(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 August 2014 (07.08.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/120207 A1

(51) International Patent Classification:
G06F 11/30 (2006.01) *G06F 11/36* (2006.01)

(21) International Application Number:
PCT/US2013/024154

(22) International Filing Date:
31 January 2013 (31.01.2013)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; Hewlett-Packard Development Company, L.P., 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **KOGAN-KATZ, Olga**; St. Shabai 19, 5623024 Yehud (IL). **HOROVITZ, Yair**; 19 Shabazi St., 56100 Yehud (IL).

(74) Agents: **MCKINNEY, Jack, H.** et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 East Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: DEPENDENCY MONITORING

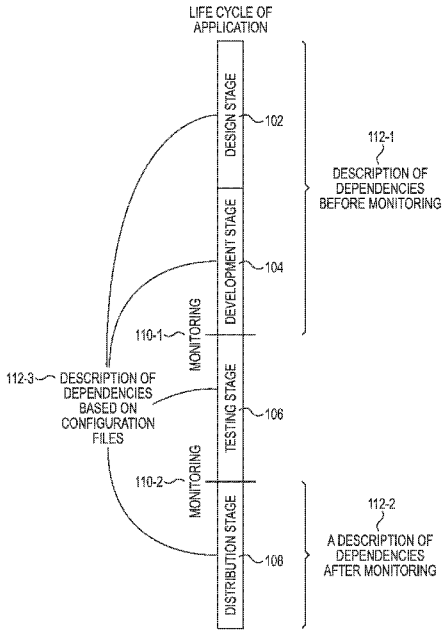

Fig. 1

(57) Abstract: Dependency monitoring can include monitoring a first application and a second application for un-expected behavior. Dependency monitoring can also include receiving a description of a number of dependencies between a number of applications wherein the description of the number of dependencies is created before monitoring of the first application and the second application begins. Dependency monitoring can include sending a message to an information technology (IT) personnel, wherein the message identifies a dependency from the number of dependencies between the first application and the second application based on the description of the number of dependencies.

WO 2014/120207 A1

Published:

— *with international search report (Art. 21(3))*

DEPENDENCY MONITORING

Background

[0001] An application can be monitored. The monitoring of an application can reveal a number of errors. The errors can be corrected by information technology (IT) personnel. Identifying the errors based on the monitoring can consume IT personnel resources.

Brief Description of the Drawings

[0002] Figure 1 is a diagram illustrating an example of a number of dependency descriptions according to the present disclosure.

[0003] Figure 2 is a diagram illustrating an example of a number of dependencies according to the present disclosure.

[0004] Figure 3 is a diagram illustrating an example of a number of events according to the present disclosure.

[0005] Figure 4 is a flow chart illustrating an example of a method for dependency monitoring according to the present disclosure.

[0006] Figure 5 is a diagram illustrating an example of a computing system according to the present disclosure.

Detailed Description

[0007] A number of applications can be associated with un-expected behavior. A number of events can be created based on the un-expected behavior. A portion of the number of events can be associated based on a dependency between the number of applications, a number of services, and/or a number of application components within an application, wherein a description of the dependencies is created during a design stage and/or during a

development stage of an application. The association between a portion of the number of events can be sent to an information technology (IT) personnel who can use the association to resolve the un-expected behavior that caused the events.

[0008] As used herein, an application can be machine-readable instructions (MRI) that are composed of a number of application components, and that provide a number of services to a user. In a number of examples, a service can be provided using a service-oriented architecture (SOA) wherein a number of applications and/or a number of application components provide the service. A number of services can include external services provided to users and/or internal services provided to a number of other applications. For example, services can include database services that an application provides to a number of other applications. That is, a service can be dependent on a number of applications and/or on a number of application components.

[0009] As used herein, an application component can define a function, e.g., a service, associated with an application. A function associated with an application can include a service that an application provides to a user. An application that includes a number of application components can perform a number of functions that provide a number of services to a user. In a number of examples, the application components in an application can depend on each other.

[0010] An application can have an associated life cycle. A life cycle can describe a number of stages that the application goes through before the application can be used by a number of users. A life cycle of an application can include a design stage, a development stage, a testing stage, a distribution stage, among other stages.

[0011] An application can be associated with un-expected behavior and/or abnormal behavior. Un-expected behavior can include un-intended behavior of an application and/or an application component. Abnormal behavior can include behavior that is outside the scope of a predefined behavior. For example, an application that has a number of errors associated with the MRI can have un-intended behavior as a result of the errors. Un-intended behavior

and/or abnormal behavior can be made manifest in a number of metrics that are monitored. The monitoring of the number of metrics that are associated with the applications can lead to the creation of a number of events. For example, if a performance threshold, e.g., metric, of a certain transaction defines that the transactions should be performed in under three seconds, then an event can be created, e.g., thrown, when the average response time is greater than three seconds. The number of events can be created when un-intended behavior and/or abnormal behavior is detected during a monitoring of the application.

[0012] As the number of events increase, the ability of IT personnel to identify the root cause of the un-expected behavior decreases. As used herein, a root cause of the un-expected behavior can be located in the MRI, a runtime configuration, and/or user error associated with an application. In a number of examples, an error in a first application can cause un-expected behavior in the first application and in a second application. An error in a first application component can cause an un-expected behavior in the first application component and in a second application component of an application. An error associated with a first service that is provided by a number of application components and/or applications can cause an un-expected behavior in the first application and in a second service. The events created based on the un-expected behavior of the first application, the second application, the first service, the second service, the first application component, and/or the second application component can be associated. That is, the resolution of a root cause of the un-expected behavior in the first application can resolve the un-expected behavior in the second application, the first service, the second service, the first application component, and/or the second application component.

[0013] IT personnel can address the un-expected behaviors, e.g., root cause of the un-expected behaviors, associated with a number of applications more effectively by grouping events than by addressing each event individually. The grouping of events can be based on a number of dependencies between a number of applications, a number of services, and/or between a number of components of an application. A description of a number of dependencies can

provide IT personnel with the information needed to identify a root cause of a number of events. Descriptions of dependencies that are created at a design stage and a development stage, e.g., before monitoring, can be more accurate and can be more easily acquired than descriptions of dependencies that are created at runtime. As used herein, examples given for dependencies between different applications can also describe examples for dependencies between different service and/or different application components of an application.

