

一种软件架构的可视化方法及装置

相关申请的交叉引用

本申请要求在2019年12月23日提交中国专利局、申请号为201911336428.2、申请名称为“一种软件架构的可视化方法及装置”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及计算机技术领域，尤其涉及一种软件架构的可视化方法及装置。

背景技术

一个软件可以被分为多个不同的功能组件，软件架构用于描述软件的构成以及软件中各个功能组件之间的关联关系。为了能够更加直观的展示一个软件的结构架构，目前可以采用二维视图，在二维平面内显示该软件中的各个功能组件，利用形状以及连接线，展示软件架构。例如，在二维视图中可以利用形状代表功能组件，不同形状代表不同的功能组件。用连接线表征功能组件之间的关联关系。

但二维视图受限于维度，已形成的二维视图中的图形并不能进行旋转平移。二维视图的展示软件架构的方法灵活性和直观性较差。

发明内容

本申请提供一种软件架构的可视化方法及装置，用以提高软件架构展示的灵活性以及直观性。

第一方面，本申请实施例提供了一种软件架构的可视化方法，该方法可以由展示装置执行，方法包括：当展示装置接收到软件的结构架构信息时，可以先根据该架构信息构建三维视图。可选的，也可以构建二维视图。在构建了三维视图之后，展示装置可以展示该三维视图，还可以展示二维视图。其中，该架构信息指示软件包括一个或多个功能组件，在三维视图可以通过一个立体图形表征一个功能组件。

用户在查看三维视图时，还可以对三维视图执行一些操作。展示装置响应于用户对三维视图的操作，可以调整三维视图，并展示调整后的三维视图。展示装置也可以调整二维视图，并在用户的触发下显示二维视图。

通过上述方法，展示装置可以通过三维视图展示软件架构，这种展示方式更加直观，另外，还可以对三维视图进行调整，能够较为灵活、方便的调整软件架构。

在一种可能的设计中，展示装置在响应于用户对三维视图的操作，调整三维视图时，可以先更新软件的结构架构信息；之后，利用更新后的软件的结构架构信息调整三维视图。

通过上述方法，展示装置通过更新架构信息调整三维视图，这种调整方式更加便捷。

在一种可能的设计中，软件的结构架构信息用于描述该软件的结构架构，软件的结构架构信息包括以下至少一个：软件中功能组件的名称、软件中功能组件之间的关联关系、功能组件占用计算资源、软件中功能组件之间的从属关系、功能组件的结构信息。

通过上述方法，软件的架构信息包括的信息越丰富，三维视图所展示的信息越多，使得三维视图更加直观。

5 在一种可能的设计中，三维视图展示软件的架构的方式有很多，例如，三维视图中的立体图形之间的连接线可以指示软件中功能组件之间的关联关系，三维视图中立体图形可以指示软件中功能组件占用的计算资源，三维视图中立体图形之间的包含关系可以指示软件中功能组件之间的从属关系，三维视图中立体图形中三维图形可以指示功能组件中的实例。

通过上述方法，三维视图可以展示较多的信息，这些信息可以在三维视图中通过图形化的方式展示，三维视图能够更加直观、灵活的展示软件的架构。

10 在一种可能的设计中，用户对三维视图的操作有很多种，例如，用户可以对三维视图中国的立体图形执行旋转操作。展示装置响应于用户对三维视图中立体图形的旋转操作，可以旋转立体图形。

通过上述方法，用户通过对立体图形的旋转，可以改变立体图形的展示方向，以便可以全方位的查看该立体图形对应的功能组件。

15 在一种可能的设计中，用户对三维视图的操作有很多种，例如，用户可以对三维视图中国的立体图形执行移动操作。展示装置响应于用户对三维视图中立体图形的移动操作，移动立体图形。

通过上述方法，用户通过对立体图形的移动，可以改变立体图形的展示位置，可以改变功能组件之间的从属关系，也可以更清晰的查看该立体图形对应的功能组件。

20 在一种可能的设计中，三维视图中立体图形指示软件中功能组件占用的计算资源的方式有很多。例如，立体图形的横截面积可以指示功能组件占用的一种计算资源，立体图形的高可以指示功能组件中实例的数量。又例如，立体图形中三个不同方向的边可以分别指示功能组件占用的一种计算资源。

25 用户对三维视图的操作有很多种。例如，用户可以对三维视图中国的立体图形执行缩放操作。展示模块响应于用户对三维视图中立体图形的缩放操作，可以缩放立体图形，以调整立体图形对应的功能组件占用的计算资源。

通过上述方法，展示装置通过缩放该立体图形，可以较为便捷的变更功能组件占用的计算资源。

30 在一种可能的设计中，当立体图形的边用于指示功能组件所占用的不同计算资源时，若用户对三维视图中立体图形的边执行拉伸操作，展示装置响应于用户对三维视图中立体图形的边的拉伸操作，拉伸边，以调整立体图形对应的功能组件所占用的边对应的计算资源。

35 若用户对三维视图中立体图形的顶点执行拉伸操作，展示装置响应于用户对三维视图中立体图形的顶点的拉伸操作，沿拉伸操作的方向等比例缩放立体图形的全部边，以调整立体图形对应的功能组件所占用各个边对应的计算资源。

通过上述方法，展示装置通过缩放该立体图形的边，可以较为便捷的变更功能组件占用的不同计算资源。

40 在一种可能的设计中，三维视图中立体图形包括三维图形，一个立体图形包括的三维图形指示立体图形对应的功能组件内的实例。用户可以对三维视图中三维视图进行移动、旋转、或缩放等操作，展示装置响应于用户对三维视图中三维图形的操作，调整三维图形。

例如，展示装置可以响应于用户对三维视图中三维图形的缩放操作，缩放三维图形，调整三维图形指示的实例占用的计算资源。

通过上述方法，展示装置通过调整该三维图形，可以较为便捷的变更功能组件中实例占用的计算资源。

5 在一种可能的设计中，三维视图中三维图形指示功能组件中实例占用的计算资源的方式有很多。例如，三维图形的横截面积可以指示功能组件中实例占用的一种计算资源，三维图形的高可以指示功能组件中实例的另一种计算资源。又例如，三维图形中三个不同方向的边可以分别指示功能组件中实例占用的一种计算资源。

10 当用户对响应于用户对三维视图中三维图形执行缩放操作时，若用户对三维视图中三维图形的边的执行拉伸操作，展示装置响应于用户对三维视图中三维图形的边的拉伸操作，沿拉伸操作的方向缩放三维图形的边，以调整三维图形对应的实例所占用的边对应的计算资源。若用户对三维视图中三维图形的顶点的执行拉伸操作，展示装置响应于用户对三维视图中三维图形的顶点的拉伸操作，沿拉伸操作的方向等比例缩放三维图形的全部边，以
15 调整三维图形对应的实例所占用的所有边对应的计算资源。

通过上述方法，展示装置通过缩放该三维图形的边或顶点，可以较为便捷的变更功能组件中实例占用的不同计算资源。

在一种可能的设计中，展示装置在调整三维视图之后，还可以根据调整后的三维视图更新二维视图，二维视图为预先构建的，用于展示软件的架构。二维视图中的二维图形代表
20 软件架构中的功能组件，二维图形之间的连接线可以表征功能组件间的关联关系，二维图形之间的包含关系可以表征功能组件之间的从属关系。

更新后，展示装置在用户的触发下，可以展示更新后的二维视图。

通过上述方法，展示装置除了展示三维视图，还可以展示二维视图，展示软件架构的方式更加多样化。

25 在一种可能的设计中，展示装置可以接收软件的架构的更新信息，更新信息包括软件中变更的功能组件的名称以及变更的功能组件的变更方式；该更新信息可以是其他装置发送的，也可以是由用户对三维视图执行操作时所产生的。展示装置可以根据更新信息更新三维视图或二维视图；并在用户触发下展示更新后的三维视图或二维视图。

30 通过上述方法，展示装置可以及时更新三维视图或二维视图，以使得三维视图和二维视图可以更加准确的展示软件的架构。

第二方面，本申请实施例还提供了一种展示装置，有益效果可以参见第一方面的描述
此处不再赘述。该装置具有实现上述第一方面的方法实例中行为的功能。所述功能可以通过硬件实现，也可以通过硬件执行相应的软件实现。所述硬件或软件包括一个或多个与上述功能相对应的模块。在一个可能的设计中，所述装置的结构中包括接收模块、构建模块、
35 以及展示模块，这些模块可以执行上述第一方面方法示例中的相应功能，具体参见方法示例中的详细描述，此处不做赘述。

