

(12) 发明专利

(10) 授权公告号 CN 101416156 B

(45) 授权公告日 2013. 03. 06

(21) 申请号 200780012353. 1

(22) 申请日 2007. 02. 21

(30) 优先权数据

11/393, 093 2006. 03. 30 US

(85) PCT申请进入国家阶段日

2008. 10. 06

(86) PCT申请的申请数据

PCT/US2007/004642 2007. 02. 21

(87) PCT申请的公布数据

W02007/120391 EN 2007. 10. 25

(73) 专利权人 微软公司

地址 美国华盛顿州

(72) 发明人 D·舒克拉 B·施米特 M·梅达

N·塔尔伯特 A·J·沙加 K·拉曼

(74) 专利代理机构 上海专利商标事务所有限公

司 31100

代理人 陈斌

(51) Int. Cl.

G06F 9/44 (2006. 01)

(56) 对比文件

US 20030144891 A1, 2003. 07. 31, 全文.

CN 1636207 A, 2005. 07. 06, 全文.

EP 0697652 A1, 1996. 02. 21, 全文.

US 20030018508 A1, 2003. 01. 23, 全文.

Claus Hagen 等. Exception Handling

in Workflow Management Systems.

《IEEE TRANSACTIONS ON SOFTWARE

ENGINEERING》. 2000, 第 26 卷 (第 10

期), 943-957.

审查员 张会

权利要求书 2 页 说明书 9 页 附图 13 页

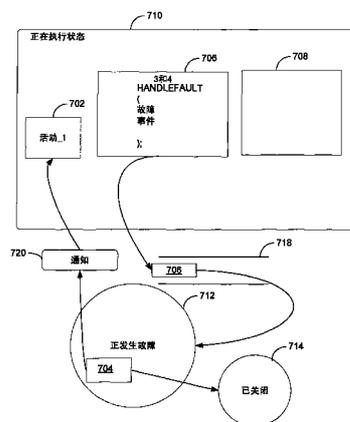
(54) 发明名称

异步处理故障事件的方法及系统

(57) 摘要

工作流的异步故障处理。定义了工作流中活动的状态自动机。状态自动机至少包括正在执行状态、正发生故障状态和已关闭状态, 并对活动的执行生存周期分类。将活动定义为包括工作项目, 且包括工作项目的执行分层结构。每一工作项目包括用于执行活动的一部分的操作。每一工作项目被转移到正在执行状态。转移的工作项目所包括的操作在正在执行状态中执行。根据执行分层结构和所包括的操作, 响应于故障事件标识所转移工作项目中的一个或多个。通过将一个或多个所标识的工作项目转移到正发生故障状态同时执行其余转移工作项目所包括的动作来异步处理故障事件。

CN 101416156 B



1. 一种用于为工作流的活动异步处理故障事件(722)的方法,所述方法包括:

为活动(702)定义状态自动机(600),所述状态自动机(600)至少包括正在执行状态(604)、正发生故障状态(608)和已关闭状态(612),所述状态自动机(600)对活动(702)的执行生存周期分类;

将所述活动(702)定义为包括多个工作项目(422),所定义的活动(702)具有用于多个工作项目(422)的执行分层结构(500),每一工作项目(422)包括用于执行活动(702)的一部分的操作;

将每一工作项目(422)转移到正在执行状态(604);

执行处于正在执行状态(604)中的转移工作项目(422)所包括的操作;

根据所述执行分层结构(500)和所包括的操作,响应于所述故障事件(722)标识所转移工作项目中的一个或多个;以及

通过将所述一个或多个标识工作项目(422)转移到所述正发生故障状态(608),同时执行未响应于故障事件(722)标识的其余转移工作项目所包括的操作来异步处理所述故障事件(722)。

2. 如权利要求1所述的方法,其特征在于,异步处理所述故障事件(722)包括调用所述一个或多个标识的工作项目(422)中的故障处理操作。

3. 如权利要求2所述的方法,其特征在于,调用所述故障处理操作包括使所述一个或多个所标识工作项目入队到调度器队列(718)中。

4. 如权利要求1所述的方法,其特征在于,还包括根据所述活动(702)的执行分层结构(500)将通知(720)从所述一个或多个所标识的工作项目(422)传播到所述其余的转移工作项目(422),所述通知(720)指示所标识的一个或多个工作项目(422)正处于正发生故障状态(608)。

5. 如权利要求4所述的方法,其特征在于,还包括响应于所传播的通知(720)将其余转移的工作项目(422)从所述正在执行状态(604)转移到所述正发生故障状态(608)。

6. 如权利要求1所述的方法,其特征在于,还包括抑制通知(720)对其余转移的工作项目(422)的传播,所述通知(720)指示所标识的一个或多个工作项目正处于正发生故障状态。

7. 如权利要求1所述的方法,其特征在于,还包括响应于所述一个或多个所标识的工作项目(422)被转移到所述正发生故障状态(608)向用户提供用于故障后处理的操作。

8. 一种用于为工作流的活动异步处理故障事件(722)的系统,所述系统包括:

为活动(406)定义状态自动机(600)的状态机(902),所述状态自动机(600)至少包括正在执行状态(604)、正发生故障状态(608)和已关闭状态(612),所述状态自动机(600)对活动(406)的执行生存周期分类;

将所述活动(406)定义为包括多个工作项目(422)的活动组件(904),所定义的活动(406)具有用于多个工作项目(422)的执行分层结构(500),每一工作项目(422)包括用于执行活动(406)的一部分的操作;

将每一工作项目(422)转移到正在执行状态(604)的调度器组件(906);

执行处于正在执行状态(604)中的转移工作项目(422)所包括的操作的执行组件(908);

根据所述执行分层结构(500)和所包括的操作,响应于故障事件(722)标识所转移工作项目中的一个或多个的标识组件(910);以及

通过将所述一个或多个标识工作项目(422)转移到所述正发生故障状态(608),同时执行未响应于故障事件(722)标识的其余转移工作项目所包括的操作来异步处理所述故障事件(722)的故障处理器(912)。

9. 如权利要求8所述的系统,其特征在于,所述故障处理器(912)使所述一个或多个所标识的工作项目(422)入队到调度器队列(718)中以将所述一个或多个所标识的工作项目转移到所述正发生故障状态(608)。

10. 如权利要求8所述的系统,其特征在于,还包括根据所述活动(406)的执行分层结构(500)将通知(720)从所述一个或多个所标识的工作项目(422)传输到所述其余的转移工作项目(422)的故障传播组件(914),所述通知(720)指示所标识的一个或多个工作项目(422)正处于正发生故障状态(608)。

11. 如权利要求10所述的系统,其特征在于,还包括响应于所传输的通知(720)将其余转移的工作项目从所述正在执行状态(604)转移到所述正发生故障状态(608)的转移组件(916)。

12. 如权利要求8所述的系统,其特征在于,根据所述异步处理故障事件(608)恢复与所述活动(406)相关联的数据的补偿组件(918)。

13. 如权利要求8所述的系统,其特征在于,还包括抑制通知(720)对其余转移的工作项目的传输的禁止组件(920),所述通知(720)指示所标识的一个或多个工作项目正处于正发生故障状态。

## 异步处理故障事件的方法及系统

### [0001] 背景

[0002] 面向过程或过程中心的程序已发展成能够处理对现实事件建模的复杂指令。过程中心的程序反映了现实过程并反映了现实实体之间的交互。现有的系统通过对商业问题建模来试图将商业问题映射为高级 workflow。然而,现实 workflow 在各个方面有所不同,诸如 (a) 执行和建模的复杂性、(b) 在设计时对流程的结构的了解、(c) 静态定义或自组织/动态的、(d) 在其生命周期的各个时间创作和编辑流程的容易程度以及 (e) 商业逻辑与核心 workflow 过程之间弱或强的关联。现有的模型不能适应所有这些因素。

[0003] 此外,大多数现有 workflow 模型是基于以下中任一:基于语言的方法(例如, BPEL4WS、XLANG/S 以及 WSFL)或基于应用程序的方法。基于语言的方法是具有帮助向用户/程序员对 workflow 过程建模的预定义构造的闭集的高级 workflow 语言。workflow 语言携带允许用户构建 workflow 模型的构造的闭集的所有语义信息。然而,语言不是可由开发人员扩展的,且表示构成 workflow 模型的原语的闭集。语言被绑定至由 workflow 系统厂商装运的语言编译器。仅 workflow 系统产品厂商可通过用产品将来版本中的一组新构造扩展语言来扩展模型。这通常要求升级与该语言相关联的编译器。此外,语言通常不会声明性地展示或定义可由其它程序容易且高效使用的功能或操作。

[0004] 基于应用程序的方法是在应用程序内具有解决域专用问题的 workflow 能力的应用程序。这些应用程序不是真正可扩展的,它们也不具有可编程模型。

[0005] 此外,采用现有的方法,复杂性、预知、动态 workflow、创作容易程度以及与业务逻辑和核心 workflow 的关联的强度的问题未被充分解决。不存在构建可视 workflow 设计器以对不同类的工作流建模的可用的可扩展、可定制且可重新宿主的工作流设计器框架。现有的系统缺乏快速应用程序开发(RAD)样式的工作流设计体验,该体验允许用户用图形设计 workflow 过程并用开发人员选择的编程语言关联业务逻辑。

[0006] 而且,workflow 过程横跨 workflow 过程模型多步处理交叉切割互不相关且纠缠的问题。例如,尽管 workflow 过程的一部分被设计为参与长期运行的事务,而同一过程的其它部分被设计用于并发执行或访问共享资源。由于设计缺陷,现有的系统不能提供执行线程的交错,而线程的交错允许用户设计活动的同步或交错执行。同一 workflow 过程的另外其它部分要求跟踪,而其它部分处理商业或应用级异常。需要对 workflow 过程的一个或多个部分应用某些行为。

[0007] 某些 workflow 建模方法是不实际的,因为它们要求对整个商业过程包括异常和人类干预在内的完整的基于流的描述。这些方法中的某些在异常发生时提供附加的功能,而其它方法排他地采用基于约束的方法而非基于流的方法来对商业过程建模。现有的系统或者实现基于流的方法或者实现基于约束的方法。这样的系统对于对众多常见的商业情形建模而言太不灵活。这些系统也缺乏异步处理异常或取消的能力。

