(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 January 2004 (29.01.2004)

(51) International Patent Classification:
G06F

(21) International Application Number:
PCT/US2003/021511

(22) International Filing Date:
9 July 2003 (09.07.2003)

(25) Filing Language:
English

(26) Publication Language:
English

(30) Priority Data:
60/396,666 17 July 2002 (17.07.2002) US


(72) Inventors: JOSHI, Shashank; 1058 Foxchase Drive, #264, San Jose, CA 95123 (US). BELLUR, Umesh; 728 Durshire Way, Sunnyvale, CA 94087 (US). OR, Yan; 217 Azalea Lane, San Ramon, CA 94583 (US). KOPER, John; 1388 Lake Street, San Francisco, CA 94118 (US). SUNDARESAN, Vinu; 922 Oranda Terrace, Fremont, CA 94536 (US).

(54) Title: BUSINESS SERVICE FOR A MULTITIER COMPUTE INFRASTRUCTURE

(57) Abstract: A business service for a multitier compute infrastructure is provided. A multitier topology map contains an inventory of network tier, application tier, and compute tier components and the relationships among the components. A business service can be defined as a logical grouping of components that transitionally implement a business process. The business service leverages the component relationships or dependencies to provide cross-tier visibility of the business process and performance analysis features.
Business Service for a Multitier Compute Infrastructure

Technical Field

[0001] This invention relates generally to distributed computing, and more particularly, to defining and to managing services as logical groupings of distributed components.

Background

[0002] With advances in computing, networking and software technology, an increasing number of applications are implemented in distributed environments, often composed of multitier compute infrastructures. For example, many Internet applications are implemented on a compute infrastructure that can be divided into three tiers: network, compute, and application. One advantage of a multitier compute infrastructure is that different tiers can function at different "levels" while still interoperating with each other. In the three-tier example, the network tier operates at the "lowest" level of the infrastructure, the compute tier operates on top of that, and the application tier operates at the "highest" level. As a result, enterprise and other applications can be distributed among the tiers in a fashion that results in greater optimization and utilization of the infrastructure. For example, if a certain functionality is desired, it is not required that the functionality be implemented in a monolithic piece of software installed on a particular computer in a specific location within the network. Rather, the overall functionality can be distributed among different components within the multitier compute infrastructure.

[0003] Software component architectures such as Java 2, Enterprise Edition (J2EE) and .NET are one approach which takes advantage of this flexibility. Software functionality is divided among different software components, each of which can run on a different host located at a different network address. Each of the software components, computers and the network topology may be optimized for efficiency, security, scalability, or other factors. For example, in the monolithic approach, a single code base and a single computer may be called upon to handle user requests for enhanced images, retrieve raw images from a large image warehouse, and perform complex calculations to enhance the images. With the component approach, one software component and server could handle user requests, another could retrieve images, and a third could perform the calculations. The overall enterprise is also more scalable since incremental capacity can be added by adding more instances of software components or servers.
[0004] One drawback of the multitier and software component approaches is that, typically, many components are used in order to implement the desired functionality. For example, the software portion can be implemented by a large number of software components, each possibly executing on a different server, with the multiple servers located on different networks. Software components may not even be executing on the same server each time. The real-time execution load can be load balanced among a server farm, for example. Currently, it is not uncommon for an enterprise application to have thousands of components, many of which must work simultaneously with each other in order for the overall enterprise application to function properly. In addition, multiple relationships between components exist within each tier, as well as across tiers of the compute infrastructure. For example, in the application tier, a web server and application server might work together to handle user requests. Cross-tier relationships can be more complex, such as those linking the web server, DNS server and access router with each other, but these often are the relationships that have a direct bearing on the availability, performance and security of the overall application.

[0005] Due to this increased complexity, managing a multitier compute infrastructure is more difficult. Typical management tools are mostly limited to a single tier. Many of these tools were developed for system administrators who are responsible only for a single tier. That is, one system administrator is responsible for networking, another for computers, and another for software loaded on the computers. Single-tier tools can give some visibility into the tier for which the system administrator is responsible, but do not give visibility into cross-tier relationships or interactions.

[0006] Single-tier tools also do not give direct visibility into the service which is a business’ end goal. For example, in the image enhancement example, the business is really interested in the delivery of enhanced images, not in the congestion level of its internal routers or the state of its internal network. The router and network are of interest only to the extent that they impact the business service of delivering enhanced images but, with single-tier tools, it is difficult, if not impossible, to determine this relationship. As a result, the relationship typically must be manually pieced together, one tier at a time, and often using knowledge that resides only in some key employee’s head. This is both time-consuming and risky – for example, if the key employee were to be unavailable.

[0007] Additionally, a lack of knowledge or incorrect knowledge about a cross-tier relationship can make the business impact of an outage unknown. For example, if an
application server requires an upgrade to fix a potential security problem, the complexity of
the cross-tier relationship may make it difficult to know which businesses processes could
be directly or indirectly affected by the application server downtime. Similarly, the scope
of the potential security problem could be unknown as well.

[0008] Other approaches attempt to address these shortcomings, yet lack visibility into
enterprise-wide performance management analytics or business metrics. Much effort is
being spent on approaches based on monitoring. OpenView and Tivoli are examples of
efforts in this general direction. Management tools can monitor individual components in
the infrastructure through instrumentation with increasing detail and sophistication. This
can give improved visibility into the individual component but does not effectively address
cross-tier visibility or the relationship between a component and a business service. For
example, processor throughput, server availability and similar metrics at best can only give
indirect visibility into business services, for example whether customers have access to an
enterprise application and can perform promised tasks at published service levels.

