An Information System for Quality Control in Wineries

M. Urbano Cuadrado, I. Luque Ruiz* and M.A. Gómez-Nieto

Department of Computing and Numerical Analysis. University of Córdoba. Campus de Rabanales, E-14071 Córdoba, Spain

Abstract: A system for the overall management of the information related to analytical processes and quality control in wineries is presented. It enables the integration of semi-automated and automated analytical processes. It has been developed in Java using the database management system Oracle 9i and can be executed both as a stand-alone program and through a standard Internet browser. It has been developed under the evolutionary incremental paradigm in order to take into account users' requirements and using UML object oriented technology to represent the complexity of the processes and the large amount of analytical information generated in wine production. Thus, a decision support system, named JWisWine, was built for monitoring wine production.

INTRODUCTION

Wine composition and production are both extremely complex (Flanzy 2000) [1] (Ribereau-Gayon 2000) [2]. Chemical monitoring and control of the overall production process is needed to provide knowledge of the quality of the raw material (grape), the intermediate and final product.

Fig. (**1**). Shows a detailed activity diagram (Booch 1999) [3] of the overall process. Analytical monitoring of the enological parameters –in parallel with the process– determines the start and end of each step, in addition to the quality achieved. The availability of analytical information at the precise time and in the appropriate format is crucial to this operation.

Advances in the automation of quality control requires: a) implementation of a system for management, organization, handling and treatment of the information generated throughout the production process with the purpose of providing the technical manager with information enough to take a decision (McGrawth JFE 1998) [4] (Muller AIM 1999) [5]; b) automation of analytical methods using instrumentation coupled to the information system (Ilyukhin FC 2001) [6].

Three common steps are involved in the analysis of the chemical parameters: sampling and possible sample treatment; instrument measurement and data collection and processing of the instruments outputs (Fig. **2**). An average of 75 and 25 samples per day (fermentation and non-fermentation periods, respectively, with a number of 5-20 analyses per sample) is carried out in the laboratories in which JWisWine has been implemented. The amount of data generated as a consequence of the variety and number of analyses justifies the implementation of an information management system with the following purposes:

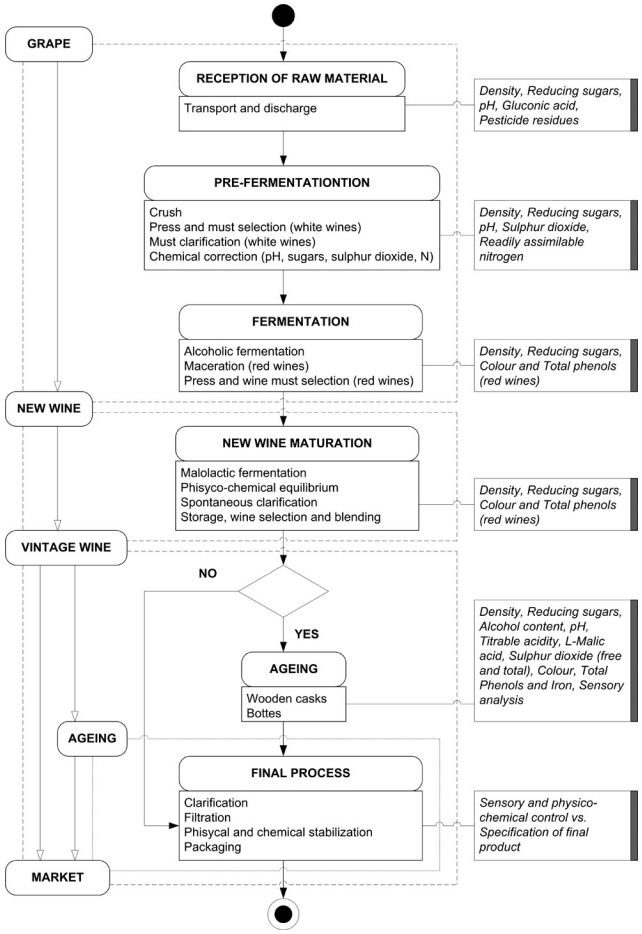

Fig. (1). Activity diagram of the wine production process.

1) To make possible the appropriate management of historical data. This includes the efficient storage of a huge amount of data corresponding to previous harvests, which can be used when and as required.