### [0008] 概述。

[0009] 本发明的实施例通过在定义 workflow 中的活动的执行生存周期的状态自动机中具有正发生故障状态来允许异步故障或异常处理。通过具有正发生故障状态,本发明的各方

面允许开发人员或程序声明性地设计用于异常或故障处理的程序,使得程序或活动的部分可在正发生故障状态中进行故障处理,而程序或活动的其它部分可不受异常或故障事件的影响。

[0010] 本发明的替换实施例允许传播或传输故障处理的通知。在其它替换实施例中,可抑制或禁止这样的通知传播或传输。此外,另外的实施例响应于来自用户的用于对处理故障后或异常后操作的输入。

[0011] 提供本概述以便以简化的形式介绍将在以下详细描述中进一步描述的一些概念。该概述不旨在标识所要求保护的的主题的关键特征或必要特征,也不旨在用于帮助确定所要求保护的的主题的范围。

[0012] 其它特征的一部分将是显而易见的,一部分将在下文中指出。

[0013] 附图简述

[0014] 图 1 是示出现有编程范例的框图。

[0015] 图 2 是示出根据本发明的实施例的工作流设计框架的虚拟化的示例性框图。

[0016] 图 3 是示出根据本发明的实施例的示例性工作流的示例性示意图。

[0017] 图 4 是示出根据本发明的实施例的用于处理工作流活动的系统的示例性计算环境的示意图。

[0018] 图 5 是示出根据本发明的实施例的工作流活动的分层结构的示意图。

[0019] 图 6 是示出根据本发明的实施例描述活动的执行生存周期的示例性状态自动机的示意图。

[0020] 图 7A 到 7E 是示出根据本发明的实施例对工作流的故障事件异步处理的框图。

[0021] 图 8 是示出根据本发明的实施例用于对工作流的活动的故障事件进行异步处理的方法的流程图。

[0022] 图 9 是示出可在其上存储本发明的各方面的示例性计算机可读介质的框图。

[0023] 附图 A 示出根据本发明的实施例声明性提出异常的示例性实现。

[0024] 相应的参考字符在各附图中指示相应的部分。

[0025] 详细描述

[0026] 首先参考图 1,框图示出了用于对诸如工作流的过程中心活动设计程序的现有的编程范例。例如,示意图示出现有的程序范例的三级虚拟化模型,托管执行环境级是最高级,处理单元是最低级。在该编程设计系统中,即使在托管执行环境级处,程序尤其是处理工作流过程的过程中心程序,缺乏适应工作流中各过程之间的复杂交互的能力和效率。

[0027] 本领域的技术人员已知,某些约束与设计软件或应用程序相关联。在这些示例中,当编写操作系统软件程序 104 时,程序代码或例程是根据处理单元 102 的类型或配置的、对一类计算体系结构(例如,IBM®可兼容、APPLE®计算机或其它系统)专用或其它约束。此外,编程语言通常需要准确标识和利用数据结构,诸如堆栈、堆、线程库或其它硬件专用结构以便操作系统 104 能正确运作。

[0028] 当处理复杂工作流过程时,现有应用程序使用托管执行环境 106(例如,其中程序可共享功能或通用的面向对象类的运行时环境)的概念,其中由一种编程语言编写的程序可调用用不同的编程语言编写的其它程序中的功能。在这样的执行环境中,采用不同编程语言的这些程序被编译成中间语言,使得托管执行环境 106 可展示不同程序的参数、自变

量或模式或功能,使得程序可彼此交互。

[0029] 尽管该执行环境 106 在程序之间创建通用的通信环境,但执行环境 106 包括可能不适于处理过程中心程序的复杂程度和容量的各种严格要求。例如,执行环境 106 要求程序被确认为特定文件格式。执行环境 106 也要求程序中的功能或操作使用由执行环境 106 定义的固定的一组功能或一类功能。

[0030] 本发明的实施例在图 2 中的可扩展基础或框架 202 上构建,以克服现有编程模型的缺点。通过允许以任何编程语言编写并以任何文件格式编排程序,本发明的各方面允许程序开发人员设计具有特定功能的程序,而不损害其功能和细节。通过将诸如 workflow 任务或过程的活动定义为要在 workflow 框架中执行的基类,开发人员可容易且高效地构建域专用(例如,诸如卫生保健行业、金融行业等中的程序的特定执行环境)的操作代码(后文中称为“操作代码”),而无需依附现有执行环境中的刚性、硬编码、不灵活且固定的一组功能或活动类。此外,体现本发明各方面的工作流基础是层叠在任何现有框架(例如,托管执行环境、操作系统环境或硬件处理单元级)上方的基于延续的运行环境。

[0031] 本发明的各方面通过允许按照任何方式或表示(例如,流程图、示意图、编号描述等)进行 workflow 设计免除按照特定文件格式定义活动的约束,只要 workflow 中的活动可根据 workflow 设计的表示构造。

[0032] 此外,workflow 框架或基础能够处理从较低级(例如,OS)提出的故障或异常,或提出以其它格式(例如,中间语言)编写的功能的异常。

[0033] 图 3 示出根据本发明的实施例的 workflow 300 的简单视图。例如,workflow 300 可以是用于处理购买定单的工作流,且该购买定单 workflow 300 可包括诸如接收购买定单、向顾客发送确认、由管理员批准购买定单等的过程或活动。

[0034] workflow 300 可从起始点 302 开始。例如,购买定单 workflow 的起始点 302 可以从顾客接收定单。workflow 300 也可包括条件语句 304(诸如“IF 语句”或“WHILE 语句”),它可被细分成附加条件语句 306 和 308。workflow 300 也可包括并行结构 310,后者进一步包括一个或多个序列或活动 312。例如,并行结构 310 包括诸如检查存货并更新可用发货者的并行处理的活动。在所示示例中,诸如“发送电子邮件”和“获得批准”的活动可并行处理。在“此处丢弃活动”316,用户可进一步将更多活动添加或补充到 workflow 300 中。为了完成 workflow 300,过程或活动将在完成步骤或点 314 结束。

[0035] 在一个实施例中,活动可被分层安排成树结构(见图 5)500 或其它执行序列。例如,活动可以是合成活动,其中活动包括与之相关联的一个以上的工作项目。在另一实施例中,活动的集合可以是合成活动。活动方法或操作可位于具有两个子或叶节点 504 和 506 的根节点 502 中。子节点 504 和 506 中的活动方法或操作(例如,分别为工作项目\_1 和工作项目\_2)可根据分层结构执行。此外,子节点 504 和 506 也可包括具有要执行的相应工作项目的其它子节点。

[0036] 在另一实施例中,活动包括以下类型中的一个或多个:简单活动、容器活动和根活动。在此实施例中,在模型中有一个根活动,根活动内有零个或任何数量的简单活动或容器活动。容器活动可包括简单或容器活动。整个 workflow 过程可用作构建较高级 workflow 过程的活动。此外,活动可以是可中断或不可中断的。不可中断的合成活动不包括可中断活动。不可中断活动没有可使活动阻塞的服务。

[0037] 而且,当执行活动以及包括在活动中的工作项目时, workflow 框架或执行上下文或环境为每一工作项目定义范围或边界。该范围或边界包括并展示了诸如要由工作项目访问的共享数据或资源、相关联的属性、处理程序、约束以及自治代理之间的交互等的信息(例如,以数据、元数据等的形式)。而且,每一活动可由采用任何编程语言的用户代码配置。例如,用户代码可表示在特定域或执行环境中编写的业务或应用程序逻辑或规则。每一活动可支持对用户代码中的执行的截取前挂钩和截取后挂钩。每一活动具有相关联的运行时环境执行语义和行为(例如,状态管理、事务、事件处理和异常处理)。活动可与其它活动共享状态或资源。此外,活动可以是原语活动,或归组成合成活动。原语或基本活动不具有子结构(例如,子活动)且因此是树结构中的叶节点。合成活动包含子结构(例如,它是一个或多个子活动的父活动)。

[0038] 图 4 是示出根据本发明的实施例的用于处理 workflow 活动的系统 400 的示意图。系统 400 包括处理器 402,它可以是处理单元或处理单元的集合。系统 400 也包括用于存储可由处理器 402 访问的数据的存储或存储器区 404。在一个实施例中,系统 400 可以是具有一个或多个处理器或处理单元(例如,处理器 402)以及系统存储器(例如,存储器区 404)并具有将包括系统存储器的各种系统组件耦合到处理器 402 的其它组件的计算机。

[0039] 在一个示例中,存储器区 404 可包括计算机可读介质(易失性、非易失性、可移动或不可移动介质),它们以用于存储诸如计算机可读指令、数据结构、程序模块或其它数据这样的信息的任何方法或技术来实现。例如,计算机存储介质包括 RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光盘存储、磁带盒、磁带、磁盘存储或其他磁存储设备、或可以用于存储所需信息并可由系统 400 访问的任何其它介质。存储器 404 也包括具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据的通信介质,且包含任何信息传递介质。本领域技术人员熟悉已调制数据信号,其一个或多个特征以将信息编码在该信号中的方式来设置与改变。诸如有线网络或直接线连接等有线介质,以及如声学、RF、红外线及其它无线介质等无线介质都是通信介质的示例。以上的任一组合也包括在计算机可读介质的范畴内。

[0040] 例如,存储器区 404 存储用于在 workflow(例如,workflow 300)中处理的多个活动 406。多个活动 406 中的每一个包括一个或多个工作项目,且工作项目可在诸如树结构(见图 5)的分层结构中组织。当处理多个活动 406 时,处理器 402 访问或执行调度器 408,它被配置成设置有组织的活动集。

[0041] 例如,处理器 408 经由组件或一组计算机可执行指令诸如调度器 408 访问多个活动 406 中的工作项目以使工作项目 422 入队或存储到队列 410。可由处理器 402 访问的分派器 412 分派工作项目 422 以便执行。例如,工作项目 422-1 可包括活动方法或活动操作 424、例程或用于执行“向用户请求输入”的功能的代码的集合。一个或多个其它活动方法、活动操作、例程或代码可被包括在每一工作项目 422 中,而不背离本发明的范围。