[0014] In the present disclosure, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration how a number of examples of the disclosure can be practiced. These examples are described in sufficient detail to enable those of ordinary skill in the art to practice the examples of this disclosure, and it is to be understood that other examples can be used and that process, electrical, and/or structural changes can be made without departing from the scope of the present disclosure.

[0015] The figures herein follow a numbering convention in which the first digit corresponds to the drawing figure number and the remaining digits identify an element or component in the drawing. Elements shown in the various figures herein can be added, exchanged, and/or eliminated so as to provide a number of additional examples of the present disclosure. In addition, the proportion and the relative scale of the elements provided in the figures are intended to illustrate the examples of the present disclosure, and should not be taken in a limiting sense.

[0016] Figure 1 is a diagram illustrating an example a number of dependency descriptions according to the present disclosure. Figure 1 includes a design stage 102, a development stage 104, a testing stage 106, and a distribution stage 108. Figure 1 also describes when a number of descriptions of a number of dependencies can be created.

[0017] A design stage 102 of an application can include the conception and the creation of a design for an application. A design of an application can define a number of dependencies that are associated with an application. For example, an application that offers Internet related services, e.g., a first service, can be dependent on a database, e.g., a second service, to store user

information or a first component of an application that provides a login service can be dependent on a second component of the application that provides graphical user interface (GUI) services. A design of the Internet related application can defines the application's dependency on a database. A design of a first component of an application can define the first component's dependency on a second component of the application. An application design can be a description of the dependency. A description can be a file that describes a dependency between a number of applications, e.g., Internet related application and database application, a number of services, and/or a number of application components.

[0018] A development stage 104 can include the creation of an application. For example, in the development stage 104 a programmer, e.g., IT personnel that creates the application, can create the MRI that are associated with the application. During the creation of the MRI, e.g., application, a programmer can include a number of dependencies in the MRI. For example, during the development of an application a programmer can include instructions in the MRI that define a dependency on a different application, a different service, and/or different components of the application. A file that includes the MRI can be a description of a number of dependencies. In a number of examples, a programmer can annotate the dependencies in a separate file, e.g., separate from the MRI, wherein the second file is a description of the dependencies.

[0019] A testing stage 106 can include the testing of the application to determine whether the application functions according to an expected behavior. During the testing stage 106 a state of an application can undergo a number of changes. Changes that are made to an application during the testing stage 106 of the application's life cycle can be classified as changes made during the development stage 104. For example, during the testing stage 106 an error, e.g., portion of the MRI that does not function according to the expected behavior of the application, can be detected. After the detection of the error, the application can re-enter the development stage 104 wherein the changes are

made. The application can re-enter the testing stage 106 once the error has been corrected.

[0020] An application can be first monitored 110-1 during the testing stage 106. The first monitoring 110-1 of an application during the testing stage can identify a number of un-expected behavior's in the application. In a testing stage 106, the monitoring can be performed by an IT personnel that performs a number of tests on the application and/or by a computing device that determines whether the application functions properly. A computing device can determine that the application functions properly by monitoring a number of metrics that are associated with the application. A number of metrics can include a central processing unit (CPU) usage, a memory usage, and/or a network usage, among other metrics.

[0021] A distribution stage 108 can include the distribution of the application to a consumer, e.g., user. That is, the design stage 102, the development stage 104, and the testing stage 106 can result in a finished application which can be distributed to a customer during a distribution stage 108. After the distribution stage 108, the application can be monitored, e.g., second monitoring 110-2. The information gathered during the monitoring of the application can be sent to IT personnel in the form of events.

[0022] In a number of previous approaches, a description of a number of dependencies between a number of applications was created during the during the distribution stage 108. The description of the dependencies can be gathered through a reverse engineering process. A reverse engineering process can formulate the dependencies by observing how the application functions during an execution of the application, e.g., MRI, without referencing the design of the application and/or the MRI that are associated with the application. For example, if a first application consistently sends network traffic to the same IP address, then there is a probability that there is a dependency between a second application that receives Internet traffic at the IP address and the first application. However, reverse engineering does not guarantee a dependency but rather describes a likelihood that a dependency exists between two applications.

[0023] In a number of examples, a first description 112-1 of dependencies between a number of applications, the number of services, and/or between components of an application can be created before the first monitoring 110-1 and the second monitoring 110-2. For example, the first description 112-1 of the dependencies can be created during a design stage 102 and/or during a development stage 104 before the first monitoring 110-1 and/or the second monitoring 110-2 of the application takes place. The second description 112-2 of the dependencies can be created after the first monitoring 110-1 and/or the second monitoring 110-2 of the application takes place. For example, the second description 112-2 can be created from a reverse engineering process.

[0024] In a number of examples, a third description 112-3 of the dependencies between a number of applications can be created based on a number of configuration files. A configuration file can provide an application with information needed by the application to function. Configuration files can be, for example, runtime configuration files. That is, at execution of the application the application can reference the configuration file to obtain settings that can be used by the application. For example, a web server application can be associated with a configuration file that defines an IP address that can be associated with the web server application. A configuration file can be conceived at a design stage 102. A configuration file can be created at a development stage 104. A configuration file can be referenced at a testing stage 106 and/or at a distribution stage 108. The configuration file can be a third description 112-3 of a number of dependencies.

[0025] The first description 112-1 of the dependencies that is created before the second monitoring 110-2 and the third description 112-3 that is created based on a configuration file can be more reliable than a second description 112-2 of the dependencies that is created after the second monitoring 110-2. The first description 112-1 created before the second monitoring 110-2 can be created by IT personnel that designed of the application and/or a programmer of the application. The third description 112-3 is based on a configuration file that was also created by a programmer and/or IT

personnel that designed the application. That is, the first description 112-1 and the third description 112-3 can be created by IT personnel that have a knowledge of how the application functions because the IT personnel designed and/or created the application. A second description 112-2 can be less reliable than a first description 112-1 because the second description 112-2 is created based on a probability and/or an observation and not on knowledge of a design of the application.

