# DAMISYS: An Overview

M.C.Fernández, O.Delgado, J.I.López, M.A.Luna, J.F.Martínez, J.F.B.Pardo, and J.M.Peña

Department of Computer Science, Universidad Politécnica de Madrid,
Boadilla del Monte, 28660 Madrid, Spain
cfbaizan@fi.upm.es, rsdm-group@nova.ls.fi.upm.es

**Abstract.** Since KDD first appeared the research has been mainly focused on the development of efficient algorithms to extract hidden knowledge. As a result, a lot of systems have been implemented during the last decade. A common feature of these systems is that they either implement a specific algorithm or they are specific for a certain domain. As new algorithms are designed, existing systems have to be adapted, which means both redesigning and recompiling. Consequently, there is an urgent need to design and implement systems in which adding new algorithms or enhancing existing ones does not require recompiling and/or redesigning the whole system. In this paper we present the design and implementation of DAMISYS (*DAta MIning SYStem*). The innovative factor of DAMISYS is that it is an engine of KDD algorithms which means that it is able to run different algorithms that are loaded dynamicly during runtime. Another important feature of the system is that it makes possible to interact with any Data Warehouse, due to the connection subsytem that has been added.

## 1 DAMISYS

The lack of easibily extendible systems integrated with *Data Warehouses* motivated out research in which the main goal was to design a system that had the features of **Extensibility**, **Code reusability**, **GUI independence**, **DBMS independence**, **Data base integration** and **Optimization support**.

In this context, the term **extensibility** means the capability to add, delete and/or update the set of algorithms the system can execute. DAMISYS is a system in which adding new algorithms does not involve either redesigning or compilating the system.

Studying in detail data mining algorithms [4] it is straightforward to see that they share some functions. Division of algorithms in basic operations makes it possible to interchange operations among different algorithms. This allows us to provide **code reusability**. Another goal DAMISYS achieves is **GUI independence**. This means that functions like user query requests, administrative tasks and system monitorization are controled by different applications using the same communication protocol.

**DBMS independence** allows DAMISYS to use multiple data repository architectures. rom now on the term data repository will be used to name the

Mukesh Mohania and A Min Tjoa (Eds.): DaWaK'99, LNCS 1676, pp. 313-317, 1999

element that holds and manages (data storage, recovery, update and query) the data we want to analyze.

Data repository services give DAMISYS the capability to use these systems to store final/intermediate results permanently/temporally. There is also other useful information, like different preprocessing results from the same original data, that can be stored to reduce system response time and to rise system performance. We have called this use of data repositories **data base integration**. DAMISYS also implements a series of mechanisms to support future optimization policies. Some of these mechanisms are: algorithm division in basic operations, intermediate result management, parallel algorithm execution, to name a few. [2].

## 2 Architecture

DAMISYS architecture has two levels of division. The first level defines a number of *subsystems*. Each of these subsystems is subdivided, in a second level, into different *modules*. We call *subsystem* to each of the components of our design that is executed in parallel with other subsystems and performs some general system functions. Any of these subsystems could be run concurrently in multiple processors in a parallel shared-memory computer.

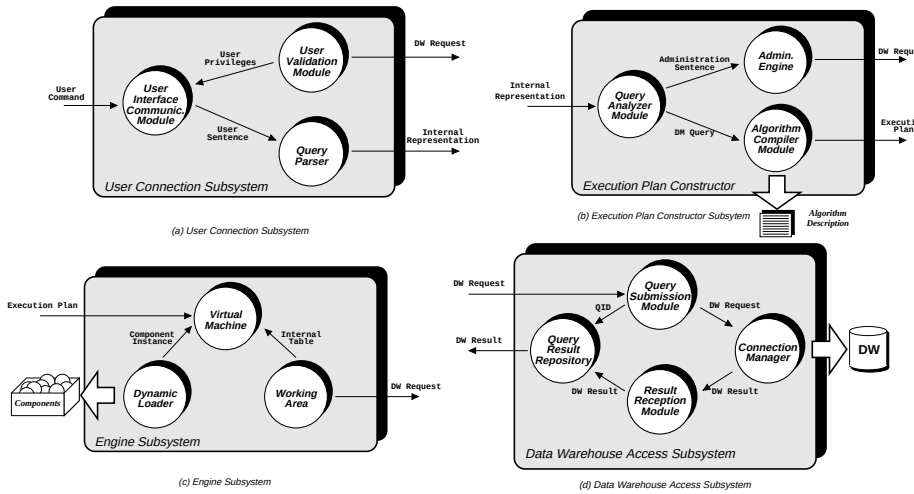


**Fig. 1.** *User Connection Subsystem*

On the other hand, *modules* achieve specific operations. The aim of this group of specific operations is to perform the general functions provided by the subsystem to others subsystems or to any external program. The differences between subsystem and module concept is that the former must be executed

in parallel with other subsystems and performs general features; the modules are not necessary concurrent components and they deal with specific operations. Each of the modules belongs to a unique subsystem and provides specific mechanisms to achieve final subsystem tasks. The architecture proposed divides DAMISYS system in four different subsystems: *User Connection Subsystem*, *Execution Plan Constructor*, *Engine Subsystem*, *Data Warehouse Access Subsystem* As a new user query is received by the *User Connection Subsystem* it is translated into an internal DAMISYS format (*Internal Representation*). The *Execution Plan Constructor* processes the query and defines how to solve it by means of a structure called *Execution Plan*. *Esecution Plans* describe which algorithms will be used to solve the query and the values of algorithm parameters. The *Engine Subsystem* takes an *Execution Plan* and executes it. The tasks described in an *Execution Plan* are divided into a series of specific transformations and functions that are called *Basic Operations*. Finally, any of the subsystems may require data from the Data Warehouse supporting the system (in order to execute algorithms) This service is provided by the *Data Warehouse Access Subsystem* that makes it possible to connect DAMISYS to any Data Warehouse system.