[0042] 一旦由分派器 412 分派了工作项目 422,处理器 402 在 414 执行工作项目 422 中的每一方法 424。在工作项目 422-1 的示例中,处理器 402 可允许用户经由用户界面(UI)输入所请求的信息或数据。在另一实施例中,处理器 402 可连接或访问外部数据源以便向用户请求输入。在完成活动方法或活动操作 424 之后,处理器 402 在 416 结束工作项目 422 的执行。在一个实施例中,处理器 402 在 418 将工作项目的正在执行状态钝化(passivate)

到数据存储 420。

[0043] 在另一实施例中,处理器 402 根据诸如图 6 中所示的自动机的状态自动机执行工作项目 422,图 6 是示出根据本发明的实施例描述与活动相关联的工作项目的处理状态的示例性状态自动机 600 的示意图。在一个实施例中,状态自动机 600 定义活动的执行生存周期。在一个示例中,状态自动机 600 可包括已初始化状态、正在执行状态以及已关闭状态(如图 4 中所示)。在另一实施例中,状态自动机 600 包括已初始化状态 602、正在执行状态 604、正在取消状态 606、正发生故障状态 608、正在补偿状态 610 和已关闭状态 612。

[0044] 例如,状态自动机 600 描述 workflow 活动中工作项目(例如,工作项目 422)的执行的过程流。如图 4 中所示的工作项目 422-1 当它在队列 410 中入队时首先被初始化。接着,工作项目 422-1 在正在执行状态(例如,图 6 中的正在执行状态 604)中执行之前从队列 410 中出队或移除到分派器 412。根据工作项目 422-1 的执行期间的参数或条件,工作项目 422-1 可前进到正在取消状态 606(例如,图 4 的正在取消状态 426)或正发生故障状态 608。在一个实施例中,工作项目 422-1 可从正在取消状态 606 前进到正发生故障状态 608。在替换实施例中,正在补偿状态 610 描述当发生故障或异常时要执行的一组操作或功能。

[0045] 例如,假定在工作项目(例如,工作项目 422-1)的执行期间发生异常,诸如函数的参数遗失。系统 400 将工作项目 422-1 转移到正发生故障状态 608。这样做,系统 400 也在将工作项目 422-1 转移到已关闭状态 612 之前在正在补偿状态 610 中执行垃圾收集(例如,将操作中之前执行的部分从高速缓存或存储器中移除、使参数值复位等)。例如,正在补偿状态 610 中的工作项目可触发诸如恢复之前用于执行其它工作项目的数据的操作。已关闭状态 612 指示活动(例如,图 5 中的活动 500)的执行已完成。

[0046] 在一个实施例中,状态自动机 600 在合成活动的工作项目之间建立关系。例如,关系规则之一可包括,在将活动树的根节点中的方法或工作项目转移到已关闭状态 612 之前,子节点中的所有工作项目应处于已初始化状态 602 或已关闭状态 612。另一规则可要求,为了将活动树的子节点中的工作项目转移到正在执行状态 604,根节点中的工作项目必须已经处于正在执行状态 604。

[0047] 在另一实施例中,可在状态自动机 600 中定义一个或多个附加状态,而不背离本发明的实施例的范围。

[0048] 接着参考图 7A 到 7E,框图示出了根据本发明的实施例对 workflow 中的故障事件的异步处理。为简单起见且并非限制,图 7A 示出包括按照树结构组织的三个子工作项目的合成活动 702:事务\_1704、事务\_2706 以及事务\_3708。如图所示,根活动 702 包括“在显示上写文本”的方法。以上工作项目的活动方法或操作也包括以下:

[0049] 事务\_1704:

[0050] { 插入文本 (“1 和 2”);

[0051] HANDLEFAULT() (处理故障);

[0052] }

[0053] 事务\_2706:

[0054] { 插入文本 (“3 和 4”);

[0055] HANDLEFAULT() ;

[0056] }

```
[0057] 事务_3708 :  
[0058] { 插入文本 (“5 和 6”);  
[0059]     暂停 180 秒 ;  
[0060]     插入文本 (“结束”);  
[0061] }
```

[0062] 在图 7B 中,事务\_1704、事务\_2706 以及事务\_3708 转移到正在执行状态 710。如图所示,事务\_1704 通过在对用户 430 的显示(例如,用户界面 428)上插入文本 (“1 和 2”)来执行所包括的操作。

[0063] 尽管处于正在执行状态 710 中,但发生了故障事件 722 或异常。故障事件 722 可包括对遗失数据、执行故障、对数据存储的不准确访问等的警告通知。在此示例中,事务\_1704 包括用于处理故障事件 722 的 `handleFault()` (故障处理) 函数 716。在一个实施例中,`handleFault` 函数 716 类似于诸如操作系统或托管执行环境的其它执行环境中用于故障处理的“捕捉”函数。因此,对 `handleFault` 函数 716 或其它“捕捉”处理器的故障传播或分派是异步的。

[0064] 当出现故障事件 722 时,事务\_1704 转移到正发生故障状态 712,且事务\_1704 转移到已关闭状态 714。在一个实施例中,响应于故障事件 722,`handleFault()` 函数 716 被调用,并被置于队列(未示出)用于处理。

[0065] 采用用于异常传播和处理的该良好定义的协议,替换实施例可处理多个异常,且可在异常的传播与正常程序执行交错时调度多个异常。

[0066] 在图 7 中,事务 2706 和事务 708 处于正在执行状态 710 中。类似于执行事务\_1704,事务\_2706 执行所包括的操作。在此示例中,将文本 (“3 和 4”)插入到显示上。此外,事务\_2706 也包括与事务\_1704 用于处理故障事件 722 的 `handleFault()` 函数 716 相似的 `handleFault()` 函数。

[0067] 在替换实施例中,`handleFault()` 函数 716 可根据活动的执行层次或执行分层结构将通知 720 传播或转送到处于正在执行状态 710 中的其余工作项目。例如,当事务\_1704 处于正发生故障状态 712 中时,`handleFault()` 函数 712 可传播通知 720(例如,“抛出”函数),使得父活动\_1702 的 `handleFault()` 函数可如同通知 720 是故障事件或异常一样来处理它。在一个实施例中,子活动可将抛出函数的目标限于其活动树中的父活动。在另一实施例中,异常处理可与活动的树形结构高度关联或连系。

[0068] 通过建立用于处理故障事件的正发生故障状态 712,本发明的实施例允许异步的故障处理或异常处理,且处于正在执行状态 710 中的其余工作项目或活动可继续执行。此外,另一替换实施例允许调度处理故障事件。例如,在响应于通知 720 时,事务 2706 可在转移到正发生故障状态 712 之前被置于调度器队列 718 中。在另一实施例中,通知 720 可被抑制,使得处于正在执行状态 710 中的其它工作项目或活动继续执行。在一个实施例中,事务\_1704 在传播或传送通知 720 之后转移到已关闭状态 714。在又一实施例中,故障传播和处理存活,并横跨钝化周期。

[0069] 在图 7D 中,事务 3708 在正在执行状态 710 中执行。例如,事务\_3708 所包括的操作在将文本 (“结束”)插入到显示上之前插入文本 (“5 和 6”)并暂停 180 秒。然而,所包括的操作不包括用于故障处理的函数。因此,在完成所包括的操作之后,在图 7E 中事务

\_3708 被转移到已关闭状态 714。此外,事务 2706 在从调度器队列 718 出队到正发生故障状态 712 之后也转移到已关闭状态 714。

[0070] 不作为限制,附图 A 示出根据本发明的实施例声明性提出异常的示例性实现。在一个实施例中,程序员或开发者可设计用于处理特定类型的故障事件或异常的故障处理器。在又一实施例中, workflow 中的工作项目或活动可能不包括函数或不能够处理故障事件。在此实施例中, workflow 执行环境处理故障事件。在又一实施例中,可经由图 4 中对用户 430 的 UI429 向用户提供一个或多个故障处理后操作。

[0071] 尽管图 7A 到 7E 顺序示出了(即,事务是顺序执行的)状态自动机的正在执行状态或部分的屏幕截图,但处于正在执行状态中的工作项目可同时处理或基本上同时处理,而不背离本发明的范围。

[0072] 图 8 是示出根据本发明的实施例用于对工作流的活动的故障事件进行异步处理的方法的流程图。例如,图 8 中所示的方法可被表示为存储在计算机可读介质中的计算机可执行指令,如图 9 所示。例如,状态机 902 在 802 为活动定义状态自动机(例如,状态自动机 600),且状态自动机至少包括正在执行状态、正发生故障状态和已关闭状态。活动组件 904 在 804 将活动定义为包括多个工作项目。所定义的活动具有用于多个工作项目的执行分层结构或执行序列(例如,树结构)。每一工作项目包括用于执行活动的一部分的操作。

[0073] 调度器组件 906 在 806 将每一工作项目转移到正在执行状态。执行组件在 808 执行处于正在执行状态中的所转移工作项目的所包括操作。在 810,标识组件 910 基于执行分层结构和所包括的操作响应于故障事件标识所转移的工作项目中的一个或多个。在 812,故障处理器 912 通过调用一个或多个所标识工作项目中的故障处理操作(例如, `handleFault()` 函数 716) 以将这一个或多个所标识的工作项目转移到正发生故障状态,同时执行未由标识组件响应于故障事件标识的其余转移工作项目所包括的操作,来异步处理故障事件。在一个实施例中,故障处理器 912 通过将一或多个所标识工作项目转移到正发生故障状态来异步处理故障事件。在又一实施例中,故障处理器 912 通过使一或多个所标识的工作项目入队到调度器队列(例如,调度器队列 718)中来异步处理故障事件。

[0074] 在替换实施例中,计算机可读介质 900 还包括故障传播组件 914,它用于根据活动的执行分层结构将通知从一个或多个所标识的工作项目传送到其余所转移的工作项目。通知 720 指示所标识的一个或多个工作项目处于正发生故障状态中。在又一实施例中,计算机可读介质 900 还包括转移组件 916,用于响应于所传送的通知将其余的所转移的工作项目从正在执行状态转移到正发生故障状态。

[0075] 在另一替换实施例中,计算机可读介质也可包括补偿组件 918,用于根据异步处理故障事件恢复或补偿与活动相关联的数据。禁止组件也可以是计算机可读介质 900 的一部分,用于抑制通知对其余所转移的工作项目的传送。

