



LarKC

*The Large Knowledge Collider
a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

D11.3 Visualization and Analysis tool and relevance feedback for instrumentation and monitoring

Coordinator: Raluca Brehar (UTC)

With contributions from: Raluca Brehar (UTC), Ionel Giosan(UTC), Mihai Negru (UTC), Andrei Vatavu (UTC),
Ioan Toma (SG), Cristi Vicas (UTC)

Quality Assessor: Danica Damljanovic (USFD)

Quality Controller: Sergiu Nedevschi (UTC)

Document Identifier:	LarKC/2008/D11.3/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	May 16, 2011
State:	final
Distribution:	public

EXECUTIVE SUMMARY

DOCUMENT INFORMATION

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	11.3	Title	Visualization and Analysis tool and relevance feedback for instrumentation and monitoring
Work Package	Number	11	Title	Instrumentation and Monitoring

Date of Delivery	Contractual	M38	Actual	31-May-11
Status	version 1.0.0		final <input checked="" type="checkbox"/>	
Nature	prototype <input checked="" type="checkbox"/> report <input type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	UTC, Softgress			
Resp. Author	Raluca Brehar	E-mail	raluca.brehar@cs.utcluj.ro	
	Partner	UTC, Softgress	Phone	+40 745 309857

Abstract (for dissemination)	
Keywords	Instrumentation, Monitoring, Visualization, Architecture

PROJECT CONSORTIUM INFORMATION

Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 STI · INNSBRUCK	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI), Universitaet Innsbruck, Innsbruck, Austria Email: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle CEFRIEL - SOCIETA CONSORTILE A RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCORP, RAZISKOVANJE IN EKSPERIMEN- TALNI RAZVOJ D.O.O., Ljubljana, Slovenia Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo Höchstleistungsrechenzentrum, Universitaet Stuttgart Stuttgart, Germany Email : gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler, Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK Email: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA http://www.utcluj.ro/		Prof. Dr. Eng. Sergiu Nedevschi TECHNICAL UNIVERSITY OF CLUJ-NAPOCA Cluj-Napoca, Romania E-mail: sergiu.nedevschi@cs.utcluj.ro
SOFTGRESS S.R.L. http://www.softgress.com/		Dr. Ioan Toma SOFTGRESS S.R.L. Cluj-Napoca, Romania E-mail: ioan.toma@softgress.com

TABLE OF CONTENTS

LIST OF FIGURES	7
1 INTRODUCTION	8
2 VISUALIZATION AND ANALYSIS ARCHITECTURE	9
2.1 Overall specification	9
2.2 Components	9
3 RELEVANCE FEEDBACK THEORETICAL BACKGROUND	23
3.1 Principal component analysis	24
3.2 Correlation based feature selection	24
3.3 Clustering and regression	25
3.4 Kernel Canonical Correlation Analysis	27
4 RELEVANCE FEEDBACK - REVISED SCENARIOS	29
4.1 Scalability analysis	29
4.2 Bottleneck prediction	31
4.3 Work-flow prediction based on raw data	33
5 RELEVANCE FEEDBACK ARCHITECTURE	35
5.1 RF training	35
5.2 RF application	36
6 END-USER GUIDE FOR VISUALIZATION	37
7 END-USER GUIDE FOR RELEVANCE FEEDBACK	39
7.1 RF training	39
7.2 RF application	42
8 CONCLUSION	43

LIST OF FIGURES

2.1	Visualization Architecture	9
2.2	Interaction and relationship between MVP components	11
2.3	Visualization Architecture - Class Diagram	13
2.4	Visualization Architecture - Class Diagram	14
2.5	Visualization Architecture - Class Diagram	15
2.6	GWT RPC for Prediction Module	17
2.7	MySQL Relational Database Schema	20
2.8	RRD access reading interface	21
2.9	Visualization Flowchart	22
3.1	Relevance Feedback Model	23
5.1	Tiers of the RF architecture	35
5.2	Detailed architecture of RF	36
7.1	Main window of the training module	39
7.2	Summary of sample training instances that were loaded by RF training module	40
7.3	RF training - the steps needed for performing the training and prediction	41
7.4	Relevance feedback integrated in the visualization component	42

1. Introduction

LarKC is developing a platform that enables the development of large-scale reasoning applications using and combining techniques from various Semantic Web related research fields. The plugable architecture of LarKC enables the interested LarKC users to test their ideas for doing reasoning. It allows them to integrate and deploy their own components, known as plug-ins in LarKC terminology, in the platform, to flexibly connect them in order to build workflows, to run and to test them. The last feature is supported by a set of tools built by WP11 that enables instrumentation and monitoring of LarKC plugins, workflows and platform. We call this tool set SIM - Semantic Instrumentation and Monitoring. SIM enables the instrumentation and monitoring of LarKC applications in particular and any java distributed system in general. It offers the means for developers to specify the metrics of interest, to instrument the code, to collect and observe how well the system and its components are performing. For example, LarKC developers can find out, using SIM how performant their plugins and workflows are, how many resources they consume, how much data they consume and produce, etc. In [1], we described the first version of SIM, the *instrumentation and monitoring* solution developed in LarKC. There are five major components that are part of SIM: instrumentation code, agents, server, visualization and relevance feedback. While the first two (i.e. instrumentation code and agents) are responsible for setting up the process and collecting the monitoring data, the last two (i.e. visualization and relevance feedback) are processing the raw monitoring data enabling the end user to take full advantage of it. More precisely, the visualization component displays monitoring data collected from the LarKC platform, plug-ins and workflows and the relevance feedback component learns uses data mining techniques and performs machine learning on top of the monitoring data. In this deliverable we describe the advances and latest developments of the two SIM components that are consuming the monitoring data namely visualization and relevance feedback.

This deliverable is organized as follows. Chapter 2 describes the final architecture and the technical implementation of the visualization component. It provides details on the front-end and back-end parts of the visualization as well as on the technologies used to implement it. Chapter 3 presents the mathematical model and algorithms that are implemented as part of the relevance feedback component. Chapter 4 revisits the scenarios proposed in [2] by specifying the input and output metrics on which the relevance feedback will operate. The architecture and the technical implementation details of the relevance feedback component are presented in Chapter 5. Chapter 6 and Chapter 7 contain a basic end user guide on how to use the visualization and relevance feedback tools. Finally, Chapter 8 concludes the deliverable.

2. Visualization and Analysis Architecture

2.1 Overall specification

Our visualization module is responsible for (1) displaying data and metrics about the LarKC platform, plug-ins and workflows and (2) for sending input metrics to the relevance feedback module and displaying the results provided by the relevance feedback. The visualization is a web-based, client-server application with several modules for representing real-time and historical data, composed of atomic and compound metrics about:

- LarKC platform in general.
- Queries that were passed to the LarKC platform.
- Workflows used to solve the queries.
- Plug-ins that compose the workflows.

Figure 2.1 describes the general architecture of the visualization framework.

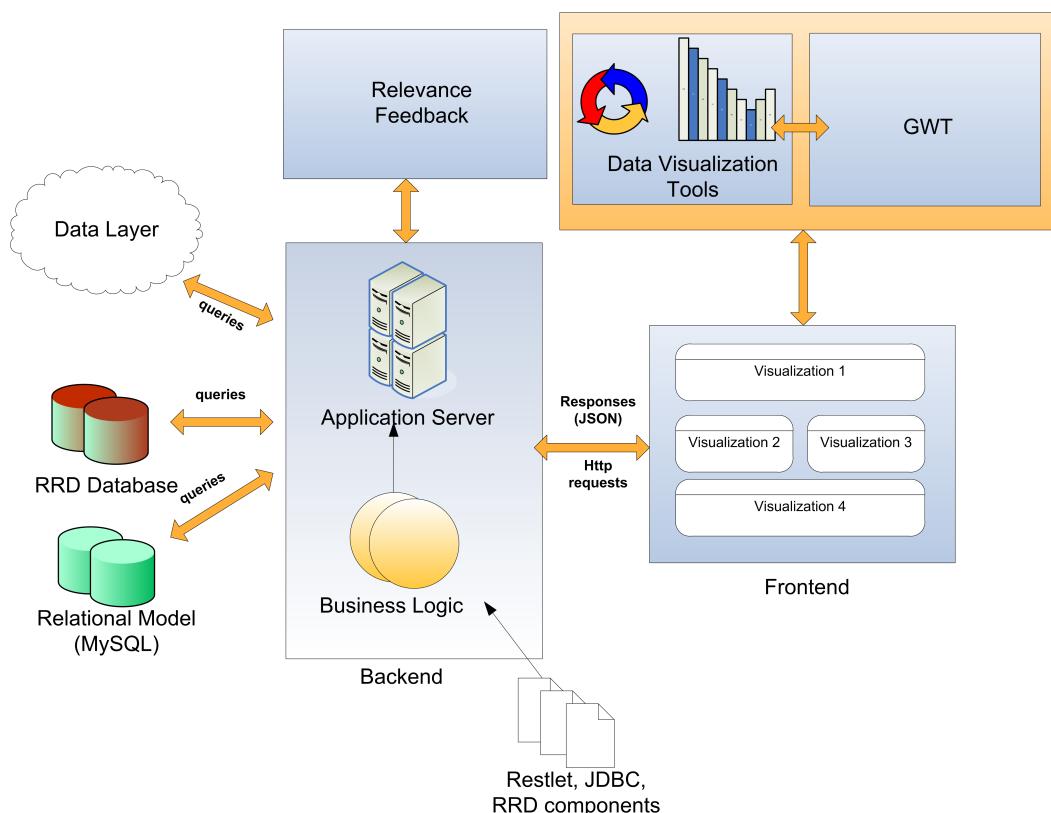


Figure 2.1: Visualization Architecture

2.2 Components

The visualization module is composed by the following main components:

Client-Side Component

Represents the front end part that runs in the end user's web browser as a rich content AJAX application. For the client-side component implementation we choose Google Web Toolkit (GWT). By using GWT the front end application is written entirely in Java and deployed as a highly optimized JavaScript application. Some of the advantages of using GWT as the main development environment for the client-side component are summarized below:

- The Java code is deployed as a highly optimized JavaScript
- Cross Browser support
- External libraries for data visualization can be easily integrated into GWT.
- Communication support with Server Side logic: RPC and JSON
- Easy development and debug. The GWT source code can be written entirely in Java.
- Asynchronous call support.
- Web application performance and fast code execution

One problem with GWT is that GWT was meant to be used as a standalone web application development platform, it was designed to work with a single HTML page while traditional web applications have several HTML pages in them. Another issue is that GWT does not allow the loading of multiple modules into a single page and does not integrate well with existing HTML elements. To overcome this problems the solution employed was the use some of the GWT best practices listed below:

- Model View Presenter pattern - MVP
- History Management
- Application Controller
- UI Binding

In the following we will describe these concepts.

