METALA: a J2EE Technology Based Framework for Web Mining*

Juan M. Hernansaez † Juan A. Botía ‡ Antonio F. Skarmeta §

Abstract

In this paper, we discuss the most important aspects of METALA, a software tool for meta-learning that we have developed to perform inductive learning in a distributed and component based fashion. The distribution comes from the use of a well posed distributed application development standard as is J2EE, and the component basis comes from the methodology we developed to integrate new learning algorithms and other software utilities into the system. We aim to use this new architecture for evaluate algorithms of Web Usage Mining, a concrete learning problem of the Web Mining area, and for taking advantage of these algorithms to build new knowledge models. These models can be used then to create and incorporate new applications and tools to the architecture. We discuss the advantages and flaws of using J2EE as the technology basis. We also compare our architecture with some other software platforms intended to solve similar Web Mining problems as METALA can solve. To illustrate the use of METALA, we present a complete Web Usage Mining life cycle process explanation.

Keywords: *Software architecture, web usage mining, inductive learning, knowledge models, J2EE, XML.*

Resumen

En este artículo, comentamos los aspectos más importantes de METALA, una herramienta software para meta-aprendizaje, que hemos desarrollado para realizar aprendizaje inductivo de manera distribuida y basada en componentes. La distribución viene por el uso de un estándar de desarrollo de aplicaciones distribuidas bien definido como es J2EE, y la base de componentes de la metodología que hemos desarrollado para integrar nuevos algoritmos y otras utilidades software dentro del sistema. Nuestro objetivo es usar esta nueva arquitectura para evaluar algoritmos de Minería de Uso Web, un problema de aprendizaje concreto del area de la Minería del Web, y aprovechar estos algoritmos para construir nuevos modelos de conocimiento. Estos modelos pueden usarse después para crear e incorporar en la arquitectura nuevas aplicaciones y herramientas. Comentaremos las ventajas y defectos de usar J2EE como base tecnológica. Además compararemos nuestra arquitectura con otras plataformas software propuestas para resolver problemas de Minería del Web similares a los que METALA puede resolver. Para

*Supported by the Spanish CICYT through the project TIC2002-04021-C02-02

†Departamento de la Ingeniería de la Información y las Comunicaciones, Facultad de Informática de la Universidad de Murcia, juanma@um.es

‡Departamento de la Ingeniería de la Información y las Comunicaciones, Facultad de Informática de la Universidad de Murcia, juanbot@um.es

§Departamento de la Ingeniería de la Información y las Comunicaciones, Facultad de Informática de la Universidad de Murcia, skarmeta@um.es

ilustrar el uso de METALA, presentamos la explicación del proceso de un ciclo de vida completo de Minería de Uso Web.

Palabras clave: *Arquitectura del software, minería de uso de la web, aprendizaje inductivo, modelos de conocimiento, J2EE, XML.*

1 Introduction

In this paper we propose **METALA**, a META-Learning Architecture. Our architecture aims to facilitate the coding of different learning paradigms and domains applications in the server side and the testing of the techniques of the incorporated paradigms from the client side.

As stated in [12], **Web Mining** (WM) can be viewed as the use of data mining techniques to automatically retrieve, extract and evaluate information for knowledge discovery from web documents and services. Although considered to be a particular application of data mining, it is clear that a separate research field is required to deal with all the problems and challenges of information mining from the Web. Currently, there are two different areas of WM: Web Resources Mining and Web Usage Mining (WUM). We focus on WUM. One of the most accepted definitions for WUM is given in [7]: it is the application of data mining techniques to large web data repositories in order to extract usage patterns. As we know, web servers around the world record data about user interaction with the web pages hosted in web servers. If we analyze the web access logs of different web sites we can know more about the user behaviour, allowing personalization or making easier the improvement of web sites design among other interesting applications.