第三方面，本申请实施例还提供了一种计算设备，所述计算设备包括处理器和存储器，还可以包括通信接口以及显示屏，所述处理器执行所述存储器中的程序指令执行上述第一方面或第一方面任一可能的实现方式提供的方法。所述存储器与所述处理器耦合，其保存
40 确定交通流量的过程中必要的程序指令和数据。所述通信接口，用于与其他设备进行通信，

如接收软件的架构信息或更新信息。所述显示屏用于在所述处器的触发下向用户显示三维视图或二维视图。

第四方面，本申请提供了一种计算设备集群，该计算设备集群包括至少一个计算设备。每个计算设备包括存储器和处理器。至少一个计算设备的处理器用于访问所述存储器中的代码以执行第一方面或第一方面的任意一种可能的实现方式提供的方法。

第五方面，本申请提供了一种非瞬态的可读存储介质，所述非瞬态的可读存储介质被计算设备执行时，所述计算设备执行前述第一方面或第一方面的任意可能的实现方式中提供的方法。该存储介质中存储了程序。该存储介质包括但不限于易失性存储器，例如随机访问存储器，非易失性存储器，例如快闪存储器、硬盘（hard disk drive， HDD）、固态硬盘（solid state drive， SSD）。

第六方面，本申请提供了一种计算设备程序产品，所述计算设备程序产品包括计算机指令，在被计算设备执行时，所述计算设备执行前述第一方面或第一方面的任意可能的实现方式中提供的方法。该计算机程序产品可以为一个软件安装包，在需要使用前述第一方面或第一方面的任意可能的实现方式中提供的方法的情况下，可以下载该计算机程序产品并在计算设备上执行该计算机程序产品。

第七方面，本申请还提供一种计算机芯片，芯片与存储器相连，芯片用于读取并执行存储器中存储的软件程序，执行前述第一方面或第一方面的任意可能的实现方式中提供的方法。

附图说明

图 1 为本申请提供的一种展示装置的结构示意图；

图 2A 至图 2B 为本申请提供的一种二维逻辑视图的示意图；

图 3 为本申请提供的一种二维运行视图的示意图；

图 4 为本申请提供的一种二维部署视图的示意图；

图 5A 为本申请提供的一种三维逻辑视图的示意图；

图 5B 至图 5H 为本申请提供的一种调整三维逻辑视图的示意图；

图 6 为本申请提供的一种三维运行视图的示意图；

图 7 为本申请提供的一种三维部署视图的示意图；

图 8A 至图 8B 为本申请提供的另一种三维逻辑视图的示意图；

图 8C 至图 8F 为本申请提供的一种调整三维逻辑视图的示意图；

图 9 为本申请提供的另一种二维部署视图的示意图；

图 10 为本申请提供的另一种三维逻辑视图的示意图；

图 11 为本申请提供的另一种三维映射视图的示意图；

图 12 为本申请提供的一种视图构建方法示意图；

图 13 为本申请提供的一种展示软件架构的界面示意图；

图 14 为本申请提供的另一种展示软件架构的界面示意图；

图 15 为本申请提供的一种视图更新方法示意图；

图 16 为本申请提供的一种计算机集群的结构示意图；

图 17 为本申请提供一种系统的结构示意图。

具体实施方式

通常，一个软件从逻辑上可以分拆为多个功能组件，每个功能组件能够用于实现一种或者一系列相关的功能。每个功能组件也可以划分为更小粒度的功能组件。在本申请实施例中功能组件包括但不限于：模块、组件、服务、微服务以及实例。

5 为方便理解，下面对前述几种不同类型的功能组件进行说明。

1、模块(module)，是指将软件按照根据功能划分的一种组成部分，本申请实施例并不限定软件的划分方式。以订票软件为例，从功能上可以划分为数据库模块、匹配模块、以及支付模块。数据库模块中保存有不同列车的发车时间、每辆列车的各个停靠站点的时间、列车的余票数量以及价格等信息；匹配模块用于根据用户的需求从数据库中匹配满足要求的列车；支付模块可以在用户的触发下指示用户进行车票支付。

10 2、组件(component)，可以作为软件架构中模块的组成部分。在本申请实施例中组件可以指基于 C/C++ 等编程语言，在逻辑上划分的独立部分。例如组件可以是一个独立的.so 或是.dll 文件。

3、实例(instance)，可以是模块，或组件，或部署了模块或组件的计算实体。计算实体可以是虚拟机或容器或其他提供计算能力的实体。每个实例占用的计算资源包括以下一种或多种：处理器、内存、磁盘空间、带宽等。

每个功能组件包括至少一个实例。一个功能组件中包括的实例可以相同，此种情况下，软件架构中相同的实例被聚集到统一功能组件中，例如订票软件的软件架构中包括多个数据库模块，多个匹配模块，多个支付模块。在一种实现方式中，该订票软件的软件架构中包括 3 个功能组件，一个功能组件包括了多个数据库模块，另一个功能组件包括了多个匹配模块，另一个功能组件包括了支付模块。在另一种实现方式中，该订票软件的软件架构中包括 3 个功能组件，一个功能组件包括了多个运行了数据库的计算实体，另一个功能组件包括了多个运行了匹配模块的计算实体，另一个功能组件包括了多个运行了支付模块的计算实体。一个功能组件中包括的实例也可以不相同，此种情况下，软件架构中功能关联的实例被聚集到统一功能组件中，例如订票软件的软件架构中包括 3 个数据库模块，3 个匹配模块，3 个支付模块和 1 个负载均衡模块。那么该订票软件的软件架构中包括 4 个功能组件，其中 1 至 3 号功能组件中的每一个功能组件包括了 1 个数据库模块、1 个匹配模块、1 个支付模块，4 号功能组件包括了负载均衡模块。其中，4 号功能组件内的负载均衡模块用于在 1 至 3 号功能组件中负载均衡。在另一种实现方式中，该订票软件的软件架构中包括 4 个功能组件，其中 1 至 3 号功能组件中的每一个功能组件包括了 1 个运行了数据库模块的计算实体、1 个运行了匹配模块的计算实体、1 个运行了支付模块的计算实体，4 号功能组件包括了运行负载均衡模块的计算实体。

35 本申请实施例提供一种软件架构的可视化方法，该方法由展示装置执行。如图 1 所示，为本申请实施例提供的一种展示装置的结构示意图，该展示装置 100 包括接收模块 110、构建模块 120、展示模块 130。

接收模块 110 可以接收待展示的软件的架构信息。该架构信息包括但不限于：该软件中包括的功能组件的名称、功能组件之间的关联关系。可选的，还可以包括功能组件的结构信息。

40 构建模块 120 根据该架构信息构建三维视图，可选的，构建模块 120 还可以构建二维视图。

展示模块 130 用于在用户的触发下展示三维视图，展示模块 130 还可以响应于用户的操作，调整三维视图。

在本申请实施例中软件架构不再局限于二维视图，而是可以按照软件的架构信息构建两种维度不同的视图，在用户的触发下，可以按照用户的需求展示三维视图。在本申请实施例中，可以通过三维视图展示软件架构，三维视图相对于二维视图，更具有直观性，所能展示的信息也更多，可以有效提高软件架构展示的灵活性。

一般，基于架构视图所展示的信息，架构视图可以分逻辑视图、运行视图、部署视图等，其中，逻辑视图侧重于展示软件中功能组件的关联关系以及从属关系；关联关系是指功能组件之间存在信息交互，存在连接；从属关系是指功能组件的包含与被包含的关系，可以理解为功能组件的进一步细化。

运行视图侧重于展示软件中功能组件的运行状态或功能组件中实例的运行状态，如故障、正常运行等。

部署视图侧重于展示软件中功能组件部署的位置，如功能组件所处的服务器、所在的地理区域等。

而从架构视图的维度上，架构视图可以分为二维视图、三维视图等等，在本申请实施例中展示软件的架构视图从维度上可以分为两种，分别为二维视图和三维视图，下面对二维视图和三维视图展示软件架构的方式进行说明。

如图 2A 所示，为一种软件架构的二维视图，在二维视图中，可以利用二维图形代表软件架构中的功能组件（也就是一个功能组件对应一个二维图形），二维图形之间的连接线可以表征功能组件间的关联关系，二维图形之间的包含关系可以表征功能组件之间的从属关系。