[0009] What is therefore needed is a logical grouping of distributed components that:
(1) provides visibility of cross-tier interactions; (2) provides metrics or performance
analysis of a business process; and (3) provides a grouping of logical applications or sub-
parts of logical applications to a business service.

Summary of the Invention

[0010] A business service specifies a logical grouping of components that collectively
perform a business process. A multitier topology map includes an inventory of application,
compute, and network tier resources available across an enterprise. The multitier topology
map also includes information about how the resources are related to each other. For
example, the multitier topology map can capture a relationship between an application and a
database that provides backend data storage for the application. An inventory of the
resources and the corresponding relationships among those resources, however, does not
define how the resources of the infrastructure work together to perform a business process.
The overall business process could be composed of many separate transactions that are
implemented by different applications. The business service groups these applications
together in a way that is meaningful for understanding the business process as a whole.

[0011] One embodiment of the present invention provides a business service that
logically groups the resources that accomplish a business process. A set of candidate
components is obtained from the multitier topology map. Candidate components include
resources, such as application components, that can compose a business process. The business service is formed by selecting the candidate components that implement the business process. For example, a business may have an ordering process that relies upon several pieces of the multitier compute infrastructure. An ordering business service can be defined to include each of the components, such as an order processing application, an inventory application, and a billing application.

[0012] In another embodiment of the present invention, the business service can be used to create a service-level view of the multitier compute infrastructure. A service-level view incorporates the dependency information from the multitier topology map to illustrate the application, compute, or network resources that implement the business process. The service-level view, therefore, can provide cross-tier visibility into the functioning of the business process.

[0013] In another embodiment of the present invention, the business service can provide precise metrics or performance analysis for the business process. Performance data can be gathered for each of the components of the relationship and aggregated for the business service. Other abstractions, such as logical applications can make it more convenient to define the business service without losing the fine component granularities available in the multitier topology map.

[0014] Further features of the invention, its nature and various advantages will be more apparent from the accompanying drawings and the following detailed description.

**Brief Description of the Drawings**

[0015] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention.

[0016] FIG. 1 is a multitier topology map in accordance with the present invention.

[0017] FIG. 2A is an illustration of a business service definition according to one embodiment of the present invention.

[0018] FIG. 2B is an illustration of a logical application definition according to one embodiment of the present invention.

[0019] FIG. 3A is a graph view of application tier dependencies for an exemplary enterprise in accordance with the present invention.

[0020] FIG. 3B is a graph view of business service dependencies according to one embodiment of the present invention.
[0021] FIG. 4 is a flowchart illustrating a method for creating a logical application and business service definition according to one embodiment of the present invention.

[0022] FIG. 5 is a flowchart illustrating a process for obtaining objects from the multitier topology map of FIG. 1 according to one embodiment of the present invention.

[0023] FIG. 6 illustrates a user interface for logical application definition according to one embodiment of the present invention.

[0024] FIG. 7 illustrates a user interface for business service definition according to one embodiment of the present invention.

[0025] FIG. 8 is a functional block diagram of computing device according to one embodiment of the present invention.

**Detailed Description of the Embodiments**

[0026] The present invention is now described more fully with reference to the accompanying figures, in which several embodiments of the invention are shown. The present invention may be embodied in many different forms and should not be construed as limited to the embodiments set forth herein. Rather these embodiments are provided so that this disclosure will be thorough and complete and will fully convey the invention to those skilled in the art. Although embodiments of the present invention are described as including discrete steps or modules, one skilled in the art will appreciate that the features or the functionalities of the present invention can be implemented in a variety of ways. For example, aspects of the present invention can be implemented using program instructions executing on a general-purpose computing device, an application-specific computing device, or in hardware (e.g., an integrated circuit). The program instructions can be distributed on a computer readable medium or storage volume. The computer readable storage volume can be available, for example, via a public network, a private network, or the Internet. Program instructions can be in any appropriate form, such as source code, object code, or scripting code.

**A. Multitier Topology Map**

[0027] FIG. 1 is a multitier topology map 50 in accordance with the present invention. In the illustrated example, the multitier topology map 50 is a data hierarchy that can be used to manage a multitier compute infrastructure. At a general level, a topology map describes the topology of a compute infrastructure. In other words, it identifies components within the infrastructure and indicates relationships between various components. The multitier topology map 50 illustrated in FIG. 1 is multitier in the sense that it identifies components...
from two or more tiers and at least one of the relationships in the map is a cross-tier relationship. The multitier topology map 50 is useful in managing a distributed computing environment because it documents the topology of the environment. The multitier topology map 50 is especially useful because it provides visibility into cross-tier interactions and relationships. That is, relationships among components of the multitier topology can be discovered, mapped, and stored in a data structure. Higher-level abstractions, such as the business services described below, can be defined to group components logically together and to make use of the component dependency information captured in the multitier topology map 50 as well as be added to it.

The visibility provided by the multitier topology map 50 has many benefits. It can reduce the time required to deploy new applications, as the process of identifying changes and relationships are automated. It can reduce the time required to troubleshoot faults within the infrastructure, as relationships between components are more easily identified. It can reduce the IT head count needed to support the infrastructure, as there is more documentation describing the infrastructure. It can also facilitate greater optimization of the infrastructure, as key relationships in the infrastructure are identified and better understood. As a result, the business typically can reduce its time to implement new deployments, business processes and policy changes. It typically can also gain a better understanding of the relationship between the multitier compute infrastructure and its ultimate business goals.

Returning to the exemplary multitier topology map 50 shown in FIG. 1, extending from root 100 are network components 105, compute components 110, and applications 115. The subcategories extending from applications 115 include logical applications 120, application components 125, business application packages 130, business services 135, and application dependencies 140. Each of these categories is described in further detail below.

