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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 July 2010 (29.07.2010)

PCT

(10) International Publication Number
WO 2010/084126 A1

(51) International Patent Classification:
G06F 9/46 (2006.01)

(21) International Application Number:
PCT/EP2010/050628

(22) International Filing Date:
20 January 2010 (20.01.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09151359.8 26 January 2009 (26.01.2009) EP

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **BOHM, Fraser, Paul** [GB/GB]; IBM United Kingdom Limited, MP 208, Hursley Park, Hursley, Winchester Hampshire SO21 2JN (GB).

(74) Agent: **STRETTON, Peter, John**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

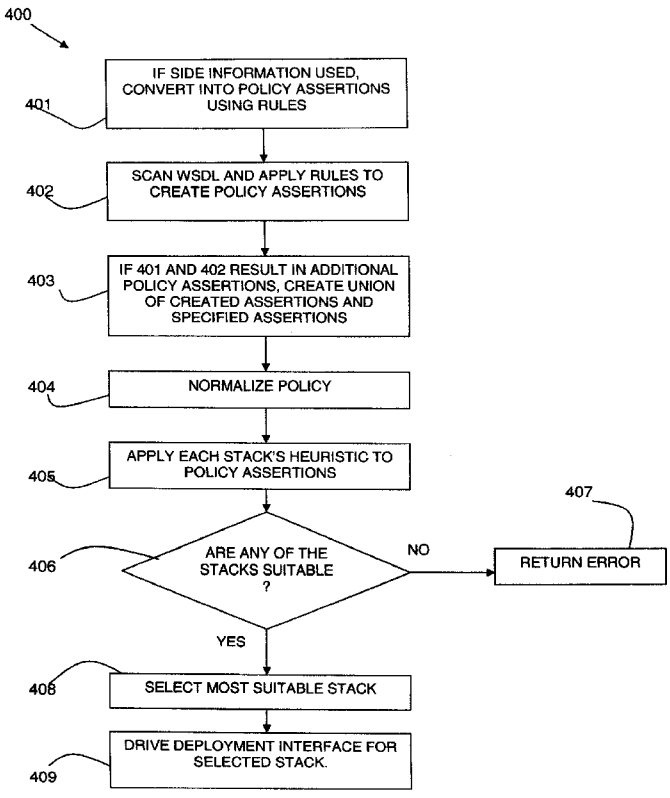
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, 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, PE, PG, PH, PL, PT, RO, RS, RU, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR SELECTION OF A RUNTIME STACK FOR DEPLOYMENT OF A WEB SERVICE

FIG. 4



(57) Abstract: A method and system for selection of a runtime stack (211, 212, 241-243) for deployment of a Web Service (220) are provided. The method includes: generating policy assertions (262A, 262B) for a Web Service (220) to be deployed; providing a scoring mechanism (215, 216, 271 -273) for each available runtime stack (211, 212, 241 -243) in which the ability of a stack (211, 212, 241-243) to support each of a plurality of policy assertions is scored; applying the scoring mechanism (215, 216, 271-273) for each available runtime stack (211, 212, 241-243) to the policy assertions (262 A, 262B) for the Web Service (220) to be deployed; and selecting a stack (211, 212, 241-243) based on the results of applying the scoring mechanism (215, 216, 271-273). The policy assertions (262A, 262B) for a Web Service (220) to be deployed can include a combination of specification defined Web Service policy assertions (222), WSDL elements (221) of the Web Service which are mapped (282) to policy assertions, and side information (223) requirements of the Web Service which are mapped (281) to policy assertions.

WO 2010/084126 A1

TM), European (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, 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))*

METHOD AND SYSTEM FOR SELECTION OF A RUNTIME STACK FOR DEPLOYMENT OF A WEB SERVICE

This invention relates to the field of configuration of Web Services. In particular, the invention relates to selection of a runtime stack for deployment of a Web Service.

In situations where a runtime system (Web Services endpoint) has available multiple Web Services stacks, the deployer has to choose the target deployment stack. The stacks will have differing functional and non-functional capabilities.

The deployer needs to understand both the relative functional and non-functional capabilities of the Web Services stacks and the tradeoffs between them. Individual Web Services may exploit or require different subsets of these capabilities and therefore the optimal stack for deployment of one Web Service may not be the optimal stack for another, again the deployer must choose and configure the appropriate stack. This choice requires significant knowledge of the stacks' characteristics as well as of the individual Web Service's requirements and is thus a time consuming and error prone process.

WS-Policy (www.w3.org/Submission/WS-Policy/) provides a standard mechanism for a Web Services provider to describe these functional and non-functional requirements and capabilities. The WS-Policy includes the following paragraphs.

"WS-Policy provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML (eXtensible Markup Language) Web services-based system. WS-Policy defines a framework and a model for the expression of these properties as policies.

WS-Policy defines a policy to be a collection of policy alternatives, where each policy alternative is a collection of policy assertions. Some policy assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation yet are critical to proper service selection and usage (e.g., privacy policy, QoS (quality of

service) characteristics). WS-Policy provides a single policy grammar to allow both kinds of assertions to be reasoned about in a consistent manner."

"Applied in the Web Services model, policy is used to convey conditions on an interaction between two Web Service endpoints. Satisfying assertions in the policy usually results in behaviour that reflects these conditions. Typically, the provider of a Web service exposes a policy to convey conditions under which it provides the service. A requester might use this policy to decide whether or not to use the service. A requester may choose any alternative since each is a valid configuration for interaction with the service, but a requester MUST choose only a single alternative for an interaction with a service since each represents an alternative configuration.

A policy assertion is supported by a requester if and only if the requester satisfies the requirement (or accommodates the capability) corresponding to the assertion. A policy alternative is supported by a requester if and only if the requester supports all the assertions in the alternative. And, a policy is supported by a requester if and only if the requester supports at least one of the alternatives in the policy. Note that although policy alternatives are meant to be mutually exclusive, it cannot be decided in general whether or not more than one alternative can be supported at the same time."

In general WS-Policy is used by a provider to communicate these requirements to a requester and sometimes by a requester to validate that a given provider meets some level of required behaviour.

According to a first aspect of the present invention there is provided a method for selection of a runtime stack for deployment of a Web Service, comprising: generating policy assertions for a Web Service to be deployed; providing a scoring mechanism for each available runtime stack in which the ability of a stack to support each of a plurality of policy assertions is scored; applying the scoring mechanism for each available runtime stack to the policy assertions for the Web Service to be deployed; and selecting a stack based on the results of applying the scoring mechanism.

Generating policy assertions for a Web Service to be deployed may include one or more of: using specification defined Web Service policy assertions; scanning a service description in WSDL (Web Services Description Language) for WSDL elements of the Web Service and mapping the WSDL elements to policy assertions; and/or converting side information requirements of the Web Service and mapping the requirements to policy assertions.

The side information may be provided by a provider or a requester of the Web Service to be deployed and the side information may relate to service requirements of the provider or requester.

Generating policy assertions for a Web Service to be deployed may include normalising a policy to include one or more alternative sets of policy assertions, and wherein applying the scoring mechanism may be applied to each set of policy assertions.

If the selection of runtime stack is for a Web Service provider, the results of the scoring mechanism may be used to select a stack which supports all the sets of policy assertions. If the selection of runtime stack is for a Web Service requester, the results of the scoring mechanism may be used to select a stack which best supports one of the alternative sets of policy assertions.

