Towards A Reusable And Extensible Adapter Framework For A Data Mining Middleware

Lai Ee Hen\textsuperscript{1} and Sai Peck Lee\textsuperscript{2}

Department of Software Engineering
Faculty of Computer Science & Information Technology
University Malaya
\textsuperscript{1}laiee@perdana.um.edu.my
\textsuperscript{2}saipeck@um.edu.my

Abstract

Data mining has been an effective technique in helping organizations to uncover hidden patterns within the organization’s data in order to gain a competitive edge. With the potential opportunities data mining could bring, it has led to an increasing demand of data mining tools. Data mining tools often face challenges to be designed to cater for a vast option of data sources, data mining techniques and reporting formats in order to support the dynamic changing requirements. New data mining techniques, data sources or reporting may not be able to be supported by some existing data mining tools. Therefore, to support this wide spectrum of options, a data mining tool needs to be designed to be extensible and reusable. Hence, we propose an architecture of a reusable and extensible data mining middleware that supports a wide spectrum of data sources, data mining techniques and reports to help organizations to in decision support.

1. Introduction

Innovation and advancement in information technology has led to extraordinary growth of data scale, which could be in the magnitude of Gigabytes or even Terabytes. Many organizations are facing a lot of challenges in an attempt to uncover valuable information hidden in their organizational repository or database, as the data that has been collected and accumulated since the existence of the organization is too large to be analyzed and interpreted manually. The situation is worsen with the data scattered throughout the organization. The need for integrating this data is a necessity before the analysis stage is performed \cite{2}. Data mining has been an effective technique in helping organizations to uncover hidden patterns within the organization’s data. It is used to help organizations to extract this useful information. Though there is some confusion between Data Mining and Knowledge Discovery in Database (KDD), they are often used interchangeably \cite{2}. Knowledge Discovery in Databases is defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data \cite{3}. Data Mining is actually one of the processes within KDD. It is defined as a process of deriving useful information from a data source through the use of creative queries \cite{1}. Through Data Mining, data collected can help organizations to predict future behaviour, to uncover business opportunities and to optimize success in order to gain an edge over the competitors. Knowing that data mining could help organizations in gaining such quality data, there is a high demand for powerful tools to help in...
interpreting the data into some useful information.

Besides the emergence of many data mining tools, many new data mining techniques have also been developed and evolved to solve different real-world problems. The tremendous growth in the amount and speed of data interchanged nowadays has also triggered an increasing growth of data sources of different formats [4]. It is definitely a challenge for data mining tools to be data sources independence [1] and data mining techniques independence. In this research, we aim to propose a flexible architecture of a data mining middleware that allows data sources, data mining techniques, and reporting independence. The proposed architecture is designed to be a platform independent middleware that allows organizations to mine data through any types of data sources such as relational database, multidimensional database, flat file, hierarchical database, object-oriented database, or XML file, to solve real-world business problems. The architecture also allows the use of any types of data mining techniques for mining useful information and generate into a wide range of reporting results.

However, ensuring such flexibility and reusability in a data mining middleware is a great challenge and careful analysis and design of the system architecture is necessary. Without careful design, it will impact the whole system and this may in turn lead to failure in achieving the implementation objective. To achieve high flexibility and reusability of the middleware, applying design pattern into the middleware system architecture would be a wise move. The reason for this is because patterns provide proven solutions to recurring design problems in a specific context [6]. A design pattern, according to the Gang-of-Four [8], is defined as a description of communicating objects and classes that are customized to solve a general design problem in a particular context. In this paper, we introduce the concept of an adapter framework as the core component in the architecture of our data mining middleware to exhibit the capability of the middleware’s architecture to support independence of data sources, mining techniques as well as reporting formats.

2. Related Work

Based on the META Group’s META spectrum report for data mining, a few data mining tools have been cited as leaders in the data mining market, namely SPSS Clementine, Oracle Data Mining, and SAS Enterprise Miner [5]. Therefore, we limit our scope of study to these few data mining market leaders. In addition, we have also chosen IBM Intelligent Miner and Microsoft Business Intelligence Development Studio that we considered as major data mining tools in our analysis.

We discovered that some of the tools produce better performance on different algorithms and as such, are able to compute better performance attributes. For example, Microsoft Business Intelligence Studio consumes a large amount of disk usage on classification algorithms as compared to the tool like SPSS Clementine but only consumes a small amount of disk usage on segmentation algorithms as compared to the same tool. Such a result might be likely caused by the implementation strategy on the algorithms by each vendor. For example, the vendor like SPSS might be more efficient in implementing classifications as compared to Microsoft. On the other hand, Microsoft might be more efficient in implementing segmentation algorithms as compared to IBM. In short, the implementation strategy of algorithms indirectly affects the performance of data mining tools. A recommended solution would be to allocate, if possible, a major percentage of data mining computations at the memory level to
minimize disk activity and maximize memory activity.