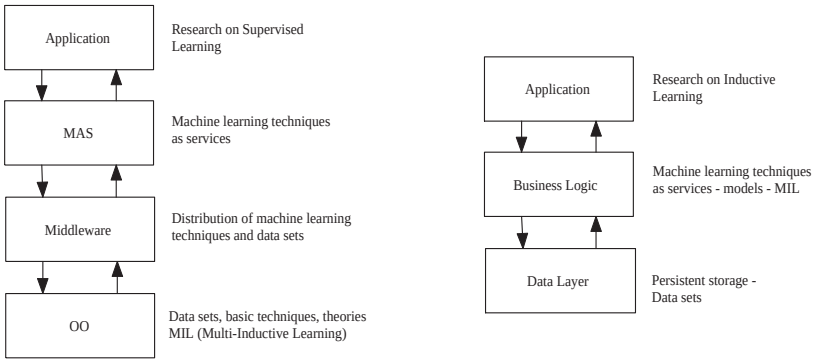
At the present time, there are three main approaches to face the problems of WUM: clustering, association rules and sequential patterns discovery. There are many techniques for the different approaches. For this paper, we have tested the Apriori algorithm [1] from the association rules approach to Web Usage Mining, and added an application which takes advantage of the learning process result (knowledge model) of such algorithm.

The original contribution of this paper consists of this J2EE based soft computing framework for WM, how it can be used to incorporate new learning paradigms and domain applications, and how it can be useful for adding new abstraction layers. For our future work we are planning to add a new *intelligent agent* layer, which can be also profitable for WM purposes. This would allow us to have most of the Web Mining phases, applications and technologies available from a built-in single framework.

The rest of the paper is organized as follows: in section 2, we present our software architecture for automated data analysis processes. After that, in section 3 we discuss the most relevant implementation issues of the new software architecture. Then, in section 4 we will show a complete WUM life cycle resulting in a knowledge model that we use to build an URL recommender application onto our architecture. In section 5 we compare some of the features of our tool with other similar WM intended tools, like [11] and [15]. Finally, in section 6 we discuss the benefits and uniqueness of METALA and we outline our future work.

2 Architecture of METALA

METALA is a software architecture which aims to guide the engineering of the information systems which support *multi-process inductive learning - MIL*. The previous version



(a) Abstraction layers of METALA-RMI

(b) Abstraction layers of METALA-J2EE

Figure 1: Abstraction layers of METALA.

of METALA (which we named METALA-RMI) was defined with four different abstraction layers (see figure 1 (a)): (1) Object-oriented layer (OO), (2) Middleware layer and (3) agents layer (MAS, Multi-Agent System). The layer (4) is the METALA application. We can compare this layer with the one of the figure 1 (b), which corresponds to the new version of METALA (which we named METALA-J2EE). In this new architecture there are only three layers: (1) data layer, (2) business logic layer and (3) METALA application layer.

METALA-RMI was based on different technologies: RMI (Remote Method Invocation) for providing distribution, LDAP for service location and data persistence, etc. Everything has to be explicitely coded using Java, including load-balancing, communications, distribution, persistent data storing and retrieving, performance, etc. Thus, the maintenance and extensibility of the system was not easy, although we provided a coding methodology based on a predesigned **business logic**, so METALA users wishing to test their algorithms could use this methodology and forget about technology-related issues. Then, we thought about using J2EE as the only technology basis of our framework.

We show below which are the underlying technologies that we are currently using and why. But first, we must explain how the METALA systems works, i.e. which is its **business logic**.

2.1 Business logic of METALA

We start from the data managed by any learning technique. The basic data unit is the *instance*. It is composed of a set of attribute values referred to a particular sample in a learning data source. A data source is a set of instances. Instances in a data source can be accessed by using a cursor named *access*.

The learning techniques of the architecture are the possible services to be offered and used. An usual operation in METALA has the following sequence of actions: first, configure the parameters needed by a learning technique (i.e. configure the *experiment*); then, launch the experiment; while the experiment is running, monitor its progress generating desired progress watches (progress logs); finally, if the experiment ends successfully, generate the associated data model, for further evaluation or utilization.