1. Network Components, Compute Components, and Applications

Network components 105, compute components 110, and applications 115 respectively correspond to the components of a multitier compute infrastructure (e.g., a distributed computing environment) that includes a network tier, compute tier, and application tier. One skilled in the art will recognize that network components 105, compute components 110, and applications 115 are one way of categorizing the functionalities of a multitier compute infrastructure and other categorizations can be used to
identify the components and the relationships among the components. The network tier generally includes network components 105, which are items that concern network communications. In a typical Internet case, it might include the access and IP network and network components hosted or implemented on these networks, such as switches, routers, load balancers, firewalls, virtual LANs (VLANs), virtual private networks (VPNs) and layer 4-7 software switches and routers. Example vendors include Cisco, Unisphere, Redback and Netscreen.

[0031] The compute tier generally includes compute components 110, which are items that provide underlying system functionality that may be used by many end user applications. Typical technology areas include computing, storage and infrastructure services. Examples of components in the compute tier include host hardware such as desktop computers, servers and processors (e.g., Sun, Intel, HP, IBM), operating systems (e.g., Solaris, Linux, NT, HP-UX, AIX), and storage devices including RAID arrays, disk farms, storage area networks and the like (e.g., EMC, Brocade). System services, such as DNS, LDAP and NFS, are also classified as part of the compute tier in this example.

[0032] The application tier generally includes applications 115, which are software that is more directed to providing end user or enterprise functionality, as opposed to system services for example. Example components include web servers (e.g., Apache, iPlanet, IIS), application servers (e.g., J2EE servers, non-J2EE servers, WebLogic, WebSphere, iPlanet, ATG, COM+, .NET), packaged applications (e.g., PeopleSoft, Siebel), legacy software, and database software (e.g., Oracle). Applications 115 define a broad category of functionality including subcategories of logical applications 120, application components 125, business application packages 130, business services 135, and application dependencies 140.

2. Logical Applications and Business Services

[0033] In one embodiment of the present invention, logical applications 120 are defined as a grouping of application components 125 and business application packages 130. In the data hierarchy of FIG. 1, logical applications 120 are illustrated at the same level as application components 125 and business application packages 130. Although illustrated at the same level within the context of the topology map, logical applications 120 define higher-level functional behavior. For example, a logical application, such as order entry, can be defined to include a business application package (e.g., a database) and a web server. Although business application packages 130 can include application components 125, logical applications 120 can directly include application components 125 as members.
This flexibility in definition enables fine granularity over the composition of logical applications 120. The abstraction provided by logical applications 120 has several advantages. In some enterprises, for example, applications are implemented at the workgroup or department level. Defining a logical application (e.g., order entry) can require components from several departments. Logical applications 120 can relate application components 125 and business application packages 130 across departments to define a higher-level functional object. For example, an order entry logical application can include a business application package from the accounting department and an application component, such as an inventory stock-on-hand module from the shipping department.

Further, logical applications 120 inherit each of the dependencies of the member components. By using a logical application abstraction, the topology of the underlying member components can be changed without affecting the definition of the logical application. Given that logical applications 120 can include pieces of functionality from distinct departments or workgroups within the enterprise, changes at the application component-level, for example, are likely. Dependencies and cross-tier interactions are described in further detail below and with reference to FIGS. 2A and 2B.

Application components 125 are software components that implement some functionality and are generally coupled together to form an application. One example of application components 125 is Enterprise Java Beans (EJBs). Business application packages 130 can be groupings of application components 125 or other applications, such as databases. Business application packages 130 also support the formation of hierarchies. For example, an enterprise application implemented using the J2EE software component architecture might consist of a number of Enterprise Java beans (EJBs), each possibly executing on different hardware. In one classification, each EJB is a separate application component and the overall enterprise application is defined as a business application package that includes the collection of EJBs. Application components 125 and business application packages 130 both can be subcomponents of a logical application.

As another example, the accounting function within a business may rely on many different pieces of software, each of which is defined as an application component. It can be useful to define a business application package “Accounting” that includes these different pieces of software. This has several advantages. For example, it gives visibility into which parts of the multitier compute infrastructure are affected by the accounting department, and also how the accounting department might be affected by changes in the
multitier compute infrastructure. Again, component memberships and granularity are not fixed. For example, a business application package “Accounts Receivable” might be defined in addition to or instead of the Accounting business application package. The Accounting business application package can be defined as including the Accounts Receivable business application package. Other variations will be apparent.

[0037] Business services 135 are logically similar to logical applications 120 but are qualitatively different. They are similar in the sense that a business service also supports the formation of hierarchies, in much the same way that logical applications 120 do. Both provide higher levels of abstraction. However, as used in this example, business services 135 provide a business centric window into the multitier compute infrastructure. They provide visibility into functions that are directly relevant to the business operation—for example, order placement, status, and tracking, billing inquires, serving web pages, and/or meeting service level commitments. In contrast, logical applications 120, while they also represent higher-level functionality, are more concerned with how functionality is packaged and deployed. For example, customers typically are concerned with order placement (a business service) but not with how the service is implemented which may be using multiple business application packages.

