**Abstract.** Two of the factors that lead the continuous growth of the internet are the evolution of the available products, from a collection of pages toward a collection of services that interoperate through the Internet, and the increasing popularity of heterogeneous computer receiving devices, such as palmtops and handheld computers. Thus, interoperability in Internet environments becomes the main technological concern. Standards specification constitutes the foundation of the web service community in which Universal Description, Discovery and Integration (UDDI) [1] is one of the members. Besides, a standard UDDI registry does not make use of semantic information because it is agnostic in relation to the web services functionality and, by itself, cannot provide automatic quick responses in locating web services on the bases of their functionality. Under the scope of this paper is the study of using of ontology in Geographic Information Systems (GIS). A new conceptual domain level is defined upon GIS shared knowledge and data, increasing the “intelligence” of GIS networks. In this paper is proposed a service oriented architecture involving the server components and client applications. In detail, we present the design and implementation of an enhanced UDDI server that provides the publishing and discovery of rich web service profiles based on context-aware semantic information.

**Keywords:** Ontology-Driven GIS, Service-Oriented Architecture (SOA), Web Services, UDDI enhancement, Application Integration.

**1. Introduction**

The standards that support GIS help people to interact with data with geographical meaning through the definition of common spatial and visual languages. To a broad community, where GIS information is distributed from several sources at the same time, the GIS networks must provide multi-participant and collaborative infrastructures to the GIS information services, so organizations can openly share and direct use the GIS information. This approach changes the GIS’s architecture paradigms from private network configurations to architectures that can provide and consume geographic information services through open Internet protocols, which give support to a wide variety of areas of information technologies [2].

Initially, GIS offered a system with digital methods for integrating, analyzing, and visualizing geographical information. Later, GIS emerged as a powerful technology for coordinate and integrate the work of an organization in communion with a common and shared geographic database. Nowadays, GIS becomes a global architecture, allowing communities of organizations to connect and share their geographic information and applications as network-based services to one
another. The interest of defining a GIS conceptual domain is the consequent advantage to use specific vocabulary that can add contextual meaning to the different GIS information services. The identification and functional description of each service forms the information model of the semantic layer inserted between the network and application levels.

The GIS communities benefit from the intelligence of a semantic-based mechanism for service location. The knowledge about the services available on a GIS application domain gives a faster response time on service location requests. On the other hand, the growing number of geospatial service providers and the need for their accurate identification lead to more refined semantics. Ontology is a way to share knowledge and data. It uses a particular vocabulary to, but not restricted to, describe entities and their functions identified in a consensual picture of a small shared world [3]. More precisely, ontology is a formal explicit description of concepts, sometime called classes, in a domain. In terms of development, object-oriented programming techniques concentrate on the operational properties of a class, whereas an ontology designer focuses on the structural proprieties of a class [4]. The translation of ontologies, using object-oriented approaches, into the directory system of a given application domain, leads to an ontology-based directory system.

The first step in building next generation GIS is the selection of applicable ontologies or the specification of a new ontology using some ontology editor. The result of this process is a systematic collection and specification of geographic entities, their properties and relations written in a formal language. The ontology editor can translate into sets of concepts the relations of the real world entities. The ontologies should be available for browsing by users as they provide metadata about the available geographic information services.

The contribution of this paper is to show how the use of contextual information can define more accurately the GIS participant services and introduce automatic control flow tools for publishing and discover them. The proposed approach follows a service-oriented architecture design style and implements an extended UDDI server platform called UDDI2M that stands for «UDDI to MESDI Middleware» [5]. The client-side application is an integration solution that binds the geographical and ontology information systems together, making use of white-box parts of the each framework. The chosen platforms are, respectively, the JUMP Unified Mapping Platform (JUMP) [6] and the Protégé knowledge-based framework [7].

The remainder of this paper is organized as follows. Section 2 narrows down the problem of finding Web services and gives an overview of the technologies chosen to add the new functionalities specified for the overall system. Section 3 discusses the architecture designs and implementation issues. Section 4 presents the conclusions and future work.

2. Discovering Web services

By gathering information about the user preferences, the service providers can advertise their services according to the user’s context. The problem that service-oriented system faces is the lack of mechanisms for context-based service discovery. The question raising is how we can define a service profile, enriched with contextual information, which can help on the creation, publication and discovery of such service profiles [8].