*Address correspondence to this author at the University of Cordoba, Department of Computing and Numerical Analysis, Campus de Rabanales, Albert Einstein Building, E-14071 Cordoba, Spain; Tel: 0034-9657-212082; Fax: 0034-957-218630; E-mails: iluque@uco.es, mangel@uco.es

1874-1339/08 2008 Bentham Open

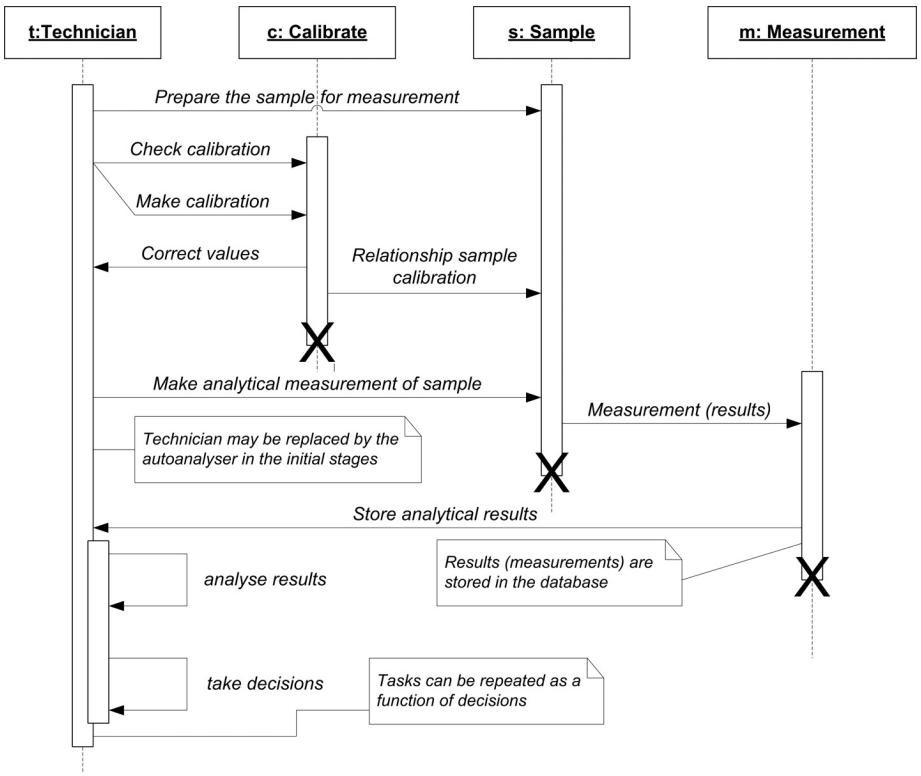

Fig. (2). Stages involved in the development of analytical measurements.

2) To have access to additional information about analyzers, analytical parameters, etc. This aspect endows the system with the traceability capacity, which allows, for example, correlating the decision about selling a given wine batch with the evolution of a given parameter (e.g. volatile acidity, ethanol, etc.).

3) To assess the quality, accuracy and reliability of the information, which require analytical instruments and software components for their controls; thus, providing the users with the results in the appropriate format.

The work presented here focuses on the management of the information. It constitutes a first step to automation of the analytical methods in wineries (Urbano IEEE-EFTA 2003) [7]. Thus, a Decision Support System (DSS), JWisWine, has been built using Java and Oracle 9i as tools, while the design of the system was carried out by both object-oriented and evolutionary incremental paradigms.

Materials and Methods

For the construction of JWisWine both the object-oriented and the evolutionary incremental software engineering paradigms have been used.

Modelling of the System Domain

In order to model the domain under study the following classes and subclasses have been established: class Parameter, for the parameters to be determined in the wine; subclass DatumParameter, which includes the parameters to be determined directly in the sample (e.g. pH measurement); *CalculatedParameter*, which comprises the parameters which are determined in an indirect way through a given algorithm for the establishment of a relationship between the *DatumParameter* and a mathematical equation (e.g. dissolved solids in wine), as shown in the class diagram (Booch 1999) [3] in Fig. (**3**).

