配置人员需要配置的参数以及所有可能的返回状态。该功能控件在流程图中相当于一个新的节点。返回状态是指功能控件执行完后的返回值，一个逻辑节点执行完后，系统将根据返回值获取下一个要执行的节点。流程图的节点之间可以通过预设的关联参数进行关联，从而生成相应的流程图，该预设的关联参数为流程图中箭头的参数，包括起始点参数及终止点参数。

[0032] 流程图配置的参数可包括：

[0033] 1．输入参数：指由调用端输入且贯穿整个流程图的参数，在流程图执行期间不会有变化，且该输入参数也不允许更改。但是如果主流程调用了子流程，主流程必须重新设定子流程相应的输入参数，当子流程执行结束并返回主流程时，主流程将恢复其原来的输入参数；

[0034] 2．输出参数：指将执行的信息返回给调用端的唯一途径且贯穿整个调用流程和所有子流程的参数，在流程图执行期间可以根据需要修改输出参数；

[0035] 3．缓存参数：缓存参数和输出参数一样，贯穿整个调用流程和所有子流程，在流程图执行期间可以根据需要修改缓存参数，但唯一不同的是流程图执行结束返回给调用端后，调用端不会收到缓存参数的信息；即缓存参数是同一个流程内不同节点之间数据传递的桥梁；

[0036] 4．配置参数：指调用功能控件时唯一的参数，调用该控件时必须具有其对应的配置参数，否则调用不成功；配置参数可以是配置人员设置的固定值，也可以设置某个输入参数或缓存参数的参数名对应的参数值；

[0037] 5．gotoif参数：指一种变形的配置参数，用于实现多个返回状态的配置参数。

[0038] 参照图1，本发明提供的业务接口开发的方法包括以下步骤：

[0039] 步骤S110、监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数；

[0040] 业务接口开发的系统提供了用户端可绘制流程图的控件，而且该控件可以根据不同的业务功能而进行分类，例如以超级QQ为例，该控件包括各种节点类型，例如主流程根节点、子流程根节点、调用节点、并发节点及逻辑节点，而逻辑节点还可以按照业务来分，例如鉴权、统计日志、赠送道具。流程图中不同类型的节点可以采用不同的形状或颜色加以区分。而且不同类型的节点均对应设置相应的参数名和描述，该参数名和描述是业务接口开发的系统中预先设置的，配置人员仅需根据描述配置相应的参数值。

[0041] 当用户端通过点击或拖取控件至界面以触发某控件后，根据该控件的类型，获取控件需要配置的参数名提示用户端进行配置。此时仅需输入对应参数名的参数值及参数值来源。该“参数值”和“参数值来源”共同表示参数名的取值。例如，“参数值来源”为“hard code”时，参数名对应的“参数值”表示配置人员配置的固定值；“参数值来源”为“from input”时，参数名对应的“参数值”则为某输入参数的参数名，其实际值为该输入参数的参数名所对应的“参数值”；“参数值来源”为“from buffer”时，参数名对应的“参数值”则为某缓存参数的参数名，其实际值为该缓存参数的参数名对应的“参数值”。

[0042] 步骤S120、接收被触发的控件的配置参数的值以生成待关联控件；

[0043] 根据被触发的控件所需的配置参数的提示信息，设置相应的参数值及参数来源，使得被触发的控件生成待关联控件。

[0044] 步骤S130、根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图；

[0045] 根据相邻两节点之间的箭头参数(关联参数)，为生成的待关联控件建立关联关系以生成流程图。在这里，当监测到相邻两节点之间的箭头连接有节点时，则自动将箭头参数设置为相应连接的节点的节点标识。

[0046] 步骤S140、遍历构建好的流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本。

[0047] 具体地，参照图2，该步骤S140包括：

[0048] 步骤S141、遍历所述流程图中的所有节点，获取每个节点的节点标识、节点类型；

[0049] 步骤S142、根据节点类型的不同，获取各个节点相应的节点信息；

[0050] 步骤S143、根据预设的代码模板，将各个节点的节点标识及节点信息赋值到所述代码模板中以生成可执行脚本。

[0051] 每个节点均具有其相应的标识，而且不同的节点其相应的节点信息也不同，如下：

[0052] (1)如果节点为主流程根节点或子流程根节点，由于其没有相应的配置参数，则获取该节点的节点名和下一个要执行的节点标识，如果下一个节点是并发节点，则还要获取并发节点所指向的标识组，并用分隔符分割，例如“|”；

[0053] (2)如果节点为逻辑节点，则获取该节点对应的代码函数名、配置参数、输入参数、缓存参数和不同返回值对应的下一个要执行的节点标识，如果下一个节点是并发节点，则还要获取并发节点所指向的标识组，并用分隔符分割，例如“|”；

[0054] (3)如果节点为调用节点，则获取该节点调用的子流程根节点的节点名、以及该子流程对应的所有输入参数，该子流程对应的所有输入参数将作为该调用节点的所有配置参数；