Model View Presenter

One of the main advantages of the MVP (Model View Presenter) pattern is that it decouples development in a way that allows multiple developers to work and test the web application simultaneously. A MVP application has four main components:

- Model - A model encompasses business objects
- View - A view contains all of the UI components that make up an application. This includes any tables, labels, buttons, text boxes, etc... Views are responsible for the layout of the UI components and have no notion of the model. Switching between views is tied to the history management within the presentation layer.

- Presenter - A presenter contains all the logic of the application, including history management, view transition and data sync via Remote Procedure Calls back to the server. For every view there exists a presenter to drive the view and handle the events that are sources by the user interface widgets within the view.
- Application Controller - The application controller handles logic that is not specific to any presenter and instead resides at the application layer. This component contains the history management and view transition logic.

The interaction and relationship between these components is presented in figure 2.2

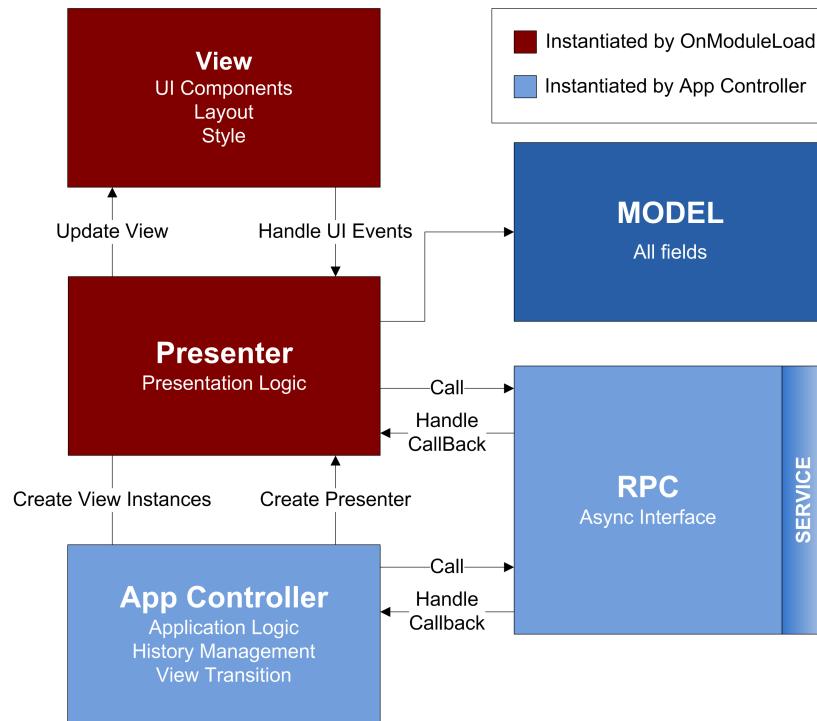


Figure 2.2: Interaction and relationship between MVP components

History Management

Usually, Ajax based applications don't interact with the browser history as the static pages. This may be frustrating for users when, for example, they want to navigate back to previous pages, because an Ajax application usually is a single page. Therefore we integrated GWT's history mechanism that responds to this problem. For each Visualization page that supports history management, the application generates a unique history token that is saved in browser history as a URL string, beginning with '#'. For example we can access the Workflows User Interface by adding "#workflows" to the end of the URL: <http://localhost/Visualization.html#workflows>.

Application Controller

We need a separate Application Controller class to handle logic beside any presenter at a higher layer of our application. Application Controller class includes history management, application event handling and logic to connect other application in-

stances.

UI Binders

Instead of writing user interfaces through code, UiBinder represent a more natural and simple way to specify this UI in a more understanding declarative XML template. Some advantages of using UIBinder are:

- UIBinders provide a better separation between User Interface and Application Behavior (Code).
- UI interfaces are easier to implement with XML, HTML and CSS than Java code.
- Works well with GWT's i18n internationalization support.
- UIBinders are easier to maintain and reuse in other applications.
- Lightweight HTML elements

However there are cases when programmatic specification of a widget is used, for example in table widgets, tree menu etc. Each UiBinder file has an associated class that allows programmatic access to the widgets.

Other Visualization Libraries Used

In order to display the relevant metrics of the Larkc Instrumentation and Monitoring architecture we are currently using two other libraries that are compatible with the google web toolkit:

- **Open Flash Charts GWT** provides a simple to use chart widget for GWT based on Open Flash Chart 2. The library includes the needed flash insertion, update and manipulation methods for the chart widget. It also includes a POJO model for the chart elements and components that assist in the generation of the JSON to provide the correct chart data for OFC 2.x API. Included Charts are: pie charts , bar charts, 3D bar charts, line charts, area charts, scattered charts, etc.
- **SmartGWT** is a GWT-based framework that allows you to not only utilize its comprehensive widget library for an application's user interface, but also ties these widgets in with your server-side for data management. Smart GWT provides an end-to-end application architecture, from UI components to server-side transaction handling. Each data widget can be connected transparently to a data source without using a data object model, like in GWT.

System Architecture

The Visualization System Architecture is presented in figure 2.3. There are several categories of classes:

- Visualization User Interface and Associated UI Binder contain the main application container in which all other UI widgets are included based on user actions.
- View Classes (Platform, Workflows and Prediction) - implement User Interface widgets. Each UI Class may have an associated UI Binder.

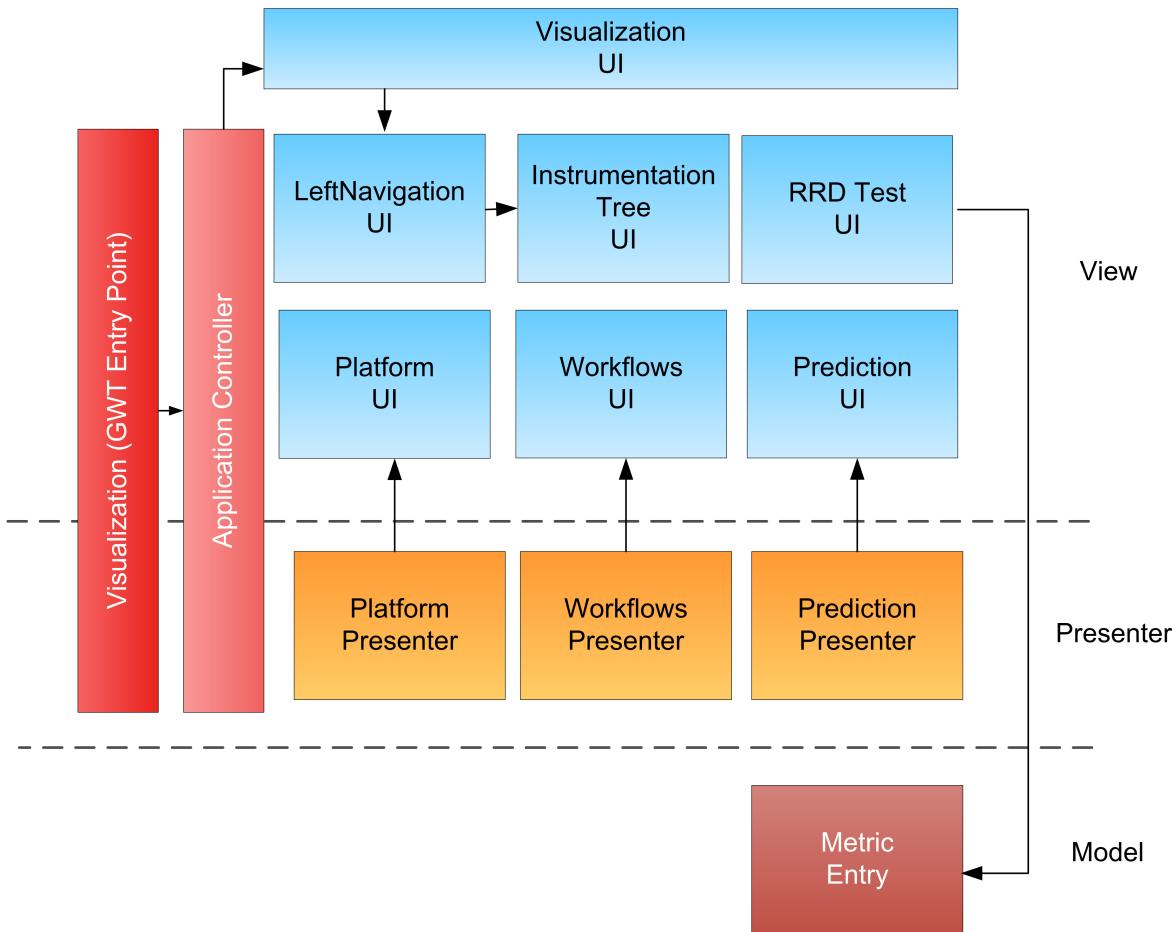


Figure 2.3: Visualization Architecture - Class Diagram

- Presenter Classes (PlatformPresenter, WorkflowsPresenter and PredictionPresenter) - are responsible for implementing the logic of the associated UI Widgets. The presenter classes are instantiated in the application controller once a new HistoryChangeEvent has been detected.
- Model Classes (Metric Entry) encompass business objects, in our case we use Metric Entries as model objects. Normally. Model classes are passed to the UI Widgets as model Data.
- Application Controller Class handles logic at the application layer.
- Visualization - represents the main entry point of the application as each GWT application must contain an entry class. At this level the main visualization container and application controller are initialized.

