Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 2868848 C 2024/04/09

(11)(21) 2 868 848

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2013/03/26
(87) Date publication PCT/PCT Publication Date: 2013/10/03
(45) Date de délivrance/Issue Date: 2024/04/09
(85) Entrée phase nationale/National Entry: 2014/09/26
(86) N° demande PCT/PCT Application No.: US 2013/033839
(87) N° publication PCT/PCT Publication No.: 2013/148651
(30) Priorité/Priority: 2012/03/28 (US13/433,162)

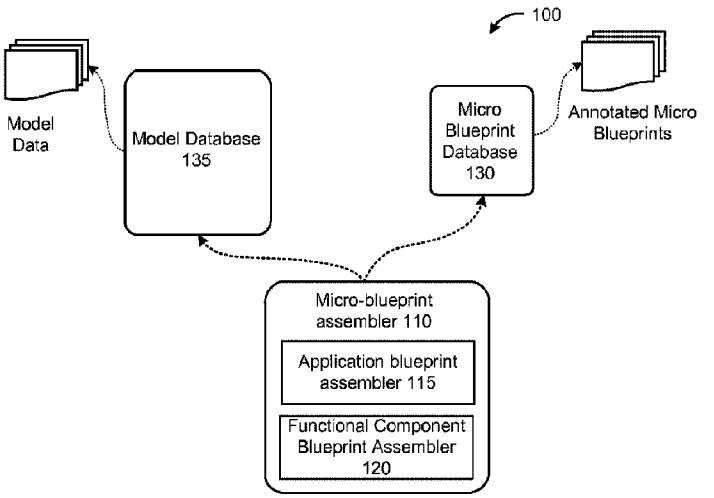
(51) Cl.Int./Int.Cl. *G06F 9/44* (2018.01)

(72) Inventeurs/Inventors:
SHARMA, ABHIJIT, IN;
KARNIK, NEERAN, IN;
GHASISAS, ABHAY, IN

(73) Propriétaire/Owner:
BMC SOFTWARE, INC., US

(74) Agent: SMART & BIGGAR LP

(54) Titre : ASSEMBLAGE AUTOMATIQUE DE BLEUS POUR ASSEMBLER UNE APPLICATION
(54) Title: AUTOMATED BLUEPRINT ASSEMBLY FOR ASSEMBLING AN APPLICATION



(57) Abrégé/Abstract:
The embodiments provide a data processing apparatus for automated blueprint assembly. The data processing apparatus includes a micro-blueprint assembler configured to receive a request for automated blueprint assembly for assembling an application, where the request specifies at least one feature, and a model database configured to store model data. The model data includes a plurality of classes and class properties. The data processing apparatus further includes a micro-blueprint database configured to store a plurality of micro-blueprints. Each micro-blueprint corresponds to a functional component of a stack element or service tier, and the functional component is annotated with one or more classes of the plurality of classes and at least one required capability and available capability. The micro-blueprint assembler is configured to generate at least one application blueprint based on the model data and the plurality of micro-blueprints according to the request.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 October 2013 (03.10.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/148651 A3

(51) **International Patent Classification**:
G06F 9/44 (2006.01)

(21) **International Application Number**:
PCT/US2013/033839

(22) **International Filing Date**:
26 March 2013 (26.03.2013)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
13/433,162 28 March 2012 (28.03.2012) US

(71) **Applicant**: **BMC SOFTWARE, INC.** [US/US]; 2101 CityWest Blvd., Houston, Texas 77042 (US).

(72) **Inventors**: **SHARMA, Abhijit**; 103, Wing 3, Wellington Mews, Lane 8, 98 Koregaon Park, Maharashtra, Pune 411001 (IN). **KARNIK, Neeran**; A-502, Tiara, Ivory Estates, Baner Road, Pune 411008 (IN). **GHASISAS, Abhay**; J-201 Samrajya, Shivateerth Nagar, Kothrud, Pune 411038 (IN).

(74) **Agent**: **SCHOLZ, Jared**; Brake Hughes Bellermann LLP, c/o CPA Global, P.O. Box 52050, Minneapolis, Minnesota 55402 (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).

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(88) **Date of publication of the international search report**:
3 January 2014

(54) **Title**: AUTOMATED BLUEPRINT ASSEMBLY FOR ASSEMBLING AN APPLICATION

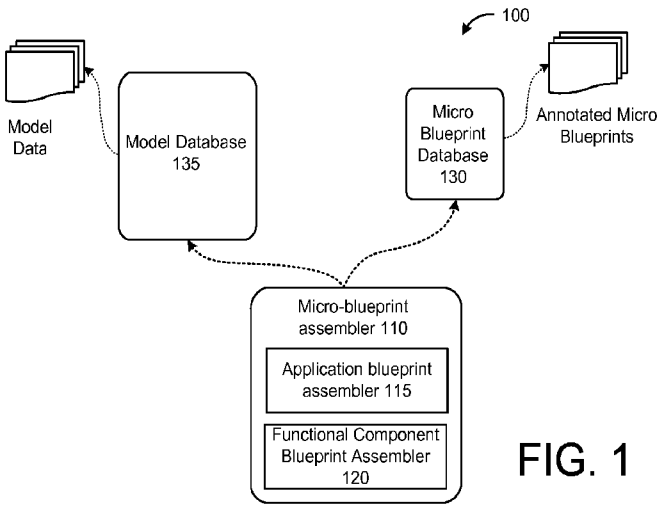


FIG. 1

(57) **Abstract**: The embodiments provide a data processing apparatus for automated blueprint assembly. The data processing apparatus includes a micro-blueprint assembler configured to receive a request for automated blueprint assembly for assembling an application, where the request specifies at least one feature, and a model database configured to store model data. The model data includes a plurality of classes and class properties. The data processing apparatus further includes a micro-blueprint database configured to store a plurality of micro-blueprints. Each micro-blueprint corresponds to a functional component of a stack element or service tier, and the functional component is annotated with one or more classes of the plurality of classes and at least one required capability and available capability. The micro-blueprint assembler is configured to generate at least one application blueprint based on the model data and the plurality of micro-blueprints according to the request.

WO 2013/148651 A3

AUTOMATED BLUEPRINT ASSEMBLY FOR ASSEMBLING AN APPLICATION

[0001]

TECHNICAL FIELD

[0002] The present disclosure relates generally to automated blueprint assembly.

BACKGROUND

[0003] A functional blueprint of an application may define the topology (e.g., the number of tiers), configuration, actions, constraints, operating systems, and software packages that need to be provisioned to deploy an application or server. A deployment blueprint for each functional blueprint defines a way (amongst many) in which the application could be provisioned in terms of mapping to resource sets and the required compute, storage, network. For example, a "QA" deployment blueprint may use a single resource set with one virtual machine (VM) to host all three tiers of application, whereas a "Production" deployment blueprint may distribute the three individual application tiers to three different resource sets.

[0004] Currently, there exist a number of different conventional methods to generate functional blueprints or deployment blueprints. However, the conventional methods provide a blueprint that is monolithic in design, for example, an application template that includes the definition of all the functional components and their respective software stacks in situ. As a result, the conventional blueprint may be unsuitable for extensive re-use. For example, there may be a number of different choices of software

CA 2868848 2019-07-22

stack elements in a software stack. With the current monolithic design, this would mean creating multiple functional blueprints for all the different options. The sheer choice of re-usable services, similar conformant infrastructure, software stack elements, and/or versions may lead to an explosion in the number of complete blueprints to be defined and maintained in a catalog. As a result, managing the conventional blueprints may be unworkable, or cumbersome at best.

SUMMARY

[0005] The embodiments provide a data processing apparatus for automated blueprint assembly. The data processing apparatus includes a micro-blueprint assembler configured to receive a request for automated blueprint assembly for assembling an application, where the request specifies at least one feature, and a model database configured to store model data. The model data includes a plurality of classes arranged in a hierarchy with relational information and the model data includes class properties for at least a portion of the plurality of classes. The data processing apparatus further includes a micro-blueprint database configured to store a plurality of micro-blueprints. Each micro-blueprint corresponds to a functional component of a stack element or service tier, and the functional component is annotated with one or more classes of the plurality of classes and at least one required capability and available capability. The micro-blueprint assembler is configured to generate at least one application blueprint based on the model data and the plurality of micro-blueprints according to the request.

[0006] In one embodiment, the request also specifies at least one constraint and environment. Also, the at least one feature may be a non-functional feature. The plurality of classes may represent different levels of the stack elements and the relational information may define relations between the plurality of classes.

[0007] The plurality of classes may include core abstract classes including at least one of an application, deployment, application server, platform runtime, operating system and database server, and each of the core abstract classes may include sub-classes corresponding to the micro-blueprints for the stack elements.

[0008] The micro-blueprint assembler is configured to generate the at least one application blueprint may include assembling a subset of the plurality of micro-blueprints

for each service tier and each stack element within each service tier according to the request.

[0009] The micro-blueprint assembler may include an application blueprint assembler configured to assemble micro-blueprints corresponding to service tiers of the application, and a functional component blueprint assembler configured to assemble micro-blueprints corresponding to stack elements for each of the service tiers.

[0010] The application blueprint assembler is configured to assemble the micro-blueprints may include obtaining micro-blueprints corresponding to the service tiers from the micro-blueprint database according to the request and the required capabilities and the available capabilities of the plurality of micro-blueprints.

[0011] The functional component blueprint assembler is configured to assemble the micro-blueprints may include obtaining micro-blueprints corresponding to the stack elements for each of the service tiers according to the request and the relational information.

[0012] The application blueprint assembler is configured to assemble the micro-blueprints may include obtaining micro-blueprints corresponding to the service tiers from the micro-blueprint database using an artificial intelligence (AI) search algorithm.

[0013] The micro-blueprint assembler may be configured to generate a list of application blueprints in an order of suitability that achieves the request.

[0014] The embodiments also provide a method for automated blueprint assembly. The method includes receiving a request for automated blueprint assembly for assembling an application, where the request specifies at least one feature, and generating at least one application blueprint based on model data and a plurality of micro-blueprints according to the request. The model data includes a plurality of classes arranged in a hierarchy with relational information and the model data includes class properties for at least a portion of the plurality of classes. Each micro-blueprint of the plurality of micro-blueprints corresponds to a functional component of a stack element or service tier, and the functional component is annotated with one or more classes of the plurality of classes and at least one required capability and available capability.

[0015] In one embodiment, the request also specifies at least one constraint and environment. Also, the at least one feature may be a non-functional feature. The

plurality of classes may represent different levels of stack elements and the relational information define relations between the plurality of classes.

[0016] The plurality of classes may include core abstract classes including at least one of an application, deployment, application server, platform runtime, operating system and database server, and each of the core abstract classes may include sub-classes corresponding to the micro-blueprints for the stack elements.

[0017] The generating the at least one application blueprint may include assembling a subset of the plurality of micro-blueprints for each service tier and each stack element within each service tier according to the request. The generating the at least one application blueprint may include assembling micro-blueprints corresponding to service tiers of the application and assembling micro-blueprints corresponding to stack elements for each of the service tiers.

