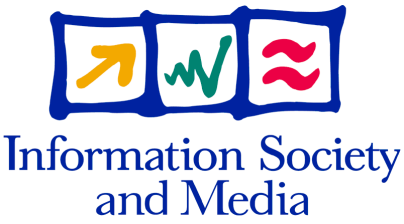
HEARTFAID

D31 – Knowledge Discovery System

Submission date: 20/03/2008
Due date of document: 31/01/2008



HEARTFAID

A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING MEDICAL-CLINICAL MANAGEMENT OF THE HEART FAILURE WITHIN THE ELDERLY POPULATION

Project summary	
Project acronym:	HEARTFAID
Project identifier:	IST – 2005 – 027107
Duration of the Project:	01/02/2006 – 31/01/2009
Project Co-ordinator:	UNICAL University of Calabria (Italy)
Thematic Priority:	Information Society Technology
Instrument:	Specific Targeted Research or Innovation Project

Consortium

- UNICAL - Università della Calabria (Italy)
- UNICZ - Università degli studi Magna Graecia di Catanzaro (Italy)
- UNIMIB - Università degli studi di Milano Bicocca (Italy)
- JUMC - Jagiellonian University Medical College (Poland)
- VMWS - Virtual Medical World Solutions Ltd (United Kingdom)
- FORTHNET - Hellenic Telecommunications and Telematic Applications Company S. A. (Greece)
- SYNAP - Synapsis s.r.l. (Italy)
- CNR - Consiglio Nazionale delle Ricerche (Italy)
- FORTH - Foundation for Research and Technology Hellas (Greece)
- RBI - Rudjer Boskovic Institute (Croatia)
- AUXOL - Istituto Auxologico Italiano (Italy)

D31 – Knowledge Discovery Systems

Document summary	
Document title:	D31 – Knowledge Discovery System
Document classification:	Derivable D31
Dissemination level:	CO
Submission date:	20 March 2008
Due date:	31 January 2008
Authors:	Dragan Gamberger – RBI Tomislav Smuc - RBI Rajko Horvat - RBI Sergio Di Bona – SYNAP Davide Guerri – SYNAP Marco Lettere – SYNAP Manolis Stratakis – Forthnet Stelios Louloudakis – Forthnet
Work package:	WP6 – End-user application and services
Report version:	1.0

Short description
The main goal of this deliverable is to define and analyze the knowledge discovery system, regarding its on-line implementation and user interaction through the HEARTFAID front end.

Change record		
Version number	Changes	Release date
0.1	Template	20 / 12 / 2007
0.2	Preliminary draft of the document	15 / 01 / 2008
0.3	First Draft of the document	08 / 02 / 2008
0.6	Advanced Draft of the document	01 / 03 / 2008
0.9	Final Draft	09 / 03 / 2008
1.0	Further contribution and final version	20 / 03 / 2008

Table of contents

Executive Summary

The D31 module "Knowledge Discovery System" has as a main goal to outline all the tasks relevant to the development, testing, implementation and online integration of the appropriate Knowledge Discovery (KD) system that will provide an efficient mechanism that will satisfy the HEARTFAID project's prerequisites.

The introduction curtails all the assignments accomplished within the module's timeframe along with a general overview of Knowledge Discovery systems relevant to HEARTFAID and the prerequisites they have to satisfy in order for them to become inseparable components of the project. It also abridges the KD's integration and implementation method invoked for the development of the HEARTFAID's Front-End.

The First Chapter is concerned with an overview of the Knowledge Discovery (KD) systems. It presents the general conditions HEARTFAID imposes on any candidate for a valid KD system and demonstrates the argumentation for the incorporation of the Random Forest (RF) algorithm as the authorized and endorsed Knowledge Discovery process for HEARTFAID.

The Random Forest (RF) algorithm is being exemplified and its main features outlined. The evaluation of RF as an irrefutable possibility is being given so as to establish the reasoning behind its selection among the various Knowledge Discovery methods. Furthermore, the Parallelized Random Forest is being presented and discussed as an alternative of the traditional RF in the case of a cluster of computers.

The RF implementation is being described and its execution requirements are meticulously given so as to depict not only the intuitiveness of the RF system but also to demonstrate its multiple usage possibilities. Reference is made to the processing power required in order for an RF system to work. Section 3.6 is significantly more technical as it presents Attribute-Relation File Format necessary for the deployment and invocation of a KD process based on the Random Forest algorithm.

At the end of the chapter, the integration of the KD engine onto the web interface is being discussed along with the way the results are being visualized.

Chapter two touches upon the integration of the KD system into the middleware platform of HEARTFAID.

Chapter three relates to the implementation of the KD system's integration into the web service. A general outline of the Knowledge Discovery web service is

given. The perspective of the realization of the service is outlined along with the basic actions and capabilities a user of the platform will be bestowed upon. The discretization of users into administrators and others, along with the assignment of appropriate credentials to them by the system's admins, is described. Further, the way each user can initiate, stop, re-start, manage and modify any KD process (s)he may choose to invoke is being outlined.

Reference is made to the project's web forms as a means to facilitate user navigation among and management of the various (private) KD tasks. The datasets' templates along with the user's capacity to modify – delete – create datasets are explained. The existence of administrative and auxiliary forms is elucidated and their usage and aim made clear.
The task scheduler by which the integration of the KD system into the web service is being realized is presented in detail. The ability to execute multiple tasks simultaneously is manifested along with ability of the user to define his/her own timeframe for the execution of the desired KD projects.

At the end of the chapter, the technicalities behind the organization of the data for the Knowledge Discovery web service are given as the latter requires the handling of many different data resources. The flexibility of the service is being demonstrated and a meticulous description of the way all the steps of the knowledge discovery process are being executed is outlined.

The use of iFrames is being presented along with a broad outline of the various security aspects one needs to account for and take into consideration when designing and implementing sensitive data processing and transfer.

At the end, the general conclusions from the work undertaken within this module's timeframe are given along with a list of relevant references.

1 Glossary of terms

TERM	DEFINITION
ANMCO	Associazione Nazionale Medici Cardiologi Ospidalieri
ARFF	Attribute-Relation File Format
eCRF	Electronic Case Report Form
GNU GPL	General Public License
HF	Heart Failure
HTML	HyperText Markup Language
KD	Knowledge Discovery
KDD	Knowledge Discovery in Databases
MPICH	A portable implementation of the Message-Passing Interface Standard
PARF	PArallel Random Forests
PDF	Portable Document Format
RF	Random Forests
ROC curve	Receiver Operating Characteristic curve
iFrame	Inline Frame
DOM	Document Object Model
IT	Information Technology
ICT	Information and Communication Technologies
HIS	Healthcare Information Systems
EHR	Electronic Health Records
DoS	Denial of Service
TCP	Transmission Control Protocol

2 Introduction

HEARTFAID is a Research and Development project aimed at devising, developing and validating an innovative knowledge based platform of services, able to improve early diagnosis and to make more effective the medical-clinical management of heart diseases within elderly population.

Of profound importance to the overall substantiation of the project is the area of Knowledge Discovery as via this, the appropriate information for the health professional about a medical patient will be attainable in order for the former to be able to attest, determine and corroborate an apt and appropriate diagnosis that will be supremely beneficial for the latter. In general, knowledge discovery can be defined as the process of identifying interesting – in our case, even life-saving - new patterns in data. These patterns can be, for instance, relations, events or tendencies, and they can reveal both regularities and exceptions. In the core of the process, data mining methods are used to extract and verify patterns along with providing descriptive and qualitative analysis for the discovered orderliness. Further, they can offer predictive systems used to forebode the future behavior of predefined elements and/or objects.

The knowledge-based platform of services proposed in the HEARTFAID project has the main goal to support both decision makers and clinicians operating in the field of heart diseases in the processes of diagnosis, prognosis, treatment and personalization of healthcare assistance to the elderly population. The widespread implementation of the services proposed will guarantee a better quality of life to pathological patients and will further reduce the number of necessary hospitalizations leading to a minimization of the social and economical impact on the healthcare system.

Analysts have always performed the task of extracting useful information from recorded data. But, the increasing volume of data in modern business and science, calls for computer-based approaches. As data sets have grown in size and complexity, there has been an inevitable shift away from direct hands-on data analysis toward indirect, automatic results extrapolation using more complex and sophisticated tools. The modern technologies of computers, networks, and sensors have made data collection and organization an almost effortless task. However, the captured data needs to be converted into useful information and knowledge. Data mining is the entire process of applying computer-based methodology, including new techniques for knowledge discovery, to appropriate and available datasets.

Although data mining is a relatively new term, the technology is not. Companies for a long time have used powerful computers to sift through volumes of data such as supermarket scanner data to produce market research reports. Continuous

innovations in computer processing power, disk storage, and statistical software are dramatically increasing the accuracy and usefulness of analysis.

The principal objective of this module is to implement the appropriate processes for knowledge discovery within the HEARTFAID databases. That will have to be integrated into the web-based Front-End so as to allow for online knowledge discovery queries on the available data stored in the system repository. Both traditional and innovative data-mining algorithms will have to be tested and invoked in order for the optimal approach to be identified and implemented.
The Random Forest (RF) approach has been implemented as the principal Knowledge Discovery model for HEARTFAID. The main arguments to support this decision have to do with the RF's unexcelled accuracy among current algorithms, its efficiency, its capability to handle thousands of input variables without variable deletion along with the fact that the generated forests can be saved for future use on other datasets.

Random forest does not overfit. One can run as many trees as it is desired. It is fast. Running on a data set with 50,000 cases and 100 variables, it produced 100 trees in 11 minutes on a 800Mhz machine. For large data sets the major memory requirement is the storage of the data itself, and three integer arrays with the same dimensions as the data. If proximities are calculated, storage requirements grow as the number of cases times the number of trees.

The Random Forest approach has been incorporated into the Web service by making use of the task scheduler which is responsible to execute and manage tasks. The task scheduler executes data mining tasks on specified data in a given order and priority. The scheduler has the ability to execute multiple jobs at the same time (multitasking) depending on the number of processors and available memory. That provides the obvious advantage of getting the desired outcome regardless of the hardware of the machine that the query is actually being sent from.

The main principle behind the integration of the RF engine into the web interface is the level of intuitiveness of the tool as well as the ease of use and the graspable display of the output. The Knowledge Discovery (KD) service is implemented as a series of interconnected web pages or web forms. The user is being given all the available information about the processing, as this is also part of the inherent characteristic of RF.

The KD system has been integrated into the Front-End by means of iFrames for the electronic Case Report Form (eCRF). That has the advantage of making the Front-End independent of any changes that might occur in the KD. Hence, the latter is invoked 'as is' both preventing any perturbations on the Front-End by possible inflationary development scenarios on the KD system and further

allowing full flexibility on the KD progression since the latter won't be hindered by any Front-End's limitations.

iFrames have been introduced into the platform since they solve the problem of a double user registration when the electronic Case Report Form (eCRF) is invoked. That offers not only an enhanced and friendlier to the user interface but also minimizes further security aspects associated with the transmission of sensitive data to a distant location. The users will get authenticated once, at the Front-End, and their session information will get propagated to the appropriate server. That is an efficient, time saving and indeed secure technique that is being popularly employed and substantiated.

The security aspects associated with every use of the web and indeed services such as these that involve sensitive data have been considered and the communication routes have been encrypted via SSL (Secure Socket Layer). The latter is the standard among web developers and is cross-browser friendly. Proper authentication schemes have been considered and implemented so as to make sure that communication over insecure networks won't prove a security concern that will hinder the project's implementation.

3 Knowledge Discovery systems

3.1 Requirements

The Knowledge Discovery (KD) is the process aimed at analyzing the available data with intention to identify patterns useful for understanding relationships in the data and for data classification. This process does not have a unique and ideal solution. Only the expert understanding of the obtained results can give the meaning to the results and only expert evaluation can select the optimal solution for the given context. Because of that the KD process is an iterative process in which many different experiments with available data have to be repeated. The basic requirement on the KD Web service is to ensure that the KD process can be efficiently performed also by people that are experts in the HF domain and not necessarily experts in the KD process itself.

The central part of the KD process is application of some machine learning approach. Although computationally this is the most complex part, it is relatively easily implemented as a part of the web service. The service enables users to select some KD algorithm and its dependent parameters and ensures that the KD algorithm successfully terminates also in the cases of ill-defined data.

Central problems of the KD service are a) data preparation that should ensure that many different experiments can be efficiently organized and realized b) presentation of the results in the form that is intuitive for the user and c) organization of the experimental setting so that user can review all previous result not necessarily obtained in the same KD session. Additionally and very specifically for the HEARTFAID project is that the service should be integrated in the platform ensuring that real platform data are continuously available for the KD process.

3.2 Random Forest

The Random Forest (RF) is a relatively new data mining tool with an excellent reputation within the scientific community. The algorithm was developed by Leo Breiman and Adele Cutler. Its quality comes from the fact that it very efficiently implements a voting scheme of a large number of independent classifiers realized as decision trees. Its resilience to noisy data and overfitting are outstanding when compared to other machine learning algorithms. Due to the robustness of the approach, the values of the few free parameters governing the performance of the algorithm do not need to be changed from their default values in order to obtain results of very good quality for practically any dataset/problem.

The real strength of the RF algorithm lies in the process of building many independent classifiers but so that their bias and correlation are low. This is the key to the high prediction accuracy of the models implemented as random forest. Low bias is obtained by growing trees to maximum depth without pruning while low correlation is obtained by bootstrapping samples with random subspace method. It is important to emphasize that the accuracy of a forest grows practically monotonically with the number of generated trees. The most relevant user defined parameter is the number of generated trees which represents the trade-off between the execution time complexity and the expected quality of the final solution.