[0038] Business services 135 offer the business throughput data that are used to understand the revenue, expense and/or customer service performance of the business. In other words, business services 135 define the relationships between the components in the multitier compute infrastructure and the enterprise’s actual business. As a result, they can provide visibility into questions such as the following. If order placement is not operating correctly, which components might be the cause? If certain components in the infrastructure are to be taken offline, what business services will be affected – order placement, web page serving, availability of certain services, other? If a business service is too slow, where is the bottleneck and why? Because of their importance and high level of abstraction, business services 135 typically are managed as a first-class entity in the system. That is, performance data or metrics can be collected from the member components and aggregated or analyzed to generate a performance analysis for the business process. One skilled in the art will appreciate an end-user response time measurement at a component granularly, for example, can be performed using the infrastructure of the multitier topology map 50.
As an example of performance analysis, an ordering service might include two transactions: one to actually place the order for a product and one to check on the status of the fulfillment of that order. The first transaction involves a web server, a B2B server that connects with a credit card company, an application server that holds the business logic for processing the order and a backend database, in addition to other network resources. The second transaction involves an overlapping set of components. In a traditional IT management structure, a network administrator is responsible for the network resources, a systems administrator is responsible for the various servers, and an applications administrator is responsible for deploying, shutting down and restarting these applications. However, none of these views gives significant visibility into the ordering service itself. The business management staff (e.g., the CIO or the CFO) are more interested in the performance of the ordering service (as opposed to any one tier) since ordering is directly linked to the health of the company's revenues for the quarter. A report on the availability rate of an individual server is not particularly interesting. The CIO is more interested in the availability of the ordering service as a whole. To accomplish this task, a business service "Ordering" is defined that includes the relevant components.

In another example, the sales force has a software system for the sales cycle, including order entry. The manufacturing division has separate software for inventory control and yet other software for tracking work in progress on the factory floor. The shipping dock has its own software for logging shipments, and the accounting department has yet another system to track accounts receivable. While each of these areas is interesting in its own right, the business is interested in tracking the progress of ordered products through the entire production process. Therefore, a business service "Order Tracking" is defined to include these different components.

As with logical applications 120, business services 135 can be defined in different ways and at different granularities. In the ordering example, "Order Placement" and "Order Fulfillment" can be defined as two separate business services, and the "Ordering" business service defined in terms of them. As another example, "Order Tracking" can be defined in terms of the application components 125 used to implement accounts receivable. Alternately, it can be defined in terms of an "Accounts Receivable" logical application, which in turn has been defined in terms of application components 125. Note that, similar to logical clusters, logical applications 120 and business services 135 have aspects of both a relationship and a component.
[0042] In the example of FIG. 1, logical applications 120 and business services 135 typically are defined in terms of application-level components: typically application components 125, business application packages 130 and business services 135. The definition typically also indicates relationships between the components, either explicitly or by inheritance or implication. In the example of FIG. 1, application components 125 typically are relatively low level components for which relationships with compute components 110, network components 105 and other application components 125 are either explicitly indicated or easily discovered. For example, a web server executing on a particular hardware server on a particular subnet has a dependency on that hardware server and subnet. These types of relationships typically are not repeated in the higher-level definitions of logical applications 120 and business services 135, but are included by inheritance. Thus, a business service “Web Page Serving” can be defined as dependent on the web server and, by implication, also dependent on the hardware server and subnet.

3. Application Relationships and Objects

[0043] Application dependency 140 is a relationship between components at the application-level, indicating that one application-level component somehow depends on another application-level component. The term “application-level component” is used here because, in this example, the term likely is not limited to just application components. Two examples of application dependency include transactional dependency and service dependency. Transactional dependency means that in order for application-level component X to service a user request, application-level component Y must be available. For example, X may be a web server responding to requests for data and Y is the corresponding database server. Service dependency is more infrastructure oriented. Service dependency refers to application-level components that are not in the direct path of a user request, but that are still necessary. Examples include components such as DNS, security or session services.

[0044] The multitier topology map 50 in FIG. 1 is application-centric. In this case, the categories extending from the root are network components 105, compute components 110 and applications 115. The subcategories extending from applications 115 are logical applications 120, application components 125, business application packages 130, business services 135 and application dependencies 140. Each of these components can be stored as objects in the topology map. One skilled the art will recognize that objects and relationships among objects can be mapped to relational database tables or otherwise stored in a data store. The term “object” is used throughout in its general sense and does not imply
implementation using object-oriented programming. Containment is not defined in separate objects. It is limited primarily to physical relationships, which are defined in the objects for the corresponding components. For example, an object for an application component indicates the hardware server on which it executes, or the object for a hardware server indicates the subnet on which it resides. Logical applications 120 and business services 135 are defined primarily at the application-level, which is why they are classified under "applications" for storage into the topology map. However, lower level dependencies are defined by inheritance.

[0045] As with tiers and components, the selection and definition of relationships to be used in the multitier topology map 50 is not unique. Other types of relationships and different definitions for the relationships described above will be apparent. For example, in an alternate implementation, service dependency and transactional dependency can be lumped together in a single category. Or transactional dependency can be defined as, in at least N% of user requests, Y is required. Or the relationships or components can be defined at a different granularity. For example, transactional dependency may mean that servicing a request from web page X requires a certain portion Y of a database, as opposed to web server X requires database Y.

[0046] As a further example, logical clustering is an item that has aspects of both a relationship and a component. It supports the formation of hierarchies. For example, execution of software may be load-balanced among hardware servers in a server farm. The logical cluster relationship can be used to define a new component that includes the load balancer and the hardware servers. A containment relationship exists between the software and the logical cluster. The relationship aspect of logical cluster is that it defines a certain relationship between components - the load balancer and hardware servers in this case. The component aspect is that the logical cluster itself is a component and can have relationships with other components. As used in this example, logical clusters are limited to lower level clusters, for example clusters of hardware and/or more infrastructure-related software.