To accomplish this desire, the UDDI technology for service description and discovery, must be used with another tool that can provide baseline semantic information edition. This kind of information, which we refer as ontology, works as context keys that the service providers can use to advertise their published services and users to reach them. Transparent web service discovery is the critical issue. The search should be well-defined in terms of interface remote access, and the descriptive information used to publish and inquiry the directory service
should be common in the context of an application domain. The traditional UDDI registries allow simplified searching and trading patterns to publish data about information providers and their advertised web services for classification. Also, the UDDI registries enable the location of web services whose identifiers are well known so users can find them electronically. Furthermore, additional semantic information about web services gives an important contribute to accurate the web service’s identification under an application domain.

The architecture implementation challenges described on this paper had two development lines: an extended version of a UDDI server, the UDDI2M catalogue, and the integration of the application platforms that will interact in real world use-case scenarios. For example, a map viewer client, like JUMP, can put into screen geo-referenced data as images or comprising layers from distinct servers. The user can enrich his client visual content selecting a bounding box, in units of a particular Spatial Reference System that represent the geographic extent of the map, by querying the catalogue UDDI2M for advertised services, according to the adopted ontology, to get available service profiles. An application wizard can help user on this steps and it finally parse the service profile to get the information service access points. Then, the map viewer client performs a service request to the selected access point, gets the Geographic Markup Language (GML) descriptor of the available geo-information and adds a new layer to the visualization area limited by the bounding box previously stated.

The figure 1 identifies the participants in the GIS architecture and the controls messages exchanged between them. The client with ontology edition features publishes an ontology instantiation in the UDDI2M catalogue. Another client with geo-data visualization mechanisms asks the catalogue for geographic information services classified by Web Map Service (WMS) or Web Feature Service (WFS), for example. After obtain the web service access points, the map viewer client can interrogate them to retrieve the geo-referenced information.

3. Architectural aspects

GIS Architectures are open, collaborative, and multi-participant systems that allow users publish, share, and use each other’s services. Technologically, they rely on SOA architectural solutions based on open and “loosely coupled” standards of the Internet. Open and modular characteristics allow subsystems to be independently created and deployed in a distributed and “loosely coupled” Internet environment using standard XML protocols. The directory services consist on standard-based metadata catalogue services. The application software available is open, interoperable, and technologically compliant with the whole range of IT standards [9].

The complexity of the actual software systems is a consequence of the broader bandwidth needs that the system is supposed to cover. The effort to make software systems more generic and easier to use is based on the possibility to analyze similar use-cases and capture the invariant parts. Doing this, we may iteratively crystallize and reutilize the
design artifacts that, with some implemented
classes, will constitute an extensible software
system.
At run-time, collaborating systems are
dependent on the functionalities of each other,
which mean that, at some compile time, the
interfaces must be known. If the interfaces of
a software system are not public or not
compatible with the community adopted
technologies and business contracts, it’s
straightforward to use interface adapters to
provide compatibility and interoperability
between dependent modules. These kinds of
dependencies are purely technological and
can be reduced choosing the correct standards
to implement remote procedure call
mechanisms between software systems. To
achieve the loose coupling goal, it should be
possible to develop a software module
without compile-time dependencies. The
vector guidelines to fulfill this objective are
the use of descriptive interfaces and Internet-
based protocols.
SOA is an architectural style that achieves
loose coupling among interacting software
systems agents by setting a small set of
simple and ubiquitous interfaces to all
participating software agents. Only generic
semantics are encoded at the interfaces. The
interfaces are may be universally available for
all providers and consumers. The interfaces
deliver only descriptive messages constrained
by an extensible schema. The messages don’t
carry out any system behavior, and the
schema limits the vocabulary and structure of
the messages. New versions of services can
be introduced, without breaking existing
services, with extensions to the schema.
Application-specific semantic is passed
through generic interfaces. Under the service-
oriented development paradigm, each service
facet may have a single interface and different
messages can be sent over it, but the messages
are descriptive, rather than instructive, and the
request decoding, along with formal
constraints checking like format, structure and
vocabulary, it’s a service responsibility.
The distributing model, on each GIS server is
deployed, consists on a number of web
services that need to maintain local
information about the rest of the collaborating
services. Thus, among the SOA architecture
must exist a mechanism for service discovery
that enable consumers to find service
providers in some applicational context.
Catalogue services are the key technology for
locating, managing and maintaining
distributed geo-resources like geospatial data
repositories and geographic information
services. Catalogue services with standardized
interfaces and operations and, based on a
well-known information model including
descriptive (thematic) metadata, enables map
viewer client applications to search for geo-
resources in very efficient ways.
After the interfaces and operations of a
catalogue service are well defined, it is left up
to the developer of the overall system to
define a specific information model that the
catalogue service shares in its contract. This
includes, but is not limited to, supported result
sets or matching algorithm selection. This
information model, that could be a XML
Schema artefact, could be important when
interoperability between catalogue service
instances is required.
The UDDI2M catalogue service is a flexible
and centralized catalogue that plays the role
do/directory service in the service-oriented
architecture proposed. The functionality of
the UDDI2M catalogue is to provide the
translation between from generic metadata
items to a Universal Resource Identifier
(URI) indicating the service access point. The
extended feature, present on UDDI2M
subsystem architecture, relative to the
standard UDDI servers, is the ability to
interrogate the UDDI server selecting
ontology concepts, defined for the specific
application domain, or user-defined metadata,
along with correspondent matching
algorithms, as rich query elements. The
UDDI2M catalogue defines in its contract a
XML Schema for the messages format
containing a description of the method name
and its parameters, the semantic model and
the web service profile structure.
3.1. Enhanced UDDI Server and Application integration issues