Data models are the pieces of knowledge induced from data by a particular learning technique. They may be stored for analysis, recreated for feeding other learning processes,

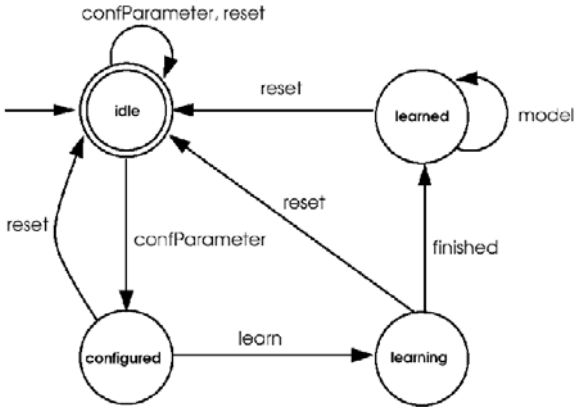


Figure 2: Learning process of any learning technique.

visualized for user-friendly evaluation and used as part of new applications, as we will see in section 4. In figure 2 we show the behaviour of all the learning services of METALA.

The business logic of METALA has not changed along its different versions. The original design of METALA and its Java interfaces and class hierarchy [2] has proved to be valid for the purposes of meta-learning, incorporating new heterogeneus learning paradigms and now, for adding and testing efficient WM soft computing techniques. Thanks to this design and to the coding facilities provided by J2EE, we were able to perform a fast migration from one to another underlying technology.

2.2 J2EE technology for data analysis

The architecture of METALA is shown in figure 3 (similar to the "Common 3-tier architecture" figure from [3]). At the top of the figure we can see the level of the clients. Here, a client can access to the METALA functionality using different front-ends. Currently, all services of METALA are offered through HTTP from a web portal[1] or pure Java clients. Nevertheless, they could be also Java applets or in PDAs (dashed lines in figure 3 indicate these possibilities).

METALA, as any EJB tool, is composed of three types of beans: *entity beans* are used to interface with the storage medium. An entity bean can pull information from a relational database or some other legacy system containing business data. *Session beans* define the application business logic, or what the application can do given some entity beans to work with. Finally, A *message-driven* bean allows J2EE applications to receive JMS (Java Message Service) messages asynchronously, which can be used to perform different actions.

In figure 3 below the clients we found the business logic level of METALA. It is contained in the Enterprise Java Beans (EJBs) *deployed* in the J2EE application server. All the services we provide implement the needed interfaces and lay on the appropiate *session beans*. All the data needed by the services lay on *entity beans*. Finally, to monitor the progress of the learning experiments we use a *message driven bean* which take cares of storing into the database all generated progress logs, reading them from an asynchronous topic publisher.

The application server provides us automatically with all the needed features of system distribution (with RMI), load balancing, service locating -with JNDI (Java Naming and

[1]Developed using the Tapestry framekork from Jakarta (http://jakarta.apache.org/tapestry/).

