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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 April 2009 (30.04.2009)

PCT

(10) International Publication Number
WO 2009/055754 A2

(51) International Patent Classification:
G06F 15/16 (2006.01) *G06F 17/00* (2006.01)

(21) International Application Number:
PCT/US2008/081234

(22) International Filing Date: 26 October 2008 (26.10.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/925,591 26 October 2007 (26.10.2007) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **SEDUKHIN, Igor**; c/o Microsoft Corporation, LCA, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **ESHNER, Daniel**; c/o Microsoft Corporation, LCA, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SWARTZ, Stephen T.**; c/o Microsoft Corporation, LCA, International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

(74) Agent: **ALLEN, Michael, B.**; Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399 (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, BR, BW, BY, BZ, CA, CH, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: MODELING AND MANAGING HETEROGENEOUS APPLICATIONS

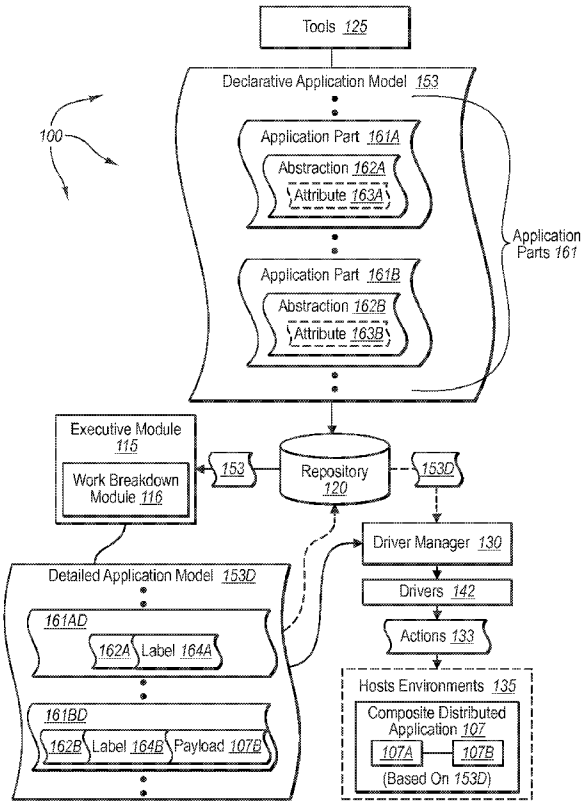

FIG. 1A

(57) Abstract: The present invention extends to methods, systems, and computer program products for modeling and managing heterogeneous applications. Application intent can be described in a relatively straight forward manner that abstracts underlying implementation details. Thus, application developers can develop applications without necessarily having to know extensive details of an underlying implementation environment. In any event, an application can be executed in different implementation environments without requiring changes to the corresponding model.

WO 2009/055754 A2

FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

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

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *without international search report and to be republished upon receipt of that report*

MODELING AND MANAGING HETEROGENEOUS APPLICATIONS

BACKGROUND

[0001] **1. Background and Relevant Art**

[0002] Computer systems and related technology affect many aspects of society. Indeed, the computer system's ability to process information has transformed the way we live and work. Computer systems now commonly perform a host of tasks (e.g., word processing, scheduling, accounting, etc.) that prior to the advent of the computer system were performed manually. More recently, computer systems have been coupled to one another and to other electronic devices to form both wired and wireless computer networks over which the computer systems and other electronic devices can transfer electronic data. Accordingly, the performance of many computing tasks are distributed across a number of different computer systems and/or a number of different computing components.

[0003] As computerized systems have increased in popularity, so have the complexity of the software and hardware employed within such systems. In general, the need for seemingly more complex software continues to grow, which further tends to be one of the forces that push greater development of hardware. For example, if application programs require too much of a given hardware system, the hardware system can operate inefficiently, or otherwise be unable to process the application program at all. Recent trends in application program development, however, have removed many of these types of hardware constraints at least in part using distributed application programs. In general, distributed application programs comprise components that are executed over several different hardware components, often on different computer systems in a tiered environment.

[0004] With distributed application programs, the different computer systems may communicate various processing results to each other over a network. Along these lines, an organization will employ a distributed application server to manage several different distributed application programs over many different computer systems. For example, a user might employ one distributed application server to manage the operations of an ecommerce application program that is executed on one set of different computer systems. The user might also use the distributed application server to manage execution of customer management application programs on the same or even a different set of computer systems.

[0005] Of course, each corresponding distributed application managed through the distributed application server can, in turn, have several different modules and components that are executed on still other different (and potentially differently configured) computer systems over different (and potentially differently configured) network connections. Thus, while this ability to combine processing power through several different computer systems can be an advantage, there are various complexities associated with distributing application program modules.

[0006] In many environments, there is multiple diverse implementation technologies composed into a distributed application. Nonetheless, the components of a distributed application operating in such an environment must behave coherently and reliably. Thus, an administrator or other user typically deals with each part of a distributed application individually and manually to make then work together. For example, to implement a distributed application an administrator or other user can be required to program individual parts of the distributed applications (e.g., Web sites, Web services, Workflows, Databases, etc.) using multiple different frameworks and hosting technologies (e.g., Web services, App servers, Database servers, etc.). The

implementation-specific parts of the distributed application are then manually configured to connect and exchange data.

[0007] Subsequently, the administrator or other user can create text documents that describe how and when to deploy and activate parts of an application and what to do when failures occur. It is generally a manual task to act on what is described in these text documents. Accordingly the process of creating and executing a distributed application is typically expensive, error prone and not change-friendly.

BRIEF SUMMARY

[0008] The present invention extends to methods, systems, and computer program products for modeling and managing heterogeneous applications. The components of a composite distributed application are described in a declarative model. The declarative model includes a description of a plurality of different application parts of the composite distributed application. One or more abstractions are inserted into the declarative model. Each abstraction describes the intended behavior of a corresponding application part in an implementation independent manner. Thus, the abstractions extend the declarative model to make the declarative model executable.

[0009] In some embodiments, an implementation attribute is set on at least one abstraction. The implementation attribute expressly indicates how the abstraction is to be tagged. In other embodiments, no implementation attributes are set. However, in either of the embodiments, each of the one or more abstractions is tagged with a label. Each label includes implementation specific details that indicate how a plurality of different technologies are to be used to implement the corresponding application part.

[0010] When an abstraction includes an implementation attribute, the abstraction can be tagged with a label in accordance with the express indication in the implementation attribute. On the other hand, when an abstraction does not include an implementation attribute, a label can be inferred (e.g., based on available resources of different technologies, from prior system settings, from default values, etc.) and then tagged to the abstraction.

[0011] The declarative model is interpreted to implement an executable instance of the composite distributed application. The executable instance of the composite distributed application is implemented using the plurality of technologies in accordance with the implementation specific details included in the labels.

[0012] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0013] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0015] Figures 1A illustrates an example computer architecture that facilitates modeling and managing heterogeneous applications.

[0016] Figure 1B illustrates an example declarative model for a composite distributed application.

[0017] Figure 1C illustrates an example declarative model for a composite distributed application including intended behaviors.

[0018] Figure 1D illustrates an example declarative model for a composite distributed application including intended behaviors with labels indicating implementation details.

[0019] Figure 1E illustrates another view of some of the components from the computer architecture of Figure 1A interpreting a declarative application model.

[0020] Figure 2 illustrates a flow chart of an example method for implementing a composite distributed application.

[0021] Figure 3 illustrates an example computer architecture for translating declarative application models.

DETAILED DESCRIPTION

[0022] The present invention extends to methods, systems, and computer program products for modeling and managing heterogeneous applications. The components of a composite distributed application are described in a declarative model. The declarative model includes a description of a plurality of different application parts of the composite distributed application. One or more abstractions are inserted into the declarative model. Each abstraction describes the intended behavior of a corresponding application part in an implementation independent manner. Thus, the abstractions extend the declarative model to make the declarative model executable.

[0023] In some embodiments, an implementation attribute is set on at least one abstraction. The implementation attribute expressly indicates how the abstraction is to be tagged. In other embodiments, no implementation attributes are set. However, in either of the embodiments, each of the one or more abstractions is tagged with a label. Each label includes implementation specific details that indicate how a plurality of different technologies are to be used to implement the corresponding application part.

[0024] When an abstraction includes an implementation attribute, the abstraction can be tagged with a label in accordance with the express indication in the implementation attribute. On the other hand, when an abstraction does not include an implementation attribute, a label can be inferred (e.g., based on available resources of different technologies, from prior system settings, from default values, etc.) and then tagged to the abstraction.

