1 Mobile Data Mining on Small Devices Through Web Services

Domenico Talia[†] and Paolo Trunfio[‡]

DEIS, University of Calabria
Via Pietro Bucci 41C
87036 Rende (CS), Italy

1.1 INTRODUCTION

Analysis of data is a complex process that often involves remote resources (computers, software, databases, files, etc.) and people (analysts, professionals, end users). Recently, distributed data mining techniques are used to analyze dispersed data sets. An advancement in this research area comes from the use of mobile computing technology for supporting new data analysis techniques and new ways to discover knowledge from every place in which people operate.

The availability of client programs on mobile devices that can invoke the remote execution of data mining tasks and show the mining results is a significant added value for nomadic users and organizations that need to perform analysis of data stored in repositories far away from the site where users are working, allowing them to generate knowledge regardless of their physical location.

This chapter discusses pervasive data mining of databases from mobile devices through the use of Web Services. By implementing mobile Web Services we allow remote users to execute data mining tasks from a mobile phone or a PDA and receive on those devices the results of a data analysis task. A prototype based on a J2ME client will be presented, by describing the data selection task, the server invocation mechanisms, and the result presentation on a mobile device.

The chapter is organized as follows. Section 1.2 introduces mobile data mining. Section 1.3 discusses the use of Web Services in mobile environments. Section 1.4 describes the design and implementation of a system we developed for mobile data

[†]E-mail: talia@deis.unical.it
[‡]E-mail: trunfio@deis.unical.it

mining based on the Web Service technology. Section 1.5 summarizes and concludes the chapter.

1.2 MOBILE DATA MINING

The goal of mobile data mining is to provide advanced techniques for the analysis and monitoring of critical data from mobile devices.

Mobile data mining has to face with the typical issues of a distributed data mining environment, with in addition technological constraints such as low-bandwidth networks, reduced storage space, limited battery power, slower processors, and small screens to visualize the results [1].

The mobile data mining field may include several application scenarios in which a mobile device can play the role of data producer, data analyzer, client of remote data miners, or a combination of them. More specifically, we can envision three basic scenarios for mobile data mining:

- The mobile device is used as terminal for ubiquitous access to a remote server that provides some data mining services. In this scenario, the server analyzes data stored in a local or distributed database, and sends the results of the data mining task to the mobile device for its visualization. The system we describe in this chapter is based on this approach.
- Data generated in a mobile context are gathered through a mobile device and sent in a stream to a remote server to be stored into a local database. Data can be periodically analyzed by using specific data mining algorithms and the results used for making decisions about a given purpose.
- Mobile devices are used to perform data mining analysis. Due to the limited computing power and storage space of today's mobile devices, currently it is not realistic to perform the whole data mining task on a small device. However, some steps of a data mining task (i.e, data selection and preprocessing) could be run on small devices.

MobiMine [2] is an example of data mining environment designed for intelligent monitoring of stock market from mobile devices. MobiMine is based on a client-server architecture. The clients, running on mobile devices such as PDAs, monitor a stream of financial data coming through a server. The server collects the stock market data from different Web sources in a database and processes it on a regular basis using several data mining techniques.

The clients query the database for the latest information about quotes and other information. A proxy is used for communication among clients and the database. Thus, when a user has to query the database, she/he sends the query to the proxy which connects to the database, retrieves the results and sends them to the client. To efficiently communicate data mining models over wireless links with limited

bandwidth, MobiMine uses a Fourier-based approach to represent the decision trees, which saves both memory on mobile device and network bandwidth.

Another example of mobile data mining system is proposed in [3]. Such system considers a single logical database that is split into a number of fragments. Each fragment is stored on one or more computers connected by a communication network, either wiredly or wirelessly. Each site is capable of processing user requests that require access to local or remote data.

Users can access corporate data from their mobile devices. Depending on the particular requirements of mobile applications, in some cases the user of a mobile device may log on to a corporate database server and work with data there. In other cases the user may download data and work with it on a mobile device or upload data captured at the remote site to the corporate database. The system defines a distributed algorithm for global association rule mining, which does not need to ship all of local data to one site, thereby not causing excessive network communication cost.

