Integrating Data Mining Services over Knowledge Portals using WSRP and AJAX Technologies

Vasile Georgescu

***Abstract*—**Integrating Distributed Data Mining functionality remotely over an Enterprise Knowledge Portal acts as a core of an enterprise-wide network system allowing the end-users to work collaboratively, quickly share information and knowledge inferred by analytical tools such as Data Mining models. All functionality can be accessed via a standard Web browser across different locations or branches of a corporation. This paper focuses on implementation solutions for developing an ecosystem of self-organizing, competing and evolving distributed data mining Web-based services. Specific options are considered, such as: designing an integrated architecture on top of which a portal is built as a gateway for knowledge exchange and intelligent business transactions; exploiting recently released standard interfaces and communications protocols that allows to integrate remote web services into portals as portlets, and to run portlets remotely, for interoperable data mining tasks; adoption of a standard Java API along with a Data Mining Engine and a Metadata Repository, allowing developing a service-oriented distributed data mining platform; XML-based standard representation of predictive models for facilitating the export and import of data mining objects; providing mechanisms for exposing and consuming distributed data mining services. We can also benefit from the new AJAX technologies, such as Google Web Toolkit (GWT), to dramatically improve the functionality of our web applications.

***Keywords*—**Business Intelligence, Knowledge-driven portals, Distributed data mining, Digital business ecosystems.

I. Providing Business Intelligence through a Knowledge-Driven Portal, Built on Top of an Ecosystem of Distributed Data Mining Services

TRADITIONAL data mining systems were largely stand-alone systems, which required all the data to be collected at one centralized location (typically, the users machine) where mining would be performed. However, organizations operating in global markets need to perform data mining on distributed and heterogeneous data sources and require cohesive and integrated knowledge from such data. The inherent distribution of data sources and the large volumes of stored data, inevitably lead to high communications costs. Therefore, it is evident that the traditional data mining model involving the co-location of users, data, and computational resources, is inadequate when dealing with collaborative business environments sharing distributed resources. The development of data mining along this dimension has lead to emergence of *Distributed Data Mining (DDM).* DDM addresses the impact of distribution of users, software and computational resources on the data mining process.

There are several emerging technologies and standards that have the potential to support a distributed data mining scenario, such as web services technologies, agent-based technologies and grid-enabled technologies.

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services can be *discovered*, *described*, and *accessed* based on XML and standard Web protocols not only over the Internet, but also over intranets or extranets. Web services could be used between partnering enterprises within a digital business environment for producing and exchanging structured information and formalized knowledge. Web services can be completely independent of the presentation, or the graphical user interface (GUI) of applications. Instead, Web services send data in XML format, and applications can add style and formatting when they receive the data. Exploiting a service-oriented architecture for exposing and consuming data mining services over the Web, allows virtual organizations to access and develop distributed data mining tasks and knowledge discovery processes for supporting very effective and reliable corporate decision making.

The emergence of such technologies makes it possible to establish a self-organizing and evolving virtual marketplace of data mining services, where clients can make ad hoc requests and service providers compete for tasks, based upon an interaction protocol. We consider this scenario to be fully compliant with the vision of Digital Business Ecosystems, lunched as part of the Lisbon strategy for European growth and competitiveness. It should be crucial for supporting knowledge management and providing business intelligence over networks of knowledge-centric organizations.

Building a knowledge-driven portal on top of a service-oriented distributed data mining system should also play a key role. Portals can be regarded as an information gateway for delivering just-in-time and self-explanatory business information over the Internet. The information could be supplied as a set of timed transactions triggered by events in business activities. In a business environment adopting business intelligence, knowledge is used to make business

Manuscript received May 31, 2007: Revised version received October 28, 2007.

V. Georgescu is with the Department of Mathematical Economics, University of Craiova, 13 A.I.Cuza, 200585 Craiova, Romania (phone: +4072-360-9006; fax: +4035-140-2211; e-mail: vgeo@central.ucv.ro).

decisions, to search for useful information, or to control the routine business processes.