3.2.1 Features of the RF

- Excellent prediction performance of constructed models.
- Automatic cross-validation.
- Relative efficient on large data bases.
- Can handle thousands of input variables without preprocessing.
- Estimates importance of variables for the classification.
- Directly generates an internal unbiased estimate of the generalization error during the forest building.
- It has an effective method for handling missing data.
- It has options for balancing classification error in unbalanced data sets.
- Generated forests can be saved for future use as models or for further forest growing.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering and locating outliers.

3.2.2 PARF - Parallelized random forest

PARF is a parallelized version of random forest. It is implemented so that it can work on a cluster of potentially many computers. In this respect RF is ideal because it can be easily and efficiently parallelized. The code is mostly written at Rudjer Boskovic Institute, and there is undergoing project to further optimize code execution, add portability, regression, visual interface and visualization of results. The data format used in PARF is the standard .arff format used in all Weka applications.

3.2.3 Execution requirements

The code is written in Fortran 90 programming language under GNU GPL 2.0 and can be compiled without modification on almost any FORTRAN compiler. It is platform independent and scalable (requires MPICH library), and is tested to work under Linux and Windows platform.

Table 1 shows average execution time. Large data sets require more system memory. If additional features are used, storage requirements as well as execution time grow.

Table 1. Execution times for basic model generation for classification.

Platform	Number of trees	Execution time
benchmark 1942 instances and 85 attributes		
AMD 2GHz, 1GB RAM	100	1.54 s
AMD 2GHz, 1GB RAM	1000	10.45 s
AMD 2GHz, 1GB RAM	10000	1 m 40.22 s
benchmark 6491 instances and 85 attributes		
AMD 2GHz, 1GB RAM	100	4.71 s
AMD 2GHz, 1GB RAM	1000	26.46 s
AMD 2GHz, 1GB RAM	10000	4m 4.75 s

Command line syntax:

```
parf -t trainset.arff
```

By default the last attribute is assumed to be the target class.

The list of all supported options can be obtained by running:

```
parf -h
```

Wherever there is a file name argument, a dash (-) can be written to signify the standard input or standard output. This is the default where the argument is optional. If an existing filename is specified for an output option, the file will be overwritten without warning. If there is an attribute argument, the attribute name or number can be used. If an argument contains spaces, it must be quoted. The arguments are separated by a comma sign and any surrounding spaces are ignored. Also, the entire list should be quoted, and not only the individual name. For example:

```
      parf -t test.arff -fd "test forest.txt" -c 3 -u 'marital
status, height'
```

3.2.4 ARFF data format

The dataset must be provided in the Attribute-Relation File Format or for short ARFF. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato. An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files have two distinct sections. The first section is the Header which is then followed by the Data section.

The Header of the ARFF file contains the name of the dataset (relation), a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
   % 1. Title: Iris Plants Database
   %
   % 2. Sources:
   %      (a) Creator: R.A. Fisher
   %                          (b)    Donor:    Michael    Marshall
(MARSHALL%PLU@io.arc.nasa.gov)
   %      (c) Date: July, 1988
   %
   @RELATION iris

   @ATTRIBUTE sepallength  NUMERIC
   @ATTRIBUTE sepalwidth   NUMERIC
   @ATTRIBUTE petallength  NUMERIC
   @ATTRIBUTE petalwidth   NUMERIC
   @ATTRIBUTE class          {Iris-setosa, Iris-versicolor, Iris-
virginica}
```

The Data section in the ARFF file looks like the following:

```
   @DATA
   5.1,3.5,1.4,0.2,Iris-setosa
```

```
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Lines that begin with a % are comments. The @RELATION, @ATTRIBUTE and @DATA declarations are case insensitive.

The ARFF Header Section

The ARFF header section contains relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:

```
@relation <relation-name>
```

where <relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute declarations

Attribute declarations have the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and its data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then it is expected that all the instances will have that attribute value in the third comma delimited column.

The format for the @attribute statement is:

```
@attribute <attribute-name> <datatype>
```

where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types:

* numeric
* <nominal-specification>

* string
* date [<date-format>]

where <nominal-specification> and <date-format> are defined below. The keywords numeric, string and date are case insensitive.

Numeric attributes

Numeric attributes can be real or integer numbers.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name-1>, <nominal-name-2>, <nominal-name-3>, ...}

For example, the class value of the Iris dataset can be defined as follows:

```
    @ATTRIBUTE class          {Iris-setosa, Iris-versicolor, Iris-
virginica}
```

Values that contain spaces must be quoted.

String attributes

String attributes enable us to create attributes containing arbitrary textual values. This is very useful in text-mining applications, as we can create datasets with string attributes and use them for example as labels. String attributes are declared as follows:

```
    @ATTRIBUTE LCC    string
```

Date attributes

Date attribute declarations take the form:

```
    @attribute <name> date [<date-format>]
```

where <name> is the name for the attribute and <date-format> is an optional string specifying how date values should be parsed and printed. The default format string accepts the ISO-8601 combined date and time format: "yyyy-MM-dd'T'HH:mm:ss".

Dates must be specified in the data section as the corresponding string representations of the date/time.

ARFF Data Section

The ARFF Data section of the file contains the data declaration line and the actual instance lines. The data section starts with @data declaration and is a single line denoting the start of the data segment in the file. The format is:

```
@data
```

The instance data

Each instance is represented on a single line, with carriage returns denoting the end of the instance. Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the n-th @attribute declaration is always the n-th field of the attribute).

Missing values are represented by a single question mark, as in:

```
@data
4.4,?,1.5,?,Iris-setosa
```

Values of string and nominal attributes are case sensitive, and any that contain space must be quoted, as follows:

```
@relation LCCvsLCSH

@attribute LCC string
@attribute LCSH string

@data
AG5,   'Encyclopedias and dictionaries.;Twentieth century.'
AS262, 'Science -- Soviet Union -- History.'
AE5,   'Encyclopedias and dictionaries.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'
```