[0055] 获取完所有节点的节点标识及节点信息后，则根据预设的代码模板，将各个节点的节点标识及节点信息赋值到所述代码模板中以生成可供外部调用的可执行脚本。该可执行脚本将对外发布，且为其分配供外部调用的统一接口名。具体地，该预设的代码模板可根据不同类型的节点设置不同的代码模板。例如主流程根节点的代码模板为：

[0056] `<root id="X" name="XXX" list="XXX"/>`

[0057] 子流程根节点名的代码模板为：

[0058] `<subroot id="X" name="XXX" list="XXX"/>`

[0059] 逻辑节点的代码模板为：

[0060]
```
<node id="X" name="XXX">
    <params />      //该节点所有params属性的参数名和参数值
    <inputs/>       //该节点所有input属性的参数名和参数值
    <buffers/>      //该节点所有buffers属性的参数名和参数值
    <states/>       //该节点所有的返回状态值
</node>
```

[0061] 调用节点的代码模板为：

[0062]

```
<调用节点名 id="XX" name="XX">
    <params/>          //该节点所有params属性的参数名和参数值
    <inputs />         //该节点所有input属性的参数名和参数值
    <buffers/>         //该节点所有buffers属性的参数名和参数值
</调用节点名>
```

[0063] 步骤S150、接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块。

[0064] 当需要调用可执行脚本时，用户端可输入相应的参数及接口名以发起外部调用请求。接收到该外部调用请求后，根据请求中的接口名获取相应的可执行脚本，即找到根节点名与接口名一致的可执行脚本。然后对其进行解析并执行。在执行过程中，还要调用所执行的节点对应的预设代码块，并根据该节点相应的配置参数执行所述预设代码块，返回执行结果。

[0065] 本发明实施例通过调用预先设置的功能控件构建流程图，并为其配置相应的参数，并在构建好流程图后，对其解析并生成相应的可执行脚本；在执行脚本时，可以调用节点相应的预设代码块执行相应的功能，从而使得业务逻辑可以以流程图的形式可视化地展示，不但简化了业务开发的过程，也降低了业务开发人员的开发难度。

[0066] 上述步骤S150中主要利用调用栈和输入参数栈进行相应的可执行脚本的执行，调用栈的每个元素均保存了当前执行脚本行对应的节点的节点标识和状态，而且一个元素包括两种状态：未执行和已执行状态。参照图3，利用调用栈和输入参数栈执行可执行脚本的具体过程如下：

[0067] 步骤S151、接收外部调用请求，并根据所述外部调用请求创建调用栈和输入参数栈，同时将调用栈和输入参数栈初始化为空；

[0068] 步骤S152、根据外部调用请求的根节点名，获取该根节点名对应的可执行脚本及节点标识，若获取成功，则转入步骤S153；若获取失败，则结束流程；

[0069] 步骤S153、向调用栈压入该根节点的节点标识，且该节点的状态为未执行；同时，向输入参数栈压入调用请求的输入参数表；

[0070] 步骤S154、判断调用栈是否为空，是则转入步骤S158；否则转入步骤S155；

[0071] 步骤S155、判断调用栈栈顶节点的状态是否为已执行；是则转入步骤156；否则转入步骤S157；

[0072] 步骤S156、删除调用栈栈顶节点及其相应的输入参数栈栈顶参数，并转入步骤S154；

[0073] 步骤S157、执行栈顶节点对应的脚本行，并在其执行完后设置该栈顶节点的状态为已执行，并根据该栈顶节点的类型进行相应的操作；转入步骤S154；

[0074] 步骤S158、将输出参数表输出。

[0075] 上述步骤S157包括：

[0076] 若栈顶节点是主流程根节点，则先设置该栈顶节点的状态为已执行，再将该节点

的list属性的所有节点的节点标识压入调用栈，并设置其状态为未执行；

[0077] 若栈顶节点是子流程根节点，则先设置该栈顶节点的状态为已执行，再将该节点的list属性的所有节点的节点标识压入调用栈，并设置其状态为未执行；

[0078] 若栈顶节点是逻辑节点，则先获取该逻辑节点相应的配置参数表，调用该逻辑节点相应的预设代码，并利用该配置参数表中的参数执行该逻辑节点对应的脚本行；在获得一返回值后设置该逻辑节点的状态为已执行，再根据该返回值将其后续执行的节点的节点标识压入调用栈中，并设置其状态为未执行；

[0079] 若栈顶节点是调用节点，则先根据该调用节点所设置的调用的子流程根节点的节点名，将该子流程根节点压入调用栈并设置该子流程根节点的状态为未执行，再将获取的该调用节点相应的配置参数表压入输入参数栈，然后设置该调用节点的状态为已执行，并重复执行步骤S154-S159，在此不再赘述。

