Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
The Journal of Systems and Software 70 (2004) 107–116
The Journal of Systems and Software
www.elsevier.com/locate/jss

Flexible retrieval of Web Services

Hai Zhuge *, Jie Liu

Knowledge Grid Research Group, Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704, Beijing 100080, China

Received 25 June 2002; received in revised form 27 December 2002; accepted 2 January 2003

Abstract

An important issue arising from Web Service applications is how to conveniently, accurately and efficiently retrieve services from large-scale and expanding service repositories. This paper proposes a flexible Web Service retrieval approach, which solves this issue by means of an orthogonal service space and establishing the multi-valued specialization relationships between services. The similarity degree between services is measured based on the specialization relationship between operations defined in services. An SQL-like flexible query language is used to support the flexible retrieval of services. The related programming environment and graphical user operation interface of the language have been implemented in the Service Grid environment. Compared with the current UDDI-based service retrieval approach, the proposed approach has the advantages of convenience, accuracy and efficiency.
© 2003 Elsevier Inc. All rights reserved.

Keywords: Flexible retrieval; Service grid; Specialization; UDDI; Web services

1. Introduction

Web Services are applications that interact with each other using Web standards such as XML, SOAP (Simple Object Access Protocol, www.w3.org/TR/SOAP), WSDL (Web Service Description Language, www.w3.org/TR/wsdl) and UDDI (Universal Description Discovery and Integration, www.uddi.org). The aim is to provide an open platform for the development, deployment, interaction, and management of globally distributed e-services. Web Services offer the possibility of running an application on distributed machines, many of which the users neither own nor control (Shirky, 2002). Web Services would change the Internet into a computing platform, rather than a medium where users primarily just view and download content (Vaughan-Nichols, 2002). People can use Web Services to access software for use as components to build their own applications and services (Michael, 2002).

A business-driven approach for component-based and service-oriented development has been proposed (Keith and Ali, 2002). The challenges to build a scalable Web Service have been pointed out (Bosworth, 2001). A solution to build a scalable and efficient service discovery framework has been proposed (Xu et al., 2001), which adds QoS feedback capability to the clients and caches the service discovery results with QoS feedbacks in the Discovery Server hierarchy. DAML markup language enables a wide variety of agent technologies for automated Web Service discovery, execution, composition, and interoperability (McIlraith et al., 2001).

The current UDDI specification is a solution to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services across the Internet (UDDI. Org, 2001a). In order to achieve the interoperability between distributed services, application developers can use the UDDI repositories to find the publicly available services on the Internet. However, the current UDDI repositories only enable users to retrieve services based on the keywords (IBM, 2001; Microsoft, 2001). In particular, if users are not familiar with the pre-specified

* Corresponding author. Tel.: +86-1062565533; fax: +86-1062567724.
E-mail address: zhuge@ict.ac.cn (H. Zhuge).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0164-1212(03)00003-7

service categories, they usually cannot get the satisfied retrieval results. Applications show that the current UDDI repositories cannot meet the needs of the business processes in efficiency and accuracy (Zhang et al., 2002).

How to conveniently, accurately and efficiently retrieve the required services from the large-scale and expanding service repositories becomes an important issue in Web Service applications. This paper solves this issue by making use of the Service Grid, which organizes services in a normalized and orthogonal multi-dimensional service space (Zhuge, 2002b,c,d).

A referential model for the service space can be expressed as *Service Grid* = (*classification-type, category, content*), where the *classification-type* dimension indicates the service classification standards, the *category* dimension reflects the detail-classified hierarchy related to a classification type, and the *content* dimension reflects the functionality of the services. The multi-valued specialization relationships and the similarity degree between services and between tModels are established in the *Service Grid*. Users' requirements are matched in the Service/tModel specialization graphs of a subspace of the entire service space so as to raise the retrieval effectiveness and efficiency. To realize the flexible retrieval of services, we have developed an SQL-like query language, the user interface, and the programming environment.

2. tModel specification and specialization

The *tModel* serves a useful purpose in discovering information about interfaces and other technical foundation concepts exposed for broad use by an individual service or registration instance (see www.uddi.org). The information that makes up a *tModel* includes: a *tModel key*, a *name*, an *optional description*, an *operator*, an *overview document*, an *identifier Bag*, a *category Bag*, and an *authorized name*. The *identifier Bag* and the *category Bag* are elements that hold zero or more instances of *keyedReference* elements. Every *tModel* further consists of a set of operations, which include the *name*, the *description*, the *input* and the *output* messages.

Four basic elements can be abstracted from a *tModel*: *operation*, *category*, *identifier* and *keyword set*. So a *tModel* can be represented as $TM : (OpSet = \{op_1 : \sigma_1 \to \tau_1, op_2 : \sigma_2 \to \tau_2, \ldots, op_n : \sigma_n \to \tau_n\}, CATEGORY, IDENTIFIER, KEYWORDS)$, where σ and τ are the input message and output message of operation *op*. The $CATEGORY$ is an optional list of name-value pairs, which includes one of the classification types. The *IDENTIFIER* is also an optional list of name-value pairs, which includes information about common forms of identification like the D-U-N-S numbers (www.uddi.org/pubs/DataStructure-v2.00-Open-20010608.pdf). The *KEYWORDS* indicates the keyword set of the *tModels*. Based on the basic concepts illustrated above, we can define the *tModel* specialization relationships as follows.