Visualization Scenarios

For the visualization interface we have taken into account the following main scenarios:

- Display all metrics for the current platform (figure 2.4)
- Display all plugins for a given Workflow

- Display all queries for a given Workflow
- Display metrics for a selected query

Figure 2.5 presents a sequence diagram showing the interaction between Workflows user interface and server for the last three scenarios.

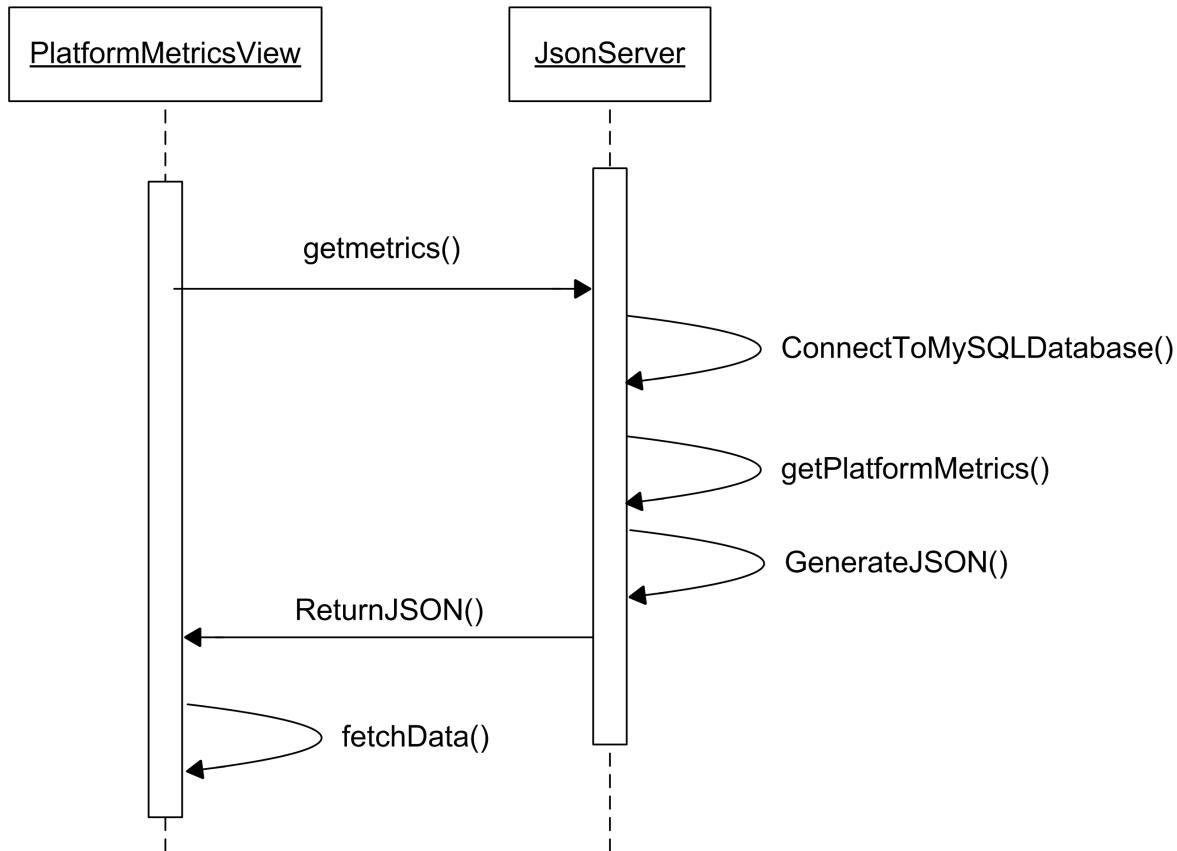


Figure 2.4: Visualization Architecture - Class Diagram

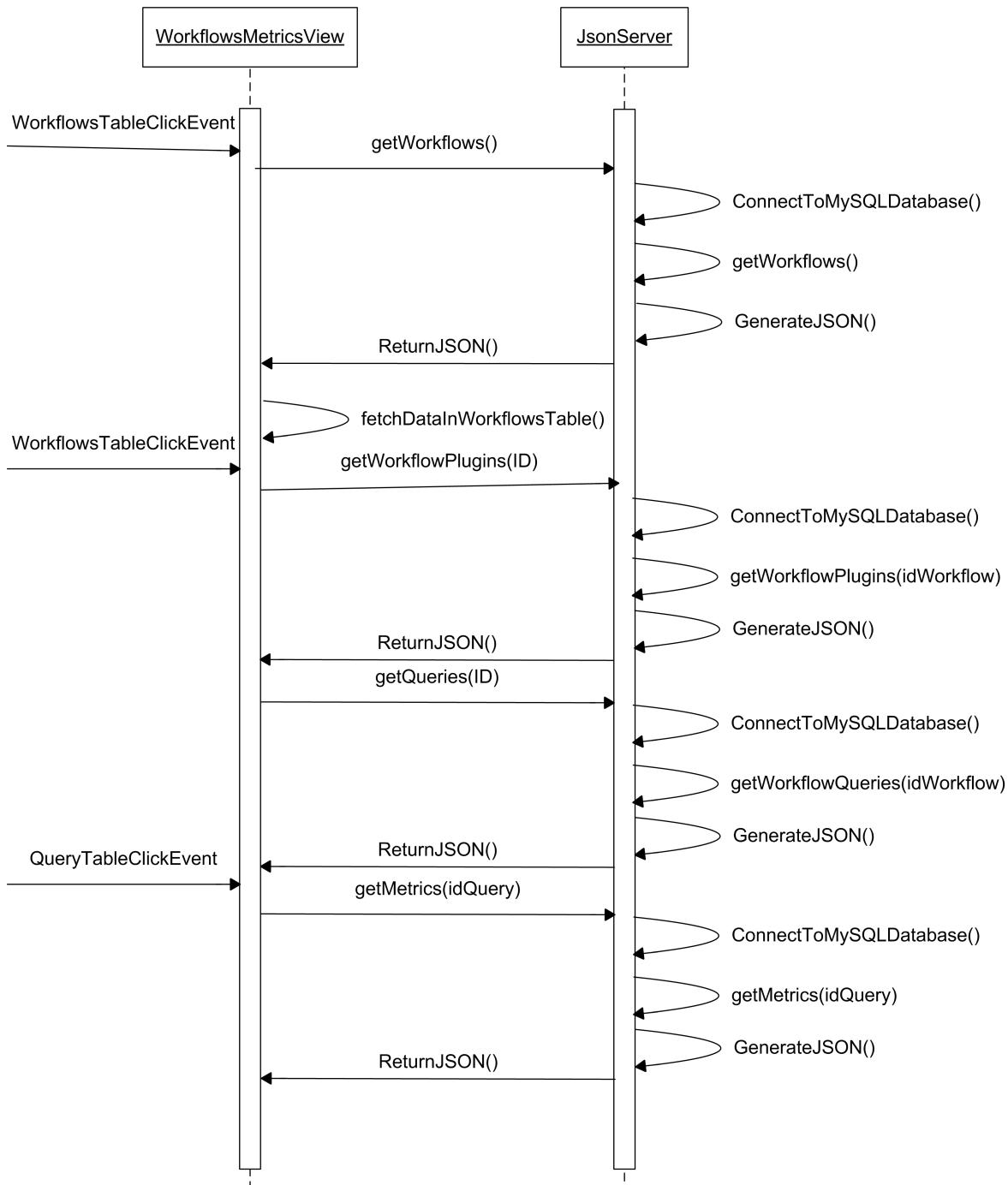


Figure 2.5: Visualization Architecture - Class Diagram

Server-Side Component

The server-side component includes a controller of business logic which coordinates requests from clients, as well as the data layer queries and responses. Based on client requests and data base (data layer) query results, actions are carried out by the server. Two mechanisms for client-server communication can be used: by making GWT Remote Procedures Calls or by retrieving JSON Data via HTTP requests.

RPC based Communication

The GWT Remote Procedure Call framework makes it easy for the client and server components of a web application to exchange Java objects over HTTP. The server-side code that gets invoked from the client is often referred to as a service. The implementation of a GWT RPC service is based on the well-known Java servlet architecture. Within the client code, we use a proxy class to make calls to the service. GWT will handle serialization of the Java objects passing back and forth the arguments in the method calls and the return value. When setting up GWT RPC, we focus on three elements involved in calling procedures running on remote servers.

- the service that runs on the server (the method being called)
- the client code that invokes the service
- the Java data objects that pass between the client and server

Both the server and the client have the ability to serialize and de-serialize data so the data objects can be passed between them as ordinary text. An important feature of GWT RPC communication is that GWT provides an embedded servlet container (Jetty) that host the servlets containing the RPC service implementation, thus allowing one to test and debug the application even in development mode. To set this up, one must add <servlet> and <servlet-mapping> elements to the web application deployment descriptor (web.xml) and point to the implementation class.

```
<!-- Servlets -->
<servlet>
  <servlet-name>PredictionServiceImpl</servlet-name>
  <servlet-class>ro.utcluj.larkc.visual.server.PredictionServiceImpl
  </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>PredictionServiceImpl</servlet-name>
  <url-pattern>/visualization/prediction</url-pattern>
</servlet-mapping>
```

In our implementation we use GWT RPC in the prediction and relevance feedback module. Figure 2.6 presents the Java classes involved in the client-server communication through the GWT RPC mechanism.