[0025] The declarative model is interpreted to implement an executable instance of the composite distributed application. The executable instance of the composite

distributed application is implemented using the plurality of technologies in accordance with the implementation specific details included in the labels.

[0026] Embodiments of the present invention may comprise or utilize a special purpose or general-purpose computer including computer hardware, as discussed in greater detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: physical storage media and transmission media.

[0027] Physical storage media includes RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0028] A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links

which can be used to carry or desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0029] Further, it should be understood, that upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to physical storage media. For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile physical storage media at a computer system. Thus, it should be understood that physical storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0030] Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0031] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system

configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0032] Figure 1A illustrates an example computer architecture 100 that facilitates modeling and managing heterogeneous applications. Depicted in computer architecture 100 are tools 125, repository 120, executive module 115, driver manager 130, drivers 142, and host environments 135. Each of the depicted components can be connected to one another over a network, such as, for example, a Local Area Network ("LAN"), a Wide Area Network ("WAN"), and even the Internet. Accordingly, each of the depicted components as well as any other connected components, can create message related data and exchange message related data (e.g., Internet Protocol ("IP") datagrams and other higher layer protocols that utilize IP datagrams, such as, Transmission Control Protocol ("TCP"), Hypertext Transfer Protocol ("HTTP"), Simple Mail Transfer Protocol ("SMTP"), etc.) over the network.

[0033] As depicted, tools 125 can be used to write declarative models for applications and store declarative models, such as, for example, declarative application model 153, in repository 120. Declarative models are used to describe the structure and behavior of real-world running (deployed) applications. Thus, a user

(e.g., distributed application program developer) can use one or more of tools 125 to create declarative application model 153.

[0034] Generally, declarative models include one or more sets of high-level declarations expressing application intent for a distributed application. Thus, the high-level declarations generally describe operations and/or behaviors of one or more modules in the distributed application program. However, the high-level declarations do not necessarily describe implementation steps required to deploy a distributed application having the particular operations/behaviors (although they can if appropriate). For example, declarative application model 153 can express the generalized intent of a workflow, including, for example, that a first Web service be connected to a database. However, declarative application model 153 does not necessarily describe how (e.g., protocol), nor where (e.g., address) the Web service and database are to be connected to one another. In fact, how and where is determined based on which computer systems the database and the Web service are deployed.

[0035] In some embodiments, a composite distributed application is described using different application parts within a declarative application model. Each application part describes a portion of a distributed application, which can be composited together to provide desired functionality. For example, declarative application model 153 includes application part 161A and application part 161B. Application part 161A and application part 161B each describe a different portion of a composite distributed application modeled by declarative application model 153. For example, application part 161A can describe a Web service and application part 161B can describe a database.

[0036] Each application part can also include an abstraction that indicates the intended behavior of the described application part. For example, abstraction 162A

and abstraction 162B can indicate the intended behavior of application part 161A and application part 161B respectively. However, abstractions do not necessarily indicate how intended behavior is to be implemented. For example, abstraction 162A can indicate a Web service internal behavior as a workflow, but not the environment where nor the technology with which the workflow is to be implemented.

[0037] Although in some embodiments, an abstraction also includes an attribute that expressly indicates implementation details for implementing intended behavior. For example, abstraction 162A can optionally include attribute 163A indicating implementation details for the intended behavior of application part 161A. Likewise, abstraction 162B can optionally include attribute 163B indicating implementation details for the intended behavior of application part 161B.

[0038] To deploy an application based on a declarative model, the declarative model can be sent to executive module 115. Executive module 115 can refine the declarative model until there are no ambiguities and the details are sufficient for drivers to consume. Thus, executive module 115 can receive and refine declarative model 153 so that declarative model 153 can be translated by drivers 142 (e.g., one or more technology-specific drivers) into a deployed application.

[0039] In general, "refining" a declarative model can include some type of work breakdown structure, such as, for example, progressive elaboration, so that the model declarations are sufficiently complete for translation by drivers 142. Since declarative models can be written relatively loosely by a human user (i.e., containing generalized intent instructions or requests), there may be different degrees or extents to which executive module 115 modifies or supplements a declarative model for deploying an application. Work breakdown module 116 can implement a work breakdown structure algorithm, such as, for example, a progressive elaboration algorithm, to

determine when an appropriate granularity has been reached and instructions are sufficient for drivers.

[0040] Executive module 115 can also account for dependencies and constraints included in a declarative model. For example, executive module 115 can be configured to refine declarative model 153 based on semantics of dependencies between elements in the declarative model 153 (e.g., one web service connected to another). Thus, executive module 115 and work breakdown module 116 can interoperate to output detailed application model 153D that provides driver 130 with sufficient information to deploy distributed application 107.

[0041] In additional or alternative implementations, executive module 115 can also be configured to refine the declarative model 153 based on some other contextual awareness. For example, executive module 115 can refine the declarative model based on the inventory of host environments 135 that may be available in the datacenter where a distributed application program will be deployed. Executive module 115 can reflect contextual awareness information in detailed application model 153D.

[0042] In addition, executive module 115 can be configured to fill in missing data regarding computer system assignments. For example, executive module 115 might identify a number of different distributed application program modules in declarative model 153 that have no requirement for specific computer system addresses or operating requirements. Thus, executive module 115 can assign distributed application program modules to an available host environment on a computer system. Executive module 115 can reason about the best way to fill in data in a refined declarative model 153. For example, as previously described, executive component 115 may determine and decide which transport to use for an endpoint based on

proximity of connection, or determine and decide how to allocate distributed application program modules based on factors appropriate for handling expected spikes in demand. Executive module 115 can then record missing data in detailed declarative model 153D (or segment thereof).

[0043] In addition or alternative implementations, executive module 115 can be configured to compute dependent data in the declarative model 153. For example, executive module 115 can compute dependent data based on an assignment of distributed application program modules to host environments on computer systems. Thus, executive module 115 can calculate URI addresses on the endpoints, and propagate the corresponding URI addresses from provider endpoints to consumer endpoints. In addition, executive module 115 may evaluate constraints in the declarative model 153. For example, the executive component 115 can be configured to check to see if two distributed application program modules can actually be assigned to the same machine, and if not, executive module 115 can refine detailed declarative model 153D to accommodate this requirement.

[0044] Accordingly, after adding appropriate data (or otherwise modifying/refining) to declarative model 153 (to create detailed application model 153D), executive component 115 can finalize the refined detailed application model 153D so that it can be translated by platform-specific drivers 130. To finalize or complete the detailed application model 153D, executive module 115 can, for example, partition a declarative application model into segments that can be targeted by any one or more platform-specific drivers 142. Thus, executive module 115 can tag each declarative application model (or segment thereof) with its target driver (e.g., the unique ID or an address of a platform-specific driver). Furthermore, executive module 115 can verify that a detailed application model (e.g., 153D) can actually be

translated by one or more platform-specific drivers, and, if so, pass the detailed application model (or segment thereof) to a particular platform-specific driver for translation.

[0045] For example, executive module 115 can be configured to tag application parts with labels indicating an intended implementation for an application part. As depicted, detailed application model 153D includes application parts 161AD and 161BD that have been refined to remove ambiguities. Corresponding abstractions 162A and 162B have been associated with labels 164A and 164B respectively. Label 164A indicates an intended implementation for the intended behavior indicated in abstraction 162A. Likewise, label 164B indicates an intended implementation for the intended behavior indicated in abstraction 162B.

[0046] When appropriate, implementation specific payloads can also be added to compliment the description of the intended behavior after an abstraction has been labeled with an intended implementation. For example, payload 107B can be added to application part 161BD. Implementation specific payloads can be used to leverage implementation specific features not represented in abstractions. For example, payload 107B can be used to leverage an intended implementation indicated by label 164B.

[0047] After refining a model, executive module 115 can forward the model to driver manger 130 or store the refined model back in repository 120 for later use. Thus, executive module 115 can forward detailed application model 153D to driver manager 130 or store detailed application model 153D in repository 120. When detailed application model 153D is stored in repository 120, it can be subsequently provided to driver manger 130 without further refinements.

[0048] Driver manager 130 can then take actions (e.g., actions 133) to implement a composite distributed application (e.g., composite distributed application 107, including executable application parts 107A and 107B) based on detailed application model 153D. Driver manager 130 interoperates with one or more drivers and translators to translate detailed application module 153D (or declarative application model 153) into one or more (e.g., platform-specific) actions 133. Actions 133 can be used to realize the intended behavior indicated in abstractions within corresponding implementation host environments indicated in associated labels.