[0080] 上述获取逻辑节点相应的配置参数表及获取调用节点相应的配置参数表均可包括：

[0081] (1)、初始化配置参数表t为空；

[0082] (2)、将该节点的所有params属性的参数名和参数值都加入配置参数表t中；

[0083] (3)、将该节点的所有inputs属性参数名和实际配置参数值加入配置参数表t中，实际配置参数值是以inputs属性的参数名对应的参数值为key从输入参数栈栈顶元素的输入参数表中找到对应的输入参数值，作为实际配置参数值；

[0084] (4)、将该节点的所有buffers属性参数名和实际配置参数值加入配置参数表t中，实际配置参数值是以buffers属性的参数名对应的参数值为缓存参数的key找到对应的缓存参数值，作为实际配置参数值。

[0085] 另外，部分业务为了接口安全，需要检查流程图中的节点是否存在循环或者递归，因此通过设置栈元素的状态，而且在压入新的元素之前，判断流程图中的节点是否存在循环，即判断调用栈中是否存在相同节点标识的元素，并且该元素的状态被设置为已经执行，若是则表示结束流程，若否则压入新的元素，并将元素的状态设置未执行。

[0086] 下面以一个赠送礼品的业务接口程序的开发为例对本发明进行具体描述，该接口名为send，外部将用户的QQ号码作为输入参数，即参数名qq，接口逻辑则为：

[0087] 1、活动的有效期为2012-10-01 00:00:00至2012-10-20 00:00:00，如果接口未开启，输出resp=1，如果接口已经过期输出resp=2，如果接口有效，输出resp=0，并执行一下逻辑，否则结束流程调用；

[0088] 2、如果该QQ号码是1~3级超级QQ用户，赠送2张自动加速卡，如果该QQ号码是4~7级超级QQ用户，赠送4张自动加速卡，如果该QQ号码不是超级QQ用户，则不赠送；调用该业务接口后将输出赠送的自动加速卡张数给调用端(参数名为num)。

[0089] 首先，根据用户端的请求构建流程图，并建立流程图之间的关联。如图4所示，椭圆形节点为主流程根节点，是外部调用的唯一入口，且其节点标识为1；菱形和矩形节点均为逻辑节点，会执行对应功能组件的代码，而且菱形节点为判断功能，例如节点标识为34、4的节点，矩形节点为执行功能，例如节点标识为39、41、22、24的节点；圆角矩形节点为调用节点，如节点标识为11、13的节点；圆形节点为子流程根节点，如节点标识为15的节点；并行线为并发节点，如节点标识为18的节点。上述节点的节点标识均为随机生成，并保证不会重

复。节点之间通过箭头进行关联，该箭头可以用于指示节点之间的流程。

[0090] 然后，根据不同的节点需求的参数进行相应的配置：

[0091] (1)节点标识为1的节点是主流程根节点，由于其没有配置参数，所以不需要进行配置参数的设置；该节点通过箭头参数可知连接的下一节点的节点标识为34，则该节点标识为1的节点的属性list为34；

[0092] (2)节点标识为34的节点是判断有效期的逻辑节点，该节点要求配置的参数名为begin和end，相应的描述为开始时间(yy-mm-dd HH:MM:SS)和结束时间(yy-mm-dd HH:MM:SS)，则设置参数名begin和end的参数值和参数值来源，如表1所示：

[0093] 表1

[0094]

参数名	描述	参数值	参数值来源
begin	开始时间(yy-mm-dd HH:MM:SS)	2012-10-01 00:00:00	hard code
end	结束时间(yy-mm-dd HH:MM:SS)	2012-10-20 00:00:00	hard code

[0095] 另外，该节点还包括gotoif参数，本实施例将该参数的数量设置为3，即“未开始”、“是”、“已过期”3个返回状态，每个返回状态对应有返回值，即箭头所指向的节点标识。

[0096] (3)节点标识为4的节点是判断QQ的黄金等级的逻辑节点，该节点要求配置的参数名uin，相应的描述是QQ号码。本实施例中该参数名所设置的参数值为“qq”，参数值来源为“from input”，表示配置参数的值由输入参数名为qq的值替换；

[0097] 另外，这个节点还包括gotoif参数，本实例将该参数的数量设置为3，即“非超级QQ”、将超级QQ用户分成了两类的“[1,3]级超级QQ用户”和“[4,7]级超级QQ用户”3个返回状态；每个返回状态对应有返回值，即箭头所指向的节点标识。当然配置人员也可以将超级QQ用户随意的分成更多类，例如分成[1,2],[3,4],[5,7]三类。

[0098] (4)节点标识为11和13是用于调用子流程的调用节点，首先要配置调用的子流程的节点名，这里设置为“call”，另外调用节点会自动生成配置参数，该配置参数表就是子流程的输入参数，如表2所示：

[0099] 表2

[0100]

ID	参数名	描述	参数值	参数值来源
11	num	赠送自动加速卡的数量	2	hard code
	qq	QQ号码	qq	from input
13	num	赠送自动加速卡的数量	4	hard code
	qq	QQ号码	qq	from input