During the course of study, tools such as Microsoft Business Intelligence Development Studio, Oracle Data Miner, and SAS Institute Enterprise Miner only support a predefined set of data sources. These tools require thorough understanding of the data source API specification if new data sources are required. For instance, Oracle Data Miner only supports JDBC compliant driver and Microsoft Business Intelligence Development Studio only supports more ODBC, OLEDB and other types of predefined data sources. Incorporating new data sources is reasonably difficult and often requires a good understanding of the underlying technologies.

SPSS Clementine is partially platform independent as SPSS Clementine releases different binaries on different platforms. Oracle Data Miner too, uses the same binaries on different platforms, and as such, is platform independent. Lastly, Microsoft Business Intelligence Development Studio is platform dependent since the tool depends on the Microsoft .NET framework which currently supports only the Windows platform.

3. Proposed Data Mining Middleware

The proposed data mining middleware is known as Java-Based Data Mining Middleware (JDMM) whose high-level architecture is depicted in Figure 1. It is a server-centric middleware that provides the flexibility in which users are allowed to plug in any types of data mining techniques. JDMM is a platform-, data source-, and data mining technique-independent middleware which is accessible from front-, back- and web-office environments.

JDMM Engine

![Figure 1: JDMM Architecture](image)

JDMM is accessible by three roles of users namely the Business Analysts, Administrators and Implementors. All users access JDMM through the web user interface which is Web JDMM and JDMM Web Configurator. These web applications are deployed in the Tomcat Servlet Container. The JDMM Implementors are the technical users having the authority to develop and plugin all the necessary adapters to connect to different data sources, data mining techniques and reporting formats through JDMM’s Web Configurator. The reason of allowing such flexibility is because users, who wish to discover any hidden knowledge through data mining, are from different industries. Their request of the mined data of course varies. Hence, this flexibility enables us to address the challenge of mining diverse knowledge in databases [9]. Most importantly, each organization, which is from a different background, is the expert in
their own field. They will be in the best position to determine and integrate the type of data mining technique that most suits their organization’s needs.

JDMM Administrator holds the responsibilities of administering all JDMM instances. Business Analyst is the non-technical user who will access JDMM through Web JDMM to extract useful information to help in decision making. All the detailed implementations and plugin of adapters are transparent to the Business Analyst.

Next, we will highlight the information flow of JDMM. Users interact with JDMM through the web application provided. The Adapter Framework enables the retrieval of data from the data sources specified by the user. After data retrieval, relevant data mining models are generated based on a specific data mining technique. The whole mining processes are the integral part of JDMM Core engine. Interpreted data are stored either into a data mining repository or to a persistent data store such as relational database. These options are configurable based on the interest of the users. The primary objective of the repository is to cache results so that computed results are not computed again. The result is a XML file that will then be published and delivered to the Client as either PDF file, XLS file or any proprietary format that has been incorporated into JDMM.

4. Adapter Framework

The Adapter Framework of JDMM allows users to connect to any specific data source to mine data using any specific data mining technique to generate any specific reporting format. The Adapter Framework consists of three main components, namely Data Source Adapter, Data Mining Technique Adapter and Reporting Adapter.

Data Source Adapter accepts a wide range of data sources such as relational databases, object-oriented databases, XML, flat file and so forth. All adapters are added into the adapter framework through some kind of plugin process. Organizations wishing to mine from any new data source in future need only to implement a new adapter to be plugged into the Adapter Framework with minimal configuration.

In order to come up with a framework that allows such flexible extension, we intend to design the Adapter Framework to follow closely the principles of Abstract Factory design pattern. Abstract Factory design pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes [8]. Figure 2 illustrates the class diagram of Data Source Adapter. There are many types of data sources available in the market, ranging from relational database, object oriented database, XML, and so forth. Each of these data sources has different behaviour and has a different set of properties. For example, the standard set of properties for a typical relational database is url, driver, user and password. To promote uniformity among different data source implementations from different vendors, we need to plugin different adapters to these different types of data sources. To make the Adapter Framework extensible, we define an abstract class AbstractDataSourceFactory class which declares a general interface to cater for any data sources. AbstractDataSourceFactory exposes all the properties needed to the client to communicate with different types of data sources such as relational database, object-oriented database, and native xml database. They are not aware of the concrete classes that they are using, such as XmlDSFactory and RdbmsDSFactory, which are the real workhorses that help them to talk to the destination data sources. Through AbstractDataSourceFactory, each client obtains an Entity instance depending on the data source that it is connected to. If a client
talks to a relational database, the instance returned is RdbmsDSEntity. This instance helps to marshal and unmarshal between Java objects and XML. Marshall, as its name implies, is responsible for doing the conversion between Java objects and XML, whereas, unMarshall, is responsible for doing the conversion between XML and Java objects. Concrete classes in the Adapter Framework like RdbmsDSEntity and XmlDSEntity implement the interface AbstractDataSourceEntity to ensure standardization and portability of the adapter framework.

Over time, it is expected that each component will undergo evolutionary adaptation or maintenance [10]. But through this design, any customization in future will not cause the reconstruction of the whole framework. However, customization can easily be done through extending the available component or classes in the adapter framework.