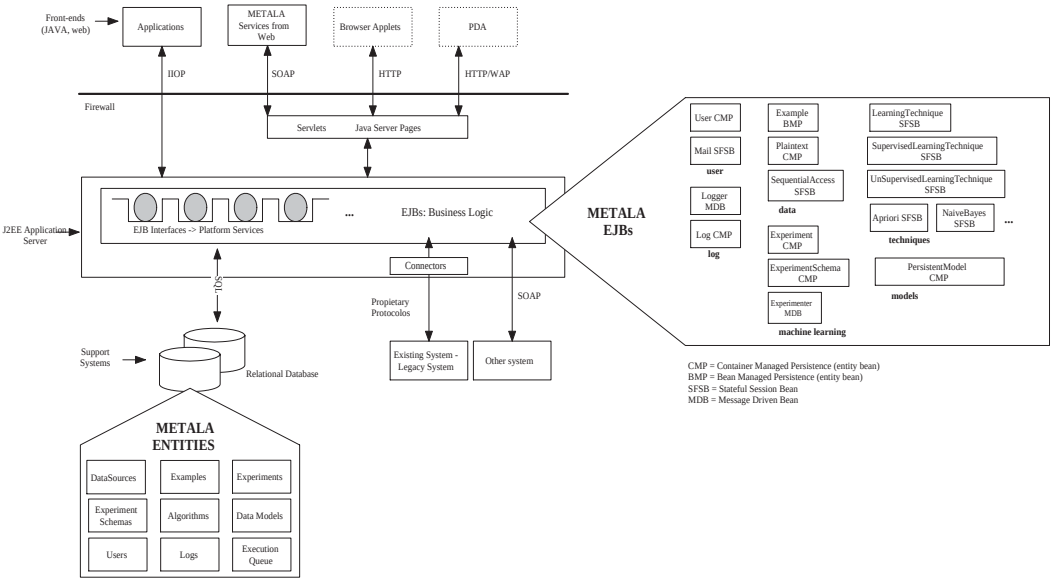


Figure 3: J2EE based architecture of METALA.

Directory Interface)-, database access -with JDBC (Java DataBase Connectivity)-, and some others. Now we do not need to code explicit connections to the database or remote systems, taking care of load balancing, updating the database, thread controlling, etc. We just concentrate on implementating the business logic and let the application server take care of connection, performance and system control issues.

Finally, at the third level, some support systems are needed. Currently we only need the database support given by a relational database but some other systems may easily be connected with the application server thanks to its connectors and to protocols as SOAP (Simple Object Access Protocol).

To facilitate the incorporation of a new learning technique into our architecture we provide a new template or programming **methodology** for adding the learning algorithm, the generable knowledge model and the technique execution progress log type. This is all you need to define in order to to make your technique available through the Internet. Even if you have already programmed your technique in Java, it is very easy to use the template and incorporate the technique to our architecture. If your technique does not belong to any of the paradigms or domain applications existing in the architecture, you may add a new paradigm just coding with pure Java all the features of the new paradigm, forgetting about other underlying-technology related issues, and then incorporate the new technique using the provided methodology. After that, if you want to take even more advantage of your added technique, you may want to create new applications based on the knowledge model generated by your technique. These applications can also be aggregated to our platform by implementing a single interface, and then they will be automatically available as services usable from any browser.

3 Design and Implementation of METALA

As we have seen in figure 3, the METALA system should be considered from two perspectives: clients and business logic. The clients can use METALA from any Internet browser, since the services of METALA are available from web pages. We can develop new Java graphic clients by using the classes of the packages paquetes javax.swing and java.awt, but we are looking for providing the access to METALA from the Web. In section 6 we point out the possibility of describing the services of METALA as real Web Services.

The second part of METALA, the business logic of the system, is obviously the most important one. It is concentrated on the EJBs and classes deployed on JBoss. We can distinguish the following functional parts:

- **user:** it manages users and their permissions on METALA.
- **data:** it deals with data acquisition, data parsing and access to data from original data sources.
- **ml (machine learning):** it defines the learning experiments and the configuration of the associated parameters.
- **techniques:** it contains the techniques of inductive learning, in different paradigms. There are two main types of learning: inductive supervised learning and inductive unsupervised learning (which WUM belongs to).
- **models:** it stores the knowledge models got from the execution of an experiment with some specific parameter for a particular learning technique.
- **log:** it defines the beans and classes associated to the monitor of the progress of the experiments; each generated log is stored in the data base and sent to a *Topic* in order to be immediately consumed by a client who is watching the progress of the experiment.
- **xml:** it describes how to transform any information piece (knowledge models, experiments, learning techniques, etc.) of METALA to XML. Any knowledge related to the learning process must be available in XML, providing communication facilities with other heterogeneus systems and platforms.

