UDDI Registry for Knowledge Discovery in Databases Services

Claudia Diamantini, Domenico Potena and **Jessica Cellini**
Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione,
Università Politecnica delle Marche - via Brecce Bianche, 60131 Ancona, Italy
{diamantini,potena,cellini}@diiga.univpm.it

Abstract

The Discovery of Knowledge from Databases (KDD) is an high complex process, where a lot of tools are needed to achieve the goal of previously unknown, potentially useful information extraction. It implies that an user has to choose, to set-up, to compose and to execute the tools more suitable to her/his knowledge discovery problem. In order to overcome this complexity, we are developing the Knowledge Discovery in Databases Virtual Mart, a support platform based on the service-oriented paradigm. In this paper we discuss how to design a core element of this platform: the UDDI service broker.

Introduction

Knowledge Discovery in Databases has been defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. 1996). As a process, in order to discovery the knowledge an user has to exploit several tools of heterogeneous types. Such tools are derived by different research area, such as database, machine learning, artificial intelligence, statistics and so on. It implies that an user must have some degree of expertises in order to face with all the needed tools. As a matter of fact, in a KDD process the user has to choose, to set-up, to compose and to execute the tools more suitable to her/his KDD problem. The complexity of the design of a KDD process is complicated in a distributed scenario, where tools are (often free) available over the Net. As a matter of fact, in this scenario, there is a lack of information about both the functionalities of tools and their technical details. Furthermore, when the tool description is available with the tool, it is not given in a standard form. Moreover, due to the continuous research activities in KDD and Data Mining fields, the amount of tools that an user can use for discovering knowledge is constantly growing. Hence, it is hoped a support to overcome the intrinsic complexity of design a KDD process.

The necessity of giving such a support to an user led us to study and to develop an open platform we called *Knowledge Discovery in Databases Virtual Mart* (KDDVM) (KDDVM site). Such a platform is based on a service-oriented architecture in order to build an open and distributed environment, where services (i.e. KDD tools) can be easily introduced, accessed, described, composed and activated (Diamantini, C., Potena, D. and Panti, M. 2005; Potena, D. and Diamantini, C. 2005).

A core element of our platform as well as any other service-oriented platform is the service broker, that is a registry and a set of APIs to browse it. The broker registry stores either the information about the available services or links to the location where the information is provided. According to the W3C (W3C site), such a registry is implemented by the Universal Description, Discovery and Integration (UDDI) standard. The UDDI allows to design a general-purpose broker, that is able to manage a minimal and generally valid set of information. Then, in order to design an effective and efficient broker service for a specific problem or domain, the UDDI standard has to be extended with domain specific information.

In the literature, there are few works focusing on UDDI extension. In (ShaikhAli, A., Rana, O.F., Al-Ali, R. and Walker, D.W. 2003) researchers have introduced another type of information, called blue pages, in addition to already existing white, yellow and green pages. Blue pages allows businesses to insert new properties associated with a service and to enable discovery of services based on these. Properties are added into blue page registry by defining name-value pairs. In particular the work focuses on properties like service leasing and quality of service. This approach is similar to one of the three approaches to UDDI extension discussed in (Liu, J., Gu, N.,Zong, Y., Ding, Z., Zhang, S. and Zhang, Q. 2005), which consists on extending the UDDI data model in order to store non-functional information like Quality of Service (QoS). Such extension allows for browsing and discovering only QoS properties but no other service property. The second approach uses tModel to refer to service properties, in this way new information can be published without modifying the UDDI data model. The last solution is to use an external database to store not standard properties of services. As well as the second approach, this solution does not modify the UDDI data model.

Our scenario is quite different and more complex compared with any other works on UDDI extension. As a matter of fact, we have to manage a domain specific information, that is structured and related each other. For this reason, the approaches presented in the literature are inadequate to our aims. So, the aim of this paper is to discuss how to extend the UDDI registry in order to manage information needed to describe a service in the KDDVM platform.

To this end, in the next section, we present the data schema and APIs of the UDDI standard. Then, in third section we show the information needed to describe a generic KDD tool. The subsequent section illustrates our approach to the design of a UDDI for KDD services. Finally some conclusions are given.

UDDI registry

The Universal Description, Discovery and Integration (UDDI site) specification is an initiative to create a global, platform-independent, open framework for Web Services publishing and discovering.