Portals lie in the core of the fundamental information infrastructure of e-businesses. They evolve towards maximizing their usefulness in order to improve business performance, competitiveness, and viability. Intelligent Portals would be knowledge driven portals. In other words, by capturing the domain-specific business knowledge, we can deploy Portals on the Web and let their behaviour be controlled by a knowledge base. In this case, a knowledge management system would play an important role in an e-business environment.

Portals are built on top of a stratified system, which typically consists of three layers: the memory repositories layer, the knowledge administration layer and the presentation layer (the Web-based portal itself, which is responsible for transporting the information to the end user).

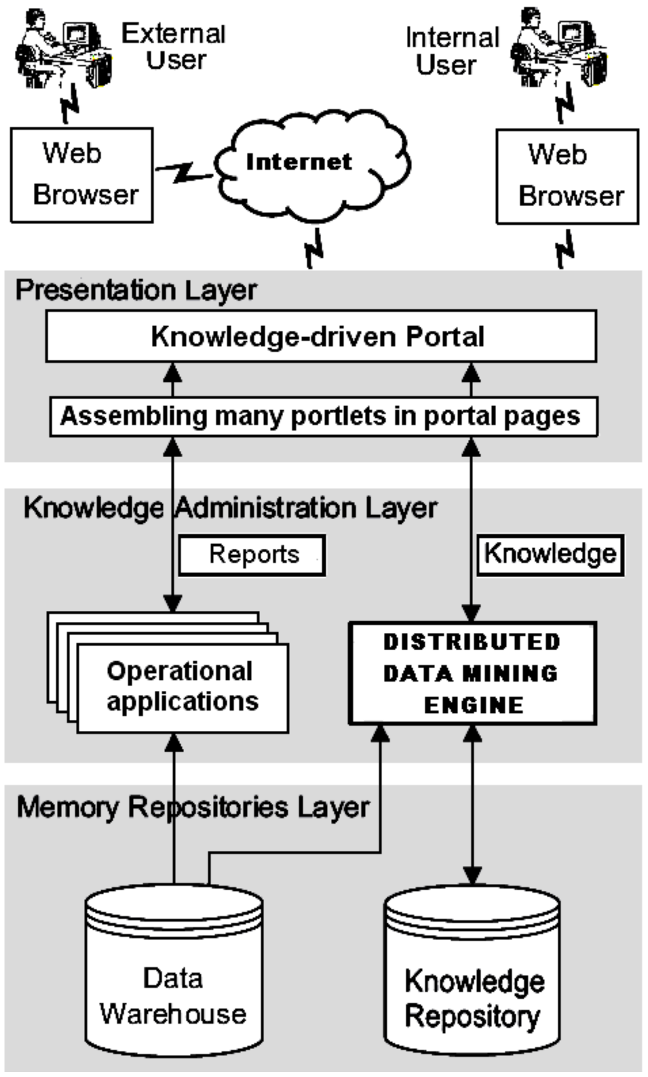

Fig. 1 A knowledge-driven portal on top of a distributed data mining system

Designing and implementing intelligent business portals for Web-enabled business applications is a challenging task. Intelligence is an executive framework for applying knowledge. When a reasoning mechanism is invoked, the relevant knowledge is retrieved and a possible answer or solution is then concluded. In a business environment, knowledge can be formalized in different forms. It could be a set of "if-then" production rules for decision-making problems, a set of facts for corporate infrastructure descriptions, or a set of procedural descriptions for the business transactions. Business activities are event driven, so the timely execution of certain business processes is crucial to the success of business. Furthermore, a reasoning mechanism may be triggered for context-sensitive reasoning and decision making. From this viewpoint, an intelligent portal is a context sensitive information/service supplier that will accompany the user through the lifetime of the transaction. It acts as an information feeder that will satisfy the information needs of different users at different times for different business processes.

Web services can play a pivotal role in providing business intelligence over portals. The concept of portlets relates Web services and portals in a direct and convenient way. In effect, Web services can be viewed as modular software "engines" that power a particular portlet, either on a one Web service to one portlet basis (e.g., a stock price portlet) or a multiple Web services to one portlet arrangement (e.g., a portlet that could combine input from many Web services offering diversified information).