Definition 1. Let $op : \sigma \to \tau$ and $op' : \sigma' \to \tau'$ be two operations. If (1) $\sigma' \subseteq \sigma$ and $\tau' \subseteq \tau$; (2) for every $\xi \in \sigma$ and $\xi' \in \sigma'$, if $\xi = \xi'$, then $op(\xi) = op'(\xi')$; and, (3) the axiom of op' implies the axiom of op, then op' is called an operation specialization from *op*, denoted as $op \text{—F}\to op'$.

Definition 2. Let $TM : (OpSet = \{op_1 : \sigma_1 \to \tau_1, op_2 : \sigma_2 \to \tau_2, \ldots, op_n : \sigma_n \to \tau_n\}, CATEGORY, IDENTIFIER, KEYWORDS)$ and $TM' : (OpSet' = \{op'_1 : \sigma'_1 \to \tau'_1, op'_2 : \sigma'_2 \to \tau'_2, \ldots, op'_n : \sigma'_n \to \tau'_n\}, CATEGORY', IDENTIFIER', KEYWORDS')$ be two *tModels*, which satisfy $TM \neq NULL$ and $TM' \neq NULL$. TM' is called an *identical-specialization* from TM, denoted as $TM \text{—I}\to TM'$ if there exists an isomorphism from TM into TM' such that: (1) for every $op \in TM$, there exists a corresponding $op' \in TM'$, which satisfies $op \text{—F}\to op'$; (2) $CATEGORY'$ is the same as or the subcategory of $CATEGORY$; (3) $IDENTIFIER'$ is the same as or the subcategory of $IDENTIFIER$ and, (4) $KEYWORDS' = KEYWORDS$.

According to the transitivity of the *identical-specialization* conditions, the following characteristics can be deduced: (1) Reflexive: $TM \text{—I}\to TM$, and (2) Transitive: $TM \text{—I}\to TM'$, $TM' \text{—I}\to TM'' \Rightarrow TM \text{—I}\to TM''$.

Definition 3. Multi-valued specialization relationships between *tModels*

(1) A *tModel* TM' is called a *partial-specialization* from another *tModel* TM, denoted as $TM \text{—P}\to TM'$, if there exists a $TM'' : (OpSet'', CATEGORY'', IDENTIFIER'', KEYWORDS'')$ that satisfies: $OpSet'' \subseteq OpSet$, $CATEGORY'' \subseteq CATEGORY$, $IDENTIFIER'' \subseteq IDENTIFIER$, $KEYWORDS'' \subseteq KEYWORDS$, and $TM'' \text{—I}\to TM'$;
(2) A *tModel* TM' is called an *extension-specialization* from another *tModel* TM, denoted as $TM \text{—E}\to TM'$, if there exists a $TM'' : (OpSet'', CATEGORY'', IDENTIFIER'', KEYWORDS'')$ that satisfies: $OpSet'' \subseteq OpSet'$, $CATEGORY'' \subseteq CATEGORY'$, $IDENTIFIER'' \subseteq IDENTIFIER'$, $KEYWORDS'' \subseteq KEYWORDS'$, and $TM \text{—I}\to TM''$;

(3) A *tModel TM′* is called a revision-specialization from another *tModel TM*, denoted as $TM \xrightarrow{R} TM'$, if there exists a $TM_1 : (OpSet_1, CATEGORY_1, IDENTIFIER_1, KEYWORDS_1)$ and a $TM_2 : (OpSet_2, CATEGORY_2, IDENTIFIER_2, KEYWORDS_2)$ that satisfies: $OpSet_1 \subseteq OpSet$, $CATEGORY_1 \subseteq CATEGORY$, $IDENTIFIER_1 \subseteq IDENTIFIER$, $KEYWORDS_1 \subseteq KEYWORDS$, $OpSet_2 \subseteq OpSet'$, $CATEGORY_2 \subseteq CATEGORY'$, $IDENTIFIER_2 \subseteq IDENTIFIER'$, $KEYWORDS_2 \subseteq KEYWORDS'$, and $TM_1 \xrightarrow{I} TM_2$;

(4) A *tModel TM′* is called a *non-specialization* from another *tModel TM*, denoted as $TM \not\rightarrow TM'$, if $KEYWORDS \cap KEYWORDS' = NULL$ and for every $op \in OpSet$, there does not exist any corresponding $op' \in OpSet'$, which satisfies $op \xrightarrow{F} op'$.

3. Web Service specification and specialization

The information for describing a service includes: a *service key*, a *business key*, a *service name*, an *optional description*, a *category Bag* and the *bindingTemplates*, which contain one or more *bindingTemplate* structures. The *bindingTemplate* holds the technical service description information related to a given business service family and it is described by the *tModelInstanceDetails*, the *accessPoint* or the *hostingRedirector*. The *tModelInstanceDetails* structure is a list of zero or more *tModelInstanceInfo* elements and forms a distinct fingerprint to identify compatible services (UDDI. Org, 2001b).

A service can be abstracted as $WS : (TM =< \{TM_1, \ldots TM_n\}, CATEGORY, KEYWORDS)$, and the specialization relationships between two services can be defined as follows.

Definition 4. A Web Service *WS′* is called an *identical-specialization* from another Web Service *WS*, denoted as $WS \xrightarrow{I} WS'$ if there exists an isomorphism from *WS* into *WS′* such that: (1) for every *TM* of *WS*, there exists a corresponding *TM′* of *WS′* which satisfies $TM \xrightarrow{I} TM'$; (2) *CATEGORY′* is the same as or the subcategory of *CATEGORY*; and, (3) $KEYWORDS' = KEYWORDS$.

Definition 5. Multi-valued specialization relationships between Web Services.