[0018] The embodiments also provide a non-transitory computer-readable medium storing instructions that when executed cause one or more processors to perform a process. The instructions comprising instructions to receive a request for automated blueprint assembly for assembling an application, where the request specifies at least one feature, and generate at least one application blueprint based on model data and a plurality of micro-blueprints according to the request. The model data includes a plurality of classes arranged in a hierarchy with relational information, and the model data includes class properties for at least a portion of the plurality of classes. Each micro-blueprint of the plurality of micro-blueprints corresponds to a functional component of a stack element or service tier, and the functional component is annotated with one or more classes of the plurality of classes and at least one required capability and available capability.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] FIG. 1 illustrates a data processing apparatus for automated blueprint assembly according to an embodiment;

[0020] FIG. 2 is a flowchart illustrating example operations of the data processing apparatus of FIG. 1 according to an embodiment;

[0021] FIG. 3 illustrates model data of a model database of FIG. 1 according to an

embodiment;

[0022] FIG. 4 illustrates an abstract model for annotation of a functional component representing a software stack element or a service according to an embodiment;

[0023] FIG. 5 illustrates a mapping between classes of the model data and the micro-blueprints corresponding to the software stack elements according to an embodiment;

[0024] FIG. 6 illustrates example micro-blueprints and their class annotations corresponding to a plurality of service tiers according to an embodiment;

[0025] FIG. 7 illustrates a flowchart for the assembly of the micro-blueprints for the plurality of service tiers according to an embodiment; and

[0026] FIG. 8 illustrates a process to assemble micro-blueprints corresponding to the software stack elements according to an embodiment.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[0027] FIG. 1 illustrates a data processing apparatus 100 for automated blueprint assembly according to an embodiment. The data processing apparatus 100 may include a micro-blueprint assembler 110, a model database 135 that stores model data, and a micro blueprint database 130 that stores micro-blueprints annotated with information from the model data.

[0028] Micro-blueprints may include software stack elements and/or service tiers. For example, each micro-blueprint may correspond to a functional component of a software stack element (e.g., Java Oracle JRE 1.6) or a service tier (e.g., web tier, application tier, or database tier). The micro-blueprints for the service tier may refer to the more general functional components of an application – web tier, application tier and database tier, for example. The micro-blueprints for the software stack elements may refer to the more specific functional components of each service tier. For example, with respect to the software stack elements, the micro-blueprints may include the software stack elements for each service tier of the application. In one embodiment, the micro-blueprints may be re-usable and compose-able blueprints for the software stack elements such as Java Runtime Environment (JRE) or application server Oracle WebLogic 11.5g

and for the service tiers such as "PetStore App-Tier Services," for example. According to the embodiments, these micro-blueprints can be combined together to assemble a complete application or functional component blueprint.

[0029] In general, a blueprint may represent re-usable canonical application templates which represent different application views. These are typically stored in a repository and made available, as part of a catalog, for users to select from for various purposes like developing an application, for example. Enterprise and application architects in collaboration with middleware, database, and operating system administrators may define these blueprints based on enterprise architecture standards, reference architectures, security standards, stable version, standards compliance, and/or performance characteristics, for example. In other words, a blueprint may be a declarative specification of different aspects or views of an application or service such as the architecture view and the deployment view. They are used for enabling various key application related cloud services and other use cases. In one application, users may use the blueprints to provision applications in a cloud environment. Generally, the blueprint encompasses two types of blueprints – functional blueprints and deployment blueprints. A functional blueprint may define the architectural view or the structure of an application/service in terms of its tiers (also referred to as functional components) and the connection between the components. Also, the functional blueprint may define the software stack elements and related artifacts (e.g., startup, install scripts) within each functional component. A deployment blueprint may define the deployment view or intent for an application in terms of resource requirements (e.g., compute, storage, and network) for deploying its various functional components. Multiple deployment blueprints may conform to a functional blueprint of an application. The multiple deployment blueprints may include a QA deployment blueprint where all three functional components (e.g., web tier, application tier and database tier) are deployed on a single virtual machine (VM) with certain CPU and memory resources, where a Production deployment blue-print may deploy each of the three functional components in three individual VMs. The multiple deployment blueprints may encompass many other variations other than the QA deployment blueprint and Production deployment blueprint.

[0030] As further explained below, the data processing apparatus 100 decomposes

the blueprint into compose-able and re-usable micro-blueprints, and the micro blueprints are annotated with one or more classes from the model data in the model database 135. Essentially, the annotated micro-blueprints map the micro-blueprints of the service or software stacks to one or more classes of the model data, where the model data also includes relational information defining relations among the classes. Using the annotated micro-blueprints and the model data, the data processing apparatus 100 automatically assembles the complete composed blueprint on the fly from the micro-blueprints so that the application can be provisioned or developed. In other words, the data processing apparatus 100 uses a model driven approach for flexible, dynamic and automated, complete blueprint composition from smaller building blocks such as the micro-blueprints. The complete application blueprint may encompass the functional blueprint or the deployment blueprint described above. This completely assembled blueprint can be used to develop the entire application including its components or tiers, and their respective software stack elements.

[0031] According to one embodiment, the micro-blueprint assembler 110 may receive a request for automated blueprint assembly for developing an application. The request may specify features, constraints, or one or more environments for assembling the application. The request may specify functional (e.g., features, services) and/or non-functional aspects (e.g., security, scalability) of assembling the application. The request may be fairly high level or granular depending on the user's requirements. For example, a request may specify a need for a particular type of application (e.g., a PetStore application) regardless of the software stack. In response, the micro-blueprint assembler 110 may generate a .NET based PetStore blueprint or a Tomcat based PetStore blueprint. In another example, the request may specify a J2EE compliant PetStore blueprint. In response, the micro-blueprint assembler 110 may generate a J2EE compliant blueprint. In one embodiment, the features may correspond to the classes of the model data, and the constraint and the environment aspects may correspond to the class properties of the classes, as further explained below.

[0032] Generally, in response to the request, the micro-blueprint assembler 110 may query the model classes and its class properties from the model data of the model database 135. The model data may be considered a semantically rich and extensible

model which captures domain knowledge in the form of key classes in the domain, class hierarchy, class properties and relationships between classes, facets or restrictions on properties or relations such as allowed values, and/or cardinality, for example. In addition, the model data also captures one or more rules related to the classes and the class properties. According to the embodiments, the model data may capture the domain knowledge related to popular applications, components, and software elements, standard application services, servers, and/or middleware, for example. The key abstract classes may be Operating System, Platform Runtimes, Application Servers, and/or Deployment, for example. In one embodiment, the model data may include a plurality of classes arranged in a hierarchy with relational information. Further, the model data may include class properties for one or more of the classes. Essentially, the plurality of classes may represent different levels of stack elements and the relational information may define relations between the plurality of classes. The model data is further explained with reference to FIG. 3.

[0033] The micro-blueprints stored in the micro-blueprint database 130 may be annotated with relevant key classes from the model data to enable automated composition. For example, an Oracle Java JRE micro-blueprint may be annotated with "OracleJRE" class from the model. This aspect is further described with reference to FIG. 5. As such, when the micro-blueprint assembler 110 receives a request that specifies a feature of a certain type of application, the micro-blueprint assembler 110 may query the appropriate class from the model data of the model database 135, and then query the micro-blueprints having the obtained one or more classes from the micro-blueprint database 130. Further, the micro-blueprints may be annotated with capabilities and required capabilities, as further described with reference to FIGS. 4 and 6.

[0034] The micro blueprint assembler 110 may encompass a flexible approach that may leverage the model data, the annotated re-usable micro-blueprints to automate assembly of a complete application blueprint that can then be used to develop the entire application including its service tiers, and the software stack elements for each service tier. In other words, a semantically rich abstract model and annotated micro-blueprints enable automated composition of the complete application blueprint composed at runtime based on the application request.

[0035] The micro blueprint assembler 110 may include an application blueprint assembler 115, and a functional component blueprint assembler 120. In other words, according to one embodiment, the process may be separated into two parts – the application blueprint assembler 115 and the functional component blueprint assembler 120. The application blueprint assembler 115 may assemble the micro-blueprints from the micro-blueprint database 130 corresponding to service tiers (e.g., web tier, application tier, and database tier). The assembly of the micro-blueprints for the service tiers is further explained with reference to FIG. 7. Going a level deeper, the functional component blueprint assembler 120 may assemble the micro-blueprints from the micro-blueprint database 130 corresponding to the stack elements for each of the service tiers. The assembly of the micro-blueprints for the stack elements for each of the service tiers is explained with reference to FIG. 8. Also, if the micro-blueprint database 130 includes a relatively large number of micro-blueprints, the application component assembler 115 may use an artificial intelligence (AI) search algorithm for locating the appropriate micro-blueprints.

[0036] FIG. 2 is a flowchart illustrating example operations of the data processing apparatus 100 of FIG. 1. Although FIG. 2 is illustrated as a sequential, ordered listing of operations, it will be appreciated that some or all of the operations may occur in a different order, or in parallel, or iteratively, or may overlap in time.

[0037] Model data may be stored in a model database (202). According to the embodiments, the model database 135 may store the model data. The model data may capture the domain knowledge related to popular applications, components, and software elements, standard application services, servers, and/or middleware, for example. The key abstract classes may be Operating System, Platform Runtimes, Application Servers, and/or Deployment, for example. In one embodiment, the model data may include a plurality of classes arranged in a hierarchy with relational information. Further, the model data may include class properties for one or more of the classes. Essentially, the plurality of classes may represent different levels of the stack elements and the relational information may define relations between the plurality of classes.

[0038] A plurality of annotated micro-blueprints may be stored in a micro-blueprint database (204). For example, the micro-blueprint database 130 may store the

plurality of annotated micro-blueprints. Each micro-blueprint may correspond to a functional component of a stack element or service tier, where the functional component is annotated with one or more classes of the plurality of classes and at least one required capabilities and available capabilities.

[0039] A request for automated blueprint assembly for assembling an application may be received (206). For example, the micro-blueprint assembler 110 may receive the request for automated blueprint assembly for assembling an application. The request may specify at least one feature (functional or non-functional). Optionally, the request may specify one or more constraints or environments for assembling the application. In other words, the request may specify functional (e.g., features, services) and/or non-functional aspects (e.g., security, scalability). The request can be fairly high level or granular depending on the user's requirements.

[0040] At least one complete application blueprint may be generated based on the model data and the plurality of micro-blueprints according to the request (208). For example, the micro-blueprint assembler 110 may generate one or more of the application blueprints having at least one feature, constraint or environment based on the model data and the plurality of micro-blueprints in response to the request. For example, the micro blueprint assembler 110 may assemble a subset of the plurality of micro-blueprints from the micro-blueprint database 130 for each service tier and for each stack element within each service tier according to the request (e.g., the features, constraints, and/or the environment). Initially, the application blueprint assembler 115 may assemble the micro-blueprints according to the service tiers of the application. In one embodiment, the application blueprint assembler 115 may obtain the micro-blueprints corresponding to the service tiers according to the request (e.g., the features, constraints, and/or environment) as well as the required capabilities and the available capabilities of the plurality of micro-blueprints in the micro-blueprint database 130. In addition, if the plurality of micro-blueprints stored in the micro-blueprint database 130 is relatively large, the application blueprint assembler 115 may use an AI search algorithm in order to obtain the micro-blueprints in the micro-blueprint database 130. Subsequently, the functional component blueprint assembler 120 may assemble the micro-blueprints corresponding to the stack elements for each of the service tiers. For instance, the functional component blueprint

assembler 120 may obtain the micro-blueprints corresponding to the stack elements for each of the service tiers according to the request and the relational information of the model data of the model database 135.

[0041] FIG. 3 illustrates the model data 300 of the model database 135 of FIG. 1 according to an embodiment. The model data may include a semantically rich model which captures abstracted domain knowledge relevant to typical application blueprints with a specific focus towards provisioning of those applications. Domain experts (e.g., architects and/or administrators) may create and maintain the model database 135. As indicated above, the model data may capture the domain knowledge related to popular applications, components, software elements, middleware, standard application services, and/or application servers, for example.