[0047] FIG. 2A is an illustration of an exemplary business service definition according to one embodiment of the present invention. The illustrated embodiment shows a business service 200 including a first business application package 205, a second business application package 210, a logical application 215, and an application component 220. As described above, the business service 200 can be represented as an object in which components, such as first business application package 205, are members. That is, the first
business application package 205 and logical application 215 are components of the higher-level object the business service 200. The business service 200 is flexible as to which components can be included in the membership. For example, logical application 215, second business application package 210, and application component 220 each represent components or groups of components at distinct hierarchical levels. One advantage of providing flexibility in the definition of the business service 200 is that a higher-level object (e.g., the logical application 215) may be over inclusive for accurate definition of the business service 200. The ability to include sub-parts of a logical application or subcomponents of a business application package enables accurate performance analysis and visibility of the cross-tier relationships from the multitier topology map 50 that affect the business service 200.

[0048] A relationship 217 is illustrated between first business application package 205 and logical application 215. Relationship 217 indicates that logical application 215 depends on first business application package 205. The dependency could be between an application component that is included in the definition of the logical application 215 and an application component member of first business application package 205. Due to object inherency, the relationship can be resolved down to the finest level of granularity available in the multitier topology map 50. In the illustrated view of business service 200, however, the granularity of the underlying relationship is not specifically illustrated. One skilled in the art will note that although business service objects are high-level abstractions, the components and corresponding relationships included in the multitier topology map 50 enable the business service definition to capture, for example, application component-level cross-tier detail.

[0049] One advantage of defining the membership of the business service 200 with higher-level objects, such as logical application 215, is that the business service 200 can be updated as underlying relationships change without redefining the membership of the business service 200 itself. For example, second business application package 210 may be a payroll application that depends on compute host_a (not illustrated) to execute the payroll application. If the payroll department decides to move the payroll application to compute host_b, the changed dependency can be reflected in the business service 200 without manual intervention or redefinition of the membership of the business service 200. In one embodiment of the present invention, data in the multitier topology map 50 is updated to reflect the changed compute host for the second business application package 210. Business service 200 is only an example and may include all or none of the types of
components illustrated, such as business application packages, logical applications, and application components.

[0050] FIG. 2B is an illustration of an exemplary logical application definition according to one embodiment of the present invention. The illustrated embodiment shows a logical application 250 including a first business application package 255, a first application component 260, database 265, and application server 270. Similar to the business service definition illustrated in FIG. 2A, the logical application 250 can be represented as an object in which components are members. As described above, the logical application 250 can be defined as grouping of application-level components. In one embodiment of the present invention, other components (e.g., compute components 110) can be implicitly included in the definition of logical application 250 by way of object inherency. Relationship 262 is illustrated between the first application component 260 and the database 265. This relationship indicates that the first application component 260 is functionally dependent upon the database 265. The first application component 260 can be, for example, a database accessing software component. The first application component 260 is also implicitly dependent upon the compute tier resources of database 265. This is an implicit dependency that may change depending upon which compute tier resources host the database 265. Therefore, implicit dependencies or relationships can be transient.

[0051] Similarly illustrated is relationship 257 between the first business application package 255 and the application server 270. One skilled in the art will recognize that there are other relationships and dependencies that are implicit from the multitier topology map 50 that are not specifically illustrated. For example, the application server 270 is dependent upon other resources that are cataloged in the multitier topology map 50, such as compute components. Each of these lower-level dependencies are implicitly includes in the membership of logical application 250. In another embodiment of the present invention, other individual or structured components (e.g., logical clusters) can be explicitly included in the membership or definition of logical application 250.

[0052] One skilled in the art will recognize that the logical application 250 and business service 200 can be represented as first-class objects and stored in the multitier topology map 50. The architecture of the multitier topology map 50 allows for flexibility when storing the objects. For example, relationship 257 can be stored directly in the object that corresponds to the logical application 250 or the relationship 257 can be stored in the object that corresponds to the application dependencies 140.
FIG. 3A is a graph view of application tier dependencies in accordance with the present invention. In this application-level view, a web server 300 is shown coupled to a second application server 315, which is further coupled to a database 320. Also illustrated is a first application server 310 coupled to the database 320. The illustrated components are coupled based on dependency information obtained from the multitier topology map 50. That is, the web server 300 depends upon second application server 315. The second application server 315 has a dependency relationship with the database 320. The dependency between the web server 300 and the second application server 315 can be inferred from the component dependency information stored in the multitier topology map 50. Because the multitier topology map 50 includes lower-level components (e.g., application components 125), the dependencies illustrated can be generalized from finer granularity information. For example, a dependency among a subcomponent of the web server 300 and a subcomponent of the application server 2 (315) can be generalized as a dependency among the higher-level components or objects.

A system operator may only be interested in a higher-level view of the topology without the underlying dependencies illustrated, such as the view illustrated in FIG. 3A. Embodiments of the present invention, however, also generate lower-level views and component-centric views of the multitier topology to provide cross-tier visibility. Component dependencies can be initially discovered or updated in an automated fashion. Alternatively, component dependencies can be manually defined.

FIG. 3B is a graph view of business service dependencies according to one embodiment of the present invention. As describe above and with reference to FIG. 2A, components of various levels within the multitier topology map 50 can be grouped together to form a business service. One benefit of including, for example, compute components 105 in the multitier topology map 50 is the ability to determine which resources are used to provide the business service. A system administrator can then determine the impact of resource availability and performance to the transactional path that implements the business service.