(1) A Web Service *WS′* is called a *partial-specialization* from another Web Service *WS*, denoted as $WS \xrightarrow{P} WS'$, if there exists a $WS'' : (TM'', CATEGORY'', KEYWORDS'')$, that satisfies $TM'' \subseteq TM$, $CATEGORY'' \subseteq CATEGORY$, $KEYWORDS'' \subseteq KEYWORDS$, and $WS'' \xrightarrow{I} WS'$.

(2) A Web Service *WS′* is called an *extension-specialization* from another Web Service *WS*, denoted as $WS \xrightarrow{E} WS'$, if there exists a $WS'' : (TM'', CATEGORY'', KEYWORDS'')$, that satisfies $TM'' \subseteq TM'$, $CATEGORY'' \subseteq CATEGORY'$, $KEYWORDS'' \subseteq KEYWORDS'$, and $WS \xrightarrow{I} WS''$.

(3) A Web Service *WS′* is called a *revision-specialization* from another Web Service *WS*, denoted as $WS \xrightarrow{R} WS'$, if there exists a $WS_1 : (TM_1, CATEGORY_1, KEYWORDS_1)$ and a $WS_2 : (TM_2, CATEGORY_2, KEYWORDS_2)$, that satisfies $TM_1 \subseteq TM$, $CATEGORY_1 \subseteq CATEGORY$, $KEYWORDS_1 \subseteq KEYWORDS$, $TM_2 \subseteq TM'$, $CATEGORY_2 \subseteq CATEGORY'$, $KEYWORDS_2 \subseteq KEYWORDS'$, and $WS_1 \xrightarrow{I} WS_2$.

(4) A Web Service *WS′* is called a *non-specialization* from another Web Service *WS*, denoted as $WS \not\rightarrow WS'$, if $KEYWORDS \cap KEYWORDS' = NULL$ and for every *TM* of *WS*, there does not exist any corresponding *TM′* of *WS′*, such that $TM \xrightarrow{I} TM'$.

The services that do not contain *tModels* but operations in the data structure can be abstracted as $WS : (OpSet = \{op_1 : \sigma_1 \rightarrow \tau_1, op_2 : \sigma_2 \rightarrow \tau_2, \ldots, op_n : \sigma_n \rightarrow \tau_n\}, CATEGORY, KEYWORDS)$, and the definition of the multi-valued specialization relationships between services is similar to Definitions 2 and 3.

Let us take two Web Services for example: WS_1 "*UDDI Services*" provided by Microsoft and WS_2 "*UDDI Business Registry inquiry*" provided by IBM Corporation. WS_1 includes six *tModels* named "*uddi-org*: *inquiry*" "*uddi-org*: *inquiry_v2*", "*uddi-org*: *publication*", "*uddi-org*: *publication_v2*", "*Microsoft-com:uddi-api-extension_v2*", and "*uddi-org*: *http*". WS_2 includes two *tModels* named "*uddi-org*: *inquiry*" and "*uddi-org*: *http*". Both the category of WS_1 and WS_2 belong to the general category of UDDI category Bag. The keywords of WS_1 include "*UDDI Services*", "*inquiry*", "*publication*" and "*Microsoft*". The keywords of WS_2 include "*Business*", "*inquiry*" and "*IBM*". The two services share two *tModels*: "*uddi-org*: *inquiry*" and"*uddi-org*: *http*". According to the service specialization relationships defined in Definition 5, WS_2 is the "*revision-specialization*" relationship of WS_1 denoted as $WS_1 \xrightarrow{R} WS_2$.

4. Flexible retrieval of web services

4.1. Similarity between tModels

Let *OpSet* and *OpSet′* be the operation sets in *TM* and *TM′*, respectively, μ be the number of operation specialization between *OpSet* and *OpSet′*. The operation-similarity, the category-similarity, the identifier-similarity and the keyword-similarity can be defined as follows:

$$\textit{operation-similarity}(\textit{OpSet}, \textit{OpSet}') = 2 \times \mu / (|\textit{OpSet}| + |\textit{OpSet}'|) \tag{1}$$

$$\textit{category-similarity}(c_1, c_2) = 1/(d(c_1, c_2) + 1) \tag{2}$$

where the category-distance of c_1 and c_2 is defined as

$$d(c_1, c_2) = \begin{cases} 0, & \text{if } c_1 \text{ is the same as } c_2 \\ 1, & \text{if } c_1 \text{ is a direct subclass of } c_2 \\ d(c_1, c_k) + d(c_k, c_2), & \text{if } c_k \text{is on the path from } c_1 \text{ to } c_2 \\ d(c_2, c_1), & \text{if } c_1 \text{ is the superclass of } c2 \\ +\infty, & \text{if there is no relationship between } c_1 \text{and } c_2 \end{cases}$$

$$\textit{Identifier-similarity} = 1/(d(i_1, i_2) + 1) \tag{3}$$

where the identifier-distance of i_1 and i_2 is defined as:

$$d(i_1, i_2) = \begin{cases} 0, & \text{if } i_1 \text{ is the same as } i_2 \\ 1, & \text{if } i_1 \text{ is a direct subclass of } i_2 \\ d(i_1, i_k) + d(i_k, i_2), & \text{if } i_k \text{ is on the path from } i_1 \text{ to } i_2 \\ d(i_2, i_1), & \text{if } i_1 \text{ is the superclass of } i_2 \\ +\infty, & \text{if there is no relationship between } i_1 \text{ and } i_2 \end{cases}$$

$$\textit{Keyword-similarity}(k_1, k_2) = 2 \times \theta / (|k_1| + |k_2|) \tag{4}$$

where θ is the number of the common keywords in k_1 and k_2.