[0026] The first description 112-1 of the dependencies created before the second monitoring 110-2 of the application, the second descriptions 112-2 created after the second monitoring 110-2 of the application, and/or the third description 112-3 of the dependencies based on a configuration file can be used together to determine a number of dependencies between a number of applications, a number of services, and/or a number of components of an application. For example, if there is an equal probability based on a second description 112-2 that a dependency exists between a first application and a second application as compared to the probability that the dependency exists between the first application and a third application, then the first description 112-1 and the third description 112-3 can be used as a tie-breaker. Other derivations of the use of the three descriptions can be used to determine whether there is a dependency between a number of applications.

[0027] Figure 2 is a diagram illustrating an example of a number of dependencies according to the present disclosure. Figure 2 includes a first application 220-1, a second application 220-2, and a third application 220-3, e.g., referred to generally as applications 220. In a number of examples, the first application 220-1, the second application 220-3, and/or the third application 220-3 can represent a first application component, a second application component, and/or a third application component, respectively. The first application 220-1, the second application 220-3, and/or the third application 220-3 can represent a first service, a second service, and/or a third service, respectively.

[0028] A second application 220-2 can be dependent on a first application 220-1. A third application 220-3 can be dependent on a first

application 220-1. In Figure 2, a dependency 224-1 between the third application 220-3 and the first application 220-1 is represented by the line connecting the third application 220-3 and the first application 220-1. A dependency 224-2 between the second application 220-2 and the first application 220-1 is represented by the line connecting the second application 220-2 and the first application 220-1.

[0029] A dependency between the second application 220-2 and the third application 220-3 can be an indirect dependency. That is, a dependency between the second application 220-2 and the third application 220-3 can be defined by a dependency of the second application 220-2 and the third application 220-3 on the first application 220-1. For example, a dependency between the second application 220-2 and the third application 220-3 can include a dependency of the second application 220-2 on first application 220-1 and a dependency of the third application 220-3 on the first application 220-1.

[0030] A dependency, e.g., dependency 224-1 and/or dependency 224-2, can be associated with un-expected behavior. For example, an error and/or un-expected behavior in the first application 220-1 can cause an error and/or un-expected behavior in the second application 220-2 even through the second application 220-2 does not have an error in the MRI that are associated with the second application 220-2 and/or in an computing device associated with the second application 220-2. Resolving the error and/or addressing the un-expected behavior in the first application 220-1 can correct and/or resolve the un-expected behavior in the second application 220-2.

[0031] Figure 3 is a diagram illustrating an example of a number of events according to the present disclosure. Figure 3 includes a number of applications 320-1, 320-2, 320-3, 320-4, 320-5, 320-6, 320-7, 320-8, 320-9, 320-10, 320-11, and 320-12, e.g., referred to generally as applications 320. In Figure 3 a dependency between applications is represented by a line connecting the applications. For example, a dependency between the first application 320-1 and the second application 320-2 is represented by the line connecting the first application 320-1 and the second application 320-2. In a number of examples, the number of applications 320 can represent, a number

of services, and/or a number of application components and the number of dependencies between the different applications can represent dependencies between services, and/or dependencies between components of an application.

[0032] In a number of examples, dependencies between applications, services, and/or components of an application can be nested. For example, a seventh application 320-7 can depend on the first application 320-1 because the seventh application 320-7 depends on the third application 320-3 which depends on the first application 320-1.

[0033] A number of events 322-1, 322-2, 322-3, 322-4, 322-5, 322-6, 322-7, 322-8, 322-9, 322-10, 322-11, 322-12, e.g., referred to generally as events 322, can be associated with the applications 320. A monitoring service can create an event that is associated with one of the applications 320 when the applications 320 exhibit un-expected behavior. For example, a number of first events 322-1 can be created when a first application 320-1 requires higher CPU usage than expected.

[0034] An un-expected behavior can be defined by an upper threshold value and/or a lower threshold value. For example, a second application 320-2 can exhibit un-expected behavior when the application 320-2 uses 20 gigabytes (GB) of network traffic in an hour and when a threshold value of usage includes a lower threshold value of 5 GB and a higher threshold value of 10 GB of network traffic in an hour. That is, the second application 320-2 exhibits un-expected behavior because the second application 320-2 used more than 10 GB of network traffic in an hour.

[0035] The expected behavior of the second application 320-2 can be monitored by monitoring a number of computing devices that are associated with the second application 320-2. For example, a computing device that executes the MRI that defines the second application 320-2 can be monitored by determining the CPU usage of the computing device.

[0036] In Figure 3, the seventh application 320-7 and the sixth application 320-6 are dependent on the third application 320-3, the fifth applications 320-5 and the fourth application 320-4 are dependent on the second application 320-2, and the second application 320-2 and the third application 320-3 are

dependent on the first application 320-1. The twelfth application 320-12 is dependent on the tenth application 320-10, the eleventh application 320-11 is dependent on the ninth application 320-9, and the ninth application 320-9 and the tenth application 320-10 are dependent on the eighth application 320-8. Resolving an un-expected behavior in the first application 320-1 can resolve the un-un-expected behavior in the seventh application 320-7 but not the un-expected behavior in the twelfth application 320-12, for example, because the twelfth application 320-12 does not depend from the first application 320-1.

[0037] A number of descriptions can define the number of dependencies. For example, a first description of a first number of dependencies can define the dependencies between the applications 320-1, 320-2, 320-3, 320-4 , 320-5, 320-6, and 320-7, while a second description of a second number of dependencies can define the dependencies between the applications 320-8, 320-9, 320-10, 320-11, and 320-12.

[0038] In a number of examples, a first description can define a dependency between the first application 320-1 and third application 320-3 while a second description can define a dependency between the seventh application 320-7 and the third application 320-3. The first description and the second description can be joined and/or referenced together to define a dependency of the seventh application 320-7 on the first application 320-1.

[0039] The events 322 can be sent to IT personnel. The IT personnel can identify a root-cause of the un-expected behavior associated with the applications 320. However, the IT personnel's ability to identify a root-cause is diminished as the events 322 associated with the un-expected behavior grow.