In the illustrated example, the web server 300, the first application server 310, and the second application server 315 are shown in more detail. A business service can be represented as a chain of component dependencies beginning from a service access point 330. The service access point 330 is a uniform resource locator (URL) or other resource identifier, such as the port number of a compute resource. For example, the service access
point 330 for an “ordering” business service could be http://ordering.company.com. Another example of the service access point 330 could be a hostname and port number (e.g., webserver.company.com:80). A business service 335 is illustrated as a grouping of components across various pieces of the multitier architecture, such as the web server 200 and the second application server 315. As described above and with reference to FIG. 2B, a business service can define the resources used across an enterprise environment at the application component level. One advantage of mapping the business service at this level is to provide precise metrics or performance analysis. Performance data can be gathered for each of the application components and analyzed for the business service. Other abstractions, such as logical applications can make it more convenient to define the business service without losing the fine component granularities available in the multitier topology map 50.

[0057] Referring again to FIG. 3B, the business service 335 includes a logical application 305 and an application component 319. Other illustrated components or objects can be determined using the dependencies or object inheritances present in the multitier topology map 50. For example, the second application server 315 includes a business application package 317. The business application package 317 further includes the application component 319 as well as other application components. In the illustrated instance, the business service 335 depends only on application component 319 and not the business application package 317 as a whole. Likewise, application component 319 depends upon the database 320. One skilled in the art will note that because application component 319 depends upon the database 320, the business application package 317 implicitly depends upon the database 320.

[0058] The transactional path for the business service 335 includes the following components: from the service access point 330 to logical application 305, the application component 319, and the database 320. One skilled in the art will observe that the logical application 305 represents a component of the web server 300. That is, logical application 305 depends upon the resources of the web server 300, which in turn depends upon compute tier resources (not illustrated). If the web server 300 becomes unavailable, for example, then the business service 335 will likewise become unavailable. A system administrator can, therefore, determine how cross-tier relationships impact the availability, performance, or other metrics of the business service 335.
In embodiments of the present invention, some relationships can be automatically discovered. For example, it can be determined that the web server 300 depends upon the second application server 315 by observing the communications between the two components. For defining the business service 335, however, the automatic discovery methods may not be able to determine which subcomponents of the web server 300 are part of the business service 335. In one embodiment of the present invention, the membership of the business service 335 is defined using an interactive user interface. Further details of embodiments for defining the membership of a business service are described below.

FIG. 4 is a flowchart illustrating a logical application and business service definition process according to one embodiment of the present invention. Components can be grouped and related to one another in a way that better reflects the enterprise’s business rather than the physical layout of the multtier compute infrastructure. The illustrated process begins with determining which object type is being defined 405. Control proceeds to step 410 when defining a logical application and to step 450 when defining a business service. Logical applications and business services both represent abstractions that include other components or objects. Therefore, the process for defining each of them is similar. Beginning with defining a logical application 410, objects are obtained from the multitier topology map 50. In one embodiment of the present invention, objects are obtained from the multtier topology map 50 by a database query. Further details of which object types are available for membership in a logical application are described below and with reference to FIG. 5.

Next, configuration data for the logical application is received 420. Configuration data can be received interactively (e.g., a graphical user interface, such as in FIG. 6, or command line interface) and non-interactively (e.g., programmatically). In one embodiment of the present invention, a user interface is used to select object to be included in the logical application definition. In another embodiment of the present invention, a configuration file is read or processed to define the membership for a logical application. Other variations will be apparent to one skilled in the art.

The logical application object is then generated 425 which includes the selected member objects. In one embodiment of the invention, the logical application object uses object inherency to include lower-level objects that are also represented by higher-level abstractions. For example, a business application package may be comprised of a set of
application components. If a business application package is included in the logical application first-class object, the set of application components is inherently included.

[0063] In another embodiment of the present invention, the logical application directly includes the lower-level objects. That is, the set of application components are directly included in the membership of the logical application. One skilled in the art will note, however, that this is a different object representation for convenience of manipulating the objects that does not affect the behavior of the logical application object. The view generator or presentation module (described further below) can resolve object inherencies to present the proper relationships.

[0064] In step 430, the logical application object that has been generated is sent to the multitier topology map 50 for storage or other processing. The multitier topology map 50 handles storage and retrieval of objects. A backend database can be coupled to the multitier topology map 50 for selectively storing and retrieving individual objects or sets of objects.

[0065] The process for defining a business service 450 is similar to the process for defining a logical application 410 described above. Objects are obtained from the multitier topology map 455. Objects can be obtained from the multitier topology map 50 by a database query or the needed objects may already be resident in computer memory. In one embodiment of the present invention, candidate objects for a business service include other business services, logical applications, and application sub-parts (e.g., application components or business application packages). Further details of which object types are available for membership in a business service are described below and with reference to FIG. 5.

[0066] Next, configuration data for the business service is received 460. Configuration data can be received interactively (e.g., a graphical user interface, such as in FIG. 7, or command line interface) and non-interactive (e.g., programmatically). In one embodiment of the present invention, a user interface is used to select object to be included in the business service definition. In another embodiment of the present invention, a configuration file is read or processed to define the membership for a business service. Other variations will be apparent to one skilled in the art.

[0067] The business service object is then generated 465 which includes the selected member objects. In one embodiment of the invention, the business service object uses object inherency to include lower-level objects that are also represented by higher-level abstractions. For example, a logical application package may be comprised of a set of
business application packages. If a logical application is included in the business service first-class object, the set of business application packages are inherently included.

[0068] In another embodiment of the present invention, the business service directly includes the lower-level objects. That is, the set of business application packages are directly included in the membership of the business service. One skilled in the art will note, however, that this is a different object representation for convenience of manipulating the objects that does not affect the behavior of the business service object generated. The view generator or presentation module (described further below) can resolve object inherencies to present the proper relationships. In step 470, the business service object that has been generated is sent to the multitier topology map 50 for storage or other processing.