[0101] (5)节点标识为15的节点是子流程根节点，该节点不需要配置参数，只需要设置节点名，这里设置为“call”，与调用节点的节点名一致。

[0102] (6)节点标识为24的节点是赠送加速卡的逻辑节点，该节点要求配置的参数名中，type和active id配置为固定值，num为赠送数量，由调用节点call指定，所以设置参数值来源为“from input”；uin为QQ号码，要求调用节点将参数名qq作为输入参数输入，所以设置参数值来源为“from input”。如表3所示：

[0103] 表3

[0104]

参数名	描述	参数值	参数值来源
type	加速卡类型(1是自动卡,2是加倍卡)	1	hard code
active id	活动号	1222	hard code
num	赠送数量	num	from input
uin	QQ号码	qq	from input

[0105] (7)节点标识为39、41、9、22的节点都是将所有配置参数拷贝到输出参数的逻辑节点。由上可知节点标识为39的节点所返回的状态为“未开始”,故将该节点的输出参数resp设置为1;节点标识为41的节点所返回的状态为“已过期”,故将该节点的输出参数resp设置为2;节点标识为9的节点所返回的状态为“非超级QQ”,故将该节点的输出参数resp设置为0;节点标识为22的节点所返回的状态为各种级别的“超级QQ”,故将该节点的输出参数resp设置为0,且输出参数num设置为由调用节点call输入。

[0106] 然后,遍历每个节点后,则生成最终的可执行脚本,如下所示:

[0107] <allFlow>

[0108] <root ID="1" name="send" list="34"/> //节点标识为1的节点所生成的代码

[0109] <node ID="34" name="dateLimit"> //节点标识为34的节点所生成的代码

[0110]

```
<params end="2012-10-20 00:00:00" begin="2012-10-01 00:0
0:00"/>
<inputs/>
<buffers/>
<states isEnd="41" notBegin="39" true="4"/>
</node>
<node ID="41" name="Response"> //节点标识为41的节点
所生成的代码
```

<params resp="2"/>
<inputs/>
<buffers/>
<states/>
</node>
<node ID="39" name="Response"> //节点标识为39的节点所生成的代码
<params resp="1"/>
<inputs/>
<buffers/>
<states/>
</node>
<node ID="4" name="SuperQQLevel"> //节点标识为4的节点所生成的代码

[0111]

<params lev1="4,7,state8" lev0="1,3,state7"/>
<inputs uin="qq"/>
<buffers/>
<states notSqq="9" state7="11" state8="13"/>
</node>
<node ID="9" name="Response"> //节点标识为9的节点所生成的代码
<params num="0" resp="0"/>
<inputs/>
<buffers/>
<states/>
</node>
<call ID="11" name="call"> //节点标识为11的节点所生成的代码
<params num="2"/>
<inputs qq="qq"/>

```
        </call>
        <call ID="13" name="call">        //节点标识为13的节点所生成
    的代码
            <params num="4"/>
            <inputs qq="qq"/>
            <buffers/>
        </call>
        <node ID="22" name="Response">        //节点标识为22的节点
    所生成的代码
            <params resp="0"/>
            <inputs num="num"/>
            <buffers/>
            <states/>
        </node>
        <node ID="24" name="speedCard">        //节点标识为24的节点
    所生成的代码
            <params ID="1222" type="1"/>
            <inputs num="num" uin="qq"/>
            <buffers/>
            <states/>
        </node>
        <subroot ID="15" name="call" list="22|24"/>      //节点标识为15
    的节点所生成的代码
    </allFlow>
```

[0112]

[0113] 生成上述可执行脚本后，则将其发布，且供外部调用的接口名为send。因此，通过业务的调用请求即可可调用该可执行脚本。假设一调用该可执行脚本的外部请求的时间在有效期内，且输入的QQ号码对应的黄金等级是2级，则后台执行该可执行脚本时使用的调用栈的变化如图5所示，该调用栈中，已执行的节点与未执行的节点将通过相应的标识进行区分，例如已执行的节点标识为灰色。该实施例中，输入参数栈最多只有两个元素，即{qq=771885922, num=2}和{qq=771885922}。当调用栈节点1入栈后，{qq=771885922}也压入输入参数栈。节点1、34、4、11使用的输入参数是{qq=771885922}。当调用节点11设置为已执行后，向输入参数栈压入元素{qq=771885922, num=2}，之后调用栈执行的节点15、22、24使用

的输入参数是{qq=771885922, num=2}。当子流程根节点15出栈时，输入参数也随之删除栈顶元素{qq=771885922,num=2}，后面执行的节点1、34、4、11又恢复原先的输入参数表{qq=771885922}。

[0114] 上述业务接口开发的系统还提供了调试功能，即生成可执行脚本后，还可以包括：接收调试请求，解析所述调试请求相应的可执行脚本，执行该脚本并在执行成功后的脚本行或脚本行对应的流程图节点添加执行标识。