Dates must be specified in the data section using the string representation specified in the attribute declaration. For example:

```
@RELATION Timestamps

@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"

@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

3.2.5 Integration with the web service

Integration of the Random forest with the web service is implemented using the task scheduler which is responsible to execute and manage tasks. Task scheduler starts the RF tool with predefined and user selected parameters. After RF tool finishes, the Report manager processes the results and outputs into KD report for later visualization, evaluation and analysis.

The following list represents basic and advanced parameters for RF tool that are available in the version implemented in the web service. Classification accuracy with cross validation is always automatically written to the console and is automatically saved in a log file by the task manager.

Basic parameters:

- Trainset file - parameter '-t' (file name needs to be specified). Determines the input training data file in the ARFF file format.
- Confusion matrix - parameter '-tc' (file name needs to be specified). Produces confusion matrix in which each column represents the instances in a predicted class, while each row represents the instances in an actual class,
- Class attribute - parameter '-c' specifies which attribute is target class,
- Number of trees to grow - parameter '-n', the minimum number of trees should be 100 and the maximum should not exceed 10000 in the web application especially for large datasets as the memory consumption and the speed of execution are affected with little or no gain in the classification accuracy,

Advanced parameters:

- Attribute importance list - parameter '-i' produces attribute list with attribute statistical significance calculated during forest building (file name needs to be specified),
- Redo the forest with N most important attributes - parameter '-v'. The forest is built twice. First time the attribute importance is calculated, and the second time the forest is built only with the N specified most important attributes,
- Redo the forest with variables more significant than N - parameter '-vs'. It is the same as the previous parameter except the threshold is given as a number.
- Votes - parameter '-tv' outputs result of the classification, and classification results for each instance separately (file name needs to be specified). This is especially useful for obtaining additional information on classification performance such as ROC curves. The output needs to be preprocessed for results.

- Outlier measure - parameter '-ot' (file name needs to be specified). An outlier is a case whose proximities to all other cases are small. A measure of outlyingness is computed for each instance in the dataset.
- Scaling (dimensionality reduction) - parameter '-st' (file name needs to be specified), and parameter '-s' which represents number of scaling coordinates. Output needs to be preprocessed prior to visualization of results.
- Class weight override - parameter '-w'. Prediction error can be balanced with this parameter.
- Save forest - parameter '-fs' (file name or prefix needs to be specified). Saves the forest for future classification run. In a Fortran version forest is saved in multiple files, with a common prefix.
- Load forest - parameter '-fl' (file name or prefix needs to be specified). Loads a forest for classification run. None of the forest growth options, as well as the training analysis options except for the proximity, outlier and scaling ones are available. Note that all the files in the Fortran version generated by the -fs option must be present at the same location, denoted by the prefix.

3.2.6 Reporting and visualization of results

Outcomes, or results of the PARF code run depend on the particular option parameters switched, therefore on the user requests. The standard version of the output of the single PARF run contains following results:

Summary of the model

- OOB accuracy, kappa of the model estimates
- Confusion matrix

Results from insight tools

- Variable importances
- Variable interactions
- Class Prototypes

The overall output of the PARF run should be delivered to the report manager module, which is responsible for the generation of the single report of the run.
Visualization of results from the PARF run is basically related to the results obtained from the insight tools: variable importances, interactions and prototypes.
Sample of the output report from the PARF run for the standard version of options is given in the Appendix A.

More advanced results from the PARF code would contain following results:

- Voted decision on all the samples of the dataset
- Scaled representation of the dataset

- Outlyingness measures

The visualization of this results is left for the offline analysis, thus they will be given in the numerical format in the report.

4 Knowledge discovery and middleware integration

As reported in the Deliverable "D28 - Integration and Interoperability middleware prototype", two different approaches have been investigated to integrate into the HEARTFAID platform of services the different modules developed by the HF Consortium:

- adopting the defined protocol, based on standard languages (e.g. XML) to interact with the Enterprise Service Bus (ESB), i.e. the core infrastructure of the middleware.
- Using the native interface of the single modules.

In the first case, the defined protocol allows each new module to interact with the core infrastructure of the middleware, i.e. the ESB. The ESB is then responsible to interact with the other components of the platform, thus guaranteeing the interoperability among all the systems operating in the entire HEARTFAID system. We can say that the middleware will act as an intermediary between two modules that need to exchange information. In other words, the middleware level is the component of the platform responsible to guarantee the efficient access and exchange of both clinical and administrative data among all the components of the platform of services. In this way, every time a new module must be added to the platform or an existing module changes, all the other components (except the ESB) can continue working without the need of being aware about the changes occurred since are not involved by these modifications.

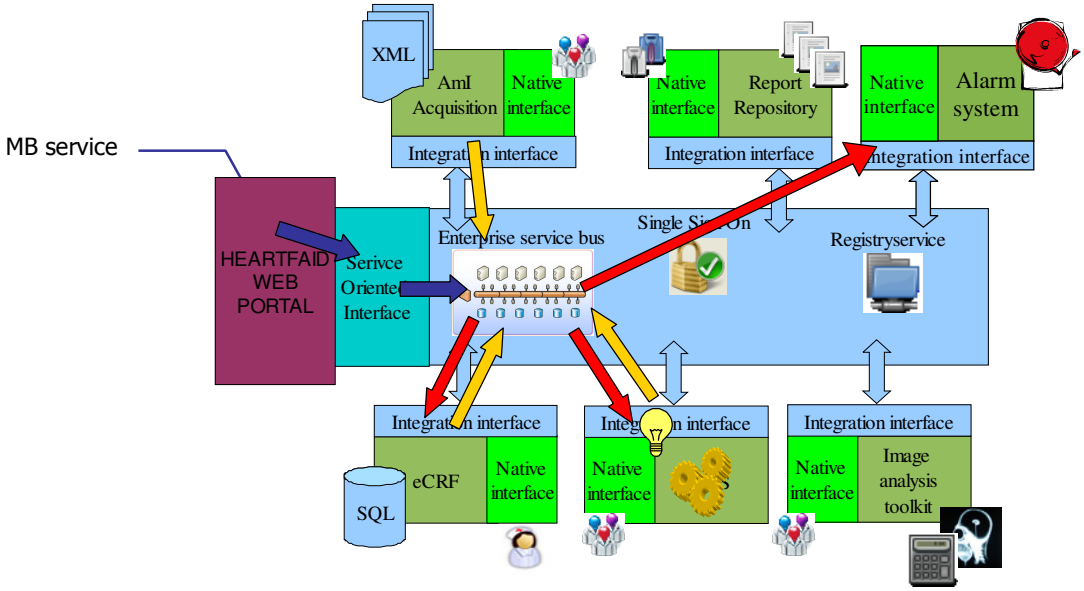
In the second case, the integration relies on the functionalities offered by the proprietary interface of each module that must be integrated with the other systems of the platform, and in particular with the HEARFAID Portal. Of course, the advantage of this approach is that it is possible to exploit all the functionalities for which the module has been implemented. Moreover, the effort to integrate the module is, most of the time, inferior with the respect to the adoption of the middleware protocol.
On the other side, however, the disadvantage is that every time the module should interact with a new component of the platform or an existing component, which is already interacting with the module, must be changed, then it is most likely that the integration itself must be revised and adapted to the new components.

We cannot say, a priori, which of the two approaches is better than the other. It depends on the specific needs, on the functional specifications, on the freedom to apply changes to the original modules and, of course, on goals that must be achieved with the HEARTFAID project.
In general, we can say that the activities related with the Knowledge Discovery are the most "research-related" if compared to the other tasks of the project.
Therefore, we decided to experiment both the two approaches, according to the specific features of the available modules:

- An integration based on the definition of a dedicated protocol for the *Model Bases for the Early Detection of Decompensation* based on *Decision Trees*, *Support Vector Machine* and *Decision Lists* (see "Deliverable D29 - Models and Methods for Knowledge Discovery" for details on the algorithms implemented and "D20 - Clinical standards and first middleware prototype" for the details about the protocol implemented);
- An integration based on native interfaces for the algorithms described in this document.

This decision was taken both technological and methodological reasons. From one side the programming languages used for the development of the Model Base was already compliant with the specifications of the ESB. On the other side, the Model Base needs to interact with more systems of the platform with respect to the Knowledge Base, therefore the use of the ESB as intermediary was definitely more recommended.
Beside the technological and methodological reasons that brought to this distinction, there is a practical reason: while the system described in this document requires a graphical interaction with the end-user (e.g. to explain the reasons of a suggestion), the Model Base only provides an alarm level that is then managed by the ESB and, if it is the case, by the Alarm System. Therefore, it is more reasonable to use the native interface of the Knowledge Base instead of re-implement them entirely in the Portal interface.

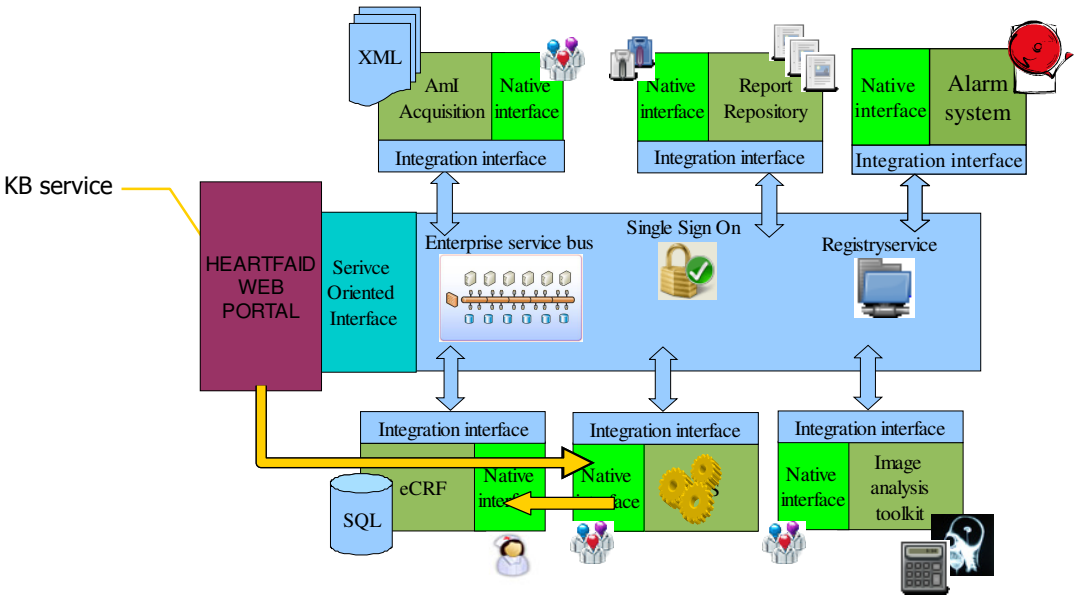
The following pictures show the two approaches adopted. In the first case, the request of the user to invoke a service of the Model Base is forwarded to the ESB through a Service Oriented Interface. The ESB is then responsible for recovering all the necessary data to perform the processing, e.g. from the home/hospital monitoring devices (i.e. from the Ambient Intelligence module) and from the Electronic Patient Record (i.e. the HEARTFAID eCRF), invoking the service of the Model Base, getting the answer and, in case, for generating an alarm by using a service of the Alarm System.



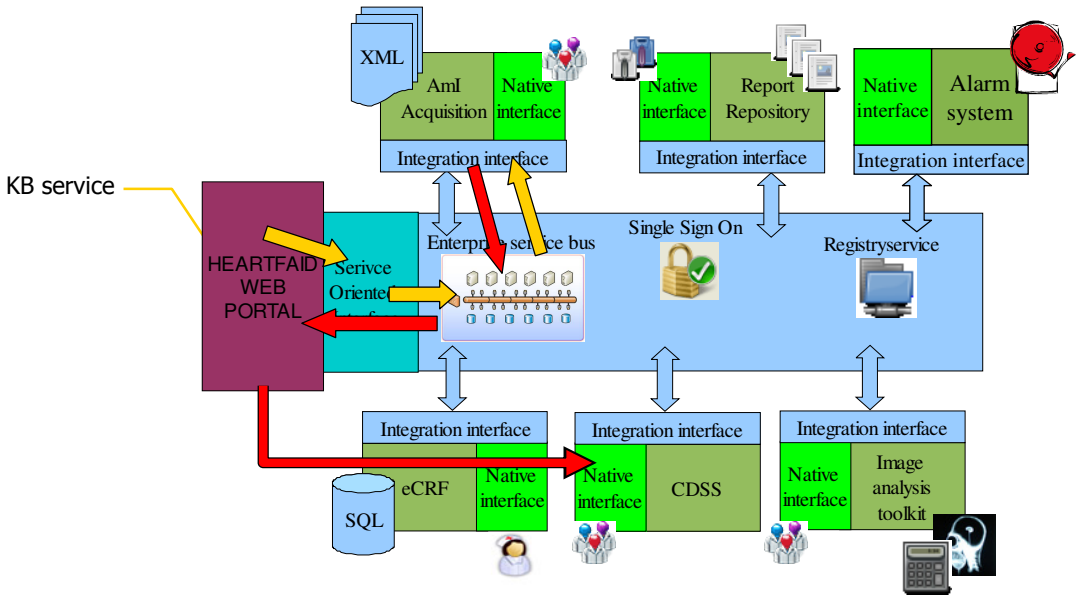
In the second approach adopted, we should distinguish between two different cases:

- The algorithms need source data that can be accessed directly using the native interface of another module;
- The algorithms need source data that must be provided by the middleware of the HEARTFAID Platform.