[0042] The model data 300 may be considered a semantically rich and extensible model which captures domain knowledge in the form of key classes in the domain, class hierarchy, class properties and relationships between classes, facets or restrictions on properties or relations such as allowed values, and/or cardinality, for example. In addition, the model data also captures one or more rules related to the classes and the class properties. Referring to FIG. 3, the model data 300 may include a plurality of classes that are arranged in a hierarchy with relational information. As shown in FIG. 3, and further described below, the plurality of classes may represent different levels of stack elements, and the relational information (e.g., "is a", "needs") may define relations between the plurality of classes.

[0043] For example, the model data 300 may include an abstract main object 405 (e.g., AbstractDCObject), and a plurality of core classes 410 that are relevant to typical application stack elements. The plurality of core classes 410 may be considered sub-classes of the abstract main object 405. In one example, the core classes may include a database server 410-1, an operating system 410-2, a platform runtime 410-3, an application server 410-4, a deployment 410-5, and/or an application 410-6. The application 410-6 may represent applications such as a PetStore application or any other type of application. An application 410-6 may include a plurality of deployments represented as deployment 410-5. The deployment 410-5 may represent deployable artifacts such as packages/archives, e.g. Web Archive (WAR), Enterprise Archive (EAR),

and/or DLL, which are deployed into application servers to instantiate an application's modules. The deployment sub-classes may be WAR, EAR, and DLL.

[0044] The application server 410-4 (e.g., AppServer) may represent an application server where the deployable packages are deployed to instantiate application modules. The J2EEServer and IIS may represent two sub-classes of AppServer and further down the hierarchy WebLogic and WebSphere are sub-classes of J2EEServer. Application servers require a platform runtime for an execution environment, e.g. a J2EEServer requires a Java JRE platform runtime.

[0045] The platform runtime 410-3 (e.g., PlatformRuntime) may represent an execution runtime environment such as a JVM for programs like an application server to execute in. JRE and .NET are sub-classes of PlatformRuntime 410-3 and further IBMJRE and OracleJRE are sub-classes of JRE. As shown in FIG. 3, the platform runtime may require an operating system to execute.

[0046] The operating system 410-2 (e.g., OperatingSystem) may represent the operating system on which the platform runtime executes. Unix and Windows are sub-classes of OperatingSystem 410-2 and further AIX and Linux are sub-classes of Unix. The database server 410-1 (e.g., DatabaseServer) may represent a database such as Oracle or MSSQL, which in turn requires an operating system 410-2.

[0047] One or more of the classes may include class properties, which further describe various properties of the class that could be used in the application request as constrains to filter the software stack elements. In other words, the model data 300 includes class properties for one or more of the classes. For example, the class properties of the operating system 410-2 may include different versions such as 32bit or 64bit. As such, the application request may specify that it desires the app tier OS version to be 64 bit. Further, the WebSphere class may include different WebSphere versions such as WebSphere Version 5 or WebSphere version 6.1. As such, the application request may specify that it desires a WebSphere application server with minimum app tier OS version to be 64 bit. Further, the JRE class may include various versions of the JRE class. For example, the JRE class may have JRE version 1.5 or JRE version 1.6.

[0048] In addition, the model data 300 may include relational information. For example, the model data includes a plurality of key relations between classes in the

domain model. In one example, the key relations may include the needs relation. The needs relation may be defined on the AbstractDCObject 405 and represent a dependency between classes from a software stack perspective. To illustrate, as shown in FIG. 3, a Deployment (e.g. WAR) requires an AppServer (e.g. WebLogic), an AppServer requires a PlatformRuntime (e.g. OracleJRE) and the PlatformRuntime requires an OperatingSystem (e.g. Linux).

[0049] Further, the needs relation is further specialized in the sub-classes of AppServer and PlatformRuntime. J2EEServer (which is an AppServer) restricts the needs relation to the PlatformRuntime to the needs JRE relation to the JRE sub-class of PlatformRuntime as J2EE based application servers can only run on the Java Runtime Environment (JRE) and not on .NET. There is further specialization defined for WebSphere uses the needsIBMJRE relation to link to a sub-class of JRE, the IBMJRE as it only runs on IBM JRE environment. These can be considered as facets or constraints on the needs relation.

[0050] The needs relation and its specializations or restrictions will be utilized in the process of building a suitable software stack automatically from micro-blueprints of the individual software elements. This is possible because the needs relation represents dependencies between classes for abstract software elements from a software stack perspective.

[0051] In addition, the model data 300 may represent non-functional aspects of the application. For example, the non-functional aspects (e.g., security, scalability, high availability) may be represented as classes in the domain model and can be used to annotate the software packages or functional component blueprints. This allows the application request to include non-functional aspects in addition to the functional aspects. For example, a clustered WebLogic app server micro-blueprint may be annotated with classes from the domain representing concepts such as "high availability." This will be selected in preference to a generic WebLogic app server micro-blueprint when the application request indicates a need for high availability.

[0052] In essence, the classes (including their properties and relations) associated with the micro-blueprints will be used to dynamically assemble multiple blueprints to achieve a complete blueprint for a software stack for each functional component to

deliver a complete blueprint for an application.

[0053] Dynamic automated assembly of a complete application blueprint from several such micro-blueprints may require a mechanism for describing their functionality in terms of classes belonging to the model data 300 described earlier, e.g., describing the functionality of the software package or service micro-blueprint. For example, the micro-blueprints are annotated with information from the model data 300 of the model database 135. In particular, the micro-blueprints may be annotated with one or more classes. In addition, each micro-blueprint is annotated with capability information, as further described below with reference to FIG. 4.

[0054] FIG. 4 illustrates an abstract model for annotation of a functional component representing a software stack element or a service tier according to an embodiment. For example, as indicated above, each micro-blueprint corresponds to a functional component 510, and the functional component may correspond to a service tier (e.g., web tier, application tier, and database tier) or a software stack element. As such, the functional component 510 may correspond to the function of the software stack element or the function of the service tier. The functional component 510 may include a plurality of required capabilities and a plurality of available capabilities. Each required capability may correspond to a different functional feature 515, which is annotated with one or more classes 520 from the model data 300. Also, each available capability corresponds to a different functional feature 525, which is annotated with one or more classes 520 from the model data 300. In other words, the abstract model may include a functional component 510 that has a plurality of available capabilities which may represent the capabilities of a respective functional component. Further, the functional component 510 may have a plurality of required capabilities which may represent the capabilities of the respective functional component. The capabilities may be considered collections of a functional feature class. The functional feature class may include an attributed type which refers to a class 520 from the model data 300, e.g., an Oracle Java JRE micro-blueprint has a functional component which is annotated with "OracleJRE" class from the model data 300.

[0055] FIG. 5 illustrates a mapping between classes of the model data 300 and the micro-blueprints corresponding to the software stack elements according to an

embodiment. For example, as shown in FIG. 5, individual micro-blueprints 501 are annotated with information from the model data 300. For example, the micro-blueprints 501 may include a PetStore EAR micro-blueprint 501-1, an Oracle WebLogic 11.5 app server micro-blueprint 501-2, a Java Oracle JRE 1.6 micro-blueprint 503-3, and an Ubuntu Linux 10.04 micro-blueprint 501-4. These types of micro-blueprints are illustrated for explanatory purposes only, where the embodiments encompass any type of software stack element. The micro-blueprints 501 may be annotated with relevant classes from the model data 300 (e.g., capabilities<FunctionalFeature> type may refer to the relevant class). For example, the Oracle WebLogic 11.5 App Server micro-blueprint 501-2 may be annotated with the WebLogic class (sub-class of J2EEServer -> AppServer -> AbstractDCObject). Also, the micro blueprints may be annotated with the more generic class or more specific class in the class hierarchy and of course with multiple classes if needed.

[0056] FIG. 6 illustrates the micro-blueprints and their class annotations corresponding to the service tiers according to an embodiment. Although FIG. 6 illustrates the example of a PetStore application, the embodiments encompass any type of application having any number of service tiers. FIG. 6 illustrates three different micro-blueprints for the web tier, the application tier, and the database tier. Each micro-blueprint corresponds to a functional component 510 and includes functional features 515 and required functional features 525. Further, the functional features 515 and the required functional features 525 are annotated with class information from the model data 300. For example, the PetStore Web Tier has capabilities 'PetStoreWebApp' and requires capabilities of 'Inventory' and 'ShoppingCart' for it to be functional. The PetStore App Tier has capabilities 'Inventory' and 'ShoppingCart' and requires capabilities of a 'DatabaseServer."

[0057] In one embodiment, the application blueprint assembler 115 may assembly the micro-blueprints for the service tiers based on a matching of required and available capabilities, which allows automatically assembly of the PetStore Web Tier, the PetStore App Tier and the PetStore DB Tier micro-blueprints into the fully fledged PetStore Application blueprint. Going one level deeper, within a functional component like PetStore App Tier, the functional component blueprint assembler 120 may assemble

the micro-blueprints for the complete software stack from the annotated micro-blueprints in the database 130, as further explained below.

[0058] As indicated above, the micro blueprint assembler 110 may receive a request to generate a complete application blueprint. For example, the application request may specify at least one feature, as well as various constraints or environments. The output of processing an application request may be a list of dynamically composed application blueprints ranked in order of suitability. In one embodiment, the application request may specify a certain aspect such as a desired functional aspect (e.g., features, services required) and/or non-functional aspects (e.g., security, scalability, high availability) represented as classes from the model data 300. For example, a request for a highly available PetStore application will expect in response a composed blueprint for the PetStore application, with the app tier functional component software stack including clustered WebLogic (or Tomcat) app server. In addition, it may also include a functional component for a software load balancer to load balance HTTP requests to the application server instances.

[0059] Also, the request may specify one or more constraints. The constraints may relate to the constraint on the classes in the form on conditions on their properties/relations which filter the options available in terms of micro-blueprints to be used during composition. For example, the request may specify a constraint such as a J2EE compliant stack with minimum J2EE conformance level of 1.5.

[0060] Also, the request may specify an environment. The environment specified has an impact on the chosen components or software stack elements. For example, in the DEV environment, Tomcat application server may be selected, whereas in the PROD environment, the clustered WebLogic app server may be selected.

[0061] Based on the information contained in the request, the micro blueprint assembler 110 is configured to generate one or more complete application blueprints from the re-usable micro-blueprints. For example, the micro blueprint assembler 110 may assemble a subset of the plurality of micro-blueprints from the micro-blueprint database 130 for each service tier and for each stack element within each service tier according to the request. Initially, the application blueprint assembler 115 may assemble the micro-blueprints according to the service tiers of the application. In one embodiment,

the application blueprint assembler 115 may obtain the micro-blueprints corresponding to the service tiers according to the request and the required capabilities and the available capabilities of the plurality of micro-blueprints in the micro-blueprint database 130. The details of compositing the service level micro-blueprints are further explained with reference to FIG. 7. In addition, if the plurality of micro-blueprints stored in the micro-blueprint database 130 is relatively large, the application blueprint assembler 115 may use an AI search algorithm, which is further explained below.

[0062] Subsequently, the functional component blueprint assembler 120 may assemble the micro-blueprints corresponding to the stack elements for each of the service tiers. For instance, the functional component blueprint assembler 120 may obtain the micro-blueprints corresponding to the stack elements for each of the service tiers according to the request. The assembly for the micro-blueprints corresponding to the stack elements is further explained with reference to FIG. 8.

[0063] FIG. 7 illustrates a flowchart for the assembly of the micro-blueprints for the service tiers according to an embodiment. Although FIG. 7 is illustrated as a sequential, ordered listing of operations, it will be appreciated that some or all of the operations may occur in a different order, or in parallel, or iteratively, or may overlap in time.

[0064] At least one feature, constraint, and/or environment are extracted from the request (702). For example, the application blueprint assembler 110 may extract at least the feature, which may specify a certain type of application. However, the feature may include any type of feature related to the assembly of application. Further, the request may specify one or more constraints and/or environments.