The scoring mechanism may give a positive value if a policy assertion is supported and a negative value if a policy assertion is not supported by a stack. The values may be combined for a set of policy assertions required for a Web Service.

A runtime stack may handle the Web Services protocol and transport processing required for a request or response.

According to a second aspect of the present invention there is provided a computer program product stored on a computer readable storage medium for selection of a runtime stack for deployment of a Web Service, comprising computer readable program code means for performing the steps of: generating policy assertions for a Web Service to be deployed; providing a scoring mechanism for each available runtime stack in which the ability of a

stack to support each of a plurality of policy assertions is scored; applying the scoring mechanism for each available runtime stack to the policy assertions for the Web Service to be deployed; and selecting a stack based on the results of applying the scoring mechanism.

According to a third aspect of the present invention there is provided a method of providing a service to a customer over a network, the service comprising: generating policy assertions for a Web Service to be deployed; providing a scoring mechanism for each available runtime stack in which the ability of a stack to support each of a plurality of policy assertions is scored; applying the scoring mechanism for each available runtime stack to the policy assertions for the Web Service to be deployed; and selecting a stack based on the results of applying the scoring mechanism.

According to a fourth aspect of the present invention there is provided a system for selection of a runtime stack for deployment of a Web Service, comprising: a runtime container including a processor and a plurality of stacks to which a Web Service could be deployed; a generator for generating policy assertions for a Web Service to be deployed; a scoring mechanism for each available runtime stack in which the ability of a stack to support each of a plurality of policy assertions is scored; applying the scoring mechanism for each available runtime stack to the policy assertions for the Web Service to be deployed; and a selector for selecting a stack based on the results of applying the scoring mechanism.

The generator may include one or more of: listing specification defined Web Service policy assertions; a mechanism for scanning a service description in WSDL for WSDL elements of the Web Service and rules mapping the WSDL elements to policy assertions; and a converter of side information requirements of the Web Service and rules mapping the requirements to policy assertions. The generator may include a normalizer for normalising a policy to include one or more alternative sets of policy assertions, and wherein applying the scoring mechanism is applied to each set of policy assertions.

If the runtime container is a provider of a Web Service, the results of the scoring mechanism may be used to select a stack which supports all the sets of policy assertions. If the runtime

container is a requester of a Web Service, the results of the scoring mechanism may be used to select a stack which best supports one of the alternative sets of policy assertions.

A runtime system which contains multiple Web Services stacks can use a combination of WS-Policy information for a Web Service and a heuristic function for each stack to choose the optimal stack to deploy that Web Service to during deployment and to configure the instance of that stack appropriately.

The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanying drawings in which:

Figure 1 is a block diagram of a Web Service architecture in which the present invention may be implemented;

Figures 2A and 2B are block diagrams of Web Service architectures in accordance with aspects of the present invention;

Figure 3 is a block diagram of a computer system in which the present invention may be implemented; and

Figure 4 is a flow diagram of a method carried out at deployment of a Web Service in accordance with the present invention.

It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference numbers may be repeated among the figures to indicate corresponding or analogous features.

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, and components have not been described in detail so as not to obscure the present invention.

The described method and system enable a runtime system which contains multiple Web Services stacks to use a combination of WS-Policy information for a Web Service and an heuristic function for each stack to choose the optimal stack to deploy the Web Service to.

Web Services can implement a service-oriented architecture (SOA). Web Services make functional building blocks accessible over standard Internet protocols independent of platforms and programming languages. Each SOA building block can play one or both of two roles: as a service provider, or as a service requester.

Figure 1 shows an exemplary Web Service architecture 100. The Web Service architecture 100 includes a service provider 110 and a service requester 120 which communicate over a network 140. A service broker 130 is provided with a Web Services registry 132.

The service provider 110 creates a Web Service and possibly publishes its interface and access information to the Web Services registry 132. Each provider must decide which services to expose, how to make trade-offs between security and easy availability, how to price the services, or, if they are free, how to exploit them for other value. The provider also has to decide what category the service should be listed in for a given broker service and what sort of trading partner agreements are required to use the service. It registers what services are available within it, and lists all the potential service recipients.

Public service brokers 130 are available through the Internet, while private service brokers are only accessible to a limited audience, for example, users of a company intranet. The Universal Description Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web Services.

The service requester 120 or Web Service client locates entries in the Web Services registry 132 using various find operations and then binds to the service provider 110 in order to invoke one of its Web Services.

Referring to Figures 2A and 2B, a described Web Services system 200 is shown with a Web Service provider 210 providing a Web Service 220 to be deployed, and a Web Service requester 240.

Figure 2A shows a Web Service provider 210 which hosts one or more Web Services stacks 211, 212 and a Web Service 220. A stack 211, 212 handles the Web Service's protocol/transport processing required for a given request/response on behalf of the Web Service 220.

The Web Service 220 which may be provided at the provider 210 or at a registry, includes: a service description 221 in WSDL (Web Services Description Language); a set of WS-Policy assertions 222; and, optionally, side information 223 concerning non-policy implementation requirements, if needed. The side information 223 is only required if all the needs of the deployed Web Service 220 are not already expressed in the WS-Policy assertions 222 associated with the external view of the Web Service 220. The side information 223 would commonly include service level agreements (timeouts, throughputs etc) which are relevant to the provider 210.

A Web Service requester 240 is provided which may request a Web Service 220 from a provider 210 via a network 290. The Web Service requester 240 has a container which hosts one or more Web Services stacks which are shown in Figure 2B.

Examples of Web Services stacks and containers which may be used by the provider and requester include: Apache Axis2 (Apache Axis2 is a trade mark of Apache Software Foundation), .NET (.NET is a trade mark of Microsoft Corporation), BEA WebLogic (BEA WebLogic is a trade mark of Oracle Corporation), GlassFish (GlassFish is a trade mark of Sun Microsystems, Inc.), WebSphere (WebSphere is a trade mark of International Business Machines Corporation), CICS TS (CICS is a trade mark of International Business Machines

Corporation), JBoss (JBoss is a trade mark of Red Hat Inc.), etc. Many runtimes only support one stack in which case the terms container/stack are synonymous; however, some containers host multiple stacks.

In the described system, a deployment agent 260A, 260B is provided which is asked by a user to deploy the Web Service 220. A deployment agent 260A, 260B may be used by the provider 210 as shown in Figure 2A or by the requester 240 as shown in Figure 2B. The deployment agent may be provided as a runtime component of the container of the provider 210 or requester 240 or as a separate entity in direct communication with the requester or in communication via a network.

Referring to Figure 2A, the deployment agent 260A includes a generator 261A of a set of policy assertions 262A for the Web Service 220 to be deployed. The set of policy assertions 262A is a combined list generated from the service description 221, the WS-Policy 222, and the side information 223 of the Web Service 220.

The generator 261A includes a converter 263A which uses a first set of rules 281 that maps the side information 223 to policy assertions 262A (possibly empty), and a second set of rules 282 that maps the WSDL elements of the service description 221 to policy assertions 262A (possibly empty). The sets of rules 281, 282 can be provided locally to a deployment agent 260A, a Web Service provider 210, or may be made available from another source via a network. The WS-Policy 222 for the Web Service 220 has specified policy assertions which can be added directly to the set of policy assertions 262A.

The generator 261A also includes a policy normalizer 264A which normalizes the policies to a required form of the set of policy assertions 262A. This results in a set of policy assertions 262A which is of the form a list of possible policy sets, including policy option 1 or option 2 or option 3 etc. and each option is a conjunction of assertions i.e. assertion 1 & assertion 2 & assertion 3.