In the first case, the user who is requesting the service should first identify the patient for whom the service itself is invoked, i.e. it should define a *Patient Context* (for details on this procedure see Deliverable "D28 – Integration and Interoperability middleware prototype"). Once a patient has been selected (i.e. the Portal has the unique ID number of the patient), the request, together with the ID of the patient, is sent to the Knowledge Base service. Then, the Knowledge Base service will use the native interface of other modules of the platform to access the data needed for the processing.



The second case is quite more complex. In fact, since the data necessary by the decision supporting algorithms to perform the processing cannot be acquired using the native interfaces of the other modules, and, on the other side, a specific protocol to interact with the ESB has not been defined, the integration was performed at "*the level of view*". The "level of view" is the most external layer of an ICT system, i.e. the layer that interacts with the end-users; in the specific case of the HEARTFAID Platform, this layer is the Portal. Therefore, the Portal, once received the request of the end-user, will be in charge of acquiring the necessary source data by invoking the correct services of the ESB through the platform Service Oriented Interface. The ESB will recover the requested data from the other modules that have been integrated using the dedicated protocol (e.g. the home monitoring data managed by the Ambient Intelligence module), and will forward these data to the Portal. Finally the Portal is able to invoke the Knowledge Base service requested by the end-user, proving all the data necessary to perform the processing (or, at least, all the data that cannot be recovered using the native interface of the other modules of the platform).



In this way it was possible to cover all the different cases that might occur to perform a Decision Support activity on source data available in the HEARTFAID platform of services:

1) The service is integrated with dedicated protocol specifically developed, and data is stored into an external o internal module of the platform and can be accessed using the dedicated protocol (concerning the use of standards refer to Deliverable "D20 - Clinical standards and first middleware prototype");
2) Data is stored into external modules of the platform and can be accessed using the native interface of the modules themselves (regardless if the service is integrated using the native interface of the dedicated protocol);
3) The service is integrated with native interface and data is stored into external modules of the platform and can be accessed only using the dedicated protocol.

5 Knowledge Discovery and Front End integration

5.1 Front-end iFrame integration

5.1.1 The iFrame HTML Element

An iFrame (inline Frame) is an HTML element which makes it possible to embed another HTML document inside the main document. In contrast to normal frames which occupy a fixed position in the browser window, an iFrame employs a specific position within an HTML document. The size of the iFrame is specified in the surrounding HTML page, so that the surrounding page can already be presented in the browser while the iFrame is still being loaded. The iFrame behaves much like an inline image and the user can scroll it out of view. On the other hand, the iFrame can contain its own scroll bar, independent of that of the surrounding page.

While regular frames are typically used to logically subdivide the content of one website, iFrames are more commonly used to insert content from another website into the current page. The embedded document can be a different one without reloading the surrounding page, by using the "target" attribute of an HTML anchor or by employing JavaScript. This makes many interactive applications possible. The main alternative to using an iFrame in these situations is editing a document's DOM tree. Sometimes invisible iFrames are also used for asynchronous communication with a server, as an alternative to XMLHttpRequest.

More recently, Mozilla Firefox, Opera and Microsoft Internet Explorer introduced contentEditable and designMode, which enables users to edit the contents of the HTML code contained in an iFrame. This feature has been used to develop rich text (WYSIWYG) editors within an iFrame element.

5.2 Knowledge Discovery web service

Web service integrates and simplifies the process of knowledge discovery in such a way that even a person which is not skilled in area of knowledge discovery can use the service and be able to interpret results. Also the process allows user to quickly navigate through variations of dataset, models and results. Figure 1 describes proposed structure which integrates process of knowledge discovery and the web service.

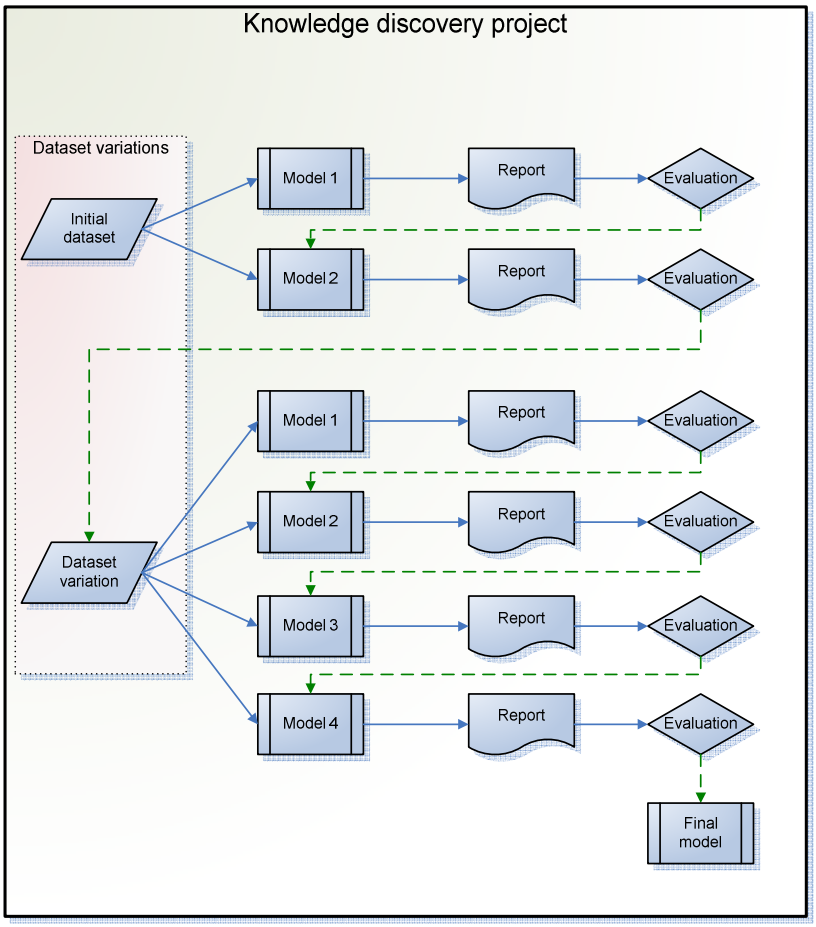


Figure 1. Diagram of the KDD process

The project is a logical representation of the knowledge discovery process. User starts with the project and with a chosen dataset which can be uploaded, selected from predefined template (historical data) or created from Heartfaid platform data. Initial dataset can be modified as the process of knowledge discovery continues, thus creating dataset variation. The user can add, delete or transform attributes or delete some instances from the initial dataset, or a initial dataset variation. New dataset can't be uploaded into an existing project, but in contrast, user must create new project with this new dataset. Every run of the data mining tool on a dataset is called a modeling task and result is called the KD model which contains different reports and visualizations. Modeling task can be run multiple times and with different parameters on any variation of the dataset contained within the project. The results of data mining tool are stored together with the modeling task for later evaluation and analysis by the user. After user evaluates results of the KD model the user can decide what to do next. User can make another model with different parameters or different data mining tool, make another dataset variation,

decide that the acquired model is good enough or start a new knowledge discovery project. The project stores dataset, dataset variations, models and results until user or administrator chooses to delete them. Every user has its own knowledge discovery projects and can't access somebody else's. Documents relevant to the current project can be attached, and every step user makes can be documented and described in detail.

5.3 Web forms

The knowledge discovery service is implemented as a series of interconnected web pages or web forms. The central part of the service is composed of web pages that are intended for KD such as starting of new project, starting of new KD tasks, reviewing the KD results and manipulation of the datasets. There are other web pages that provide functionalities which are not related to the KD such as informative pages, pages for user management and management of KD tasks. Additionally, there are administrative pages for adding new data mining tools and dataset templates.

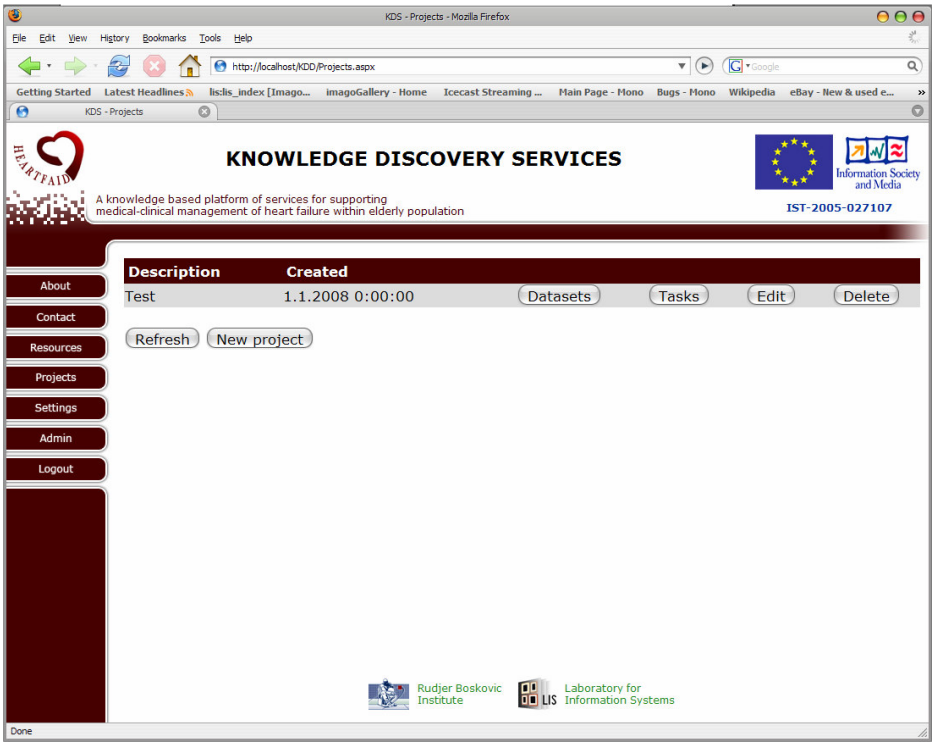


Figure 2. The list of all projects available to the user after its log in.

The project web form contains list of the projects for each user (Figure 2). The user can delete old, create new or manage existing projects. Each project contains list of dataset variations which can contain one or more models and their results. Project also contains one or more external documents (PDF files, Word documents, HTML links or similar) related to this project. Each KD project has its components hierarchically organized. When the user opens project he can see list of associated datasets, and when he selects dataset he can see all of the KD models related to that dataset. This structured organization of the project components allows the user to easily navigate to datasets, KD models and their

results. KD project can't contain multiple datasets but only the variations of the initial dataset. The user can make modifications to the datasets contained within this project.

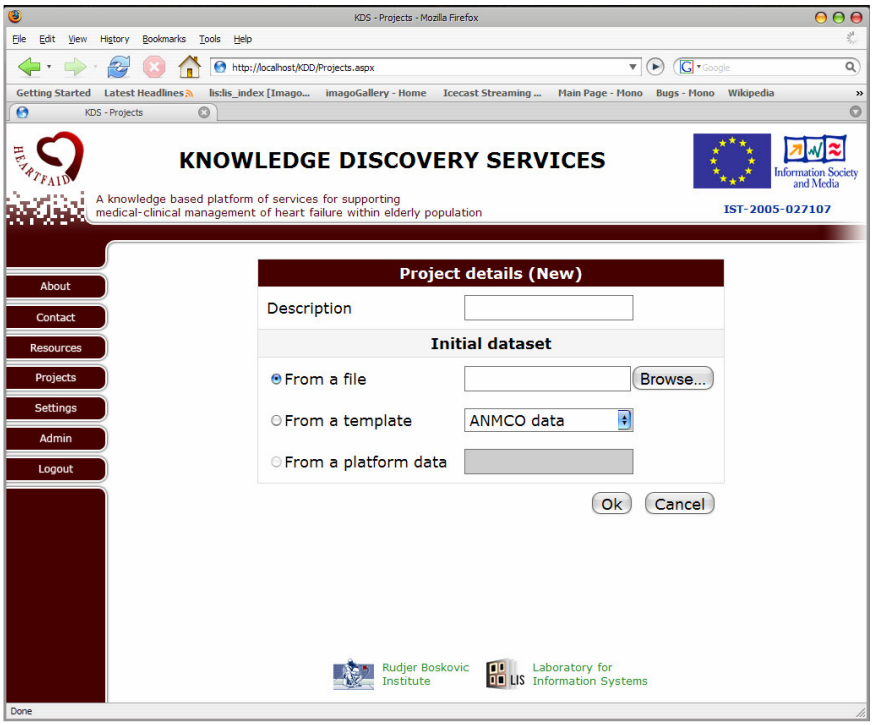


Figure 3. Adding of a new project

When new project is created (Figure 3) the user can choose the source for the initial dataset. The dataset source can be a local file from the user's computer, a template dataset from the web service, or data taken from the Heartfaid platform database.

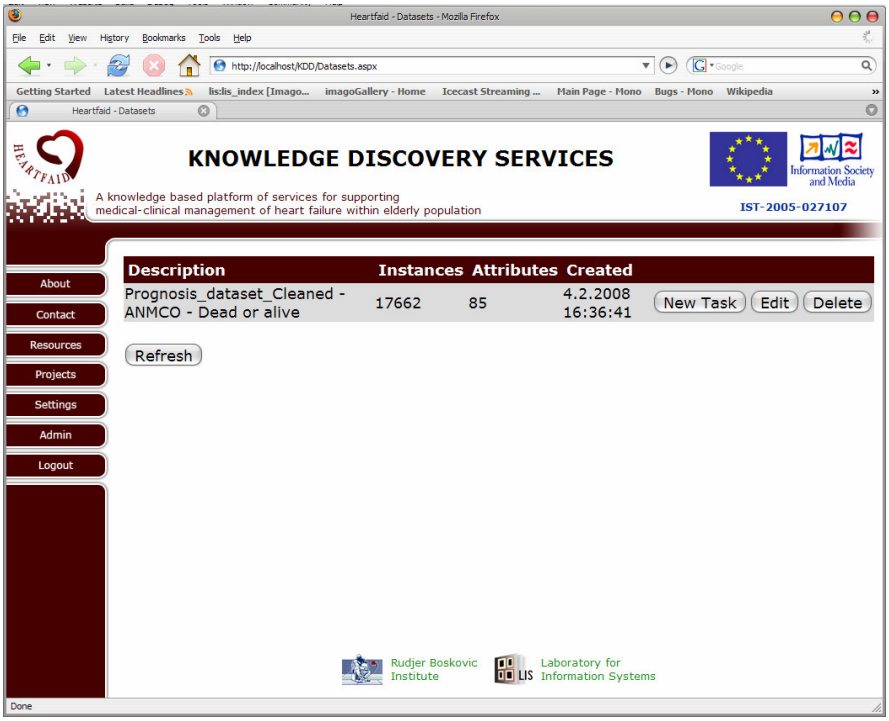


Figure 4. List of datasets in the selected project

List of available datasets (Figure 4) enables user to modify or delete datasets or start a new data mining task on any of the available datasets.

Users need to have the ability to modify datasets. The modifications include deleting of the existing attributes, constructing of a new attributes and deleting of the instances of data that are for example outliers. When user saves this modified dataset it is stored as dataset variation. Dataset variation can have further variations and it is important that user can describe modifications he has done along with an automatic description of what has been modified.

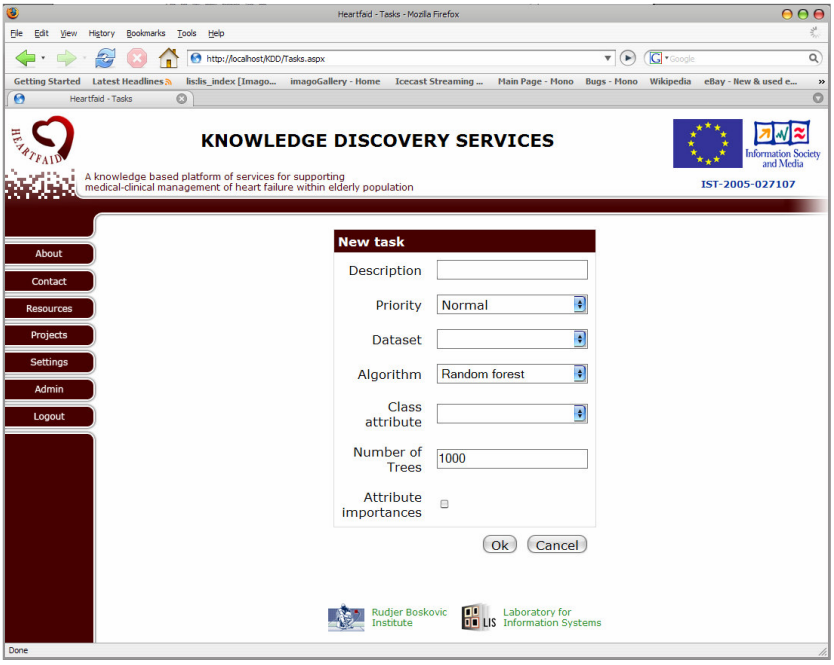


Figure 5. Adding of a new data mining task

When user adds a new data mining task he can select target class and modify data mining parameters based on the selected data mining tool (Figure 5). The parameters are a fixed list, and in general different for each available data mining tool.

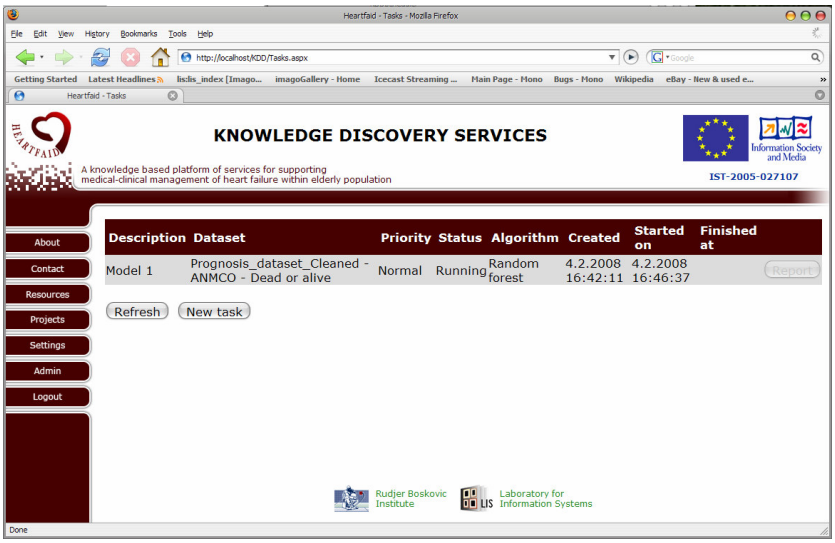


Figure 6. An example of a running data mining task

After user submits the data mining task for execution the scheduler executes this task and stores results into the database. Figure 6 represents a running KD task. In Figure 7 is the report of a KD RF task. Each data mining tool can have multiple results which are combined to a single report. Report contains visual and numerical representations of the results.

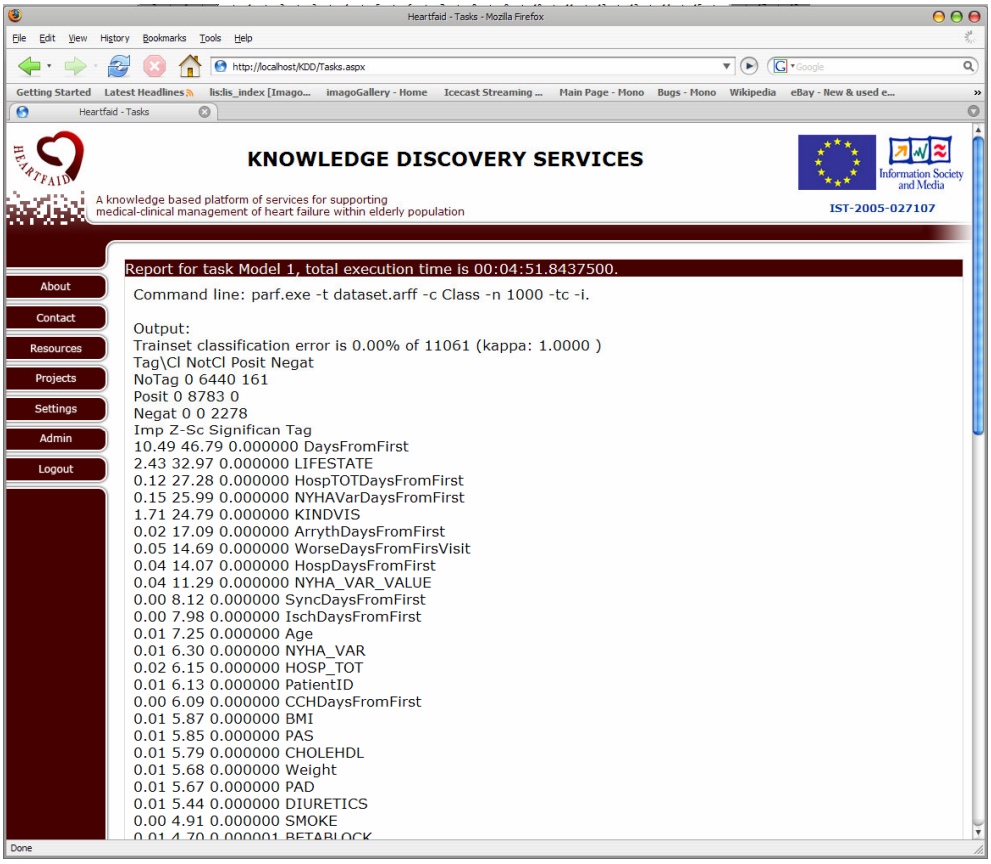


Figure 7. A result of a data mining task

Administrative and auxiliary forms

Figure 8. System settings web form