JSON Based Communication

We use a RESTful framework (Restlet) on the server in order to handle asynchronous requests and to encapsulate objects into JSON (JavaScript Object Notation)

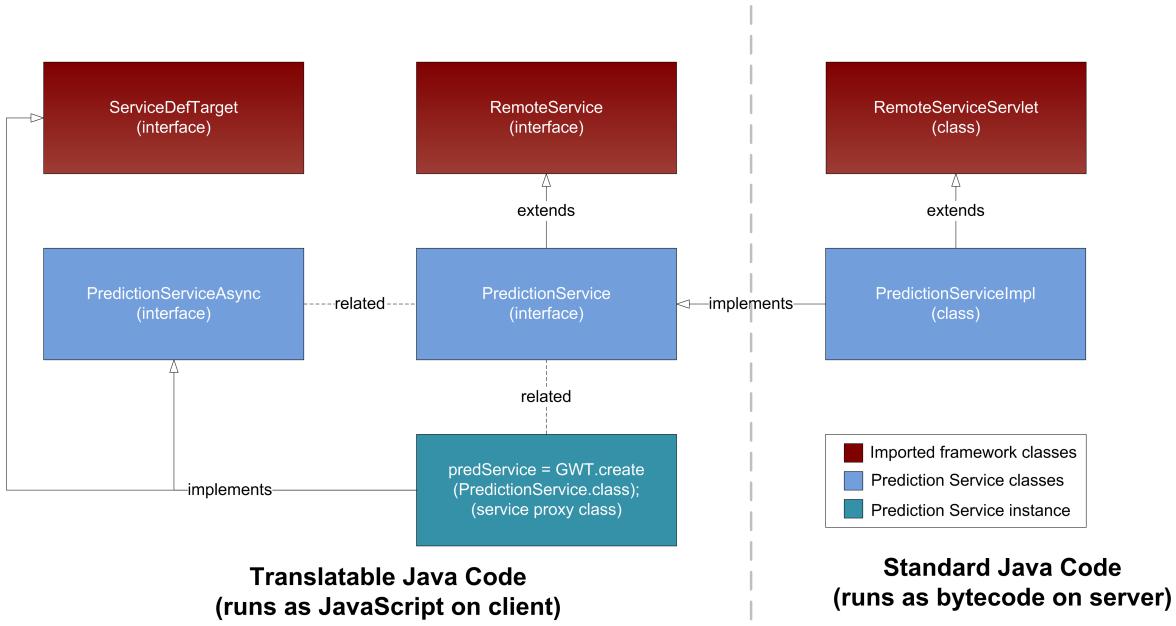


Figure 2.6: GWT RPC for Prediction Module

format. A JSON data interface is developed to allow maximum flexibility in building components that are using Instrumentation and Relevance Feedback results. By providing data in JavaScript Object Notation (JSON) format we allow other LarKC plug-ins to use our information in a simple and standardized way.

Visualization API

In order to get access to the server data, each HTTP request should respect the following format:

`http://SERVER_ADDRESS:8182/url?GET_PARAMETERS`

Where:

- SERVER_ADDRESS - is the address of the Visualization Service that handles the HTTP Request.
- GET_PARAMETERS - includes the GET parameters encoded in the URL address. Based on the parameters' type and value a corresponding JSON object is returned to the client. Query parameters are represented in ?Name=Value format.

We used different types of parameters based on the desired results. For example, if the Visualization Server resides on the localhost, then a query with the following URI might be composed:

`http://localhost:8182/test?dbtype=mysql&command=getworkflows`

Bellow we summarize the actual query parameters:

- dbtype: the type of database to be used.
- values:
 - `dbtype=mysql` - requesting use of MySQL database

- *dbtype=rrd* - requesting use of RRD database

Working with RRD Databases:

- rrd : the RRD table name.
 - values: *rrd=DBNAME*, where DBNAME could have one of the following values: *IOIn*, *IOOut*, *IRQPercent*, *IRQTime*, *IdlePercent*, *IdleTime*, *SwapIn*, *SwapOut*, *ysPercent*, *SysTime*, *SystemLoadAverage*, *SystemOpenFileDescriptorCount*, *otalSystemFreeMemory*, *TotalSystemUsedMemory*, *TotalSystemUsedSwapSpace*, *serPercent*, *UserTime*, *WaitPercent*, *WaitTime*
- start: Specifies the start time stamp of the first Metric Value to return.
 - Example: *start=1291807946100*
- end: Specifies the end time stamp of the last Metric Value to return.
 - Example: *end=1291807950000*
- resolution: the sampling resolution.
 - Example: *resolution=1*

The following example shows a HTTP request to get all metric values between a start and an end time stamp at a resolution of 1.

<http://localhost:8182/test?dbtype=rrd&rrd=TotalSystemUsedMemory&start=1291807946100&end=1291807950000&resolution=1>

The sample response is encapsulated in JSON format:

```
[
{"ts":1291807946100,"value":1251167.04},
 {"ts":1291807946400,"value":1251167.04},
 {"ts":1291807946700,"value":1251167.04}
 ...
]
```

Where:

- ts - is the metric timestamp
- value - represents the metric's value.

Working with MySQL Databases:

- command: a command specifying the sql query to be executed.
 - values
 - * *command=getworkflows* - list all workflows from the current platform
 - * *command= getplugins&idworkflow=ID* - list all plugins corresponding to a workflow with a given ID
 - * *command=queries&idworkflow=ID* - list all queries corresponding to a workflow with a given ID

- * *command=getquerymetrics&idquery=ID* - list all metrics corresponding to a query with a given ID
- tablename: the mysql database table name to be used.
 - Example: *tablename=plugins* - uses plugins table.

The following example shows a HTTP request to get all metric from MySQL database for a query with the ID=1:

<http://localhost:8182/test?dbtype=mysql&command=getquerymetrics&idquery=1>

The result JSON has the following form:

```
[{"MetricName": "QueryTimestamp",
 "MetricValue": "2011-01-18 14:50:37",
 "Timestamp": "2011-01-18 14:50:37.0"},

 {"MetricName": "QueryContent",
 "MetricValue": "PREFIX foaf: <http://xmlns.com/foaf/0.1/>    SELECT ?name1
 WHERE {?person1 foaf:knows ?person2 .
 ?person1 foaf:name ?name1 . ?",
 "Timestamp": "2011-01-18 14:50:37.0"},

 ...
 ]
```

Data Layer Component

For data persistence the visualization interacts with two kinds of databases:

- Relational DB: MySQL database - permits to create a consistent, logical representation of metrics and other relevant features. The MySQL queries and responses are coordinated by the server through the JDBC connector.
- RRD database: able to store time-series values. In order to handle this type of data we use the RRDTool component.

MySQL database

The relational data base schema we propose is described in Figure 2.7.

In the defined model, the “Platforms” table stores data about each instance of the platform that has been instrumented (a unique identifier is provided to each platform). Each platform has a list of workflows that have been run by it and the connection between the platform and the workflows is done via the table “Platform_Workflows”.

Each workflow can be run for a query (also uniquely identified) or more queries and the linking is made by “Queries_Workflows” table.

The relation between the list of plug-ins and a workflow for which they are run is done via the table “Workflows_Plugins”. The description of plugins is stored in table “Plugins” and the description of workflows is stored in table “Workflows”.

The information about metrics is collected in table “Metrics” that lists the names of all the metrics and “MTypes” that contains the types of metrics (like atomic, compound, query metric, workflow metric, plug-in metric, etc). The values of the metrics are stored in the tables: “Platforms_metrics”, “Workflows_Metrics”, “Plugins_Metrics”, “Queries_Metrics”.

RRD database

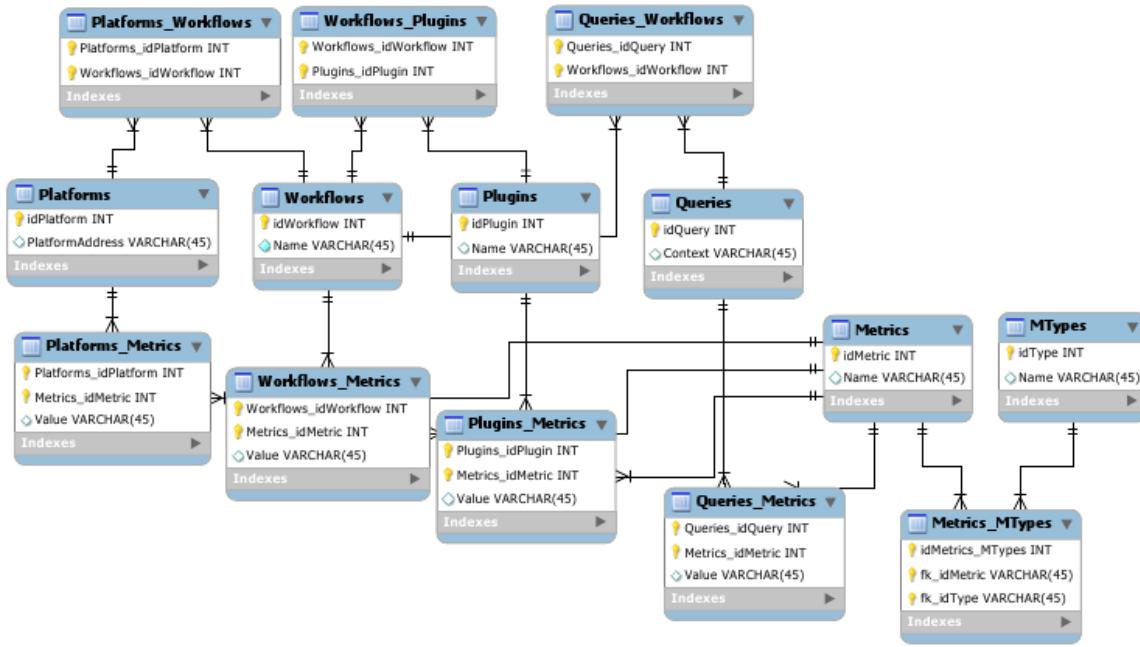


Figure 2.7: MySQL Relational Database Schema

An important aspect of monitoring is the ability to view real time data, such as the processor load, memory and other resource utilization etc. This type of data can best be represented as a time series. We cannot, however, continuously store such data, because the storage size will continually increase. Therefore, only relatively recent data should be stored. Old data should be aggregated. The de facto standard for high performance storage and access to such data is RRDTool¹.