[0049] Accordingly, composite distributed application 107 can be implemented in host environments 153. Each application part can be implemented in a separate host environment (indicated in an associated label) and connected to other application parts via corresponding consumer and provider endpoints. Interconnected consumer and provider endpoints can also be modeled.

[0050] For example, Figure 1B illustrates a first alternate view of declarative application model 153. In Figure 1B, declarative application model 153 models a Distributed Application named "MyBusinessProcessService". The distributed application model 153 models application part 161A and 161B. External provider endpoint 171 is configured to receive requests and delegate requests to provider endpoint 165A (a provider endpoint for application part 161A). Consumer endpoint 166A (a consumer endpoint for application part 161A) is connected to provider endpoint 165B (the provider endpoint for application part 161B). Accordingly, the first alternate view of declarative application model 153 in Figure 1B indicates an intent to have a composite distributed application that receives requests and is composed of two application parts: process and data.

[0051] Figure 1C illustrates a second alternate view of declarative model 153 including intended behaviors. In Figure 1C, application part 161A indicates that "MyProcess" is a Web service. Abstraction 162A indicates that the intended behavior of the Web service is a Work Flow including a flow (e.g., trivially, a sequence) of activities, long-running state, and activities themselves. Optionally, abstraction 162A can include attribute 163A expressly indicating an implementation environment for implementing Work Flow behavior.

[0052] Likewise, application part 162A indicates that "MyData" is a Database. Abstraction 162B indicates that the intended behavior of "MyData" is Data Entities including schema, relationships, and a static data. Optionally, abstraction 162B can include attribute 163B expressly indicating an implementation environment for implementing Data Entities behavior.

[0053] Figure 1D illustrates an alternate view of detailed application model 153D including intended behaviors with labels indicating implementation details. The alternate view of detailed application model 153D can result from executive module 115 refining the second alternate view of descriptive application model 153. In Figure 1D, label 166A indicates that the Work Flow behavior indicated in abstraction 162A is to be implemented in a "WCF/WF-WAS" (Windows® Workflow Foundation/Windows® Communication Foundation environment in a Windows® Activation Service host) environment. Likewise, label 166B indicates that the Data Entities behavior indicated in abstraction 162B is to be implemented in a "SQL" environment in a Microsoft SQL Server host.

[0054] An application part is further associated with a host indicator which can indicate a host (e.g., a specified computer system) that is to provide required implementation environment for the application part. For example, host indicator

167A indicates that WASHost 135B" is to provide a "WF/WCF-WAS" environment for implementing the intended Work Flow behaviors of abstraction 162A. Likewise, host indicator 167B indicates that "MySQLServer" is to provide an "SQL" environment for the intended Data Entities behaviors of abstraction 162B. The corresponding drivers can then process corresponding abstractions to implement the intended behavior in the indicated implementation environment in the indicated host. For example, WASHost 135B can be configured by the corresponding driver to provide a "WF/WCF-WAS" environment for implementing the intended Work Flow behaviors of abstraction 162A.

[0055] For example, Figure 1E illustrates another view of some of the components from computer architecture 100 interpreting declarative application model 153. As depicted, drivers 142 include a plurality of drivers including Axis-Tomcat driver 142A (for implementing intended behavior based on Apache Axis framework hosted in Apache Tomcat), WF/WCF-WAS driver 142B (for implementing intended behavior based on WF/WCF frameworks hosted in WAS), and ASMX-IIS Driver 142C (for implementing intended behavior based on Microsoft ASP.NET framework hosted in Microsoft Internet Information Server).

[0056] Host environments 135 depict resulting host environments that can be utilized by corresponding drives. For example, Axis-Tomcat driver 142A can translate intended behavior (e.g., from an abstraction) into actions for execution based on Apache Axis Framework 136A within Apache Tomcat Host 135A. Similarly, WF/WCF-WAS driver 142B can translate intended behavior into actions for execution based on WF/WCF Framework 136B within WASHost 135B. Likewise, ASMX-IIS Driver 142C can translate intended behavior into actions for execution based on ASP .Net framework 136C within IIS Host 135C.

[0057] Figure 2 illustrates a flow chart of an example method 200 for implementing a composite distributed application. Method 200 will be described with respect to the components and data of computer architecture 100.

[0058] Method 200 includes an act of describing the components of a composite distributed application in a declarative model, including describing a plurality of different application parts of the composite distributed application (act 201). For example, referring to Figure 1A, tools 125 can be used to describe components of composite distributed application 107 in declarative application model 153. As depicted, declarative application model 153 describes a plurality of application parts 161, including application parts 161A (a first application part) and 161B (a second application part). As previously described, application parts 161A and 161B can be different types of application parts, such as, for example, process and data.

[0059] Describing a composite distributed application can also include describing corresponding provider and consumer endpoints for the composite distributed application and provider endpoints as well as connections there between. For example, in the alternate view of declarative application model in Figures 1B and/or 1C, tools 125 can be used to describe external provider endpoint 171, provider endpoint 165A, consumer endpoint 165B, and provider endpoint 166A as well as the connection between consumer endpoint 165B and provider endpoint 166A.

[0060] Describing a composite application can also include setting implementation attributes on abstractions to expressly indicate the technologies that are to be used to implement application parts. For example, tools 125 can be used to set attributes 163A and/or 163B on application parts 161A and 161B respectively. As previously described, attribute 163A can expressly indicate an implementation

environment for application part 161A and attribute 163B can expressly indicate an implementation environment for application part 161B.

[0061] Method 200 includes an act of inserting one or more abstractions into the declarative model, each abstraction describing the intended behavior of a corresponding application part in an implementation independent manner, the abstractions extending the declarative model to make the declarative model executable (act 202). For example, tools 125 can be used to insert abstractions 162A and 162B into declarative application model 153. Abstractions 162A and 162B described the intended behavior of application parts 161A and 161B respectively in an implementation independent manner. Accordingly, abstractions 162A and 162B extend declarative application model 153 to make declarative application model 153 executable.

[0062] Method 200 includes an act of tagging each of the one or more abstractions with a label, each label including implementation specific details that indicate how the plurality of different technologies are to be used to implement the corresponding application part (act 203). For example, executive module 115 can tag application part 161AD (a refined application part 161A) and application part 161BD (a refined application part 161B) with labels 164A and 164B respectively. Labels 164A and 164B include implementation specific details that indicate how technologies, such as, for example, Axis-Tomcat, WF/WCF-WAS, ASMX-IIS, SQL, etc., are to be used to implement corresponding application parts 161A and 161B respectively.

[0063] Executive module 115 can determine how to tag an application part in a variety of different ways. When an abstraction includes an implementation attribute, executive module 115 can tag the abstraction in accordance with the express

indication in the implementation attribute. For example, when abstraction 162A includes attribute 163A (e.g., expressly indicating WF/WCF-WAS implementation), executive module 155 can configure label 164A in accordance with implementation details of attribute 163A (e.g., indicating that abstraction 162A is to be implemented using WF/WCF-WAS in WASHost). On the other hand, when an abstraction does not include an implementation attribute, a label can be inferred (e.g., based on available resources of different technologies, from prior system settings, from default values, etc.) and then tagged to the abstraction. For example, based on the level of available SQL resources compared to other types of database resources, executive module 115 can configure label 164B to indicate that abstraction 162B is to be implemented using SQL in MySQLServer host.

[0064] Method 200 includes an act of interpreting the declarative model to implement an executable instance of the composite distributed application, the executable instance of the composite distributed application implemented using the plurality of technologies in accordance with the implementation specific details included in the labels (act 204). For example, driver manager 130 can transfer the various application parts of detailed application model 153D to appropriate drivers in accordance with corresponding labels. Referring again to Figure 1E, driver manager can transfer application part 161A to WF/WCF-WAS driver 142B based on label 164A indicating WF/WCF-WAS as the host environment for abstraction 162A. WF/WCF-WAS driver 142B can translate intended behavior represented in abstraction 162A into actions 133A (a subset of actions 133). Performance of actions 113A result in executable application part 107A being configured and later executed in WF/WCF framework environment 136B within WAS Host 135B.

[0065] Driver manager 130 can similarly transfer application part 161B to an SQL driver based on label 164B indicating SQL as the host environment for abstraction 162B. The SQL driver can translate intended behavior represented in abstraction 162B into another subset of actions 133. Performance of the other subset of actions results in application part 107B (see Figure 1A) being configured and later executed in SQL environment within an SQL host. If appropriate, payload 107B can also be delivered to the SQL environment (e.g. static data).