The UDDI project supplies a solution both for businesses to publish information about services they provide and for customers to search for available services. UDDI standard allows businesses to describe their business services, how services can be accessed, what information they want to make public, and what information they choose to keep private. On the other hand, UDDI provides a set of standard functionalities to discover published services. In order to support the discovery process, the UDDI provides three types of information about Web Services:

- white pages: name and contact details;
- yellow pages: categorization based on business and service types;
- green pages: technical data about services;

The UDDI specifications take advantage of World Wide Web Consortium (W3C site) and Internet Engineering Task Force (IETF site) standards such as Extensible Markup Language (XML), HTTP, Domain Name System (DNS) protocols, Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP) messaging specifications.

The UDDI data model (Figure 1) is defined by four basic entities: businessEntity (modeling business information), businessService (high level service description), tModel (describing a technical model representing a web service type, a protocol used by web services and a category system), and bindingTemplate (mapping between businessService and tModels).

businessEntity: The businessEntity is the principal entity of UDDI. It contains the core details of the business: its name, descriptions of the business, details of personal contacts within the business, identifiers used to refer to the business (identifierBag), categorical classifications for the business (categoryBag), and a list of services provided by the business. Name and contact details define the white pages information of the UDDI registry. A businessEntity contains one ore more instances of a businessService.

businessService: Each businessService describes a single service provided by an instance of a business entity. A

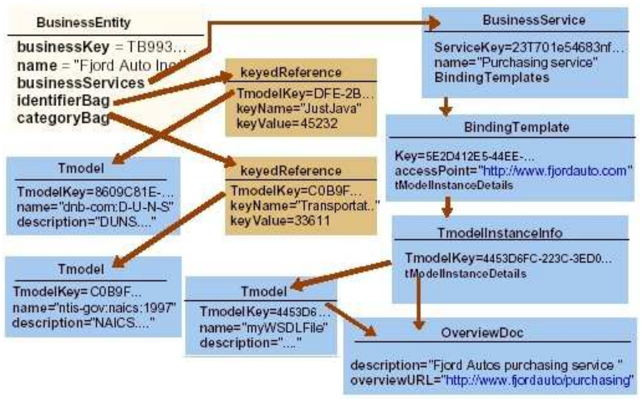

Fig. 1: UDDI data model.

service is characterized by a name, a description, a classification of the service type (category bag), and technical information (described by one or more binding templates). Information about categorizations of both businessEntity and businessService allows to define the yellow pages registry.

bindingTemplate: Each businessService refers to one or more binding templates mainly representing the technical information about the access point (the URL) of the service. Each bindingTemplate references to one or more instances of tModel, describing the technical information about how to access the service. BindingTemplate information is typically retrieved by an organization interested in using the related businessService. This particular information can then be used for dynamic invocation at runtime. Together the bindingTemplate and the tModel define the green pages (or technical information) about services.

tModel: A tModel, or technical model, represents two concepts within UDDI: a technical specification for a given service type, or a model for a particular identifier or taxonomy. For examples tModels can refer to WSDL files, XML DTDs or schemas, and classification schemes. A unique tModel describing a service type can be addressed by different services. A tModel gives information about its name, the name of the organization that publishes it, a list of categories identifying the service type, and pointers to technical specifications (e.g. interface definitions, message formats, and message and security protocols).

Within a UDDI registry, each core entity is assigned a unique identifier according to a standard scheme, called the UDDI key.

In businessEntity as well as in businessService, the categoryBag tag is used to provide a classification of services (yellow page). Inclusion of classifications is optional, but greatly enhances search behavior of the broker.

The UDDI specifications also include definitions for web services interfaces in order to allow programmatic access to the registry information (APIs). These are divided into two logical parts that are the inquiry API and the publisher's API.

The inquiry API provides access to UDDI registries to extract both all the instances belonging to a same entity and all the details of instances meeting the specified criteria. The publisher's API consists of both save call and delete call for each entity of registry.

KDD Web Services

The KDD process includes several steps, that implies the interaction of various tools derived from different domains. The heterogeneity of tools as well as the goal-driven and domain dependent nature of any KDD problem introduce complexity in the design of a KDD process. Such a complexity is mainly due both to the huge amount of tools the user can choose and to the expertise needed facing various KDD tasks. Assuming a repository of KDD tools is available to the user, he/she has to pass through different activities in order to manage his/her KDD processes, he/she has:

- to browse the tool repository and to obtain information about the tools;
- to choose the more suitable tools on the basis of a number of characteristics: tools performances (complexity, scalability, accuracy), the kind of data tools can be used for (textual/symbolic data, numeric, structured data, sequences, ...), the kind of goal tools are written for (data cleaning, data mining, visualization, ...) the kind of data mining task (classification, rule induction, ...);
- to prepare data conforming to the tool input format;
- to manage the execution of the tool, in particular to properly set tool parameters for the data and problem at hand;
- to design the KDD process by tool composition.