A RRD database (which in the usual DBMS parlance would be best described as just a table) contains one or more data series, but all of them must be sampled at the same time points (i.e. there is a single time stamp column, and any number of value columns.) The number of the entries in the table is limited to a configurable value. Besides the values themselves, the database can also contain aggregated (archived) data. Archived data is obtained by applying an aggregation function (such as AVERAGE, MAX, MIN) on accumulated data. The number of aggregated entries is also fixed and configurable.

Because all the columns must share the same time stamp, and we chose to use multiple databases, each database containing a single data column, corresponding to one atomic metric. Currently, we have the following databases: *IdlePercent*, *IdleTime*, *IOIn*, *IOOut*, *IRQPercent*, *IRQTime*, *SwapIn*, *SwapOut*, *SysPercent*, *SystemLoadAverage*, *SystemOpenFileDescriptorCount*, *SysTime*, *TotalSystemFreeMemory*, *TotalSystemUsedMemory*, *TotalSystemUsedSwapSpace*, *UserPercent*, *UserTime*, *WaitPercent*, *WaitTime*. More databases can be created as needed.

RRDTool is a C and script based application. In order to use RRD from Java we needed a Java based implementation. Currently there are two such implementations and they are almost equivalent. The first one is RRD4J² distributed under the Apache

¹<http://www.mrtg.org/rrdtool>

²<https://rrd4j.dev.java.net/>

2.0 license and the second one is JRobin³ distributed under the LGPL license. Because they are similar we provided interfaces for both of them.

Our interface allows both writing data into the RRD and reading data from it. Writing data is straight-forward, and is called directly from the server code. Data reading (publishing) is done using a RESTlet component⁴. The data is transmitted using the JavaScript Object Notation (JSON⁵). The object is an array, containing time stamp – value pairs.

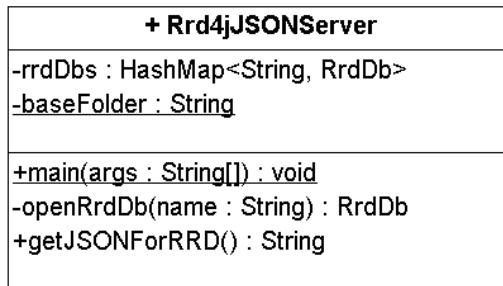


Figure 2.8: RRD access reading interface

The interaction between the main visualization modules and the users is presented in figure 2.9.

³http://www.jrobin.org/index.php/Main_Page

⁴<http://www.restlet.org/>

⁵<http://www.json.org/>

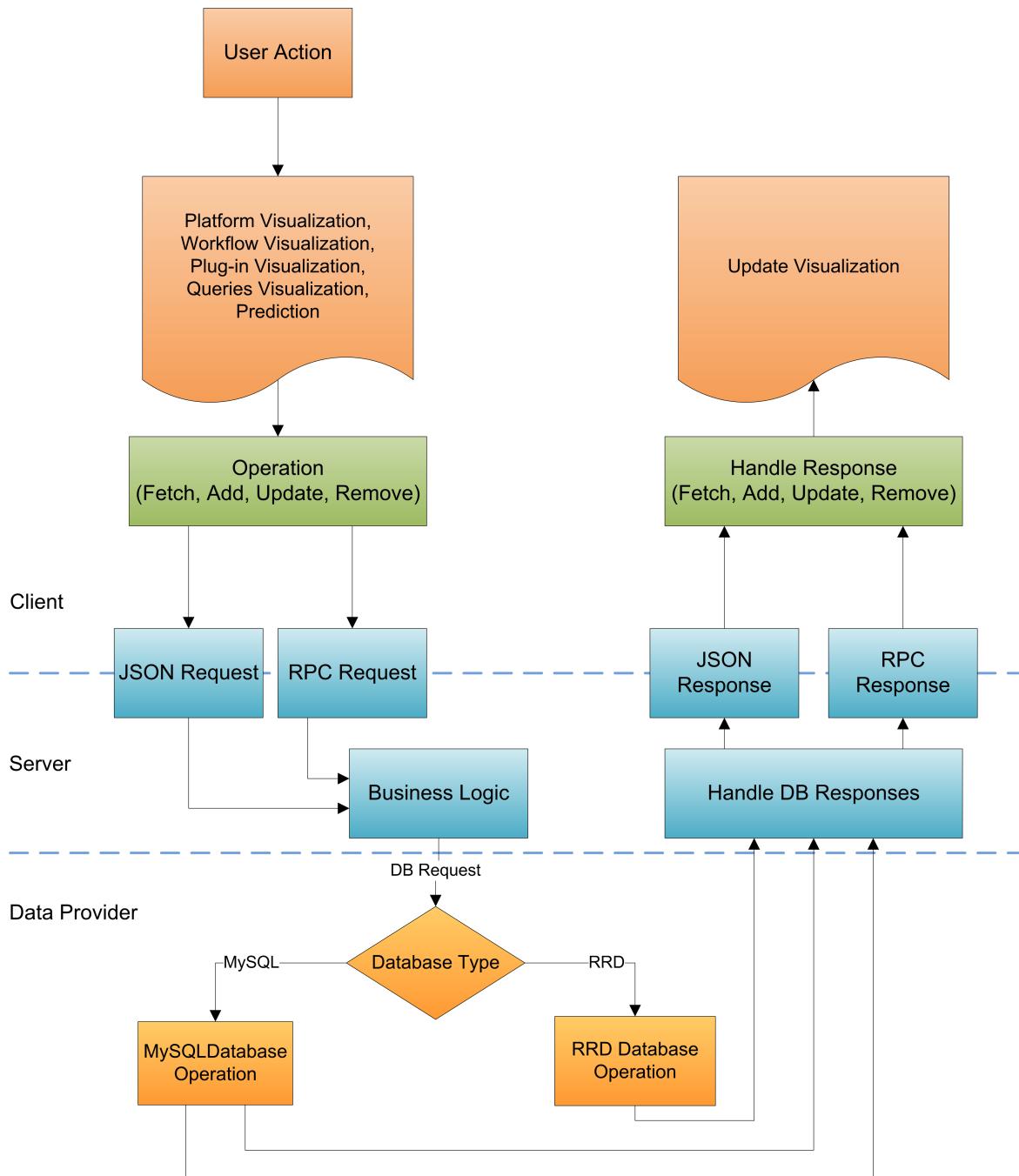


Figure 2.9: Visualization Flowchart

3. Relevance Feedback Theoretical Background

In order to define the mathematical model of the relevance feedback module we have considered several data mining algorithms. We will particularly refer to:

- Data preprocessing / analysis using principal component analysis and correlation based feature selection
- Machine learning using clustering and regression and multivariate data analysis using kernel canonical correlation analysis.

A mathematical model for the relevance feedback component is defined and presented in figure 3.1. The relevance feedback problem is formalized by considering a set of input metrics derived from SPARQL information and plugins and a set of output metrics referring to those resources that are measured and monitored. Relevance feedback module should take the values for the input metrics and find a relation between them and the output metrics.

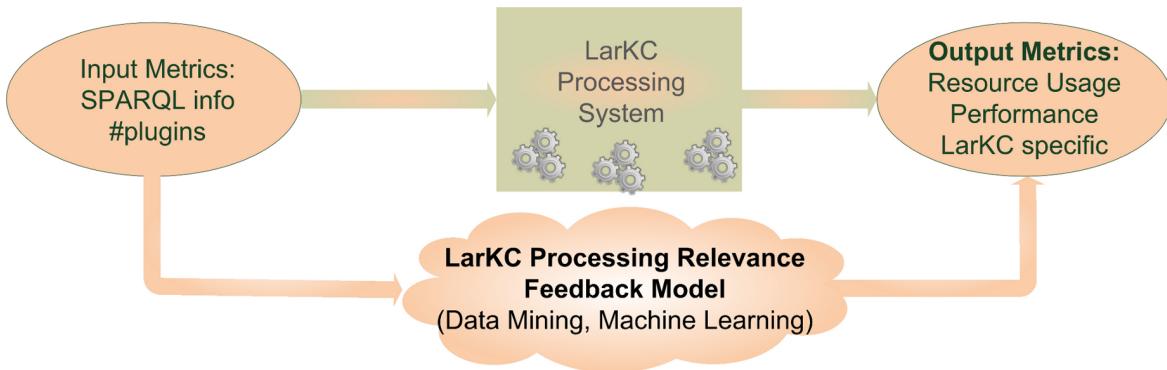


Figure 3.1: Relevance Feedback Model

Consider the following notations:

- The set of input metrics: $I = I_1, \dots, I_n$
- The set of output metrics: $O = O_1, \dots, O_m$
- The measurements/observations obtained from instrumentation and monitoring module are a matrix with p instances:

$$\begin{pmatrix} I_{11} & I_{12} & \dots & I_{1n} & O_{11} & O_{12} & \dots & O_{1m} \\ I_{21} & I_{22} & \dots & I_{2n} & O_{21} & O_{22} & \dots & O_{2m} \\ \dots & \dots \\ I_{p1} & I_{p2} & \dots & I_{pn} & O_{p1} & O_{p2} & \dots & O_{pm} \end{pmatrix}$$

Another notation:

$$\begin{pmatrix} Inst_{11} & Inst_{12} & \dots & Inst_{1u} \\ Inst_{21} & Inst_{22} & \dots & Inst_{2u} \\ \dots & \dots & \dots & \dots \\ Inst_{p1} & Inst_{p2} & \dots & Inst_{pu} \end{pmatrix}, u = m + n$$

Several operations can be applied to this mathematical input. First a data analysis or preprocessing can be carried out. Its role is to eliminate irrelevant features or to select the most relevant ones(CFS, PCA). Then, a function that defines the relation between input and output metrics can be defined (in our approach we use clustering plus regression or KCCA).