Definition 6. Let $\alpha, \beta, \gamma, \lambda \in [0, 1]$ be the weights of operation-similarity, category-similarity, identifier-similarity and keyword-similarity, $\alpha + \beta + \gamma + \lambda = 1$. The similarity degree between two *tModels* can be measured by the following formula:

$$\begin{aligned} \textit{TM-similarity}(\textit{TM}, \textit{TM}') = {} & \alpha \times \textit{operation-similarity}(\textit{OpSet}, \textit{OpSet}') \\ & + \beta \times \textit{category-similarity}(\textit{CATEGORY}, \textit{CATEGORY}') \\ & + \gamma \times \textit{identifier-similarity}(\textit{IDENTIFIER}, \textit{IDENTIFIER}') \\ & + \lambda \times \textit{keyword-similarity}(\textit{KEYWORDS}, \textit{KEYWORDS}') \end{aligned} \tag{5}$$

4.2. Similarity between services

The similarity degree between services reflects the degree of sharing *tModels* or operations.

Definition 7. Let $\alpha, \beta, \gamma \in [0, 1]$ be the weights of *tModel*-similarity, category-similarity, and keyword-similarity, $\alpha + \beta + \gamma = 1$, *TM* and *TM′* be two *tModel* sets in *WS* and *WS′*, and υ be the number of *identical-specialization* relationship between *TM* and *TM′*. The similarity degree between *WS* and *WS′* can be denoted as:

$$\begin{aligned} \textit{WS-similarity}(\textit{WS}, \textit{WS}') = {} & \alpha \times \textit{TMS-similarity}(\textit{TM}, \textit{TM}') + \beta \times \textit{category-similarity}(\textit{CATEGORY}, \textit{CATEGORY}') \\ & + \gamma \times \textit{keyword-similarity}(\textit{KEYWORDS}, \textit{KEYWORDS}') \end{aligned} \tag{6}$$

$$\textit{TMS-similarity}(\textit{TM}, \textit{TM}') = 2 \times \upsilon / (|\textit{TM}| + |\textit{TM}'|) \tag{7}$$

Definition 8. Let $\alpha, \beta, \gamma \in [0, 1]$ be the weights of operation-similarity, category-similarity, and keyword-similarity, $\alpha + \beta + \gamma = 1$. The similarity degree between two *services* that share operations can be measured by the following formula:

$$\begin{aligned} \textit{WS-similarity}(WS, WS') = {} & \alpha \times \textit{operation-similarity}(OpSet, OpSet') \\ & + \beta \times \textit{category-similarity}(CATEGORY, CATEGORY') \\ & + \gamma \times \textit{keyword-similarity}(KEYWORDS, KEYWORDS') \end{aligned} \tag{8}$$

5. Implementation

5.1. General architecture and approach

The general architecture of the proposed approach is illustrated in Fig. 1, which mainly consists of three layers: user interface, the Service Grid, and the UDDI repositories.

Application developers need to provide the source information about a service or a *tModel* through the Grid Operation Language (GOL) programming environment, while the end users can use the Graphical User Interface (GUI) to provide source information. Referring to the specialization relationships between services and between *tModels*, the Service Grid adopts heuristic graph searching on the multi-valued specialization hierarchy to acquire the target *service* ID or *tModel* ID and then to retrieve the service repository and *tModel* repository to get the detail. If there are no matching services or *tModels* found, the Service Grid will send SOAP messages encapsulating users' requirements to the UDDI repositories, and then get the returned retrieval result.

The service repository defines a set of service specialization graphs (*SSG*). Each *SSG* is a graph in which nodes are services, and arcs are the multi-valued specialization relationships between them. The *tModel* repository defines a set of *tModel* specialization graphs (*TSG*). Each *TSG* includes *tModels* nodes and specialization relationships between *tModel* nodes. The services in an *SSG* include the *tModels* defined in the *tModel* repository.

The similarity degree between services (*SSD*) and between *tModels* (*TSD*) can be recorded in a multi-valued specialization matrix, denoted as $SSD/TSD = (sd_{ij})_{e\times e}$, where e denotes the total number of services/*tModels* in the repository, sd_{ij} represents the similarity degree between S_i and S_j, which satisfies $sd_{ii} = 1$, $sd_{ij} = sd_{ji}$, and $sd_{ij} \in [0, 1]$.

The establishment of multi-valued specialization relationships and similarity degree between services and between *tModels* consists of the following steps:

(1) *Service retrieval*. Retrieve services and *tModels* from UDDI repositories and store them in the Service Grid with a specified classification-type, category and content coordinate.
(2) *Meta-information extraction*. Extract the meta-information of services and *tModels* in the Service Grid. The meta-information for describing a service in the Service Grid includes: *classification-type*, *category*, *content*, *service Key*, *service name*, *tModel Key*, *category Bag*, and *keywords*. Accordingly, the meta-information for describing a *tModel* in the Service Grid includes: *classification-type*, *category*, *content*, *tModel Key*, *tModel name*, *overview Document*, *operation*, *category Bag*, *identifier Bag*, and *keywords*.

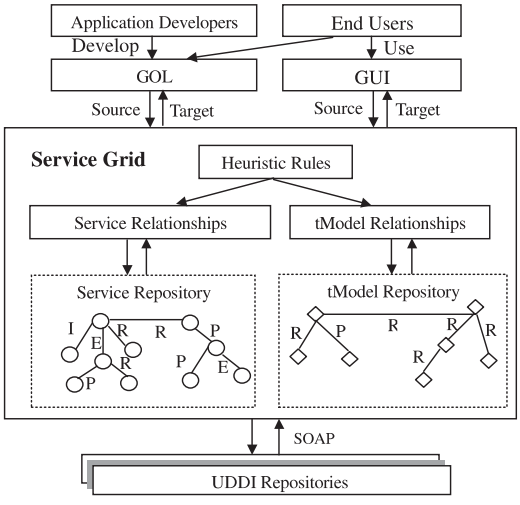

Fig. 1. General architecture of the flexible service retrieval approach.