[0076] 尽管结合诸如图 4 的系统 400 的示例性计算系统环境进行了描述,但本发明的实施例可用于众多其它通用或专用计算系统环境或配置。计算系统环境不旨在对本发明的任何方面的使用范围或功能提出任何限制。而且,计算系统环境不应被解释为对在示例性操作环境中所示组件的任何一个或组合有任何依赖性 or 要求。适用于本发明各方面的公知的计算系统、环境和 / 或配置示例包括,但不限于:个人计算机、服务器计算机、手持式或膝上型设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费者电子产品、移动电

话、网络 PC、小型机、大型机、包括上述系统或设备中的任一个的分布式计算机环境等。

[0077] 本发明的各实施例可以在由一个或多个计算机或其他设备执行的诸如程序模块等计算机可执行指令的通用上下文中描述。一般而言，程序模块包括但不限于：执行特定的任务或实现特定的抽象数据类型的例程、程序、对象、组件和数据结构。本发明的各方面也可以在分布式计算环境中实现，其中任务由通过通信网络链接的远程处理设备执行。在分布式计算环境中，程序模块可以位于包括存储器存储设备在内的本地和远程计算机存储介质中。

[0078] 在操作中，系统 400 执行如在诸如图 8 的各附图中所示出的计算机可执行指令来实现本发明的各方面。

[0079] 除非另有指定，否则此处所示和所述的本发明各实施例的操作的执行或进行的次序不是必需的。也就是说除非另有指明，否则各操作可按照任何次序执行，且本发明的实施例可以包括比本文所公开的或多或少的操作。例如，构想了在另一操作之前、同时或之后执行或进行某一操作是在本发明各方面的范围之内。

[0080] 本发明的各实施例可以用计算机可执行指令来实现。计算机可执行指令可以被组织为一个或多个计算机可执行组件或模块。本发明的各方面可以用任何数量的这些组件或模块及其任何组织来实现。例如，本发明的各方面不限于在各附图和本文中示出的特定的计算机可执行指令或者特定的组件或模块。本发明的其他实施例可以包括具有比在本文中示出和描述的或多或少的功能的不同计算机可执行指令或组件。

[0081] 当介绍本发明或其实施例的各方面的各元素时，冠词“一”、“一个”、“该”和“所述”指的是存在该元素的一个或多个。术语“包括”、“包含”、“具有”旨在是包括性的并且指的是可以有除所列元素之外的其它元素。

[0082] 尽管详细描述本发明的各方面，但显然修改和变化是有可能的，而不背离所附权利要求书中定义的本发明各方面的范围。在不背离本发明各方面的范围的情况下，可对以上构造、产品和方法进行各种改变，以上描述中所包含的以及在附图中所示出的所有一切旨在应被解释为说明性并且没有限制意义。

[0083] 附录 A

[0084]           <myActivities:Sequence x:Name= " myWorkflow " x:Class = " myApp.  
myWorkflow "

[0085]           xmlns:myActivities= " http://schemas.com/myActivities "

[0086]           xmlns:x= " http://schemas.microsoft.com/winfx/2006/xaml "

[0087]           xmlns= " http://schemas.microsoft.com/winfx/2006/xaml/  
workflow " >

[0088]                       <myActivities:WriteLine Text= " One " />

[0089]                       <myActivities:WriteLine Text= " Two " />

[0090]                       <ThrowActivity FaultType= " {x:Type

[0091]   InvalidOperationException} " />

[0092]                       <myActivities:WriteLine Text= " Unreachable code " />

[0093]                       <FaultHandlersActivity>

[0094]                       <FaultHandlerActivity FaultType= " {x:Type

```
[0095]     InvalidOperationException} " >
[0096]         <myActivities:WriteLine Text= " Three " />
[0097]         </FaultHandlerActivity>
[0098]         <FaultHandlerActivity FaultType= " {x:Type
[0099]     AppDomainUnloadedException} " >
[0100]         <myActivities:WriteLine Text= " Four " />
[0101]         </FaultHandlerActivity>
[0102]     </FaultHandlersActivity>
[0103] </myActivities:Sequence>
[0104]     <myActivities:Sequence x:Name= " myWorkflow " x:Class= " myApp.
myWorkflow "
[0105]     xmlns:myActivities= " http://schemas.com/myActivities "
[0106]     xmlns:x= " http://schemas.microsoft.com/winfx/2006/xaml "
[0107]     xmlns= " http://schemas.microsoft.com/winfx/2006/xaml/
workflow " >
[0108]         <myActivities:WriteLine Text= " Hello World " />
[0109]         <ThrowActivity FaultType= " {x:Type
InvalidOperationException} " />
[0110]         <myActivities:WriteLine Text= " Unreachable Code " />
[0111]     </myActivities:Sequence>
```