The service provides management (Figure 8) of system parameters, users, data mining tools, dataset templates, user projects and KD tasks that are executing. Administrative web pages can only be accessed by system administrators.

Administrators can introduce new data mining tools to the service. This is accomplished by adding of a new assembly which will be executed by task scheduler. Administrator must then define all of the parameters that this new data mining tool will have. Sometimes it is useful to add dataset templates which are available to all users. They are usually retrospective datasets like the ANMCO dataset represented in Figure 9. The major advantage of templates is that they can be prepared off-line. In this way all inconsistencies, outliers, and examples with a lot of unknown values can be eliminated in preprocessing. In this way the quality of induced results can be guaranteed.

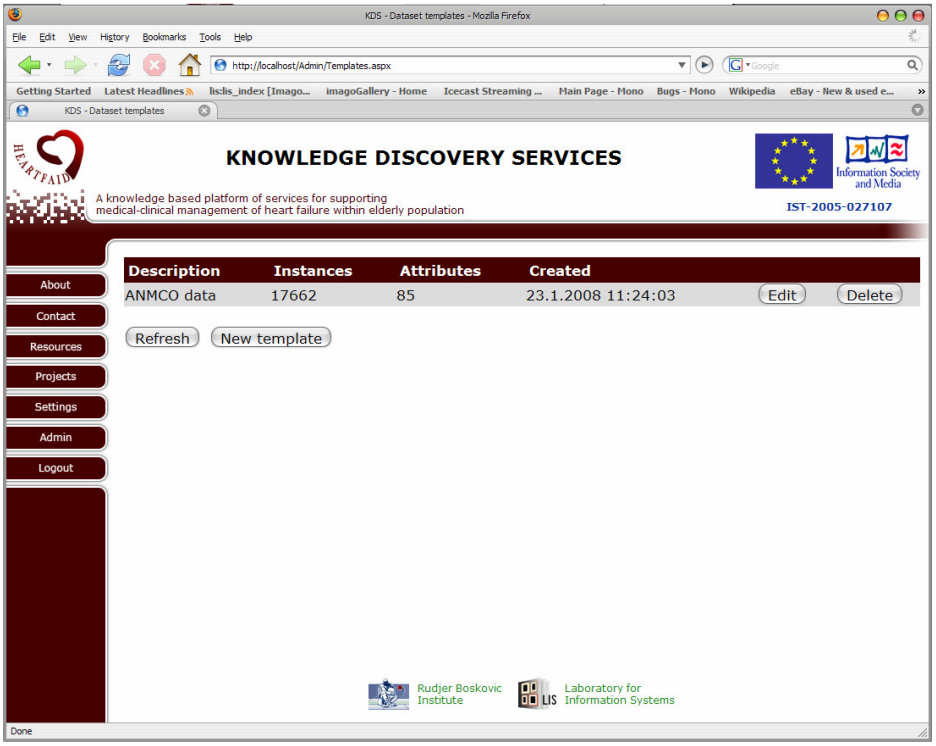


Figure 9. Listing of existing templates. In this case it is the retrospective ANMCO dataset.

Finally, the service has some other administrative tasks like registration of new users and management of their permissions (Figure 10) and login form for registered users (Figure 11). The service can also contain web pages with resources in the form of manuals, examples of data mining and scientific papers that are related to the application and use of knowledge discovery processes, tools and other skills that are relevant to the field of KD in medical applications.

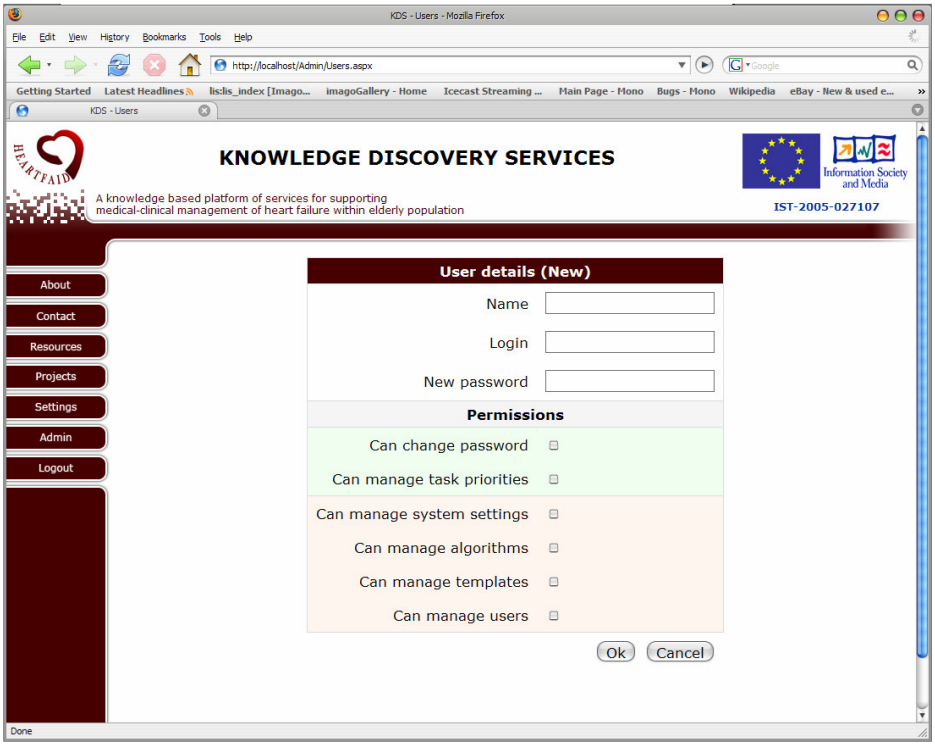


Figure 10. Adding of a new user to the service.

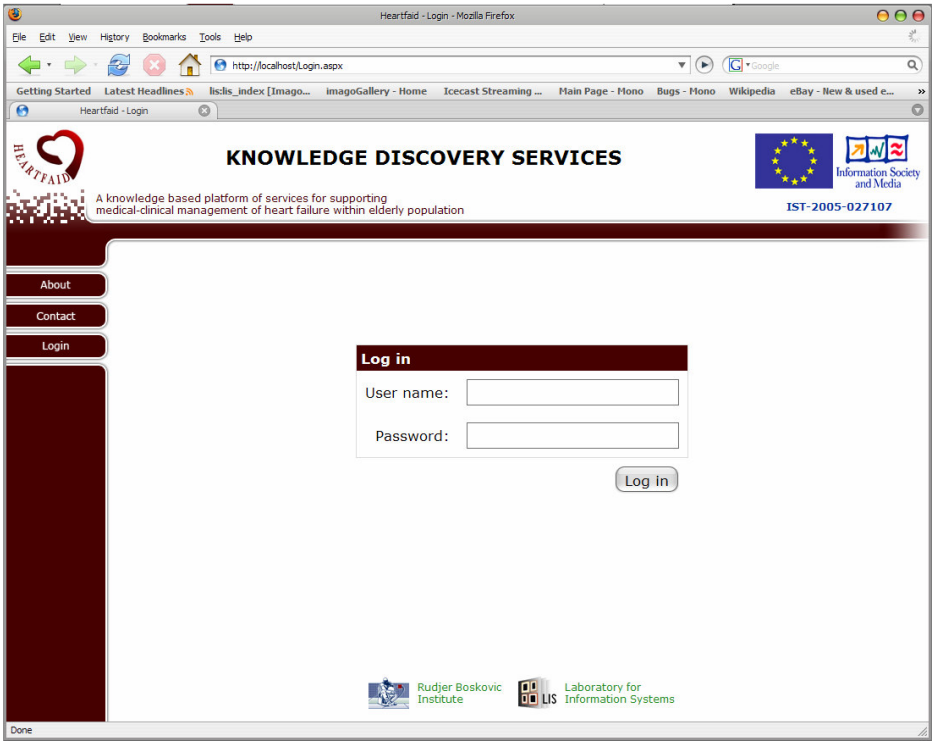


Figure 11. User login web form

5.4 Task scheduler

Task scheduler is a service component that executes, in a given order and priority, data mining tasks on a specified dataset (figure 2). Execution of a KD task is the one of the most important parts of the service. From one side there are web forms that communicate with users, in between is a database and on the other side is a task scheduler which communicates with and executes data mining tools. Task scheduler periodically checks for new tasks submitted by users, and if there are free resources executes them. New tasks aren't started before old ones finish. For really large datasets and data mining models there is a limit on memory consumption and executing time. If any task violates memory or time limit it is terminated with error status and explanation of the cause of such an action. The scheduler has the ability to execute multiple jobs at the same time depending on the number of processors and available memory. Before task executes task scheduler updates task status to signalize to both user and the platform that this task is executing. Next, datasets and parameters are prepared, and finally data mining tool is executed. The task scheduler functions independently of the web service, and doesn't have to execute on the same machine that is serving the web service. User can interact with this layer indirectly through KD tasks which are stored in the database. User can also stop or postpone data mining tasks. When task is completed the task scheduler sends and event to the Report manager. The report manager gathers results and outputs from data mining tool, preprocesses them and generates report which, when completed, is stored into database. The KD report can be in HTML or PDF format. The report is then presented to the user by the KD web interface. The KD report is used for evaluation and analysis of the results.

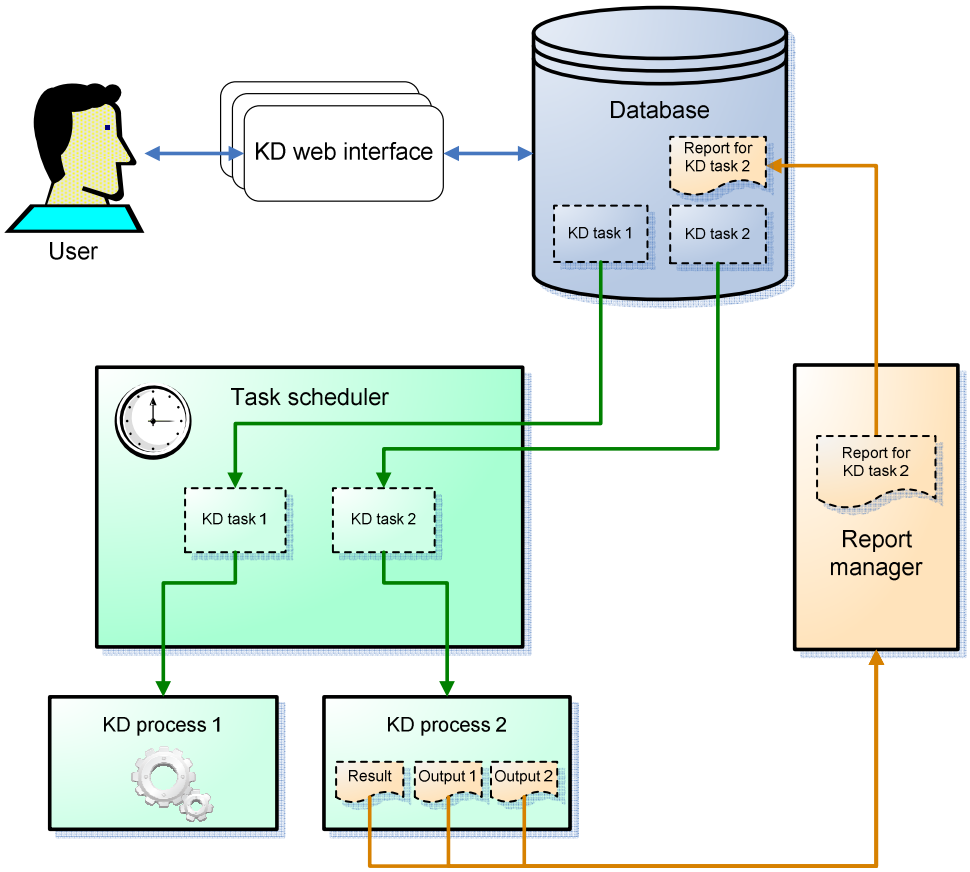


Figure 12. Functional diagram of the Task scheduler.

The diagram on figure 12 represents a process that Task scheduler does every time a new KD task is submitted. The process starts with a user submitting a new KD task through web interface. The data about this new KD task is written into database, and Task scheduler, which periodically checks for new KD tasks, prepares and starts a new KD process based on a KD task. After KD process completes, an event is issued to Task scheduler as well as Report manager. Task scheduler then stores back information about KD process into database and frees resources. Report manger gathers all required results and outputs from KD process, processes them and composes a report which is then stored into Database. After the process is completed report and KD process information is available to the user.

5.5 Data organization for the knowledge discovery web service

Knowledge discovery is a complicated process that requires handling of many different resources which need a place to store information about users, projects, datasets, results, as well as information about knowledge discovery tools and processes. On one side of a knowledge discovery process are data mining tools, and on the other side are users and their permissions to use different parts of web interface and data mining tools. In between is a database that connects these two parts and stores all the data about data mining tools, data mining processes, projects, datasets and knowledge discovery results. Inside database each user has its own workspace which is distinguished by user ID. This workspace is presented to the authenticated user in a form of a web forms which are structured and connected together in such a way to represent a knowledge discovery process. Workspace contains customized user settings, projects, datasets, data mining tasks and its reports. Only web interface and task scheduler have permission to access this database directly. Web interface enables users to interact with this database through web forms, while at another side task scheduler executes data mining tools and stores their results. The database is structured in such a way to ensure the flexibility and accessibility of the service. The following tables provide detailed description of database and its structure which enables execution of all steps of the knowledge discovery process.

Table 2. Information about users and their permissions

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
Name	varchar	100	NOT NULL
Login	varchar	45	NOT NULL
Password	varchar	45	default NULL
Permissions	unsigned int	10	NOT NULL default '0'

Different users have different permissions. Some are system administrators, while other are simple knowledge miners who do not require access to system management and its resources. Each user is distinguished by its ID and each has its own projects, project associated datasets and data mining tasks.

Table 3. This table contains list of all available data mining tools

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
Name	varchar	45	NOT NULL
Executable	text		NOT NULL

The service contains many different data mining tools. Each tool has its advantages depending on a problem being solved. The data mining tools or algorithms are distinguished by its ID. The next table contains list of all available parameters for each data mining tool defined in this table.

Table 4. Available parameters for each of the knowledge discovery tools

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
AlgorithmID	unsigned int	10	NOT NULL
Name	varchar	45	NOT NULL
Description	mediumtext		NOT NULL
Type	unsigned int	10	NOT NULL
DefaultValue	varchar	45	default NULL
MinValue	varchar	45	default NULL
MaxValue	varchar	45	default NULL
Parameter	tinyint	1	NOT NULL default '1'
Visible	tinyint	1	NOT NULL
CanBeNull	tinyint	1	NOT NULL

Every data mining tool has its own list of features available to the user. These features are controlled by the parameters passed to the tool. Some parameters are available only to the system such as dataset filename, while others are available to the users. Each parameter has its description, parameter type and default value. Depending on the parameter type numeric parameters can have its minimum and maximum values, some parameters need input from the user, some can be empty and some are hidden. Parameters can be added or removed as, for example, a new version of data mining tool is added to the service.

Table 5. Knowledge discovery projects

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
UserID	unsigned int	10	NOT NULL
Description	mediumtext		NOT NULL
DatasetID	unsigned int	10	NOT NULL
Created	datetime		NOT NULL

Projects are a central part of knowledge discovery process from a user's point of view. Every knowledge discovery process starts with a new project, and every project belongs to only one user which is distinguished by user ID. Project contains dataset variations, data mining tasks and their reports. Users can also store project related comments and its documentation (table 7).

Table 6. Project related datasets

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
ProjectID	unsigned int	10	NOT NULL
Description	mediumtext		NOT NULL
DataLength	unsigned int	10	NOT NULL default '0'
Data	mediumblob		
Instances	unsigned int	10	NOT NULL
Attributes	unsigned int	10	NOT NULL
Created	datetime		NOT NULL

Each project has its initial dataset on which user makes modifications. These modifications, when saved, produce dataset variation. Each dataset variation can have its own variations. These modifications are made by the user as many times as needed, until user is satisfied with its results. Each dataset is a matrix composed of instances (rows) and attributes (columns). Each attribute is defined by its size and type. Dataset is stored in 'Data' field in a plain text format. This sets a limit to approximately 16 MB per each dataset.

Table 7. Project related documents and files

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
ProjectID	unsigned int	10	NOT NULL
OriginalName	mediumtext		NOT NULL
Name	mediumtext		NOT NULL
Path	mediumtext		NOT NULL
Type	varchar	10	NOT NULL
Created	datetime		NOT NULL

Each project can contain one or more documents or files related to the research within this project. The user can add, delete, download or view these files or documents.

Table 8. Dataset templates

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
Description	mediumtext		NOT NULL
DataLength	unsigned int	10	NOT NULL default '0'
Data	mediumblob		
Instances	unsigned int	10	NOT NULL
Attributes	unsigned int	10	NOT NULL
Created	datetime		NOT NULL