图 2A 所示的二维视图中包括五个功能组件，分别为功能组件 1、功能组件 2、功能组件 3、功能组件 4 以及功能组件 5。功能组件 1 利用线性框表示，功能组件 2、功能组件 3、功能组件 4 以及功能组件 5 利用特定的二维图标表示。线性框中包括代表功能组件 2 和功能组件 3 的二维图标，表示了功能组件 1 与功能组件 2 和功能组件 3 之间的从属关系，也即功能组件 1 中包括功能组件 2 和功能组件 3。代表功能组件 2 的二维图标与代表功能组件 4、功能组件 5 的二维图标之间有连接线，表明功能组件 2 与功能组件 4、功能组件 5 之间存在关联。同样的，功能组件 3 与功能组件 5 之间存在关联。

由于图 2A 所示的二维逻辑视图中一些功能组件是利用二维图标表示的，二维图标不便于进行划分，无法进一步展示这些功能组件的内部结构。作为一种可能的实施方式，在二维视图中功能组件也可以均采用线性框表征，这样当对线性框进行划分后，可以利用线性框划分的多个组成部分，表征功能组件的内部结构。如图 2B 所示，为一种软件架构的二维视图，二维视图中包括五个功能组件，分别为功能组件 1、功能组件 2、功能组件 3、功能组件 4 以及功能组件 5。这五个功能组件均用线性框表示。功能组件 1 对应的线性框中包括代表功能组件 2 和功能组件 3 的线性框，表示了功能组件 1 与功能组件 2 和功能组件 3 之间的从属关系，也即功能组件 1 中包括功能组件 2 和功能组件 3。代表功能组件 2 的线性框与代表功能组件 4、功能组件 5 的线性框之间有连接线，表明功能组件 2 与功能组件 4、功能组件 5 之间存在关联。同样的，功能组件 3 与功能组件 5 之间存在关联。另外，代表功能组件 2 的线性框中进一步进行划分，划分为三个相同的小线性框，每个小线性框可以表征该功能组件的一个实例，表明功能组件 2 中包括 3 个实例。同样的，功能组

件 3 中包括 4 个实例。功能组件 4 中包括 2 个实例，功能组件 5 中包括 2 个实例。在二维视图中尽可能多的展示信息。

如图 2A 和图 2B 所示的二维视图侧重展示软件中功能组件的关联关系和包含关系，可以看做是软件架构的二维逻辑视图。由于运行视图需要展示功能组件的运行状态，目前二维视图下功能组件的运行状态可以借助文字信息，在二维逻辑视图的基础上，通过文字信息显示各个功能组件的运行状态。如图 3 所示，图 3 中每个功能组件的名称旁用文字的方式显示了功能组件的运行状态。图 3 中的文字显示，功能组件 1、功能组件 2 和功能组件 3 运行正常，功能组件 5 运行故障，功能组件 4 运行缓慢。

部署视图展示功能组件的部署位置，在二维视图下功能组件的部署位置也可以借助文字信息显示在二维逻辑视图的基础上。如图 4 所示，图 4 中每个功能组件旁用文字的方式显示了功能组件的部署位置。图 4 中的文字显示，功能组件 1、功能组件 2 和功能组件 3 部署在服务器 1 中，功能组件 5 部署在服务器 2 中，功能组件 4 部署在服务器 3 中。

从图 2A 至图 4 中，二维视图受限于维度，能够展示的信息有限，显示的方式也不够生动、直观。

下面对本申请实施例提供的软件架构的三维视图进行说明，功能组件中的实例相同或不同，三维视图展示软件架构的方式存在一定的区别，下面分别进行说明：

方式一：功能组件中的实例不同，功能组件中的实例互相配合实现该功能组件的功能。

在三维视图中，可以利用立体图形代表软件架构中的功能组件，立体图形之间的连接线可以表征功能组件间的关联关系。立体图形之间的包含关系可以表征功能组件之间的从属关系。立体图形内部的构成可以表征功能组件中的实例。

如图 5A 所示，为本申请实施例提供的一种软件架构的三维视图，三维视图中包括五个功能组件，分别为功能组件 1、功能组件 2、功能组件 3、功能组件 4 以及功能组件 5。功能组件 1 利用扁平的长方体表示，功能组件 2、功能组件 3、功能组件 4 以及功能组件 5 利用高度较大的长方体表示。扁平的长方体上放置了代表功能组件 2 和功能组件 3 的长方体，表示了功能组件 1 与功能组件 2 和功能组件 3 之间的从属关系，也即功能组件 1 中包括功能组件 2 和功能组件 3，功能组件 2 和功能组件 3 之间相互配合实现功能组件 1 所实现的功能。代表功能组件 2 的长方体与代表功能组件 4、功能组件 5 的长方体之间有连接线，表明功能组件 2 与功能组件 4、功能组件 5 之间存在关联，例如，功能组件 2 与功能组件 4、功能组件 5 之间可以进行信息交互。同样的，功能组件 3 与功能组件 5 之间存在关联。另外，代表功能组件 2 的长方体中进一步进行划分，划分为三个相同的小长方体，每个小长方体可以表征该功能组件的一个实例，表明功能组件 2 中包括 3 个实例，功能组件 2 中包括的 3 个实例相互配合实现功能组件 2 所实现的功能。同样的，功能组件 3 中包括 4 个实例。功能组件 4 中包括 2 个实例，功能组件 5 中包括 2 个实例。

如图 5A 所示的三维视图侧重展示软件中功能组件的关联关系和包含关系，可以看做是软件架构的三维逻辑视图。

展示模块 130 可以向用户展示如图 5A 所示的三维逻辑视图，用户在查看该三维逻辑视图时，还可以对三维逻辑视图执行一些操作，该操作包括但不限于：旋转、平移、缩放。构建模块响应于用户的操作，构建模块 120 可以对三维逻辑视图中功能组件对应的立体图形进行旋转、平移、缩放等，调整三维逻辑视图，展示模块 130 展示调整后的三维逻辑视图。

操作一、旋转

用户可以通过光标对三维逻辑视图中功能组件对应的立体图形执行旋转操作，构建模块 120 响应于该旋转操作，构建模块 120 可以按照光标的旋转方向，旋转该功能组件对应的立体图形。通过旋转该立体图形，用户可以全方位的查看该功能组件。

5 参见图 5B，当光标移动至功能组件 2 对应的长方体时，显示向右旋转的箭头 501 和向左旋转的箭头 502，用户可以通过点击向右旋转的箭头 501 或向左旋转的箭头 502，使得功能组件 2 对应的长方体向右旋转，或向左旋转，图 5B 以点击向左旋转为例。

作为一种可能的实施方式，三维逻辑视图中每个功能组件对应的立体图形设置有旋转操作区域，光标移动至旋转操作区域，用户触发光标在该旋转操作区域移动的移动方向以及移动角度指示该功能组件对应的长方体的旋转方向和旋转角度。光标的移动角度可以是基于长方体上某一点为固定点，移动前光标与该固定点的第一直线、和移动后光标与该固定点的第二直线，这两条直线之间的夹角确定。

参见图 5C，功能组件 2 对应的长方体可以设置为旋转操作区域，用户触发光标在该长方体向上移动，以该长方体的 A 点为固定点，移动前光标 503 与 A 点的连线、移动后的光标 504 与 A 点的连线，这两个连线的夹角为 30 度，响应于该旋转操作，构建模块 120 可以将该长方体以该长方体的中心线为轴，逆时针旋转三十度。

在上述说明中以用户触发光标移动执行旋转操作为例，可以理解，当显示该三维逻辑视图的显示屏为触摸屏时，用户也可以通过在触摸屏上的手势（如用户在触摸屏上点触立方图形，并触摸屏上画弧线）执行旋转操作。

20 操作二、移动

用户可以通过光标对三维逻辑视图中功能组件对应的立体图形执行移动操作，构建模块 120 响应于该移动操作，可以按照光标的移动方向和移动位置，移动该功能组件对应的立体图形，将该功能组件对应的立体图像移动到光标停留的位置。三维逻辑视图中立体图形中的移动可以改变软件架构中功能组件之间的从属关系，通过移动操作，用户更方便的变更功能组件之间的从属关系。另外，三维视图中各个立体图形之间也可能存在遮挡，通过移动三维视图中立体图形的位置，也能够使用户更加清晰的查看该三维逻辑视图。

参见图 5D，当光标移动至功能组件 2 对应的长方体时，显示向右平移的箭头 505 和向左平移的箭头 506，用户可以通过点击向右平移的箭头 505 或向左旋转的箭头 506，使得功能组件 2 对应的长方体向右平移，或向左平移，图 5D 中以点击向左旋转的箭头 506 为例。