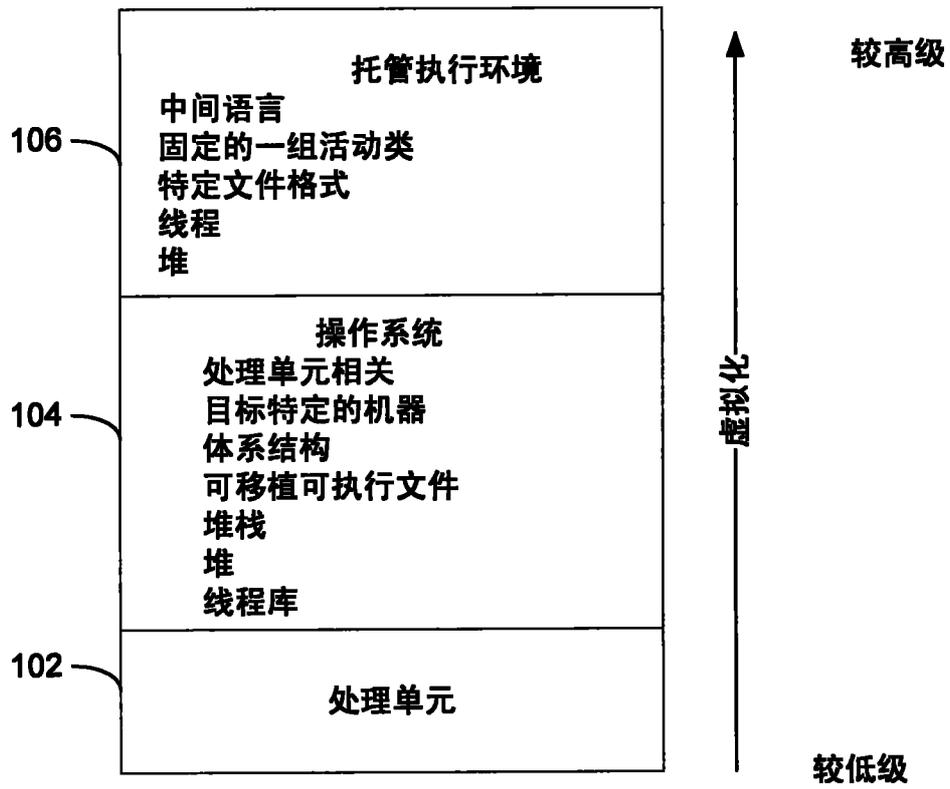


图 1

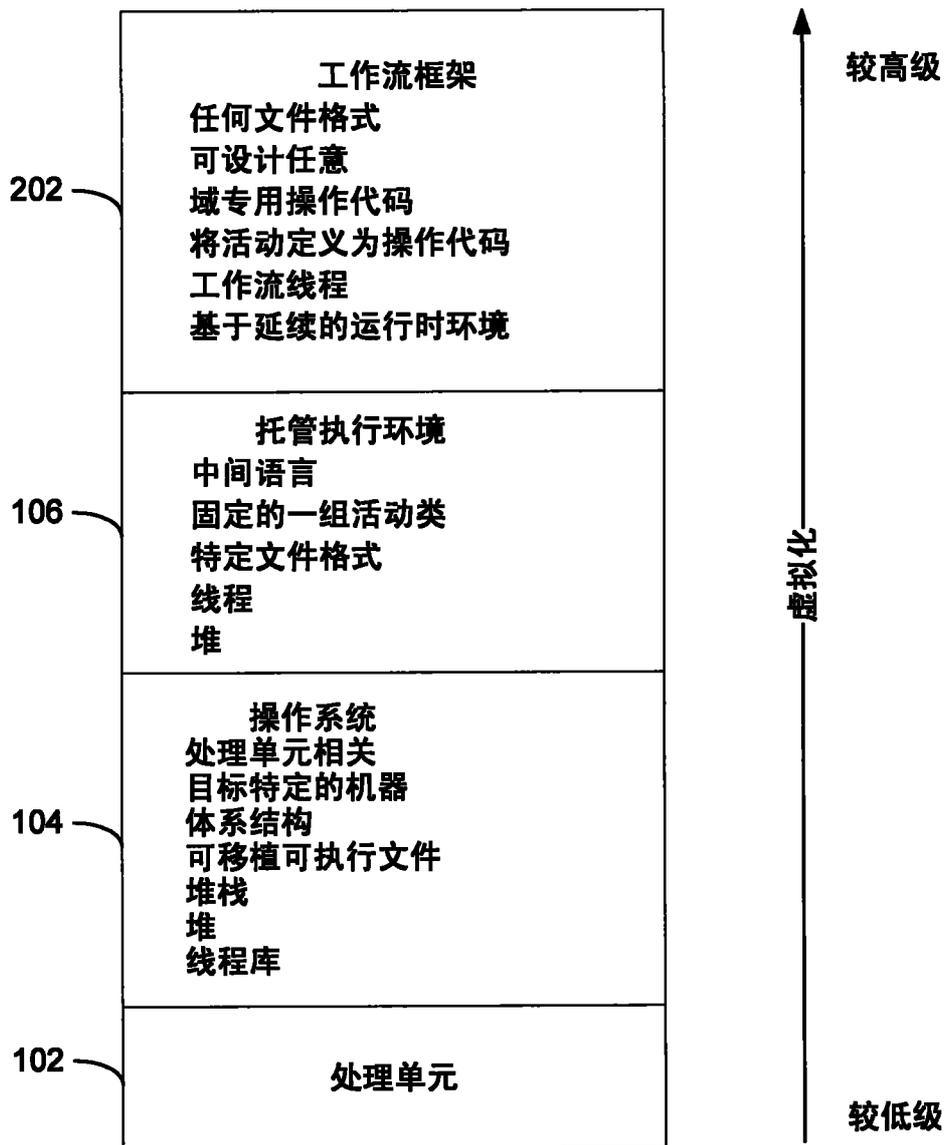


图 2

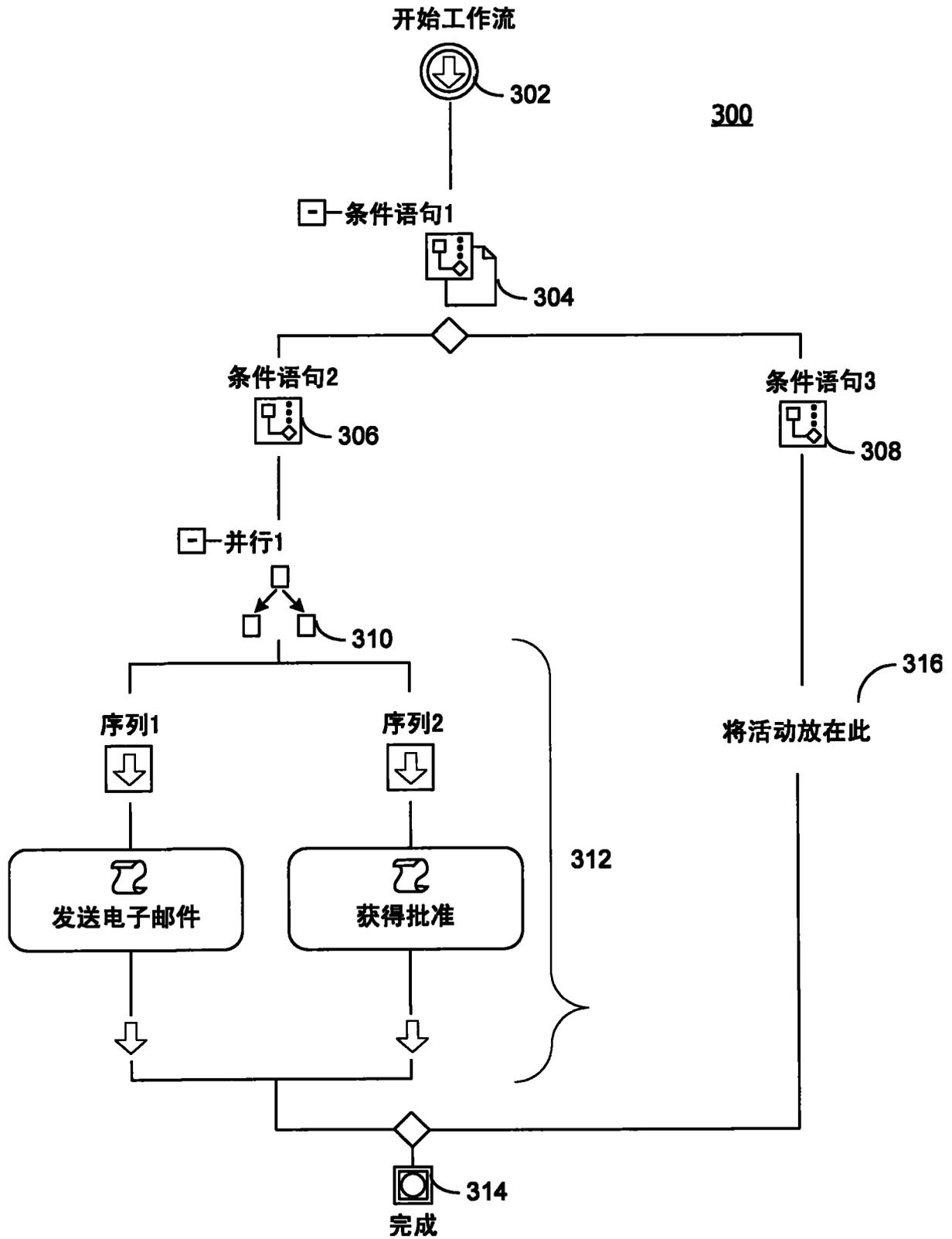


图 3

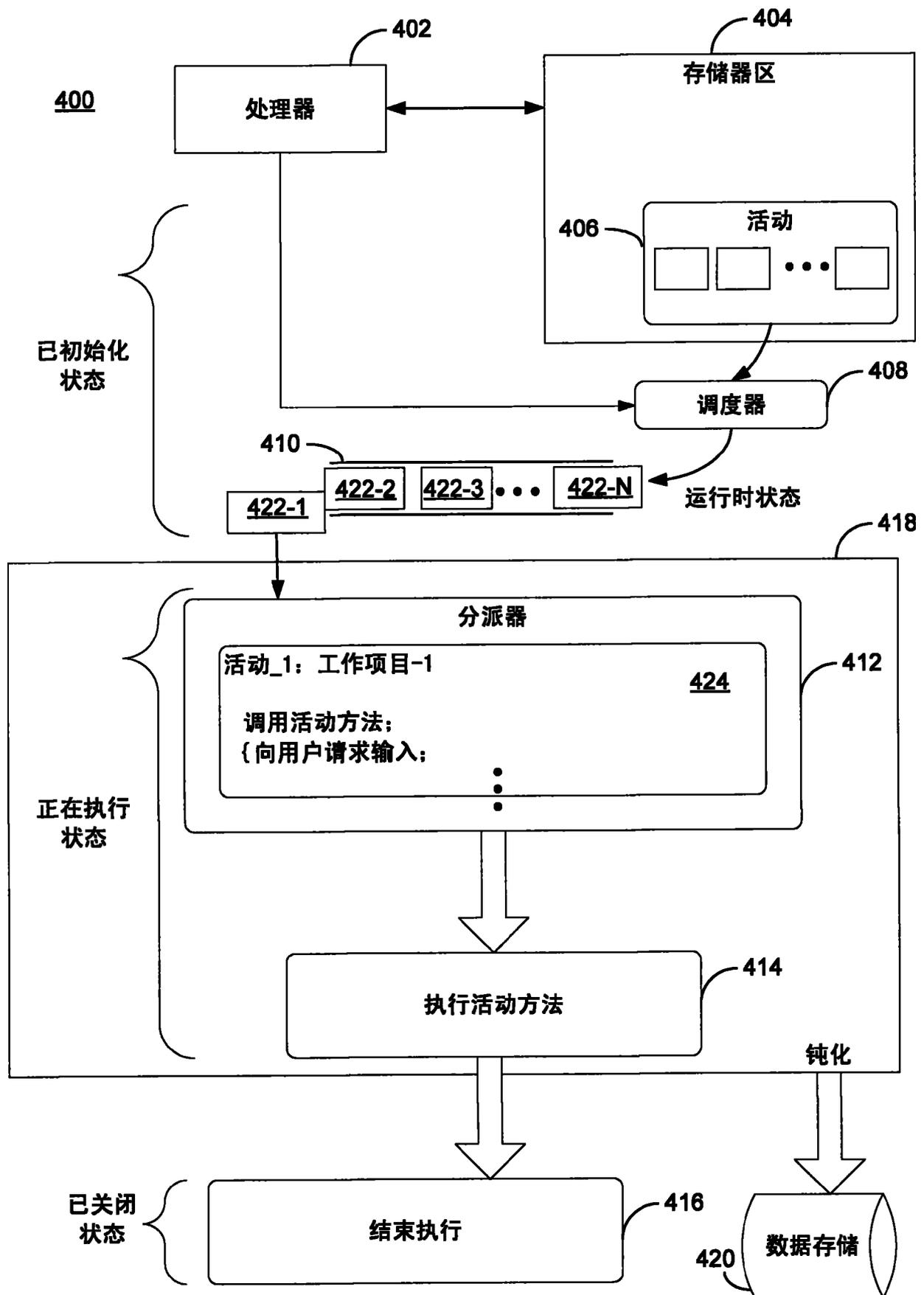