[0065] Linked model classes are extracted from the request (704). For example, the application blueprint assembler 110 may extract the linked model classes from the request. In other words, the features may correspond to one or more of the plurality of classes, and the constraint and/or environment may correspond to one or more of the class properties. As such, the application blueprint assembler 100 may obtain the relevant classes for the features, constraints, and the environment.

[0066] The application blueprint assembly process is called (706). For example, the application blueprint assembler 110 may start the application blueprint assembly

process, and the application blueprint assembler 115 may start to assemble the micro-blueprints corresponding to the service tiers of the application.

[0067] The micro blueprints having the extracted classes are obtained (708). For example, the application blueprint assembler 115 may query the micro-blueprint database 130 in order to obtain the micro-blueprints having the extracted classes. For example, the application blueprint assembler 115 may search the micro-blueprint database 130 in order to obtain a subset of micro-blueprints based on the required capabilities and the available capabilities of the micro-blueprints such that one or more of the available and required capabilities match.

[0068] The annotated classes of a current micro blueprint are checked to determine if they match the constraints specified in the request (710). For example, if the request includes one or more constraints, the application blueprint assembler 115 determines if the constraints specified in the request match the class properties of model data 300 corresponding to the current micro-blueprint.

[0069] The micro blueprint is checked to determine whether it is admissible in the selected environment (712). For example, the application blueprint assembler 115 may determine if the current micro blueprint is admissible in the environment specified in the request.

[0070] The micro-blueprint is obtained (714). For example, if meeting the conditions of 710 and 712, the application blueprint assembler 115 may obtain the micro blueprint from the micro-blueprint database 130. This process may be repeated for each of the micro-blueprints obtained in 708.

[0071] The above described process of FIG. 7 may be configured in a set of instructions, which when executed cause one or more processors to perform a series of functions. The instructions may include:

```
// Process to generate complete composed blueprint for application
for all Functional & Non Functional Features in Request {
        for all Model Classes in Features {
                Call function process(Classes, Constraints, Environment)
        }
        // Above yields a list of candidate composed complete blueprints with micro
        //blueprints for each of the related tiers or functional components
        List<Blueprint> composedBlueprints
        for all composedBlueprints {
```

```
            for each tier microBlueprint in a composedBlueprint {
                  Invoke Process to generate the composed blueprint for the software
                  stack of the component
                  Add this to the tier microBlueprint
                        }
            }
      }

List<MicroBlueprint> function process(Classes, Constraints, Environment) {
      List<MicroBlueprint> microBlueprints
            for all Classes {
                  microBlueprints = Find MicroBlueprint(s) in Repository annotated
            with these classes i.e.
            class is subset FunctionalComponent.Capabilities.FunctionalFeature.type
            }

            for all microBlueprints {
            Check if current MicroBlueprint annotated classes match the specified
tier level constraints (if any)
      if(no) Remove from microBlueprints
            }
            for all microBlueprints satisfying constraints {
                  Check if current MicroBlueprint is admissible in the selected
Environment
                  if(no) Remove from microBlueprints
            }

            for all microBlueprints dynamically discover the linked functional
component micro-blueprints {
            // recursive step
            List<MicroBlueprint> linkedMicroBlueprints =
Call function process (Current.MicroBlueprint.Required Capabilities.Classes,
Constraints, Environment)
      if(linkedMicroBlueprints empty)  Remove current from microBlueprints
      else  Link linkedMicroBlueprints to current MicroBlueprint
                  }
            }
            return microBlueprints
```

[0072] Next, the functional component blueprint assembler 120 is configured to assemble micro blueprints corresponding to software stack element for each of the service tiers. For example, the functional component blueprint assembler 120 is configured to dynamically compose packages/software stack elements to build a software stack for each service tier.

[0073] In one embodiment, the functional component blueprint assembler 120 utilizes the needs relation and its specializations or restrictions in the process of building a suitable software stack automatically from micro-blueprints of the individual software elements. This is possible because the needs relation represents dependencies between classes for abstract software elements from a software stack perspective.

[0074] FIG. 8 illustrates a process to assemble micro-blueprints corresponding to the software stack elements according to an embodiment. Although FIG. 8 is illustrated as a sequential, ordered listing of operations, it will be appreciated that some or all of the operations may occur in a different order, or in parallel, or iteratively, or may overlap in time.

[0075] The micro-blueprints for the software stack elements may be obtained for each service tier (802). For example, the functional component blueprint assembler 120 may be configured to obtain the micro-blueprints for the service tier micro-blueprints obtained in the process of FIG. 7.

[0076] The annotated classes for the micro-blueprints are obtained (804). For example, the functional component blueprint assembler 120 may obtain the relevant annotated classes from the obtained micro-blueprints. The process may determine if the current annotated class is the operating system class (806). For example, the functional component blueprint assembler 120 may determine if the current annotated class is the operating system class. If yes, the process is stopped (808) because it signals the end of the software stack element. If no, the process continues to 810. Although this particular example utilizes the operating system software element stack as an ending point, the embodiments encompass using any type of software stack element as the ending point.

[0077] The linked classes pointed to by the relational information are obtained (810). For example, the functional component blueprint assembler 120 may obtain the linked classes point to by the needs relation. Each feature, constraint, and environment contained in the request is applied to the classes (812). For example, the functional component blueprint assembler 120 may apply each feature, constraint, and environment contained in the request to the classes. Subsequently, the process may repeat itself for each software stack element in each of the service tiers.

[0078] The above described process of FIG. 8 may be configured in a set of

instructions, which when executed cause one or more processors to perform a series of functions. The instructions may include:

// Process to generate composed blueprint for a tier or functional component consisting of a suitable software stack
// automatically from micro-blueprints of the individual software elements
- Get app Deployment micro blueprint for Functional Component or tier e.g. for PetStore App Tier it is the PetStore EAR micro blueprint
- Get annotated class(es) for the Deployment micro blueprint e.g. for PetStore EAR it is EAR class from the model
- Carry on in a recursive manner until you reach the OS class - stop here as you have the software stack {
- For annotated class get linked class(es) pointed to by the needs* relation e.g. for EAR above the needs relation points to J2EEServer class
- If a more specialized needs* relation is present then one chooses the linked specific class - otherwise there could be many choices
for the linked classes
e.g. Deployment can be linked to any AppServer
- Apply specified class properties level constraints to filter the classes to get the matching ones e.g. a constraint like
J2EEServer.conformanceLevel > 1.5 can filter out .NET app servers
- Look at non functional requirements to filter out options that do not match them e.g. if HA is required then instead of WebLogic Blueprint
WebLogic clustered Blueprint is chosen
- Also filter for Environment e.g. Dev, Prod may have different requirements
}

[0079] Also, as explained above, if the number of micro-blueprints is relatively large, the application blueprint assembler 110 may assemble the micro-blueprints using an AI search algorithm.

[0080] An AI search is used in a lot of different problems where the search space is very large e.g. in games, robot motion planning. AI search algorithms make it computationally very feasible to solve a range of search like problems. This problem has been formulated as an Artificial Intelligence (AI) search problem where the solution is the fully assembled complete blueprint with both the software stack definitions as well as the tier definitions included.

[0081] More specifically, the problem is formulated as an A* (AStar AI Search Algorithm) search problem. Briefly the A* algorithm is heuristic based "best-first" search

algorithm which starts from an initial state and takes one to a goal state(s), at any current state choosing the best next state based on the heuristic equation provided below.

[0082] **Eq.** (1) $f(n) = g(n) + h(n)$

[0083] The parameter g(n) is the lowest cost incurred in the chosen path the search algorithm has taken so far to the current state (e.g., the path formed out of the chosen states so far). The cost of a particular path segment is defined in a problem specific manner.

[0084] The parameter h(n) is the estimated cost of the remaining path to the goal state(s) from the current state. All of these parameters, the goal state(s), functions and cost can be defined as needed in the problem. In this case, a functional component with a set of capabilities and a set of required capabilities may represent a current state. There may be several functional components in the micro-blueprint database 130 that either individually or in combination satisfy the required capabilities.

[0085] For example, in particular example, the "web tier" component forms the current state "State n". The web tier component has required capabilities "Inventory" and "Shopping Cart". Thus, querying the micro-blueprint database 130 may yields three components which either partially or completely satisfy these required capabilities namely "app tier" (completely meets requirements), "inventory service" (partially) and "shopping cart service" (partially). Potential next states are generated (e.g., "State n + 1" and State n + 2") from these in such a way that each state completely satisfies the current state "State n" requirements. In this case, "State n + 1" comprises of the "inventory service" and the "shopping cart service" components and "State n + 2" is comprised of the "app tier" component.

[0086] Next, the optimum state is chosen amongst these states as the optimum state. This cost can include heuristics like choose state with minimum number of components, also considering the constraints, for example. This also feeds into the cost function g(n). Thus, a path that least cost or best is built.

[0087] The goal state is reached by keeping track of accumulated "capabilities" of the chosen states so far and the accumulated "required capabilities". Once the accumulated "required capabilities" are a subset of accumulated "capabilities", the complete blueprint is determined and the process stops searching.

[0088] As a result, the data processing apparatus of the embodiments may decompose the blueprint into compose-able and re-usable micro-blueprints, and the micro blueprints are annotated with one or more classes from the model data in the model database 135. Essentially, the annotated micro-blueprints map the micro-blueprints of the service or software stacks to one or more classes of the model data, where the model data also includes relational information defining relations among the classes. Using the annotated micro-blueprints and the model data, the data processing apparatus automatically assembles the complete composed blueprint on the fly from the micro-blueprints so that the application can be provisioned or developed. In other words, the data processing apparatus 100 uses a model driven approach for flexible, dynamic and automated, complete blueprint composition from smaller building blocks such as the micro-blueprints. The complete application blueprint may encompass the functional blueprint or the deployment blueprint described above. This completely assembled blueprint can be used to develop the entire application including its components or tiers, and their respective software stack elements.

[0089] Implementations of the various techniques described herein may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Implementations may implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program, such as the computer program(s) described above, can be written in any form of programming language, including compiled or interpreted languages, and can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

[0090] Method steps may be performed by one or more programmable processors executing a computer program to perform functions by operating on input data and generating output. Method steps also may be performed by, and an apparatus may be

implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

[0091] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. Elements of a computer may include at least one processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer also may include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory may be supplemented by, or incorporated in special purpose logic circuitry.

[0092] To provide for interaction with a user, implementations may be implemented on a computer having a display device, e.g., a cathode ray tube (CRT) or liquid crystal display (LCD) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

[0093] Implementations may be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation, or any combination of such back-end, middleware, or front-end components. Components may be interconnected by any form or medium of

digital data communication, e.g., a communication network. Examples of communication networks include a local area network (LAN) and a wide area network (WAN), e.g., the Internet.

[0094] While certain features of the described implementations have been illustrated as described herein, many modifications, substitutions, changes and equivalents will now occur to those skilled in the art.

WHAT IS CLAIMED IS:

1. A data processing apparatus for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the data processing apparatus including:

a micro-blueprint assembler configured to generate at least one application blueprint, and to receive a request for automated blueprint assembly for automatically assembling the application blueprint, the request specifying at least one feature;

a model database configured to store model data, the model data including a plurality of classes arranged in a hierarchy with relational information specifying relationship links among the plurality of classes;

a micro-blueprint database configured to store a plurality of micro-blueprints, each micro-blueprint corresponding to a functional component corresponding to a service tier comprising one of a web tier, an application tier, and a database tier, each micro-blueprint being annotated with one or more classes of the plurality of classes from the model data of the model database; and

the micro-blueprint assembler configured to generate the application blueprint for the application by assembling micro-blueprints from the micro-blueprint database, wherein, when generating the at least one application blueprint, the micro-blueprint assembler is configured to: apply the at least one feature to the classes of the model data of the model database to obtain a class that is relevant to the at least one feature of the request, and query the micro-blueprint database to obtain at least one micro-blueprint having the obtained class and at least one micro-blueprint linked to the obtained class by the relational information.