As application components within a portal, portlets are an encapsulation of content and functionality. They are reusable, have messaging and event recognition capabilities, and combine Web-based content, application functionality, and access to resources.

A software architecture assembling many portlets in portal pages could be considered to make up an application of a comprehensive set of Distributed Data Mining Services.

II. Adopting a New Standard for Delivering Distributed Applications over Portals: Web Services for Remote Portlets

Standard interfaces and communications protocols have recently been defined to allow the integration of remote web services into portals as portlets, and to run portlets remotely.

The Web Services for Remote Portlets (WSRP) specification is a product of the Organization for the Advancement of Structured Information Standards (OASIS), and defines a web-service interface for interacting with presentation-oriented web services that are designed to be plug-and-play adapters for portals. By defining a set of common interfaces, WSRP allows portals to display remotely-running portlets inside their pages without requiring any additional programming by the portal developers. To the end-user, it appears that the portlet is running locally within their portal, but in reality the portlet resides in a remotely-running portlet container, and interaction occurs through the exchange of SOAP messages. These SOAP-based Web services process user interactions and provide mark-up fragments for aggregation by portals.

WSRP defines a WSDL interface description for invocation of WSRP services. It defines how to publish, find and bind WSRP services and meta data. WSRP specifies rules for using markup that is emitted by WSRP services, along with applicable security mechanisms, billing information and so on. To allow for easy integration of their content in portals, content providers can use WSRP to surface the content as remote portlets, publishing them as WSRP services in the public, global UDDI directory. Once the content provider has published a WSRP service in the UDDI registry, administrators of portals who wish to use content from the content provider can look up the content provider's business entry in the universal description, discovery and integration

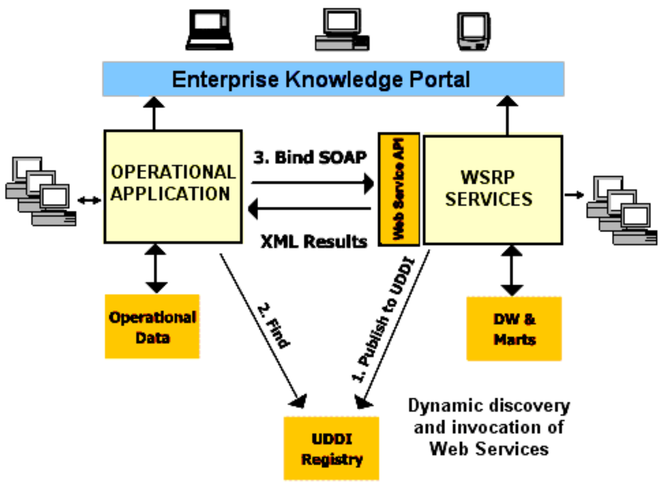

Fig. 2 WSRP used for Plug-and-Play Portal Adapters (Source: OASIS)

(UDDI) registry and bind to the WSRP service that offers the desired content. (Figure 2).

Portal administrators can use their portal's administration facility to browse a registry for WSRP services and select them for automatic integration into the portal. This plug-and-play approach enables content providers to write one interface to their source content for Web services and then describe that Web service using WSDL and WSRP. Similarly, portal vendors only have to develop one generic adapter that is designed to recognize WSRP and WSDL and then incorporate that generic adapter into their portal product.

While local portlets can be expected to provide a large part of the base functionality for portals, the remote portlet concept allows dynamic binding of a variety of remote portlet Web services without any installation effort or code running locally on the portal server. Also, portals may wrap local portlets and publish them as remote portlet Web services for integration by other portals. Conversely, remote portlet Web services can be integrated into portals by wrapping them in a proxy written to the local portlet API.

Figure 3 is an example of a high-level portal architecture that may be employed for combined use of local and remote portlets as well as making local portlets available via WSRP for use by other portals.