Note that, as already pointed, there is a concrete methodology to add new learning services. It is defined upon some Java interfaces, which are separated in packages corresponding to the functional parts described above. They are:

- **Package data:**
  - *AccessFactory:* it provides the static access to the data source; the type of used access (random, sequential, etc.) is specified by the *schema* of the experiment.
  - *DataAccess:* it shows all the methods that can be used when accessing to a particular data source. All the data sources in a typical machine learning application may be divided into a learning part and an evaluation part. The first part of the instances of the data source is used for the learning process, and the second one to estimate the error of the learning process.
  - *DataSet:* it groups the features of any data source of METALA.
  - *DataSetFactory:* it provides in a static way a data source of a specific type. The instances of the data source are stored in the database.

  - *ExampleHandler:* it provides the needed methods to manage the instances of a data source.
  - *Instance:* general definition of any data instance of METALA.

- **Packages ml and techniques:**
  - *LearningTechnique:* general definition of a learning algorithm of METALA.
  - *SupervisedLearningTechnique:* general definition of a supervised learning algorithm.
  - *UnSupervisedLearningTechnique:* general definition of an unsupervised learning algorithm (for example, the ones from WM).
  - *Experiment:* definition of an experiment of a learning algorithm, storing its state, diagnostic, elapsed time, etc.
  - *ExperimentSchema:* configuration of an experiment of a learning algorithm, with information about used parameters, type of algorithm, etc.

- **Package log:**
  - *MLLogValue:* definition of the data unit stored in a logging process along the execution of a learning algorithm of METALA.
  - *LogClient:* needed methods to generate (anotate) a data unit in a logging process.

- **Package models:**
  - *Model:* related to the generic model generated from a learning algorithm, which can be stored in the database and used to make inferences.
  - *Inference:* representation of an inference of a data model, for the inductive learning algorithms which need to make inference.

- **Package xml:**
  - *XMLItem:* needed features to serialize any important element of METALA into a XML document.

3.1 Some implementation decisions

As implementation notes, we should mention that current coding lays on the EJB 2.0 specification[2], and that all created entity beans are of type CMP (Container Managed Persistence). The reason for using CMP is again to delegate to the application server the responsability of choosing the best moment to dump memory information into the database, to roll back a transaction due to an unexpected error, etc. (i.e. all the aspects which fall out of the learning business logic). However, sometimes it is advisable to code "manually" certain parts of a process. For example, the data handling at instance (data sample) level should be extremely efficient to increase overall system performance. The application server and the EJB specification allows this by providing BMP (Bean Managed Persistence) or just programming directly with the underlying technologies (in our case, JDBC) those critical parts as if it was not on any application server. So we can take the best of both programming styles. This is exactly what we have done to develop our software architecture. In this way, the performance with the J2EE application server support (providing enough machine

[2] http://java.sun.com/products/ejb/docs.html

resources for its execution) is not noticeably reduced in comparation with direct technology programming.

We should also mention the METALA class hierarchy, which is based on the data and logic model of [2] and represented by Java interfaces. The interfaces are implemented by entity and session beans, which are placed in the hierarchy providing local and remote interfaces (which are the final services to be used). This hierarchy contains many inheritance relations, which allowed us to define a fairly simple coding methodology.

About the concrete underlying technologies that we use, there are some reasons to choose JBoss as the J2EE application server. First, it is open source, which is an important advantage for us. There exists an important community of developers which use this application server, providing lots of information and documents. There is also a lot of free and purchasable documentation available from the JBoss home page (http://www.jboss.org). Second, this application server is always kept up-to-date, and new versions appear oftenly. Third, JBoss is highly configurable, which allows us to "play" with different configurations, and some of them seem to be more suitable for our software architecture. Besides, it meets the requirements of any powerful application server, such as clustering, last EJB specification compliance, etc. Finally, it is always shipped with the last third party technologies (open source) distributions. However, as JBoss is full J2EE compliant, there should not be important problems to migrate to other J2EE compliant applications servers, if needed. We are using some other tools which aims to facilitate this possible migration, as XDoclet (http://xdoclet.sourceforge.net).