Dataset templates are datasets which any user can use as initial dataset for a knowledge discovery project. Templates can contain some retrospect or historical data. The structure of a table and its contents is identical to a table which holds project datasets (table 6) except for the project ID field.

Table 9. KD task priorities

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
Name	varchar	45	NOT NULL

KD task priorities are used by the service to determine priority and order of executing tasks. The higher priority the task will be processed sooner. The possible values defined by this table are High, Normal and Low. These values are descriptive and aren't used by task scheduler directly, but are used to present descriptive values to the user.

Table 10. KD task statuses

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
Name	varchar	45	NOT NULL

Every KD task can have its execution status. The status is used by both user and the Task scheduler to determine what tasks are to be executed, are executing or are finished. The status values defined by this table are Waiting, Running, Finished, Stopped and Error. These descriptive values are presented to the user.

Table 11. KD tasks

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
DatasetID	unsigned int	10	NOT NULL
PriorityID	unsigned int	10	NOT NULL
StatusID	unsigned int	10	NOT NULL
Description	mediumtext		NOT NULL
AlgorithmID	unsigned int	10	NOT NULL
DatasetID	unsigned int	10	NOT NULL
Created	datetime		NOT NULL
Started	datetime		default NULL
Finished	datetime		default NULL

Each dataset can have many data mining tasks run by the user, and each task belongs to one dataset. Dataset ID describes to which dataset this task belongs. KD task has its description, execution priority and status. User can stop, delete or postpone execution of any task. When user creates a new task he must also select which data mining tool is going to be used with this task along with its parameters. The task parameters related to the data mining tool are stored separately in a following table (table 12). The task scheduler modifies task status and its timestamps to notify both the user and the service of a task execution status.

Table 12. Data mining tool parameters for each KD task

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
TaskID	unsigned int	10	NOT NULL
ParameterID	unsigned int	10	NOT NULL
Value	varchar	255	default NULL

This table contains list of data mining tool parameters along with its default or user chosen value. Each parameter is owned by a specific task and its description is linked with parameter definition in data mining tool parameters table. The task scheduler passes these parameters to the data mining tool which then processes them and according to the passed parameters produces different reports or classification results.

Table 13. Reports generated by Report manager

Column name	Type	Size	Attributes
ID	unsigned int	10	NOT NULL auto_increment PRIMARY KEY
TaskID	unsigned int	10	NOT NULL
Description	mediumtext		NOT NULL
Report	mediumtext		NOT NULL
Created	datetime		NOT NULL

Each KD process produces different results and outputs generated by a data mining tool. These results and outputs are gathered and processed by the Report manager and then stored in this table as a KD report. These results and outputs can then be presented in many ways. For example, results can be presented to the user as a table, graph, sequence of numbers or a plain text.

6 Security of data transactions

As more IT activities are being automated and an increasing number of computers are being used to store vital and sensitive information, the need for secure computer systems becomes more apparent. Computer security consists largely of defensive methods used to detect and thwart would-be intruders.

Software may crash, communication networks may go down, hardware components may fail and human operators may make mistakes. As long as these failures cannot be directly attributed to some deliberate human action, they would not be classified as security issues. Accidental failures would count as reliability issues. Operating mistakes would be attributed to usability issues. Security is concerned with intentional failures. There may not always be a clear intent to achieve a particular goal, but there is at some stage a decision by a person to do something they are not supposed to do.
Protecting data from malicious computer users continues to grow in importance. Whether preventing unauthorized access to personal documents, ensuring compliance with externally imposed regulations, or ensuring the integrity of sensitive information, all applications require increased security to protect data.

With the proliferation of stored data in all environments, organizations face an increasing requirement to both temporarily and permanently retain information. The storage medium for housing this information becomes a prime target for attack. If an outsider can successfully penetrate the data storage, the intruder can potentially gain information that violates privacy, that discloses valuable secrets, or that prevents the access of legitimate users. The deleterious effects of such an attack are truly unquantifiable. The avoidance of this obviously unfavorable condition is a main focal point. It is of paramount importance to actively pursue the options for securing stored information and consequently develop potential schemes for ensuring information confidentiality, integrity, and availability without substantially degrading performance.

6.1 Security in the Healthcare Sector

Information and communication technologies (ICT) made their entry into the healthcare sector more than 30 years ago and are acquiring a more and more significant role with the generalized use of Healthcare Information Systems (HIS). As they evolve and become more integrated, HIS are incorporating data from electronic healthcare records (EHR), both clinical and administrative, into coherent databases. An electronic healthcare record is defined as a repository of data relative to the health of an individual receiving healthcare service, stored in a way that can be processed and transmitted in a secure fashion so that it becomes accessible to authorized users. Profiting from advances in the field of knowledge management, storage management and networking, EHRs have allowed

healthcare organizations to use the data they contain to generate new management indicators and produce new sources of knowledge.

In the healthcare sector, development of EHRs facilitate the access to clinical data and management information for all healthcare stakeholders, allowing them to make better informed decisions. This improvement of decisional quality affects many activities of the healthcare value chain, from the diagnostic of patients to the management of a hospital. The emergence of the EHR as well as the increase in the utilization of ICT in the management of and research in healthcare services has produced large quantities of data concerning patients, healthcare delivery and biomedical research. EHRs are integrating multimedia data formats that are aggregated in clinical data warehouses by an increasing number of Healthcare organizations.

In the healthcare sector, particular ethical considerations come into play, which have to be considered when security concerns are addressed. Information contained in the healthcare record of a patient needs particular protection. On the one hand the EHR of a patient can reveal incident and private information about him/her whereas on the other hand their divulgation can have significant and non-medical impacts for the individual (e.g. loss of employment). As certain authors describe, putting in place a safe infrastructure is essential for the acceptation of ICT by the users of the healthcare system.

In any fiduciary relationship, *'confidentiality is respect for people's secrets'*. In medical contexts, *'The patient's right to confidentiality of his/her own individual health care information'* emerges from *'ethical principles'*, commonly the *'right to autonomy'*. Medical confidentiality is legally and professionally binding, and confidential information is defined as *'details about an individual patient'*. Any ICT should abide by the confidentiality principle and thus should put in place all the appropriate security mechanisms in order to prevent any dissemination of sensitive data onto unauthorized individuals.

According to, the goals of information security in healthcare are:

1. To ensure the privacy of patients and the confidentiality of health care data (prevention of unauthorized disclosure of information)
2. To ensure the integrity of health care data (prevention of unauthorized modification of information)
3. To ensure the availability of health data for authorized persons (prevention of unauthorized or unintended withholding of information or resources)

Achieving those goals is a difficult task which requires extensive planning and research along with meticulous focus on possible software and hardware security loopholes.

6.2 Risk Analysis

Risk management of Healthcare Information Systems (HIS) is generally a subset of information system risk management [Watson, 2004]. Informally, risk is the possibility that some incident or attack can cause damage to an organization or individual. An attack against an IT system consists of a sequence of actions, exploiting weak points in the system, until the attacker's goals have been achieved. To assess the risk posed by the attack we have to evaluate the amount of damage being done and the likelihood of the attack occurring. This likelihood will depend on the attacker's motivation and on how easy it is to mount the attack. In turn, this will further depend on the security configuration of the system under attack.

Risk can be, informally, estimated to be in direct proportion of Assets, Threats and Vulnerabilities of any given HIS

$$\text{Risk} = \text{Assets} \times \text{Threats} \times \text{Vulnerabilities}$$

6.2.1 Assets

In an IT system, assets include:

- Hardware: laptops, servers, PDAs, medical equipment etc.
- Software: applications, database management system etc.
- Data and Information: essential data for running and planning the HIS, sensitive data about the patients, data about the medical personnel etc.
- Reputation