(3) *Operation specialization establishment*. Load the overview document of the *tModels*, extract the name, description, input messages and output messages of each operation and then establish the operation specialization according to Definition 1.
(4) *Multi-valued specialization relationships establishment*. Establish *tModel* specialization graphs (*TSG*) according to Definitions 2 and 3, and establish service specialization graphs (*SSG*) according to Definitions 4 and 5.
(5) *Similarity degree computing*. Compute the similarity degree between *tModels* and between services and store the similarity degree in relational tables *TSD* and *SSD*.

Steps 1–5 can be accomplished by either the designers of the Service Grid or the assistant tool.

Further more, a set of reuse-association rules has been established for the flexible Web Service retrieval, which are similar to the model inheritance rules proposed in (Zhuge, 2000) and can be used for the heuristic retrieval (Cilia and Buchmann, 2000). These rules include the following types: (1) $WS_1 —\gamma\rightarrow WS_2$, $WS_2 —\gamma\rightarrow WS_3 \Rightarrow WS_1 —\gamma\rightarrow WS_3$, where $\gamma \in \{I, P, E\}$; (2) $WS_1 —\text{I}\rightarrow WS_2 \Rightarrow WS_1 —\gamma\rightarrow WS_2$, where $\gamma \in \{P, E, R\}$; and, (3) $WS_1 —\text{I}\rightarrow WS_2$, $WS_2 —\gamma\rightarrow WS_3 \Rightarrow WS_1 —\gamma\rightarrow WS_3$, where $\gamma \in \{P, E, R\}$. These reuse-association rules have been proved (Zhuge, 1998, 2002a).

5.2. Flexible Web Service query language

GOL is an XML-based programming language and environment proposed for application developers to retrieve the required services through a set of SQL-like commands. A major form of the GOL statement can be described as:

<GOL Statement>::= Get [ALL | DISTINCT]<Target Service>
FROM<Service-Repository>
[WHERE<Condition-Expressions>]
[GROUP BY<Element>] [HAVING<Condition-Expressions>]
[SORT BY <Element>][ASC | DESC];
<Target Service>::= Web Service | tModel
<Service-Repository>::= Local | Universal
<Condition-Expressions>::= <Condition-Expression>AND | OR<Condition-Expression>
<Condition-Expression>::= $WS_1 \rightarrow$ name = <name-expressions> | $WS_1 —\text{I}\rightarrow WS_2$ | $WS_1 —\text{E}\rightarrow WS_2$ | $WS_1 —\text{P}\rightarrow WS_2$ | $WS_1 —\text{R}\rightarrow WS_2$ | $WS_1 —\backslash\rightarrow WS_2$ | $sd(WS_1, WS_2) > \beta$

The Service-Repository contains either local registered services or universal registered services. The Condition-Expression is a Boolean expression about the source name, the multi-valued specialization relationships, and the similarity degree between the source and the target. A *get-statement* can be embedded in another GOL statement by replacing the *where* condition-expressions with another *get-statement*.

For example, the statement of getting the target services similar to the source service named "UDDI Business Registry inquiry", can be expressed as follows, where the target is the *identical-specialization* relationship of the source and the similarity degree between them is bigger than 0.5.

GET Web Service
FROM Local
WHERE $WS_1 \rightarrow name =$ "UDDI Business Registry inquiry" and $WS_1 —\text{I}\rightarrow WS_2$ and $sd(WS_1, WS_2) > 0.5$;

5.3. Interface of service retrieval in the service grid

The proposed flexible service retrieval approach has been implemented and the prototype is available at http://kg.ict.ac.cn. An interface of the Service Grid is shown in Fig. 2, where the service operations, such as "*Get*", "*Put*", "*Browse*", etc., are arranged at the up portion. The scalable classification-type hierarchy, which includes *NAICS*1997, *SIC*1987, and *VS Web Service Search Categorization* (www.uddi.org), is arranged on the left portion of each operation interface. A two-dimensional service space is displayed on the central portion, where the vertical dimension represents the category information in correspondence with the selected classification-type and the horizontal dimension represents the content of the services. Users' operation requirements are arranged at the low portion. The retrieval process carries out as follows:

(1) Fill in the source name. User can also get assistance to find the source name by clicking the "*Find Source*" link.
(2) Choose the expected specialization relationship between the source and the target.