4 A Sample Application

Since the Apriori algorithm is included in the WM domain application of the architecture, we can use its respective user interface from the Internet for testing. Let us check a complete life cycle WUM process:

1. First of all, we choose the WM domain, then the web usage mining approach and after that we select the Apriori algorithm. We then select a new experiment for the algorithm.

2. The service shows which parameters should be configured before the experiment could be launched, with information about how they must be configured. For the preprocessing step [5], we must set the session expiration time. For the algorithm itself, we must provide a minimum support value and a minimum confidence value [1].

3. After configuring the experiment, we launch it. While it is running we can check its progress or test some other service of METALA. We may also leave the tool and check later the conclusion and results of the experiment, or abort its execution.

4. The preprocessing step may also generate readable and storable progress logs or messages, as part of the overall learning technique process. i.e. these logs may be used to follow up the progress and the results of the preprocessing step.

5. When the experiment has finished, we got a knowledge model that can be browsed (or viewed, or whatever the METALA developer consider the best represetantion of its data model) for evaluation. The model is also stored in a XML document for other usages. This model type may also have associated applications. For example, for the WUM approach we have developed an URL recommender application available for all WUM models (see figure 4). If we choose the model obtained from Apriori, the

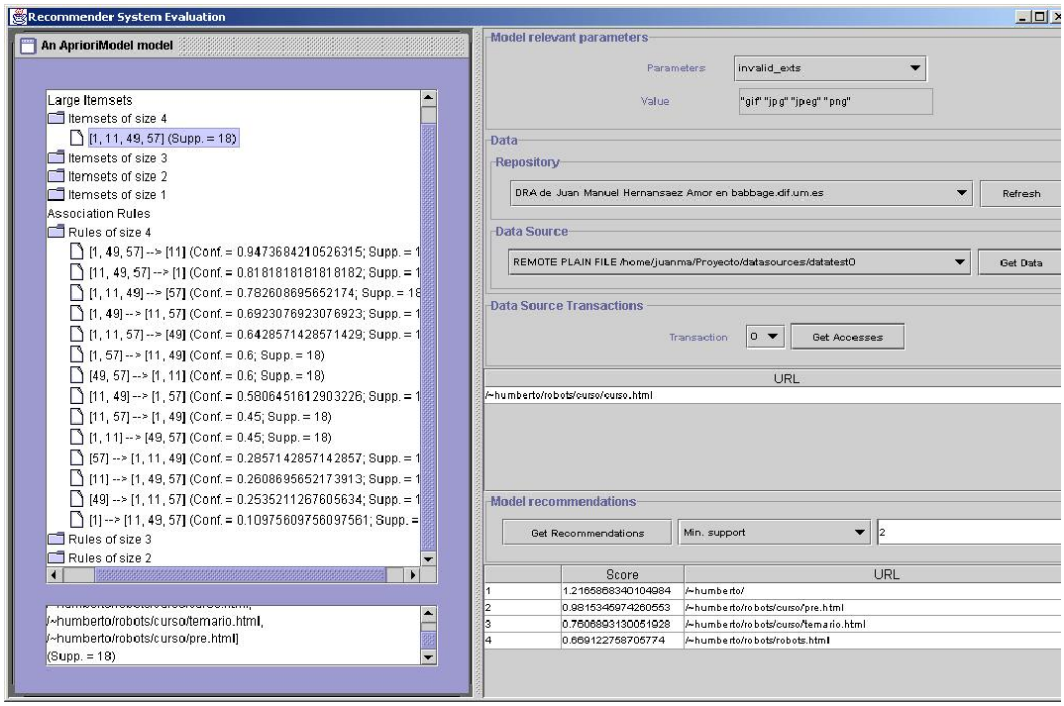


Figure 4: URL recommender based on a WUM knowledge model.

application takes this model as its knowledge source and provides URL recommendations for a given observated user URL traversal path and for given desired values of recommendation support and confidence.