Another promising application of mobile data mining is the analysis of streams of data generated from mobile devices. Some possible scenarios are patient health monitoring, environment surveillance, and sensor networks. The VEhicle DAta Stream mining (VEDAS) system [4] is an example of mobile environment for monitoring and mining vehicle data streams in real time. The system is designed to monitor vehicles using on-board PDA-based systems connected through wireless networks. VEDAS continuously analyzes the data generated by the sensors located on most modern vehicles, identifies the emerging patterns, and reports them to a remote control center over a low-bandwidth wireless network connection. The overall objective of VEDAS is supporting drivers by characterizing their status, and helping the fleet managers by quickly detecting security threats and vehicle problems.

1.3 MOBILE WEB SERVICES

The *Service Oriented Architecture* (*SOA*) model is widely exploited in modern scientific and business-oriented scenarios to implement distributed systems in which applications and components interact each other independently from platforms and languages.

Currently *Web Services* are the most important implementation of the SOA model. Their popularity is mainly due to the adoption of universally accepted Internet technologies such as XML and HTTP. The use of Web Services fosters the integration of distributed applications, processes, and data, optimizing the deployment of systems and improving their efficiency. In particular, integration represents an important competitive factor in Business-to-Business (B2B) scenarios, where information systems can be very heterogeneous and complex.

Recently, a growing interest on the use of Web Services in mobile environments has been registered. *Mobile Web Services* make it possible to integrate mobile devices with server applications running on different platforms, allowing users to access and compose a variety of distributed services from their personal devices.

The remainder of this section discusses the basic characteristics of the SOA model, introduces the main Web Services concepts, and discusses current solutions for the implementation of Web Services in mobile environments.

1.3.1 The Service Oriented Architecture

The *Service Oriented Architecture* (*SOA*) is a model for building flexible, modular, and interoperable software applications. Concepts behind SOA are derived from component-based software, the object-oriented programming, and some other models. The SOA model enables the composition of distributed applications regardless of their implementation details, deployment location, and initial objective of their development. An important principle of service oriented architectures is, in fact, the reuse of software within different applications and processes.

A service oriented architecture is essentially based on a collection of services. A *service* is a software building block capable of fulfilling a given task or business function. It does so by adhering to a well defined interface which specifies required parameters and the nature of the result (a contract between the client of the service and the service itself). A service, along with its interface, must be defined in the most general way to allow utilization in different contexts and for different purposes.

Once defined and deployed, services operate independently of the state of any other service defined within the system. However, service independence does not prohibit to have services cooperating each other to achieve a common goal. The final objective of SOA is to provide for an application architecture where all functions are defined as independent services with well-defined interfaces, which can be called in sequences to form business processes [5].

Service oriented architectures are based on interactions between three roles: *provider*, which is the entity that provides the service; *consumer*, which requires and uses the service; *registry*, which publishes information about available services. Three types of interactions are possible among these roles: a provider publishes information about the service to a registry; a consumer queries the registry to find the desired service; the consumer interacts directly with the service provider.

1.3.2 Web Services

Web Services are an Internet-based implementation of the SOA model. Basically, Web Services are software services that can be described, discovered, and invoked by using XML formalisms and standard Internet protocols such as HTTP [6]. The use of XML as basic language permits to share data independently from underlying platforms and programming languages. At the same time, the use of standard Internet protocols allows to exploit software and hardware infrastructures that are already available for Internet applications such as the Web.

Web Services differs in many respects from classical distributed architecture based on remote components such as RMI, CORBA and DCOM. While Web Services

use a platform-independent formalism for message exchange, classical architectures use low-level binary communications, thus data encoding completely depend from specific technologies.

Another important difference is that Web Services are thought for coarse-grained services, while classical architectures are mainly designed to support fine-grained components. In other terms, Web Services expose their functionalities at a higher level of abstraction, while remote components expose low-level operations that are mainly related to implementation aspects.

Web Services exploit a set of XML-based standard technologies for service description (WSDL), communication between clients and services (SOAP), and service discovery (UDDI).