3.1 Principal component analysis

Principal component analysis (PCA) is one of the most valuable results from applied linear algebra. PCA is used abundantly in all forms of analysis in data mining because it is a simple, non-parametric method of extracting relevant information from confusing data sets. PCA is a way to identifying patterns in data, expressing the data in order to highlight the correlations such as similarities and dissimilarities. It is important because it's hard to visualize and make computations with the patterns of high dimensional data. The advantage is the data compression by reducing the number of dimensions without hopefully much losing of data information.

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of correlated variables into a set of values of uncorrelated variables called principal components. It is used for dimensionality reduction, more precisely it allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional space. The goal of PCA is to reduce the dimensionality of the data while retaining as much as possible of the variation present in the original dataset.

The first principal component has as high a variance as possible (accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to the preceding components. The best low-dimensional space can be determined by the "best" eigenvectors of the covariance matrix of variables (i.e., the eigenvectors corresponding to the largest eigenvalues also called "principal components").

We use PCA for selecting the most relevant input metrics from the set $I = I_1, \dots, I_n$ previously defined. The data that is analyzed for metrics dimensionality reduction refers to the measurements/observations from the matrix $Inst$. We keep those metrics that have the largest eigenvalues and we project all the data on the selected dimensions.

3.2 Correlation based feature selection

Feature selection, also known as variable selection, feature reduction, attribute selection or variable subset selection, is the technique of selecting a subset of relevant features for building robust learning models. Feature selection algorithms typically fall into two categories: feature ranking and subset selection. Feature ranking ranks the features by a metric and eliminates all features that do not achieve an adequate score. Subset selection searches the set of possible features for the optimal subset.

Feature subset selection is the process of identifying and removing as much of the irrelevant and redundant information as possible. Machine learning algorithms differ in the amount of emphasis they place on feature selection. At one extreme are algorithms such as the simple nearest neighbor learner which classifies novel examples by retrieving the nearest stored training example using all the available features in its distance computations. Towards the other extreme lie algorithms that explicitly try to focus on relevant features and ignore irrelevant ones.

The CFS algorithm is a heuristic for evaluating the worth or merit of a subset of features. This heuristic takes into account the usefulness of individual features for predicting the class label along with the level of inter-correlation among them. The hypothesis on which the heuristic is based can be stated: Good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.

The purpose of feature selection is to decide which of the initial features to include in the final subset and which to ignore. Various heuristic search strategies such as hill climbing and best are often applied to search the feature subset space in reasonable time. CFS starts from the empty set of features and uses a forward best first search with a stopping criterion of five consecutive fully expanded non-improving subsets.

In our approach, CFS is used for subset selection of input metrics from the set $I = I_1, \dots, I_n$. The further algorithms are applied only on the selected attributes for data instances. This makes the clustering process faster and the regression model more accurate by taking into account just the important metrics for the output metrics' prediction.

3.3 Clustering and regression

Generally, the relevance feedback model that we propose can be described as a function that takes a set of inputs and tries to define an output value:

$$O_j = f_j(I), j = \overline{1, m}$$

In the above formula, the function f_j can be:

- a) a clustering rule followed by a regression model which return a numerical value
- b) a clustering rule and a numbering (ranking) to:
 - b1) return a probability
 - b2) return a list of nominal values

The clustering process takes the set of instances and forms q groups (clusters):

$$\text{Cluster}(Inst, K) \Rightarrow C_1, \dots, C_q$$

where:

- q is the total number of clusters
- K is the set of clustering criteria:

$$K = \{K_i | K_i \in I, K_i \text{ is a clustering criterium}\} \Rightarrow K \subseteq I$$

The cluster number p on criteria K is:

$$C_p^K = \{Inst_i | Inst_i \in Inst, \forall Inst_j \in C_p^K, \text{similar}_K(Inst_i, Inst_j)\}$$

$$\bigcup_{j=1}^q C_j^K = Inst, \bigcap_{j=1}^q C_j^K = \emptyset$$

$$C^K = \{C_p^K | p = \overline{1, q}\}, \text{ where } q \text{ is the cardinality of } C^K$$

Given a new workload Q characterized by a set of input parameters (metrics):

$$I_W = (I_{W1}, I_{W2}, \dots, I_{Wn})$$

The objective is to find the cluster where the new workload belongs to:

$$C_b^K = C_j^K | similar_K(I_W, I_i), \forall I_i \in C_j^K, \text{ where } j = \overline{1, q}, q = |C^K|$$

The way that the prediction is applied in all previously mentioned cases is described in the next paragraphs.

Case a)

Find a regression function which returns a numeric value. It will be used in prediction of a numeric value of an output metric O_p :

$$O_p \in O, p \in \{1, 2, \dots, m\}$$

$$f_p : I \rightarrow \Re$$

$$O_p = f_p(I)$$

The function f_p is obtained by a regression method applied on the instances from the selected cluster C^K :

$$f_p(I) = \sum_{i=1}^n a_i I_i + a_0$$

The predicted value of O_p for the given input I_w is:

$$O_{pW} = f_p(I_W)$$

Case b1)

Find a probability function which returns a real value between 0 and 1. It is used in prediction of a nominal output metric O_p . The values of O_p are from the set $\{n_1, \dots, n_s\}$.

Define s probability functions f_{p1}, \dots, f_{ps} which output the probability values for $\{n_1, \dots, n_s\}$.

Inside the cluster C_b^K perform a selection on fixed nominal values $\{n_1, \dots, n_s\}$. The result is a set of selection clusters:

$$C_{b1}^{K1} = Select(C_b^K, K1), K1 = \{n_1\}$$

...

$$C_{bs}^{Ks} = Select(C_b^K, Ks), Ks = \{n_s\}$$

$$C_{bt}^{Kt} = \{Inst_i | Inst_i \in C_b^K, \forall Inst_j \in C_b^K, exact_{Kt}(Inst_i, Inst_j)\}, t = \overline{1, s}$$

The appearance probability of the nominal value n_t for input parameters I_W is n_{tW}

$$f_{pt}(I) = \frac{|C_{bt}^{Kt}|}{|C_b^K|}$$

$$n_{tW} = f_{pt}(Inst_W) = \frac{|C_{bt}^{Kt}|}{|C_b^K|}$$

Case b2)

This case is used for getting the best configuration (nominal attribute) that maximizes the values from a set of constraints applied on the input metrics. The constraints set $Z \subseteq I$ (cost functions, performance metrics, etc.) is:

$$Z = \{Z_1, \dots, Z_c\}$$

Consider the previous case b1) and the possible values for that nominal attribute n_1, \dots, n_s . The selection clusters are known: $C_{bt}^{Kt}, t = \overline{1, s}$.

Compute the mean performance for each input configuration for each cluster:

$$\mu_t = avg_Z(C_{bt}^{Kt}), t = \overline{1, s}$$

$$\mu_t = (\mu_{t1}, \dots, \mu_{tc})$$

$$Kt = \{n_t\}$$

Sort descending the selection clusters C_{b1}, \dots, C_{bs} by considering the mean performance $\mu_t, t = \overline{1, s}$ as sorting criteria \Rightarrow the sorted result will be the set of selection clusters S_{b1}, \dots, S_{bs} . The output consist in the ordered list of nominal values from S_{b1}, \dots, S_{bs} :

$$n(S_{b1}), \dots, n(S_{bs})$$

which represent the ordered list of nominal values from the best one to the worst one.

3.4 Kernel Canonical Correlation Analysis

A widely used method in multivariate statistics is Kernel Canonical Correlation Analysis (KCCA) mentioned by [3] and [4] as a variation of Canonical Correlation Analysis (CCA). If PCA identifies dimensions of maximum variance in a dataset and the data is projected onto these dimensions, CCA is a generalization of PCA. The canonical correlation analysis finds dimensions of maximum correlation considering both datasets, but its disadvantage is the fact that it is unable to identify which known input instances are qualitatively similar to an unknown instance [5].

Given two multivariate datasets, KCCA computes basis vectors for subspaces in which the projections of the two datasets are maximally correlated. This method has been used successfully by [5] for modeling system performance. KCCA defines similarity between known instances and unknown instances using a kernel function. The correlation analysis is done on pairwise distances not on the raw data itself.

In our approach we have followed the method described by [5] for modeling system performance. The feature vectors are defined by the input and output metrics. To each input observation (input metrics for a given instance) it corresponds an output observation (output metrics for a given instance). A similarity between any two input feature vectors and between any two output vectors is defined. KCCA uses kernel functions to compute distance metrics between all pairs of input vectors and pairs of output vectors.