Figure 4 illustrates each of the primary actors within a WSRP architecture. The heart of a portal is a portlet container. Portlets are the reusable Web components for building portals. Portlet containers execute portlets and manage their life cycle. Additionally, portals offer an aggregation mechanism with which one can arrange multiple portlets on a single page. By definition, portlets generate markup fragments, not a complete page. A portlet is one of many components on a page. The

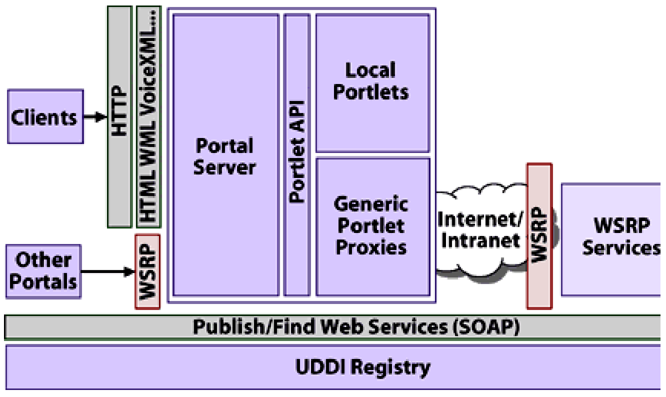

Fig. 3 Enterprise Portal Architecture Using WSRP (Source: OASIS)

WSRP specification enables portlets to be exposed as Web

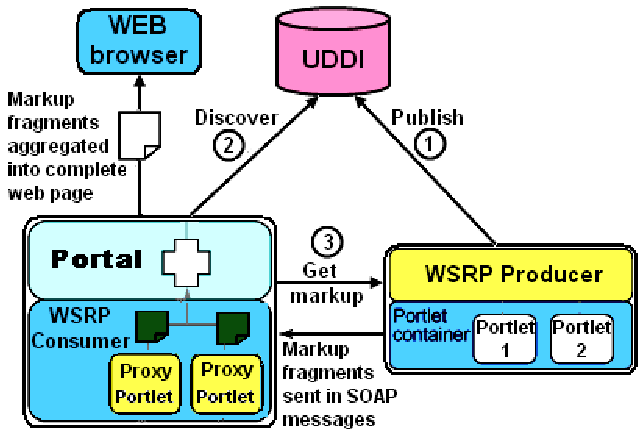

Fig. 4 A typical publish-find-bind usage scenario involving WSRP

services and consumed by remote portals.

III. A Service-Oriented Architecture Designed to Exploit the Java Data Mining Functionality Remotely over Portals

A. *Java Data Mining Standards and Specifications*

In the last years there is a large support for adopting the distributed data mining scenario, consisting of several emerging data mining standards and initiatives that facilitate more open interaction and interoperation for Web-based distributed data mining.

An important initiative has been to produce a standard API specification in order to leverage data mining functionality. With a standard API, users can exploit multiple products for solving business problems by applying the most appropriate algorithm implementation without investing resources to learn each vendor's proprietary API.

Java Data Mining (JDM), which is an emerging data mining standard, following Sun's Java Community Process as a Java Specification Request (JSR), explicitly addresses this need. It has been released as JDM 1.0 under the JSR-73 specification and the next release (JDM 2.0) comes under the JSR-247 specification. The architecture of JDM consists of three logical components, the API, the Data Mining Engine (DME), and the Metadata Repository (MR).

The Java Data Mining API proposes a pure Java API for developing data mining applications. The API is the exposed programming interface that provides access to the services provided by the DME. The API shields the data mining user from the actual implementation in the DME and any associated subcomponents used by the DME.

The DME is the engine that provides services that can be consumed by users through the API mentioned above. The DME can be implemented as a server in which case it is called a Data Mining Server.

The third component is the Metadata Repository, which is used to persist data mining objects. These persisted data mining objects are again used by the DME for data mining operations. The MR can exist as an ordinary file system or can be a relational database.

The three logical components can be grouped into one physical system or they can exist independently as separate components.

JDM divides mining objects into *data specification*, *settings*, *model*, *test metrics*, and *task* objects. The *MiningObject* interface encapsulates the common characteristics of these objects, such as name, description, object identifier, and the mining object type.

Users can get a *connection object* to access the DME by using a connection factory. Connection objects also provide access to the objects present in the MR, being able to create, retrieve and delete such mining objects via the DME.