The *Web Services Description Language* (*WSDL*) [7] is used for describing the interfaces of Web Services. Basically, WSDL allows to describe the operations that are provided by a service, and the input and output messages to be exchanged with the service for each operation.

The *Simple Object Access Protocol* (*SOAP*) [8] defines a standard formalism for exchanging messages between clients and Web Services. SOAP messages can be exchanged over several transport protocols, but typically they are transported by using HTTP.

Finally, *Universal Description, Discovery, and Integration* (UDDI) [9] is a registry for publishing and discovering Web Services. It is used by service providers to publish their Web Services, and by service consumers to locate Web Services on the basis of their interface definition.

1.3.3 Web Services in mobile environments

The market of mobile devices such as smartphones and PDAs is expanding very fast, with new technologies and functionalities appearing every day. Even if such devices share a common set of functionalities, they run on many different platforms, which makes integration with server applications problematic. As in standard wired scenarios, Web Services can be exploited in mobile environments to improve interoperability between clients and server applications independently from the different platforms they execute on.

Basically, there are three architecture models for implementing Web Services in mobile environments [10]:

- a *wireless portal network*;
- a *wireless extended Internet*;
- a *Peer-to-Peer (P2P) network*;

In a wireless portal network there is a gateway between the mobile client and the Web Service provider. The gateway receives the client requests and takes care of

issuing corresponding SOAP requests and returning responses in a specific format supported by the mobile device.

In the wireless extended Internet architecture, mobile clients interact directly with the Web Service provider. In this case mobile clients are true Web Services clients and can send or receive SOAP messages.

Finally, in a P2P network, mobile devices can act both as Web Service clients and providers. This capability of acting both as consumer and provider can be particularly useful in systems such as ad hoc networks. It is not currently implemented in real systems, but it represents the more general model that can offer very interesting opportunities for mobile services in a near future.

In most application scenarios, mobile devices act only as Web Service consumers. In these cases, the choice between the wireless portal network and the wireless extended Internet architecture mainly depends on the level of performance required by the application.

The wireless extended Internet configuration requires mobile devices with XML/SOAP processing capabilities. This introduces additional processing load on the device and some traffic overhead for transporting SOAP messages over the wireless network [11]. While the additional processing load could be negligible in most devices, the traffic overhead can affect response time in presence of wireless connections with limited bandwidth.

On the other hand, a wireless portal network architecture requires the intermediation of a gateway that acts as proxy between client requests and service providers. This allows to use a set of optimizations (e.g., data compression, binary encodings) for reducing the amount of data transferred over the wireless link, but these methods generally depend on the specific structure of data used by the application [10], so its applicability is limited.

Following either the wireless portal network or the wireless extended Internet architecture, some researchers studied how to improve functionalities and performance of Web Services in mobile environments.

Adaçal and Bener [10] proposed an architecture that includes the three standard Web Service roles (provider, consumer, and registry) and three new components: a service broker, a workflow engine, and a mobile Web Service agent. The mobile Web Service agent acts as a gateway to Web Services for mobile devices and manages all communication among mobile devices and the service broker or the workflow engine. The agent, which is located inside the mobile network, receives the input parameters required for service execution from the mobile device and returns the executed service. It also selects services according to user preferences and context information such as location, air-link capacity, or access-network type.

Chu, You and Teng [12] proposed an architecture that divides the application components into two groups: local components, which are executed on the mobile device, and remote components, which are executed on the server side. The system is able to dynamically reconfigure application components for local or remote execution to optimize a utility function derived from the user preferences. This approach

implements a *smart client* model, which is in contrast with that of *thin client* (which is only capable of rendering a user interface) generally implemented in wired scenarios.

Zahreddine and Mahmoud [13] proposed an approach for Web Service composition in which an agent performs the composition on behalf of the mobile user. In the proposed architecture, the client request is sent to a server that creates an agent on behalf of the user. The request is then translated into a workflow to be performed by the agent. The agent looks for services that are published in a UDDI registry, retrieving the locations of multiple services that suit the request requirements. The agent then creates a specific workflow to follow, which entails the agent travelling from one platform to another completing the tasks in the workflow.