[0066] Driver manager 130 can also interoperate with any utilized drivers to maintain provider and consumer endpoints to and from a resulting composite distributed application and to, from, and between various executable application parts of a composite distribute application.

[0067] In some embodiments, a declarative application model includes a plurality of interrelated model elements (e.g., within an application part, abstraction, etc.) that declare how a composite distributed application is to be configured and executed. Tools 125 can be used to create and modify declarative application models including model elements in accordance with the principles of the present invention. When a declarative application model is received, a driver manager can map included model elements to corresponding drivers and translators that can interpret them. From resulting interpretations, the driver manager can formulate an execution plan for actions to be performed on the host environment to configure and activate a distributed application.

[0068] Thus, a model interpreter can be configured to parse a received model and generate a corresponding execution plan. To generate an execution plan from a received model, the model interpreter can identify a model's constituent model elements and their relationships to one another. The model interpreter 243 can then

call correspond model element interpreters (translators) to retrieve a sequence of actions for each model element. The model interpreter then combines the sequence of actions for each model element into a single sequence of actions for inclusion in an execution plan.

[0069] An action can include a variety of different properties. For example, an item can be Idempotent. That is, the same action successfully executed again on the same model element will result in the same output. An action can have a unique identifier that identifies an operation. If two actions are to do the same thing, the two actions can have same identifiers. For example, two activities that install a file can have the same identifier when the file has the same absolute path.

[0070] An action can also be associated with a corresponding reverse action. The reverse action can reverses the effects of the action. For example, the reverse of a CopyFile would be a procedure that removes a file or restores an original file that was copied over. An action can be associated with a corresponding update action that applies a newer version of a resource over an existing resource.

[0071] Figure 3 illustrates an example computer architecture 300 for translating declarative application models. As depicted, computer architecture 300 includes driver manager 330. Driver manager 330 can be configured similarly to (or even the same as) driver manger 130 depicted in computer architecture 100. Driver manager 330 hosts drivers and translators and receives inputs. For example, driver manager 300 hosts a plurality of drivers, such as, for example, wcf-IIS driver 342 and Aspx-IIS driver 352.

[0072] Each driver can include a model interpreter (e.g., model interpreters 343 and 353 respectively) configured to parse a model and generate an execution plan. Thus, upon receiving a model, each driver can identify constituent resources and their

relationships to one another, call appropriate resource interpreters to retrieve sequences of actions, and assemble sequences of actions into an execution plan.

[0073] Driver manager 300 also hosts a plurality of translators, such as, for example, operating system technology translator 361, IIS technology translator 371, and WCF technology translator 381. Each translator can include a resource interpreter (e.g., resource interpreters 362, 372, and 382 respectively) for parsing model elements describing a resource and assembling a sequence of actions that can be executed on what the resource represents. Each translator can correspond to a portion of the environment. For example, operating system technology translator 361 corresponds to operating system 391, IIS technology translator 371 corresponds to IIS 392, and WCF technology transfer 381 corresponds to WCF 393.

[0074] Accordingly, when a translator receives a model element describing a resource, the resource represents some object within the corresponding portion of the environment. For example, when operating system technology translator 361 receives a resource the resource corresponds to an object in operating system 391 (e.g., a service.svc file). Similarly, when IIS technology translator 371 receives a resource the resource corresponds to an object in IIS 392 (e.g., the location of a user's Web service – "/myservice"). Likewise, when WCF technology translator 381 the resource corresponds to an object in WCF 393 (e.g., a Web.config).

[0075] Generally, dispatcher 371 is configured to receive a model and dispatch the model to the appropriate driver. Thus, upon receiving a model, dispatcher 371 can forward the model to type locator 341. Type locator 341 can receive the model and based on the model locate the appropriate type of driver for processing the model (e.g., based on a corresponding label, such as, for example, label 164A or label 164B). Type locator 341 can return a driver ID identifying the appropriate type of drive back

to dispatcher 371. Dispatcher 371 can then use the driver ID to dispatch the model to the identified appropriate driver.

[0076] Drivers can also utilize type locator 341 to locate appropriate translators for model element describing resources included in a model. For each resource in a model, a driver can submit the resource to type locator 341. Type locator 341 can receive the resource and based on the resource locate the appropriate type of translator for translating the resource. Type locator 341 can return a translator ID identifying the appropriate type of translator back to the driver. The driver can then use the translator ID to dispatch the resource to the identified appropriate translator

[0077] The appropriate translator parses the received resource to assemble a sequence of actions for a received command and returns the sequence of actions back to the driver. The driver then assembles the different sequences of actions into a single execution plan. The driver then executes the execution plan. Actions in the execution plan are executed in sequence, operating directly on the environment (e.g., on one or more of operating system 391, IIS 391, and WCF 392), to realize the intent collectively represented in model 353 and command 329.

[0078] Thus, for example, dispatcher 371 can receive detailed application model 353D, including a plurality of interrelated model elements declaring how to configure and execute an application. Dispatcher 371 can send detailed application model 353D to type locator 341. Type locator 341 can detailed application model 353D. Type locator 341 can process detailed application model 353D to determine the type of driver appropriate for processing detailed application model 353D. In response to the determination, type locator 341 can send driver ID 384 (an identifier for driver 342) to dispatcher 371.

[0079] Dispatcher 371 can utilize driver ID 384 to forward detailed application model 353D to Wcf-IIS driver 342. Model interpreter 343 can parse detailed application model 353D to identify model elements within detailed application model 353D and the model elements' relationships to one another. For example, model interpreter 343 can identify model element 354A, 354B, 354C, etc. (e.g., within application parts, abstractions, etc). Wcf-IIS driver 342 can send identified model elements, such as, for example, model element 354A, to type locator 341. Type locator 341 can receive model element 354A. Type locator can process model element 354A to determine the type of translator appropriate for translating model element 354A.

[0080] Determining the type of translator can be based on a specified technology (e.g., one of an operating system, network protocol, data type, etc.) indicated in and/or related to model element 354A. For example, type locator 341 can determine that operating system technology translator 361 is the appropriate driver for translating model element 354A based on model element 354A indicating or being related to operating system 391. In response to the determination, type locator 341 can send translator ID 355 (an identifier for translator 361) to Wcf-IIS driver 342. Similar determinations can be made for model elements 354B, 354C, etc. included in model 353.

[0081] Wcf-IIS driver 342 can utilize translator ID 355 to forward model element 354A to operating system technology translator 361. Resource interpreter 362 can translate model element 354A into action sequence 363. Action sequence 363 includes a plurality of actions, such as, for example, actions 363A, 363B. etc, that are to be executed in operating system 391 to implement the intent of detailed application model 353D.

[0082] Similarly, Wcf-IIS driver 342 can utilize an appropriate translator ID to forward model element 354B to IIS technology translator 371. Resource interpreter 372 can translate model element 354B into action sequence 373. Action sequence 373 includes a plurality of actions, such as, for example, actions 373A, 373B. etc, that are to be executed in IIS 392 to implement another portion of the intent of detailed application model 353D.

[0083] Likewise, Wcf-IIS driver 342 can utilize an appropriate translator ID to forward model element 354C to WCF technology translator 381. Resource interpreter 382 can translate model element 354C into action sequence 383. Action sequence 383 includes a plurality of actions, such as, for example, actions 383A, 383B. etc, that are to be executed in WCF 393 to implement further portion of the intent of detailed application model 353D.

[0084] Further model elements can also be sent to any of operating system technology translator 361, IIS technology translator 371, WCF technology translator 381, as well as to other appropriate technology translators (not shown), based on a model element indicating and/or being related to a specified technology.

[0085] Model interpreter 343 can receive action sequence 363 from operating system technology translator 361, action sequence 373 IIS technology translator 371, action sequence 383 WCF technology translator 381, etc. Each action sequence is a subset of the total actions that are to be performed to fully implement the intent of detailed application model 353D. Model interpreter 343 can assemble action sequences 363, 373, 383, etc. into execution plan 344. Action sequences 363, 373, 383, etc., are assembled in a designated order based on the relationship between corresponding model elements in detailed application model 353D. Wcf-IIS driver 342 can execute execution plan 344 to implement a composite distributed application

based on the intent for detailed application model 353D that is to use portions of operating system 391, IIS 392, and WCF 393, etc.