Class Unit represents the scale in which a parameter is measured (e.g. ethanol concentration expressed in percentage, mg mL-1, etc.), together with the class Parameter, – which represents the property which characterizes the material under study (e.g. ethanol)– yield the new class Magnitude, which represents a parameter measured in a given unit (e.g. volatile acidity measured in absorbance unit).

The advantages of using the object-oriented paradigm for modeling the problem corresponding with the definition previously mentioned can be appreciated in the summarized Java code definition of the main classes as follows: Eckel 2002) [8] (Anderson and Stone 1999) [9]:

```
/* Definition of the class Parameter * /
        public abstract class Parameter
        extends java.lang.Object
        implements PersistentSQL
/* It implements the interface PersistentSQL with
the methods to access to the database */
/ * Definition of some of the attributes of the
class * /
        protected java.lang.String alias
```

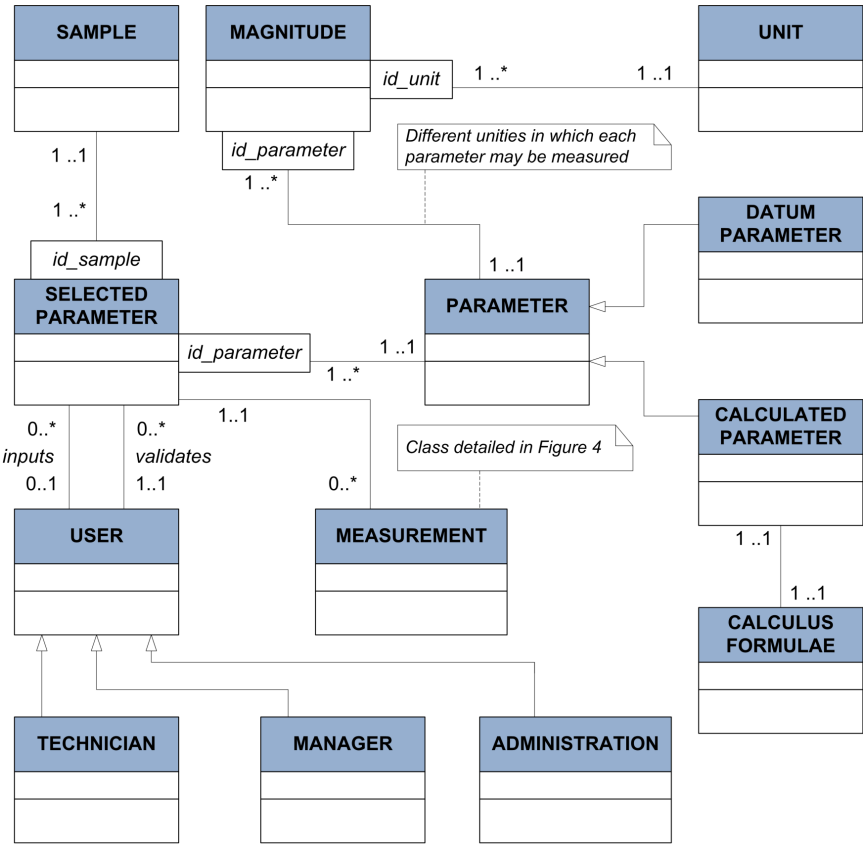

Fig. (3). Class diagram for parameters and samples.

```
/* It contains the alias of the parameter, used in
the formulas */
        protected Magnitude default
/* It contains a reference to the default magnitude
*/
        protected java.util.Vector magnitudes
/* It contains all the magnitudes in those a value
of this parameter can be expressed */
        protected java.lang.String id_parameter
/* It contains the name of the parameter */
/* Other class attributes */
        cache, connection, debugParameter,
        stateSQL, deletedMagnitudes, ninstances,
        changedname, objcache, idChanged
/* The constructors definition */
        Parameter()
/* Constructor without arguments */
        Parameter(java.lang.String n,
java.lang.String alias, Unit u)
/* It creates a parameter of name 'n', alias'
alias' and the unit 'u' as their default unit */
/* Definition of some of the class methods */
        Magnitude addMagnitude(Unit u)
/* It allows that the parameter is expressed in the
unit 'u', creating a new magnitude associated to
the parameter */.
        Magnitude[] getMagnitudes()
/* This method returns an array with all the magni-
tudes defined in the system for this parameter */
        Magnitude getDefaultMagnitude()
/* It locates and returns the default magnitude
associated to this parameter */
        void setNormalInterval(Unit u, double min,
double max)
/* This method fixes the maximum and minimum values
for measurements expressed in the magnitude that
associates the unit 'u' and the parameter */
        void setNormalInterval(Magnitude m, double
min, double max)
/* This method overloads the previous ones and
fixes the normal interval for the magnitude indi-
cated in 'm' */
        void loadSample(Sample s)
/* This method relates the parameter with the sam-
ple 's' */
        abstract Measurement createMeasurement()
/* It creates and returns a new measurement */
        boolean isMeasurementNormal(Measurement m)
```

```
/* This method checks if the indicated value in the
measurement is between the maximum and minimum val-
ues of the parameter */
/* Other class methods */
        addToCache, storeParameter, addMagnitude,
        deleteParameter, disablePersistentCache,
        deleteMagnitude, enablePersistentCache,
        equalsParameter, finalizeParameter, ge-
        tAlias, getStoreException, getConnection,
        getSQLDeleted, getSQLState, getFromCache,
        getMagnitudes, getIdParameter, getNewSQL,
        getParameter, getParameters, getSQL, hash-
        Code, isAliasDefined, isIdDefined, isMagni-