The strategic approach to make up this proposed architecture was application integration, both on functional and information levels, supporting their ability to exchange information and leverage processes in real time. In client application frameworks, the interaction between the user and the different application functionalities in real-time is almost a pure technological play.

In GIS, the client tools, such as map viewers, ontology editors and catalogue graphical managers can be integrated with the objective in mind that the integration product could add some control automatisms to some well identified process flows, like a catalogue inquiry for an information service using some ontology concept.

A clear trend is the movement away from information-oriented to service-based integration. Information-oriented integration provides an inexpensive mechanism to integrate applications because, in most instances, there is no need to change the applications. While information-oriented integration provides a functional solution for many application integration problem domains, it is the integration of both application services and application methods that generally provide more value in the long run [10].

Service-based application integration is not a new approach. Distributed object platforms, like Java Remote Method Invocation (RMI) or Common Object Request Broker (CORBA), are well-known mechanisms to bind applications together at the service level, but they imply tight coupling and, consequently, compile-time dependencies. However, the new notion of Web services is to reduce the non-functional dependencies as we attempt to identify a new mechanism that is better able to leverage the power of the Internet to provide access to remote application services through a well-defined interface and directory service, i.e., UDDI.

The uses for this type of integration are endless, including the creation of composite applications, or applications that aggregate the processes and information of many applications. For example, using this paradigm, application developers simply need to create the interface and add the application services by binding the interface to as many Internet-connected application services as are required.

The new functionality object of this paper is the integration of two client application frameworks: an extensible framework for custom spatial data processing called JUMP, and framework for ontology edition called Protégé. A parallel development line is the enhanced UDDI catalogue service UDDI2M providing both application programming interface (API) and graphical user interface (GUI) for remote registry administration.

Both frameworks have white-box parts formed by abstract classes that can be inherited to build plug-in components. These components are then referenced in the configuration points of each framework. This method of developing large-scale components on top of frameworks can turn much easier the addition of new functionalities to an application because part of the mechanisms rely on the framework itself.

The JUMP framework is constituted by a set of user interface graphical classes including the implementation of mechanisms to present rendered geo-referenced data in screen. It also defines an extensible set of classes for generic plug-in development.

The Protégé ontology framework is divided in two main configuration points: storage backends that let the user to write his/her ontology in different formats; user interface components that can show different types of editors, or any GUI container, plus format transformation operations. The solution to accomplish application integration was creation of an “Ontology Editor” plug-in in the JUMP framework and a plug-in for the UDDI2M catalogue management GUI running on the Protégé platform.