[0086] Accordingly, embodiments of the present invention facilitate modeling and managing heterogeneous applications. Application intent can be described in a relatively straight forward manner that abstracts underlying implementation details. Thus, application developers can develop applications without necessarily having to know extensive details of an underlying implementation environment. In any event, an application can be executed in different implementation environments without requiring changes to the corresponding model.

[0087] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

What is claimed:

1. At a computer system within a computing environment (100), the computing environment (100) including a plurality of different technologies (391,392,393) for implementing distributed applications, a method for implementing a composite distributed application (107), the method comprising:

an act of describing the components of a composite distributed application (107) in a declarative model (153), including describing a plurality of different application parts (107A, 107B) of the composite distributed application (107);

an act of inserting one or more abstractions (162A, 162B) into the declarative model, each abstraction describing the intended behavior of a corresponding application part (107A, 107B) in an implementation independent manner, the abstractions (162A, 162B) extending the declarative model (107) to make the declarative model (153) executable;

an act of tagging each of the one or more abstractions (162A, 162B) with a label (164A, 164B), each label (164A, 164B) including implementation specific details that indicate how the plurality of different technologies (391,392,393) are to be used to implement the corresponding application part (107A, 107B) ; and

an act of interpreting the declarative model (153) to implement an executable instance of the composite distributed application (107), the executable instance of the composite distributed application (107) implemented using the plurality of technologies (391,392,393) in accordance with the implementation specific details included in the labels (164A, 164B).

2. The method as recited in claim 1, further comprising:

an act of setting an implementation attribute on at least one inserted abstraction, the implementation attribute including implementation details expressly indicating how the abstraction is to be tagged to indicate an intended implementation.

3. The method as recited in claim 2, wherein the act of tagging each of the one or more abstractions with a label comprises an act of tagging the at least one inserted abstraction for implementation based on the implementation details in the implementation attribute.

4. The method as recited in claim 1, wherein the act of describing the components of a composite distributed application in a declarative model comprises an act of describing a process application part and a data application part of the composite distributed application.

5. The method as recited in claim 4, wherein the act of describing a process application part and a data application part of the composite distributed application comprises an act of describing a Web Services and Databases.

6. The method as recited in claim 1, wherein the act of describing the components of a composite distributed application in a declarative model comprises:

an act of describing endpoints connecting the plurality of different application parts to one another; and

an act of describing an external provider endpoint for communicating with the composite distributed application.

7. The method as recited in claim 1, wherein the act of tagging each of the one or more abstractions with a label comprises an act of inferring how an abstraction is to be tagged based on one or more of: available system resources of different technologies, prior system settings, and default system values.

8. The method as recited in claim 1, wherein the act of interpreting the declarative model to implement an executable instance of the composite distributed application comprises an act of implementing different application parts of the composite distributed application in different host environments.

9. The method as recited in claim 1, wherein the computer system further includes one or more drivers, each driver configured to process abstractions for a specified combination of different technologies, the computer system further including one or more translators, each translator configured to process model elements representing objects within a specified technology from among the plurality of different technologies, and wherein the act of interpreting the declarative model to implement an executable instance of the composite distributed application comprises:

for each abstraction:

an act of identifying a driver that is configured to process a described application part corresponding to the combination of technologies indicated for the abstraction;

an act of forwarding the application part to the identified driver; and

an act of the identified driver parsing the application part to identify model elements and their relationship to one another;

for each identified model element:

an act of identifying a translator configured to translate model elements for the specified technology corresponding to the identified model element;

an act of sending the model element to the translator;

an act of receiving a sequence of actions that are to be performed within the specified technology to partially implement the composite distributed application, the sequence of actions being a subset of the total actions that are to be performed to fully implement the composite distributed application; and

an act of assembling the received sequence of actions for the model element into an execution plan, the received sequence of actions assembled into the execution plan in a designated order with respect to sequences of actions received for other model elements based on the model element's relationship to other model elements in the received declarative model; and

an act of executing the execution plan to implement the composite distributed application, execution of the execution plan including executing the sequences of actions received for each model element in the designated order.

10. A computer program product for use at a computer system within a computing environment, the computing environment including a plurality of different technologies for implementing distributed applications, the computer program product for implement a method for implementing a composite distributed application, the computer program product comprising one or more physical storage media having stored thereon computer-executable instructions that, when executed at a processor, cause the computer system to perform the method including the following:

describe the components of a composite distributed application in a declarative model, including describing a plurality of different application parts of the composite distributed application;

insert one or more abstractions into the declarative model, each abstraction describing the intended behavior of a corresponding application part in an implementation independent manner, the abstractions extending the declarative model to make the declarative model executable;

tag each of the one or more abstractions with a label, each label including implementation specific details that indicate how the plurality of different technologies are to be used to implement the corresponding application part; and

interpret the declarative model to implement an executable instance of the composite distributed application, the executable instance of the composite distributed application implemented using the plurality of technologies in accordance with the implementation specific details included in the labels.

11. The computer program product as recited in claim 10, further comprising computer-executable instructions that, when executed, cause the computer

system to set an implementation attribute on at least one inserted abstraction, the implementation attribute including implementation details expressly indicating how the abstraction is to be tagged to indicate an intended implementation.

12. The computer program product as recited in claim 11, wherein computer-executable instructions that, when executed, cause the computer system to tag each of the one or more abstractions with a label comprises computer-executable instructions that, when executed, cause the computer system to tag the at least one inserted abstraction for implementation based on the implementation details in the implementation attribute.

13. The computer program product as recited in claim 10, wherein computer-executable instructions that, when executed, cause the computer system to describe the components of a composite distributed application in a declarative model comprises comprise computer-executable instructions that, when executed, cause the computer system to describe a process application part and a data application part of the composite distributed application.

14. The computer program product recited in claim 10, wherein computer-executable instructions that, when executed, cause the computer system to describe the components of a composite distributed application in a declarative model comprise computer-executable instructions that, when executed, cause the computer system to:

describe endpoints connecting the plurality of different application parts to one another; and

describe an external provider endpoint for communicating with the composite distributed application.

15. The computer program product as recited in claim 10, wherein computer-executable instructions that, when executed, cause the computer system to tag each of the one or more abstractions with a label comprises computer-executable instructions that, when executed, cause the computer system to infer how an abstraction is to be tagged based on one or more of: available system resources of different technologies, prior system settings, and default system values.

16. The computer program product as recited in claim 10, wherein computer-executable instructions that, when executed, cause the computer system to interpret the declarative model to implement an executable instance of the composite distributed application comprises computer-executable instructions that, when executed, cause the computer system to implement different application parts of the composite distributed application in different host environments.

17. At a computer system, a method for describing the intended executable behavior of a composite distributed application (107) in an implementation-independent manner, the method comprising:

an act of describing a composite distributed application (107) in a declarative model (153), the declarative model having a plurality of model elements, each model element representing a portion of the intended executable behavior of the composite

distributed application (107), description of the composite distributed application including at least:

an act of describing a first application part (107A) of the composite distributed application (107), the first application part being a first specified type of application part, including:

an act of declaring a first abstraction (162A) for the first application part (107A), the first abstraction (162A) describing the intended execution behavior of the first application part (107A) ; and

an act of describing a second part (107B) of the composite distributed application (107), the second part (107B) being a second different specified type of application part, including:

an act of declaring a second abstraction (162B) for the second application part (107B), the second abstraction (162B) describing the intended execution behavior of the second application part (107B).

18. The method as recited in claim 17, wherein the act of describing a composite distributed application in a declarative model comprises an act of describing a composite distributed application, wherein the first application part is process and the second application part is data.

19. The method as recited in claim 17, wherein the act of describing a first application part of the composite distributed application comprises an act of setting a first implementation attribute on the first abstraction to expressly indicate the technologies that are to be used to implement the first application part.

20. The method as recited in claim 17, further comprising:

an act of describing an external provider endpoint for the composite distributed application , the external provider endpoint configured to:

receive requests from components external to the composite distributed application; and

delegate the requests to the first part provider endpoint for processing.

21. The method as recited in claim 17, wherein the act of describing a first application part of the composite distributed application further includes:

an act of declaring a first application part provider endpoint that provides information processed at the first application part to external components;

an act of declaring a first part consumer endpoint that consumes information from external components; and

wherein an act of describing a second part of the composite distributed application further includes:

an act of declaring a second application part provider endpoint that provides information to external components, the second application part provider endpoint connectable to consumer endpoints including the first application part consumer endpoint; and

an act of declaring a second application part consumer endpoint that consumes information from external components, the second application part consumer endpoint connectable to provider endpoints.