[0115] 用户端的调试请求中包括需要调试的起始节点信息，例如起始节点名或节点标识。根据用户端的调试请求，则获取调试的起始节点，获取成功后，将其压入调用栈，并将调试请求的输入参数表压入输入参数栈中，并执行栈顶节点对应的脚本行。在执行的同时，还判断调用栈中各节点的状态是否为已执行，是则对应添加执行标识，例如对流程图的对应节点进行染色等。以便调试人员根据流程图的节点标识情况，而有针对性地对流程图进行修改。

[0116] 同时，上述业务接口开发的系统还提供了统计功能，即将所有调用的输入和调用流程图所执行过的节点标识按照执行顺序生成日志，该生成的日志格式为：外部传进来的输入参数表(多个时用#分隔)|执行顺序(用#分隔)|输出参数表(多个时用#分隔)。例如执行完上述send的外部调用请求后，即获得统计日志为：

[0117] uin=771885922|1#34#4#11#15#22#24|resp=0#num=2

[0118] 通过上述生成的日志，配置人员按照节点标识的调用顺序，就可以查询任何输入的执行结果或者任何节点的流量。

[0119] 参照图6，提出本发明一种业务接口开发的系统，其包括：

[0120] 流程图配置模块110，用于监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数；接收被触发的控件的配置参数的值以生成待关联控件；根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图；

[0121] 流程图解析模块120，用于遍历所述流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本；

[0122] 脚本执行模块，用于接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块。

[0123] 具体地，该流程图配置模块110可提供流程图构建的控件，以及相应控件所需配置的参数。配置人员则可以调用该控件，并为调用的控件设置相应的参数值，从而将所调用的控件通过预设的关联参数生成流程图。然后流程图解析模块120则将形成的流程图进行解析，生成可供外部调用的可执行脚本。关于流程图解析模块120对流程图的具体解析过程可参照前面方法所述。当需要调用可执行脚本时，用户端可输入相应的参数及接口名以发起外部调用请求。脚本执行模块130接收到该外部调用请求后，则根据请求中的接口名获取相应的可执行脚本，即找到根节点名与接口名一致的可执行脚本。然后对其进行解析并执行。在执行过程中，还要调用所执行的节点对应的预设代码块，并根据该节点相应的配置参数执行所述预设代码块，返回执行结果。

[0124] 本发明实施例通过调用预先设置的功能控件构建流程图，并为其配置相应的参数，并在构建好流程图后，对其解析并生成相应的可执行脚本；在执行脚本时，可以调用节

点相应的预设代码块执行相应的功能，从而使得业务逻辑可以以流程图的形式可视化地展示，不但简化了业务开发的过程，也降低了业务开发人员的开发难度。

[0125] 上述业务接口开发的系统还包括：

[0126] 调试模块130，接收调试请求，解析所述调试请求相应的可执行脚本，执行该脚本并在执行成功后的脚本行或脚本行对应的流程图节点添加执行标识。

[0127] 用户端的调试请求中包括需要调试的起始节点信息，例如起始节点名或节点节点标识。根据用户端的调试请求，则获取调试的起始节点，获取成功后，将其压入调用栈并将调试请求的输入参数表压入输入参数栈中，并执行栈顶节点对应的脚本行。在执行的同时，调试模块140还判断调用栈中各节点的状态是否为已执行，是则对应添加标识，例如对流程图的对应节点进行染色等。以便调试人员根据流程图的节点标识情况，而有针对性地对流程图进行修改。

[0128] 该业务接口开发的系统提供的调试功能使得调试入口可以为流程图中的任何节点，而且在调试过程中，将对执行成功的节点添加标识，从而使得调试人员可以快速地找到调试失败的问题。

[0129] 上述业务接口开发的系统还包括：

[0130] 脚本执行模块140，用于在可执行脚本执行期间，将所有调用的输入和调用流程图所执行过的节点的节点标识按照执行顺序生成日志。

[0131] 上述业务接口开发的系统还提供了统计功能，即在可执行脚本执行期间，脚本执行模块140同时将所有调用的输入和调用流程图所执行过的节点的节点标识按照执行顺序生成日志，该生成的日志格式为：外部传进来的输入参数表(多个时用#分隔)|执行顺序(用#分隔)|输出参数表(多个时用#分隔)。例如执行完上述send的外部调用请求后，即获得统计日志为：

[0132] uin=771885922|1#34#4#11#15#22#24|resp=0#num=2

[0133] 通过上述生成的日志，配置人员按照节点标识的调用顺序，就可以查询任何输入的执行结果或者任何节点的流量。

[0134] 一般，大型的业务通常都需要知道每个用户在后台代码的调用记录，以此来做数据分析。因此，通过在可执行脚本的执行过程中，自动将执行过的节点的节点标识按执行顺序生成日志，数据统计人员可以根据该生成的日志统计用户调用接口的执行情况。