The next figure illustrates the use of the frameworks hot-spots.
The system architecture presented fulfills the SOA guidelines but it also can be seen in logical terms where the traditional 3-tier model describes the functional level of each architecture component. The database logic level is performed on the jUDDI server which is an implementation of the UDDI 2.0 standard. The business logic layer is defined and implemented in the UDDI2M catalogue service, which pretends to provide a higher level of abstraction of the operations that can be made on physical level. Simpler functions are given to the catalogue users like ontology retrieval, publishing or discover of rich content service profiles. Further on this paper, we will detail the composition of each functional packages of the UDDI2M catalogue service. The user interface and applicational layer is taken cared by the JUMP enhanced client, product of the application integration.

The communication between layers is made using Simple Access Object Protocol (SOAP) [11], in which messages are exchanged between the UDDI2M thin-client and web adapter, as they are specified in the UDDI2M XML schema. This schema defines the service profile structure, adding some extended features comparing to the standard UDDI information model, and defines the interface operations that the UDDI2M catalogue service exposes, specifying the type of message, arguments and return values.

The following figure presents the logical architecture, the target analysis of this paper.

**3.2. Realization**

The application frameworks where chosen to be free, based on java, open-source projects and extensible in the sense that they should support component plug-in development by inheriting abstract classes, or directly use default implementations, for that purpose. These classes define statement models, where is represented their inner behavior, in such a way that the framework can execute transparently the component code.

The JUMP platform provides both main UI and application programming interface (API) and represents a highly extensible framework for developing and running custom spatial data processing applications. Through the plug-in mechanism of the JUMP platform, a visual component with the Protégé ontology editor is selectable on the JUMP workbench. Making possible the ontology edition on the client side obligates the application to provide a mean to upload the ontology project into the UDDI2M catalogue, so other clients can browse the ontology on their own stations. To accomplish this, a plug-in component was built over the Protégé framework based on the UDDI registry client included on the Sun’s Java Web Services Development Pack (JWSDP). The integration process achieved to introduce some automatic flow control in terms of creation and publishing the ontology that all the system components will have.
shared access. The UDDI2M information model contains definitions of the ontology concepts, the UDDI page classes, along with generic metadata fields, all collected to make part of the service profile descriptor. The aggregated knowledge in the service profile is the key to enable a rich and unified discovery mechanism. The UDDI2M catalogue implements business logic, defined for GIS to extend the standard UDDI API, and exposes more generic interfaces that work upon the service profiles and ontology XML documents. The standard UDDI server chosen was the Apache project jUDDI and the interface with it is made by UDDI4J, which is an IBM library to access UDDI servers. The UDDI4J is a Java implementation of the client side of UDDI with everything that the UDDI2M catalogue needs to publish, find, and bind a service profile.

Both UDDI2M and jUDDI services have a web interface accessed through Axis, an Apache implementation of SOAP, and they deployed on the JBoss application server, an open source implementation of the Java 2 Platform, Enterprise Edition (J2EE) 1.4 specification.

3.3. Evaluation

The development of the UDDI2M catalogue was carried out toward Agile approaches [12] and some design pattern instantiations. The software project was written on the Eclipse platform and packaged according the UDDI2M logic functionalities, and also includes a unit test battery written over JUnit. Next we will briefly describe each module present on the UDDI2M catalogue and a correspondent generic use-case scenario.

The OntologyLoader module implements a single method that receives an ontology instance, which we call taxonomy or knowledge base, in a XML document and creates the UDDI registry structures necessary to afterwards reconstitute the ontology relations. These relations between ontology concepts are taxonomic, in the sense that they support the hierarchical class structure by establishing “kind-of” associations between classes.

The Protégé XML plug-in is used to transform the taxonomy from the client knowledge-base format (standard text files, OWL, etc.) to XML, and this document is then published through the OntologyLoader facet. This operation is made on the system setup, since all participating services must know each other’s identification. The operations of advertise and query services will may use ontology instances values present in the knowledge base.

The Visitors module instantiates the Visitor design pattern present on the Gang of Four (GoF) design pattern catalogue. It represents operations to be performed on the elements of the UDDI2M object structure. The Visitors class model allows the addition of new operations without changing the class definitions of the elements on which they operate. The creation and deletion operations of UDDI page objects that make part of UDDI2M data model, like BusinessEntities, BusinessServices and ServiceBindings are invoked by the instances of this objects but defined and implemented on separate visitor instances. The UDDI2M information model classes accept a generic visitor with a single method visit that the specialized visitors implement. The Visitors module implements the submission and search operations of service profiles to support the correspondent UI actions.