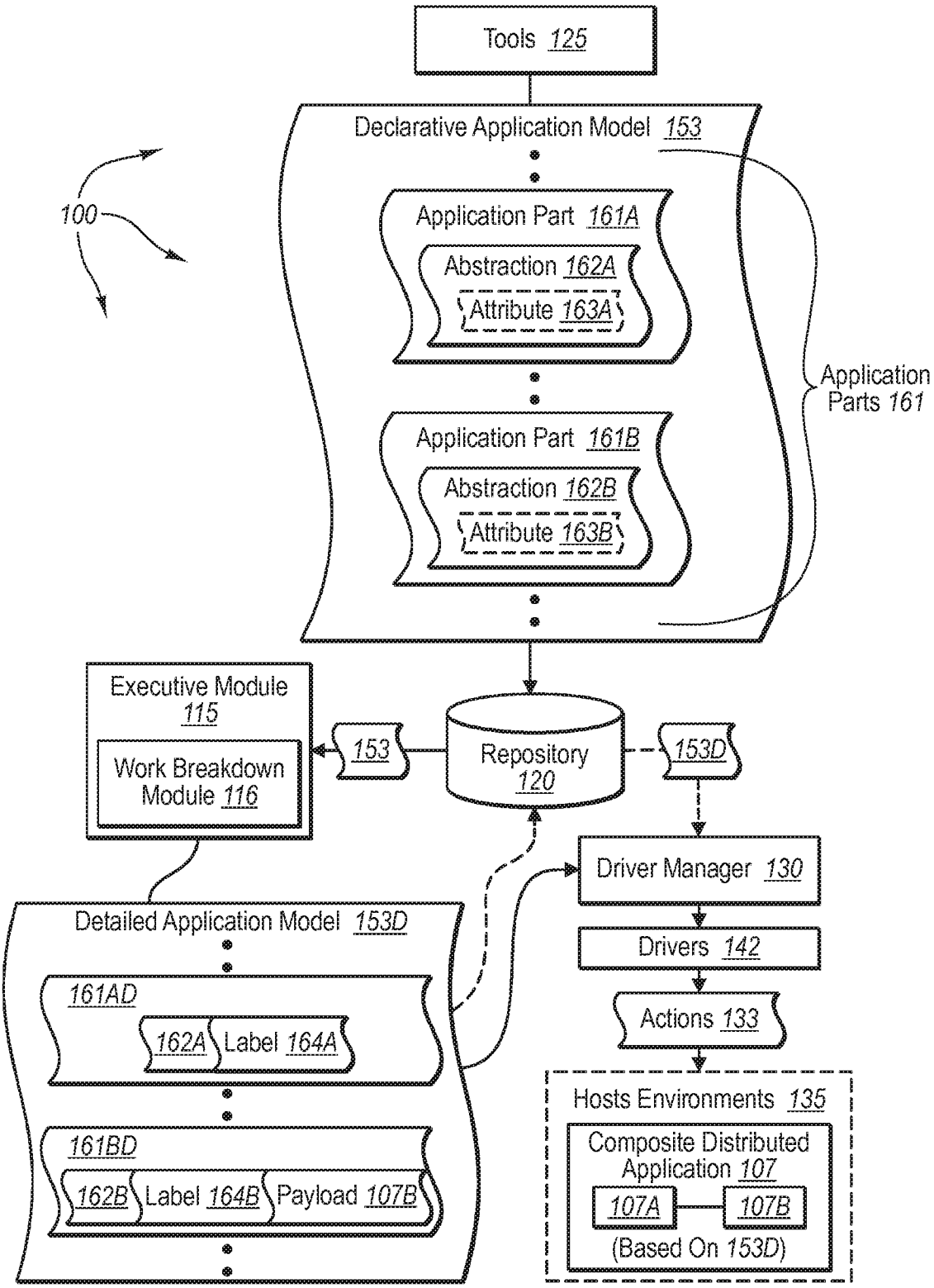
Tools 125
Declarative Application Model 153
100
Application Part 161A
Abstraction 162A
Attribute 163A
Application Part 161B
Abstraction 162B
Attribute 163B
Application Parts 161
Executive Module 115
Work Breakdown Module 116
153
Repository 120
153D
Driver Manager 130
Drivers 142
Actions 133
Hosts Environments 135
Composite Distributed Application 107
107A
107B
(Based On 153D)
Detailed Application Model 153D
161AD
162A
Label 164A
161BD
162B
Label 164B
Payload 107B

FIG. 1A

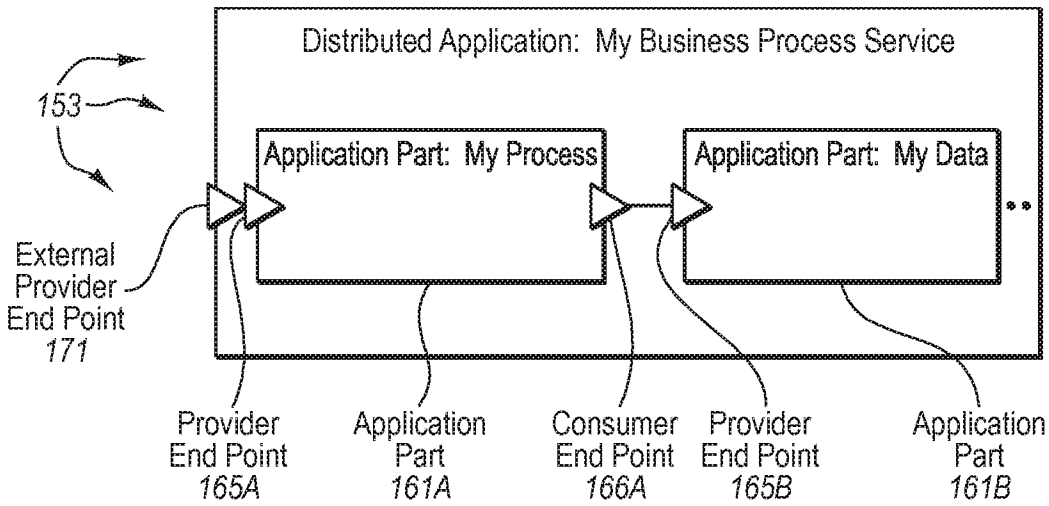
Distributed Application: My Business Process Service
153
Application Part: My Process
Application Part: My Data
External Provider End Point 171
Provider End Point 165A
Application Part 161A
Consumer End Point 166A
Provider End Point 165B
Application Part 161B

FIG. 1B

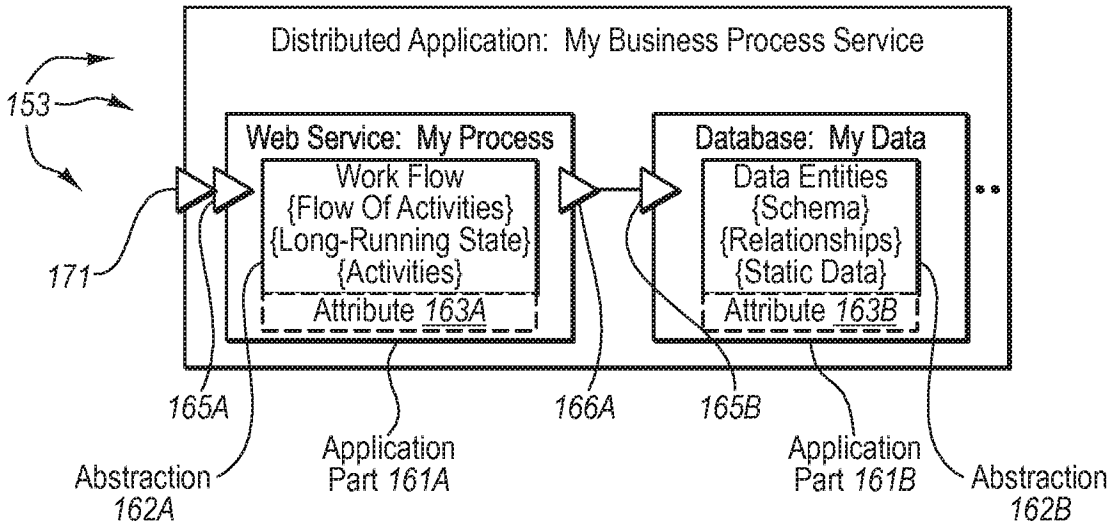
Distributed Application: My Business Process Service
153
Web Service: My Process
Work Flow
{Flow Of Activities}
{Long-Running State}
{Activities}
Attribute 163A
Database: My Data
Data Entities
{Schema}
{Relationships}
{Static Data}
Attribute 163B
171
165A
166A
165B
Abstraction 162A
Application Part 161A
Application Part 161B
Abstraction 162B