        tudeDefined, isPersistentCacheEnabled, re-
        build, retrieve, removeFromCache, setAlias,
        setConnection, setSQLState, setDefaulfMag-
        nitude, setIdParameter, toString, unload-
        Sample, isSampleValidable, isMeasurement-
        Normal
/* Definition of the class DatumParameter * /
        public class DatumParameter
        extends Parameter
/* It inherits the attributes and methods from the
class Parameter */
/* Definition of some of the class methods */
        Measurement createMeasurement()
/* This method provides an implementation to the
superclass method */
/* Definition of the class CalculatedParameter * /
        public class CalculatedParameter
        extends Parameter
/* It inherits the attributes and methods from the
class Parameter */
/* Definition of the class attributes */
        protected CalculusFormulae calculusFormulae
/* It stores a reference to the calculation formu-
lae that evaluates this parameter in their default
units */
/* Definition of some of the class methods */
        void loadSample(Sample s)
/* This method overloads the superclass method */
        Measurement createMeasurement()
/* This method provides an implementation to the
superclass method */
        CalculusFormulae getCalculusFormulae()
/* This method returns the calculation formulae
that allows to evaluate the CalculatedParameter */
        DatumParameter[]getDatumParameter()
/* This method consults the calculation formulae of
the CalculatedParameter and returns all the parame-
ters involved in the formulae */
/* Definition of the class Magnitude * /
        public abstract class Magnitude
        extends java.lang.Object
        implements PersistentSQL
/* It implements the interface PersistentSQL with
the methods to access to the database */
/* Definition of some of the class attributes */
        protected TranslationFormulae translation-
formulae
/* It contains a reference to the conversion formu-
lae that calculates this magnitude */
        protected java.lang.String name
/* It contains the name of the magnitude */
        protected Parameter parameter
/* It contains the parameter associated to the mag-
nitude */
        protected Unit unit
/* It contains the unit in which the parameter can
be expressed */
        protected double maximumValue
/* It contains the maximum value for evaluation of
the associate parameter in the associate units */
        protected double minimumValue
/* It contains the minimum value for evaluation of
the associate parameter in the associate units */
```

For Instance:

- The attribute magnitudes can only be accessed from the class Parameter, because only the instances (objects) of this class have permission to manage that property.
- The inheritance property allows that both the *DatumParameter* and *CalculatedParameter* classes inherit the Parameter class structure. Also, these classes include their characteristic attributes and methods; thus encapsulating their definition.
- Different constructor methods can be defined in order to facilitate the system development (see the constructor methods for the Parameter class).
- The functionality of overload's mechanism is illustrated in the method *createMeasurement* of the Parameter class, which is redefined in the *CalculatedParameter* and *DatumParameter* classes to adapt it to their particular behaviour.

The information domain can be represented by three main blocks, namely:

1) The samples and the analytical parameters to be monitored.
2) The analytical measurements performed along the wine production process.
3) The quality assessment and the control of the integrity and accuracy of information.