JDM covers the functional areas of classification, regression, attribute importance, clustering, association, feature extraction, time series, and anomaly detection. These are supported by such supervised and unsupervised learning algorithms as decision trees, neural networks, Naive Bayes, Support Vector Machine, K-Means, Apriori, Non-negative Matrix Factorization, and ARIMA.

B. The Service Oriented Architecture Enabled by the Java Data Mining Standard

The most important feature of JDM is to enable a service oriented architecture for supporting distributed, loosely coupled applications. Data mining Web services provide an opportunity to facilitate integration of multiple data mining software implementations in a single application, as well as to enable language-independent development.

JDM includes as specification a Web services definition for data mining based on the JDM UML (Unified Modeling Language) model. It provides a high degree of interoperability between Java and Web service interfaces: common metadata, object structure, and capabilities. Applications can use the Java interface to generate and consume XML objects more easily than the import and export tasks. JDM's design to keep large inputs and results at the JDM server makes it naturally amenable to Web service design. This specification includes the JDM WSDL types for document style Web services and the XML schema definition for objects defined in JDM to enable JDM developers to support the Web services definition. A data model consistent with the JDM object model is defined in the JDM XML Schema. This schema follows a similar inheritance hierarchy of the JDM UML model and defines a complex type for each object in JDM.

An XML Schema provides mechanisms to define and describe the structure, content, and to some extent semantics of XML documents. JDM XML Schema definitions support multiple uses, such as interchanging data mining objects among data mining engines (DMEs), defining Web services, storing data mining objects as XML documents, and integrating JDM implementations with non-Java applications.

JDM Schema types have a one-to-one mapping with the JDM API classes or interfaces. JDM Schema types use the same name of the associated Java classes or interfaces, and maintain the same object inheritance and relationships as in the API whenever possible. In addition, the JDM schema follows JAX-RPC Java-XML data type mapping guidelines.

The JDM standard defines a Web service's interface (JDMWS) that uses the XML Schema and provides functionality consistent with the JDM Java API. However, unlike a Java API, Web services typically define a small number of operations with relatively large messages to reduce the number of service calls to be made. Actually, JDM Web services define 11 operations, categorized into three types: mining task execution and monitoring, mining object management, and capability discovery.

In order to maintain a consistent API for both Java and Web services applications, JDM Standard uses AX-RPC, which provides an easy way to wrap the JDM API implementation as JDMWS. JAX-RPC is the Java API for XML-based Remote Procedure Call (RPC). JAXRPC API is used to develop Java-based portable Web services using WSDL-to-Java and Java-to-WSDL standard mappings. The JAX-RPC runtime system automates the conversion of Java method calls and objects to appropriate SOAP request and response messages in a Web services environment.

C. Integrating JDM with WSRP to Expose and Consume Distributed Data Mining Services Remotely over Enterprise Knowledge Portals

An Enterprise Knowledge Portal (EKP) adds new functionalities to the traditional Enterprise Information Portals (EIPs) in some important respects. An EKP is goal-directed toward knowledge production, knowledge acquisition, knowledge transmission, and knowledge management and also focuses upon, provides, produces, and manages information about the validity of the information it supplies.

In the case of EKPs the renewed emphasis on *data mining and analytical applications* will be particularly strong since these have a critical role in producing new knowledge.

Business Intelligence tools and Data Mining tools in particular need to integrate with Enterprise Knowledge Portal offerings to be successful in a competitive environment.

A critical point when designing a service-oriented architecture for providing distributed data mining functionality is how the content of web services received as SOAP messages are visualized for end-users. Traditional web services are data-oriented and do not include any user interaction or presentation functionality. Obviously, it would be much more convenient from a portal perspective if web services would appear as visual, user-facing services including presentation and application logic.

The Web Services for Remote Portlets (WSRP) standard explicitly addresses this need for a web service interface. It provides interoperability between different kinds of intermediary applications and visual, user-facing web services, allowing them to be invoked through generic portlet proxies on the portal side.