[0040] In a number of examples, a first description of the dependencies between the seventh application 320-7 and the sixth application 320-6 can be created after the monitoring of the seventh application 320-7 and the sixth application 320-6 began. The first description of the dependencies can determine that the applications are equally likely to be dependent from the first application 320-1 or the eighth application 320-8. A second description of the dependencies between the seventh application 320-7 and the sixth application 320-6 can be created before the monitoring of the seventh application 320-7

and the sixth application 320-6. The second description can identify the seventh application 320-7 and the sixth application 320-6 as being dependent on the first application 320-1. The second description can be used to determine whether the seventh application 320-7 and the sixth application 320-6 depend from the first application 320-1 or the eighth application 320-8. That is, the first description of the dependencies and the second description of the dependencies can be used to identify that the sixth application 320-6 and the seventh application 320-7 depend from the first application 320-1.

[0041] In a number of examples, a number of descriptions of a number of dependencies can be created by different sources. For example, an IT personnel, e.g., first source, that designed the seventh application 320-7 can create a first description of the dependency of the seventh application 320-7 on the third application 320-3, while a programmer, e.g., second source, that developed the third application 320-3 can create a second description of the dependency of the third application 320-3 on the first application 320-1. The descriptions of dependencies from different sources can be sent to an IT personnel to assist the IT personnel in identifying a number of dependencies.

[0042] Figure 4 is a flow chart illustrating an example of a method for dependency monitoring according to the present disclosure. At 430, a first application and a second application can be monitored for un-expected behavior. Un-expected behavior can be caused by an error in the first application, the second application, and/or a different application. For example, un-expected behavior associated with a first application can be caused by an error in the first application when the cause of the un-expected behavior is not dependent on the second application and/or the different application. Un-expected behavior associated with a first application can be caused by an error in the second application when the first application depends from the second application. Un-expected behavior associated with a first application and a second application can be caused by an error in the different application when the first application and the second application depend on the different application.

[0043] At 432, a description of a number of dependencies between a number of applications is received, wherein the description of the number of dependencies is created before monitoring of the first application and the second application begins. A description of the number of dependencies between the number of application that is created at a design stage and/or a development stage can be a description that is created before the monitoring of the first application and the second application begins. A monitoring of an application can only begin when the application exists, e.g., after the application has been developed.

[0044] At 434, a message can be sent to an IT personnel, wherein the message identifies a dependency from the number of dependencies between the first application and the second application based on the description of the number of dependencies. The message can also group a portion of the un-expected behavior when a root cause of the un-expected behavior is based on a dependency between the first application and the second application. Grouping a portion of the un-expected behavior can allow an IT personnel to focus on addressing un-expected behavior based on the dependency and not each of the un-expected behaviors that caused the number of notifications individually.

[0045] Figure 5 is a diagram illustrating an example of a computing system according to the present disclosure. The computing system 556 can utilize software, hardware, firmware, and/or logic to perform a number of functions.

[0046] The computing system 556 can be a combination of hardware and program instructions configured to perform a number of functions, e.g., actions. The hardware, for example, can include one or more processing resources 540 and other memory resources 544, etc. The program instructions, e.g., machine-readable instructions (MRI), can include instructions stored on memory resource 544 to implement a particular function, e.g., an action such as dependency based event grouping.

[0047] The processing resources 540 can be in communication with the memory resource 544 storing the set of MRI executable by one or more of the

processing resources 540, as described herein. The MRI can also be stored in a remote memory managed by a server and represent an installation package that can be downloaded, installed and executed. A computing device 556, e.g., server, can include memory resources 544, and the processing resources 540 can be coupled to the memory resources 544 remotely in a cloud computing environment.

[0048] Processing resources 540 can execute MRI that can be stored on internal or external non-transitory memory 544. The processing resources 540 can execute MRI to perform various functions, e.g., acts, including the functions described herein among others.

[0049] As shown in Figure 5, the MRI can be segmented into a number of modules, e.g., a monitoring module 546, a dependency module 548, and a message module 550, that when executed by the processing resource 540 can perform a number of functions. As used herein a module includes a set of instructions included to perform a particular task or action. The number of modules 546, 548, and 550, can be sub-modules of other modules. For example, the monitoring module 546 and the dependency module 548 can be sub-modules and/or contained within a single module. Furthermore, the number of modules 546, 548, and 550 can comprise individual modules separate and distinct from one another.

[0050] In the example of Figure 5, a monitoring module 546 can comprise MRI that are executed by the processing resources 540 to monitor a number of applications, a number of services, and/or a number of components of an application. A computing device that monitors the applications, the services and/or the components of an application can also monitor a number of computing devices that are associated with the application, the services, and/or the components of an application. A computing device can be associated with an application when the application is being executed on the computing device, when the application sends information through a communication link to the computing device, among other associations between the computing device and the application. A computing device that monitors the application can create a number of events that identify un-expected behavior that is associated with an

application, services, and/or components of the application. A number of events that are associated with different applications, services, and/or components of an application can be caused by an error in a single application. For example, an error in a first application can cause un-expected behavior in a second application and a third application, an error in the first service can cause an un-expected behavior in the second service and a third service, and/or an error in the first application can cause an un-expected behavior in a first component of a second application and a second component of the second application.

[0051] A dependency module 548 can comprise MRI that are executed by the processing resources 540 to describe a number of dependencies between a number applications, a number of services, and/or a number of components of an application. A dependency between the first application and the second application, the first service and the second service, and/or the first application component and the second application component can be a direct dependency and/or an indirect dependency. A direct dependency can include, for example, a first application that depends directly from a second application, a first service that depends directly from a second service, and/or a first application component that depends directly from a second application component. An indirect dependency can be, for example, a first application that depends from a third application that depends from a second application, a first service that depends from a third service that depends from a second service, and/or a first application component that depends from a third application component that depends from a second application component. An indirect dependency between the first application and the second application can also exist when the first application and the second application depend on a third application. An indirect dependency between the first service and the second service can also exists when the first service and the second service depend on a third service. An indirect dependency between the first application component and a second application component can also exists when the first application component and the second application component depend on a third application component. The dependencies can be expressed in a description of the dependencies. A description of the dependencies can be a file that expresses

the dependencies. A description of the dependencies can also be a database that stores information on the dependencies. The description of the dependencies that is created before the application is developed and/or designed can be a description of the dependencies that is created before the application is monitored.