The Handlers module includes a set of delegate handlers to operate on UDDI tModels records. The tModel objects contain metadata about the various specifications implemented by a given Web service. The delegated functions required are the creation, finding and deletion of tModels. This module is use by other top-level modules like Visitors.

The MetadataFactory module implements a single method to create generic MetadataItem objects. These objects hold user-defined metadata, name-value pairs that the UDDI2M server binds to a BusinessService instance. The MetadataItem objects constitute the extension to the standard UDDI information
model. They carry metadata binds that only exists on the context of one organization and not to all the participating ones. The registry is agnostic in relation to these metadata but the user is able to write external matching algorithms, using an extensible configuration module, to perform rich queries on the UDDI2M catalogue. To operate this kind of rich query, the user selects the class name of the matching algorithm and specifies the name-value pair to be matched or found. The MatchingAlgorithms module defines a statement model to perform matching operations on MetadataItem objects. It instantiates the GoF Command design pattern to encapsulate a matching request as an object. By this, the client can parameterize the requests passing to it a collection of actual values that will be matched with the formal value of the MetadataItem object. The matching results are then returned to the invoker code.

The WebService module defines interfaces for knowledge base uploads and downloads, more the service profile publishing and discover subsystems. Each facet defines a higher-level interface that makes the subsystem easier to use. These interfaces receive service profiles messages as method parameters according to the UDDI2M XML Schema, and they are submitted to validation against it.

In practice, the metadata items, both taxonomy nodes and generic metadata items, are stored in UDDI tModel records. To support classification, the UDDI information model defines “category bag” type objects which bind a collection of metadata item referenced keys to any object defined in UDDI.

Next we present an example scenario and a description of the control flow that executes on the UDDI2M catalogue service.

When, in the JUMP map viewer, a client user selects a graphical area in the screen, he may want to ask the UDDI2M catalogue for registered WFS providers under that area. A pre-condition for it be successful is some WFS provider be registered in the UDDI2M catalogue with the service access point. To make possible more accurate queries based on specific visual area constraints, the WFS provider must classify its service with a generic name-value metadata pair containing, for example, a BoundingBox item with the value of the polygon dimensions that the service covers. The user also selects the class name of a matching algorithm appropriated for BoundingBox items and invokes the search operation on the UDDI2M catalogue. The catalogue service processes the existent metadata bindings to check if the user bounding box contains some registered services bounding box. If the registry found a true matching, the correspondent service profile is built and returned to the map viewer client. The JUMP WFS plug-in will parse the service profile to get the target access point and create a request to it. After invoking the WFS service, the client code processes the returned Geographic Markup Language (GML) information and asks the JUMP render engine to present the geo-data in a different layer.

At software code level, the actions taken inside the UDDI2M catalogue on the referred use-case are shown on the figure 4. Next, we will describe in detail the control flow that exists among the several UDDI2M modules. From the map viewer application, the UDDI2M catalogue gets the rich query elements, i.e., the name and value of the metadata binding to be matched and the matching class name. Using introspection, the UDDI2M server picks up the matching class object instance constructed with the formal value of the metadata item. A visitor object of type “find” is created to build the data structures in the case of metadata matching. The visitor object will iterate on the jUDDI structures looking for metadata bindings with the name passed in the search facet. For the BusinessServices that have a metadata binding with this name, the visitor invokes the matching algorithm. If a true matching is found, the visitor creates the service profile structures that will be serialized to XML and returned to the map viewer application. Then, the JUMP catalogue wizard presents the data in a user friendly format, so one of the
services URI can be selected. A request to one of the URIs is made using JUMP procedures to obtain the GML data.

3.4. Results

The product of the work reported on this paper is a global GIS architecture, designed on SOA style basis, in where coexist different types of software systems such as: powerful client applications, e.g., the JUMP platform; GIS servers like WMS and WFS; and an enhanced UDDI server, the UDDI2M catalogue. To the overall system was given the basis to support an ontology information model for context-based service profile publishing and discover operations.