[0135] 以上所述仅为本发明的优选实施例，并非因此限制其专利范围，凡是利用本发明说明书及附图内容所作的等效结构或等效流程变换，直接或间接运用在其他相关的技术领域，均同理包括在本发明的专利保护范围内。

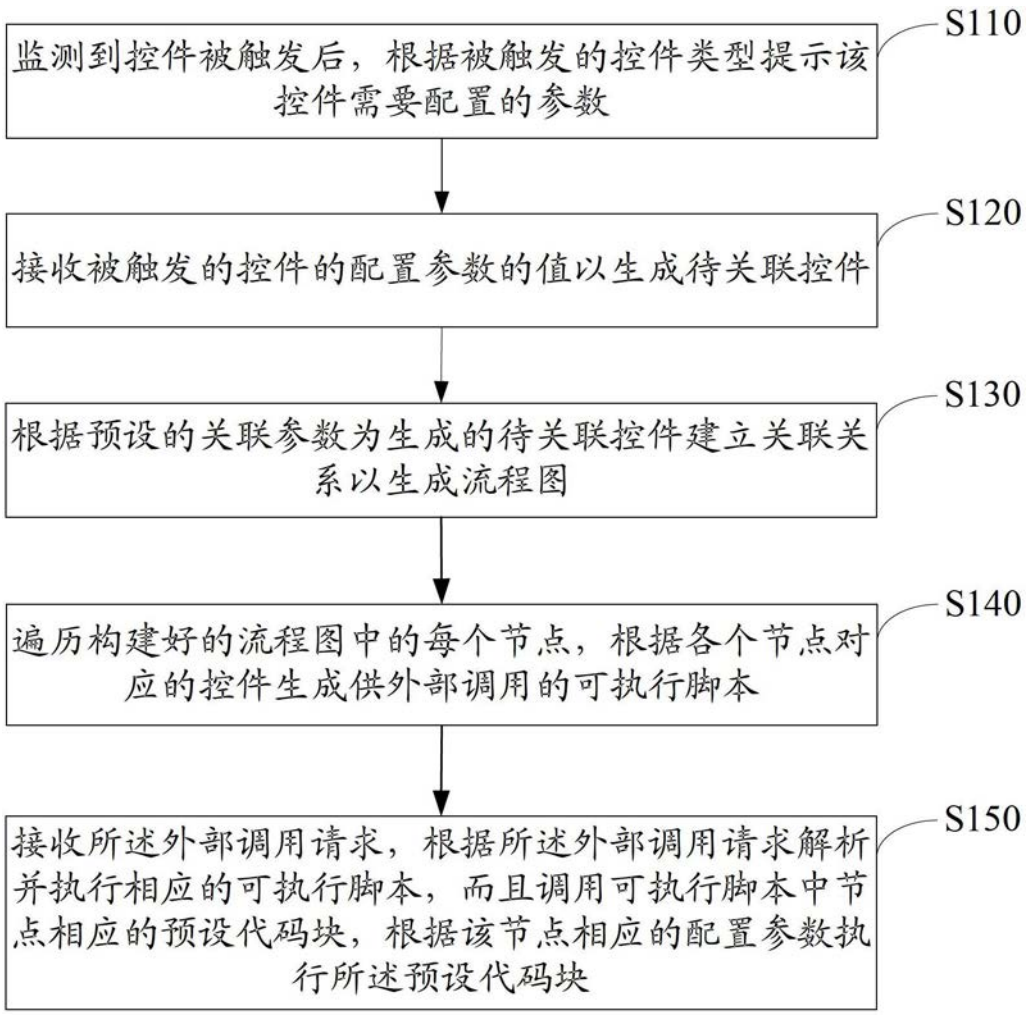

S110
监测到控件被触发后，根据被触发的控件类型提示该控件需要配置的参数
S120
接收被触发的控件的配置参数的值以生成待关联控件
S130
根据预设的关联参数为生成的待关联控件建立关联关系以生成流程图
S140
遍历构建好的流程图中的每个节点，根据各个节点对应的控件生成供外部调用的可执行脚本
S150
接收所述外部调用请求，根据所述外部调用请求解析并执行相应的可执行脚本，而且调用可执行脚本中节点相应的预设代码块，根据该节点相应的配置参数执行所述预设代码块

