Fernando Berzal, Ignacio Blanco,
Juan-Carlos Cubero, and Nicolas Marin

Component-based Data Mining Frameworks

OLAP Vs. OLTP in the middle tier.

Both researchers and practitioners accept that decision support systems (DSSs) have specific needs that cannot be properly addressed by conventional information systems. Online Transaction Processing (OLTP) systems work with relatively small chunks of information at a time, while DSS applications require the analysis of huge amounts of data. Online Analytical Processing (OLAP) [1], data mining [3] and data warehouses [7] emerged during the last decade in order to fulfill the expectations of executives, managers, and analysts (also known as knowledge workers).

We have also witnessed the flourishing of component-based frameworks during the last few years (see *Communications*' special section on object-oriented application frameworks, Oct. 1997 and *Communications*' special section on component-based enterprise frameworks, Oct. 2000). These frameworks are intended to help developers to build increasingly complex systems, enhancing productivity and promoting component reuse in well-defined patterns. Nowadays those systems are widely used in enterprises throughout the world, but they usually provide only low-level information processing capabilities, since they are OLTP-application-oriented. Here, we propose the development of custom-tailored component-based frameworks to solve DSS problems, although this approach can be extended to a wide range of scientific applications.

A Component-based Data Mining Framework

An open framework for the development of data mining algorithms and DSSs should include capabilities to analyze huge datasets, cluster data, build classification models, and extract associations and patterns from input data. The conceptual model for such a system is shown in Figure 1.

The data miner (the user of the system) has to analyze large datasets and he or she needs to make use of data mining tools to perform a task. Data is gathered and data mining algorithms are used in order to build knowledge models that summarize the input data. Those models may provide the information our user needs, or they may just suggest new ways to explore the available data. Moreover, those knowledge models could be used as input to other mining algorithms in order to solve second-order data mining problems. Both knowledge models and dataset metadata might be stored for later use in the back-end database (for instance, the Object Pool).

Component-based frameworks such as Enterprise JavaBeans (see java.sun.com/products/j2ee) and Microsoft .NET (see www.microsoft.com/com/net) are based on a common architectural pattern, a.k.a. the Enterprise Component Framework [5]. A simplified representation of this pattern is shown in Figure 2. This pattern, modeled as a parameterized collaboration in UML [6], contains six classifier roles which are depicted as rectangles:

- **Client**. Any entity that requests a service from a component in the framework. These requests could be performed using special-purpose data mining query languages, for example, OLE DB for Data Mining (see www.microsoft.com/data/oledb).

  Instead of calling the component directly, the client internally uses a pair of proxies that relay calls from the client to the component. This level of indirection is, however, hidden

PAUL WATSON

from the client perspective; it makes location transparency possible and, when needed, it supports message interception.

- **Factory proxy**. Performs object factory operations, common to all framework components (such as create or find) and facilitates class methods.
- **Remote proxy**. Handles operations specific to each kind of component (such as inspection, parameter setting, and so forth) and facilitates instance methods.
- **Component**. Both datasets and knowledge models are components in a data mining framework. Data mining algorithms could also be considered as components on their own, but they are just used through factory proxies to build knowledge models.
- **Container**. Represents the framework's runtime environment, and holds the components and both proxy roles. This containment is shown by aggregation links in Figure 2.

The container supports distributed computing services, such as security, interprocess communication, persistence, and hot deployment. Transactions are also supported by enterprise frameworks (such as EJB/.NET frameworks) but are not needed in a data mining environment. Such an environment, however, needs scheduling, monitoring, and notification mechanisms to manage data mining tasks. Commercial application servers may be suitable for e-business applications, but they lack the stricter control of computing resources data mining systems require. This shortage is also applicable to a wide range of scientific applications by extension. Custom-tailored frameworks should include capabilities such as CPU/memory/database usage monitoring, resource discovery, reconfigurable load balancing, and a higher degree of freedom for programmers to implement distributed algorithms.