In order to give support to an user in facing with all these activities, we are developing an extensible service oriented Knowledge Discovery support system called *Knowledge Discovery in Databases Virtual Mart*(KDDVM) (Diamantini, C., Potena, D. and Panti, M. 2005).

Such system is formed by three kinds of services: basic, support and advanced services. The former includes tools implementing data mining algorithms as well as any other data manipulation algorithm. Advanced services provide an "intelligent" support to the user. In particular, they are mainly devoted to proactively help users in the choice of tools most suitable to her/his goals. Finally, the core layer of KDDVM contains support services, that mainly give support to browse and discovery, to compose and to activate tools.

Describing KDD services

We divided the information needed to accomplish the above described tasks into four main parts: information in order to discover the needed tools, information in order to physically locate and to execute the tool, description of the tool I/O interface, KDD tools that can be linked up with the described tool.

In particular, the first and last parts play a leading role in KDDVM. As a matter of fact, in order to give support to users with different degrees of expertise and knowledge, we give a description of functionalities of any KDD tools, according to both a common vocabulary and a categorization of the KDD tool with respect to a predefined KDD taxonomy (Potena, D. and Diamantini, C. 2005). Such a structure is an extension of the Data Mining taxonomy DAMON (Cannataro, M. 2003), which covers the other KDD tasks, like data cleaning, data transformation, data selection, visualization and so on. A taxonomical structure is a natural way to characterize the KDD tools domain in terms of the implemented task. The goal of each KDD task can be achieved by various and different methods, e.g. for the classification task an user can use a Decision Tree, a Neural Network, a Fuzzy Set Approach as well as a Genetic Algorithm. For each method are available a lot of algorithms, e.g. C4.5, ID3 and CART are Decision Tree algorithms. Finally any KDD tool is a specific implementation of a given algorithm. For example, let us consider the tool BVQ_train and its taxonomical description: ***task***←*classification*, ***method***←*Vector Quantizer* and ***algorithm***←*Bayes Vector Quantizer*. Such information means that the tool is an implementation of the Bayes Vector Quantizer(BVQ) algorithm (Diamantini, C. and Spalvieri, A. 1998), a Data Mining algorithm that performs the classification task by a particular vector quantizer method.

Furthermore, in order to achieve a KDD goal, the exploitation of a single tool is often not sufficient, rather a KDD process is typically composed of many different tools, such that the output of a tool represents the input to another one. Then, in order to support the user in such a tool composition activity, the description of a tool should be enriched with information about KDD tools that can be linked up with the described one (we named such kind of information *linkable modules*).

Extension of UDDI for KDD domain

In this section, we show in how UDDI can be integrated into KDDVM platform in order to provide browsing and discovering of KDD services. To this end, we studied how information about KDD tools can be stored in a UDDI schema in order to introduce specific functionalities for KDD services.

The UDDI standard defines two ways to add new information into registries. One possibility is to define a tModel in order to address, by the *overviewDoc* field (see Fig.1), the WSDL description. In this way, WSDL and UDDI work together for web services advertisement. As a matter of fact, a WSDL document defines how to invoke a service. It provides information on the data being exchanged, the sequence of messages for an operation, the location of the service and the description of bindings (e.g. SOAP or HTTP). The other way is to use a categoryBag to classify services w.r.t. their functionalities. A categoryBag is a collection of keyedReference structures. Each keyedReference provides a name-value pair, that assumes values in a particular domain described by the related tModel (UDDI site).

In our aim, each of the presented solutions can not be used by itself. Indeed, the WSDL is insufficient to describe all information needed to describe KDD tools, while the second solution introduces some limits about classification of structured data. As a matter of fact, it is not possible represent structured relationships, like both "taxonomy" and "linkable modules", in a categoryBag. The above analysis of the standard led us to study a specific approach in order to overcame UDDI limits.

In order to design the UDDI for KDDVM platform, we adopt different solutions for each part of the information related to a KDD tool (see the previous section). Information about tool location and execution are well described by WSDL, then we use tModel in order to address the WSDL description of the tool. The description of the I/O interface contains useful elements for supporting user in the tool activation as well as in the tools composition, e.g. I/O description, default value and validity range. The standard WSDL is not able to describe such domain specific information, then we need to add new tags into the WSDL, defining an extension of WSDL(eWSDL) for the KDDVM platform (Potena, D. and Diamantini, C. 2005). Several examples of eWSDL description are available at (KDDVM site).