图1

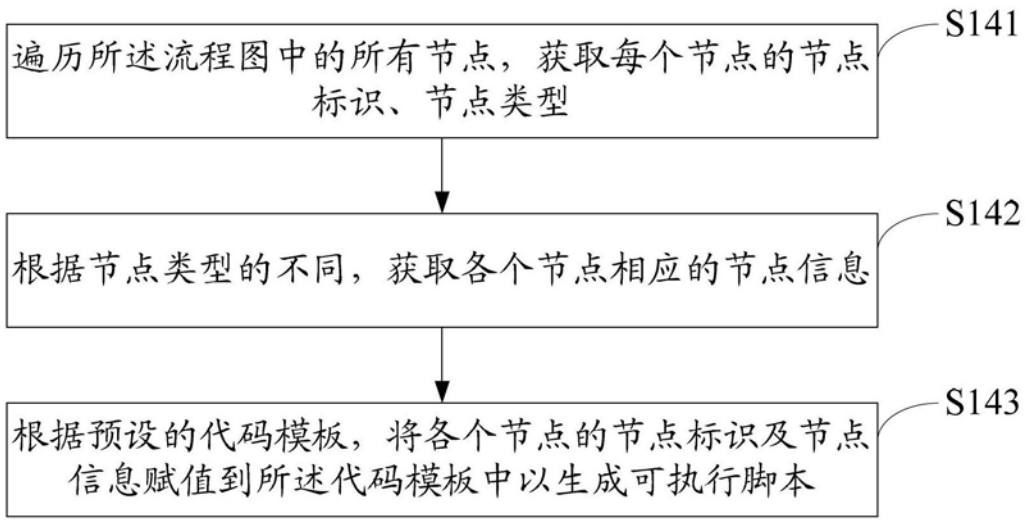

S141
遍历所述流程图中的所有节点，获取每个节点的节点标识、节点类型
S142
根据节点类型的不同，获取各个节点相应的节点信息
S143
根据预设的代码模板，将各个节点的节点标识及节点信息赋值到所述代码模板中以生成可执行脚本