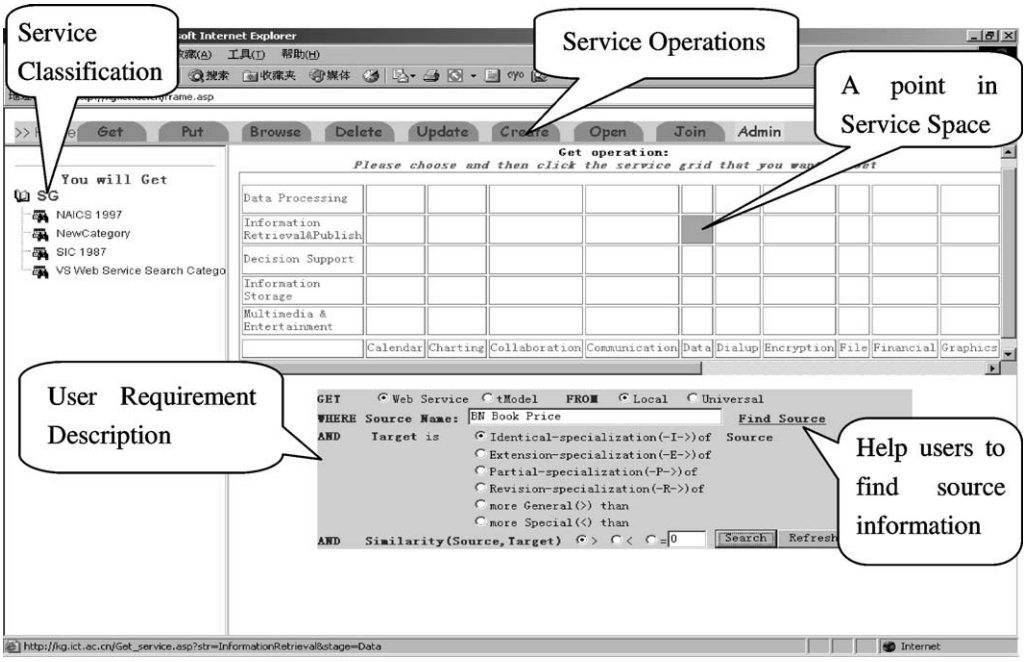

Fig. 2. An interface of the Service Grid.

(3) Fill in the expected similarity degree between the source and the target.
(4) Click the button "*Search*" to carry out the retrieval.

The Service Grid retrieves the required services/*tModels* in the repositories and returns a list to users. For example, the list of the target services that are "*Identical*" *specialization* of the source named "BN Book Price" is displayed in Fig. 3.

Since users usually cannot remember the name of the source services or *tModels*, the "*Find Source*" function can help them search in the service (*tModel*) repository by the interface as shown in Fig. 4. The "*Find Source*" process carries out either by name or by keywords conjoined by "and", "or" and "not". Fig. 5 shows the result interface of candidate services satisfying users' requirements. Users select one of the services from the list and then click the button "*Ok*" to shift the screen to Fig. 2, and the information of the selected item will be automatically filled in the source name field.

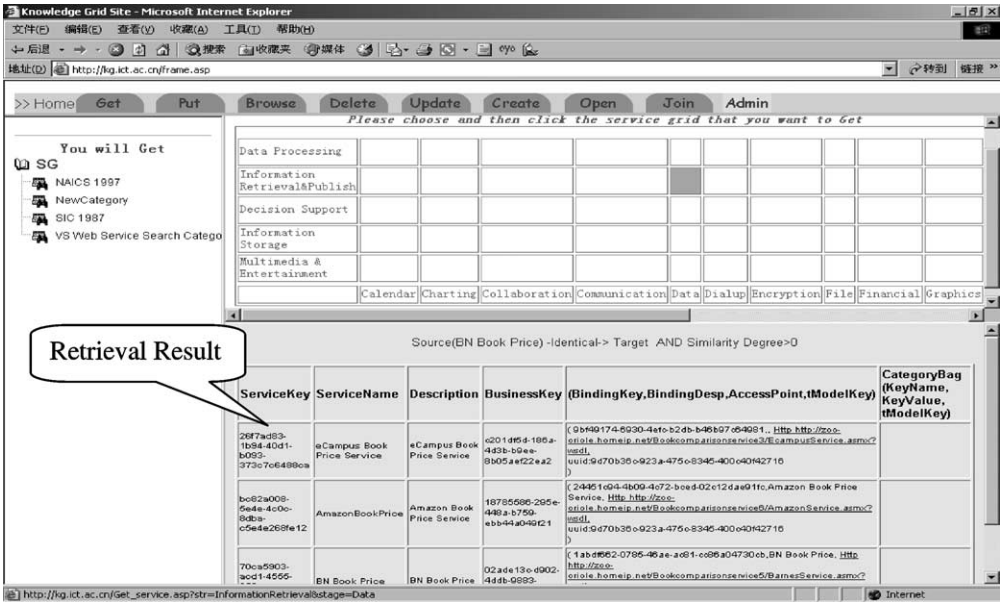

Fig. 3. Interface of retrieval result.