The one or more Web Services stacks 211, 212 each has a deployment interface 213, 214 which is understood by the deployment agent 260A. Each of the Web Services stacks 211,

212 in the container has a heuristic function 215, 216 or scoring mechanism which scores the stack's suitability for policy assertions.

The deployment agent 260A includes a stack selector 265A which applies the heuristic functions 268A of the available stacks 211, 212 to the set of policy assertions 262A for the Web Service 220 to be deployed.

The deployment agent 260A also includes a deployment interface driver 266A for driving a selected stack's deployment interface 213, 214 and an error return 267A if no stack is selected.

Referring to Figure 2B, a Web Service requester 240 is shown in more detail which hosts one or more Web Services stacks 241-243 on which a Web Service 220 from a provider 210 may be deployed. The requester 240 also hosts an application 244 for executing the Web Service 220. The application 244 may include side information 245 defining internal policy of the application. A stack 241-243 handles the Web Service's protocol/transport processing required for a given request/response on behalf of another component in the container 240 such as the application 244.

The deployment agent 260B of the requester 210 is similar to that of the provider 240 described in Figure 2A. The deployment agent 260B includes a generator 261B of a set of policy assertions 262B for the Web Service 220 to be deployed. The set of policy assertions 262B is a combined list generated from the service description 221, the WS-Policy 222, and the side information 223 of the Web Service 220. In addition, the generator 261B may use the side information 245 of the requester 210 in the policy assertions 262B.

The generator 261B includes a converter 263B which uses a first set of rules 281 that maps the side information 223, 245 to policy assertions 262A (possibly empty), and a second set of rules 282 that maps the WSDL elements of the service description 221 to policy assertions 262B (possibly empty). The sets of rules 281, 282 can be provided locally to a deployment agent, a Web Service requester, or may be made available from another source via a network. The WS-Policy 222 for the Web Service 220 has specified policy assertions which

can be added directly to the set of policy assertions 262B.

The generator 261B also includes a policy normalizer 264B which normalizes the policies to a required form of the set of policy assertions 262B.

The one or more Web Services stacks 241-243 each has a deployment interface 251-253 which is understood by the deployment agent 260B. Each of the Web Services stacks 241-243 in the container 240 has a heuristic function 271-273 which scores the stack's suitability for policy assertions.

The deployment agent 260B includes a stack selector 265B which applies the heuristic functions 268B of the available stacks 241-243 of the requester 240 to the set of policy assertions 262B for the Web Service 220 to be deployed.

The deployment agent 260B also includes a deployment interface driver 266B for driving a selected stack's deployment interface 251-253 and an error return 267B if no stack is selected.

When a Web Service 220 is deployed to a provider 210, the provider's deployment agent 260A chooses a stack 211, 212 in the provider container. Sometime later, when a requester 240 wants to interact with the Web Service 220, a service consumer is deployed into the requester container and the requester deployment agent 260B selects which of the requester container's stacks 241-243 to user. If another distinct requester also wants to use the Web Service 220, then the same process is repeated. Not all requesters and providers need use the described deployment agent to select the stack, as it is an internal process to each container.

Referring to Figure 3, an exemplary system for implementing the deployment agent, a Web Service provider, or a Web Service requester includes a data processing system 300 suitable for storing and/or executing program code including at least one processor 301 coupled directly or indirectly to memory elements through a bus system 303. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program

code in order to reduce the number of times code must be retrieved from bulk storage during execution.

The memory elements may include system memory 302 in the form of read only memory (ROM) 304 and random access memory (RAM) 305. A basic input/output system (BIOS) 306 may be stored in ROM 304. System software 307 may be stored in RAM 305 including operating system software 308. Software applications 310 may also be stored in RAM 305.

The system 300 may also include a primary storage means 311 such as a magnetic hard disk drive and secondary storage means 312 such as a magnetic disc drive and an optical disc drive. The drives and their associated computer-readable media provide non-volatile storage of computer-executable instructions, data structures, program modules and other data for the system 300. Software applications may be stored on the primary and secondary storage means 311, 312 as well as the system memory 302.

The computing system 300 may operate in a networked environment using logical connections to one or more remote computers via a network adapter 316.

Input/output devices 313 can be coupled to the system either directly or through intervening I/O controllers. A user may enter commands and information into the system 300 through input devices such as a keyboard, pointing device, or other input devices (for example, microphone, joy stick, game pad, satellite dish, scanner, or the like). Output devices may include speakers, printers, etc. A display device 314 is also connected to system bus 303 via an interface, such as video adapter 315.

Referring to Figure 4, a flow diagram 400 of a method carried out by a described deployment agent is shown.

If side information is used, convert 401 the side information into a set of WS-policy assertions using rules. Scan the WSDL and apply rules to create 402 another set of policy assertions. If steps 401 or 402 resulted in any additional assertions, create 403 an effective policy as the union of the created assertions and the specified ones.

The policy as defined in the WS-Policy specification is normalized 404 to result in a policy which is of the form a list of possible policy sets. So the policy option 1 or option 2 or option 3 etc. and each option is a conjunction of assertions i.e. assertion 1 & assertion 2 & assertion 3.

Each stack's heuristic is applied 405 to the resulting set of policy assertions.

It is determined 406 if none of the stacks are suitable, and if this is the case, an error is returned 407. If there are suitable stacks, the most suitable stack is selected 408 and the deployment interface for that stack is driven 409 passing the information from the normalized policy.

A provider's stacks must support all choices of policy sets in a policy. However, a requester could choose a policy set and therefore does not need the selected stack to support all policy sets.

Assertions

The policy assertions that may be defined are open ended, in that WS-Policy defines a framework and individual other specifications define assertions for their specific use. The following are some examples but not a definitive list.

Some assertions are straightforward and are listed below with exerts from the relevant specifications. These specifications simply define a single assertion each which says the capability defined in that specification must be used. They can have the optional attribute that says may be used, rather than must. More complex assertions include WS-SecurityPolicy assertions.

WS-RM (Web Service policy assertion for Reliable Messaging)

The normative outline for the RM assertion is:
<wsrmp:RMAssertion [wsp:Optional="true"]? ... >

...
</wsrmp:RMAssertion>
The following describes additional, normative constraints on the outline listed above:
/wsrmp:RMAssertion □
A policy assertion that specifies that WS-ReliableMessaging protocol MUST be used when sending messages. □□

WS-AT (Web Service policy assertion for Atomic Processing)

The normative outline for the Atomic Transaction policy assertion is: □
<wsat:ATAssertion [wsp:Optional="true"]? ... >
...
□</wsat:ATAssertion>
The following describes additional, normative constraints on the outline listed above:
/wsat:ATAssertion

A policy assertion that specifies that an Atomic Transaction coordination context MUST be flowed inside a requester's message. From the perspective of the requester, the target service that processes the transaction MUST behave as if it had participated in the transaction. For application messages that use a SOAP binding, the Atomic Transaction coordination context MUST flow as a SOAP header in the message.