[0052] A message module 550 can comprise MRI that are executed by the processing resources 540 to send a message to an IT personnel identifying the dependencies. A message can provide IT personnel with access to the event and/or the description of the dependencies based on which the events are grouped.

[0053] A memory resource 544, as used herein, can include volatile and/or non-volatile memory. Volatile memory can include memory that depends upon power to store information, such as various types of dynamic random access memory (DRAM) among others. Non-volatile memory can include memory that does not depend upon power to store information. Examples of non-volatile memory can include solid state media such as flash memory, electrically erasable programmable read-only memory (EEPROM), phase change random access memory (PCRAM), magnetic memory such as a hard disk, tape drives, floppy disk, and/or tape memory, optical discs, digital versatile discs (DVD), Blu-ray discs (BD), compact discs (CD), and/or a solid state drive (SSD), etc., as well as other types of computer-readable media.

[0054] The memory resource 544 can be integral or communicatively coupled to a computing device in a wired and/or wireless manner. For example, the memory resource 544 can be an internal memory, a portable memory, and a portable disk, or a memory associated with another computing resource, e.g., enabling MRIs to be transferred and/or executed across a network such as the Internet.

[0055] The memory resource 544 can be in communication with the processing resources 540 via a communication path 560. The communication path 560 can be local or remote to a machine, e.g., a computer, associated with the processing resources 540. Examples of a local communication path 560 can include an electronic bus internal to a machine, e.g., a computer, where the

memory resource 544 is one of volatile, non-volatile, fixed, and/or removable storage medium in communication with the processing resources 540 via the electronic bus. Examples of such electronic buses can include Industry Standard Architecture (ISA), Peripheral Component Interconnect (PCI), Advanced Technology Attachment (ATA), Small Computer System Interface (SCSI), Universal Serial Bus (USB), among other types of electronic buses and variants thereof.

[0056] The communication path 560 can be such that the memory resource 544 is remote from a processing resource, e.g., processing resources 540, such as in a network connection between the memory resource 544 and the processing resource, e.g., processing resources 540. That is, the communication path 560 can be a network connection. Examples of such a network connection can include local area network (LAN), wide area network (WAN), personnel area network (PAN), and the Internet, among others. In such examples, the memory resource 544 can be associated with a first computing device and the processing resources 540 can be associated with a second computing device, e.g., a Java® server. For example, processing resources 540 can be in communication with a memory resource 544, wherein the memory resource 544 includes a set of instructions and wherein the processing resources 540 are designed to carry out the set of instructions.

[0057] As used herein, "logic" is an alternative or additional processing resource to perform a particular action and/or function, etc., described herein, which includes hardware, e.g., various forms of transistor logic, application specific integrated circuits (ASICs), etc., as opposed to computer executable instructions, e.g., software firmware, etc., stored in memory and executable by a processor.

[0058] As used herein, "a" or "a number of" something can refer to one or more such things. For example, "a number of widgets" can refer to one or more widgets.

[0059] The above specification, examples and data provide a description of the method and applications, and use of the system and method of the present disclosure. Since many examples can be made without departing from

the spirit and scope of the system and method of the present disclosure, this specification merely sets forth some of the many possible embodiment configurations and implementations.

What is claimed:

1. A method for dependency monitoring comprising:

monitoring a first application and a second application for un-expected behavior;

receiving a description of a number of dependencies between a number of applications wherein the description of the number of dependencies is created before monitoring of the first application and the second application begins; and

sending a message to an information technology (IT) personnel, wherein the message identifies a dependency from the number of dependencies between the first application and the second application based on the description of the number of dependencies.

2. The method of claim 1, wherein monitoring the first application and the second application includes creating a number of events that can be used to identify un-expected behavior in the first application and the second application.

3. The method of claim 2, wherein creating the number of events includes:

creating a first number of events that are associated with the first application and a second number of events that are associated with the second application;

wherein the first number of events identify a first un-expected behavior in the first application; and

wherein the second number of events identify a second un-expected behavior in the second application.

4. The method of claim 3, wherein the dependency between the first application and the second application is based on the first number of events, the second number of events, and the number of dependencies.

5. The method of claim 4, wherein the dependency between the first application and the second application is based on a dependency between the first application and a third application and a dependency between the second application and the third application.

6. A non-transitory machine-readable medium storing instructions for dependency monitoring executable by a machine to cause the machine to:

monitor a first application component and a second application component that are part of a same application;

receive a first description of a first number of dependencies between a number of application components of the same application wherein the first description of the first number of dependencies is created before the first application component and the second application component are monitored; and

receive a second description of a second number of dependencies, wherein the second description of the second number of dependencies is created after the first application component and the second application component are monitored;

send a message to an IT personnel identifying a dependency between the first application component and the second application component wherein the dependency is based on the first number of dependencies and the second number of dependencies.

7. The medium of claim 6, wherein the instructions executable to send the message to the IT personnel include instructions to base the dependency on a third number of dependencies.

8. The medium of claim 7, wherein the instructions executable to base the dependency on the third number of dependencies include instructions to create a third description from a number of run-time configuration files that are associated with the first application component and the second application component.

9. The medium of claim 6, wherein the instructions are executable to create the second description at a design stage of the first application component and the second application component before the development stage of the first application component and the second application component.

10. The medium of claim 6, wherein the instructions are executable to create the second description at a development stage of the first application component and the second application component.

11. A system for dependency monitoring, comprising:

a processing resource in communication with a memory resource, wherein the memory resource includes a set of instructions, executable by the processing resource to:

receive a number of events during a monitoring of a first service and a monitoring of a second service;

receive a first description of a first number of dependencies, wherein the first description of the first number of dependencies is created after the monitoring of the first service and the monitoring of the second service begins;

identify a first dependency, from the first number of dependencies, wherein the first dependency is a dependency of the first service and the second service on a third service;

identify a second dependency, from the first number of dependencies, wherein the second dependency is a dependency of the first service and the second service on a fourth service;

select one of the first dependency and the second dependency based on a second description of a second number of dependencies, wherein the second description is created before the monitoring of the first service and the monitoring of the second service begins; and

send a message to an IT personnel, wherein the message associates a portion of the number of events with a selected dependency and wherein the message identifies an associated service.