作为一种可能的实施方式，三维逻辑视图中每个功能组件对应的立体图形设置有移动操作区域，光标移动至移动操作区域，用户触发光标在该移动操作区域的移动方向以及移动位置指示该功能组件 2 对应的长方体的移动方向和移动位置。

参见图 5E，功能组件 2 对应的长方体的上表面可以设置为移动操作区域，用户触发光标在该长方体的上表面向斜上方移动，移动前光标 507 和移动后光标 508 的位置参见图 5E，构建模块 120 可以将该长方体向斜上方移动，移动至光标停留的位置。

前述内容中以构建模块 120 移动的该功能组件对应的立体图形的方式为例进行说明，用户不仅可以对立体图形执行移动操作，也可以对立体图形之间的连接线进行移动，例如可以改变连接线所连接的立体图形。响应于该移动操作，构建模块 120 可以按照光标的移动方向和移动位置，改变该连接线所连接的立体图形，将该连接线所连接的立体图形变为

光标所在的立体图形。三维逻辑视图中立体图形之间连接线的移动可以改变软件架构中功能组件之间的关联关系，通过移动操作，用户更加方便的变更功能组件之间的关联关系。

参见图 5F，当光标移动至连接线 L 的端点 P1，端点 P1 连接功能组件 4，用户可以通过拖动该连接线的端点 P1，改变该连接线 L 的端点 P1 所连接的立体图形，拖动后端点 P1 所在的位置为 P2，P2 位于功能组件 5，拖动后，连接线 L 连接功能组件 5 对应的长方体和功能组件 3 对应的长方体。

在上述说明中以用户触发光标移动执行移动操作为例，可以理解，当显示该三维逻辑视图的显示屏为触摸屏时，用户也可以通过在触摸屏上的手势（如用户在触摸屏上点触立方图形，并触摸屏上画直线或弧线）执行移动操作。

操作三、放大或缩小操作（简称缩放操作）

用户可以通过光标对三维逻辑视图中功能组件对应的立体图形执行放大或缩小操作，构建模块 120 响应于该放大或缩小操作，可以按照光标的移动方向和移动位置，放大或缩小该功能组件对应的立体图形。通过缩小或放大该立体图形，用户可以更加清楚的查看三维逻辑视图，通过放大该立体图形也便于用户查看该功能组件的内部结构。

作为一种可能的实施方式，三维逻辑视图中每个功能组件对应的立体图形设置有缩放操作区域，光标移动至缩放操作区域，用户触发光标在该缩放操作区域的移动方向可以指示该功能组件对应的立方图形放大或缩小，光标在该缩放操作区域的移动位置可以指示该功能组件对应的立方图形放大或缩小的程度。例如，可以将三维视图中的立方图形（如长方体）的各条边以及各个顶点设置为缩放操作区域，三维视图中的立体图形（如长方体）的一条边的移动指示该缩小或放大操作为立体图形的一个维度上的放大或缩小。例如，在长方体的长、宽或高的方向进行放大。光标在各个顶点的移动指示该缩小或放大操作为等比例缩小或放大，也就是说，该立体图形的各个面均进行同等程度的缩小或放大。

参见图 5G，功能组件对应的长方体的各条边以及各个顶点可以设置为缩放操作区域，用户触发光标在功能组件 2 对应的长方体的边 B（长方体的宽度方向的边）向右移动，移动前光标 509 和移动后光标 510 的位置参见图 5F，构建模块 120 可以将该长方体进行放大，增加该长方体的长度，也就是增加边 C 的长度，放大至光标停留的位置。

参见图 5H，用户触发光标在该长方体的顶点 C 向斜上方移动，移动前光标 511 和移动后光标 512 的位置参见图 5G，构建模块 120 可以将该长方体进行等比例放大，放大至光标停留的位置。

在上述说明中以用户触发光标移动执行缩放操作为例，可以理解，当显示该三维逻辑视图的显示屏为触摸屏时，用户也可以通过在触摸屏上的手势（如用户在触摸屏双指点触立方图形所在区域，触摸屏上双指移动方向相对远离或靠近。又例如，用户在触摸屏双指点触立方图形的某一边或顶点，在触摸屏上画直线或弧线）执行移动操作。

由上可以看出图 5A 所示的三维视图所展示的软件架构与图 2A 所述的二维视图所展示的软件架构基本相同，图 5A 与图 2A 比较，还能够进一步展示每个功能组件的内部结构。二维视图中利用二维图形（如线性框、图标）表征功能组件，二维图形只能进行移动、缩放。三维视图中利用三维图形（如长方体）表征功能组件，三维图形可以进行旋转、移动、缩放。显然，相较于二维图形，三维图形能够展示的细节更多，展示的信息更丰富，展示方式也更加灵活、直观。

此外，本申请实施例还提供了三维运行视图以及三维部署视图，下面结构图 6 和图 7

分别进行介绍：

如图 6 所述，为本申请实施例提供的一种三维运行视图，在如图 5A 所示的三维逻辑视图的基础上，可以变更代表功能组件对应的长方体的颜色，利用不同颜色表征功能组件的不同运行状态。

5 图 6 中，白色表示功能组件运行正常，黑色条纹表示功能组件运行缓慢，运行缓慢的原因为该功能组件的磁盘空闲空间达到临近值，黑色表示功能组件运行故障。从图 6 中可以较为直接的获知功能组件 2 和功能组件 3 运行正常，功能组件 5 运行故障，功能组件 4 运行缓慢。相较于如图 4 所示的二维运行视图，三维运行视图能够更加形象的展示功能组件的运行状态。

10 需要说明的是，图 6 中利用颜色表征功能组件的运行状态的方式仅是举例，在实际应用中，颜色可以不局限在白色、黑色条纹以及黑色，运行状态也可以不局限在正常、缓慢以及故障。例如，可以利用同种颜色的明暗程度表征功能组件的运行效率，以用绿色表征功能组件的运行正常为例，绿色越深，表明功能组件的运行效率越高，绿色越浅，表明功能组件的运行效率越低。凡是能够通过颜色表征功能运行状态的方式均适用于本申请实施例。图 6 只显示了较大粒度的功能组件的运行状态，采用类似的方式，通过变更功能组件中包括的功能组件对应的立体图形（如小长方体）的颜色，通过不同的颜色可以表征小粒度的功能组件的不同运行状态。

15 如图 7 所述，为本申请实施例提供的一种三维部署视图，在如图 7 所示的三维逻辑视图的基础上，可以增加能够表示功能组件部署位置的三维图形，利用功能组件所在的三维图形表示功能组件所在的位置。

图 7 中增加了表征服务器的三维图形，功能组件 1、功能组件 2、以及功能组件 3 部署在服务器 1，功能组件 4 部署在服务器 2，功能组件 5 部署在服务器 3。从图 7 中可以较为直接的获知各个功能组件的部署位置。相较于如图 4 所示的二维部署视图，三维部署视图更加便于查看。

25 展示模块 130 可以向用户展示如图 6 所示的三维运行视图以及如图 7 所示的三维部署视图时，用户在查看该三维运行视图或三维部署视图时，也可以对该三维运行视图或三维部署视图执行旋转、平移、缩放等操作，构建模块 120 响应于用户的操作，可以对三维运行视图或三维部署视图中功能组件对应的立体图形进行旋转、平移、缩放。构建模块 120 可以对三维运行视图或三维部署视图中功能组件对应的立体图形进行旋转、平移、缩放的方式与构建模块 120 可以对三维逻辑视图中功能组件对应的立体图形进行旋转、平移、缩放的方式相同，具体可参见前述内容，此处不再赘述。

30 应需注意的是，前述内容对旋转、平移以及缩放操作进行了说明，本申请实施例并不限于旋转、平移以及缩放操作这三种操作。例如用户还可以通过光标或手势的方式，可以触发构建模块 120 删除三维视图中的立体图形（也就是删除软件架构中该立体图形对应的功能组件），删除三维视图中的三维图形（也就是删除软件架构中该三维图形对应的功能组件），删除三维视图中的立体图形之间的连接线（也就是删除软件架构中该立体图形对应的功能组件之间的关联关系）。也可以触发构建模块 120 添加三维视图中的立体图形（也就是在软件架构中增加该立体图形对应的功能组件），添加三维视图中的三维图形（也就是在软件架构中增加该三维图形对应的功能组件），添加三维视图中的立体图形之间的连接线（也就是添加软件架构中该立体图形对应的功能组件之间的关联关系）。也可以触