Given p instances we form an input matrix K_x of dimension $p \times p$ whose (i, j) entry is the kernel evaluation $k_x(I_i, I_j)$. Another matrix K_y is also formed and its (i, j) entry is the kernel evaluation $k_y(O_i, O_j)$. Since $k_x(I_i, I_j)$ represents similarity between I_i and I_j and the same is valid for $k_y(O_i, O_j)$, the kernel matrices K_x and K_y are symmetric and their diagonals are equal to one.

There are some restrictions on the two matrices K_x and K_y . They should be symmetric and positive semidefinite. The specific kernel functions that are used depend on

the input and output features of the given system. One can experiment with Gaussian kernels for example.

The algorithms considers the two kernel matrices K_x and K_y formed on the input metrics and output metrics of each instance and it solves the generalized eigenvector problem [5]:

$$\begin{bmatrix} 0 & K_x K_y \\ K_y K_x & 0 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \lambda \begin{bmatrix} K_x K_x & 0 \\ 0 & K_y K_y \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}$$

This procedure finds subspaces in the linear space spanned by the eigenfunctions of the kernel functions such that projections onto these subspaces are maximally correlated. As stated by [5] KCCA produces a matrix A consisting of the basis vectors of a subspace onto which K_x may be projected, giving $K_x \times A$, and a matrix B consisting of basis vectors of a subspace onto which K_y may be projected, such that $K_x \times A$ and $K_y \times B$ are maximally correlated. We call $K_x \times A$ and $K_y \times B$ the input and output projections.

When a test instance comes, the kernel function k_x is applied to it and then its projection is found. We look for nearest neighbors in the projection input space and find the corresponding projections in the output projection space. The difficult problem is to map back from the output projection space into the original raw data space. A solution is to look for the corresponding raw values of the output projection neighbors and define the output values for the test instance based on them.

4. Relevance Feedback - Revised Scenarios

This chapter revises our original proposal and adjusts the scenarios proposed in [2]. For each scenario we define the input and output metrics on which the relevance feedback will operate.

- a) Provide a detailed list of metrics that will be used - atomic and compound
- b) Discussion about aggregated values - will they be used in prediction, how will they be used?

In [2] we have proposed three scenarios that deal with the relevance feedback module. These scenarios are:

- Scalability analysis
- Bottleneck prediction
- Work-flow prediction based on raw data

In what follows we will describe each of these scenarios and we will provide the exact list of metrics that are input and output for solving the prediction problem expressed by the scenario. Some of the original scenarios have been revised. We consider input the feature vector that comprises all the features needed by the relevance feedback module in order to provide a satisfactory result for a given scenario. We consider output the feature vector that will be provided as result by the relevance feedback module for a given scenario.

4.1 Scalability analysis

Description: given a query and a work-flow predict the resources and execution time for a successful accomplishment of the query such as:

- a) Given a query and a workflow predict resources per platform, this meaning:
 - The time needed until the platform will provide an answer to the given query, in the conditions of a workflow;
 - Total amount of memory that could be used for a successful execution of a given query;
 - Total number of interactions with the data layer;
 - The average CPU load during the execution of the query/workflow

Input metrics:	Intermediate metrics:	Output metrics:
-query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb -workflow metrics WorkflowPluginsNb parse(WorkflowPlugins) -> list		QueryTotalResponseTime avg(PlatformMemoryDim,queryExec) avg(PlatformCPULoad,queryExec) sum(PluginDataLayerAccessNo)

A preprocessing step of parsing the QueryNamespace, QueryDataSetSources and WorkflowPlugins is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics as clustering criteria. All the output metrics are the result of a regression method inside the previously found cluster. The values for PlatformMemoryDim and PlatformCPUload are computed by an average of all the instant values that are measured during the query execution. The PluginDataLayerAccessNo is obtained by adding all the values corresponding to each plugin.

- b) Given a query and a workflow (formed of plugins P1, P2, P3, P4) predict the resources used by each plugin:

Input metrics:	Intermediate metrics:	Output metrics:
<ul style="list-style-type: none"> -query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb -workflow metrics WorkflowPluginsNb parse(WorkflowPlugins) -> list 		<ul style="list-style-type: none"> PluginTotalExecutionTime -> for each plugin PluginThreadsStartedNb -> for each plugin PluginNodesNb -> for each plugin

A preprocessing step of parsing the QueryNamespace, QueryDataSetSources and WorkflowPlugins is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics as clustering criteria. All the output metrics are the result of a regression method inside the previously found cluster. The values for PluginTotalExecutionTime, PluginThreadsStartedNb and PluginNodesNb are output for each plugin (P1, P2, P3, P4) of the workflow.

- c) Given a query and a workflow (formed of plugins P1, P2, P3, P4) predict the outputs provided by each plug-in. By outputs we understand number of triples:

Input metrics:	Intermediate metrics:	Output metrics:
<ul style="list-style-type: none"> -query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb -workflow metrics WorkflowPluginsNb parse(WorkflowPlugins) -> list 		<ul style="list-style-type: none"> PluginOutputSizeInTriples -> for each plugin PluginOutputSizeInBytes -> for each plugin

A preprocessing step of parsing the QueryNamespace, QueryDataSetSources and WorkflowPlugins is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics as clustering criteria. All the output metrics are the result of a regression method inside the

previously found cluster. The values for PluginTotalExecutionTime, PluginOutputSizeInTriples and PluginOutputSizeInBytes are output for each plugin (P1, P2, P3, P4) of the workflow.

4.2 Bottleneck prediction

Description: Predict if the platform will provide a result to the query and the work-flow will have a successful execution (success/failure analysis) or find the probability that the work-flow chosen for my query to have a successful execution.

- a) Given a query find the probability that it will be run successfully considering only the parameters of the query. This scenario will find the first 3 best workflows and for each of the three workflows it will provide the probability of error.

Input metrics:	Intermediate metrics:	Output metrics:
<code>-query metrics</code> <code>QuerySizeInBytes</code> <code>QuerySizeInTriples</code> <code>QueryNamespacesNb</code> <code>parse(QueryNamespace) -> list</code> <code>QueryVariablesNb</code> <code>QueryDataSetSourcesNb</code> <code>parse(QueryDataSetSources) -> list</code> <code>QueryOperatorsNb</code> <code>QueryResultOrderingNb</code> <code>QueryResultLimitNb</code> <code>QueryResultOffsetNb</code>	<code>QueryCompletionStatus</code> <code>parse(WorkflowPlugins) -> list</code>	<code>Top 3 of: {</code> <code>WorkflowPluginsNb</code> <code>WorkflowPlugins -> list</code> <code>ProbOfError</code> <code>}</code>

A preprocessing step of parsing the QueryNamespace and QueryDataSetSources is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics and intermediate metrics as clustering criteria. Then, inside each cluster, the ProbOfError is computed by finding the ratio between how many queries have the completion status (QueryCompletionStatus) with value "failed" and the total number of queries.

- b) Given a query and a workflow find the probability of a successful execution considering the parameters of the query and of the workflow. In this scenario we will return a single probability of error - for the input workflow. This scenario is suitable for situations in which the input workflow is not in the top three best matches found in case a). When we say that, we consider the parameters of the workflow we may include the metrics for each plugin that is in the given workflow.

Input metrics:	Intermediate metrics:	Output metrics:
<code>-query metrics</code> <code>QuerySizeInBytes</code> <code>QuerySizeInTriples</code> <code>QueryNamespacesNb</code> <code>parse(QueryNamespace) -> list</code> <code>QueryVariablesNb</code> <code>QueryDataSetSourcesNb</code> <code>parse(QueryDataSetSources) -> list</code> <code>QueryOperatorsNb</code> <code>QueryResultOrderingNb</code> <code>QueryResultLimitNb</code> <code>QueryResultOffsetNb</code> <code>-workflow metrics</code> <code>WorkflowPluginsNb</code> <code>parse(WorkflowPlugins) -> list</code>	<code>QueryCompletionStatus</code>	<code>ProbOfError</code>

A preprocessing step of parsing the QueryNamespace, QueryDataSetSources and WorkflowPlugins is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics and the QueryCompletionStatus metric as clustering criteria. Inside the obtained cluster,

the ProbOfError is computed by finding the ratio between how many queries have the completion statuses (QueryCompletionStatus) with value "failed" and the total number of queries.

- c) Given a query find the probability of error and the best suited workflows considering the current status of the platform (i.e. other queries that are running and the resources used in the current moment). The current status of the platform is given by platform metrics, that will be input for the current scenario.

Input metrics:	Intermediate metrics:	Output metrics:
<ul style="list-style-type: none"> -query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb -platform metrics PlatformExecutionTime PlatformCPUload PlatformMemoryUsage PlatformMemoryDim PlatformGarbageCollectingTime 	QueryCompletionStatus	Top 3 of: { <ul style="list-style-type: none"> WorkflowPluginsNb WorkflowPlugins -> list ProbOfError }

A preprocessing step of parsing the QueryNamespace, QueryDataSetSources is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics and the QueryCompletionStatus metric as clustering criteria. Inside each obtained cluster, the ProbOfError is computed by finding the ratio between how many queries have the completion statuses (QueryCompletionStatus) with value "failed" and the total number of queries. The best top three situations (which have less ProbOfError) will be displayed.

- d) Given a query and a workflow find the probability of error considering the current status of the platform (i.e. other queries that are running and the resources used in the current moment).