Identification of assets should be a relatively straightforward systematic exercise. They can be valued according to their importance.

6.2.2 Vulnerabilities

Vulnerabilities are weaknesses of a system that could be accidentally or intentionally exploited to damage assets. In an IT system, typical vulnerabilities include:

- Accounts with system privileges where the default password has not been changed
- Applications with unnecessary privileges
- Applications with known flaws
- Weak access control settings on resources

Vulnerabilities can be rated according to their impact (level of criticality). A vulnerability that allows an attacker to take over a systems account is more critical than a vulnerability that gives access to an unprivileged user account. A vulnerability that allows an attacker to completely impersonate a user is more critical than a vulnerability where the user can only be impersonated in the context of a single specific service.

6.2.3 Threats

Threats are actions by adversaries who try to exploit vulnerabilities to damage assets. There are various ways to identify threats. For example, Microsoft's STRIDE threat model for software security lists the following categories:

- Spoofing identities: the attacker pretends to be somebody else
- Tampering with data: e.g. security settings are changed to give the attacker more privileges
- Repudiation: a user denies having performed an action
- Information disclosure: sensitive information is disclosed to inappropriate parties
- Denial of Service (DoS): DoS attacks can make web sites temporarily unavailable
- Elevation of privilege: a user gains more privileges on a computer system than he/she is entitled to

Threats can be rated according to their likelihood. The likelihood depends on the difficulty of the attack, on the motivation of the attacker and on the number of potential attackers.

6.3 Front end security

Organizations that use the web to collect and transmit sensitive data need to secure their websites and indeed their communication routes. SSL (Secure Socket Layer) is the most widely used Internet Security Protocol, supported by all the major web browsers. SSL adds a security layer between application protocols and TCP, so applications explicitly have to ask for security. Thus, application code has to be changed, but the required changes are not much more than edit operations, for example replacing a TCP connect call in the pre-SSL application with an SSL-connect call. The SSL-connect call will initiate the cryptographic state parameters and make the original TCP connect call.

Client and server have to protect the parameters of the security contexts they have established. Otherwise, the security provided by SSL will be compromised.

6.3.1 Login Authentication

Each user of the platform will be assigned security credentials and a profile. The credentials will be relevant to the user authentication during log-on whereas the profile will be used in order to attribute to the user appropriate permissions and privileges. More advanced authentication techniques (e.g. based on smart cards, ID cards, fingerprints, voice, retina scans etc.) will be investigated. Even though this goes beyond the HEARTFAID project, it is aimed to provide the premises for such technologies to be easily integrated into the platform at a later stage.
Unauthenticated users may only access the login page. The server validates the user who, then, will gain to the various services available via the web portal. The authentication, authorization and profiling procedures will be totally transparent to the user.

The authentication scheme that is implemented is based on the Kerberos authentication protocol. The latter allows individuals communicating over a non-secure network to prove their identity to one another in a secure manner. Its designers aimed primarily at a client-server model, and it provides mutual authentication — both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks.

There are several phases to Kerberos authentication. In the first phase, the client obtains credentials to be used to request access to 'kerberized' services. In the second phase, the client requests authentication for a specific service. In the final phase, the client presents those credentials to the service.

Kerberos is an authentication protocol, not an authorization protocol. That is, it verifies the identities of both the client and the server, but it does not include any information about whether the client has a right to use the services provided by the server. The authorization process will be a matter of the individual applications hosted on the HEARTFAID server.

6.3.2 Multiple Module Identification

An important aspect to be taken into account concerns the access modalities of the different modules operating within the platform. The platform, due to the integration of the various services onto the middleware platform, bridges the operation of the different, independent modules. Each of these services is provided with its own authentication procedure and profiling mechanisms.

It is out of doubt that the platform should guarantee a single sign-on approach, whereby the user will be authenticated only once and the platform will be in charge in order to invoke all the authentication procedures of the modules

involved for the services requested. This activity should totally be transparent for the user itself.

6.3.3 Logging

Session information exists for every user logged onto the web application at any given moment in time. Both successful connections and requests that violate the current user's access rights are written to a file for later inspection. In either case, the user will be notified of either an access violation or of a valid application invoking request.

6.3.4 iFrame Security Issues

iFrames have been implicated in many malicious code attacks, due to a series of common vulnerabilities. This was evident in many 2007 web based threats, notably the so-called 'Italian Job' of June, 2007. An iFrame can be planted on an unsuspecting legitimate website, leading the casual viewer into an infection threat.

Another issue that exposes an iFrame vulnerability involves the IE Document.ExecCommand() method being executed and taking inappropriate action on a user's computer. The IE security model's restriction of not letting the Document.ExecCommand() execute, is not present if the method is invoked on an iFrame. This could allow a malicious web site operator to read the contents of files on visiting users' computers, if he/she knew the name of the file and the folder in which it resided. The vulnerability would not allow the malicious user to list the contents of folders, create, modify or delete files, or to usurp any administrative control over the machine.

6.3.5 Cross-Frame Scripting and Security

Since a browser can simultaneously display unrelated documents in its various windows and frames, certain rules must be enforced to protect data integrity and privacy of information. All rules about script interaction apply equally to windows, dialog boxes, frameSets, frames, and iFrames.

For most content, only interactions with content from the same domain are allowed. Scripts that attempt to access parts of the object model to which they do not have access are blocked with a "permission denied" error.

While domain security can prevent certain types of content interaction, it is important to understand that this restriction is necessary to ensure security. For

example, without domain security, a rogue page could "snoop" on another page or, using DHTML, manipulate its content.

Microsoft has identified a serious security vulnerability that could potentially affect many web sites and web site users. The vulnerability, known as "Cross-Site Scripting", is equally possible on all vendors' products, and does not result from a defect in any of them. Instead, it results from certain common web coding practices.

Cross-Site Scripting would potentially enable a malicious user to introduce executable code of his choice into another user's web session. Once the code is running, it could take a wide range of actions, from monitoring the user's web session and forwarding a copy to the malicious user, to changing what's displayed on the user's screen. Even more seriously, the script could make itself persistent, so that the next time the user returned to the web site; the malicious user's script would start running again.

The long-term solution to the problem requires web sites and web site developers to review their code and verify that it adheres to secure coding practices. However, both users and webmasters must be aware of this problem and make sure that they take all the appropriate steps in order to minimize the possibility that such a deleterious event will occur.

7 Conclusion

Within the framework of the D31 module "Knowledge Discovery Systems", we have been concerned with the development, testing, implementation and online integration of the appropriate Knowledge Discovery (KD) system that will provide an efficient mechanism that will satisfy the HEARTFAID project's prerequisites.

We have expanded on the applicability, the efficiency, the intuitiveness and the accuracy of the Random Forest (RF) model that has, for such reasons, been incorporated and implemented. As more and more data are available and vast data repositories are to be analyzed, high processing power along with a systematic, proficient and effective method for data mining is anticipated. Especially in the health sector where accuracy, meticulousness and exactitude are of profound importance, intuitive algorithms have to be applied.

Within the project's framework, preliminary tests have been performed of the completeness and efficacy of the RF modeling method and have all indicated its appropriateness for the purposes and provisions of HEARTFAID. Not only does it assimilate an unparalleled accuracy among similar algorithms but also it may efficiently execute regardless of the magnitude of the appropriate database. Furthermore, given the increased complexity of the medical input along with its high dependence on specific circumstance, RF has the additional advantage of being able to handle thousands of input variables without variable deletion or withdrawal. Moreover, it offers an experimental method for detecting variable interactions, something particularly noteworthy for the later interpretation of the output.