2. The data processing apparatus of claim 1, wherein the request also specifies at least one constraint and environment.

3. The data processing apparatus of claim 1, wherein the at least one feature is a non-functional feature, the non-functional feature specifying a security feature or a scalability feature.

Date Reçue/Date Received 2023-04-14

4. The data processing apparatus of claim 1, wherein the plurality of classes represent different levels of stack elements for the service tier.

5. The data processing apparatus of claim 1, wherein the plurality of classes include core abstract classes representing at least one of the application, deployment, application server, platform runtime, operating system and database server, and each of the core abstract classes include sub-classes corresponding to the micro-blueprints for stack elements for the service tier.

6. The data processing apparatus of claim 1, wherein the micro-blueprint assembler includes:

an application blueprint assembler configured to assemble micro-blueprints corresponding to the web tier, the application tier, and the database tier of the application; and

a functional component blueprint assembler configured to assemble micro-blueprints corresponding to stack elements for each of the web tier, the application tier, and the database tier.

7. The data processing apparatus of claim 6, wherein the application blueprint assembler is configured to obtain the micro-blueprints by obtaining micro-blueprints based on a matching of required capabilities and available capabilities of the micro-blueprints.

8. The data processing apparatus of claim 6, wherein the functional component blueprint assembler is configured to assemble the micro-blueprints, including:

obtaining micro-blueprints corresponding to the stack elements for each of the web tier, the application tier, and the database tier according to the request and the relational information.

Date Reçue/Date Received 2023-04-14

9. The data processing apparatus of claim 6, wherein the application blueprint assembler is configured to assemble the micro-blueprints, including:

obtaining micro-blueprints from the micro-blueprint database using a heuristic based best-first search algorithm.

10. The data processing apparatus of claim 1, wherein the micro-blueprint assembler is configured to generate a list of application blueprints in an order of suitability that achieves the request.

11. A method for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the method including:

receiving a request for automated blueprint assembly for automatically assembling an application blueprint of the application, the request specifying at least one feature;

accessing a model database storing model data, the model data including a plurality of classes arranged in a hierarchy with relational information specifying relationship links among the plurality of classes;

accessing a micro-blueprint database configured to store a plurality of micro-blueprints, each micro-blueprint corresponding to a functional component corresponding to a service tier comprising one of a web tier, an application tier, and a database tier, each micro-blueprint being annotated with one or more classes of the plurality of classes from the model data of the model database; and

generating the application blueprint by assembling micro-blueprints from the micro-blueprint database, including: applying the at least one feature to the classes of the model data of the model database to obtain a class that is relevant to the at least one feature of the request, querying the micro-blueprint database to obtain at least one micro-blueprint having the obtained class and at least one micro-blueprint linked to the obtained class by the relational information.

Date Reçue/Date Received 2023-04-14

12. The method of claim 11, wherein the request also specifies at least one constraint and environment.

13. The method of claim 11, wherein the at least one feature is a non-functional feature, the non-functional feature specifying a security feature or a scalability feature.

14. The method of claim 11, wherein the plurality of classes represent different levels of stack elements for the service tier.

15. The method of claim 11, wherein the plurality of classes include core abstract classes representing at least one of the application, deployment, application server, platform runtime, operating system and database server, and each of the core abstract classes include sub-classes corresponding to the micro-blueprints for stack elements for the service tier.

16. The method of claim 11, wherein generating the application blueprint includes:

assembling the micro-blueprints for each stack element for the service tier according to the request.

17. A non-transitory computer-readable medium storing instructions that when executed cause one or more processors to perform a process for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the instructions comprising instructions to:

receive a request for automated blueprint assembly for automatically assembling an application blueprint of the application, the request specifying at least one feature;

access a model database storing model data, the model data including a plurality of classes arranged in a hierarchy with relational information specifying relationship links among the plurality of classes;

access a micro-blueprint database configured to store a plurality of micro-blueprints, each micro-blueprint corresponding to a functional component corresponding to a service tier comprising one of a web tier, an application tier, and a database tier, each

Date Reçue/Date Received 2023-04-14

micro-blueprint being annotated with one or more classes of the plurality of classes from the model data of the model database; and

generate the application blueprint for the application by assembling micro-blueprints from the micro-blueprint database, including: apply the at least one feature to the classes of the model data of the model database to obtain a class that is relevant to the at least one feature of the request, query the micro-blueprint database to obtain at least one micro-blueprint having the obtained class and at least one micro-blueprint linked to the obtained class by the relational information.

18. The non-transitory computer-readable medium of claim 17, wherein the micro-blueprints are re-usable templates representing building blocks of the application.

19. A data processing apparatus for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the data processing apparatus including:

at least one processor;

a non-transitory computer-readable medium storing instructions that when executed by the at least one processor are configured to implement:

a micro-blueprint assembler configured to generate at least one blueprint for the application, and to receive a request for automated blueprint assembly for assembling the application, the request specifying at least one constraint;

a model database configured to store model data, the model data including an arrangement of a plurality of classes with relational information specifying relationships among the plurality of classes;

a micro-blueprint database configured to store a plurality of micro-blueprints, the plurality of micro-blueprints including first micro-blueprints providing one or more functional features, each first micro-blueprint corresponding to an individual service tier, the plurality of micro-blueprints including second micro-blueprints, each second micro-blueprint corresponding to an individual stack element, the first and second micro-blueprints being annotated with class information of the plurality of classes such that the first and second micro-blueprints are mapped to one or more classes of the model data,

Date Reçue/Date Received 2023-04-14

the micro-blueprint assembler configured to query the micro-blueprint database to obtain a first subset of the first micro-blueprints for service tiers of the application based on a matching of the one or more functional features and a second subset of the second micro-blueprints for stack elements for the service tiers such that the first and second subsets meet the at least one constraint of the request and the relational information of the model data, the micro-blueprint assembler configured to assemble at least one complete blueprint for the application from the first and second subsets.

20. The data processing apparatus of claim 19, wherein the at least one constraint of the request includes at least one functional feature and at least one non-functional feature, the at least one non-functional feature specifying a security feature or a scalability feature.

21. The data processing apparatus of claim 19, wherein the service tiers include a web tier, an application tier, and a database tier.

22. The data processing apparatus of claim 19, wherein each of the plurality of classes includes at least one class property, the first and second micro-blueprints being annotated with the at least one class property from one or more of the plurality of classes.

23. The data processing apparatus of claim 19, wherein the plurality of classes represent components of the application, the components including operating systems, database servers, application servers, execution environments, and deployment artifacts, the plurality of classes including abstracted domain knowledge information for the components.

24. The data processing apparatus of claim 19, wherein the one or more functional features are annotated with class properties of the plurality of classes from the model data, the micro-blueprint assembler being configured to obtain the first subset of the first micro-blueprints based on the matching of the one or more functional features of the first

Date Reçue/Date Received 2023-04-14

micro-blueprints and a matching of the at least one constraint of the request to the class properties.

25. The data processing apparatus of claim 19, wherein the micro-blueprint assembler is configured to obtain the second subset of the second micro-blueprints using the relational information of the model data as a guide for determining which second micro-blueprint to evaluate in view of the at least one constraint of the request.

26. The data processing apparatus of claim 19, wherein the relational information specifies dependencies between the plurality of classes, the micro-blueprint assembler being configured to query a class derived from the request from the model data of the model database, and then query the plurality of micro-blueprints having the obtained class from the micro-blueprint database.

27. The data processing apparatus of claim 19, wherein the plurality of micro-blueprints are re-usable micro-blueprints such that the plurality of micro-blueprints are available for assembling a secondary application.

28. The data processing apparatus of claim 19, wherein the micro-blueprint assembler is configured to obtain the first subset of micro-blueprints using an artificial intelligence (AI) search algorithm.

29. The data processing apparatus of claim 19, wherein the micro-blueprint assembler is configured to generate multiple complete blueprints for the application including a first complete blueprint and a second complete blueprint, the micro-blueprint assembler being configured to provide the first and second complete blueprint according to a level of suitability that achieves the at least one constraint of the request.

30. A method for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the method being performed by at least one processor, the method comprising:

Date Reçue/Date Received 2023-04-14

receiving a request for automated blueprint assembly for assembling the application, the request specifying at least one constraint;

accessing a model database configured to store model data, the model data including an arrangement of a plurality of classes with relational information specifying relationships among the plurality of classes;

accessing a micro-blueprint database configured to store a plurality of micro-blueprints, the plurality of micro-blueprints including first micro-blueprints providing one or more functional features each first micro-blueprint corresponding to an individual service tier, the plurality of micro-blueprints including second micro-blueprints, each second micro-blueprint corresponding to an individual stack element, the first and second micro-blueprints being annotated with class information of the plurality of classes such that the first and second micro-blueprints are mapped to one or more classes of the model data;

querying the micro-blueprint database to obtain a first subset of the first micro-blueprints for service tiers of the application based on a matching of the one or more functional features and a second subset of the second micro-blueprints for stack elements for the service tiers such that the first and second subsets meet the at least one constraint of the request and the relational information of the model data; and

assembling at least one complete blueprint for the application from the first and second subsets.

31. The method of claim 30, wherein the at least one constraint of the request includes at least one functional feature and at least one non-functional feature, the at least one non-functional feature specifying a security feature or a scalability feature.

32 The method of claim 30, wherein the service tiers include a web tier and an application tier.

33. The method of claim 30, wherein each of the plurality of classes includes at least one class property, the first and second micro-blueprints being annotated with the at least one class property from one or more of the plurality of classes.

Date Reçue/Date Received 2023-04-14

34. The method of claim 30, wherein the plurality of classes represent components of the application, the components including operating systems, database servers, application servers, execution environments, and deployment artifacts, the plurality of classes including abstracted domain knowledge information for the components.

35. The method of claim 30, wherein the one or more functional features are annotated with class properties of the plurality of classes from the model data, wherein the querying includes obtaining the first subset of the first micro-blueprints based on the matching of the one or more functional features of the first micro-blueprints and a matching of the at least one constraint of the request to the class properties.

36. The method of claim 30, wherein the querying includes obtaining the second subset of the second micro-blueprints using the relational information of the model data as a guide for determining which second micro-blueprint to evaluate in view of the at least one constraint of the request.

37. A non-transitory computer-readable medium storing instructions that when executed cause one or more processors to perform a process for automated blueprint assembly, a functional blueprint that defines an architectural view or a structure of an application or a service in terms of tiers, the instructions comprising instructions to:

receive a request for automated blueprint assembly for assembling the application, the request specifying at least one constraint;

access a model database configured to store model data, the model data including an arrangement of a plurality of classes with relational information specifying relationships among the plurality of classes;

access a micro-blueprint database configured to store a plurality of micro-blueprints, the plurality of micro-blueprints including first micro-providing one or more functional features, each first micro-blueprint corresponding to an individual service tier, the plurality of micro-blueprints including second micro-blueprints, each second micro-blueprint corresponding to an individual stack element, the first and second micro-

Date Reçue/Date Received 2023-04-14

blueprints being annotated with class information of the plurality of classes such that the first and second micro-blueprints are mapped to one or more classes of the model data;

query the micro-blueprint database to obtain a first subset of the first micro-blueprints for service tiers of the application based on a matching of the one or more functional features and a second subset of the second micro-blueprints for stack elements for the service tiers such that the first and second subsets meet the at least one constraint of the request and the relational information of the model data; and

assemble at least one complete blueprint for the application from the first and second subsets.