Due to this design pattern, clients are exposed only to AbstractDataSourceFactory and AbstractDataSourceEntity. This simplifies the way in which objects request services from one another because they can use the same message for an entire family of objects. Thus, code maintenance is also simplified as well. Such an approach is called polymorphism and through polymorphism, adapters can be simply plugged into the framework. For example, if organizations wish to communicate with any new data source such as Object-Oriented Database, the general requirement is to develop a plugin adapter (XDSFactory and XDSEntity) to inherit from AbstractDataSourceFactory and AbstractDataSourceEntity. The new plugin is then plugged into the adapter framework highlighted in grey depicted in Figure 2. This class diagram is identical for the Reporting Adapter and Data Mining Technique Adapter depicted in Figure 3 and Figure 4 respectively.

Users from diverse industries will have different interest of knowledge. Providing a limited set of data mining techniques will not fulfil the dynamic changing demand of users. In addition, different data mining techniques will have their specialties to assist the data mining process. To tackle this issue, the proposed architecture of Data Mining Technique Adapter enables users to develop and plugin any types of data mining techniques based on hypothesis testing, time series, normal distribution, binomial distribution, poisson distribution or other types of algorithms.

Figure 2: Data Source Adapter
Lastly, the objective of Reporting Adapter is to generate any types of reporting formats from the mined data. These reporting formats may range from pdf file, Excel spreadsheet, csv file, text file, xml file, html/htm file and other types of format.

According to Marcello Castellano et al, developers often encountered problems with applications that contain mixture of data access code, business logic code, and presentation code. This problem is mainly due to the high coupling of classes which cause difficulties in reusing classes that are highly dependent. Through Model-View-Controller (MVC), this problem can be solved due to the clean separation of the model, view and controller layers. MVC is an architecture design pattern for interactive applications. It is divided into three primary key components namely Model that is responsible for the business domain state knowledge, View that is responsible for the presentation view of the business domain, and Controller that is responsible for controlling the flow of and state of user input [11]. MVC decreases code inter-
dependencies and duplication. In addition, it makes an application more flexible as the componentized behaviour of MVC allows developers to integrate any new components into the existing application. Hence, MVC architectural pattern is used to develop the analysis model of the Data Source Adapter, and Data Mining Technique Reporting Adapter as depicted in Figure 5, Figure 6, and Figure 7 respectively.

For Data Source Adapter, AbstractDataSourceEntity and its corresponding sub-classes such as XmlDSEntity, RdbmsDSEntity and XDSEntity are represented as the Model. AbstractDataSourceEntity will has a variation point available to be extended by variances such as XmlDSEntity, RdbmsDSEntity, and so forth. The main objective of these ‘Entity’ classes is to hold specific business domain data for data mining processes. Client is represented as the View which functions as the caller to different entities such as XmlDSEntity and RdbmsDSEntity (the callee). Finally, AbstractDataSourceFactory and its corresponding child classes such as XmlDSFactory, RdbmsDSFactory and XDSFactory are represented as the Controller. The Controller translates interactions with the Client into actions to be performed by the Entity. AbstractDataSourceFactory has a variation point at which the variances (XmlDSFactory, RdbmsDSFactory and XDSFactory) are to be extended. These ‘Factory’ classes help in instantiating different concrete classes pertaining to different data sources to perform data extraction processes that are distinctive to each data source. The structure of Data Source Adapter is similar to Data Mining Technique Adapter and Reporting Adapter in which the ‘Factory’ classes are represented as the Controller. The Controller in Data Mining Adapter is responsible to perform data mining activities in which potentially useful information from data is being extracted and stored into the Entity. The Entity thus functions as a data store of extracted information to be further interpreted by the Controller in Reporting Adapter. Hence, the Controller in the Reporting Adapter is to manipulate information into meaningful sources such as text HTML reports, non-text HTML reports and Excel spreadsheets.

![Analysis Model of Data Source Adapter](image)

Figure 5: Analysis Model of Data Source Adapter
Through the adoption of different design patterns into the Adapter Framework of JDMM, it enables JDMM to be highly reusable, flexible and maintainable. This flexibility allows JDMM to support unlimited data sources, data mining techniques and reports. Any new data mining techniques, data sources or reports can be plugged into the Adapter Framework by just inheriting from the abstract factory classes. The adapter framework is highly reusable in such a way as for any customization, users are only required to inherit from any of the concrete classes such as XDsFactory, XRdbmsFactory and so forth, instead of creating a new plugin adapter from scratch.

5. Conclusion

A known strength of JDMM adapters is that they are designed based on a pluggable framework. Through employing a pluggable framework, we believe that the JDMM adapters are simple to extend. JDMM APIs is designed to provide a standard interface, allowing third parties to create plugins that interact with JDMM. As such, implementing custom JDMM adapters to extend from JDMM is fairly simple. A major challenge would be to implement a stable API that allows third-party plugins other than JDMM compliant plugins to function as part of JDMM.
Towards a Reusable and Extensible Adapter Framework for a Data Mining Middleware

References


[7] International Conference on Software and Data Technologies, Volume SE, Barcelona, Spain

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, (1994), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series, Chapter 1