Using the software infrastructure proposed on this paper, client workstations can run the enhanced JUMP application which integrates advanced visualization and manipulation tools of visual data-sets with ontology edition and advanced catalogue management.

The final software artifact, written at the join point of the two development lines, i.e., the UDDI2M catalogue service and the client application integration process, is a XML Schema with the description of service profiles, plus the method formal descriptions of the UDDI2M catalogue service for documentation purposes. It describes, in a structured form, the aggregation of the different types of metadata supported by the catalogue. The ones matching the UDDI information model, the bindings made on knowledge base concept names, and external metadata defined in exclusive user contexts. The service profiles are the transactional units that the UDDI2M catalogue service exchanges with its clients, through the UDDI2M web service interface.

4. Conclusions

We have presented a new technological approach to the web service paradigm, adapting the client infrastructures to a business logic service that can easily add semantic context to the information stored in a standard UDDI server.

The integration of the JUMP and Protégé frameworks enables easy creation and management of a domain-specific ontology. With the knowledge of the service profiles, the catalogue user has a descriptive form to write publish and inquiry messages and sent them through the web service adapter. The use of different storage backends and transformation plug-ins frees the ontology editor from a specific ontology format, since the essential ontology content is in the structural knowledge description about the specific application domain. The XML transformation plug-in is an ontology language adaptor giving to the UDDI2M ontology loader a uniform ontology representation.

The UDDI2M catalogue service is an extensible and configurable business service with a web service interface. It contains a library for ontology uploading, Visitor design pattern instantiations to implement the jUDDI related operations, a generic metadata item factory and a well-defined interface for definition of matching algorithms receivers.

Improvements were made on both client and server technology sets, providing a context-aware environment between service providers and consumers. The information model specified in the UDDI2M XML Schema is the contract made by the catalogue and its clients. The UDDI2M catalogue enables service providers to relate their services to a generally
accepted semantic information, and give to service requesters a rich query mechanism, creating conceptual environments, shared by a group of heterogeneous services and applications, that have the ontology as the baseline information support for given context domains.

5. OpenGIS® and future work

The Application Profile for the OpenGIS® Catalogue Services Specification v2.0 defines how to discover, retrieve and manage services metadata among with other types of metadata. This application profile document specifies the interfaces, bindings, and encodings required to publish and access digital catalogues of metadata for geospatial data, services, and applications that comply with the given profile. Metadata act as generalised properties that can be queried and returned through catalogue services for resource evaluation and, in many cases, invocation or retrieval of the referenced resource. Future work on this specification may include the extension of the profile to interoperate with UDDI service registry instances [13]. The compliance of the catalogue UDDI2M with the mentioned specification could constitute an interest focus of development.

The OpenGIS® Web Map Context specification states how a specific grouping of one or more maps from one or more map servers can be described in a portable, platform-independent format for storage in a repository or for transmission between clients. A Context document includes server information about the set of layers in the overall map and necessary operational metadata, e.g. the bounding box, for client software to reproduce the map. The Context document could be saved from one client session and transferred to a different client application to start up with the same context. Contexts could be catalogued and discovered, thus providing a level of granularity broader than individual layers [14]. Since a Context document is structured using XML, the UDDI2M catalogue can store bindings made between the Context XML file URI and description elements.

6. Another use-case scenario

Assuming that the UDDI2M catalogue can be used to store the web location of a Web Map Context document, as an access point for a business service, could be necessary to introduce some authentication procedures guarantying that only the target recipient user can have access to the service profile containing the Context document URI. To ensure authentication in query operations, published service profiles may include a generic metadata item created with the target user name and a password. When querying the catalogue, users need to add a query metadata item with their user name and encoded password with some well-known algorithm. The catalogue will try to instantiate the encrypting algorithm class and decode the password sent in the service profile. The query operation will only return the result if a true password matching was found.

7. Acknowledgements

This research was supported in part by the MEDSI (Management Decision Support for Critical Infrastructures) European project. The objective of MEDSI project is to develop a web-based integrated set of software services as a tool to enhance the capabilities of crisis planners and crisis managers in both private and governmental organizations. It’s expected that MEDSI can produce software that can handle with data fusion of geospace, organizational and other territorial and operational areas, using standardized IT equipment.
8. References