MTOM/XOP □(Web Services policy assertion for SOAP Message Transmission Optimization Mechanism/XML-binary Optimized Packaging)

The normative pseudo schema for the MTOM policy assertion is:
<wsoma:MTOM wsp:Optional? .../>
The following describes additional constraints on the pseudo schema listed above:
/wsoma:MTOM □

A policy assertion that specifies that MTOM MUST be used in messages sent to the Web Service. It also specifies that responses from the Web Service MUST be optimized using

MTOM, i.e. that the messages must be sent using the multipart/related; type=application/xop+xml mime type. □□

Heuristic Functions

The heuristic functions or scoring mechanisms of the stacks may be created in a number of different ways.

In one embodiment, the heuristic function for a given stack returns a positive value if the stack supports the input assertions set and a negative value/exception if not. A well designed heuristic function will give higher positive values to 'better' stacks for that set of assertions. This simple function assumes the assertions are distinct.

In another embodiment, a more complex method would be to create a value correlated on subsets of policy assertions from the input set or to do further detailed analysis of individual assertion parameters.

A provider defines the services and its external policy. If the provider defines the Web Service as supporting multiple options (policy choices) for communicating with it, a requester can choose any of those options and expect it to work. Thus, when choosing a stack for the provider to deploy to, it has to support all the options as the requester could choose to use any of them. In terms of the heuristic, the provider should not choose any stack that returns a negative value/error for any policy set in the normalized policy.

A requester retrieves the policy and service definition from the provider or a registry. It may also have its own side information. The requester only has to support one way of talking to the provider so it can choose a stack that returns a negative value/error for a policy set in the normalized policy as long as at least one of the policy sets does not return a negative value /error.

To summarise, a provider must support all choices so must select a stack which does so and a requester only needs to support one choice so can select the a stack that does not support

some options if it is 'best suited' to supporting one of the remaining ones. This is included in the heuristic functions which will then differ for a provider or a requester stack selection.

Simple Heuristic Embodiment

An example container has two stacks Stack A and Stack B; however more than two stacks can be used. In this example case, it is assumed that stack A is high function but high performance cost and B is the opposite. The heuristic creator has to decide which policy assertions are important and which are not.

1. For each assertion A(x) a non-negative (0..anything) value is given for each stack that supports that assertion.

The 'better' that stack supports that assertion, the higher the value given. 'Better' in this case is a qualitative decision on the part of the heuristic designer.

If the stack does not support the assertion at all, it is given a negative value.

2. All assertions not evaluated above have a default value of 0.

3. The heuristic takes as input the normalized policy from the deployment agent.

4. For each option in the policy, it evaluates the assertions in that option.

a. If any assertion (A(x) as defined in stage 1) results in an negative value then that value is returned.

b. Otherwise, return a value obtained by summing the values for all the assertions (A(x)) in that option (as defined in stage 1).

5. The heuristic then selects the highest value obtained from evaluating each option and returns this as a result of the heuristic.

Since in this simple example Stack A is strictly worse in terms of performance when compared to Stack B, the heuristic designer can simply say for Stack A that A(x) returns 0 for each supported assertions and -1 if unsupported. For Stack B, A(x) could return 1 for each supported assertion and -1 for each unsupported assertion. This would mean that, if all the assertions for a given Web Service are supported in Stack B, it will choose that over Stack A.

More complex Heuristic Embodiment

This example is also limited to two stacks, and it is assumed that this is for selection of a provider of the Web Services stack. (The heuristic would change if it was for the requester's stack selection). In this case, the set of assertions are expanded to include three specification defined assertions as described above. Two hypothetical assertions are created that can be derived from WSDL and another hypothetical assertion that can be derived from side information.

Specification Defined Assertions:
1.<wsrmp:RMAssertion/>
2.<wsat:ATAssertion/>
3.<wsoma:MTOM/>

Implied from WSDL assertions:
4.<imp:SOAP11/> <-! Service must use SOAP 1.1
5.<imp:SOAP12/> <-! Service must use SOAP 1.2

Side Information assertion:
6.<side:UsesDB/> <-! Service makes uses of a Data Base

The heuristic designer has to analyze the relative behaviour of the stacks in question with regard to support or preference for these capabilities.

To demonstrate a more complex heuristic some choices can be made depending on

combinations of capabilities rather than being distinct.

Stack A:

- Does not support SOAP 1.2
- Very fast for SOAP 1.1
- Slow Database Access
- Fast at RM (Reliable Messaging)
- Fast at MTOM
- OK at WS-AT (Atomic Transactions)
- Very Poor performance when RM & MTOM are deployed together.

Stack B:

- Supports SOAP 1.1 & 1.2
- OK performance for both SOAP 1.1 & SOAP 1.2
- Fast Database Access
- Poor at RM (Reliable Messaging)
- Good at MTOM
- OK at WS-AT (Atomic Transactions)
- WS-AT & MTOM do not work together at all.

To create the heuristic, the numerical function that can evaluate a single set of policy assertions is needed.

The policy assertions above are numbered as 1..6 so A1..A6 in the example algorithm below refers to those assertions. A(x) has a value of 1 if the assertion x place down the list is present in the evaluated policy choice, and 0 if not.

A possible heuristic for the above stacks could be:

For Stack A:

If A(5) = 1 then value = -1 ! No support for SOAP 1.2

otherwise
value = A(1) * 50 ! fast for WS-RM
+ A(2) * 25 ! Ok for WS-AT
+ A(3) * 50 ! fast for MTOM
+ A(4) * 100 ! Very Fast for SOAP 1.1
+ A(6) * 1 ! Slow Database Access
If A(1) & A(3) then value =
value - 100 ! deal with the fact that RM & MTOM together are not good.

For Stack B:

If A(4)=1 & A(3)=1 then value = -1 ! No support for MTOM & WS-AT at the same time
otherwise
value = A(1) * 1 ! poor for WS-RM
+ A(2) * 25 ! Ok for WS-AT
+ A(3) * 25 ! good for MTOM
+ A(4) * 25 ! Ok for SOAP 1.1
+ A(5) * 25 ! Ok for SOAP 1.1
+ A(6) * 100 ! Fast Database Access

Example of evaluating a normalized policy against the two heuristics.

Example Policy
<wsp:Policy>
<wsp:ExactlyOne>
<wsp:All>
<wsrmp:RMAssertion/>
<wsat:ATAssertion/>
<imp:SOAP11/>
<side:UsesDB/>
</wsp:All>
<wsp:All>

<wsrmp:RMAssertion/>
<wsat:ATAssertion/>
<imp:SOAP12/>
<side:UsesDB/>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

This policy describes a service that must use WS-RM and WS-AT, could use either SOAP 1.1 or 1.2 and makes use of a data base.

So as described in the simple heuristic, the function defined above is used to evaluate each individual set of policy assertions and give a final value of the lowest value for each set. If the stack selection is for the requester, a single stack can be selected and the highest value can be selected.

For stack A and choice one (<wsrmp:RMAssertion/>
<wsat:ATAssertion/><imp:SOAP11/><side:UsesDB/>), the result is:
A(1) = 1, A(2) = 1, A(3) = 0, A(4) =1, A(5) = 0, A(6) =1
running the function for stack A, gets 1*50 + 1*25 + 0*50 + 1*100 + 1*1 = 176.

For stack A and choice two (<wsrmp:RMAssertion/>
<wsat:ATAssertion/><imp:SOAP12/><side:UsesDB/>), running the function stack A, gets a value of -1 (as A(5) is true).

This gives us a final value of -1 for stack A, as that was the lowest value for the two choices.

Doing the same for stack B's heuristic gives us 151 for choice one and the same for choice two. So the final value if 151.

This would mean that the deployment agent would choose to deploy the service on stack B.