6. Since the configuration of the experiment has been saved in a *experiment schema*, we may repeat the experiment with different parameters values. This way, if we are not satisfied with the model accuracy, we can get another model using less restrictive values for the parameters of the learning technique.

The URL recommender that we have built is available as a pure Java client to be used directly, and also as a Java applet and as a web page to be accessed from any Internet browser. Nevertheless, we are defining all the services of METALA as Web Services (see section 6). At the moment, these services are available as pure Java clients and as web pages of the web portal that we have developed, based on the interfaces described above. Let us review this application as an example of a METALA built-in application. In figure 4 we show the pure Java client version of the application.

As we can see in figure 4, at the left side of the application window we find the representation of an Apriori knowledge model, for the WUM domain application. The application just invokes the *visualize* method of the **Model** interface to load its representation on the left frame. On the right frame we define the control panel of the application itself. Here we can choose the current browsing pattern of an user, give the desired values of support and confidence and finally obtain the recommended URLs basing on the obversed browsing pattern. The recommendations are given with a value score.

5 Comparison with Other Platforms

There are many tools, frameworks, and research initiatives intended for WM. Most of them concentrate on the required WM tasks, providing an acceptable solution for a concrete WM problem [4] or for the problems of a particular WM approach as WUM [13] [9].

Some of them even combine WUM with other WM approaches such as Web Structure Mining [6] or Web Content Mining [10] to aid in a more effective WUM and to provide user profiling, content management and a publishing mechanism for adaptive Web sites. We consider these kinds of general frameworks as the most interesting ones, because they could offer a solution for a global WM process, providing multiple valid applications, and using different learning techniques into the same system.

However, to update these systems is not easy. New techniques and plugins to be added required a system reconsideration and maybe a new system implementation. They did not have the required level of abstraction to deal with extensibility issues, which is one of the strengths of our proposed architecture. Moreover, those systems were very technology-dependant. The METALA underlying J2EE application server represents a relatively modern technology approach to Java programming, which aims to support technology-indepedent Java systems by concentrating on the required business logic of the system.

One of the platforms most similar to METALA is proposed in [11]. In this paper, two architectures - *Whoweda* and *Wiccap*- are joined into a single platform to provide, with the Whoweda system, extraction and retrieval of information from the Web and mining and knowledge discovery; with the Wiccap we obtain XML documents from the knowledge gained from the Web by Whoweda, so that we can get structured information to perform data mining with semi-structured Web data. The Wiccap system can be viewed as an application of the information provided by the Whoweda system. What is more interesting for us is the capability of the Whoweda system of incorporating new algorithms by using some skeleton framework to guarantee a set of interfaces required for inter-component communication.

In our case, METALA is built on a single architecture, and we are not limited to the WM domain. Our framework is thought to support almost any supervised and unsupervised learning paradigm, for educational and research purposes, in a distributed environment for concurrent users. The platform from [11] is regarding data warehousing and concerned about the efficient application of the Wiccap system. Unfortunately, we do not have information about implementation issues of [11], so we cannot stablish more coincidences and differences among both platforms.

Finally, another interesting work in progress with some similarities with our platform can be found in [15]. In this work the authors have implemented several techniques for some WUM problems, as association rules mining (Apriori algorithm) classification (Naive Bayes algorithm) and clustering (k-means, o-cluster). Again, a higher abstraction is missing and coding the different algorithms required some extra effort.

6 Conclusions and Future Work

In this paper we have shown how the J2EE technology can support our soft computing framework, named METALA. In this way we have reached a high abstraction level for easily coding and testing machine learning techniques, new paradigms, and new domain applications as Web Mining (WM). For WM we have tested one of the algorithms of the Web Usage Mining approach to WM, reviewing a complete WM life cycle and providing an URL recommender tool. Any knowledge piece added to METALA (as algorithms, models, applications, etc.) is immediately available through HTTP, thanks to the underlying framework and to

the J2EE technology support.