38. The non-transitory computer-readable medium of claim 37, wherein at least one constraint of the request includes at least one functional feature and at least one non-functional feature, the at least one non-functional feature specifying a security feature or a scalability feature.

Date Reçue/Date Received 2023-04-14

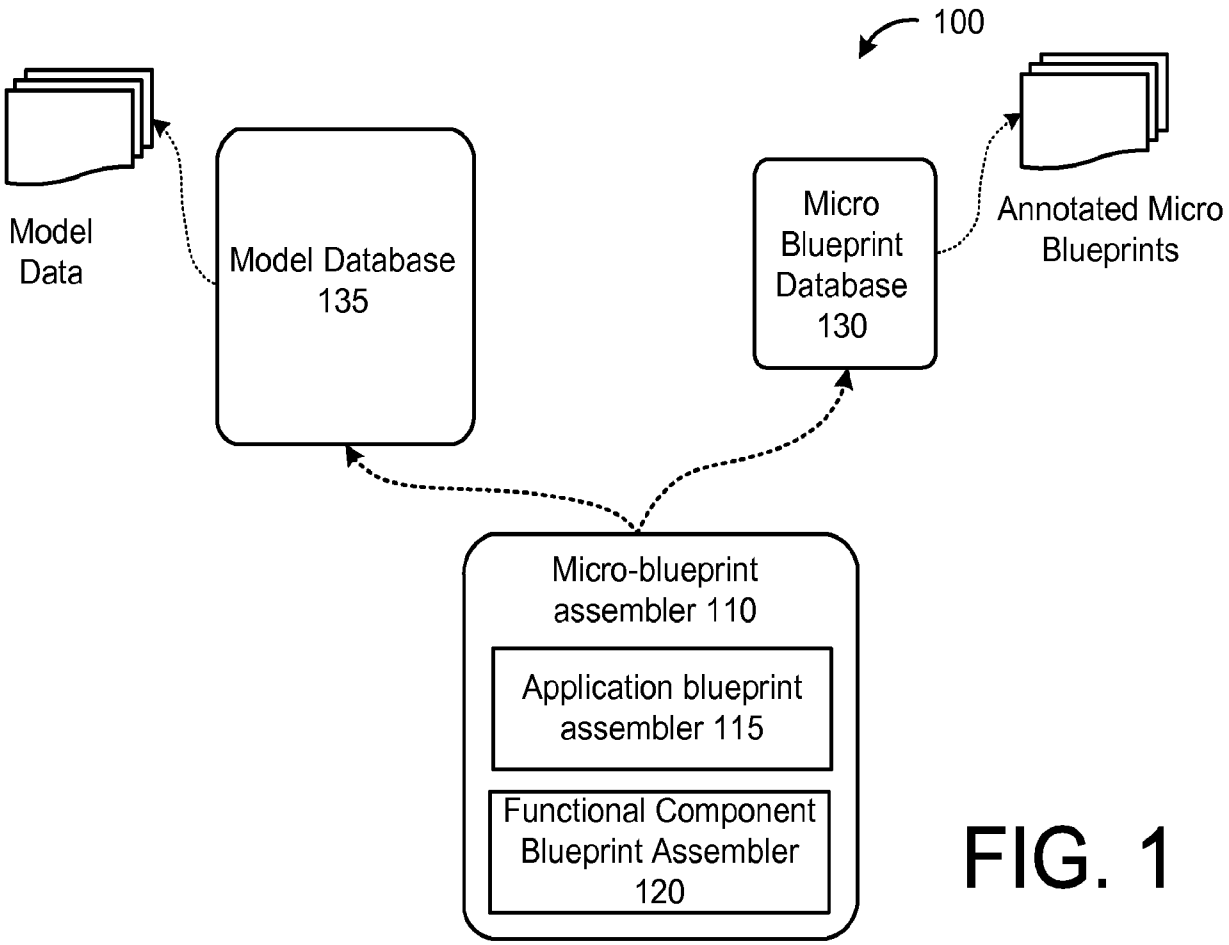

100
Model Data
Model Database 135
Micro Blueprint Database 130
Annotated Micro Blueprints
Micro-blueprint assembler 110
Application blueprint assembler 115
Functional Component Blueprint Assembler 120

FIG. 1

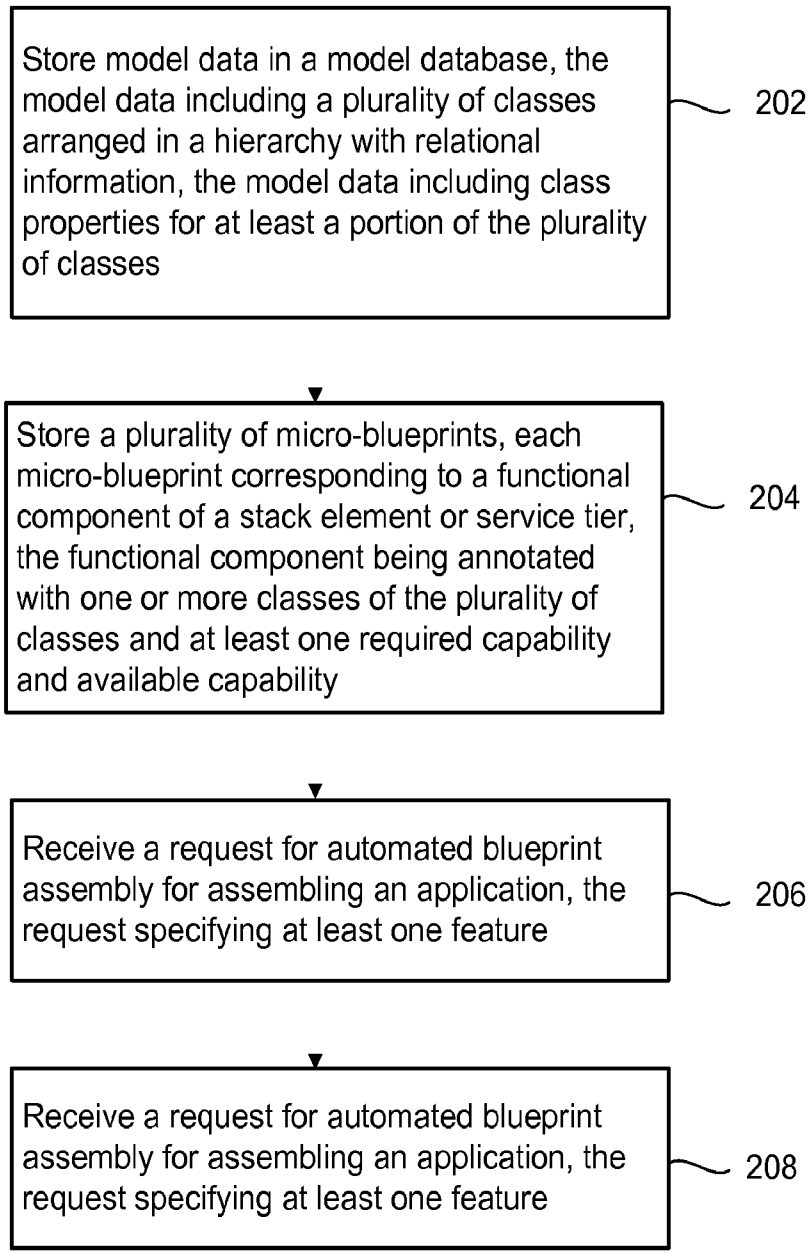

Store model data in a model database, the model data including a plurality of classes arranged in a hierarchy with relational information, the model data including class properties for at least a portion of the plurality of classes
202
Store a plurality of micro-blueprints, each micro-blueprint corresponding to a functional component of a stack element or service tier, the functional component being annotated with one or more classes of the plurality of classes and at least one required capability and available capability
204
Receive a request for automated blueprint assembly for assembling an application, the request specifying at least one feature
206
Receive a request for automated blueprint assembly for assembling an application, the request specifying at least one feature
208

FIG. 2

300

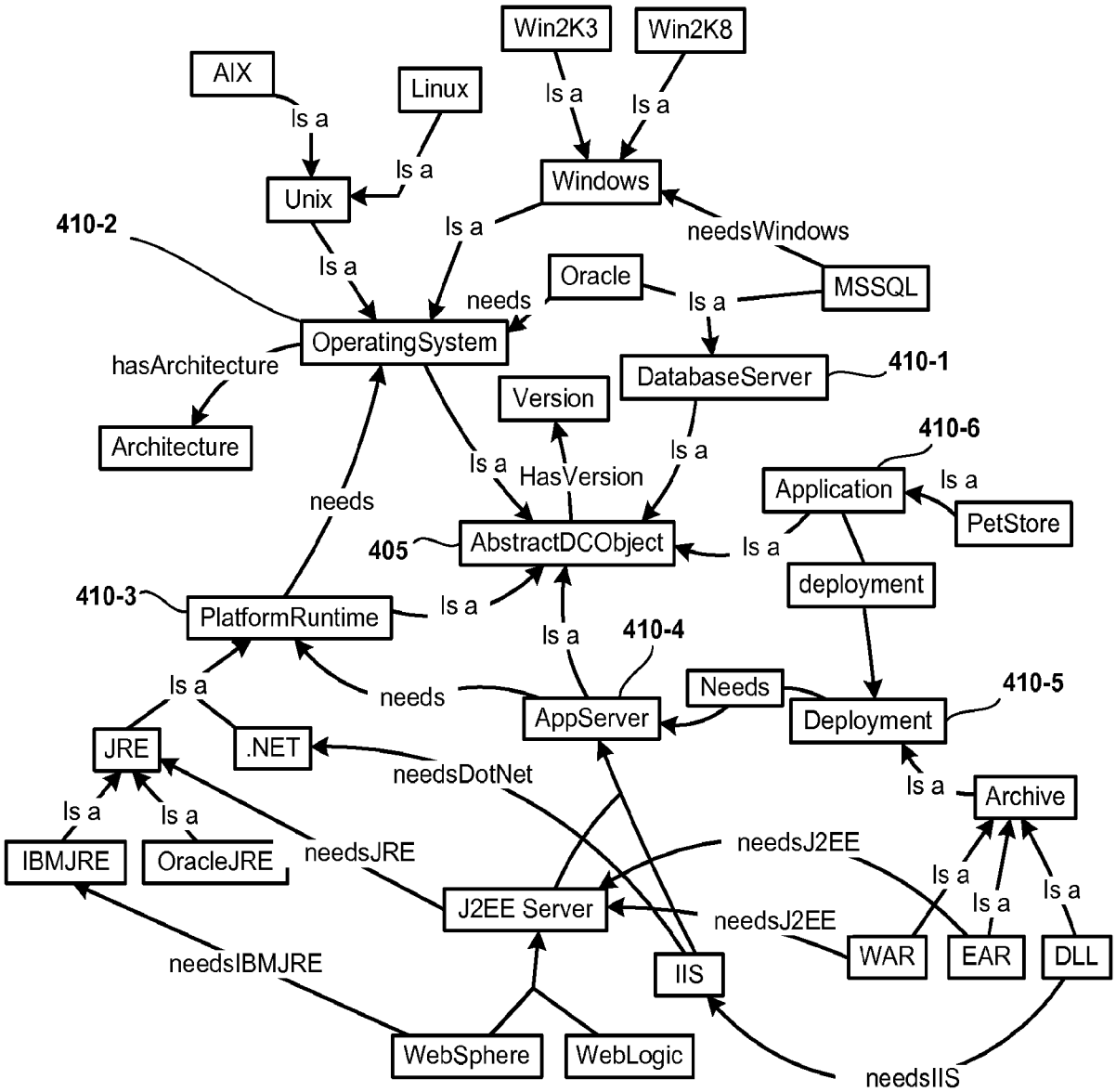

Win2K3
Win2K8
AIX
Linux
Is a
Is a
Is a
Is a
Windows
Unix
410-2
Is a
Is a
needsWindows
Oracle
MSSQL
Is a
needs
OperatingSystem
hasArchitecture
DatabaseServer
410-1
Version
Architecture
410-6
Is a
HasVersion
Is a
Application
Is a
PetStore
needs
405
AbstractDCObject
Is a
deployment
410-3
PlatformRuntime
Is a
Is a
410-4
Is a
needs
Needs
410-5
AppServer
Deployment
JRE
.NET
needsDotNet
Is a
Archive
Is a
Is a
IBMJRE
OracleJRE
needsJRE
needsJ2EE
Is a
Is a
Is a
J2EE Server
needsJ2EE
WAR
EAR
DLL
needsIBMJRE
IIS
WebSphere
WebLogic
needsIIS