Besides these and other research works on architectural aspects, some industries worked on the implementation of a software library, named JSR-172 [14], which provides standard access to Web Services from mobile devices. JSR-172 is available as an additional library for the Java 2 Micro Edition (J2ME) platform [15]; thus, it can be used on mobile devices that support the Java technology.

The main goal of JSR-172 is to enable interoperability of J2ME clients with Web Services. It does so by providing:

- APIs for basic manipulation of structured XML data, based on a subset of standard APIs for XML parsing.
- APIs and conventions for enabling XML-based RPC communication from J2ME, including:
  - definition of a strict subset of the standard WSDL-to-Java mapping, suitable for J2ME;
  - definition of stub APIs based on this mapping for XML-based RPC communication;
  - definition of runtime APIs to support stubs generated according to the mapping above.

Our system, described in the next section, implements a wireless extended Internet architecture. We used the JSR-172 library for the implementation of its client applications.

1.4 SYSTEM DESIGN AND IMPLEMENTATION

In this section we describe the design and implementation of the system. As mentioned before, the goal of the system is supporting mobile data mining on small devices, such as cellular phones or PDAs, through the use of Web Services. First we introduce the system architecture and describe the design of system components. Then, we present the functionality of the system and its implementation.

1.4.1 General architecture

The system is based on the client-server architecture shown in Fig. 1.1.

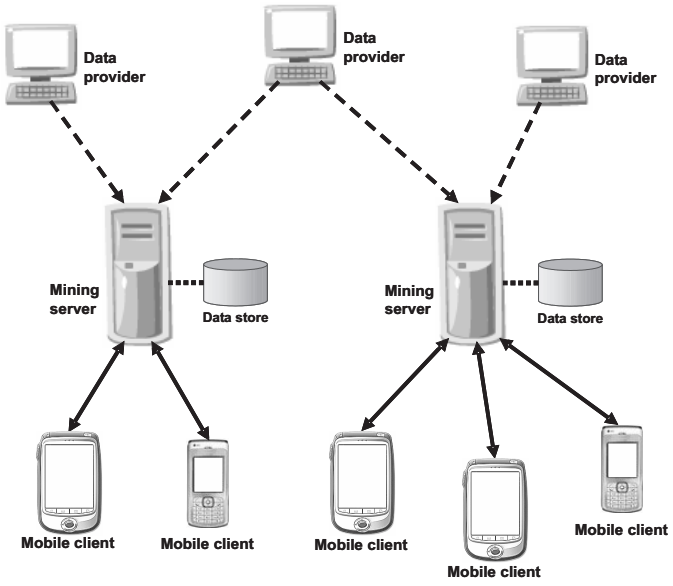

Fig. 1.1 General architecture of the system

The architecture includes three types of components:

- **Data providers**: the applications that generate the data to be mined.
- **Mobile clients**: the applications that require the execution of data mining computations on remote data.
- **Mining servers**: server nodes used for storing the data generated by data providers and for executing the data mining tasks submitted by mobile clients.

As shown in Fig. 1.1, data generated by data providers is collected by a set of mining servers that store it in a local data store. Depending on the application requirements, data coming from a given provider could be stored in more than one mining server.

The main role of mining servers is allowing mobile clients to perform data mining on remote data by using a set of data mining algorithms. Once connected to a given server, the mobile client allows a user to select the remote data to be analyzed and the algorithm to be run. When the data mining task has been completed on the mining server, the results of the computation are visualized on the user device either in textual or visual form.

1.4.2 Software components

In this section we describe the software components of mining servers and mobile clients.

1.4.2.1 Mining server Each mining server exposes its functionalities through two Web Services: the *Data Collection Service* (*DCS*) and the *Data Mining Service* (*DMS*). Fig. 1.2 shows the DCS and DMS and the other software components of a mining server.