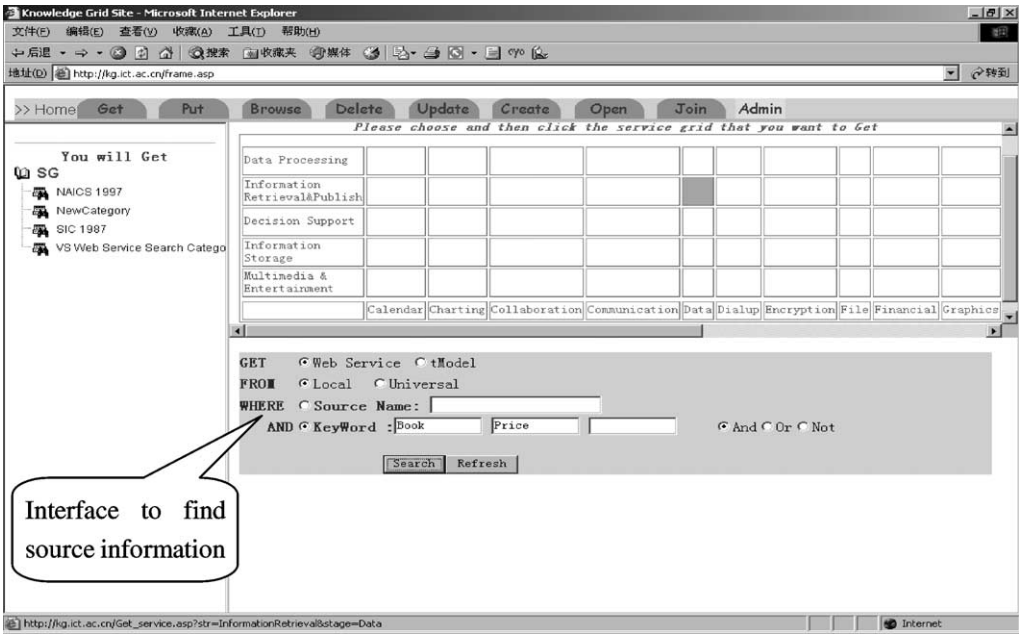

Fig. 4. General retrieval interface.

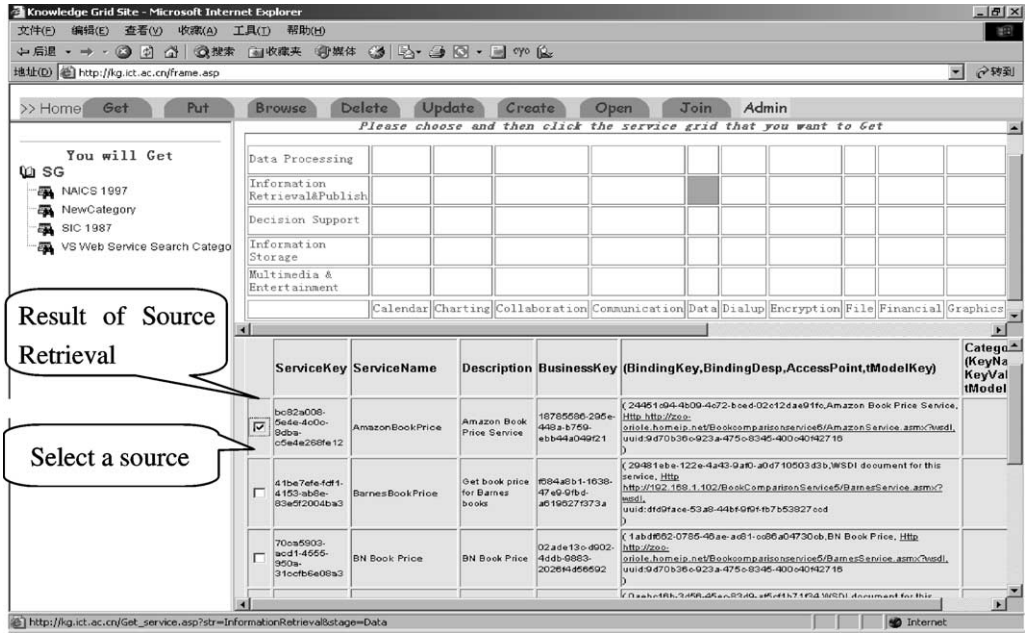

Fig. 5. Interface of displaying and selecting sources.

6. Discussion and comparison

We show the advantages of the proposed approach (named FRWS, Flexible Retrieval of Web Services) by comparing it with the current UDDI retrieval approach in ''Description'', ''Discovery'' and ''Integration'' aspects, which are the key notions in Web Services.

About the ''Description'' feature of Web Services, the proposed approach establishes a kind of semantic relationship between Web Services by means of multi-valued specialization relationships and the similarity measurement between services and between *tModels*. Reuse-association rules are derived from the semantic relationship to support flexible Web Service retrieval. So the proposed approach provides semantic information to support service retrieval, e.g., it

Table 1
Comparisons between FRWS and current UDDI

Category	Feature Items	FRWS	Current UDDI
Description	Support service specialization*	Yes	No
	Support tModel specialization*	Yes	No
	Support operation specialization*	Yes	No
	Support similarity measurement between services*	Yes	No
	Support similarity measurement between tModels*	Yes	No
Discovery	UDDI repository-based	Yes	Yes
	Service Grid support*	Yes	No
	Support keyword-based retrieval	Yes	Yes
	Retrieve services from multiple UDDI repositories*	Yes	No
	Heuristic rules support*	Yes	No
	Visualized service space*	Yes	No
	SOAP message invocation	Yes	Yes
	Support SQL-like retrieval command*	Yes	No
Integration	WSDL Support	Yes	Yes
	Semantic relationships support*	Yes	No
Others	Operating objects	Services & tModels	Services & tModels
	Universal service view	Yes	Yes

supports referential retrieval, i.e., retrieve services with reference of known services. By comparison, the current Web Service retrieval approaches only describe the data structure and interoperation interface of services.

As for the "Discovery" feature of Web Services, the current Web Service retrieval approaches retrieve services via UDDI registration center by two ways: keyword-based user interface and SOAP message invocation. By comparison, the proposed retrieval approach carries out through the media of Service Grid, which uniformly organizes services of multiple UDDI repositories in an orthogonal service space. And the Service/tModel specialization graphs further reduce the retrieval scope. So the proposed approach enables more accurate and efficient service retrieval. The proposed approach provides the following three retrieval interfaces: (1) *Graphical user interface*, which takes the advantages of the visualized service space and the SQL-like query template. Users can easily find the source services and the target services by using the interface. (2) *Service Grid supported SOAP message invocation*. Applications can use the services defined in the Service Grid, which can further invoke services through SOAP message. (3) *SQL-like query language GOL and the supported programming environment*. The GOL borrows the syntax and semantics from the standard SQL language, and implements the service retrieval on the basis of the XML query language (Bonifati and Ceri, 2000; Buneman et al., 2002). Application developers can directly operate services in the Service Grid through a set of GOL commands.