FIG. 3

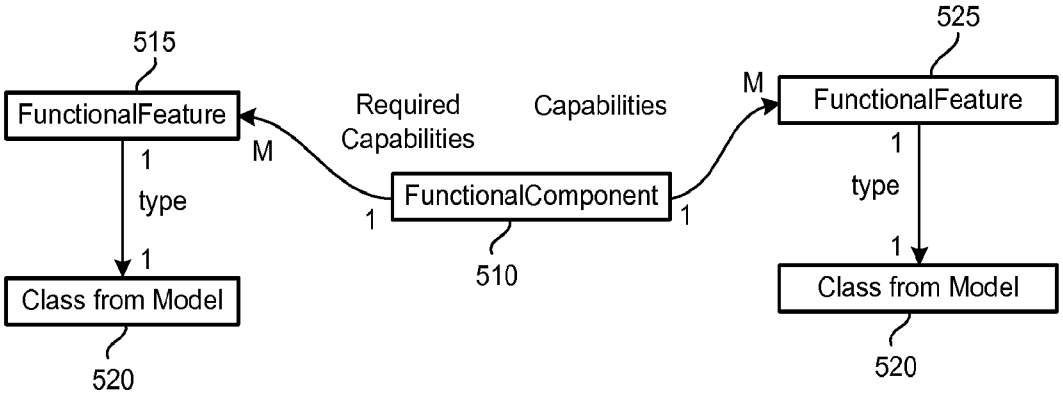

515
FunctionalFeature
1
type
1
Class from Model
520
M
Required Capabilities
1
FunctionalComponent
510
Capabilities
M
1
525
FunctionalFeature
1
type
1
Class from Model
520

FIG. 4

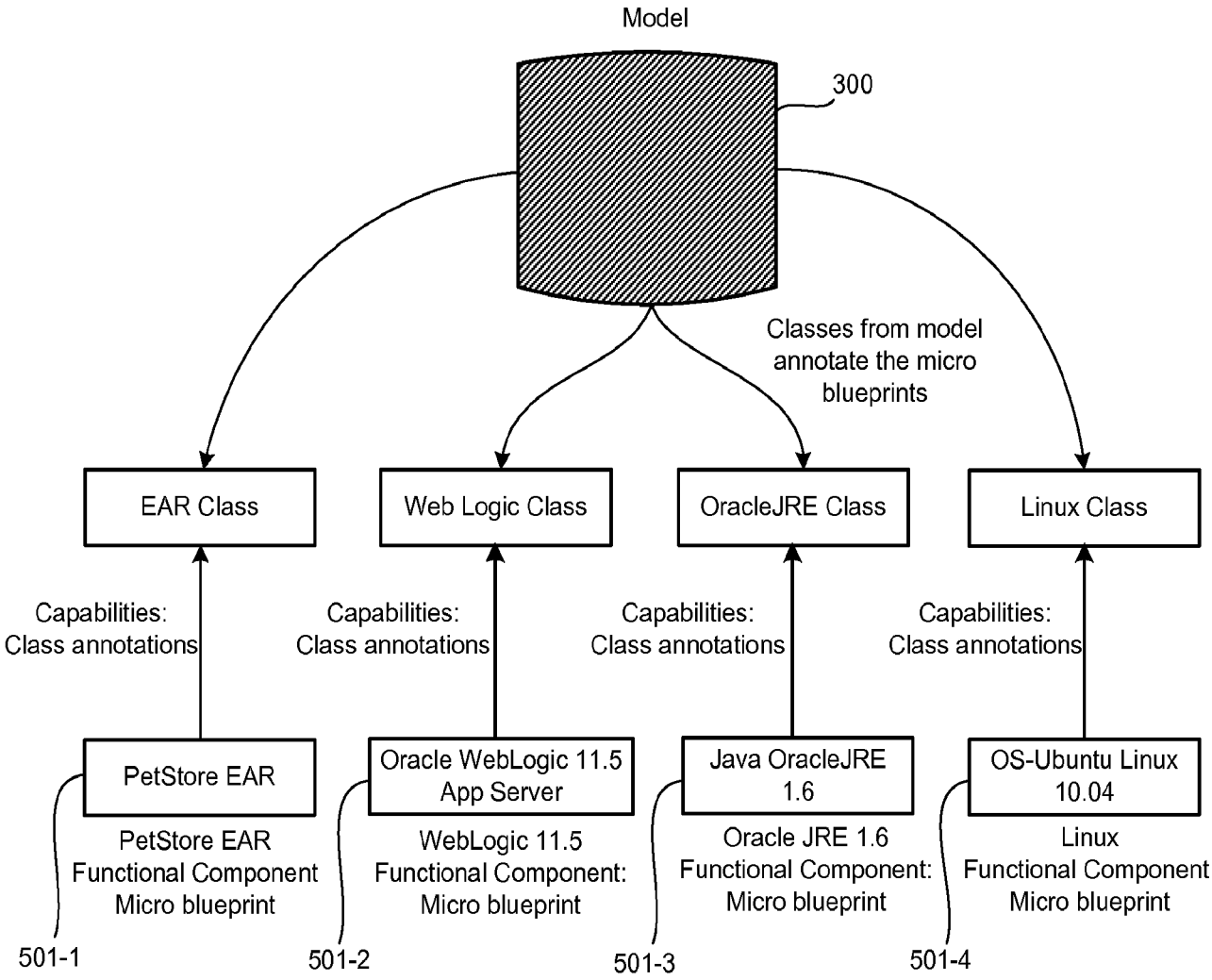

Model
300
Classes from model annotate the micro blueprints
EAR Class
Web Logic Class
OracleJRE Class
Linux Class
Capabilities: Class annotations
Capabilities: Class annotations
Capabilities: Class annotations
Capabilities: Class annotations
PetStore EAR
Oracle WebLogic 11.5 App Server
Java OracleJRE 1.6
OS-Ubuntu Linux 10.04
PetStore EAR Functional Component Micro blueprint
WebLogic 11.5 Functional Component: Micro blueprint
Oracle JRE 1.6 Functional Component: Micro blueprint
Linux Functional Component: Micro blueprint
501-1
501-2
501-3
501-4

FIG. 5

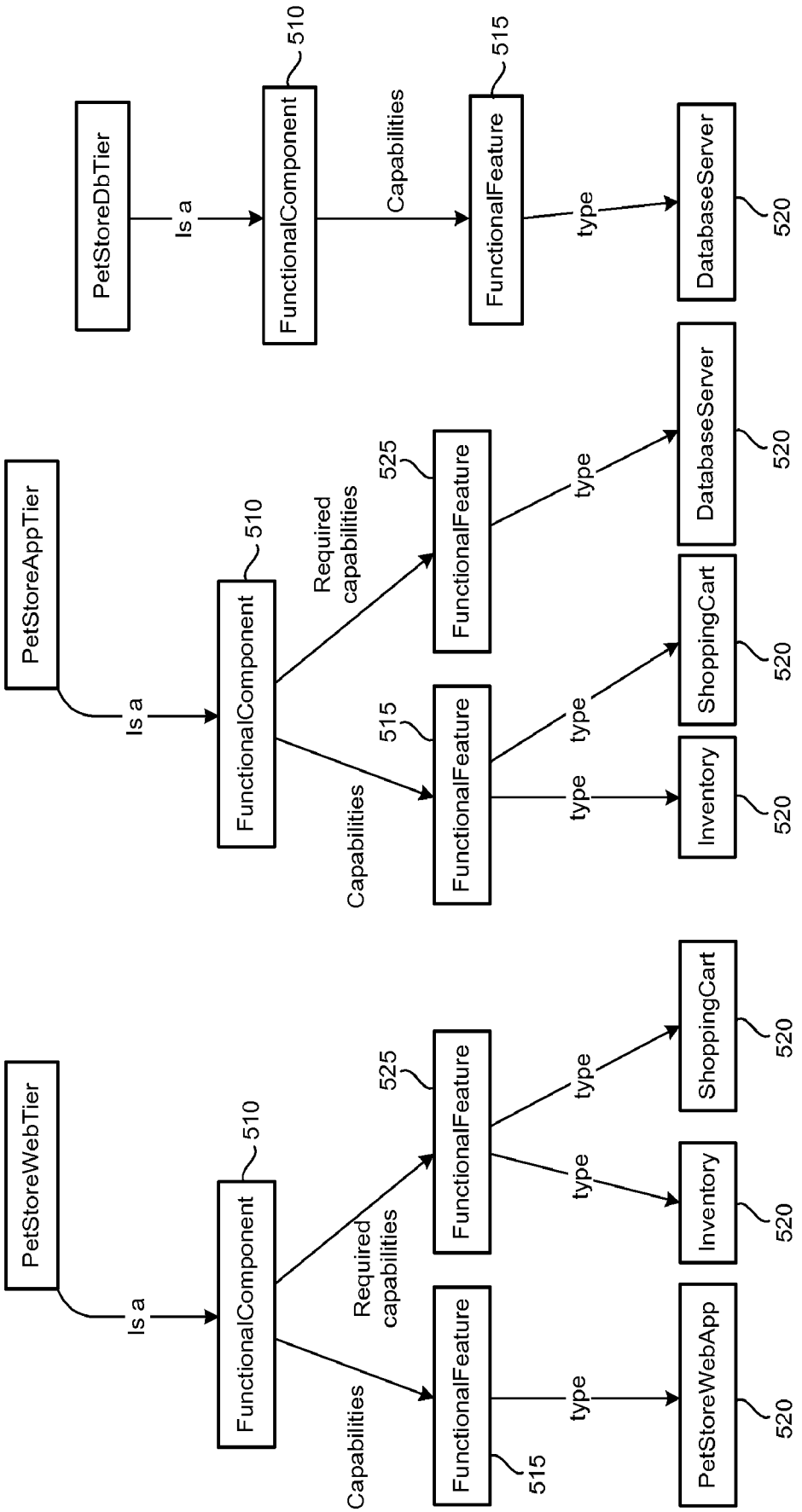

PetStoreDbTier
Is a
FunctionalComponent
510
Capabilities
FunctionalFeature
515
type
DatabaseServer
520
PetStoreAppTier
Is a
FunctionalComponent
510
Required capabilities
525
FunctionalFeature
type
DatabaseServer
520
Capabilities
515
FunctionalFeature
type
ShoppingCart
520
type
Inventory
520
PetStoreWebTier
Is a
FunctionalComponent
510
Required capabilities
525
FunctionalFeature
type
ShoppingCart
520
type
Inventory
520
Capabilities
515
FunctionalFeature
type
PetStoreWebApp
520