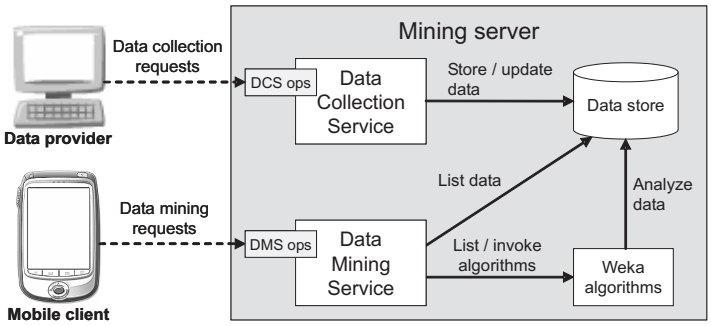

Fig. 1.2 The software components of a mining server

The DCS is invoked by data providers to store data on the server. The DCS interface defines a set of basic operations for uploading a new data set, updating an existing data set with incremental data, or deleting an existing data set. These operations are cumulatively indicated as *DCS ops* in the figure. Data uploaded through the DCS is stored as plain data sets in the local file system. As shown in the figure, the DCS performs either store or update operations on the local data sets in response to data providers requests.

The DMS is invoked by mobile clients to perform data mining tasks. Its interface defines a set of operations (*DMS ops*) that allow to: obtaining the list of the available data sets and algorithms, submitting a data mining task, getting the current status of a computation, and getting the result of a given task. Table 1.1 lists the main operations implemented by the DMS.

The data analysis is performed by the DMS using a subset of the algorithms provided by the Weka library [16], which includes a large collection of machine learning algorithms written in Java for data classification, clustering, association rules discovery, and visualization. When a data mining task is submitted to the DMS, the appropriate algorithm of the Weka library is invoked to analyze the local data set specified by the mobile client.

1.4.2.2 Mobile client The mobile client is composed by three components: the *MIDlet*, the *DMS Stub*, and the *Record Management System* (*RMS*) (see Fig. 1.3)

Table 1.1 Data Mining Service operations

Operation	Description
`listDatasets`	Returns the list of the local data sets.
`listAlgorithms`	Returns the list of the available DM algorithms.
`submitTask`	Submits a DM task for the analysis of a given data set using a specific DM algorithm. Returns a unique *id* for the task.
`getStatus`	Returns the current status of the task with a given *id*. The status of a task can be *running*, *done*, or *failed*.
`getResult`	Returns the result of the task with a given *id*, either in textual or visual form.

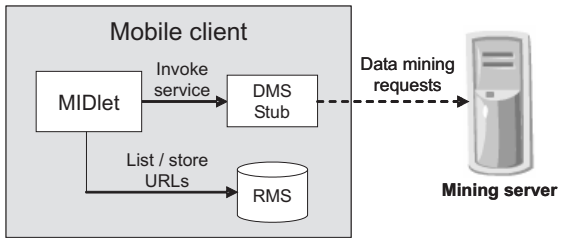

Fig. 1.3 The software components of a mobile client

The MIDlet is a J2ME application allowing the user to perform data mining operations and visualize their results. The DMS Stub is a Web Service stub allowing the MIDlet to invoke the operations of a remote DMS. The stub is generated from the DMS interface to conform with the JSR 172 specifications, introduced in the previous section. Even if the DMS Stub and the MIDlet are two logically separated components, they are distributed and installed as a single J2ME application.

The RMS is a simple record-oriented database that allows J2ME applications to persistently store data across multiple invocations. In our system, the MIDlet uses the RMS to store the URLs of the remote DMSs that can be invoked by the user. The list of URLs stored in the RMS can be updated by the user using a MIDlet functionality.

1.4.3 Functionality of the system

In the following we describe the typical steps that are executed by the client and server components, to perform a data mining task in our system:

1. The user starts the MIDlet on his/her mobile device. After started, the MIDlet accesses the RMS and gets the list of remote mining servers. The list is shown to the user who selects the mining server to connect with.

2. The MIDlet invokes the `listDatasets` and `listAlgorithms` operations of the remote DMS in order to get the lists of data sets and algorithms that are available on the server. The lists are shown to the user who selects the data set to be analyzed and the mining algorithm to be used.