图 4

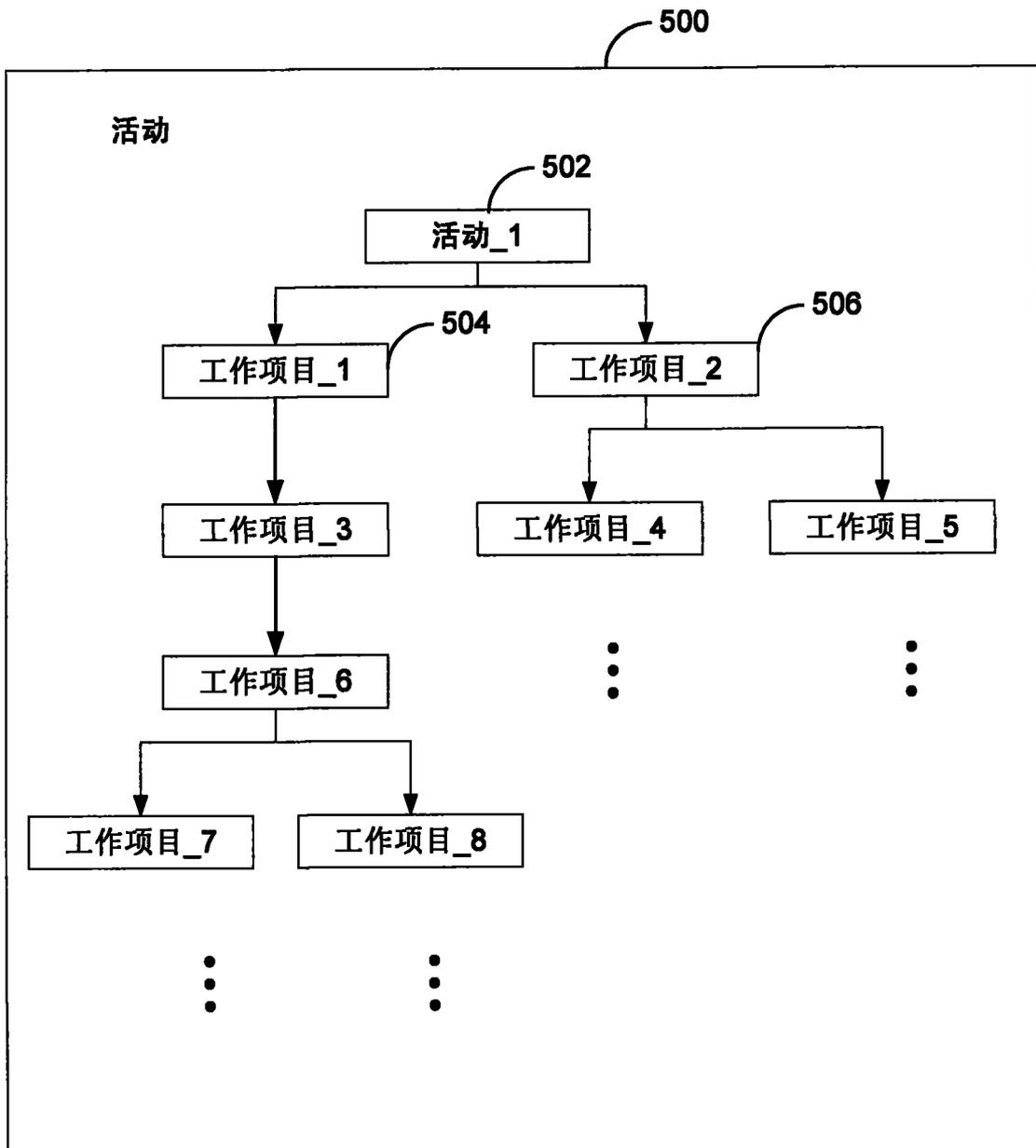


图 5

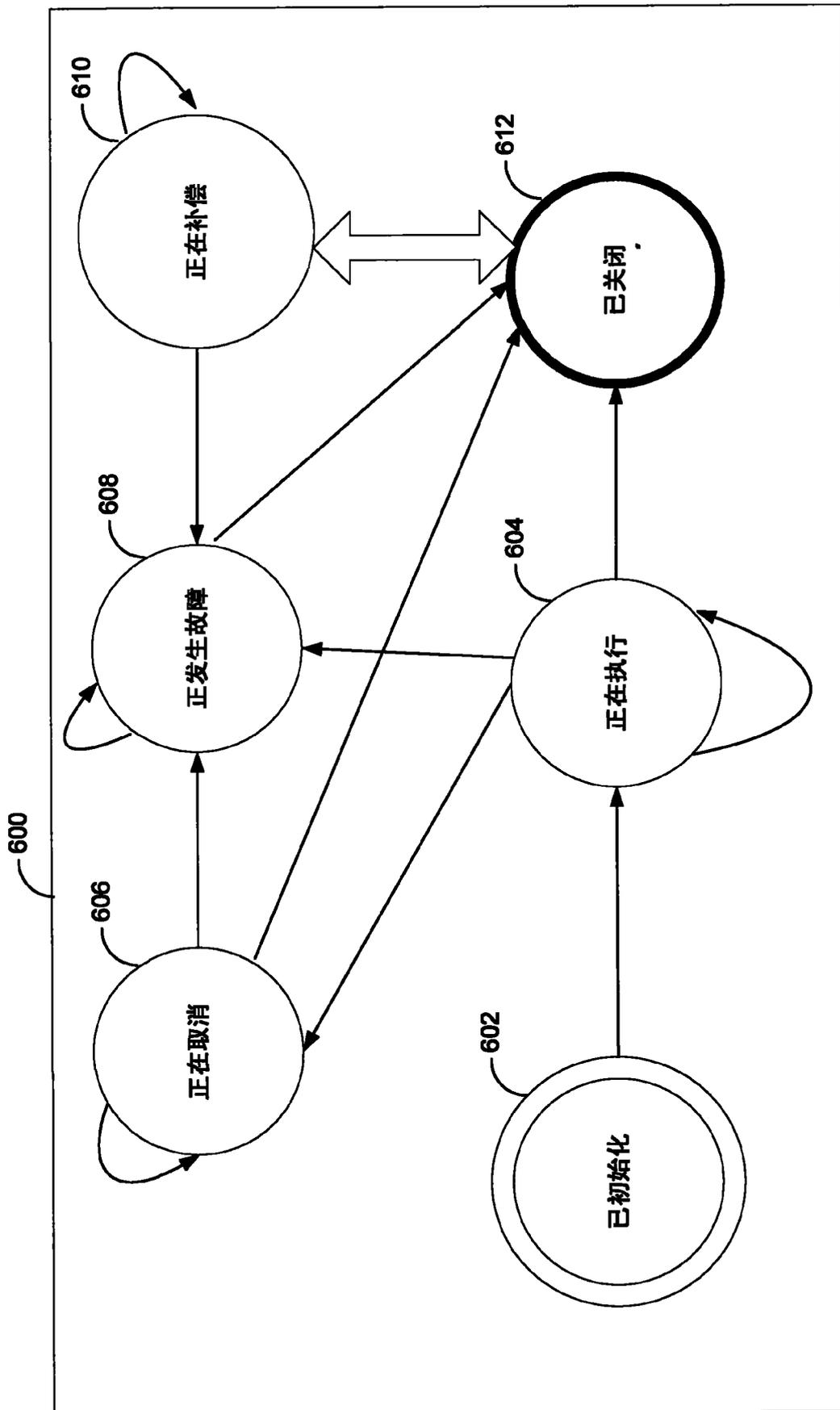


图 6

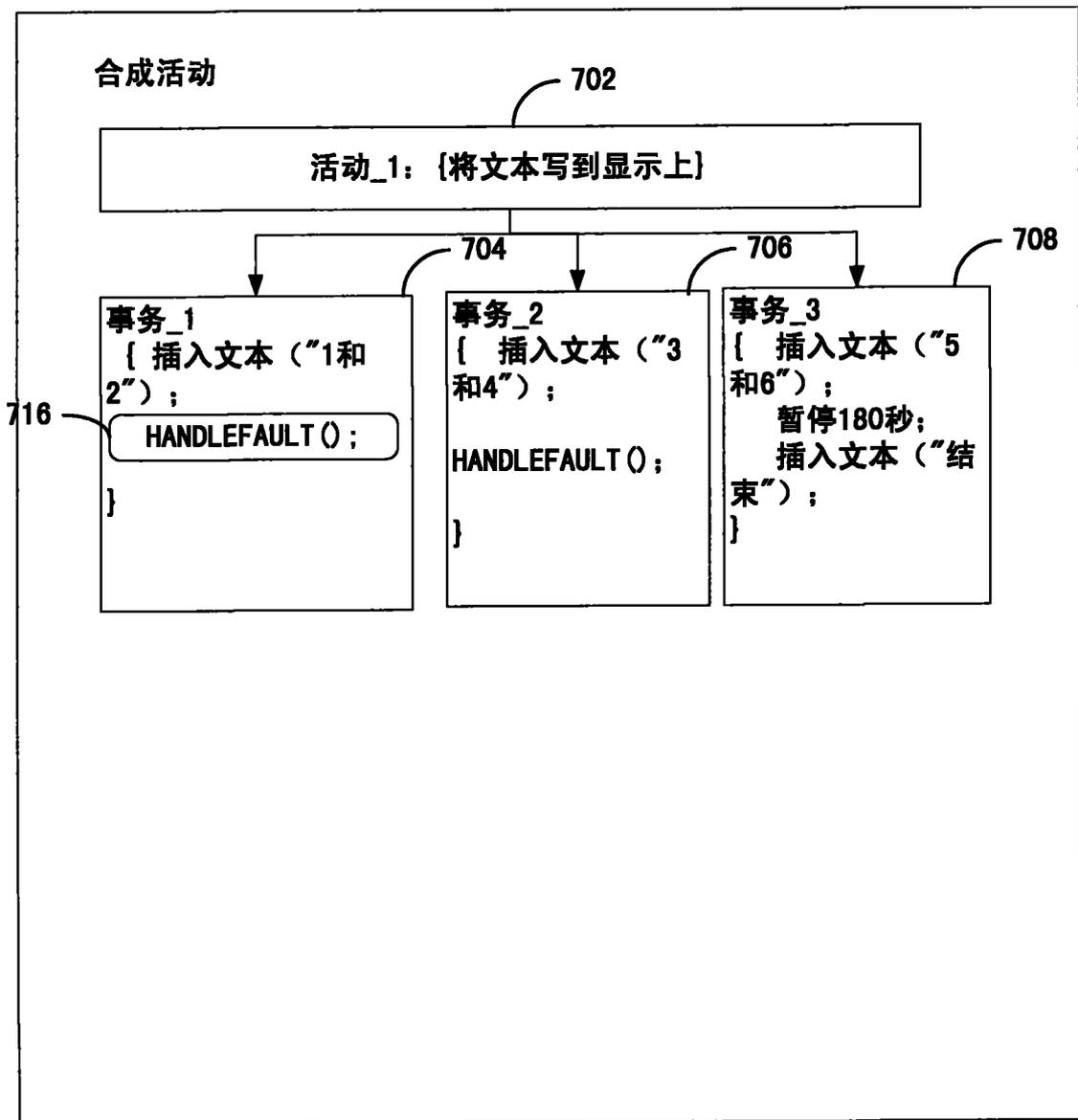


图 7A