12. The system of claim 11, wherein the instructions executable to send the message include instruction to identify that the associated service is causing an un-expected behavior from which the number of events are created.

13. The system of claim 12, wherein the instructions executable to send the message include instruction to define the second description as being created by a number of sources that describe the second dependency of the first service and the second service on the associated service.

14. The system of claim 13, wherein the instructions executable to define the second description include instructions to:

identify each of the number of sources as a programmer that developed one of the first service, the second service, or the associated service, or a designer that designed one of the first service, the second service, or the associated service; and

include an identification of the number of sources in the message.

15. The system of claim 14, wherein the instructions executable to send the message to the IT personnel include instructions to identify the first service, the second service, and the associated service, wherein the associated service is one of the third service and the fourth service.

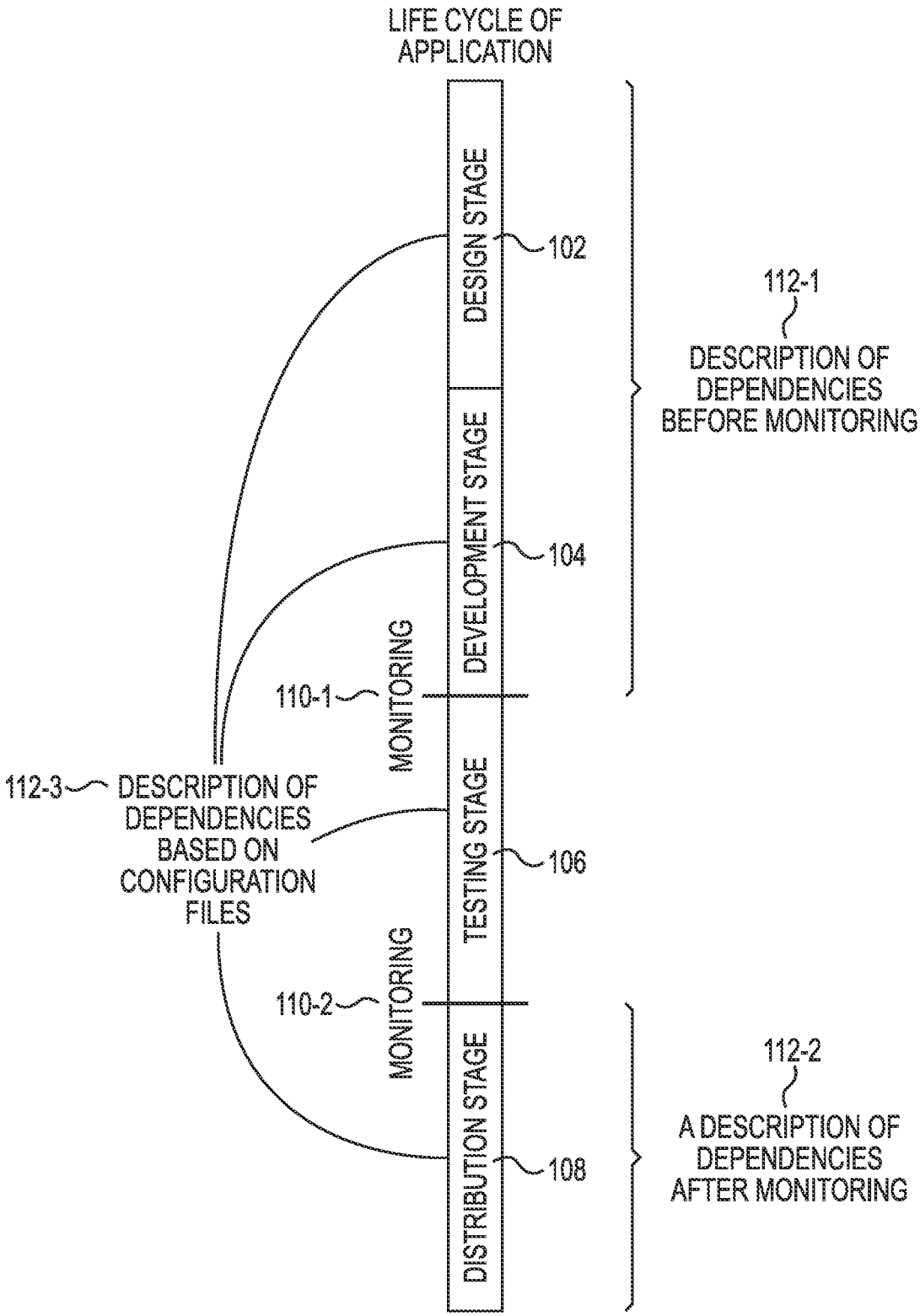
LIFE CYCLE OF APPLICATION
DESIGN STAGE
102
DEVELOPMENT STAGE
104
TESTING STAGE
106
DISTRIBUTION STAGE
108
MONITORING
110-1
MONITORING
110-2
112-1
DESCRIPTION OF DEPENDENCIES BEFORE MONITORING
112-2
A DESCRIPTION OF DEPENDENCIES AFTER MONITORING
112-3
DESCRIPTION OF DEPENDENCIES BASED ON CONFIGURATION FILES

Fig. 1

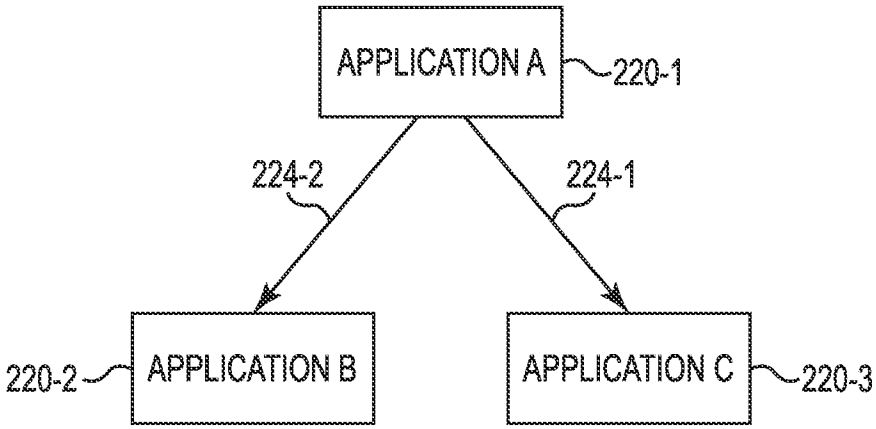
APPLICATION A
220-1
224-2
224-1
220-2
APPLICATION B
APPLICATION C
220-3