[0069] As described above, the business service object is a first-class object. This means that the business service object can contain other data or methods. For example, it can be useful to store technical, administrative, or management contact information about the business service. This would allow a systems administrator to inform the appropriate manager, for example, if there is a problem with the ordering service. Additionally, the business service object could be used to maintain state information about the status of the service (e.g., uptime) or to collect performance related data. Further, the business service object could include performance analysis methods to process the collected metrics.

[0070] FIG. 5 is a flowchart illustrating a process for obtaining objects from the topology map of FIG. 1 according to one embodiment of the present invention. The multitier topology map 50 includes much data about the components of the multitier compute infrastructure. To ensure correctness and convenience to the user, objects are retrieved from the multitier topology map 50 and filtered for user selection when defining the membership of a particular abstraction (e.g., a business service).

[0071] The illustrated process begins with determining object type 505. The object type represents the type of abstraction that is being defined (e.g., a business service object). The object type may already be known from step 405 of FIG. 4, however, one skilled in the art will note that the process for obtaining objects could be executed periodically or in conjunction with updates to the multitier topology map 50, which is not formally part of the object definition process described above.

[0072] Referring again to FIG. 5, there are illustrated seven general categories of objects that are retrieved from the multitier topology map 50: web server 510, application server 520, database 530, logical application 540, application component 550, other
application components 560, and hosts 570. For each of the general categories, there are subcategories that are identified. The subcategories include components that were both automatically discovered in a runtime discovery process and those that were user-defined. According to one embodiment of the present invention, objects that are identified in the categories or associated subcategories are filtered in step 580 to extract the object name and information about the object's hierarchical placement in the multitier compute infrastructure. As described above, this ensures correctness of selection and also enhances user convenience. The multitier topology map 50 contains much data about the components stored therein. Filtering therefore reduces the amount of data presented to the user to that which is needed for proper selection.

[0073] Web server 510 includes subcategories of clustered 512, individual instance 514, and user-defined 516. Examples of clustered web servers include those using load-balancer technologies such as Resonate. Examples of individual web servers include Apache, iPlanet, and Microsoft IIS. Further, other web servers could be manually defined or user-defined components.

[0074] Application server 520 includes subcategories of clustered 522, individual instance 524, and user-defined 526. Examples of clustered application servers include those using load-balancer technologies such as Resonate. Examples of individual application servers include BEA WebLogic or other J2EE application servers. Further, other application servers could be manually defined or user-defined components which were not automatically discovered in the infrastructure.

[0075] Database 530 includes subcategories of individual instance 532 and user-defined 534. Examples of individual databases include Oracle, IBM DB2, or Microsoft SQL Server instances. Other instances or structured components (e.g., clusters) can be manually defined or user-defined components. Logical application 540 includes sub-parts of applications 542. Sub-parts of applications 542 are business application packages and application software components, such as EJBs. Application component 550 includes legacy 552 and other processes 554. An examples of legacy 552 includes a proprietary customer application system. An example of other processes 554 includes non-distributed system processes that are not otherwise specified or identified in a discovery process. Other application components 560 represent other pieces of component architecture functionality, such as non-Java or non-J2EE servers.
Hosts 570 includes compute tier resources that can be included in the logical application or business service definition. In some enterprise environments, system administrators may not know the particular components (e.g., business application packages) that should be included in the business service definition. A system administrator likely knows, however, that a particular compute resource (e.g., a Sun Microsystems server) is needed to perform an order fulfillment process that is used in the “ordering” service.

After retrieval from the multitier topology map 50, the identified objects are then filtered to produce a set of candidate objects 580. Depending on the object type determined in step 505, the filtering extracts parameters from the identified objects. Parameters such as object name and the object’s relationship to other components are then used to generate a view of candidate objects 585. One embodiment of a generated view is a tree structured graphical interface for selecting the candidate objects. Another embodiment of a generated view is an XML file that can be programmatically parsed for the candidate objects. After generating the view, control is returned to the calling process where additional steps can be performed using the generated view.

FIG. 6 illustrates a user interface for logical application definition according to one embodiment of the present invention. When an interactive mode is desired, a user is presented with a tree-view of the candidate components. In the illustrated example, the user can select candidate objects 605 by dragging the object to the first logical application 640. A logical application, such as logical application 640, is defined by the objects that the user drags or otherwise selects for inclusion. In this example, the user has chosen to define the first logical application 640 to include database A 642, web server 644, and host A 646. Component 616 is a subcomponent of web server 644 (e.g., a Java server page) and has been implicitly included by object inherency. The user could have chosen to include component 616 without its parent object web server 644. One advantage of fine granularity object selection is more specific service definition and, therefore, enhanced cross-tier visibility. Although the illustrated interface defines one logical application, one skilled in the art will appreciate that a similar interface can be used to define multiple logical applications.

FIG. 7 illustrates a user interface for business service definition according to one embodiment of the present invention. The business service definition interface is similar to the logical application interface. When an interactive mode is desired, a user is presented with a tree-view of the candidate components. In the illustrated example, the user
can select candidate objects 705 by dragging the object to the first business service 740. A business service, such as business service 740, is defined by the objects that the user drags or otherwise selects for inclusion. Generally, applications 710, business services 744, and components 730 can be included. The objects described in FIG. 6 can be included as well. A business services can include another business service in its definition. For example, an “ordering” service might include an “order status” service as a subcomponent. In this example, the user has chosen to define the first business service 740 to include the first logical application 742, business service 744, and component 1 (746).