FIG. 1C

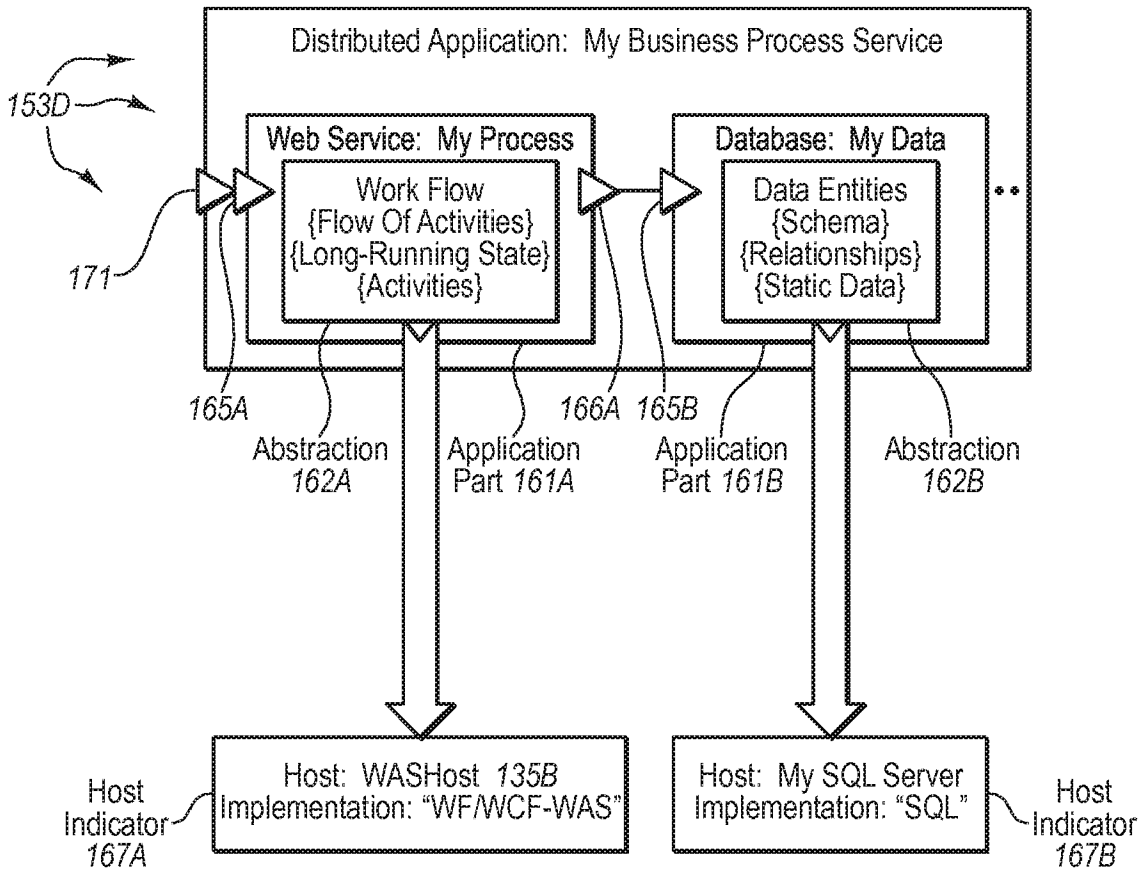
Distributed Application: My Business Process Service
153D
171
Web Service: My Process
Work Flow
{Flow Of Activities}
{Long-Running State}
{Activities}
Database: My Data
Data Entities
{Schema}
{Relationships}
{Static Data}
165A
Abstraction
162A
Application
Part 161A
166A
165B
Application
Part 161B
Abstraction
162B
Host
Indicator
167A
Host: WASHost 135B
Implementation: "WF/WCF-WAS"
Host: My SQL Server
Implementation: "SQL"
Host
Indicator
167B

FIG. 1D

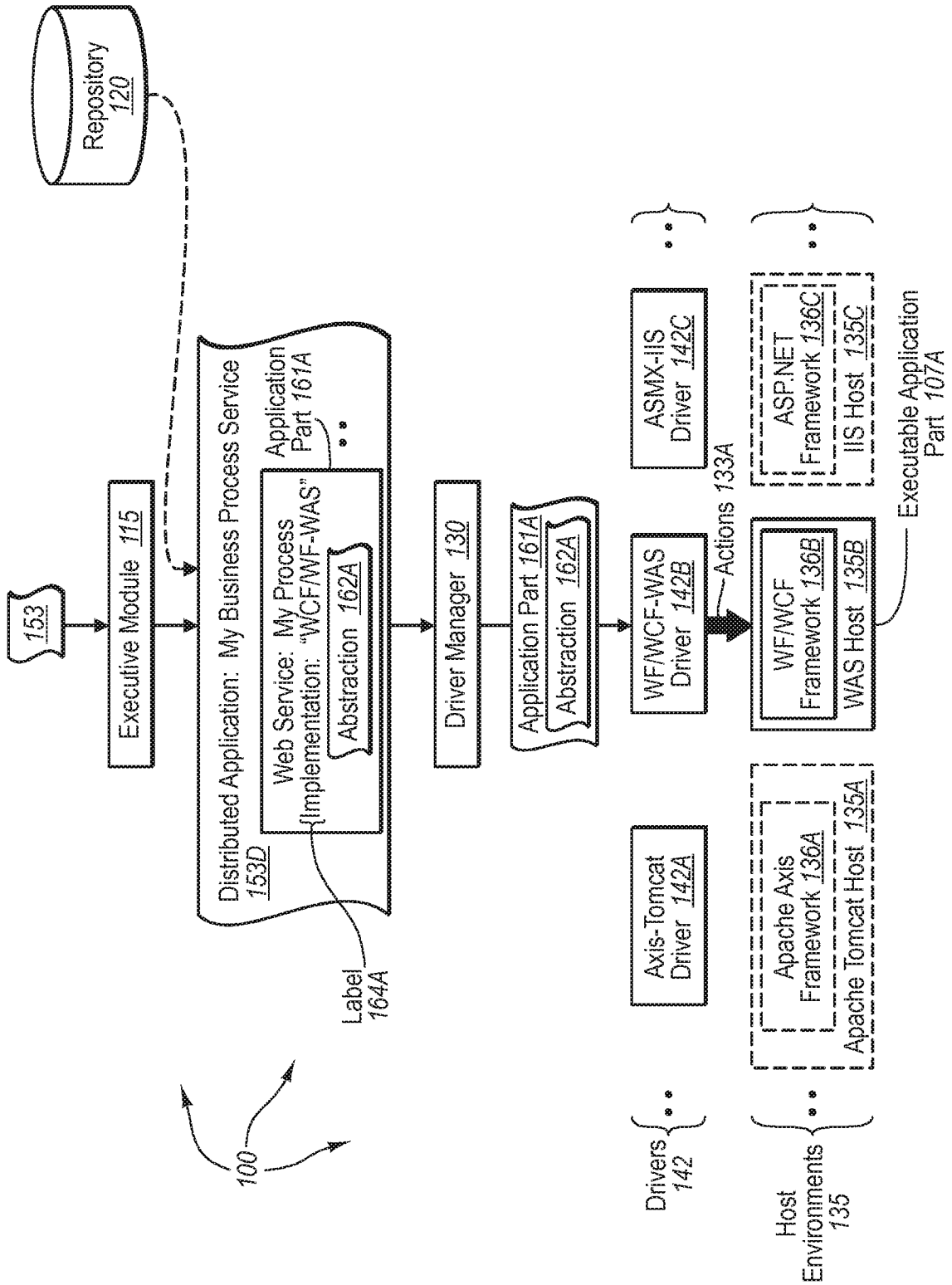
Repository 120
153
Executive Module 115
Distributed Application: My Business Process Service 153D
Web Service: My Process Implementation: "WCF/WF-WAS"
Application Part 161A
Abstraction 162A
Label 164A
Driver Manager 130
Application Part 161A
Abstraction 162A
100
Drivers 142
Axis-Tomcat Driver 142A
WF/WCF-WAS Driver 142B
ASMX-IIS Driver 142C
Actions 133A
Host Environments 135
Apache Axis Framework 136A
Apache Tomcat Host 135A
WF/WCF Framework 136B
WAS Host 135B
ASP.NET Framework 136C
IIS Host 135C
Executable Application Part 107A