Fig. (**3**). shows, through a class diagram (Booch 1999) [3], the modeling for the representation of the samples and

analytical parameters to be monitored. In this diagram, *SelectedParameter* –an association between the classes Sample and Parameter– represents the total of the analytical parameters to be monitored in a given sample at a given time.

The association between Parameter and Unit classes (Fig. **3**) makes possible to express a given analytical parameter in different units through the Magnitude class, as required. In this way, the range of normal values of the parameter can be expressed in a different manner.

Fig. (**4**). shows some of the classes that represent the information concerning the analytical measurements during wine production. The class Measurement represents the measurements with time, which becomes specific for a given parameter through *DatumMeasurement* and *CalculatedMeasurement*.

The model also enables plotting the calibration curves with the instrumental data checking by association between classes *CalibrationCurve* and *Analyser*. In addition, the data obtained in the analytical measurements can be transformed into chemical values or magnitudes defined for the selected parameter by association between the *CalibrationCurve* and *CalculatedMeasurement* classes.

Moreover, the most important classes and associations that constitute the third block focused on the control of security, integrity and quality of the information (Fig. **3**). The system takes into account the importance of the users in the security and quality of the production process. Class User represents different users involved in the information system through Technician, Administration and Manager subclasses.

Modelling of the Function Domain

Three types of users are recognized by the system: Technicians, Managers and Administration users. Technician users are responsible for the daily analyses in the winery laboratory, namely: general information management (parameters, units, magnitudes and so on); inclusion of new samples and parameters to be monitored in these samples; validation of results, generation of reports, and the basic functions related with the management of information generated in the wine analysis processes.

In addition to all the privileges that correspond to the Technician users, Manager users are responsible for the management of the daily activities in the laboratory, the evaluation of the workload, the testing of both the results and the analytical equipment, and the analysis of the results as a function of both origin and time. In this way, the Managers can develop the reports that support the decisions to be made in the winery. This type of user can ask for information of the system using non pre-established criteria for searching and the system will provide managers with real time responses in the appropriate format (either tabular or graphical). For example, the evolution on time of the measurements of the analytical parameters in a sample, comparison of these results between samples of the same or different origin and year, etc., can be requested.

Finally, the functionality assigned to the Administration users is similar to the one these users have in the conventional information systems. They are responsible for the management of system's users and the privileges they have. Also, they must maintain the security, integrity and privacy of the information involved in the system.

Technology Used in the Development of the System

JWisWine is a system developed in Java (Anderson and Stone 1999) [9] (Zakhour 2006) [10] under a three-layer structure (Fig. **5**) as shown in the deployment diagram. The information is updated using the DBMS Oracle 9i (Dorsey and Hudicka 1999) [11] (Loney and Koch 2002) [12] under an object-relational model which implements properly the