发构建模块 120 变更三维视图中的立体图形的颜色（也就是变更软件架构中该立体图形对应的功能组件的运行状态），变更三维视图中的三维图形的颜色（也就是变更软件架构中该三维图形对应的功能组件的运行状态）。

方式二：一个功能组件中多个实例相同。

5 在三维视图中，可以利用立体图形代表软件架构中的功能组件，立体图形可以指示功能组件占用的计算资源。立体图形之间的连接线可以表征功能组件间的关联关系。立体图形中包括的三维图形表征该功能组件中包括的实例。

立体图形指示功能组件占用的计算资源的方式有许多种，下面列举其中两种：

10 第一种，立体图形的横截面的面积指示功能组件中实例占用的第一计算资源（如处理器、内存、磁盘空间等）。立体图形的高度可以指代功能组件中实例的数量。

可选的，立体图形中的三维图形的横截面的面积指示功能组件中一个实例占用的一种计算资源，立体图形中的三维图形的高度指示功能组件中一个实例占用的另一种计算资源。

图 8A 所示，为本申请实施例提供的一种软件架构的三维视图，该三维视图中包括三个功能组件，分别为功能组件 1、功能组件 2、以及功能组件 3。功能组件 1、功能组件 2、
15 以及功能组件 3 均用长方体表示。长方体的上表面的面积用于指示功能组件中实例的处理器数量。可见，功能组件 2 对应的长方体的上表面积大于功能组件 3 对应的长方体的上表面积，功能组件 2 中实例中的处理器数量大于功能组件 3 中实例中的处理器数量，同理，功能组件 3 中实例中的处理器数量大于功能组件 1 中实例中的处理器数量。长方体的高度用于指示功能组件中实例的数量。可见，功能组件 2 对应的长方体的高度大于功能组件 1
20 对应的长方体的高度，功能组件 2 中实例中的数量大于功能组件 1 中实例的数量，同理，功能组件 3 中实例的数量大于功能组件 1 中实例的数量。

功能组件 1 对应的长方体包括 2 个上表面与该长方体相同、高度存在差异的小长方体，功能组件 2 对应的长方体包括 3 个上表面与该长方体相同、高度存在差异的小长方体，功能组件 3 对应的长方体包括 3 个上表面与该长方体相同、高度相同的小长方体。

25 每个小长方体用于表征该功能组件中一个实例，小长方体上表面用于表示该实例中的处理器，上表面的面积越大表明该实例的处理器数量越多，小长方体的高度用于表示该实例中的内存，小长方体的高度越大表明该实例的处理器内存越大。可见，图 8A 中，同一功能组件中的实例的处理器数量相同，功能组件 1 中包括两个内存大小不同的实例，功能组件 2 中包括三个内存大小不同的实例，功能组件 3 中包括三个内存大小相同的实例。
30 代表功能组件 2 的长方体与代表功能组件 1、功能组件 3 的长方体之间有连接线，表明功能组件 2 与功能组件 1、功能组件 3 之间存在关联。

图 8A 中，同一功能组件中的实例的处理器数量相同，若同一功能组件中的实例的处理器数量不同，也可以将功能组件对应的长方体划分为上表面积不同的小长方体，小长方体的上表面积不同表明该功能组件中的实例的处理器数量不同。功能组件对应的长方体的上表面积可以指示功能组件中实例所占用的处理器的最大数量。

第二种、三维视图中立体图形的不同方向的边分别指示不同的计算资源，如立体图形中长、宽以及高分别指示一种计算资源。

具体到立体图形中的三维图形，三维图形的长度、宽度、高度分别指示功能组件中一个实例占用的计算资源。

40 图 8B 所示，为本申请实施例提供的另一种软件架构的三维视图，该三维视图中包括

三个功能组件，分别为功能组件 1、功能组件 2、以及功能组件 3。功能组件 1、功能组件 2、以及功能组件 3 均用长方体表示。长方体的宽指示功能组件中所占用的处理器，长方体的长指示功能组件中所占用的内存，长方体的高指示功能组件中所占用的磁盘空间。

功能组件对应的长方体可以进一步划分，划分为多个小长方体，每个小长方体代表一个实例，小长方体的宽度代表实例的处理器数量，小长方体的宽度越大，表明实例中处理器的数量越多；小长方体的长度代表实例的内存，小长方体的长度约大，表明实例中内存的空间越大，小长方体的高度代表实例的磁盘空间，小长方体的高度越大，表明该实例的磁盘空间越多。

功能组件 1 对应的长方体包括 2 小长方体，功能组件 2 对应的长方体和功能组件 3 对应的长方体包括 3 个小长方体。

如图 8A 和图 8B 所示的三维视图侧重展示软件中功能组件的关联关系和包含关系，可以看做是软件架构的三维逻辑视图。

展示模块 130 可以向用户展示如图 8A 或图 8B 所示的三维逻辑视图，用户在查看该三维逻辑视图时，还可以对三维逻辑视图执行一些操作，该操作包括但不限于：旋转、移动、缩放。构建模块 120 响应于用户的操作，可以对三维逻辑视图中功能组件对应的立体图形进行旋转、平移、缩放等操作。构建模块 120 可以对三维逻辑视图中功能组件对应的立体图形进行旋转、移动的方式可以参见前述内容，此处不再赘述。

由于如图 8A 或图 8B 所示的三维逻辑视图展示了功能组件中的资源，功能组件对应的长方体的放大或缩小表明该功能组件中资源发生变化，下面对缩放操作进行说明。

用户可以通过光标对三维逻辑视图中功能组件对应的立体图形执行放大或缩小操作，响应于该放大或缩小操作，构建模块 120 可以按照光标的移动方向和移动位置，放大或缩小该功能组件对应的立体图形，变更该功能组件所占用的计算资源。用户还可以通过光标对功能组件对应的立体图形中的三维图形执行放大或缩小操作，构建模块 120 响应于该放大或缩小操作，可以按照光标的移动方向和移动位置，放大或缩小该功能组件对应的立体图形中的三维图形。通过缩小或放大该三维图形，用户可以更加清楚的查看三维逻辑视图，还可以变更该功能组件中实例所占用的计算资源。

作为一种可能的实施方式，三维逻辑视图中每个功能组件对应的立体图形，以及该立体图形中包括的三维图形设置有缩放操作区域，光标移动至缩放操作区域，用户触发光标在该缩放操作区域的移动方向可以指示该功能组件对应的立体图形或该立体图形中的三维图形放大或缩小，光标在该缩放操作区域的移动位置可以指示该功能组件对应的长方体或该立体图形中的三维图形放大或缩小的程度。例如，三维视图中的立方图形或三维图形的各条边以及各个顶点可以设置为缩放操作区域。三维视图中的立体图形或三维图形（如长方体）的一条边的移动指示该缩小或放大操作为立体图形或三维图形的一个维度上的放大或缩小，表明立体图形所表征的功能组件所占用的一种计算资源发生变化，或三维图形所占用的一种计算资源发生变化。例如，在长方体的长度、宽度或高度的方向进行放大。光标在各个顶点的移动指示该缩小或放大操作为等比例缩小或放大，也就是说，该立体图形的各个面均进行同等程度的缩小或放大，表明立体图形所表征的功能组件所占用的所有计算资源发生变化，或三维图形所占用的所有计算资源发生变化。

针对如图 8A 所示的三维逻辑视图，将功能组件对应的长方体的各条边以及各个顶点设置为缩放操作区域。若功能组件 1 中各个实例的处理器数量需要增加，参见图 8C，用

用户可以触发光标在功能组件 1 对应的该长方体的边 D (长方体的宽度方向的边) 向右移动, 移动前光标 801 和移动后光标 802 的位置参见图 8C, 构建模块 120 可以将该长方体进行放大, 增加该长方体的长度, 也就是增加边 E 的长度, 放大至光标停留的位置, 增加该功能组件对应的长方体上表面的面积。

5 针对如图 8A 所示的三维逻辑视图, 将功能组件对应的长方体的各条边以及各个顶点设置为缩放操作区域。若功能组件 1 中一个实例的内存的空间缩减, 参见图 8D, 用户可以触发光标在功能组件 1 对应的长方体中的一个小长方体的边 F (小长方体的长度方向的边) 向上移动, 移动前光标 803 和移动后光标 804 的位置参见图 8D, 构建模块 120 可以将该小长方体的下表面向上移动, 缩小至光标停留的位置, 减少该功能组件对应的长方体中的一个小长方体的高度。