Fig. 2

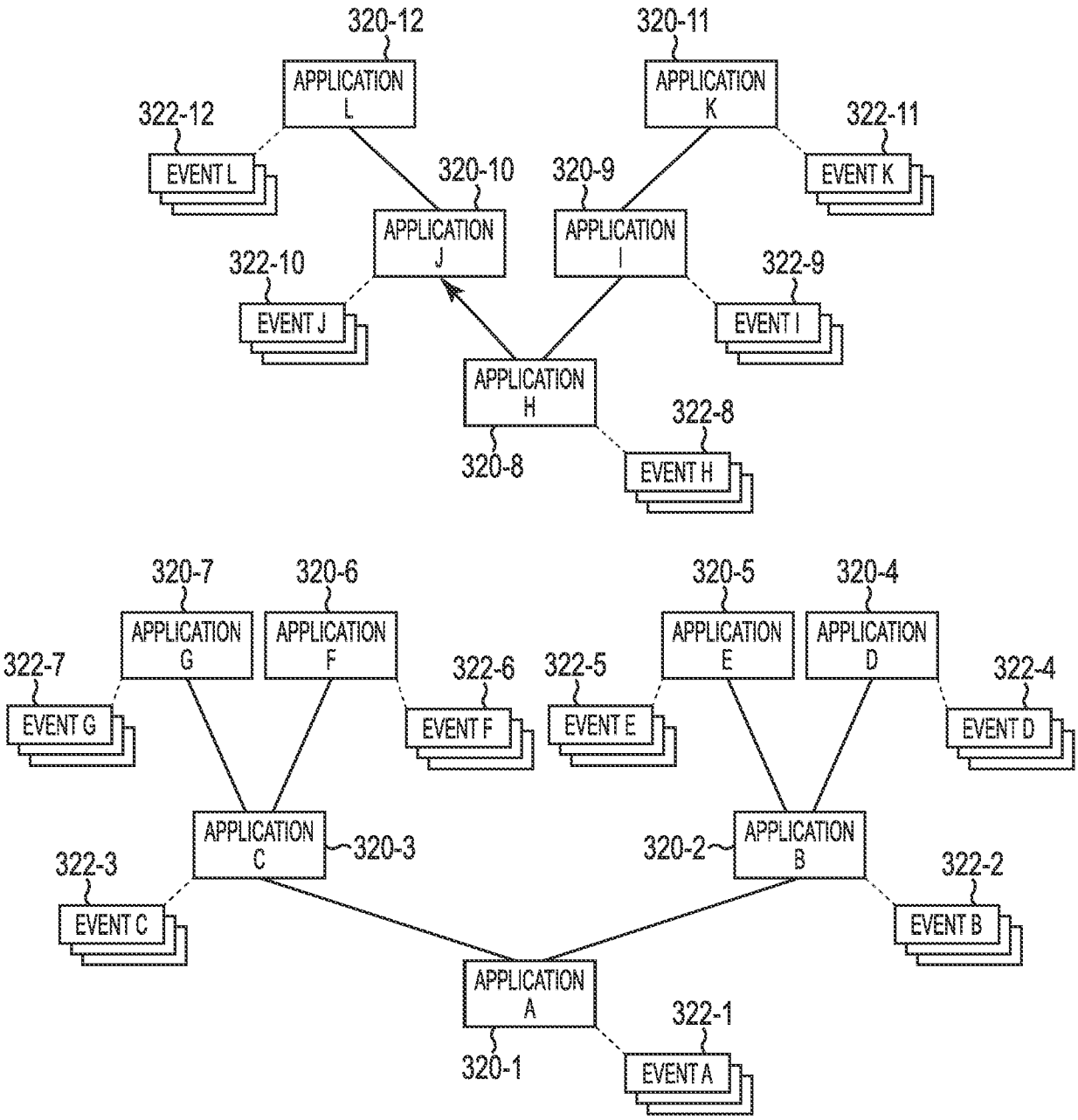
320-12
APPLICATION L
320-11
APPLICATION K
322-12
EVENT L
322-11
EVENT K
320-10
APPLICATION J
320-9
APPLICATION I
322-10
EVENT J
322-9
EVENT I
APPLICATION H
322-8
EVENT H
320-8
320-7
APPLICATION G
320-6
APPLICATION F
320-5
APPLICATION E
320-4
APPLICATION D
322-7
EVENT G
322-6
EVENT F
322-5
EVENT E
322-4
EVENT D
APPLICATION C
320-3
320-2
APPLICATION B
322-3
EVENT C
322-2
EVENT B
APPLICATION A
322-1
EVENT A
320-1