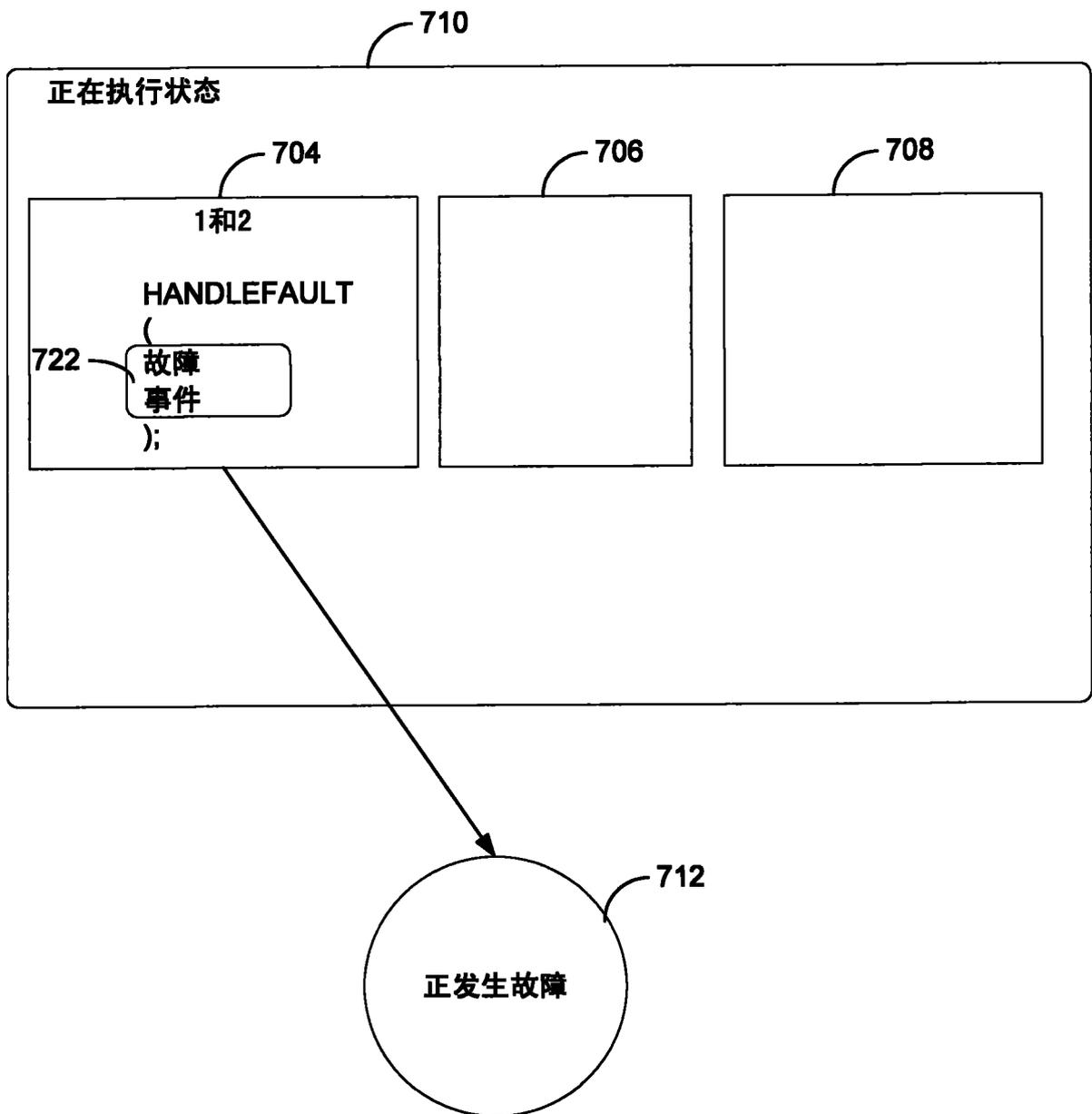


图 7B

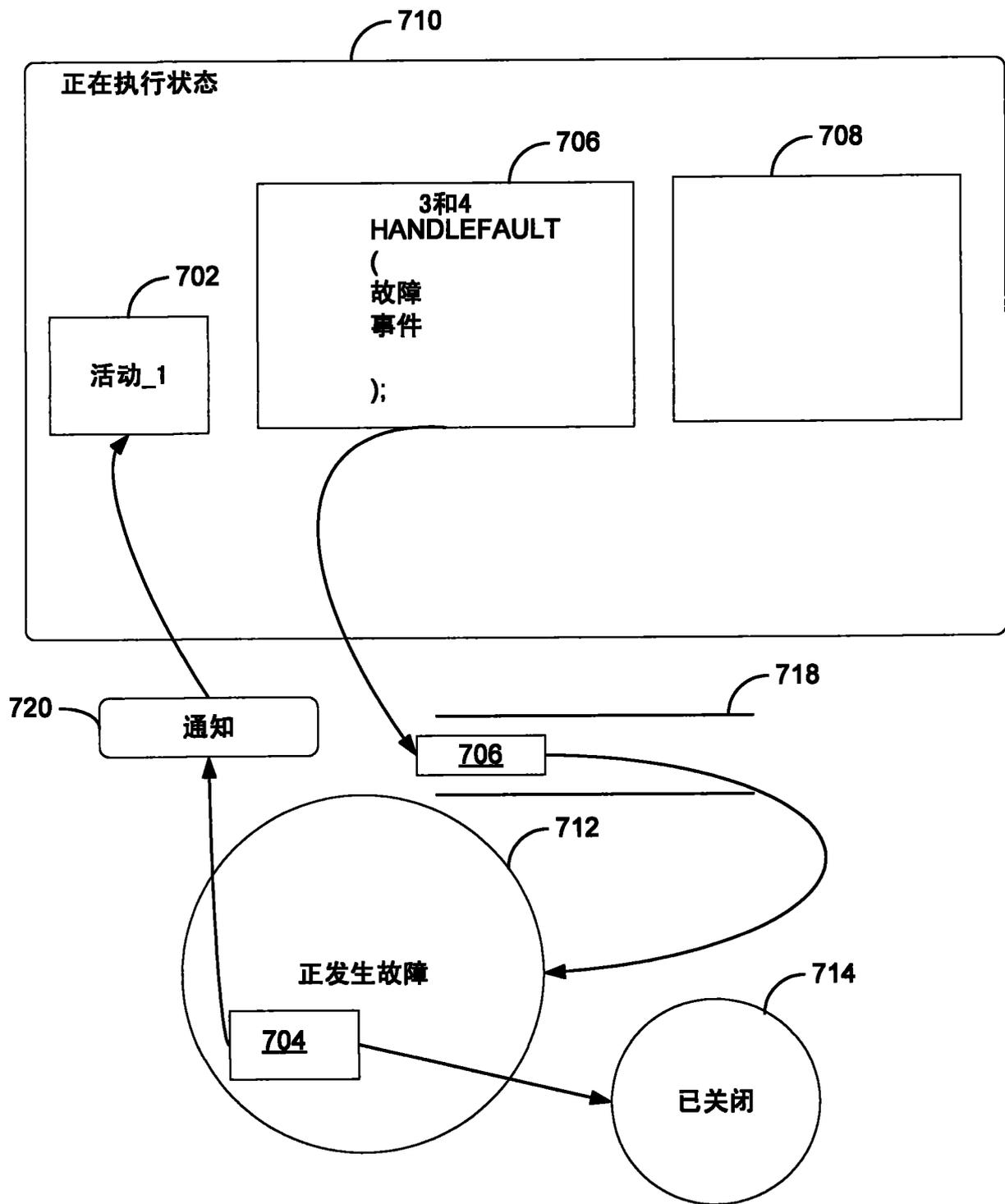


图 7C

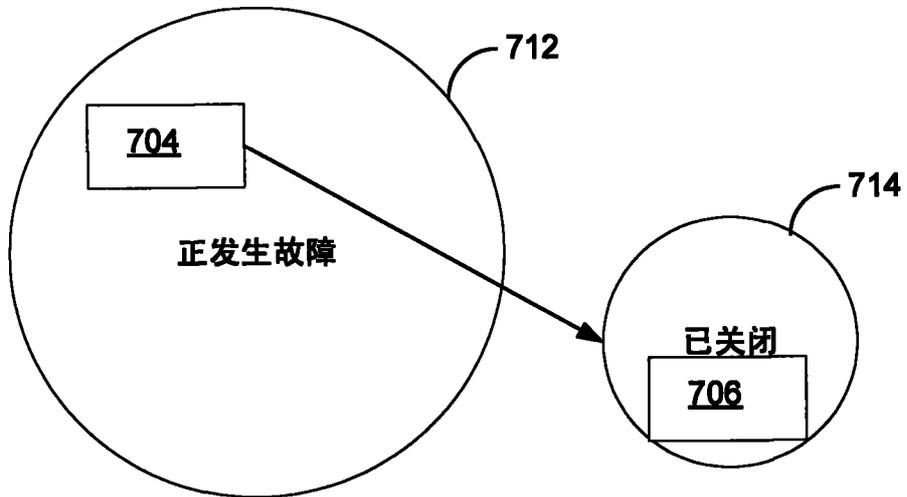
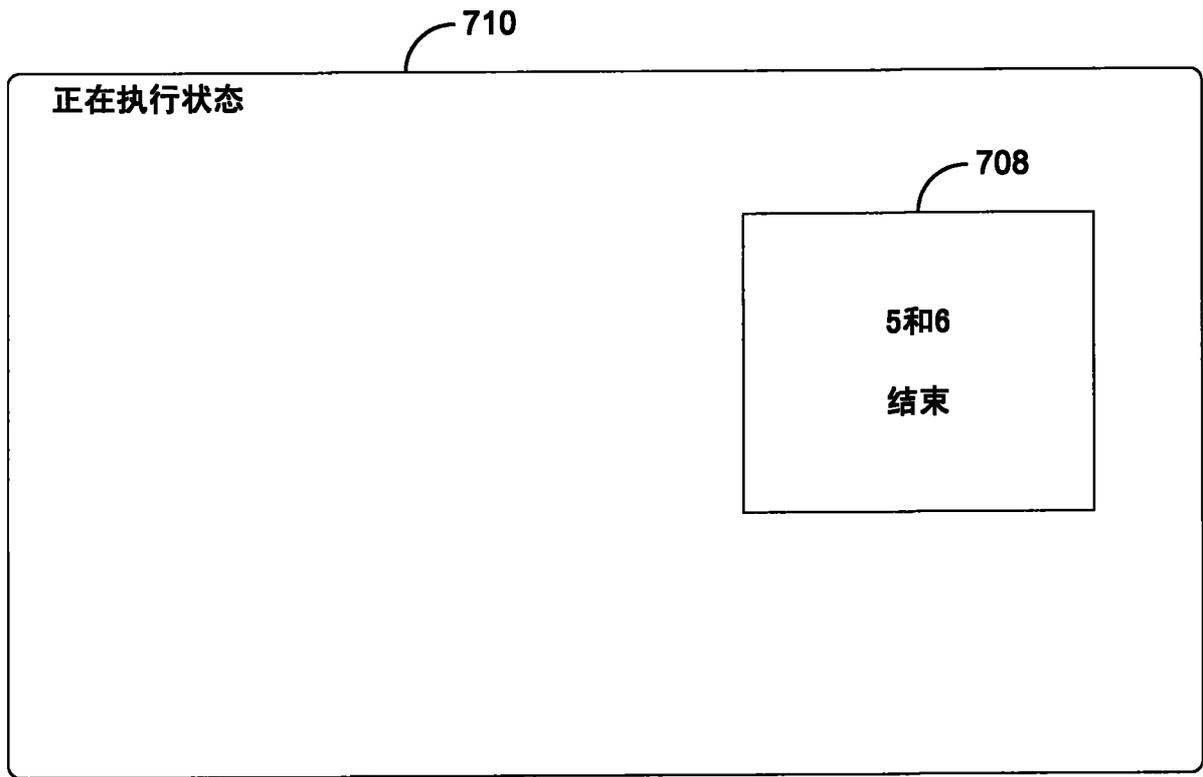