Even though the RF algorithm has been proven to be mostly appropriate for HEARTFAID, additional provisions have been made in order for system administrators to be able to add new data mining tools. Not only can they modify the templates for the datasets that will be available to all the users of the platform, but also they will be able to manage the Knowledge Discovery algorithms that will be attainable for performing KD tasks.

The KD tasks are all integrated within a single web service that will simplify the Knowledge Discovery process so as to enable even an amateur in the field to be able to deploy, execute and interpret KD tasks. Furthermore, the output should facilitate easy navigation among the various datasets, models and results. The idea is rather simple, yet very profound; each user may start his/her-own project with a specified dataset that has been a priori uploaded onto the repository. The dataset could derive from either a predefined template (historical data) or form HEARTFAID platform's already accumulated data. Each user's KD project will be made private. Furthermore, every project should enable the user to attach relevant documents to the performed KD process, describe the models acquired by

executing modeling tasks and/or outline any changes that were made to the studied dataset.

The knowledge discovery service implements via a series of interconnected web pages or web forms. New projects, new tasks, dataset manipulation and result revision will all be made available online via the webpage. Auxiliary pages will incorporate and implement new user registration, existing user login and dataset selection. There will also be some additional web pages aimed at the platform's administrators letting them add new tools and templates. The system's admins will be also able to control each user's permissions and attributes letting them access to the appropriate to their credentials forms and services.

The project's web forms will curtail all the relevant info about the activity of a particular user. In particular, a list of his/her projects, a list of uploaded documents, an outline of project components will all be made available on the web form. The user will be able to manage the datasets, add – delete – modify appropriate documents, create – manage – extricate concocted projects and navigate to all of the project components (e.g. datasets, models, modeling results).

Each webpage, relative to its content and purpose, will incorporate appropriate links to websites, web-documents, book citations and bibliography that will empower the user to capitalize on the offered services. Provisions have been made in order to enable the platform's administrator and users to enrich and update the available resource lists.

All the data mining tasks performed on the platform will be made feasible via a task scheduler. The latter has the ability to execute multiple tasks at the same time (multitasking). It is up to the user to schedule the KD tasks (s)he has in mind as it is possible to stop, postpone or force start any data mining process. Once the task is complete, the results are stored in the web service repository for later evaluation and analysis by the user.

We have presented a concise account of all the data mining tasks and services offered by the platform along with the techniques and methods invoked in order to increase efficiency and effectiveness. We believe that the platform will be so robust as to allow enhanced usage and provide the best possible means to the medical professional to derive the optimal outcome within the HEARTFAID's framework.

8 References

1. Random forest home page, http://www.stat.berkeley.edu/~breiman/
2. Breiman, L. (1996a), Bagging Predictors, Machine Learning, Vol. 24, No. 2, pp. 123-140. http://citeseer.ist.psu.edu/breiman96bagging.html
3. Breiman L., "Random Forests", Machine Learning, volume 45, Kluwer Academic Publishers, Boston, pp. 5-32., 2001., http://citeseer.ist.psu.edu/breiman01random.html
4. Breiman L., Cutler A., Random Forests, http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm
5. Breiman L., Cutler A., ENAR Short Course, http://www.stat.berkeley.edu/users/breiman/RandomForests/ENAR_files/frame.htm
6. PARF project home page, http://www.irb.hr/en/research/projects/it/2004/2004-111/
7. Weka project home page, http://www.cs.waikato.ac.nz/~ml/weka/
8. An Overview of Computer security, Shireesh Reddy Annam, cs/0110043v1 (2001)
9. Computer Security, Dieter Gollmann, John Wiley and Sons Ltd, Second Edition (2006)
10. Securing Data in Storage: A Review of Current Research, Paul Stanton, cs/0409034v1 (2004)
11. Safe Data Sharing and Data Dissemination on Smart Devices, Luc Bouganim, Cosmin Cremarenco, François Dang Ngoc, Nicolas Dieu, Philippe Pucheral cs/0510013v1 (2005)
12. Confidentiality Issues in Information Systems in Social Care, Janet Cooper, Christine Urquhart, Glasgow Caledonian University (2004)
13. http://www.leger.ca/pages/EN/healthcare.htm
14. http://msdn.microsoft.com/msdnmag/issues/06/11/ThreatModeling/default.aspx
15. Medical ethics: four principles plus attention to scope, Gillon, R. (1994). British Medical Journal. 309: 184-188
16. Minimal breaches of confidentiality in health care research: a Canadian perspective. Journal of Medical Ethics, Emson, H. E. (1994). 20, 3: 165-168
17. Users' views of palliative care services: ethical implications. Nursing Ethics, Woods, S.; Beaver, K. and Luker, K. (2000), 7, 4: 314-326
18. Privacy, confidentiality, and electronic medical records, Barrows, RC Jr, Clayton, PD, Journal of the American Medical Informatics Association, 1996, pages 139-148
19. Writing Secure Code, Howard Michael and David LeBlanc, Microsoft Press, Redmond, WA, 2nd Edition (2002)
20. The Resurrecting Duckling: security issues for ubiquitous computing, Stajano, F.; Anderson, R. Computer, Volume: 35 Issue: 4 Part: Supplement Page(s): 22-26 (2002)

21. http://en.wikipedia.org/wiki/Kerberos_%28protocol%29
22. http://developer.apple.com/documentation/Security/Conceptual/Security_Overview/Concepts/chapter_3_section_8.html#//apple_ref/doc/uid/TP30000976-CH203-CHDGBCDF

Appendix A - Sample KD report

Dataset summary

Dataset name:	sample.arff
Number of trees in RF:	100
Number of cases:	6851
Number of attributes:	14
Classes in the problem:	2 {Alive, Dead}
Number of used attributes:	13
Attributes to split on:	4

Problem runtime options:

Generate confusion matrix:	on
Generate variable importances:	on
Generate variable interactions:	on
Proximate cases radius:	200
Number of prototypes per class:	2
Class weights reassignment:	Alive=1; Dead=4

Runtime command:
./parf -t MIS_ANMCO/M65_res/m65_short_TS.arff -n 100 u Sex -tc -i -ii -p 200 -yn 2 -yp 100 -ya -w 1,4 --verbose

Model summary:

OOB trainset accuracy: 33.73%
Kappa: 0.2635

Table 14. Confusion matrix

		Target class		
		No Class	**Alive**	**Dead**
Classified as	**Not classified**	0	0	0
	Alive	0	3477	1545
	Dead	0	755	1074

Table 15. Attribute importances

Tag	Imp	Z-Sc	Significance
DIABETES	5.19	23.45	0.000000
NYHA	3.12	15.03	0.000000
PAS	1.99	10.06	0.000000
THIRDTONE	1.21	9.01	0.000000
BETABLOCK	1.64	8.87	0.000000
PAD	0.99	6.54	0.000000
BMI	1.17	6.11	0.000000
SMOKE	0.95	5.36	0.000000
CF	0.85	5.18	0.000000
Age	0.54	3.45	0.000282
ACE	0.21	2.02	0.021435
DIURETICS	0.16	1.64	0.050576

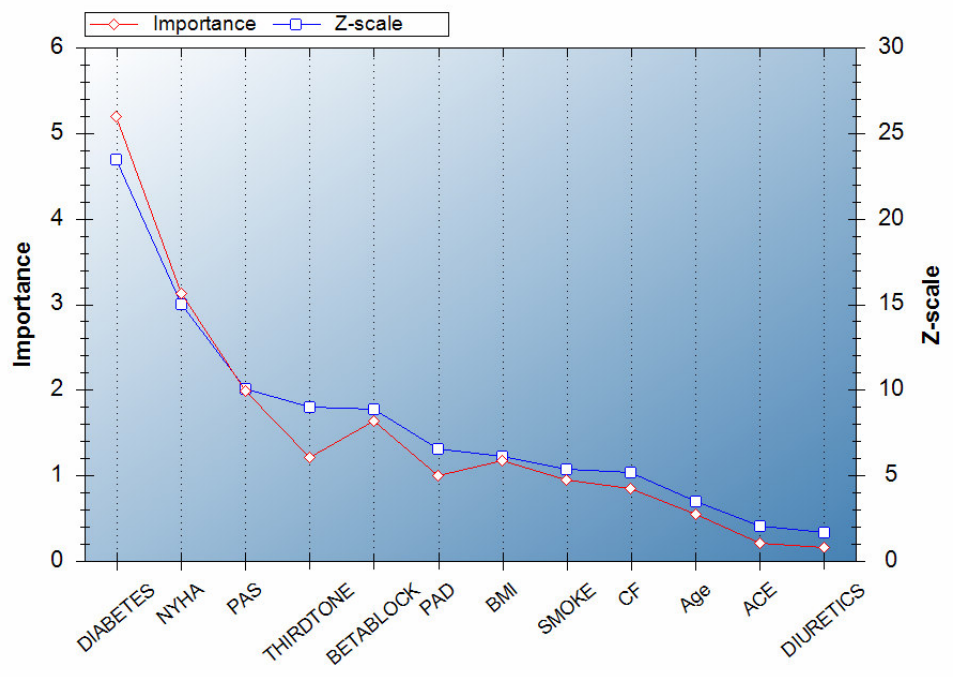


Figure 13. Attribute importance plot

Table 16. Variable interactions

	Age	NYHA	BMI	PAS	PAD	CF	THIRD-TONE	SMOKE	DIABE-TES	ACE	DIURE-TICS	BETA-BLOCK
Age	0	-67	-57	-129	-64	-40	-49	-5	-25	-7	-12	-30
NYHA	-67	0	-72	-59	-77	-95	-27	-55	-55	-34	-102	-71
BMI	-57	-72	0	5	-77	-100	-28	-57	-89	-11	-14	-103
PAS	-129	-59	5	0	-50	-94	-74	-114	-90	-11	11	-1
PAD	-64	-77	-77	-50	0	-77	-73	-33	-28	-26	-40	-33
CF	-40	-95	-100	-94	-77	0	8	-33	-44	-99	-26	-53
THIRDTONE	-49	-27	-28	-74	-73	8	0	-71	-85	-43	-9	-88
SMOKE	-5	-55	-57	-114	-33	-33	-71	0	-11	-67	32	-115
DIABETES	-25	-55	-89	-90	-28	-44	-85	-11	0	-10	-52	-48
ACE	-7	-34	-11	-11	-26	-99	-43	-67	-10	0	-46	-4
DIURETICS	-12	-102	-14	11	-40	-26	-9	32	-52	-46	0	-55
BETABLOCK	-30	-71	-103	-1	-33	-53	-88	-115	-48	-4	-55	0

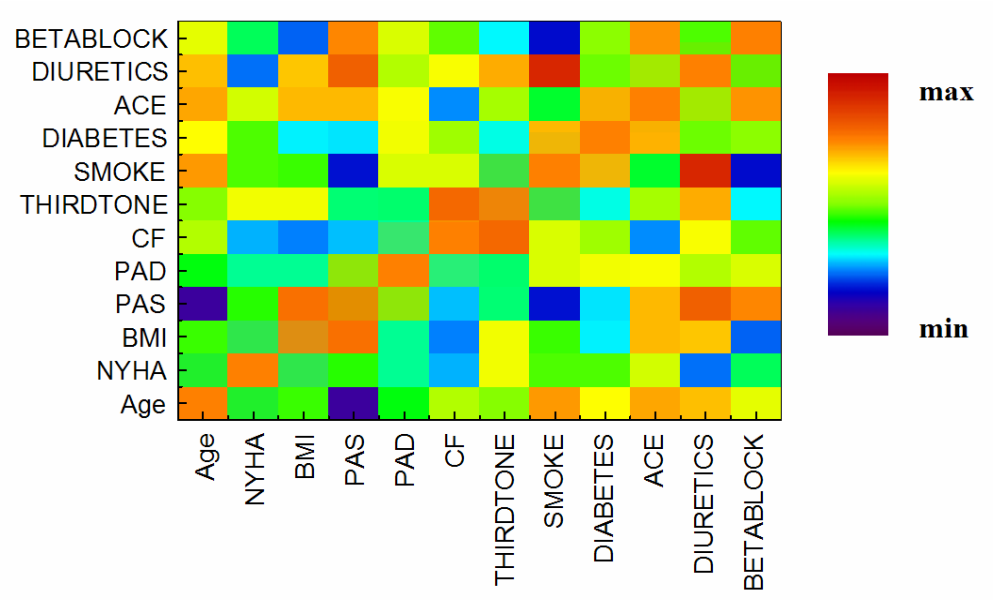


Figure 14.Variable interaction plot

Table 17. Class prototypes - summary

Class	Alive			Alive			Dead			Dead		
Prototype number:	1			2			1			2		
Prototype size:	100			98			93			83		
*** Variable/type**												
Age / num	70	72	75	67	69	72	71	76	83	71	75	78
NYHA/cat												
Cat: 1	8			31			2			0		
Cat: 2	91			68			14			9		
Cat: 3	0			1			63			82		
Cat: 4	1			0			21			9		
BMI / num	26.6	28.1	29.4	24.2	25.3	27	21.2	23.4	24.3	18.8	20.2	23.9
PAS / num	130	135	140	130	140	150	105	115	125	90	100	110
PAD / num	80	80	80	80	80	85	70	70	75	60	60	70
CF / num	76	85	90	56	60	65	68	70	82	70	75	85
THIRDTONE / Cat												
Cat: No	94			98			33			86		
Cat: Yes	6			2			67			14		
SMOKE / Cat												
Cat: ?	72			10			89			38		
Cat: Ex	13			70			6			47		
Cat: No	11			15			4			12		
Cat: Yes	4			5			1			3		
DIABETES / Cat												
Cat: ?	1			0			83			9		
Cat: No	86			92			16			66		
Cat: Yes	13			8			1			25		
ACE / Cat												
Cat: No	54			16			27			22		
Cat: Yes	46			84			73			78		
DIURETICS / Cat												
Cat: No	7			75			2			0		
Cat: Yes	93			25			98			100		
BETABLOCK / Cat												
Cat: No	2			1			98			98		
Cat: Yes	98			99			2			2		

- numeric variables are represented via min-median-max values for categorical variables % distribution per category values are given

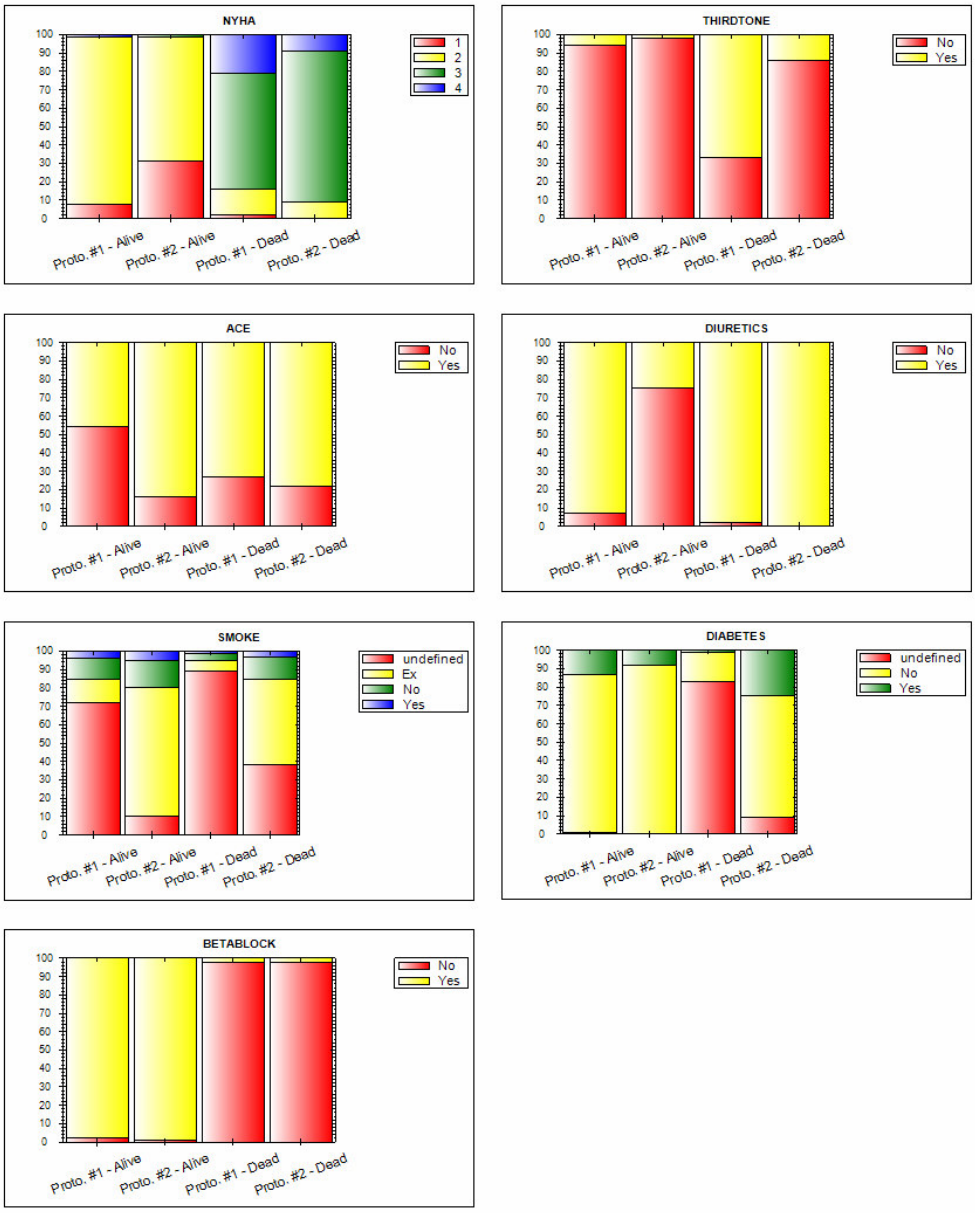


Figure 15. Visual representation of categorical attributes across four distinct prototypes. This graph shows only categorical attributes, as numerical attributes are represented by different graph. For each prototype there is a bar showing percentage of each category - value in different colour.

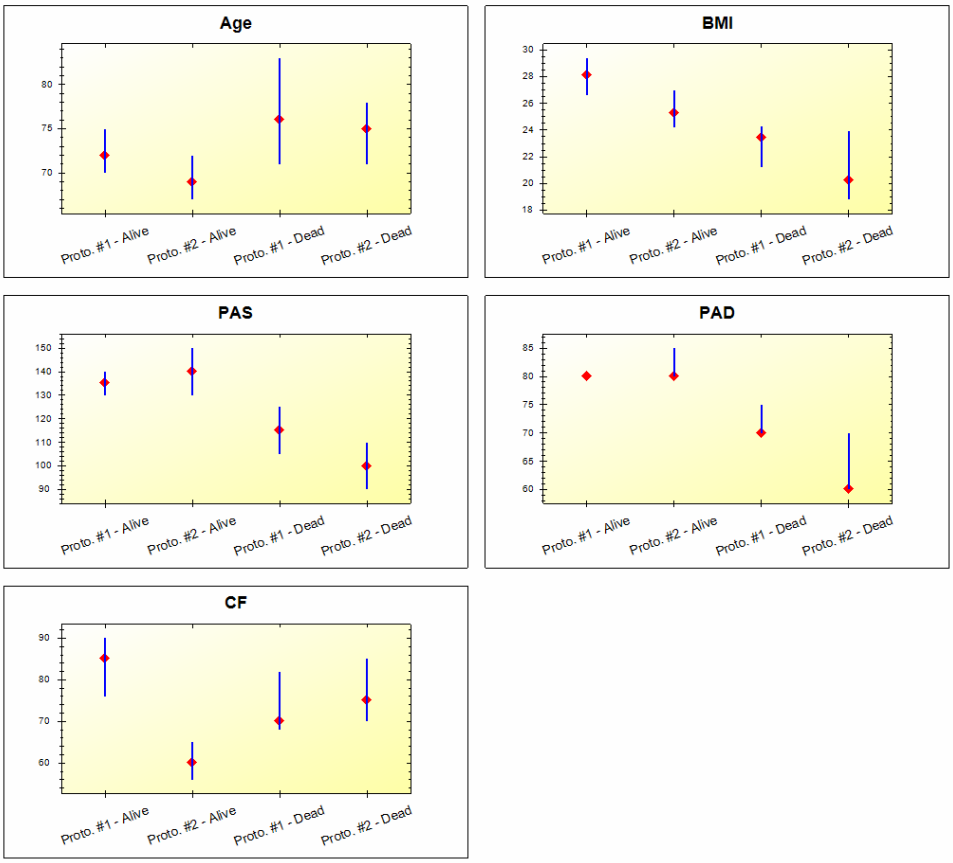


Figure 16. Visual representation of the distribution of numerical attributes across four distinct prototypes. For each prototype there is a bar depicting minimum, median and maximum of the distribution of values for the particular attribute.