If the second choice is removed from the policy (i.e. saying that the service does not need to support SOAP 1.2), then the comparison would now favour stack A (176 > 151).

Worked Example

A working example is given in CICS Transaction Server (CICS is a trade mark of International Business Machines Corporation).

CICS TS intends to support an Apache Axis2 based Web Services stack alongside its existing native Web Services stack. At deployment time CICS will interrogate the WS-Policy information to build a list of the required capabilities for the Web Service.

For example:

- Web Service A uses WS-AT, WS-Security(UserNameToken, no encryption, no signature) and MTOM/XOP;
- Web Service B uses WS-Addressing and WS-Security(Encryption);
- Web Service C uses no WS-* capabilities.

An heuristic function is called for each stack which assigns a weighted value for each capability and produces a final suitability value for that stack. For example, the most obvious weighting is a large negative value if the stack does not support the capability.

Using the above example it can be know that using WS-Security(encryption) performs much better in the Axis2 stack than the native stack and thus weight the choice heavily in that direction for Web Service B, but that if all other things are equal (as in Web Service C) the native stack heuristic should return a higher value, as the performance is better.

The runtime then dynamically creates a runtime configuration for the stack that is selected by the system on the basis of the heuristic and associates it with that Web Service.

The advantage over the manual choice approach is that the extensive knowledge as to the relative stack capabilities and QoS can be encapsulated in the choice function (which uses

Policy and WSDL as an input) and frees the deployer from the burden of understanding these details and having to make the choice. This approach can be used alongside a manual choice approach.

The described deployment agent may be provided to a client as a service over a network.

The invention can take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment containing both hardware and software elements. In an embodiment, the invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

The invention can take the form of a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. For the purposes of this description, a computer usable or computer readable medium can be any apparatus that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus or device.

The medium can be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. Examples of a computer-readable medium include a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read only memory (ROM), a rigid magnetic disk and an optical disk. Current examples of optical disks include compact disk read only memory (CD-ROM), compact disk read/write (CD-R/W), and DVD.

Improvements and modifications can be made to the foregoing without departing from the scope of the present invention.

CLAIMS

1. A method for selection of a runtime stack for deployment of a Web Service, comprising:

generating (403) policy assertions (262A, 262B) for a Web Service (220) to be deployed;

providing (405) a scoring mechanism (215, 216, 271-273) for each available runtime stack (211, 212, 241-243) in which the ability of a stack to support each of a plurality of policy assertions is scored;

applying the scoring mechanism (215, 216, 271-273) for each available runtime stack (211, 212, 241-243) to the policy assertions (262A, 262B) for the Web Service (220) to be deployed; and

selecting a stack (211, 212, 241-243) based on the results of applying the scoring mechanism (215, 216, 271-273).

2. A method as claimed in claim 1, wherein generating (403) policy assertions (262A, 262B) for a Web Service (220) to be deployed includes using specification defined Web Service policy assertions (222).

3. A method as claimed in claim 1 or claim 2, wherein generating (403) policy assertions (262A, 262B) for a Web Service (220) to be deployed includes:

scanning a service description (221) in WSDL (Web Services Description Language) for WSDL elements of the Web Service (220) and mapping (282) the WSDL elements to policy assertions.

4. A method as claimed in any one of claims 1 to 3, wherein generating (403) policy assertions (262A, 262B) for a Web Service (220) to be deployed includes:

converting side information (223) requirements of the Web Service (220) and mapping (281) the requirements to policy assertions.

5. A method as claimed in claim 4, wherein the side information (223) is provided by a provider (210) or a requester (240) of the Web Service (220) to be deployed and the side information (223) relates to service requirements of the provider (210) or requester (240).

6. A method as claimed in any one of the preceding claims, wherein generating policy assertions (262A, 262B) for a Web Service (220) to be deployed includes normalizing (404) a policy to include one or more alternative sets of policy assertions, and wherein applying the scoring mechanism (215, 216, 271-273) is applied to each set of policy assertions.

7. A method as claimed in claim 6, wherein if the selection of runtime stack (211, 212, 241-243) is for a Web Service provider (210), the results of the scoring mechanism are used to select a stack (211, 212, 241-243) which supports all the sets of policy assertions.

8. A method as claimed in claim 6, wherein if the selection of runtime stack (211, 212, 241-243) is for a Web Service requester (240), the results of the scoring mechanism are used to select a stack (211, 212, 241-243) which best supports one of the alternative sets of policy assertions.

9. A method as claimed in any one of the preceding claims, wherein the scoring mechanism gives a positive value if a policy assertion is supported and a negative value if a policy assertion is not supported by a stack (211, 212, 241-243).

10. A method as claimed in claim 9, wherein the values are combined for a set of policy assertions required for a Web Service (220).

11. A method as claimed in any one of the preceding claims, wherein a runtime stack (211, 212, 241-243) handles the Web Services protocol and transport processing required for a request or response.

12. A computer program comprising computer program code to, when loaded into a computer system and executed thereon, cause said computer system to perform all the steps of the method of any one of claims 1 to 11.

13. A system for selection of a runtime stack for deployment of a Web Service, comprising:

a runtime container (210, 240) including a processor and a plurality of stacks (211, 212, 241-243) to which a Web Service (220) could be deployed;

a generator (261A, 261B) for generating policy assertions (262A, 262B) for a Web Service (220) to be deployed;

a scoring mechanism (215, 216, 271-273) for each available runtime stack (211, 212, 241-243) in which the ability of a stack (211, 212, 241-243) to support each of a plurality of policy assertions is scored;

applying the scoring mechanism (215, 216, 271-273) for each available runtime stack (211, 212, 241-243) to the policy assertions (262A, 262B) for the Web Service (220) to be deployed; and

a selector (265A, 265B) for selecting a stack (211, 212, 241-243) based on the results of applying the scoring mechanism (215, 216, 271-273).

14. A system as claimed in claim 13, wherein the generator (261A, 261B) includes listing specification defined Web Service policy assertions (222).

15. A system as claimed in claim 13 or claim 14, wherein the generator (261A, 261B) includes:

a mechanism for scanning a service description (221) in WSDL for WSDL elements of the Web Service (220) and rules (282) mapping the WSDL elements to policy assertions.