Researchers interested in testing new algorithms only have to concentrate on the business logic of the new algorithms. There are many chances of easily adding these algorithms to METALA even if they have been already coded outside our framework, on condition that no specific technology (just pure Java) is involved. The coding process of new algorithms, models, etc. has been greatly simplified thanks to the programming **methodology** we provide, which overcomes some of the limitations (specially with inheritance relations) of the J2EE specification.

However, this abstraction capability and coding facility is only the tip of the iceberg. The J2EE application server has many strengths, and one of them is its capability to connect heterogeneous systems. At the present time, we are working on providing support to the paradigm of software agents and multiagent systems [14] in order to take into account user profiles based on stereotypes. In this way we will be able to add adaptability to our tool. Using stereotypes we can define classes of users, so that each user may feel more identified with one or another class. Thus, we can give valuable information about (a) choosing which algorithm may be more convenient for a particular problem and (b) how to present the results. On the other hand, using an **ontology** to model the different algorithms and the adaptation capabilities by means of agents learning, we can provide our tool with a mechanism to progressively execute the learning services more efficiently. To integrate in METALA the needed multiagent support we have at our disposal some alternatives [3]: on one hand, the needed agents may be built using standard EJBs. These agents would be used from the EJB container (variant *EJB interface*). On the other hand, we can use a specific agent container, working together with the EJB container (variant *container interface*). Both solutions have strengths and flaws, which will be analysed in further works.

Another of our work in progress is the developping of a computational **grid**. It is clear that we need a huge computational power to take advantage of the METALA experiments and simulations. Although the application server provides clustering, we need a somehow more abstract view of the METALA services and applications. Moreover, we must include a security infraestructure for our system, such as a public key infrastructure, since we define Web Services available to everyone (using the Web Services Description Language, WSDL), and we also need multi-platform support for the different involved technologies or possible migrations. We are planning to build this grid infrastructure onto our J2EE based architecture, in a non-overlapping way. For this task, we are evaluating the possibility of using the *Globus tookit version 3.0*, an open source implementation of **OGSI** (Open Grid Services Infrastructure [8]).

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] J.A. Botia, A.G. Skarmeta, M. Valdes, and A. Padilla. Metala: A meta-learning architecture. In *Proceeding of the 7th Fuzzy Days*, Dortmund, Germany, October 2001.

[3] S. Brantschen and T. Haas. Agents in a j2ee world. Technical report, Whitestein Technologies AG, 2002.

[4] M-S. Chen, J.S. Park, and P.S. Yu. Efficient data mining for path traversal patterns. *Knowledge and Data Engineering*, 10(2):209–221, 1998.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[6] R. Cooley, P. Tan, and J. Srivastava. Websift: the web site information filter system, 1999.

[7] R. Cooley, P-N. Tan, and J. Srivastava. Discovery of interesting usage patterns from web data. In *WEBKDD*, pages 163–182, 1999.

[8] Global Grid Forum. Open grid services infrastructure (ogsi) version 1.0, 2003.

[9] A. Joshi and R. Krishnapuram. On mining web access logs. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 63–69, 2000.

[10] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[11] K-L. Ong, W-K. Ng, and E-P. Lim. A web mining platform for enhancing knowledge management on the web. In *Proc. of Int. Workshop on Integrating Data Mining and Knowledge Management (in conj. with 1st IEEE Int. Conf. on Data Mining)*, San Jose, California, 2001.

[12] S.K. Pal, V. Talwar, and P. Mitra. Web mining in soft computing framework: Relevance, state of the art and future directions. In *IEEE Transactions on Neural Networks*, volume 13 of *5*, pages 1163–1177, 2002.

[13] M. Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134, 2000.

[14] M. J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley and Sons, New York, USA, 2002.

[15] J. Xu, Y. Huang, and G. Madey. A research support system framework for web data mining. In J. T. Yao and P. Lingras, editors, *WI/IAT 2003 Workshop on Applications, Products and Services of Web-based Support Systems*, Halifax, Canada, 2003.