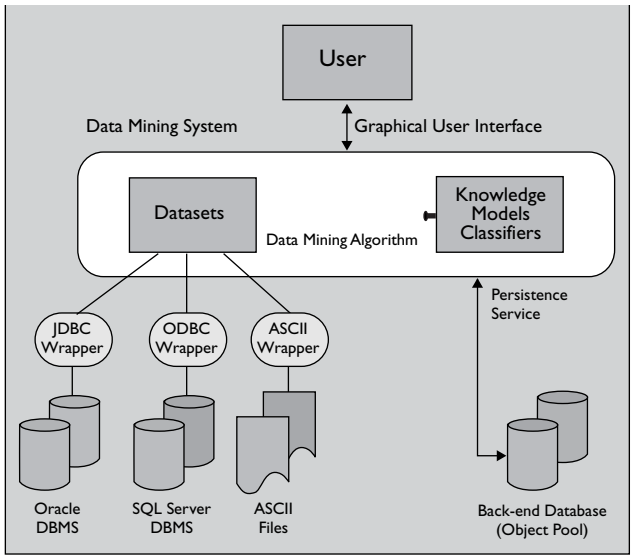

Figure 1. A conceptual model for DSSs.

- **Persistence service**. Permits the storage and retrieval of framework components. This persistence service can be managed and coordinated by the container. Dataset metadata, discovered knowledge models, and user session information are all candidates to be saved for future use in the back-end database (the Object Pool). This back-end database is also useful to provide a reliable computing environment (preserving the system state against power outages, for example).

Design Principles

We believe every component-based, data mining framework should focus its design efforts into two major objectives:

- *Transparency*. Both for users and programmers. Users do not need to be aware of the underlying complexity of the system, while programmers should be able to create new framework components just by implementing a core set of well-defined interfaces.
- *Usability* [4]. As any other software system, data mining systems are used by people and system usability is critical for user acceptance, provided that knowledge workers are not usually knowledgeable about computers. Users should be able to store anything they can reuse in future sessions, or even share the information they obtain. This groupware focus is especially important in data mining applications, where the discovered knowledge must be properly represented and communicated.

Component-based Dataset Modeling

Let us consider, for example, the case of assembling the datasets

that are used as input to build knowledge models. These datasets may come from heterogeneous information sources.

Data mining tools usually work with tables in the relational sense. Each table contains a set of fixed-width tuples that can be obtained either from relational databases or any other information source (ASCII or XML files, for example).

All tabular datasets have a set of columns (also called attributes). Each one of them has a unique identifier and an associated data type (strings, numbers, dates, and so forth). A flexible tool should allow the specification of order relationships among attribute values and the grouping of attribute values to define concept hierarchies.

A data mining system should also be capable of performing heterogeneous queries over different databases and information sources. The independently retrieved datasets, in fact, might be processed further in order to join them with other datasets (data integration), to standardize concept representations and eliminate redundancies (data cleaning), to compute aggregations (data summarizing), or just to discard part of them (data filtering).

All the aforementioned operations involving datasets can be performed using powerful formal models and query languages. However, typical users are not prepared to use such models and languages to define the customized datasets they need. They would probably reject a system that requires them to learn any complex formalism. In order to improve system acceptance, we propose a bottom-up approach. A family of dataset-building components should provide users with all the primitives they need to build their own datasets from the available data sources:

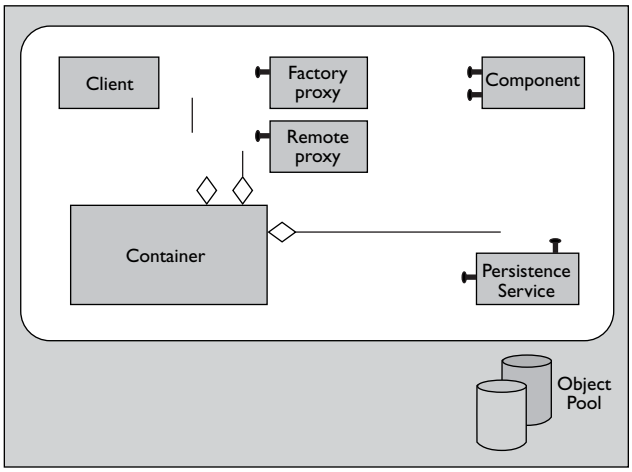

Figure 2. A simplified representation of the Enterprise Component Framework.