As for the "Integration" feature of Web Services, the current UDDI standard describes the interoperation interface of services by using the WSDL. The proposed approach mainly focuses on service retrieval, and it does not change the integration basis of the UDDI. But, the features in the "description" and "discovery" aspects enable the proposed approach to have the higher "recall" (Han and Kamber, 2000) in service retrieval result so that more relevant services are available to support the service integration.

Table 1 summarizes the comparison of the two approaches in the above three aspects, where the "*" denotes the distinguished features of the proposed approach. The major common points are that both the retrieval approaches comply with the UDDI standard, and support the keyword-based retrieval, the SOAP message invocation and the universal service view.

From the above analysis, we can see that the proposed approach is a kind of semantic-based retrieval approach. The semantics of the target service is expressed by using relevant services. Integrated with the Service Grid, the proposed approach enables users to retrieve services more conveniently, efficiently and effectively.

7. Conclusions

This paper proposes a flexible service retrieval approach to solve the issues of convenience, efficiency and accuracy in large-scale Web Service retrieval. Compared with the current approaches, the advantages of the proposed approach include three aspects. (1) *Convenience*. On one hand, the visualized service space helps users to focus their intention. On

the other hand, users can flexibly describe their retrieval destination by choosing the specialization relationships and determining the similarity degree between the source and the target. (2) *Efficiency*. The efficiency is obtained by two ways of limiting the search space: the multi-dimensional service space and the multi-type Service/tModel specialization relationships. (3) *Effectiveness*. Besides the keyword-based approach, the proposed approach also enables retrieval based on the specialization relationships and similarity measurement. So users can get more related services that cannot be obtained by just using the keyword-based approach.

Ongoing work is to apply the proposed approach to service integration and real applications, and realize active service mechanism (Zhuge, 2002d).

Acknowledgement

The research work was supported by the National Science Foundation of China (NSFC).

References

Bonifati, A., Ceri, S., 2000. Comparative analysis of five XML query languages. SIGMOD Record 29 (1), 68–79.
Bosworth, A., 2001. Developing Web services. In: Proceedings of the 17th International Conference on Data Engineering (ICDE'01).
Buneman, P. et al., 2002. Keys for XML. Computer Networks. 39. 473–487, http://www.elsevier.com/locate/comnet.
Cilia, M., Buchmann, A.P., 2000. An active functionality service for E-business applications. SIGMOD Record 31 (1), 24–30.
Han, J., Kamber, M., 2000. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, Los Altos, CA.
IBM, 2001. Web Services and UDDI. http://www.ibm.com/services/uddi.
Keith, L., Ali, A., 2002. A goal-driven approach to enterprise component identification and specification. Communications of the ACM 45 (10), 45–52.
McIlraith, A., Tran, C., Honglei, Z., 2001. Semantic Web services. IEEE Intelligence Systems 16 (2), 46–53.
Michael, S., 2002. Web services beyond component-based computing. Communication of the ACM 45 (10), 71–76.
Microsoft, 2001. Microsoft advanced UDDI search. http://uddi.microsoft.com/search.aspx.
Shirky, C., 2002. Web Services and context horizons. Computer (September), 98–100.
UDDI. Org, 2001a. UDDI Version 2.0 API specialization UDDI open draft specification.
UDDI. Org, 2001b. UDDI Version 2.0 data structure reference UDDI open draft specification.
Vaughan-Nichols, S.J., 2002. Web Services: beyond the hype. Computer (February), 18–21.
Xu, D., Nahrstedt, K., Wichadakul, D., 2001. QoS-aware discovery of wide-area distributed services. In: Proceedings of the 1st International Symposium on Cluster Computing and the Grid.
Zhang, L.J. et al., 2002. XML-based advanced UDDI search mechanism for B2B integration. In: Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002).
Zhuge, H., 1998. Inheritance rules for flexible model retrieval. Decision Support Systems 22 (4), 390–397.
Zhuge, H., 2000. A problem-oriented and rule-based component repository. Journal of Systems and Software 50, 201–208.
Zhuge, H., 2002a. A process matching approach for flexible workflow process reuse. Information and Software Technology 44 (8), 445–450.
Zhuge, H., 2002b. A knowledge grid model and platform for global knowledge sharing. Expert Systems with Applications 22 (4), 313–320.
Zhuge, H., 2002c. VEGA-KG: A way to the knowledge Web. In: Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA.
Zhuge, H., 2002d. Clustering soft-device in Semantic Grid. Computing in Science and Engineering (Nov./Dec.), 60–63.

Hai Zhuge is a full professor at the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include: knowledge management, knowledge grid and the semantic web, problem-oriented model base systems, component reuse, cognitive-based software process model, inter-operation model for group decision, and web-based workflow model. He is now the leader of the China Knowledge Grid research group (http://kg.ict.ac.cn). He is serving in the editorial board of a number of international journals. He is the author of one book and over 40 papers appeared mainly in leading international conferences and the following international journals: *Communications of the ACM*, *IEEE Intelligent Systems*, *IEEE Computing in Science and Engineering*, *IEEE Transactions on Systems, Man, and Cybernetics*; *Information and Management*; *Decision Support Systems*; *Journal of Systems and Software*; *International Journal of Cooperative Information Systems*; *Expert Systems with Applications*, *Knowledge-based Systems*; *Information and Software Technology*; and, *Lecture Notes in Computer Science*.

Jie Liu is a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.