FIG. 1

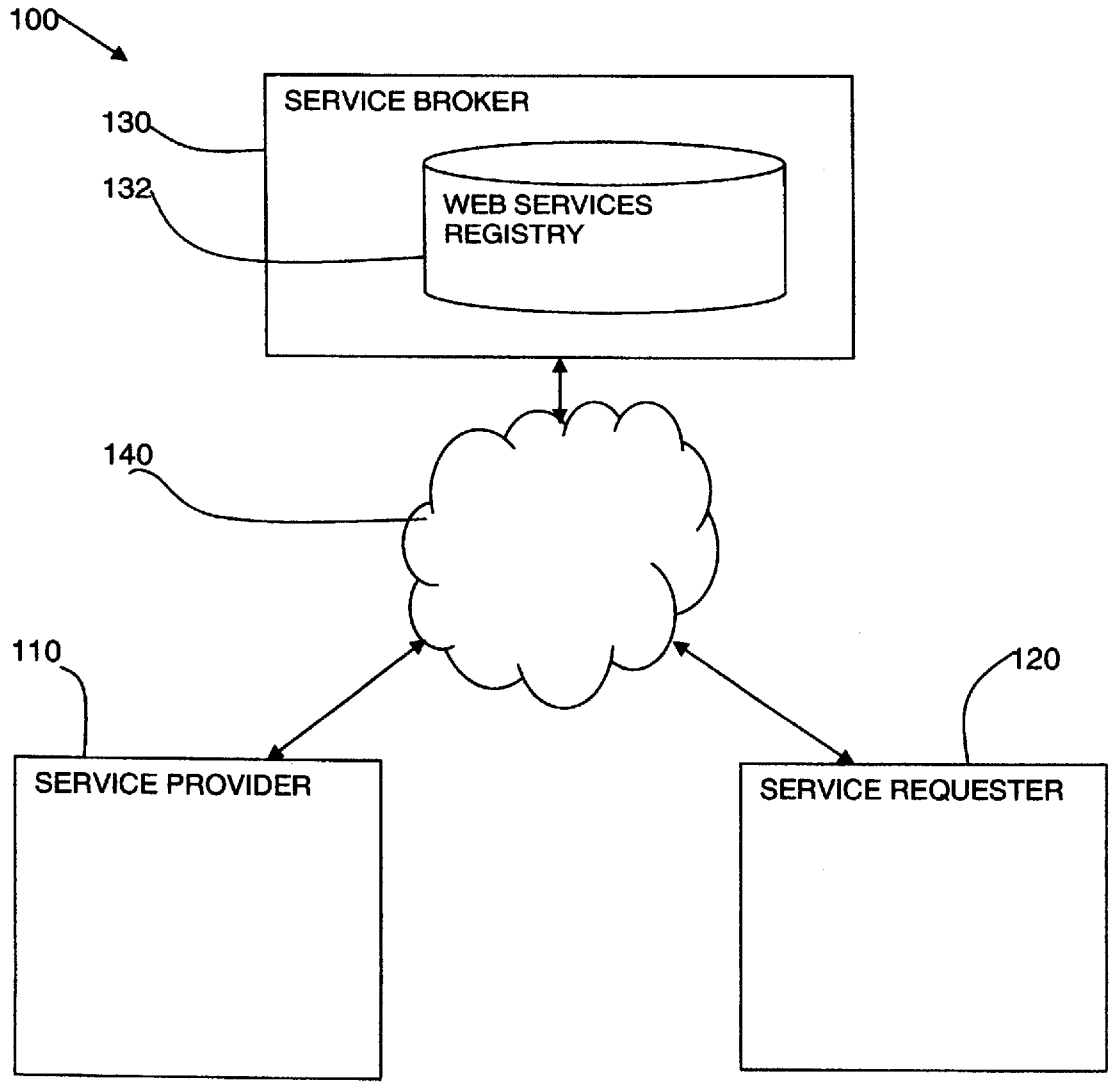

100
SERVICE BROKER
130
132
WEB SERVICES REGISTRY
140
110
120
SERVICE PROVIDER
SERVICE REQUESTER

FIG. 2A

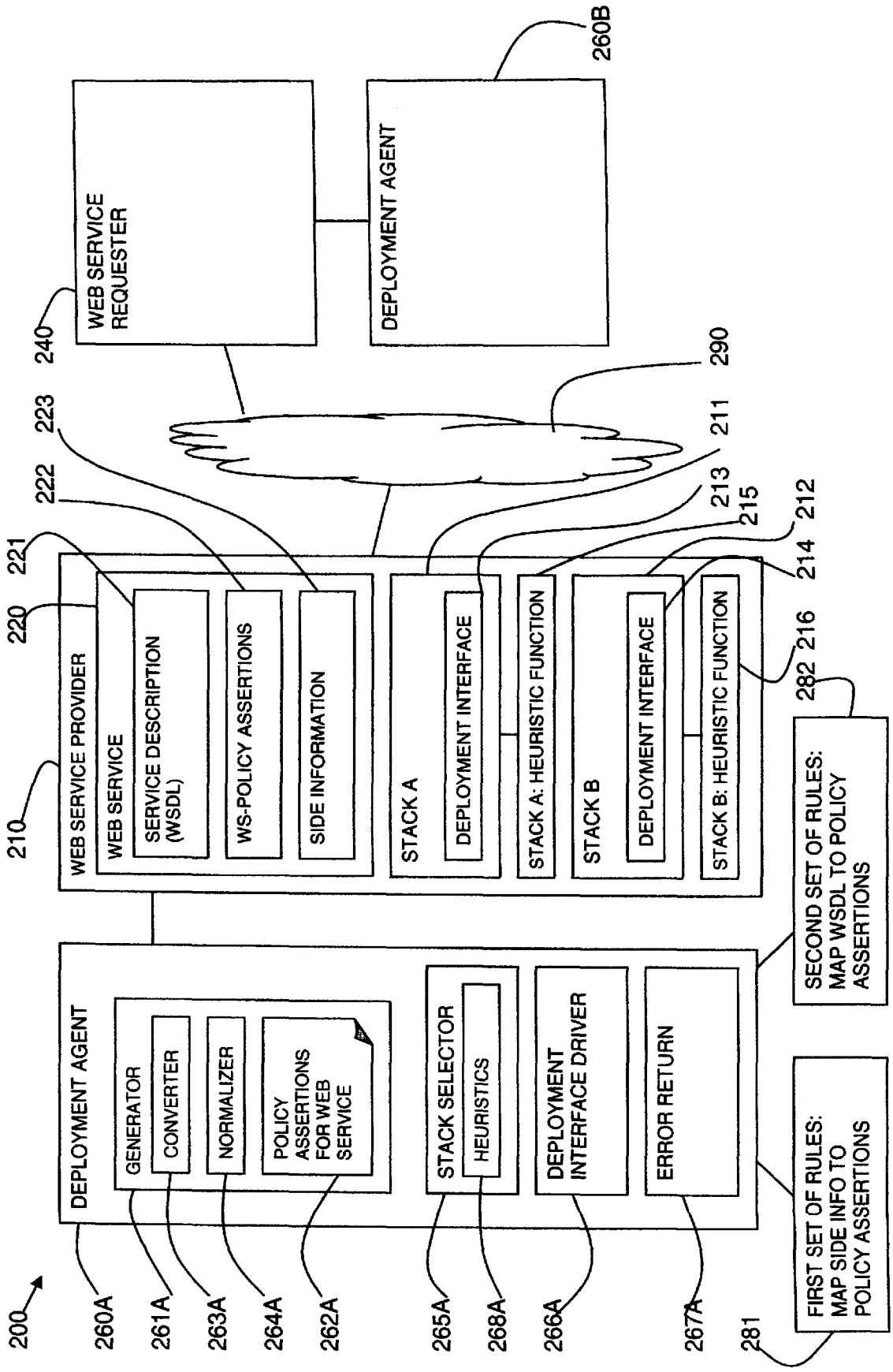

200
260A
261A
263A
264A
262A
265A
268A
266A
267A
281
DEPLOYMENT AGENT
GENERATOR
CONVERTER
NORMALIZER
POLICY ASSERTIONS FOR WEB SERVICE
STACK SELECTOR
HEURISTICS
DEPLOYMENT INTERFACE DRIVER
ERROR RETURN
FIRST SET OF RULES: MAP SIDE INFO TO POLICY ASSERTIONS
SECOND SET OF RULES: MAP WSDL TO POLICY ASSERTIONS
282
210
WEB SERVICE PROVIDER
220
221
WEB SERVICE
SERVICE DESCRIPTION (WSDL)
222
WS-POLICY ASSERTIONS
223
SIDE INFORMATION
STACK A
DEPLOYMENT INTERFACE
213
STACK A: HEURISTIC FUNCTION
215
STACK B
DEPLOYMENT INTERFACE
212
214
STACK B: HEURISTIC FUNCTION
216
211
290
240
WEB SERVICE REQUESTER
DEPLOYMENT AGENT
260B