10 针对如图 8B 所示的三维逻辑视图, 将功能组件对应的长方体的各条边以及各个顶点设置为缩放操作区域。若功能组件 1 中一个实例的处理器数量需要增加, 参见图 8E, 用户可以触发光标该实例对应的小长方体的边 G (小长方体的长度方向的边) 向下移动, 移动前光标 805 和移动后光标 806 的位置参见图 8E, 构建模块 120 可以将该小长方体在宽度方向向上进行放大, 增加该长方体的宽度, 放大至光标停留的位置。

15 针对如图 8B 所示的三维逻辑视图, 将功能组件对应的长方体的各条边以及各个顶点设置为缩放操作区域。若功能组件 1 中一个实例的内存的空间缩减, 参见图 8F, 用户可以触发光标在该实例对应的小长方体的边 P (小长方体的宽度方向的边) 向左移动, 移动前光标 807 和移动后光标 808 的位置参见图 8F, 构建模块 120 可以将该小长方体在长度方向向上进行缩小, 减少该长方体的长度, 缩小至光标停留的位置。

20 本申请实施例还提供了三维运行视图以及三维部署视图, 下面结构图 9 和图 10 分别进行介绍:

25 如图 9 所述, 为本申请实施例提供的一种三维运行视图, 在如图 8A 所示的三维逻辑视图的基础上, 可以变更代表功能组件对应的长方体、或实例对应的小长方体的颜色, 利用不同颜色表征功能组件的不同运行状态, 或实例的不同的运行状态。

图 9 中, 白色表示功能组件运行正常, 黑色条纹表示功能组件运行缓慢, 运行缓慢的原因该功能组件的磁盘空闲空间达到临近值, 黑色表示功能组件运行故障。从图 9 中可以较为直接的获知功能组件 1 的实例运行正常, 功能组件 2 中的一个实例运行故障, 两个实例运行正常, 功能组件 3 中的两个实例运行缓慢, 一个实例运行正常。

30 需要说明的是, 图 9 中利用颜色表征功能组件的运行状态的方式仅是举例, 在实际应用中, 颜色可以不局限在白色、黑色条纹以及黑色, 运行状态也可以不局限在正常、缓慢以及故障。例如, 可以利用同种颜色的明暗程度表征功能组件的运行效率。

35 如图 10 所述, 为本申请实施例提供的一种三维部署视图, 在如图 10 所示的三维逻辑视图的基础上, 可以增加能够表示功能组件部署位置的三维图形, 利用功能组件所在的三维图形表示功能组件所在的位置。

图 10 中增加了表征服务器的三维图形, 功能组件 1 部署在服务器 1, 功能组件 2 和功能组件部署在服务器 2。

40 展示模块 130 可以向用户展示如图 9 所示的三维运行视图以及如图 10 所示的三维部署视图时, 用户在查看该三维运行视图或三维部署视图时, 也可以对该三维运行视图或三维部署视图执行旋转、移动、缩放等操作, 构建模块 120 响应于用户的操作, 可以对三维

运行视图或三维部署视图中功能组件对应的立体图形进行旋转、移动、缩放。构建模块 120 可以对三维运行视图或三维部署视图中功能组件对应的立体图形进行旋转、移动、缩放的方式与构建模块 120 可以对三维逻辑视图中功能组件对应的立体图形进行旋转、移动、缩放的方式相同，具体可以参见前述内容，此处不再赘述。

需要说明的是，三维视图中除了逻辑视图、运行视图以及部署视图，还可以提供实体装置与功能组件的映射视图。当软件架构中的功能组件与实体装置有对应关系是指该功能组件的功能实际应用中，可以通过该实体装置实现，映射视图中可以绘制该实体装置以及存在对应关系的功能组件，利用实体装置与功能组件之间的连接表征功能组件与实体装置之间的对应关系。如图 11 所示，为本申请实施例提供的一种三维映射视图，该三维映射视图中绘制服务器机箱的三维结构，功能组件 1、功能组件 2 以及功能组件 3 与该服务器机箱之间存在对应关系，在图 11 中，利用功能组件与服务器机箱之间的箭头表明该对应关系。

前述内容，对二维视图、三维视图展示软件架构的方法，以及三维视图的变更方法进行了说明。下面对二维视图和三维视图的构建方法进行说明。二维视图和三维视图的构建方法包括两方面，一方面为软件架构的视图的初步构建方法，另一方面软件架构的视图的更新方法。其中，软件架构的视图的更新方法可以分为基于更新信息的更新方法，以及根据更新的一种视图更新另一种视图的方法。

下面结合图 12 对本申请实施例提供软件架构的视图的初步构建方法进行说明，如图 12 所示，该方法包括：

步骤 1201：接收模块 110 接收软件的架构信息，该架构信息中可以包括软件中的功能组件的名称、功能组件之间的关联关系。

该架构信息可以是其他设备发送给接收模块 110 的，例如，其他设备可以向接收模块 110 发送视图展示请求，该视图展示请求用于请求软件的视图，视图展示请求携带有该软件的架构信息。

该架构信息也可以是在用户的触发下所产生的，例如，用户可以在一个界面输入该软件的架构信息，接收模块 110 在检测到用户的输入操作后，接收该架构信息。

该架构信息中除了软件中包括的功能组件、功能组件之间的关联关系之外，还可以包括其他信息，该架构信息中包括的信息与视图所需展示的信息有关。

举例来说，若后续只需展示软件的逻辑视图，该架构信息中可以包括该软件中各个功能组件的名称、各个功能组件之间的关联关系。若软件中功能组件之间存在包含关系，该架构信息中还可以包括功能组件之间的包含关系。可选的，还可以包括功能组件的结构信息，该结构信息指示该功能组件的内部结构，例如功能组件中包括实例的数量、实例的数量、处理器的数量、所能占用的磁盘空间的大小等。

若后续只需展示软件的运行视图，该架构信息中可以包括该软件中各个功能组件的名称、功能组件之间的关联关系、各个功能组件的运行状态。若软件中功能组件之间存在包含关系，该架构信息中还可以包括功能组件之间的包含关系。该架构信息中还可以包括功能组件中的结构信息。

若后续只需展示软件的部署视图，该架构信息中可以包括该软件中各个功能组件的名称、功能组件之间的关联关系、各个功能组件的位置信息。若软件中功能组件之间存在包含关系，该架构信息中还可以包括功能组件之间的包含关系。

若后续需展示软件的逻辑视图、运行视图以及部署视图，该架构信息可以包括各个功

能组件的名称、各个功能组件之间的关联关系、各个功能组件的运行状态以及各个功能组件的位置信息。若软件中功能组件之间存在包含关系，该架构信息中还可以包括功能组件之间的包含关系。

步骤 1202: 构建模块 120 根据该架构信息构建二维视图，并将二维视图转换为三维视图。

构建模块 120 获取该架构信息，可以按照视图构建的难易程度，先构建二维视图，之后将二维视图转换为三维视图。构建模块 120 也可以按照视图展示信息的丰富程度，先构建三维视图，之后将三维视图转换为二维视图。

下面对二维视图、三维视图的构建以及两种视图之间的转换方式进行说明：

(1) 构建二维视图。

构建模块 120 可以读取架构信息，确定该软件中包括功能组件的数量以及各个功能组件的名称，构建相同数量的二维图形，每个二维图形代表一个功能组件（也就是说，每个功能组件对应一个二维图形），并在每个二维图形上添加上对应的功能组件的名称。

构建模块 120 还可以通过架构信息确定功能组件之间的关联关系，在存在关联关系的功能组件所对应的二维图形间增加连接线。

若该架构信息中还包括各个功能组件的运行状态，构建模块 120 可以在每个功能组件对应的二维图形上添加描述该功能组件的运行状态的文字信息。

若该架构信息中还包括各个功能组件的位置信息，构建模块 120 可以在每个功能组件对应的二维图形上添加描述该功能组件的位置信息的文字信息。

(2) 构建三维视图。

构建模块 120 可以读取架构信息，确定该软件中包括功能组件的数量以及各个功能组件的名称，构建相同数量的立体图形，每个立体图形代表一个功能组件（也就是说，每个功能组件对应一个立体图形），并在每个立体图形上添加上对应的功能组件的名称。