3. The MIDlet invokes the `submitTask` operation of the remote DMS, passing the data set and the algorithm selected by the use with associated parameters. The task is submitted in a batch mode: whenever the task has been submitted, the DMS returns a unique *id* for it, and the connection between client and server is released.

4. After the task submission, the MIDlet monitors its status by querying the DMS. To this end, the MIDlet periodically invokes the `getStatus` operation, which receives the *id* of the task and returns its current status (see Table 1.1). The polling interval is an application parameter that can be set by the user.

5. Whenever the `getStatus` operation returns *done*, the MIDlet invokes the `getResult` operation to receive the result of the data mining analysis. Depending on the type of data mining task, the MIDlet asks the user to choose how to visualize the result of the computation (e.g., pruned tree, confusion matrix, etc.).

1.4.4 Implementation

All the system components, but the Data Collection Service, have been implemented and tested. The mobile client has been implemented using the Sun Java Wireless Toolkit [17], which is a widely adopted suite for the development of J2ME applications. The Data Mining Service has been implemented using Apache Axis [18], an open source Java platform for creating and deploying Web Services applications.

The small size of the screen is one of the main limitations of mobile device applications. In data mining tasks, in particular, a limited screen size can affect the appropriate visualization of complex results representing the discovered model. In our system we overcome this limitation by splitting the result in different parts and allowing a user to select which part to visualize at one time. Moreover, users can choose to visualize the mining model (e.g., a cluster assignment or a decision tree) either in textual form or as an image. In both cases, if the information does not fit the screen size, the user can scroll it by using the normal navigation facilities of the mobile device.

As an example, Fig. 1.4 shows two screenshots of the mobile client taken from a test applications. In this example, the MIDlet is executed on the emulator of the Java Wireless Toolkit, while the Data Mining Service is executed on a server machine using the Web Service container provided by Apache Axis. The screenshot on the left shows the menu for selecting which part of the result of a classification task must be visualized, while the screenshot on the right shows the result, in that case the pruned tree resulting from classification.

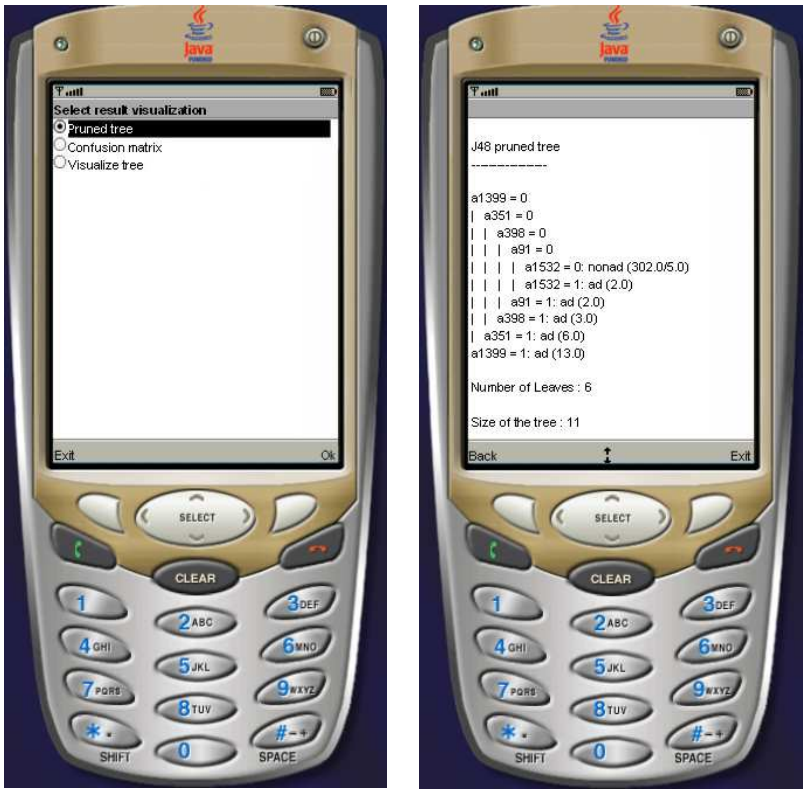

Fig. 1.4 Two screenshots of the client applications running on the emulator of the Sun Java Wireless Toolkit. On the left, the menu for selecting which part of the result must be visualized; on the right, visualization of the selected classification tree.