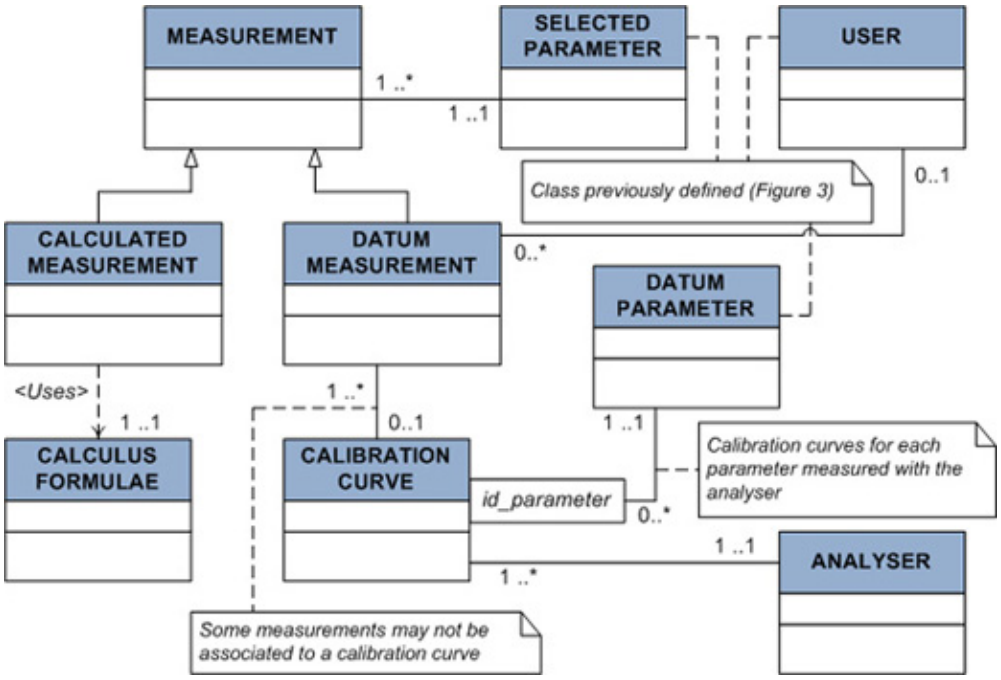

Fig. (4). Class diagram for measurements.

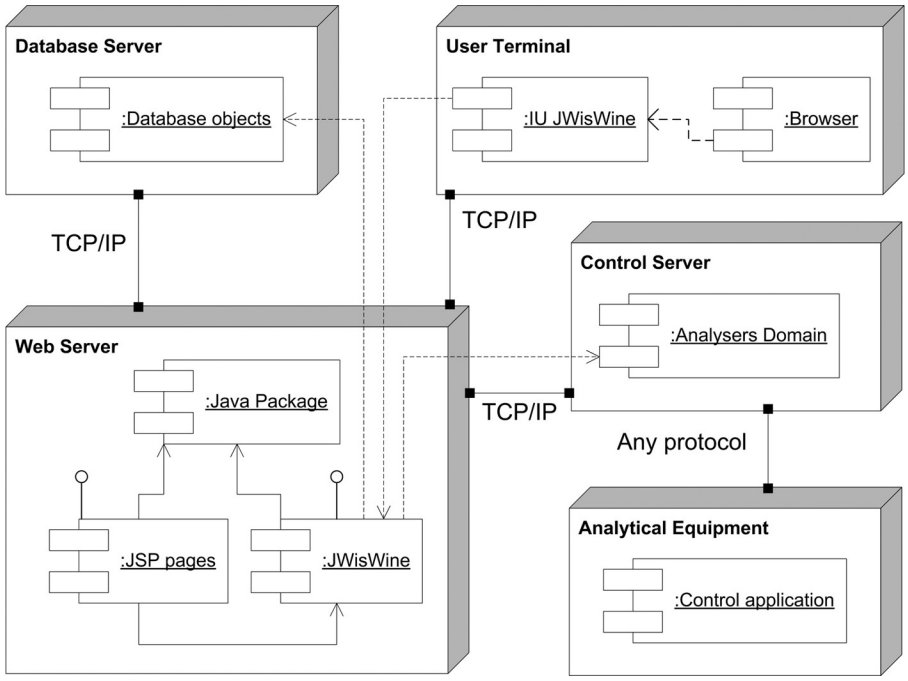

Fig. (5). Deployment diagram of the system architecture.

classes model in the data base server in charge of giving the data service to the users. From their terminal, that provides the calculation service, the users access to the information through the Web server –using Apache (Laurie and Laurie 2002) [13] for this service– and by both JDBC and SQLJ procedures (Morisseau-Leroy 2001) [14]. The Web server communicates with the database server, transmits the user requirement and gets the results back to him/her, presented through a Java interface.

RESULTS AND DISCUSSION

Software Developed

JWisWine is a DSS (Decision Support System) composed by two tools, the first one is a client application through which the users load samples, introduce chemical data, generate daily reports, etc. Thus, this tool is in charge of managing the data generated daily in the winery. The other is a web application through which managers use the information extracted from historical data for making some decisions.

User's Participation

The participation of different users –managers and technicians– in the development of the system has been implemented by the evolutionary incremental paradigm. This allowed the dynamic specification of the system requisites in order to take into account the objectives pursued by companies.

The system users have overall control and monitoring of the data even the data is acquired automatically or entered manually (e.g. if the data from the analyser is collected by the computer through a digital interface or the data is manually introduced by the user, respectively). In addition, each type of user has a different access to the system regarding the assigned privileges and all the monitored activities in the system.