图2

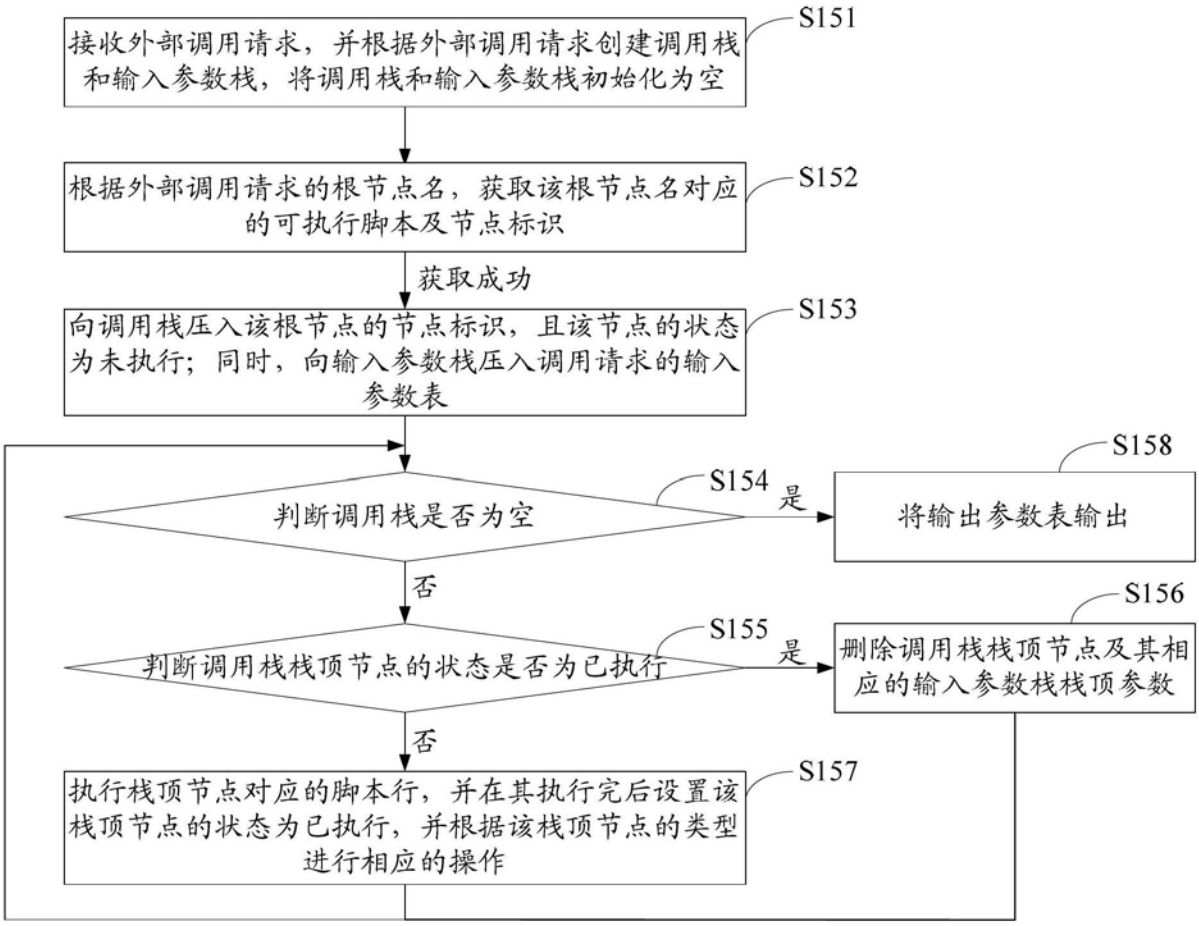

接收外部调用请求，并根据外部调用请求创建调用栈和输入参数栈，将调用栈和输入参数栈初始化为空
S151
根据外部调用请求的根节点名，获取该根节点名对应的可执行脚本及节点标识
S152
获取成功
向调用栈压入该根节点的节点标识，且该节点的状态为未执行；同时，向输入参数栈压入调用请求的输入参数表
S153
判断调用栈是否为空
S154
是
将输出参数表输出
S158
否
判断调用栈栈顶节点的状态是否为已执行
S155
是
删除调用栈栈顶节点及其相应的输入参数栈栈顶参数
S156
否
执行栈顶节点对应的脚本行，并在其执行完后设置该栈顶节点的状态为已执行，并根据该栈顶节点的类型进行相应的操作
S157

图3

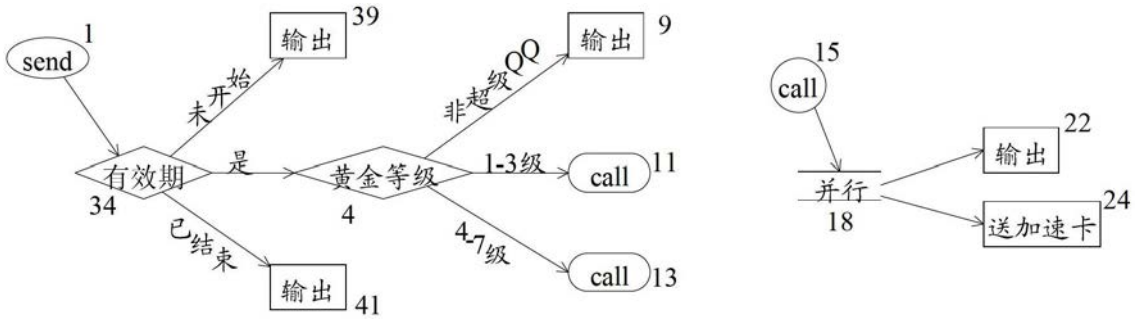

send
1
输出
39
未开始
有效期
34
是
黄金等级
4
非超QQ级
输出
9
1-3级
call
11
4-7级
call
13
已结束
输出
41
call
15
并行
18
输出
22
送加速卡
24

图4

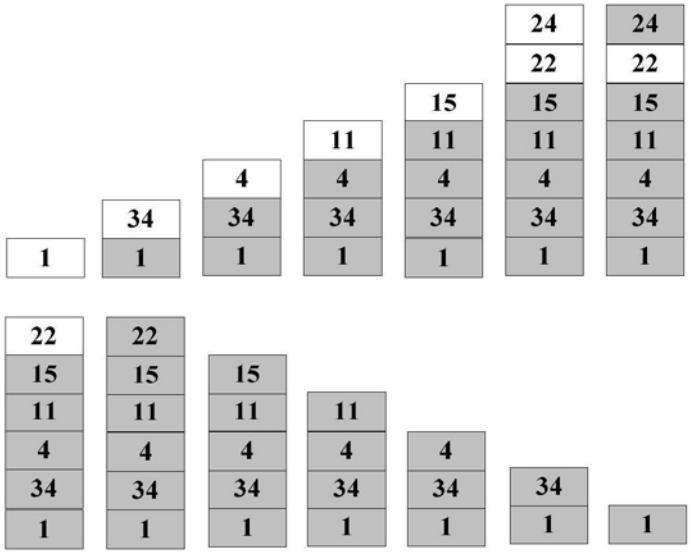

1
34
1
4
34
1
11
4
34
1
15
11
4
34
1
24
22
15
11
4
34
1
24
22
15
11
4
34
1
22
15
11
4
34
1
22
15
11
4
34
1
15
11
4
34
1
11
4
34
1
4
34
1
34
1
1

图5

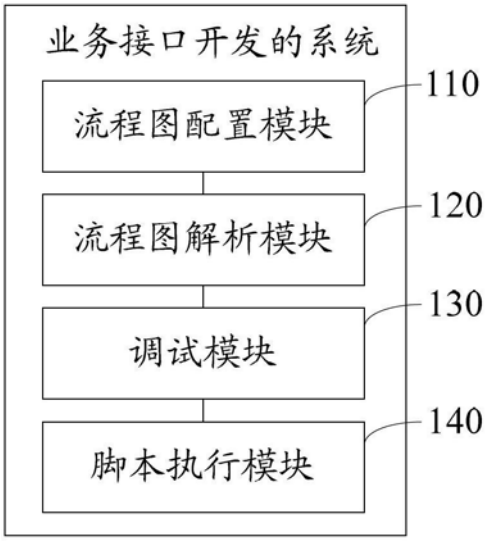

业务接口开发的系统
流程图配置模块
110
流程图解析模块
120
调试模块
130
脚本执行模块
140

图6