FIG. 6

5/7

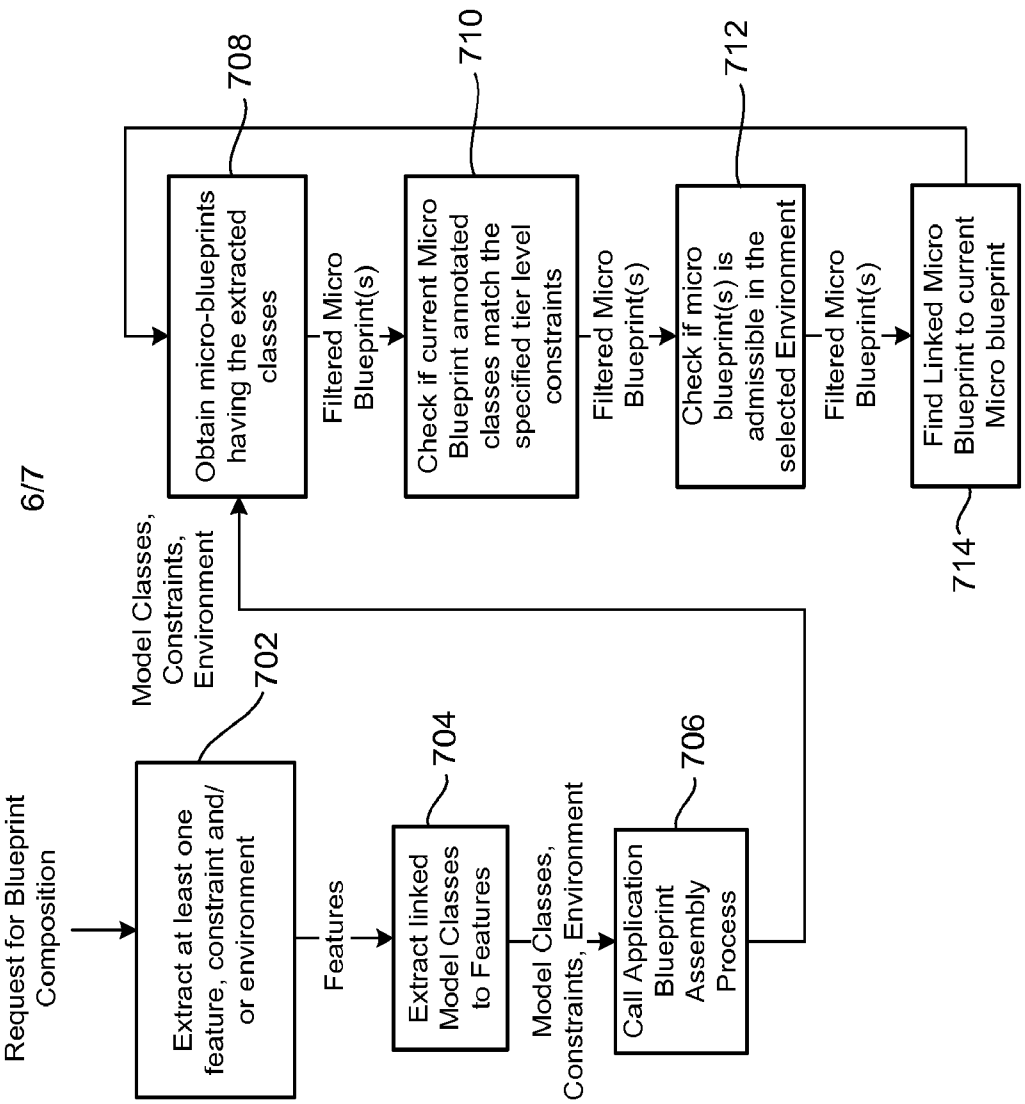

6/7
Request for Blueprint Composition
Extract at least one feature, constraint and/ or environment
Features
Extract linked Model Classes to Features
Model Classes, Constraints, Environment
Call Application Blueprint Assembly Process
Model Classes, Constraints, Environment
Obtain micro-blueprints having the extracted classes
Filtered Micro Blueprint(s)
Check if current Micro Blueprint annotated classes match the specified tier level constraints
Filtered Micro Blueprint(s)
Check if micro blueprint(s) is admissible in the selected Environment
Filtered Micro Blueprint(s)
Find Linked Micro Blueprint to current Micro blueprint
702
704
706
708
710
712
714

FIG. 7

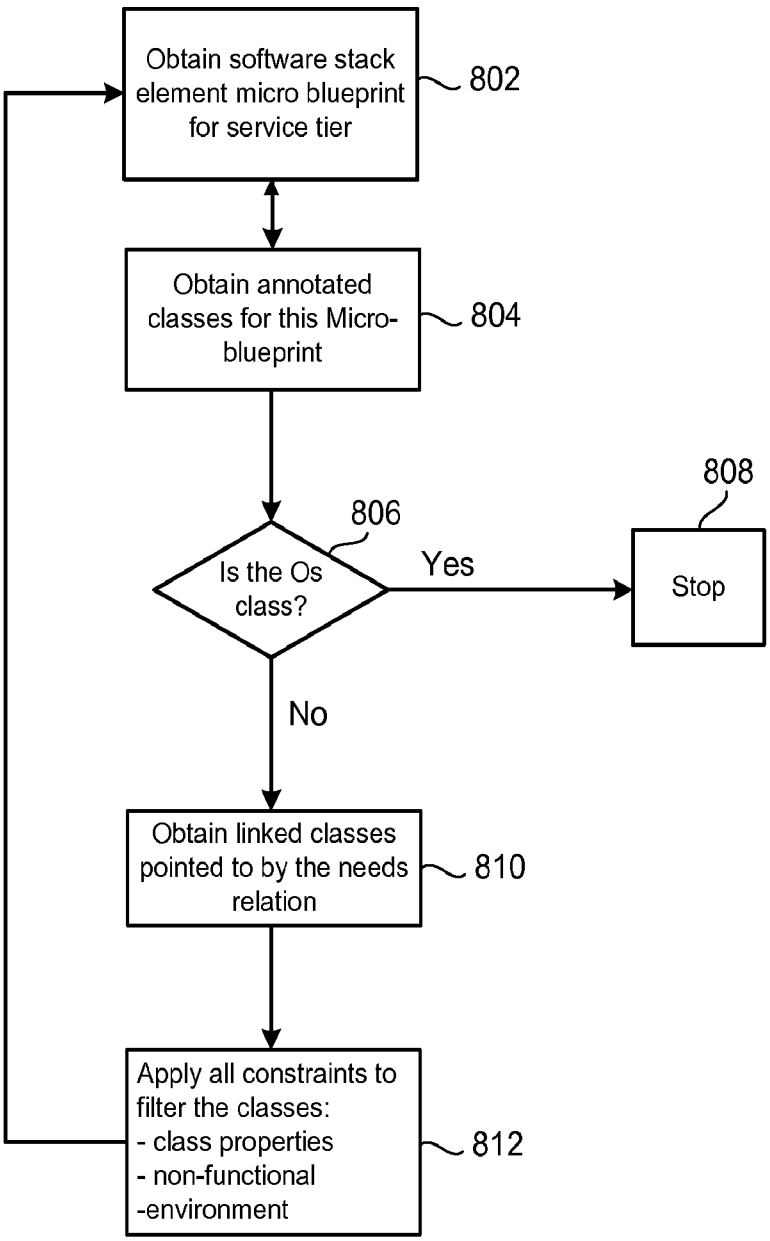
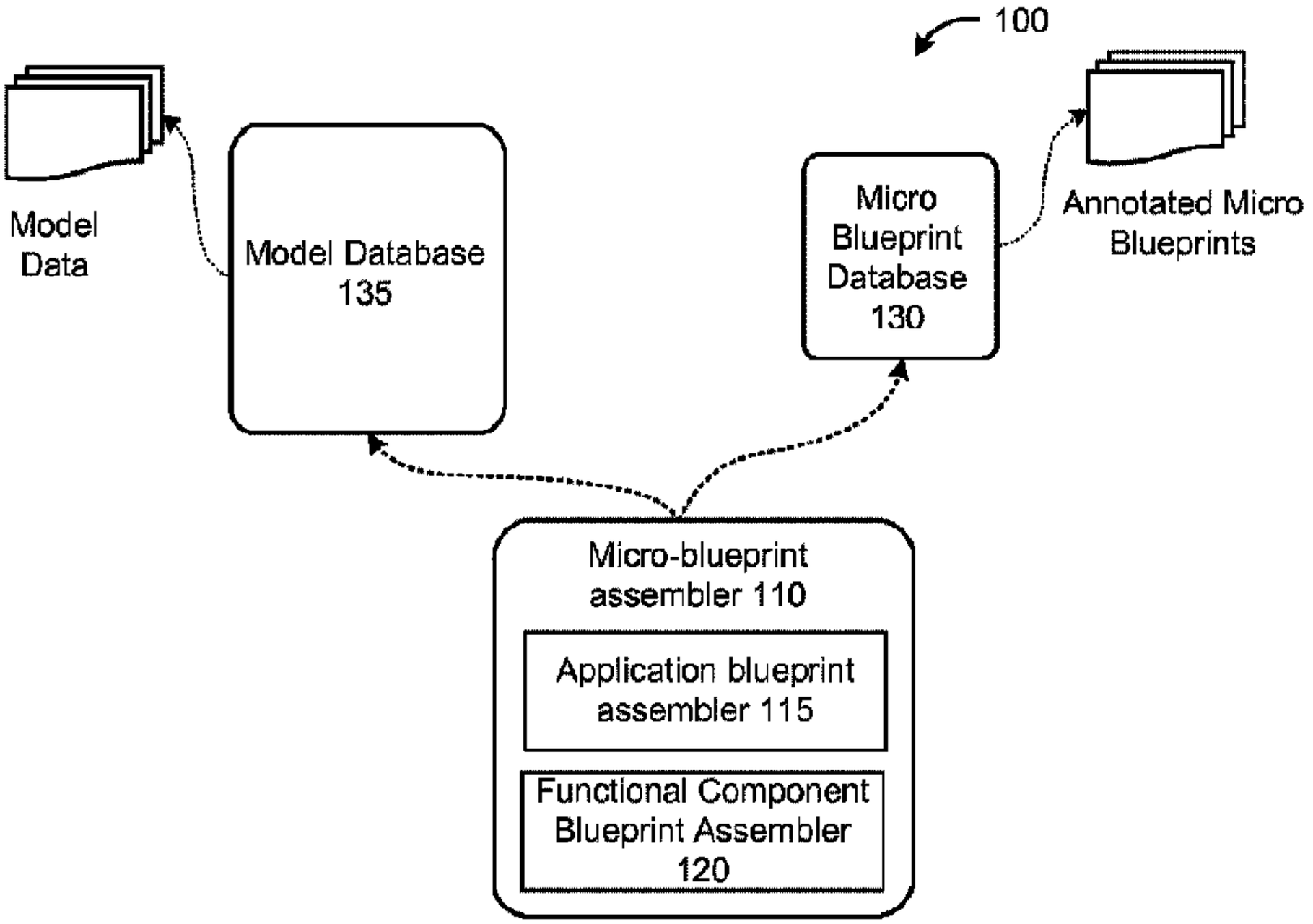

Obtain software stack element micro blueprint for service tier
802
Obtain annotated classes for this Micro-blueprint
804
806
Is the Os class?
Yes
808
Stop
No
Obtain linked classes pointed to by the needs relation
810
Apply all constraints to filter the classes:
- class properties
- non-functional
-environment
812

FIG. 8


100
Model Data
Model Database 135
Micro Blueprint Database 130
Annotated Micro Blueprints
Micro-blueprint assembler 110
Application blueprint assembler 115
Functional Component Blueprint Assembler 120