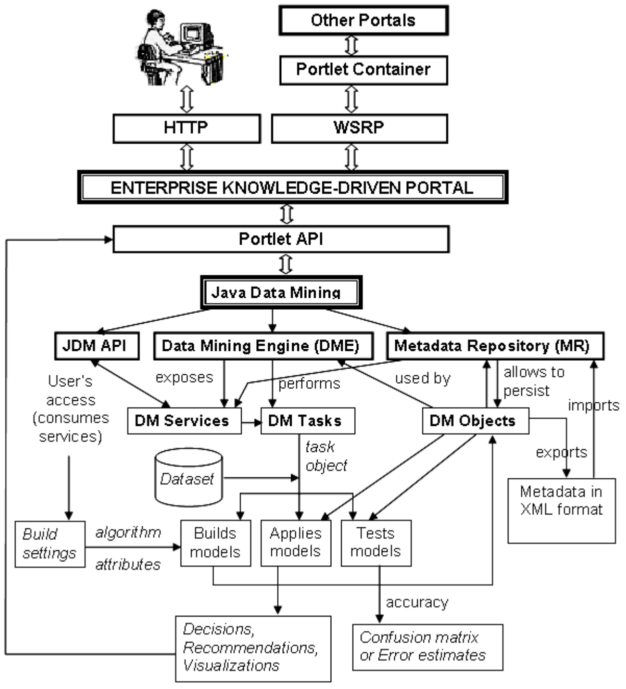

Fig. 5 A WSRP-enabled, service-oriented architecture, designed to exploit Java Data Mining functionality remotely over portals

The WSRP-compliant architecture assumes that clients access portal implementations via the HTTP protocol. When aggregating pages for portal users, portals typically invoke all portlets belonging to a user's page through a Portlet API for locally installed portlets. Local Portlets run on the portal server itself. They may be deployed by installing portlet archive files on portal servers and are typically invoked by the portal server directly through local method calls. Remote Portlets run as web services on remote servers that are published in a UDDI directory to allow for easy finding and binding. Typically portlet proxies will invoke WSRP services to which they are bound via the SOAP protocol. Applications use WSRP services by creating remote portlet instances, sending events or requesting markup and finally destroying remote portlet instances when they are not needed anymore.

Integrating such ITC-based standards and technologies, makes it possible to provide business ecosystems with Web-based digital applications (such as distributed Web services running on remote portals). This may lead to knowledge-centric organizations equipped with very effective and efficient knowledge management capabilities.

Our vision of an ecosystem of self-organizing, competing and evolving distributed data mining Web-based services, coupled with the technology of remote portlets, is illustrated in figure 5, which represents a service-oriented architecture designed to exploit Java Data Mining functionality remotely over portals.

IV. An Ajax-Based Solution to Developing Rich Web Application over Portals

WSRP is a great framework for building dynamic configurations for portals by composing web pages from remote portlet containers. However, this technology focuses only on using distributed pieces for assembling web architectures, but its basic ingredients are still classical, leading to form-based applications.

In order to build more and more complex web applications there is a need for improving their functionality in a way that mimics the functionality of desktop applications. The recent advances in Ajax technology provide an appropriate solution to reaching this goal.

Ajax applications improve web server performance by holding application state and relieving the server of this task. This means that instead of loading a new page for each action, Ajax applications can execute several actions together on the client side and submit data as a batch only when necessary.

Google Web Toolkit (GWT) is the most recent and probably the most revolutionary Ajaxian framework. Since it was announced by Google in June 2006, it became the new wave of Java-based web application framework. Unlike the frameworks that came before it, GWT is not centered on classic, form-based applications. Instead, it represents a completely different approach to building web applications that are designed to look very similar to desktop applications, while running in a browser: rather than writing browser-side code in HTML and JavaScript, with Java servlets or JSF components to handle the server side of the application, GWT lets developers write the whole application, from client to server, in Java, gaining all of the advantages of Java as a programming language. This became possible because at the core of GWT is a Java to JavaScript compiler that produces code capable of running on standard browsers such as Internet Explorer, Firefox, Mozilla, Safari, and Opera. The compiler converts the Java syntax into a set of browser-compliant JavaScript and HTML files, utilizing JavaScript versions of commonly used Java libraries like Vector, HashMap, and Data. Beyond the compiler, GWT also includes a large user interface library of widgets (text boxes, drop-down menus, menu bars, tree controls, dialog boxes) and panels, libraries to perform asynchronous server communication through HTTP or remote procedure calls (RPCs), tools to interoperate with other web applications using JavaScript, JSON and XML and so on. These make it effortless to build a web application that looks more like a desktop application.

