Expert Systems with Applications 36 (2009) 7882–7887

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

The design and use of the TMiner component-based data mining framework

Fernando Berzal *, Juan-Carlos Cubero, Aída Jiménez

IDBIS Research Group, Department of Computer Science and Artificial Intelligence, ETSIIT, University of Granada, Granada 18071, Spain

ARTICLE INFO

Keywords:
Data mining
Component-based systems
Application frameworks
Software architecture
Design patterns
Design guidelines
Usage modes

ABSTRACT

This paper provides some practical guidelines for the design of data mining frameworks. It describes the rationale behind some of the key design decisions that guided the design, development, and implementation of the TMiner component-based data mining framework. TMiner is a flexible framework that can be used as a stand-alone tool or integrated into larger business intelligence (BI) solutions. TMiner is a general-purpose component-based system designed to support the whole KDD process into a single framework and thus facilitate the implementation of complex data mining scenarios.

1. Introduction

Traditional on-line transaction processing systems, also known as OLTP systems, work with relatively small chunks of data at a time, while on-line analytical processing systems, or OLAP systems, require the analysis of huge amounts of data (Chaudhuri & Dayal, 1997). It comes as no surprise that OLAP systems have very specific needs that conventional application frameworks do not properly address.

This fact has led to the development of data mining (Tan, Steinbach, & Kumar, 2006; Han & Kamber, 2006) and data warehousing (Widom, 1995; Kimball & Ross, 2002), which try to satisfy the expectations of the so-called knowledge workers (executives, managers, and analysts).

This paper describes the rationale behind some key design decisions that led to the development of a component-based data mining framework called TMiner. As we will see, TMiner can be used as a flexible stand-alone data mining tool, but it has also been designed so that it can be easily incorporated into larger business intelligence solutions.

It should be noted that the tools and techniques TMiner collects somewhat overlap with existing Machine Learning algorithm collections, such as Weka (Witten & Frank, 2005). However, TMiner is more that a mere collection of independent algorithms for data mining tasks that can be directly applied on prepared datasets or invoked from your own code.

Some open-source and commercial data mining libraries (Prudsys, 2008; Rapid-I, 2008) include facilities for their integration into actual enterprise systems. TMiner also provides usage modes specially designed for its tight integration into larger solutions.

TMiner is a general-purpose component-based system designed to support the whole KDD process into a single framework and thus facilitate the implementation of complex data mining scenarios. In this sense, TMiner is designed to be useful in a wide variety of application domains, in sharp contrast to domain-specific data mining systems such as iKDD or SA. While the interactive knowledge discovery and data mining system, iKDD, was designed for particular bioinformatics-related problems (Etienne, Wachmann, & Zhang, 2006), Perttu Laurinen's Smart Archive, SA, has been proposed for implementing data mining applications using data streams (Laurinen, Tuovinen, & Roning, 2005).

The rest of our paper is organized as follows. Section 2 describes the architectural design of the TMiner framework and its component model. Section 3 describes the facilities TMiner offers for different usage scenarios, from the casual user who wants to perform simple data analysis tasks and the researcher who needs a more thorough experimentation, to the systems integrator who needs to incorporate data mining features into final solutions. Finally, Section 4 concludes our paper with some comments on the current status of TMiner and our expectations for its future.

2. TMiner component model

This section describes the TMiner component model, what TMiner components look like and the basic infrastructure provided by the TMiner framework for the design and use of new components.

2.1. TMiner components

A software component is "a unit of composition with contractually defined interfaces and explicit context dependencies" (Szyperski, Gruntz, & Murer, 2002). These context dependencies are specified by stating the required interfaces and the acceptable execution platform(s) for the software component. TMiner compo-

* Corresponding author.
E-mail addresses: fberzal@decsai.ugr.es (F. Berzal), jc.cubero@decsai.ugr.es (J.-C. Cubero), aidajm@decsai.ugr.es (A. Jiménez).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.11.033

nents have well-defined I/O ports whose specific properties can be easily consulted by means of the `TMinerComponentMetadata` object that is attached to every `TMinerComponent` (see Fig. 1).

Software components are units of independent deployment, in contrast to objects in object-oriented programming (OOP), which are mere units of instantiation. Components are also units of third-party composition and they typically have no (externally) observable state. However, as objects in OOP, they encapsulate their state and behavior.