As described above, components 730 can be included in the definition of the first business service 740 because a component representing a higher-level of abstraction (e.g., a logical application) may be over inclusive. A business manager or systems administrator that has specific knowledge of the compute infrastructure can select a component 746 (i.e., a sub-part of a logical application). For example, the accounting department may define an accounting logical application. The “ordering” business service may be dependent only upon a particular sub-part of the accounting application, such as order entry. One advantage of fine granularity object selection is more specific business service definition and, therefore, enhanced cross-tier visibility and performance analysis. Although the illustrated interface defines one business service, one skilled in the art will appreciate that a similar interface can be used to define multiple business services.

FIG. 8 is a functional block diagram of computing device according to one embodiment of the present invention. The illustrated computing device includes memory module 800, input/output module 805, and processor 810 are each illustrated coupled to data bus 815. Alternatively, one or more of the modules can be directly coupled together. The memory module 800 can include software modules of executable program instructions. The data bus 815 provides an interface to other functional units, for example, the input/output module 805 or the processor 810. In the embodiment illustrated in FIG. 8, the memory module 800 includes a presentation module 830, a filter module 835, a buffer 840, a topology map interface module 845, and an object manipulation module 850 each of which is coupled to the data bus 815.

Program instructions included in the illustrated modules generally implement the processes described above and with reference to FIGS. 4 and 5. For example, the presentation module 830 includes program instructions for generating views of candidate objects 585. The filter module 835 includes program instructions for filtering the objects to
produce a set of candidate objects 580. The topology map interface module 845 includes program instructions for retrieving and for storing objects in the multitier topology map 50. The object manipulation module 850 includes program instructions for implementing and for managing the business service and logical application definition processes. Each of the program modules can use buffer 840 for object storage. In one embodiment of the present invention, the topology map interface module places objects retrieved from the multitier topology map in the buffer 840 and gathers objects from buffer 840 for storage to the multitier topology map 50.

[0083] Having described embodiments of a business service for a multitier compute infrastructure (which are intended to be illustrative and not limiting), it is noted that modifications and variations can be made by persons skilled in the art in light of the above teachings. It is therefore to be understood that changes may be made in the particular embodiments of the invention disclosed that are within the scope and spirit of the invention as defined by the appended claims and equivalents.
What is claimed is:

1. A method for defining a business service, the method comprising:
optaining at least one object from a multitier topology map, the multitier topology
map identifying components from at least two tiers of a multitier compute
infrastructure and indicating relationships between components including
at least one cross-tier relationship between components;
selecting a plurality of components from the at least one object that correspond to
the business service; and
generating a business service object for the selected components.

2. The method of claim 1 wherein the step of selecting further comprises:
receiving configuration data from at least one from the group of an interactive
user interface, a command line interface, and a non-interactive interface.

3. The method of claim 1 further comprising:
sending the business service object to the multitier topology map for storage.

4. The method of claim 1 wherein the business service object includes at least one of
performance data and administrative contact data.

5. The method of claim 1 wherein the business service object inherently includes an
application component.

6. The method of claim 1 wherein the step of obtaining further comprises:
generating a set of candidate objects from the multitier topology map;
filtering the set of candidate objects to extract at least the object name; and
producing a view of the candidate objects for selection.

7. The method of claim 6 wherein the set of candidate objects includes at least one
of a logical application and a sub-part of a logical application.

8. A method for generating a business service for a multitier compute infrastructure,
the method comprising:
identifying a service access point for the business service;
receiving a plurality of components from a multitier topology map, the plurality
of components including at least one cross-tier relationship;
selecting a subset of components from the plurality of components that
correspond to the business service; and
mapping the subset of components to the service access point.

9. The method of claim 8 wherein the mapping further comprises:
generating an object for the business service; and
including the subset of components as members of the object.

10. The method of claim 9 further comprising:
storing the object in the multilayer topology map.

11. An apparatus for defining a business service, the apparatus comprising:
a topology map interface module configured to obtain at least one object from a
multitier topology map, the multitier topology map identifying
components from at least two tiers of a multitier compute infrastructure
and indicating relationships between components including at least one
cross-tier relationship between components;
an object manipulation module communicatively coupled to the topology map
interface module configured to select the components from the at least one
object that correspond to the business service;
a presentation module communicatively coupled to the object manipulation
module configured to generate a business service object for containing the
selected components; and
the topology map interface module further configured to send the business service
object to the multitier topology map for storage.

12. The apparatus of claim 11 wherein the object manipulation module is further
configured to:
receive configuration data from at least one of an interactive user interface, a
command line interface, and a non-interactive interface.

13. The apparatus of claim 11 wherein the business service object includes at least
one of performance data and administrative contact data.

14. The apparatus of claim 11 wherein the business service object inherently includes
an application component.

15. The apparatus of claim 11 further comprising:
a filter module configured to filter a set of candidate objects from the multitier
topology map and to produce a view of the candidate objects for selection.
16. The apparatus of claim 15 wherein the set of candidate objects includes at least one of a logical application and a sub-part of a logical application.

17. A computer readable medium comprising:
   program instructions for obtaining at least one object from a multitier topology map, the multitier topology map identifying components from at least two tiers of a multitier compute infrastructure and indicating relationships between components including at least one cross-tier relationship between components;
   program instructions for selecting the components from the at least one object that correspond to the business service;
   program instructions for generating a business service object for containing the selected components; and
   program instructions for sending the business service object to the multitier topology map for storage.

18. The computer readable medium of claim 17 further comprising:
   program instructions for generating a set of candidate objects from the multitier topology map;
   program instructions for filtering the set of candidate objects to extract at least the object name; and
   program instructions for producing a view of the candidate objects for selection.
FIG. 1
FIG. 6
FIG. 7