The HTTP library provided by GWT allows getting better performance with asynchronous communication by sending and receiving data from a server asynchronously. Actually, GWT applications do not need to fetch new HTML pages while they execute. The specific mechanism GWT uses for interacting with a server across a network is making a remote procedure call (RPC), by which it connects to a Java servlet and makes invoking methods on the server as easy as making a local method call. The nature of asynchronous method calls requires the caller to pass in a callback object that can be notified when an asynchronous call completes. After an asynchronous call is made, all communication back to the caller is via the passed-in callback object.

We used GWT to build two web applications for running over our knowledge portal:

- an Application Designer for helping the user to automatically generate XML code that describes complex data mining tasks;
- an Application Manager that allows users to register for having access to application resources, and to manage all the actions for which they granted permission, such as connecting to the data mining engine hosted by a server, connecting to remote databases, editing, running and modifying complex data mining applications, visualizing the results, and so on.

GWT allows fast asynchronous browsing and editing both application code and application data and uses Data Access Objects (DAOs) to access complex data structures on the server. A code generator is used to automatically serialize Java objects to and from JSON and XML. The client-side application uses the computational resources on the server by using GWT-enabled server integration mechanisms, such as integrating with action-based PHP scripts, or with a GWT-RPC servlet.

V. Conclusion

This paper aimed to outline the key directions of our research on providing Business Intelligence and Distributed Data Mining functionality within Digital Business Ecosystems. We focused on: designing an integrated architecture on top of which a portal is built as a gateway for knowledge exchange and intelligent business transactions; exploiting recently released standard interfaces and communications protocols that allows to integrate remote web services into portals as portlets, and to run portlets remotely, for interoperable data mining tasks; adoption of a standard Java API along with a Data Mining Engine and a Metadata Repository, allowing developing a service-oriented distributed data mining platform; XML-based standard representation of predictive models for facilitating the export and import of data mining objects; providing mechanisms for exposing and consuming distributed data mining services. An Enterprise Knowledge Portal that integrate Distributed Data Mining functionality either locally or remotely acts as a core of an enterprise-wide network system allowing the end-users to work collaboratively, quickly share information and knowledge inferred by analytical tools such as Data Mining models. In order to build powerful web applications that look and act very similar to desktop applications, we used a Java centric Ajaxian framework: Google Web Toolkit. All functionality can be accessed via a standard Web browser across different locations or branches of a corporation.

References

[1] Dewsbury R., *Google Web Toolkit Applications*, Prentice Hall, 2008.
[2] Geary D., *Google Web Toolkit Solutions*, Prentice Hall, 2008.
[3] Hanson R., Tacy A., *GWT in Action. Easy Ajax with the Google Toolkit*, Manning, 2007.
[4] Hornick M.F, Marcade E, Venkayala S, *Java Data Mining. Strategy, Standard, and Practice*. Morgan Kaufmann, 2007.
[5] Holzner S., *Ajax Bible*, John Wiley & Sons, 2007.
[6] Keith J., *DOM Scripting. Web Design with JavaScript and the Document Object Model*, Friensof (Apress), 2005.
[7] Ian H. Witten, Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, San Francisco, Morgan Kaufmann, 2005.
[8] Lauriat S. M., *Advanced Ajax. Architecture and Best Practices*, Prentice Hall, 2007.
[9] "JSR 071: Java Data Mining 1.1 specification and API", http://jcp.org/en/jsr/detail?id=073.
[10] "JSR 247: Java Data Mining 2.0", http://www.jcp.org/en/jsr/detail?id=247
[11] Olson S.D., *Ajax on Java*, O'Reilly, 2007.
[12] "OASIS Web Services for Remote Portals Web Site", http://oasis-open.org/committees/wsrp.
[13] "JSR 168: Portlet API" http://www.jcp.org/jsr/detail/168.jsp.
[14] Zammetti F. W., *Practical JavaScript, DOM Scripting, and Ajax Projects*, Apress, 2007.