Obviously, components still act through objects in object-oriented systems, but they do not only contain classes. They can also contain additional resources. In TMiner, components must include a component descriptor describing its I/O ports and any additional metadata that might be required for the component to be used in practice. Fig. 2 shows an example of such a descriptor in XML format. The component descriptor includes the names, textual descriptions, and required interfaces for all the component I/O ports, as well as default values for component parameters. This information is extremely useful, for instance, in the automatic generation of the user interface for data mining tools (component developers do not need to worry about user interface issues and they can just focus on the development of the component themselves).

In a data mining framework such as TMiner, the user has to analyze large datasets with the help of mining tools and techniques. Data is gathered from different data sources and data mining algorithms are used in order to build knowledge models (Berzal, Blanco, Cubero, & Marín, 2002). Hence TMiner components fall into two main categories:

- `TMinerModels` represent the entities data miners work with. They may provide the information our user needs to access different data sources (i.e. dataset metadata). They can also be descriptive or predictive models built from those data sources. They can even be used as the input to other mining algorithms in order to solve second-order data mining problems.
- `TMinerTasks` represent the tasks data miners must perform to analyze data. They are the active objects that users need to build new models.

All components in TMiner must be serializable (i.e. they can be stored as byte sequences for later reconstruction). This is necessary because `TMinerModels` must be stored for later use, while `TMinerTasks` might need to be transferred to different processing nodes in a distributed computing environment.

Additionally, TMiner core classes include some standardized interfaces designed to simplify the representation, distribution, and use of the models discovered in the KDD process. For instance, `XMLable` components can be represented as XML documents, hence facilitating their storage for later use, as well as their visualization in standard web browsers (with the help of the corresponding XSLT style sheets). Likewise `SQLable` components, such as many different kinds of symbolic classification models, can be converted into SQL scripts that might be invaluable in practice, since they provide a very convenient method for using such models on relational databases.

Any information system can be described by a structure, a mechanism, and a policy following Perry and Kaiser's SMP model (Perry & Kaiser, 1991). In our case, TMiner models determine the structure of the system. TMiner tasks, which are responsible for the implementation of data mining techniques, are the mechanisms that let us solve data mining problems. Finally, the set of rules and strategies imposed by the system environment are used to establish its usage and security policies. This is the job of the TMiner framework we now proceed to describe.

2.2. The TMiner framework

A component framework is "a collection of rules and interfaces (contracts) that govern the interaction of components plugged into the framework" (Szyperski et al., 2002). Component frameworks can also be seen as components that plug into higher-level component frameworks (e.g. when integrated into larger solutions).

Component-based frameworks are intended to help developers to build increasingly complex systems, enhance productivity and promote component reuse by means of well-defined patterns (Fayad & Schmidt, 1997; Larsen, 2000). Such frameworks are widely-used in enterprise information systems, but they usually only provide low-level information processing capabilities, since they are OLTP-application-oriented. In contrast, TMiner is a component-based framework that has been custom-tailored to solve decision support problems, even though it should be noted that its approach could also be of use in a wide range of scientific computing applications.

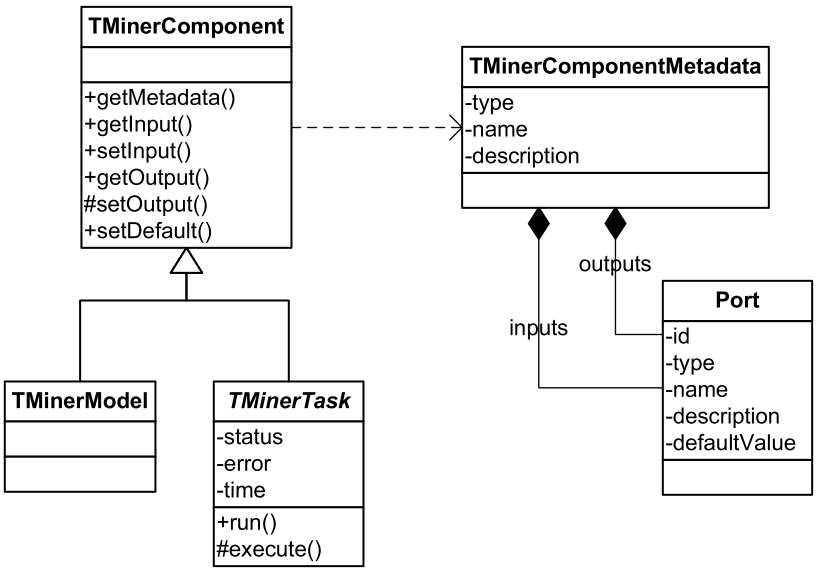

Fig. 1. TMiner component model base classes.