FIG. 1E

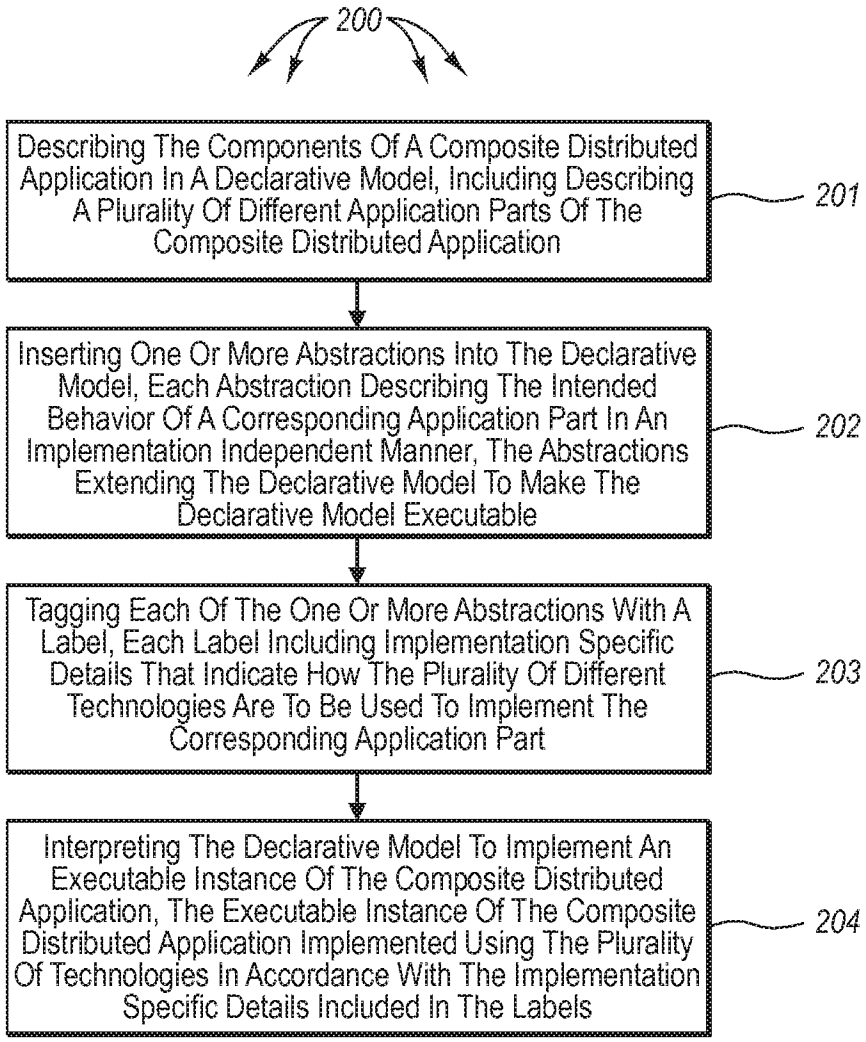
200
Describing The Components Of A Composite Distributed Application In A Declarative Model, Including Describing A Plurality Of Different Application Parts Of The Composite Distributed Application
201
Inserting One Or More Abstractions Into The Declarative Model, Each Abstraction Describing The Intended Behavior Of A Corresponding Application Part In An Implementation Independent Manner, The Abstractions Extending The Declarative Model To Make The Declarative Model Executable
202
Tagging Each Of The One Or More Abstractions With A Label, Each Label Including Implementation Specific Details That Indicate How The Plurality Of Different Technologies Are To Be Used To Implement The Corresponding Application Part
203
Interpreting The Declarative Model To Implement An Executable Instance Of The Composite Distributed Application, The Executable Instance Of The Composite Distributed Application Implemented Using The Plurality Of Technologies In Accordance With The Implementation Specific Details Included In The Labels
204

FIG. 2

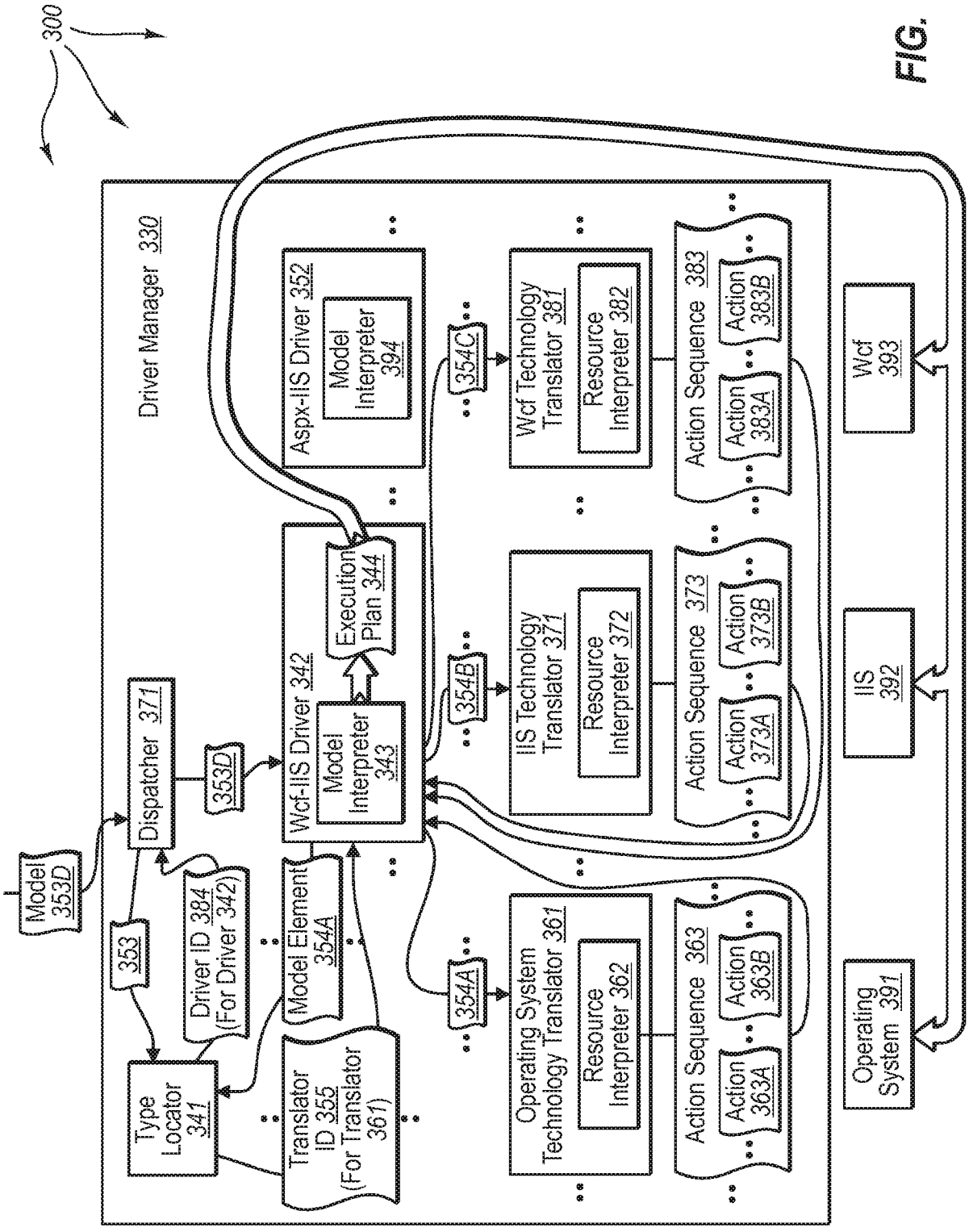
300
Driver Manager 330
Model 353D
353
Dispatcher 371
353D
Type Locator 341
Driver ID 384 (For Driver 342)
Translator ID 355 (For Translator 361)
Model Element 354A
Wcf-IIS Driver 342
Model Interpreter 343
Execution Plan 344
354A
354B
354C
Operating System Technology Translator 361
Resource Interpreter 362
Action Sequence 363
Action 363A
Action 363B
IIS Technology Translator 371
Resource Interpreter 372
Action Sequence 373
Action 373A
Action 373B
Aspx-IIS Driver 352
Model Interpreter 394
Wcf Technology Translator 381
Resource Interpreter 382
Action Sequence 383
Action 383A
Action 383B
Operating System 391
IIS 392
Wcf 393

FIG. 3