Evolution of JWisWine Functionality

The aim of this sub-system was to endow wineries with a tool capable to assist the management of the analytical monitoring in the following aspects:

- To establish the appropriate information flow in the overall system in order to unify and classify the working procedures.
- To know in real time the working load of the winery as the system has previously stored the types of samples and the parameters to be analysed in each sample.
- To visualize, modify, verify and validate the daily results from any workstation in the Local Area Network.
- To move –after validation– the results and all data of interest from a given sample to the historical repository. Warning messages advise users about the existence of outliers.
- To store all data related to instruments usage and technicians activity.

The aim of this sub-system is to extract information from the data stored in the historical repository. The most remarkable tasks of this sub-system are as follows:

- To know the evolution of a given parameter attending to a preset searching criteria. The information can be obtained through a standard browser and presented in graphs or tables.

- To obtain series of statistical data as either the change or standard deviation of any parameter as a function of time.
- To group samples attending to the values of one or more parameters, including the possibility of applying PCA (principal components analysis).
- To introduce control samplers in order to extract information of the functioning of the analytical equipment – namely, Sewart graphics–. Traceability of historical data with both the equipment and regression data used is also available.

CONCLUSSIONS

Firstly, JWisWine is a valuable tool to manage historical data. Thus, users are able to extract information structured in different manners depending on the requirements. The access to the data of interest is achieved in real time, overcoming limitations related to manual searches. Wine quality is improved because of the major knowledge of the wine making.

On the other hand, daily handling of data is also improved by easy handling of the interactive interfaces. The introduction and validation of data, generation of reports and other operational tasks are carried out by simple procedures and protocols that are friendly to users, who only require a short learning period.

JWisWine can be modified functionally with few changes in the architecture and components of the system due to the open, scalable and independent characteristics of both the supporting model and the technology used.

REFERENCES

[1] C. Flanzy. Enología: Fundamentos Científicos y Tecnológicos (Enology: Scientific and Technological Fundamentals). Madrid: AMV-Mundi Prensa, 2000.

[2] P. Ribereau-Gayon, Y. Glories, A. Maujean, D. Dubourdieu. Handbook of Enology. Vol. 2: The Chemistry of wine. Stabilization and Treatments. Chichester: John Wiley & Sons Ltd. 2000

[3] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language. User Manual. Amsterdam: Addison-Wesley Longman Inc, 1999.

[4] M.J. McGrath, J.F. O'Connor, S. Cummins. "Implementing a process control strategy for the food processing industry". *J. Food Engineer*, vol. 35 (3), pp. 313-321, 1998.

[5] E. Muller, M. Bassin, J.P. Troyon, P. Nowak. "Implementation of rapid result management systems in the metals industry". *Lab. Autom. Inform. Manag*, vol. 34(1), pp. 31-40 1999.

[6] S.V. Ilyukhin, T.A. Haley, R.K. Singh. "A survey of automation practices in the food industry". *Food Control*, vol. 12 (5), pp. 285-296, 2001.

[7] M. Urbano, M.D. Luque de Castro, M.A: Gómez-Nieto. 2003. "Automation of flow injection methods in the winery industry through a computer program based on a multilayer model". Proceedings of 9th IEEE International Conference on Emerging Technologies and Factory Automation, 2003, pp. 530-536.

[8] B. Eckel, Thinking in Java (Third Edition). New York: Prentice Hall, 2003.

[9] J.C. Anderson and B.L. Stone. Manual de Oracle Jdeveloper (Oracle JDeveloper Manual). Madrid: McGraw-Hill/Oracle Press, 1999.

[10] S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber. The Java Tutorial: A Short Course on the Basics, 4th Edition. New York: Prentice Hall, 2006.

[11] P. Dorsey and J.R. Hudicka. Oracle8 Database Design Using UML Object Modelling. New York: McGraw-Hill Osborne Media, 1999.

[12] K. Loney and G. Koch. Oracle 9i The Complete Reference. New York: Oracle Press, 2002.

[13] B. Laurie and P. Laurie. Apache: The Definitive Guide (2 edition). Sebastopol, USA: O'Reilly; 2002.

[14] J.N. Morisseau-Leroy, M. Solomon, G. Momplaisir. Oracle9i SQLJ Programming. New York: Osborne-Oracle Press, 2001.

Received: April 01, 2008 Revised: May 25, 2008 Accepted: May 26, 2008