### 2.1 User Connection Subsystem

It provides communication services to GUI external applications and it transforms messages sent by these applications into an *Internal Representation* This subsystem also provides user validation and role checking each time a user connection is established. User interfaces do not need to be executed on the same machine where the DAMISYS is running, as its communication interface is able to provide remote request submission, as well as concurrent user interfaces connection. *User Connection Subsystem* has been divided into three modules (see Figure 1(a)):

*User Interface Communication Module* controls the information exchanged between DAMISYS and remote GUIs. It provides abstract interface functions that hides protocol-dependent implementations. On the other hand, *Query Parser* analyses and checks lexical and syntax sentence construction and translates it into a DAMISYS format. Semantical checking is performed by another module.

### 2.2 Execution Plan Constructor

This subsystem creates *Execution Plans* from the *Internal Representation* of user sentences using a high description of the algorithm. The *Execution Plan Constructor* has been structured in three modules (see Figure 1(b)):

The service offered by this subsystem starts when an *Internal Representation* from *User Connection Subsystem* is submitted to *Query Analyzer Module*: In case of an administration command, it is sent to the *Administration Engine*, otherwise it is compiled by the *Algorithm Compiler Module* that uses a high level description of the appropriate algorithm, sets the values of the

algorithm parameters, and finally, this module submits an *Execution Plan* to the *Engine Subsystem* and requests its execution service[1].

### 2.3 Engine Subsystem

This is the most important component of DAMISYS architecture because this subsystem deals with the resolution of *Execution Plans*. This function is achieved by translating them into a chain of transformations that are implemented in *components* that are loaded dynamicly. These component executions are called *instances*. This subsystem is composed by the following modules: **Virtual Machine**, **Dynamic Loader** and **Working Area**.

Once the *Execution Plan* is obtained, the engine performs a series of steps in order to get a chain of *Basic Operations* ready to be started to run inside *Virtual Machine* module the engine. Thus, in this module, *Execution Plans* are read and interpreted, to obtain the group of *Basic Operation* which are needed to execute the algorithm. All this process requires the next steps: *Execution Plan* interpretation; Construction of *Basic Operations* chain; Execution of the chain; and Result returning. The *Working Area* contains the *internal data components* and their manager as well as different system resources. In order to be able to manage the amounts of data used and created into *Engine Subsystem* some structures are required. These structures are called *Internal Tables*, and represent data base tables, which are read and written by the *Basic Operations* chains. The *Internal Tables* are stored in the *Working Area*. The main feature of *Internal Tables*, from the point of view of memory usage, is the pages division. *Dynamic Loader* module offers *instances* of *Basic Operations* of algorithms. The load of basic components of an algorithm is done when system needs its execution. Loader module disposes of a Basic Components Cache where it sets those components loaded at that moment into system.

### 2.4 Data Warehouse Access Subsystem

This module provides a common communication method between DAMISYS and any Data Warehouse system. This subsystem is divided into four main modules: **Query Submission Module**, **Query Result Receiver**, **Query Result Repository** and **Connection Manager**. *Query Submission Module* receives the requests and processes them before sending commands to the Data Warehouse system. Queries are temporally stored in the *Query Result Repository* *Query Submission Module* runs without interruption and it does not wait for query results. As a consequence, multiples queries could be solved in parallel.

When an answer is received from the Data Warehouse *Query Result Reception Module* matchs this answer with the request stored in *Query Result Repository* and submits the message to the requester subsystem.

[1] In order to apply a specific algorithm this module has to compile its high level description. This algorithm description is defined using *DAMISYS/ALG* language. *DAMISYS/ALG* grammar has a C++-like syntax with some simplifications. Detailed syntax of this language is a broad topic to be completely described in this paper.

Finally, *Connection Manager* provides a series of functionalities that allow to the other modules to access the Data Warehouse services. This module implements the abstract interface between DAMISYS system and the data source. This function avoids direct interaction among the rest of the modules of this subsystem and the specific protocol required for a particular Data Warehouse architecture in a concrete configuration[2].

## 3 Conclusions and Future Work

All the objectives proposed in the section 1 has been completely achieved.

Although optimization mechanisms are implemented, there are only some naive optimization policies developed. Our research is now focused on provide more complex and useful policies that may enhance DAMISYS system performance. The addition of new policies does not require a new design of any of the subsystems, because new policies only need subsystem mechanisms to perform their action, and these mechanisms are already available.

TCP/IP protocol may be translated into CORBA communication. This change could be performed to interconnect DAMISYS system with GUI applications and Data Warehousing system, as well as, to distribute DAMISYS subsystems among different computers.

## References

1. Fayyad, U.M.; Djorgovski, S.G.: Automating the Analysis and Cataloging of Sky Surveys. Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1.996: 471–493
2. Graefe, G.: Volcano, an Extensible and Parallel Dataflow Query Processing System. IEEE Trans. on Knowledge and Data Eng., 1.994: 120–135
3. Holsheimer, M.; Kersten, M.L.: Architectural Support for Data Mining. Technical Report, CWI, Number CS-R9429, 1.994
4. Menasalvas, E.: Integrating Relational Databases and KDD Process: Mathematical Modelization of the Data Mining Step of the Process. Phd Thesis, disserted Politachnical University (UPM), Spain, 1.998
5. Matheus, C.J.; Piatesky-Shaphiro, G.: Selecting and Reporting What is Interesting: The KEFIR Application to Heatlhcare Data. Advances in Knowledge discovery and Data Mining, AAAI/MIT Press, 1.996: 399–421

[2] The basic protocol calls are implemented by specific Data Warehouse Drivers (DWD).