The system has been tested using some data sets from the the UCI machine learning repository [19], and some data mining algorithms provided by the Weka Library (see Section 1.4.2). Our early experiments show that the system performance depends almost entirely on the computing power of the server on which the data mining task is executed. On the contrary, the overhead due to the communication between MIDlet and Data Mining Service does not affect the execution time in a significantly way, since the amount of data exchanged between client and server is very small. In general, when the data mining task is relatively time consuming, the communication overhead is a a negligible percentage of the overall execution time.

1.5 SUMMARY

This chapter discussed pervasive data mining of databases from mobile devices through the use of Web Services. By implementing mobile Web Services we allow remote users to execute data mining tasks from a mobile phone or a PDA and receive on those devices the results of a data analysis task. A prototype based on a J2ME client has been discussed by describing the data selection task, the server invocation mechanisms, and the result presentation on a mobile device.

Mobile data mining is not yet in a mature phase, however is represent a very promising area for users and professionals that need to analyze data where users, resource and applications are mobile. On the basis of our experience, we can conclude that the combined use of a service-oriented approach with mobile programming technologies makes easier the implementation of mobile knowledge discovery applications that manage heterogeneous and pervasive scenarios where data and processing can migrate across different locations.

REFERENCES

1. S. Pittie, H. Kargupta, B. Park. Dependency detection in MobiMine: a systems perspective. Information Sciences, 155(3-4): 227-243 (2003)

2. H. Kargupta, B. Park, S. Pitties, L. Liu, D. Kushraj, K. Sarkar. Mobimine: monitoring the stock marked from a PDA. ACM SIGKDD Explorations, 3(2): 37-46 (2002)

3. F. Wang, N. Helian, Y. Guo, H. Jin. A Distributed and Mobile Data Mining System. Proc. Int. Conf. on Parallel and Distributed Computing, Applications and Technologies (2003)

4. H. Kargupta, R. Bhargava, K Liu, M. Powers, P. Blair. S. Bushra, J. Dull. VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. Proc. SIAM Data Mining Conference (2003)

5. K. Channabasavaiah, K. Holley, E. M. Tuggle. Migrating to a service-oriented architecture. http://www-106.ibm.com/developerworks/library/ws-migratesoa (visited: Jan. 2007)

6. Web Services Activity. http://www.w3.org/2002/ws (visited: Jan. 2007)

7. Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl (visited: Jan. 2007)

8. Simple Object Access Protocol (SOAP). http://www.w3.org/TR/soap (visited: Jan. 2007)

9. Universal Description, Discovery, and Integration. http://www.uddi.org (visited: Jan. 2007)

10. M. Adaçal, A. B. Bener. Mobile Web Services: A New Agent-Based Framework. IEEE Internet Computing, 10(3): 58-65 (2006)

11. M. Tian, T. Voigt, T. Naumowicz, H. Ritter, J. Schiller. Performance Considerations for Mobile Web Services. Computer Communications, 27(11): 1097-1105 (2004)

12. H. Chu, C. You, C. Teng. Challenges: Wireless Web Services. Proc. Int. Conf. Parallel and Distributed Systems (ICPADS 04), IEEE CS Press (2004)

13. W. Zahreddine, Q. H. Mahmoud. An Agent-based Approach to Composite Mobile Web Services. Proc. Int. Conf. on Advanced Information Networking and Applications (AINA'05), IEEE CS Press (2005)

14. JSR 172: J2ME Web Services Specification. http://jcp.org/en/jsr/detail?id=172 (visited: Jan. 2007)

15. Java Micro Edition. http://java.sun.com/javame (visited: Jan. 2007)

16. H. Witten, E. Frank. Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann (2000)

17. Sun Java Wireless Toolkit. http://java.sun.com/products/sjwtoolkit (visited: Jan. 2007)

18. Apache Axis. ws.apache.org/axis (visited: Jan. 2007)

19. The UCI Machine Learning Repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html (visited: Jan. 2007)