Fig. 3

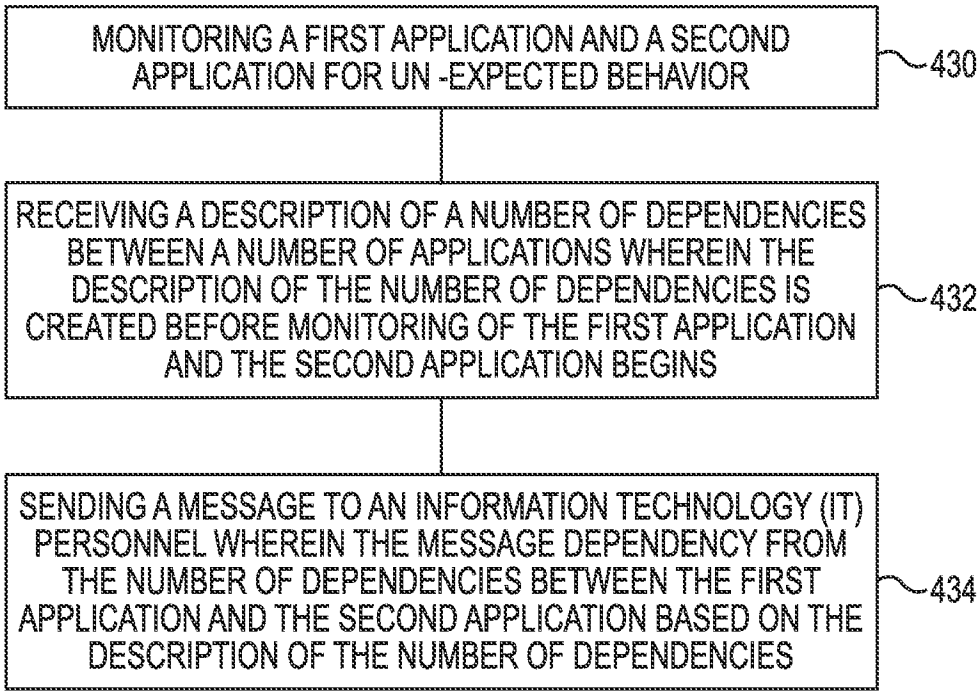
MONITORING A FIRST APPLICATION AND A SECOND APPLICATION FOR UN -EXPECTED BEHAVIOR
430
RECEIVING A DESCRIPTION OF A NUMBER OF DEPENDENCIES BETWEEN A NUMBER OF APPLICATIONS WHEREIN THE DESCRIPTION OF THE NUMBER OF DEPENDENCIES IS CREATED BEFORE MONITORING OF THE FIRST APPLICATION AND THE SECOND APPLICATION BEGINS
432
SENDING A MESSAGE TO AN INFORMATION TECHNOLOGY (IT) PERSONNEL WHEREIN THE MESSAGE DEPENDENCY FROM THE NUMBER OF DEPENDENCIES BETWEEN THE FIRST APPLICATION AND THE SECOND APPLICATION BASED ON THE DESCRIPTION OF THE NUMBER OF DEPENDENCIES
434

Fig. 4

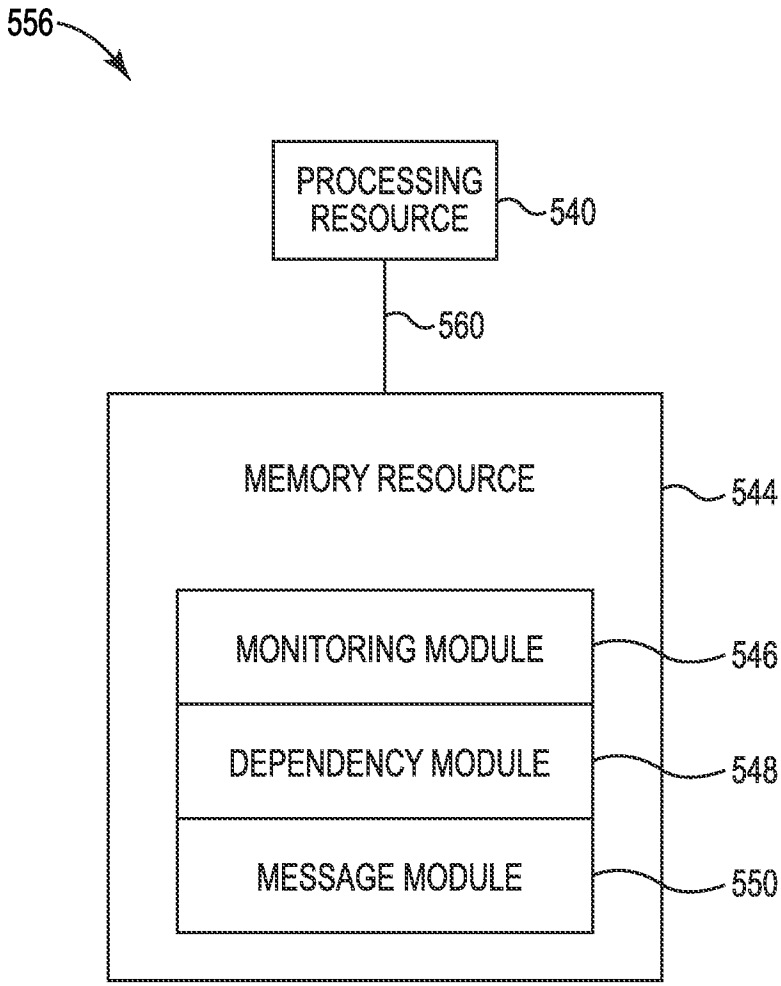
556
PROCESSING RESOURCE
540
560
MEMORY RESOURCE
544
MONITORING MODULE
546
DEPENDENCY MODULE
548
MESSAGE MODULE
550

Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/024154

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/30(2006.01)i, G06F 11/36(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 11/30; G06F 9/445; G06F 9/44; G06F 17/00; G06F 11/36

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: application, description, dependency, message, and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2008-0148231 A1 (THOMAS WEBER) 19 June 2008 See paragraphs [0041]-[0056], [0065], claims 11-16, and figures 2, 9.	1-15
A	US 2008-0092136 A1 (SAM PULLARA) 17 April 2008 See paragraphs [0017]-[0019], claim 1, and figure 4.	1-15
A	US 2012-0072884 A1 (SOEREN BALKO et al.) 22 March 2012 See paragraphs [0018]-[0024], and figure 2.	1-15
A	US 2006-0161892 A1 (ASHOK ANAND et al.) 20 July 2006 See paragraphs [0033]-[0036], and figures 4, 5.	1-15
A	US 7657554 B2 (SHAWN M. MURPHY et al.) 02 February 2010 See column 11, lines 11-47, figure 3, and claim 1.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 October 2013 (28.10.2013)	**28 October 2013 (28.10.2013)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
KANG, Sung Chul
Telephone No. +82-42-481-8405

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/024154

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008-0148231 A1	19/06/2008	EP 1933237 A1	18/06/2008
		WO 2008-071448 A1	19/06/2008
		WO 2008-071448 A8	23/10/2008
US 2008-0092136 A1	17/04/2008	US 2005-0034101 A1	10/02/2005
		US 7328427 B2	05/02/2008
		US 8185873 B2	22/05/2012
US 2012-0072884 A1	22/03/2012	None	
US 2006-0161892 A1	20/07/2006	US 7472377 B2	30/12/2008
US 7657554 B2	02/02/2010	US 2006-0101034 A1	11/05/2006