FIG. 2B

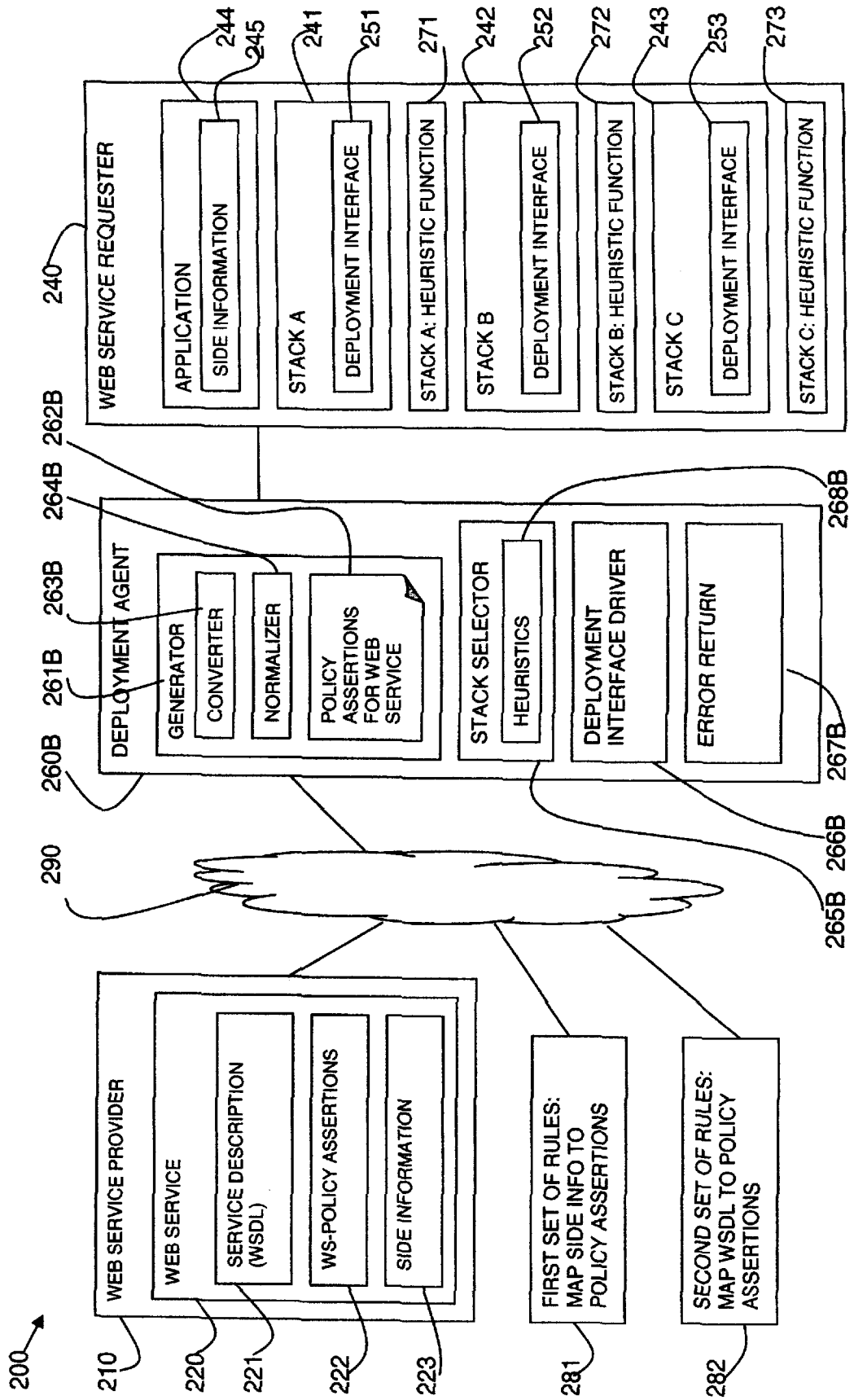

200
210
WEB SERVICE PROVIDER
220
WEB SERVICE
221
SERVICE DESCRIPTION (WSDL)
222
WS-POLICY ASSERTIONS
223
SIDE INFORMATION
281
FIRST SET OF RULES: MAP SIDE INFO TO POLICY ASSERTIONS
282
SECOND SET OF RULES: MAP WSDL TO POLICY ASSERTIONS
290
260B
DEPLOYMENT AGENT
261B
GENERATOR
263B
CONVERTER
NORMALIZER
264B
POLICY ASSERTIONS FOR WEB SERVICE
262B
240
WEB SERVICE REQUESTER
244
APPLICATION
245
SIDE INFORMATION
241
STACK A
251
DEPLOYMENT INTERFACE
271
STACK A: HEURISTIC FUNCTION
242
STACK B
252
DEPLOYMENT INTERFACE
272
STACK B: HEURISTIC FUNCTION
243
STACK C
253
DEPLOYMENT INTERFACE
273
STACK C: HEURISTIC FUNCTION
265B
STACK SELECTOR
HEURISTICS
266B
DEPLOYMENT INTERFACE DRIVER
267B
ERROR RETURN
268B