构建模块 120 还可以通过架构信息确定功能组件之间的关联关系，在存在关联关系的功能组件所对应的立体图形间增加连接线。

若该架构信息中可以包括功能组件中的结构信息。构建模块 120 可以根据功能组件中的结构信息对该功能组件所对应的立体图形进行划分，利用划分出空间用于表征该功能组件的结构。例如，功能组件中包括三个实例，构建模块 120 可以将该功能组件所对应的立体图形均分为三个小的立体图形，例如每个小的立体图形表征一个实例。

若该架构信息中还包括各个功能组件的运行状态，构建模块 120 可以变更每个功能组件对应的立体图形的颜色，将该功能组件的颜色变更为该功能组件的运行状态所对应的颜色。

若该架构信息中还包括各个功能组件的位置信息，构建模块 120 可以增加描述位置的三维图形，将位置信息相同的功能组件所对应的三维图形放置的同一个三维图形中，还可以根据功能组件的位置信息设置该三维图形的名称。

(3)、二维视图和三维视图的转换。

构建模块 120 在构建了二维视图后，可以根据二维视图中的二维图形转换为立体图形，若二维视图中包括用于描述功能组件运行状态的文字信息，构建模块 120 可以根据该文字信息将该功能组件对应的立体图形的颜色变更该功能组件运行状态所对应的颜色。若二维视图中包括用于描述功能组件位置信息的文字信息，构建模块 120 可以根据该文字信息，

增加三维图形，将位置信息相同的功能组件对应的立体图像放置同一个三维图形中。由此完成二维视图到三维视图的转换。

需要说明的是，当架构信息中还包括功能组件的结构信息，由于二维视图中没有展示功能组件的结构信息。构建模块 120 可以读取架构信息中的功能组件的结构信息，根据功能组件中的结构信息对该功能组件所对应的立体图形进行划分，利用划分出的空间（如三维图形）用于表征该功能组件的结构。

（4）、三维视图和二维视图的转换。

构建模块 120 在构建了三维视图后，可以根据三维视图中的立体图形转换为二维图形，若三维视图中各个功能组件所对应的立体图形具备不同的颜色，构建模块 120 可以根据功能组件所对应的立体图形的颜色确定该功能组件的运行状态，在二维图形上添加描述根据该运行状态的文字信息。若三维视图包括了用于描述功能组件的位置的三维图形，构建模块 120 可以删除该三维图形，在该三维图形中包括的立体图形转换后的二维图形上添加文字信息，添加的文字信息可以是该三维图形的名称。由此完成三维视图到二维视图的转换。

步骤 1203：展示模块 130 在用户的触发下展示二维视图或三维视图。

构建模块 120 在构建了二维视图以及三维视图之后，可以根据用户的触发展示该二维视图或三维视图。

示例性的，展示模块 130 可以向用户提供二维视图和三维视图这两种展示选项，如图 13 所示的界面所示，用户可以根据自身的需求，选择二维视图或三维视图，展示模块 130 在接收到用户的触发后，可以在展示区域显示用户所选择的视图。

作为一种可能的实施方式，展示模块 130 可以向用户提供多种展示选项，例如展示模块 130 可以向用户提供如图 14 所示的显示界面，该界面中提供了二维视图、三维逻辑视图、三维部署视图、三维运行视图。以使用户可以有针对性的查看软件架构。

用户可以根据自身需要选择所需的视图，用户在点击了相应的选项之后，在视图展示区域可以展示所选择的视图。

通常，软件架构是可以发生变更的。例如，对软件的架构信息进行更新（如软件进行优化、重组等操作）时，软件架构中包括的功能组件的数量、功能组件之间的关联关系等会发生变化。为了能够准确的显示软件架构，当软件架构发生变化时，软件架构的视图也可以发生一定的更新。又例如，当三维视图和二维视图中一个视图发生更新，三维视图和二维视图中的另一个视图也需要发送更新。

下面结合图 15 对本申请实施例提供软件架构的视图的更新方法中进行说明，如图 15 所示，该方法包括：

步骤 1501：接收模块 110 接收软件的架构的更新信息，该更新信息中包括软件架构中所变更的功能组件，可选的，该更新信息中还可以包括变更的关联关系。

该更新信息携带的信息与软件发生的变更有关，下面列举该更新信息中可能包括的几类信息：

1、当软件架构中的功能组件发生变更时，该更新信息中可以包括变更的功能组件的名称以及变更的方式。

在本申请实施例中，功能组件的变更方式包括但不限于删除功能组件、添加功能组件。当删除功能组件时，该更新信息中可以包括删除的功能组件的名称。当添加功能组件时，该更新信息中可以包括添加的功能组件的名称。可选的，该更新信息中还可以包括添加的

功能组件与其他功能组件的关联关系。

2、当软件架构中的功能组件的运行状态发生变更时，该更新信息中可以包括变更的功能组件的名称、以及变更后该功能组件的运行状态。

3、当软件架构中的功能组件的部署位置发生变更时，该更新信息中可以包括变更的功能组件的名称、以及变更后该功能组件的位置信息。

4、当软件架构中的功能组件的结构发生变更时，该更新信息中可以包括变更的功能组件的名称、以及变更后该功能组件的结构信息。

例如，可以变更该功能组件中所占用计算资源，如增加或减少实例、增加或减少实例所能占用的磁盘空间、处理器或内存等。

接收模块 110 接收该更新信息的方式与接收模块 110 接收该架构信息的方式相同，具体可参见前述内容，此处不再赘述。

作为一种可能的实施方式该更新信息，也可以是构建模块 120 在检测到用户对二维视图或对三维视图执行一些操作所产生的，如用户对二维视图或三维视图进行移动、缩放操作时，构建模块 120 在检测到该操作后，生成软件的架构信息，生成该软件的更新信息。

步骤 1502: 构建模块 120 根据该更新信息更新二维视图和三维视图。构建模块 120 根据该更新信息更新二维视图和三维视图时，构建模块 120 可以先更新二维视图，之后利用更新后二维视图更新三维视图。构建模块 120 可以先更新三维视图，之后利用更新后三维视图更新二维视图。构建模块 120 也可以分别更新二维视图和三维视图。

更新信息中包括的信息不同，构建模块 120 更新二维视图和三维视图的方式也不同，下面分别进行说明：

该更新信息携带的信息与软件发生的变更有关，下面列举该更新信息中可能包括的几类信息：

1、该更新信息包括变更的功能组件的名称以及变更的方式。

对于二维视图，当变更为删除功能组件，构建模块 120 可以在二维视图中删除该功能组件所对应的二维图形，还可以删除该功能组件所对应的二维图形与其他功能组件所对应的二维图形之间的连接线。

对于三维视图，当变更为删除功能组件，构建模块 120 可以在三维视图中删除该功能组件所对应的立体图形，还可以删除该功能组件所对应的立体图形与其他功能组件所对应的立体图形之间的连接线。

对于二维视图，当变更为添加功能组件，构建模块 120 可以在二维视图中添加该功能组件所对应的二维图形，还可以增加该功能组件所对应的二维图形与其他功能组件所对应的二维图形之间的连接线。

对于三维视图，当变更为添加功能组件，构建模块 120 可以在三维视图中添加该功能组件所对应的立体图形，还可以增加该功能组件所对应的立体图形与其他功能组件所对应的立体图形之间的连接线。

对于三维视图，当变更为调节功能组件所占用的计算资源或功能组件中实例所占用的计算资源，构建模块 120 可以在三维视图中调整该功能组件所对应的立体图形的大小（如调整立体图形的横截面积、高度、长度、宽度），该实例所对应的三维图形的大小（如调整三维图形的横截面积、高度、长度、宽度）。

2、该更新信息中包括变更的功能组件的名称、以及变更后该功能组件的运行状态。

对于二维视图, 构建模块 120 可以在二维视图中根据变更后该功能组件的运行状态更新描述该功能组件运行状态的文字信息。

对于三维视图, 构建模块 120 可以在三维视图中根据变更后该功能组件的运行状态更新该功能组件所对应的立体图形的颜色。

5 3、该更新信息中包括变更的功能组件的名称、以及变更后该功能组件的位置信息。

对于二维视图, 构建模块 120 可以在二维视图中根据变更后该功能组件的位置信息更新描述该功能组件位置信息的文字。

对于三维视图, 构建模块 120 可以在三维视图中根据变更后该功能组件的位置信息更新该功能组件所对应的立体图形所在的三维图像的名称。

10 前述内容介绍了分别更新三维视图和二维视图的方式, 下面对构建模块 120 利用更新后的二维视图更新三维视图, 以及利用更新后的三维视图更新二维视图的方式进行说明:

(1)、构建模块 120 利用更新后的二维视图更新三维视图。

1、更新后的二维视图中功能组件发生变更, 如删除功能组件或添加功能组件。

15 构建模块 120 确定在更新后的二维视图中删除的功能组件, 在三维视图中删除相同功能组件所对应的立体图形, 还可以删除该功能组件所对应的立体图形与其他功能组件所对应的立体图形之间的连接线。

20 构建模块 120 确定在更新后的二维视图中添加的功能组件, 在三维视图中添加相同功能组件所对应的立体图形, 还可以根据二维视图中记录该功能组件与其他功能组件的关联关系, 在三维视图中增加该功能组件所对应的立体图形与其他功能组件所对应的立体图形之间的连接线。

2、更新后的二维视图中功能组件的运行状态发生变更。

构建模块 120 确定更新后的二维视图中变更的更新描述功能组件运行状态的文字信息, 根据该文字信息在三维视图中更新相同功能组件所对应的立体图形的颜色。

3、更新后的二维视图中功能组件的位置信息发生变更。

25 构建模块 120 确定更新后的二维视图中变更的描述功能组件位置信息的文字信息, 根据该文字信息, 在三维视图中更新该功能组件所对应的立体图形所在的三维图形的名称。

(2)、构建模块 120 利用更新后的三维视图更新二维视图。

1、更新后的三维视图中功能组件发生变更, 如删除功能组件或添加功能组件。

30 构建模块 120 确定更新后的三维视图中删除的功能组件, 在二维视图中删除相同功能组件所对应的二维图形, 还可以删除该功能组件所对应的二维图形与其他功能组件所对应的二维图形之间的连接线。

35 构建模块 120 确定更新后的三维视图中添加的功能组件, 在二维视图中添加相同功能组件所对应的二维图形, 还可以根据三维视图中记录该功能组件与其他功能组件的关联关系, 在二维视图中增加该功能组件所对应的二维图形与其他功能组件所对应的二维图形之间的连接线。

2、更新后的三维视图中功能组件的运行状态发生变更。

构建模块 120 确定更新后的三维视图中变更的功能组件所对应的立体图形的颜色, 根据该颜色修改二维视图中, 描述该功能组件运行状态的文字信息。

3、更新后的三维视图中功能组件的位置信息发生变更。

40 构建模块 120 确定更新后的三维视图中变更的功能组件所对应的立体图形所在的三维

图形的名称, 根据该三维图形的名称修改二维视图中, 描述该功能组件位置信息的文字。

需要说明的是, 当三维视图中功能组件所占用计算资源发生变更时, 如前述内容中图 8C 至图 8F 所示的变更。当二维视图可以展示功能组件的内部结构, 则根据三维视图中功能组件对应的立体图形的变更方式或该立体图形中三维图形的变更方式更新二维视图中相同功能组件对应的二维图形。例如, 如三维视图中功能组件对应的立体图形的高度增大, 功能组件包括的实例数量增大, 可以增加二维视图中相同功能组件对应的线性框的数量。但若二维视图未展示功能组件的内部结构, 则可以不更新三维视图。

步骤 1503: 展示模块 130 在用户的触发下展示更新后的三维视图或二维视图。

构建模块 120 对二维视图和三维视图更新之后, 展示模块 130 可以执行步骤 1503, 展示模块 130 执行步骤 1503 的方式与展示模块 130 执行步骤 903 的方式相同, 具体可参见前述内容, 此处不再赘述。

基于与方法实施例同一发明构思, 本申请实施例还提供了一种计算机集群, 用于执行上述方法实施例所示的方法, 相关特征可参见上述方法实施例, 此处不再赘述, 如图 16 所示, 为本申请实施例提供的一种计算机集群, 该计算机集群中包括多个计算设备 1600, 每个计算设备 1600 间通过通信网络建立通信通路。

每个计算设备 1600 中包括总线 1601、处理器 1602、通信接口 1603 以及存储器 1604, 可选的, 计算设备 1600 中还可以包括显示屏 1605。处理器 1602、存储器 1604 和通信接口 1603 之间通过总线 1601 通信。

其中, 处理器 1602 可以由一个或者多个通用处理器构成, 例如中央处理器 (central processing unit, CPU), 或者 CPU 和硬件芯片的组合。上述硬件芯片可以是专用集成电路 (application-specific integrated circuit, ASIC)、可编程逻辑器件 (programmable logic device, PLD) 或其组合。上述 PLD 可以是复杂可编程逻辑器件 (complex programmable logic device, CPLD)、现场可编程逻辑门阵列 (field-programmable gate array, FPGA)、通用阵列逻辑 (generic array logic, GAL) 或其任意组合。

存储器 1604 可以包括易失性存储器 (volatile memory), 例如随机存取存储器 (random access memory, RAM)。存储器 1604 还可以包括非易失性存储器 (non-volatile memory, NVM), 例如只读存储器 (read-only memory, ROM), 快闪存储器, 硬盘 (hard disk drive, HDD) 或固态硬盘 (solid-state drive, SSD)。存储器 1604 还可以包括上述种类的组合。

存储器 1604 中存储有可执行代码, 处理器 1602 可以读取存储器 1604 中的该可执行代码实现功能, 还可以通过通信接口 1603 与其他计算设备进行通信, 处理器 1602 还可以触发显示屏 1605 向用户显示信息, 如本申请实施例中如图 2A 至图 11 中的信息。在本申请实施例中, 处理器 1602 可以实现展示装置 100 的一个或多个模块 (如接收模块 110、构建模块 120、以及展示模块 130 中的一个或多个模块) 的功能, 这种情况下, 存储器 1604 中存储有展示装置 100 的一个或多个模块 (如接收模块 110、构建模块 120、以及展示模块 130 中的一个或多个模块)。

在本申请实施例中, 多个计算设备 1600 中的处理器 1602 可以协调工作, 执行本申请实施例提供的支持多变更的软件代码变更方法。

如图 17 所示, 为本申请实施例提供的一种系统架构, 该系统架构中包括客户端 200 和部署有变更装置的云端设备 300, 客户端 200 与云端设备 300 通过网络连接, 该云端设备 300 位于云环境中, 可以是部署在云数据中心中的服务器或者虚拟机, 图 17 中, 仅是

以该展示装置部署在一个云端设备 300 为例,作为一种可能的实施方式,该展示装置可以分布式地部署在多个云端设备 300 上。

如图 17 所示,客户端 200 包括总线 201、处理器 202、通信接口 203、存储器 204 以及显示屏 205。处理器 202、存储器 204 和通信接口 203 之间通过总线 201 通信。其中,处理器 202 和存储器 204 的类型可以参见处理器 1602 以及存储器 1604 的相关说明,此处不再赘述。存储器 204 中存储有可执行代码,处理器 202 可以读取存储器 204 中的该可执行代码实现功能。处理器 202 还可以触发显示屏 205 向用户显示信息,如本申请实施例中如图 2A 至图 11 中的信息。处理器 202 还可以通过通信接口 203 与云端设备进行通信。例如处理器 202 可以通过显示屏 205 提示用户输入软件的架构信息,通过通信接口 203 将架构信息反馈给云端设备 300。还可以通过通信接口 203 接收三维视图或二维视图,并通过显示屏 205 展示三维视图或二维视图。又例如处理器 202 可以通过显示屏 205 检测用户对三维视图的操作,产生更新信息,通过通信接口 203 将更新信息反馈给云端设备 300。还可以通过通信接口 203 接收调整后的三维视图或二维视图,并通过显示屏 205 展示调整后的三维视图或二维视图。

如图 17 所示,云端设备 300 包括总线 301、处理器 302、通信接口 303 以及存储器 304。处理器 302、存储器 304 和通信接口 303 之间通过总线 301 通信。其中,处理器 302 和存储器 304 的类型可以参见处理器 1602 以及存储器 1604 的相关说明,此处不再赘述。存储器 304 中存储有可执行代码,处理器 302 可以读取存储器 304 中的该可执行代码实现功能,还可以通过通信接口 303 与客户端 200 进行通信。在本申请实施例中,处理器 302 可以实现展示装置 100 的功能,这种情况下,存储器 304 中存储有展示装置 100 的接收模块 110、构建模块 120、以及展示模块 130 中的一个或多个模块。

处理器 302 通过通信接口 303 从客户端 200 接收到软件的架构信息(或更新信息)后,可以调用存储器 304 中存储的模块实现本申请实施例提供的软件架构的可视化方法,生成三维视图,处理器 302 可以通过通信接口 303 向客户端 200 发送三维视图。