- **Wrappers** are responsible for providing uniform access to different data sources. Data stored as sets of tables in relational databases can be retrieved performing standard SQL queries through call-level interfaces such as JDBC and ODBC. Data stored in other formats (locally as ASCII/XML files or remotely in DSTP servers [see www.ncdm.iuc.edu/ dstp], for example) require specific wrappers. In fact, any accessible information source requires its own suitable wrapper.
- **Joiners** are used to join multiple datasets. They allow the user to combine information coming from different sources. Joiners are also useful to include lookup fields into a given dataset (as in data warehouse star schemas) and to specify relationships between two datasets from the same source (for example, master/detail relationships).
- **Aggregators** summarize datasets in order to provide a higher-level view of the available data. Aggregations are useful in a wide range of OLAP applications, where trends are much more interesting than particular details. Common aggregation functions include **MAX, MIN, TOP, BOTTOM, COUNT, SUM,** and **AVG**.
- **Filters** perform a selection over the input dataset to obtain subsets of the original input dataset. In data mining applications, filters can be used to perform samplings, to build training datasets (such as when using cross-validation in classification problems), or just to select the data we are interested in for further processing.
- **Transformers** are also needed to modify dataset columns.

Encoders are used to encode input data, for example, providing a uniform encoding schema to deal with data coming from different sources. Encoders are useful to ensure that real-world entities are always represented in the same

way, even when represented differently in different data sources. In fact, the same entity could even have several representations in a given data source. This kind of component is useful for data cleaning and integration.

Extenders are used to append new fields to a given dataset. Those fields, known as calculated fields, are useful for managing dates and converting measurement units. The value of a calculated field is completely determined by the other field values in the same tuple. A calculated field could be specified using a simple arithmetic expression (including operators such as +, -, *, and /) or even more complicated algorithms (involving *if* statements and table lookups, for example).

These components can be combined easily in tree-like structures to build highly personalized datasets. These datasets are amenable to standard query optimization techniques, therefore improving system performance. Using our approach, even computer-illiterate users are able to use complex data mining systems by linking dataset modeling components.

COMMERCIAL ENTERPRISE APPLICATION servers (the containers in the framework pattern) are currently restricted to OLTP applications, and we believe it is time for system architects to focus on higher-level information processing capabilities in order to take advantage of the vast computing resources available with current corporate intranets. Although we have proposed a component-based framework model for building a data mining system here, our approach is extendible to any CPU-intensive computing application. C

REFERENCES

1. Chaudhuri, S. and Dayal, U. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, Mar. 1997.
2. Codd, E.F., Codd, S.B., and Salley, C.T. *Providing OLAP to User-Analysts: An IT Mandate*. Hyperion Solutions Corporation, Sunnyvale, CA., 1998; www.hyperion.com/products/whitepapers.
3. Han, J. and Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2000.
4. Juristo, N., Windl, H. and Constantine, L. (Eds.) Usability engineering. *IEEE Software 18*, 1 (Jan./Feb. 2001).
5. Kobryn, C. Modeling components and frameworks with UML. *Commun. ACM 43*, 10, (Oct. 2000), 31–38.
6. Object Management Group. *OMG Unified Modeling Language Specification*, 1.4, 1st. Ed. (Sept. 2001); www.omg.org/technology/uml/.
7. Widom, J. Research problems in data warehousing. In *Proceedings of the 1995 International Conference on Information and Knowledge Management* (CIKM'95), Nov. 29–Dec. 2, 1995, Baltimore, MD.

FERNANDO BERZAL (berzal@acm.org) is a researcher at the Intelligent Databases and Information Systems research group in the Department of Computer Science and Artificial Intelligence at the University of Granada, Spain.

IGNACIO BLANCO (iblanco@ual.es) is an assistant professor at the University of Almeria, Spain.

JUAN-CARLOS CUBERO (jc.cubero@decsai.ugr.es) is an associate professor in the Department of Computer Science and Artificial Intelligence at the University of Granada, Spain.

NICOLAS MARIN (nicm@descsai.ugr.es) is an assistant professor in the Department of Computer Science and Artificial Intelligence at the University of Granada, Spain.