FIG. 3

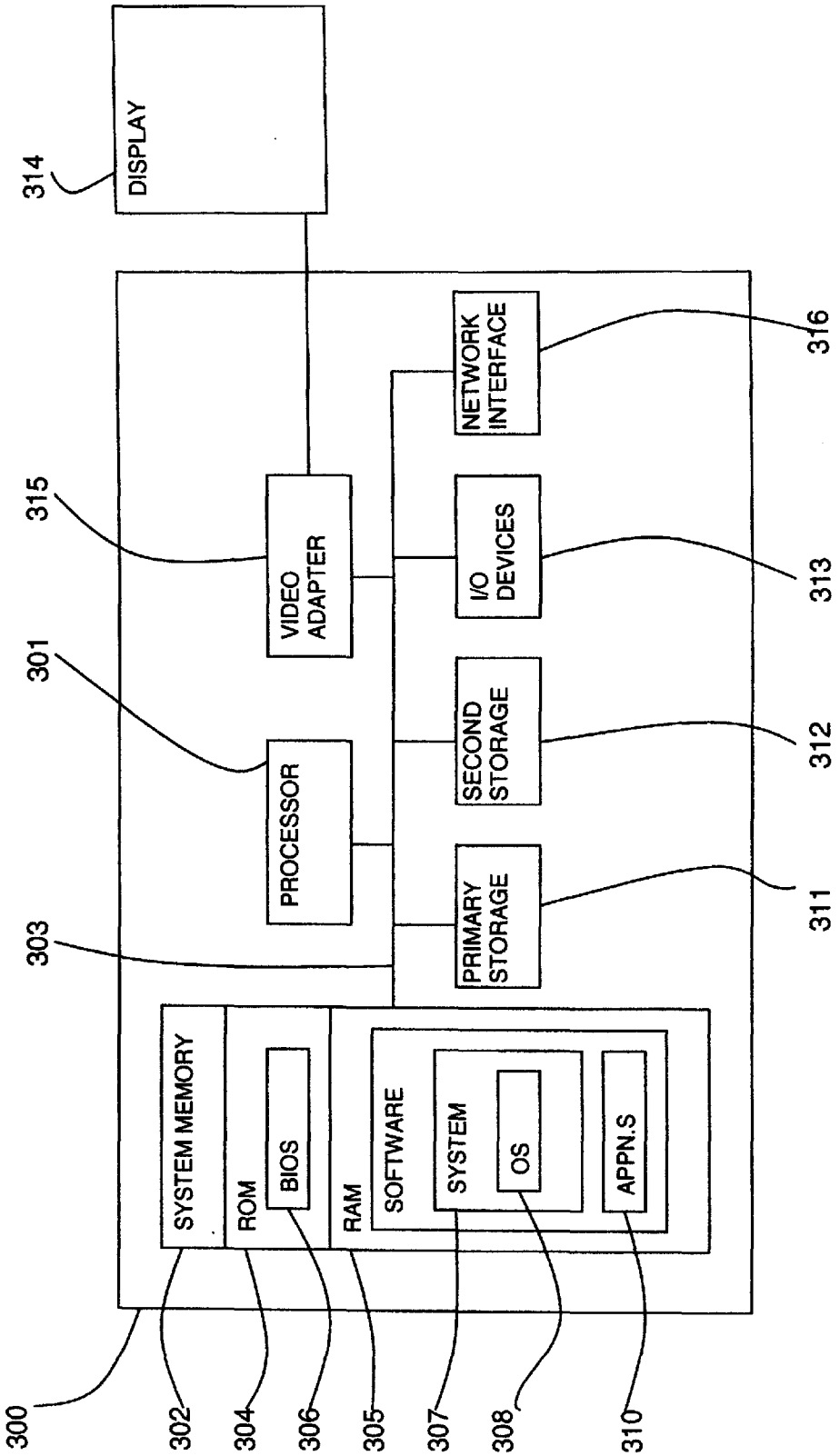
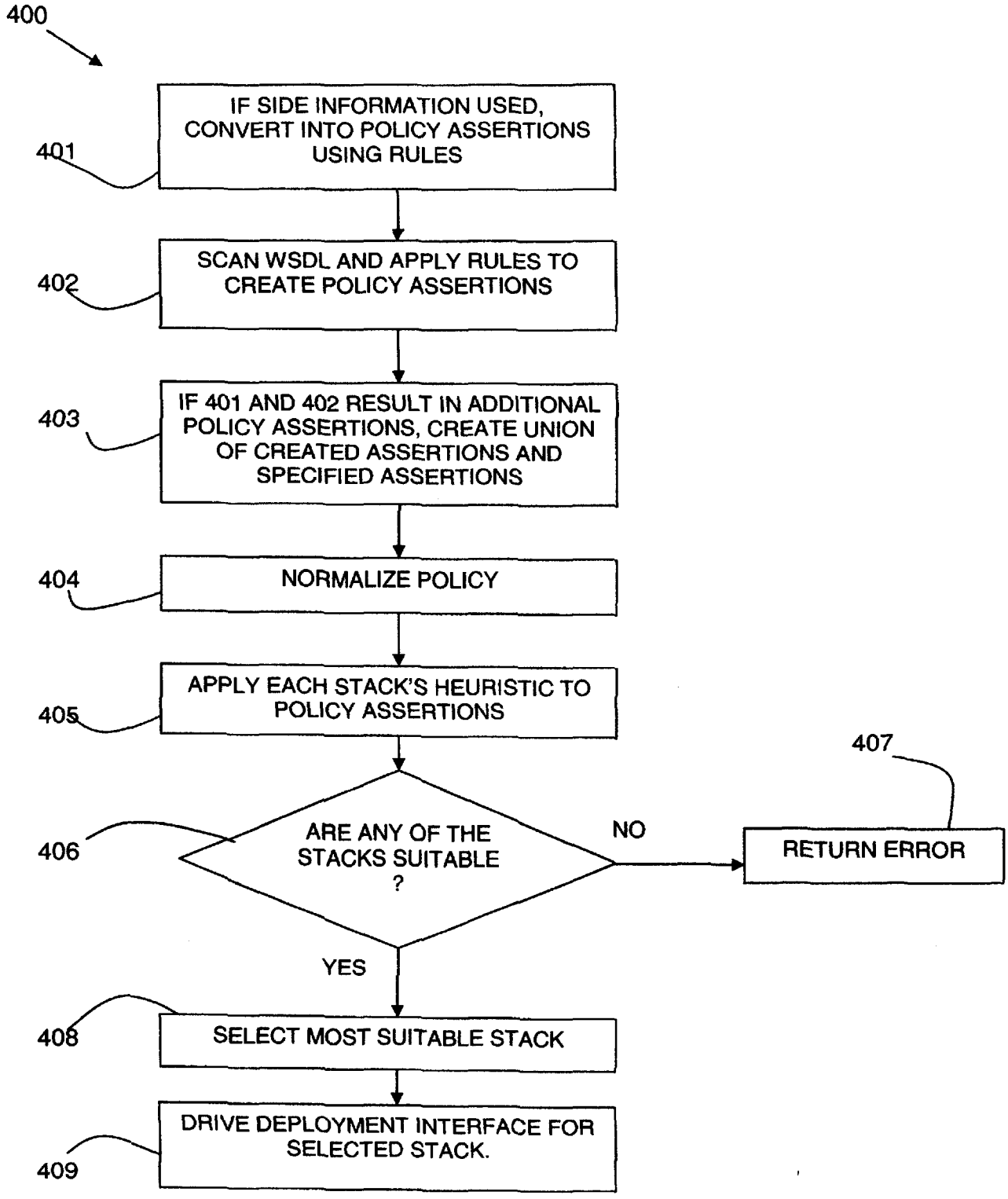

DISPLAY
314
NETWORK INTERFACE
316
VIDEO ADAPTER
315
I/O DEVICES
313
PROCESSOR
301
SECOND STORAGE
312
PRIMARY STORAGE
311
303
SYSTEM MEMORY
ROM
BIOS
RAM
SOFTWARE
SYSTEM
OS
APPN.S
300
302
304
306
305
307
308
310

FIG. 4


400
IF SIDE INFORMATION USED, CONVERT INTO POLICY ASSERTIONS USING RULES
401
SCAN WSDL AND APPLY RULES TO CREATE POLICY ASSERTIONS
402
IF 401 AND 402 RESULT IN ADDITIONAL POLICY ASSERTIONS, CREATE UNION OF CREATED ASSERTIONS AND SPECIFIED ASSERTIONS
403
NORMALIZE POLICY
404
APPLY EACH STACK'S HEURISTIC TO POLICY ASSERTIONS
405
ARE ANY OF THE STACKS SUITABLE ?
406
NO
RETURN ERROR
407
YES
SELECT MOST SUITABLE STACK
408
DRIVE DEPLOYMENT INTERFACE FOR SELECTED STACK.
409

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/050628

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/46

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	AARON SKONNARD: "Understanding WS-Policy" INTERNET CITATION August 2003 (2003-08), pages 1-8, XP002355708 Retrieved from the Internet: URL:http://msdn.microsoft.com/library/en-us/dnwebsrv/html/understwspol.asp?frame=true> [retrieved on 2005-11-24] page 1 -----	1-15

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

☐ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 March 2010	06/04/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	de Man, Ronald