图 7D

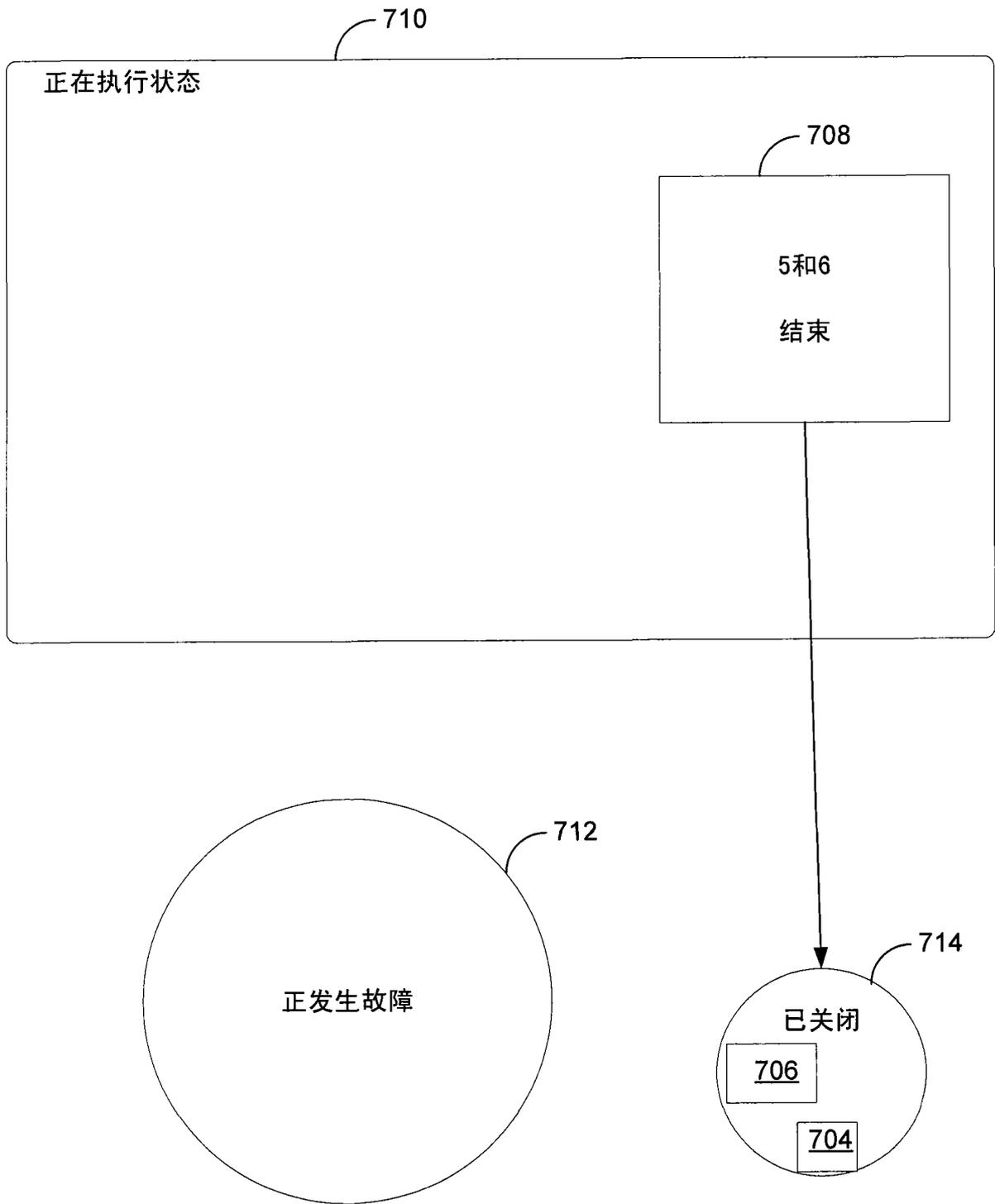


图 7E

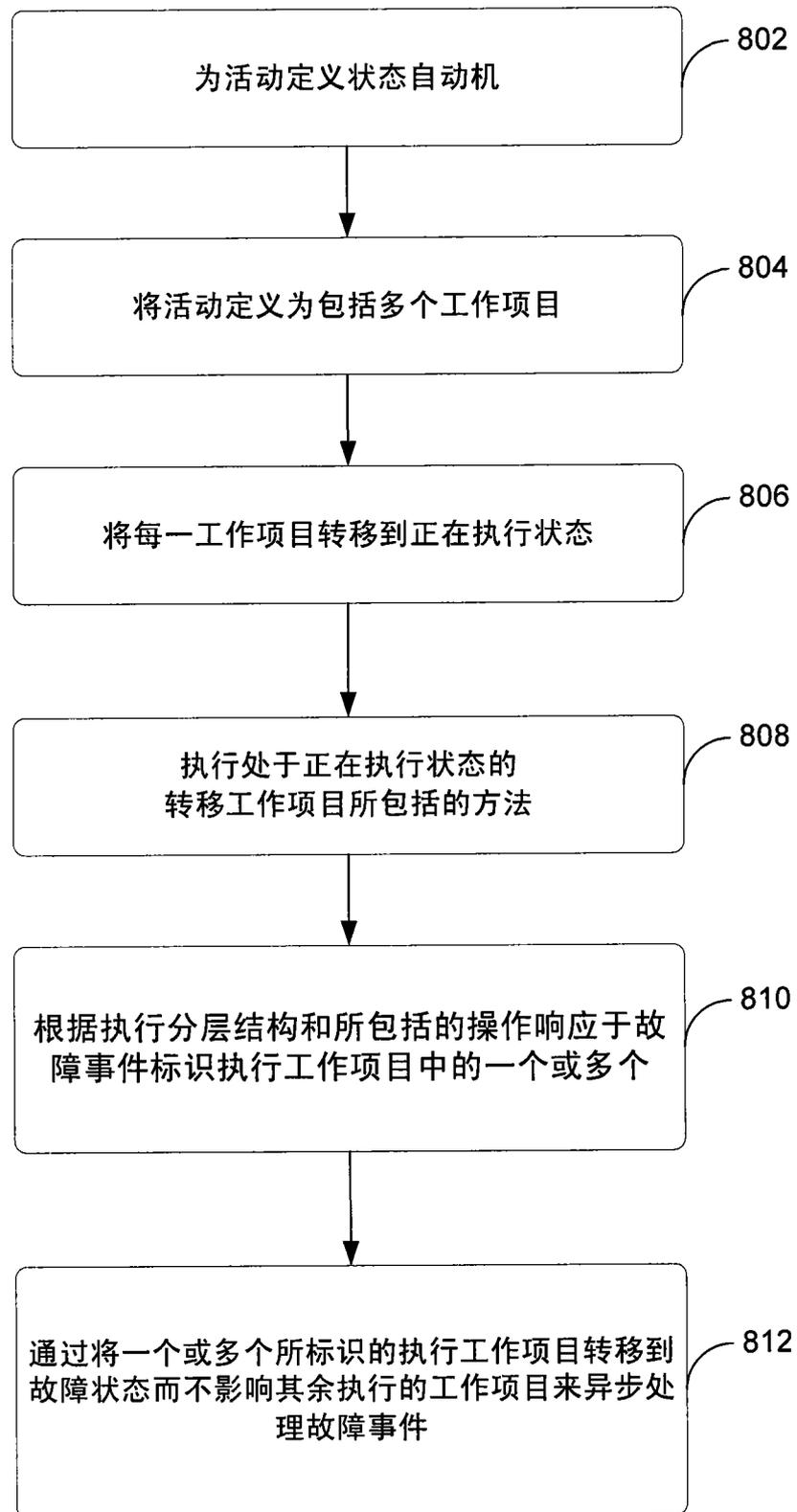


图 8

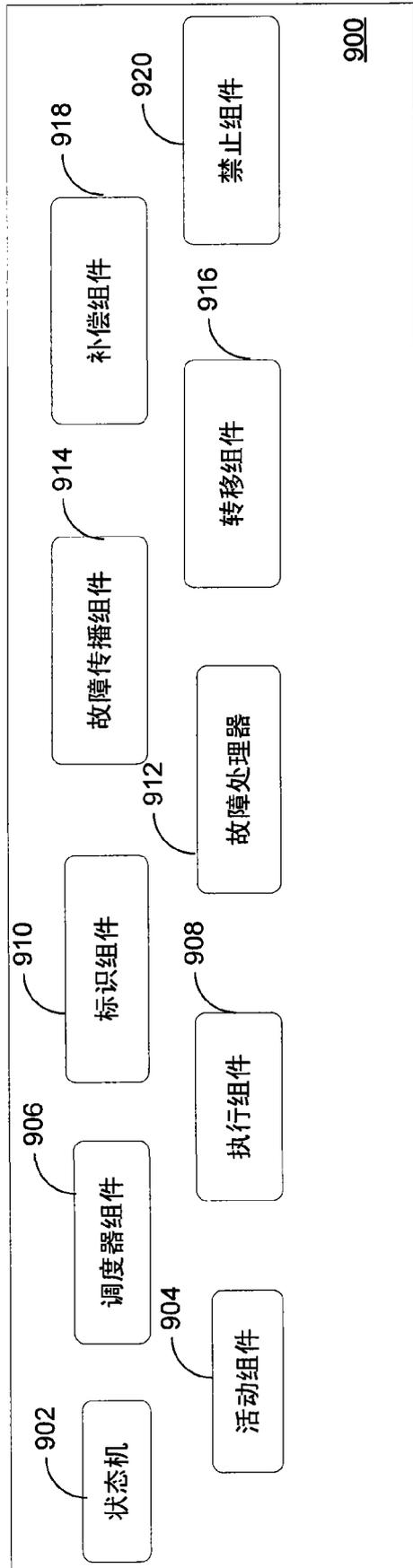


图 9