As regards the other two parts, i.e. KDD taxonomy and linkable modules, they are information related to KDD domain, that are not included into WSDL standard. Taxonomy defines algorithm classification and linkable modules defines algorithm that can be linked up with the described one. These information can be represented as concept of an ontology of the KDD domain, that has the algorithm as core concept. Such an ontology is exploited not only by the broker for discovering services, but even by some advanced services in order to give a proactive support to the user in the choice of the most suitable tools w.r.t. her/his KDD problem. For this reason, the information about taxonomy and linkable modules of any KDD tools, that are registered in the broker, should be available both locally to the platform and at the same time. Thus, we decided to introduce a database, external to the UDDI, in order to manage such information, rather than include it into the eWSDL description of each KDD service, that is usually located at tool's provider. In Fig. 2 we show the overview of our approach.

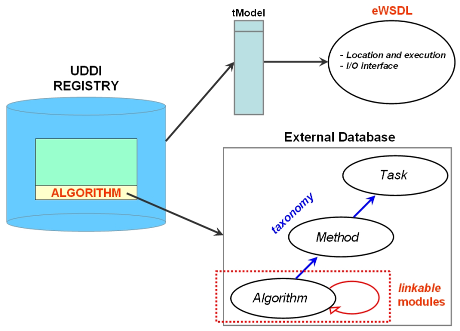

Fig. 2: Overview of our solution.

The proposed solution has the advantage of adding domain specific information into the broker registries, without modifying the UDDI data model. Furthermore, in (Liu et alii, 2005) authors proved that the use of an external database allows to store additional information related to service relationships and constraints with a high discovery efficiency. The use of a tModel to address an extended WSDL involves some problems in the design a general-purpose broker: any providers must use the same property definition schema and should extend the discovery interface (i.e. the APIs). These disadvantages decay when we design a domain specific broker working on a specific platform. As a matter of fact, the KDDVM broker points out to KDD services provided by KDDVM platforms, where the KDD services are already described using the eWSDL. Since the external files are distributed, the problem related to the use of tModel is the discovery efficiency, that decreases when the number of different KDD service providers increases.

Conclusions

In this paper we presented a study about the UDDI standard extension in order to manage the information related to a generic KDD service. Such a solution is based on the combined exploitation of tModel and an external database. TModel is used to address the service descriptor, usually available at the service provider. Such descriptor contains information about location, execution and I/O interface of a tool. The external database stores both information needed to categorize the tool with respect to the KDD domain, and information on tools the can be combined with the described one. This work is part of the Knowledge Discovery in Databases Virtual Mart project, a more general project for the development of an open and extensible environment where users can look for implementations, suggestions, evaluations, examples of use of tools implemented as services. At present we are developing a service broker for KDDVM that is based on the solution presented in this paper.

References

Cannataro, M. 2003. Knowledge Discovery and Ontology-based services on the Grid. In *The First GGF Semantic Grid Workshop*.

Diamantini, C. and Spalvieri, A. 1998. Quantizing for Minimum Average Misclassification Risk. *IEEE Trans. on Neural Networks* 9(1):174–182.

Diamantini, C., Potena, D. and Panti, M. 2005. Developing an Open Knowledge Discovery Support System for a Network Environment. In *Proc. of the 2005 Int. Symposium on Collaborative Technologies and Systems*, 274–281.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. 1996. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.

IETF site. http://www.ietf.org.

KDDVM site. http://babbage.diiga.univpm.it:8080/axis.

Liu, J., Gu, N.,Zong, Y., Ding, Z., Zhang, S. and Zhang, Q. 2005. Service Registration and Discovery in a Domain-Oriented UDDI Registry. In *Proc. of the 2005 the 5th Int. Conf. on Computer and Information Technology*. IEEE.

Potena, D. and Diamantini, C. 2005. Description of Knowledge Discovery Tools in KDTML. In *Proc. of Computer and Their Applications track of the 1st Int. conf. CCSP 2005*, 144–248. Kuala Lumpur, Malaysia: IEEE.

ShaikhAli, A., Rana, O.F., Al-Ali, R. and Walker, D.W. 2003. UDDIe: an extended registry for Web services. In *Applications and the Internet Workshops*, 85–89.

UDDI site. http://www.uddi.org.

W3C site. http://www.w3c.org.