Input metrics:	Intermediate metrics:	Output metrics:
<ul style="list-style-type: none"> -query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb -workflow metrics WorkflowPluginsNb parse(WorkflowPlugins) -> list -platform metrics PlatformExecutionTime PlatformCPUload PlatformMemoryUsage PlatformMemoryDim PlatformGarbageCollectingTime 	QueryCompletionStatus	ProbOfError

A preprocessing step of parsing the QueryNamespace and QueryDataSetSources is needed to obtain a list of individual components. Then a clustering method is

performed by considering all the processed input metrics and intermediate metrics as clustering criteria. Then, inside each cluster, the ProbOfError is computed by finding the ratio between how many queries have the completion statuses (QueryCompletionStatus) with value “failed” and the total number of queries.

4.3 Work-flow prediction based on raw data

Description: Given a query, predict possible work-flows to solve it.

- a) Given a query, return a list of best workflows that may solve it.

Input metrics:	Intermediate metrics:	Output metrics:
<ul style="list-style-type: none"> -query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb 		<p>Top 3 of: { WorkflowPluginsNb WorkflowPlugins -> list }</p>

A best workflow can be defined as:

- i. minimum amount of memory
- ii. minimum CPU usage
- iii. minimum amount of memory and CPU usage
- iv. minimum amount of memory and minimum execution time
- v. minimum number of threads and minimum execution time

For i) and ii) we will follow the steps:

- cluster by query similarity $\Rightarrow C_1, C_2, \dots, C_n$
- find the cluster to which the given query belongs, let it be C_q
- inside C_q cluster by workflow identity (same plug-ins in the workflow) $\Rightarrow C_{q1}, \dots, C_{qw}$
- for each $C_{qi}, i = \overline{1, w}$ find the average memory or CPU usage
- return the first 3 workflows having minimum average memory or average CPU usage

For iii)-v) we will follow the steps:

- cluster by query similarity $\Rightarrow C_1, C_2, \dots, C_n$
- find the cluster to which the given query belongs, let it be C_q
- inside C_q cluster by workflow identity (same plug-ins in the workflow) $\Rightarrow C_{q1}, \dots, C_{qw}$
- define a cost function $f_c(p_1, p_2)$ where p_1, p_2 can be:
 - case iii) p_1 = amount of memory, p_2 = CPU usage
 - case iv) p_1 = memory, p_2 = execution time
 - case v) p_1 = threads, p_2 = execution time

- a weight can be associated to each constraint; let a be the weight defined for p_1 and b the weight defined for p_2
- the weight a represents the degree of importance of constraint p_1 ; for defining this weight we will consider the range of values of p_1 and the range of values of p_2 ; the weight a will be proportional with the ratio of these ranges
- the cost functions can be:
 - $|a * p_1 + b * p_2|$
 - $\sqrt{a * p_1 + a * p_1 + b * p_2 + b * p_2}$
 - other norms
- select the workflow that offers maximum performance that is minimum amount of resources, and minimum execution time

b) Given a query and a plugin return the mostly used workflow

Input metrics:	Intermediate metrics:	Output metrics:
-query metrics QuerySizeInBytes QuerySizeInTriples QueryNamespacesNb parse(QueryNamespace) -> list QueryVariablesNb QueryDataSetSourcesNb parse(QueryDataSetSources) -> list QueryOperatorsNb QueryResultOrderingNb QueryResultLimitNb QueryResultOffsetNb	Plug-in name Plug-in type (decider, selector, reasoner...)	WorkflowPluginsNb WorkflowPlugins -> list

A preprocessing step of parsing the QueryNamespace and QueryDataSetSources is needed to obtain a list of individual components. Then a clustering method is performed by considering all the processed input metrics and intermediate metrics (Plug-in name and Plug-in type) as clustering criteria. Then the size of each cluster (the number of contained instances) is computed. The result is the workflow that has the highest number of instances considering all determined clusters.

5. Relevance Feedback Architecture

The relevance feedback component comprises two main modules:

- training module - performs data mining on top of instrumented data; Its output is a prediction model.
- application module - applies the prediction model to new data. It interacts with the visualization component from which it receives test data.

5.1 RF training

The whole architecture of the RF training module follows the model-view-controller pattern. The prototype of the module contains several layers:

- graphical user interface
- controller components
- processing components

Figure 5.1 depicts the simplified architecture containing three layers and the interconnection between them.

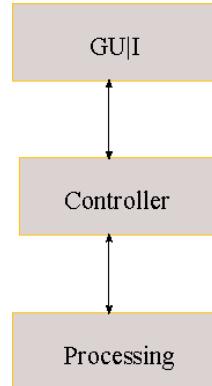


Figure 5.1: Tiers of the RF architecture

The entities of the graphical user interface have the view role, that is they realize the interaction between the user and the RF training application. The controller entities receive information from GUI, they process it and they send the corresponding commands to the processing entities. Processing comprises entities that connect to the data base and retrieve information from it, entities that perform the data mining methods, entities that capture the results of the training process.

The training module has as main functionality the application of data mining methods on the instrumented data and its output is a function able to model the relation between some input metrics and output metrics.

The operations supported by the training module are:

- data loading: allows us to choose the data source that contains the training instances. It is designed to open CSV files and data base configuration files.

- data analysis: offers two algorithms: principal component analysis and correlation based feature selection. The instances that have been loaded can be analyzed by these two algorithms and some of the irrelevant features or instances can be removed.
- training of the RF function based on interest scenario: provides clustering and regression prediction method and KCCA method. They are applied on the whole training set or on the data that results after applying PCA or CFS.

A general architecture is given in Figure 5.2.

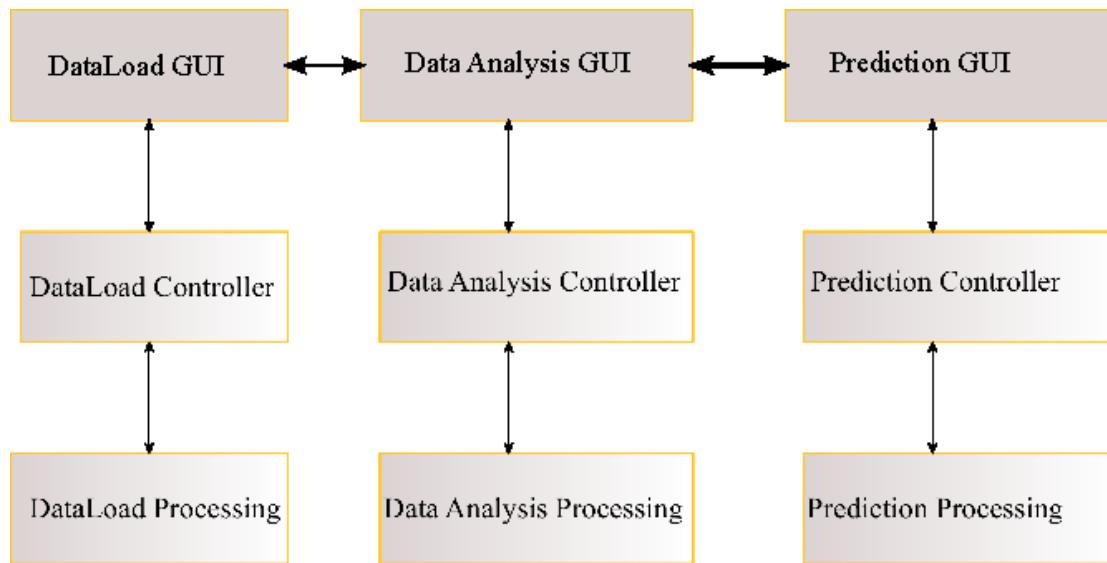


Figure 5.2: Detailed architecture of RF

5.2 RF application

The application module consists in applying the models trained by RF training to some test instances. Currently, the results of the clustering and regression have been integrated in the visualization module. The resulting architecture is explained in chapter 2 in section 2.2.

6. End-user guide for Visualization

The visualization module is a web-based, client server application. It can be installed and used on any computer. Our running demo on some initial data can be found at: <http://larkc.utcluj.ro/visual/>.

The software programs that need to be installed in order to start using the visualization module are:

1. Apache Tomcat version 7.0.6 or later, or other compatible application server¹.
2. MySQL Server version 5.0.24² or later.

The source code for visualization can be downloaded from: <https://github.com/semantic-im/sim-vis>. It contains the following:

- MySQLDb
- RestServer
- Visualization_Metrics
- Visualization_Prediction

The steps that need to be done are:

1. Install mysql db server
 - <http://dev.mysql.com/doc/refman/5.1/en/installing.html>
 - Create an empty data base with the name “larkc” (default user is “root” and password is “1111”)
 - Import sql file: larkc-core-2011-01-28.sql into your database
2. For RestServer:
 - (a) Run the rest server with the input arguments:
`java -cp ..\lib\jrobin-1.5.9.1.jar;..\lib\org.restlet.jar;..\lib\rrd4j-2.0.5.jar \Rrd4jJSONServer "path\to\the\RestServer\!rrds"`
 - (b) Add the following dependencies to the project: the .jar files in RestServer \lib.
3. Visualization_Metrics and Visualization_Prediction
 - (a) For Visualization_metrics it is recommended to run Eclipse with GWT plugin installed.
 - (b) You should add the external libraries the following dependencies: lib\ofcgwt.jar and lib\smartgwt.jar.
 - (c) For now visualization metrics and the prediction page are two separate projects.
4. Running all the projects:

¹<http://tomcat.apache.org/>

²<http://dev.mysql.com/>

- (a) In order to run the project the mysql server should be running.
- (b) Start RestServer
- (c) Run Visualization components (Metrics and Prediction) - recommended to use Eclipse to run them because it is easier to debug them.

7. End-user guide for Relevance Feedback

7.1 RF training

The training module can be downloaded from: [github/sim/sim-rf/training](https://github.com/LARKC/sim/tree/sim-rf/training). After downloading the demo, start relFedLarkc.exe application. It will open a dialog window that looks as in Figure 7.1. The dialog based window has several tabs:

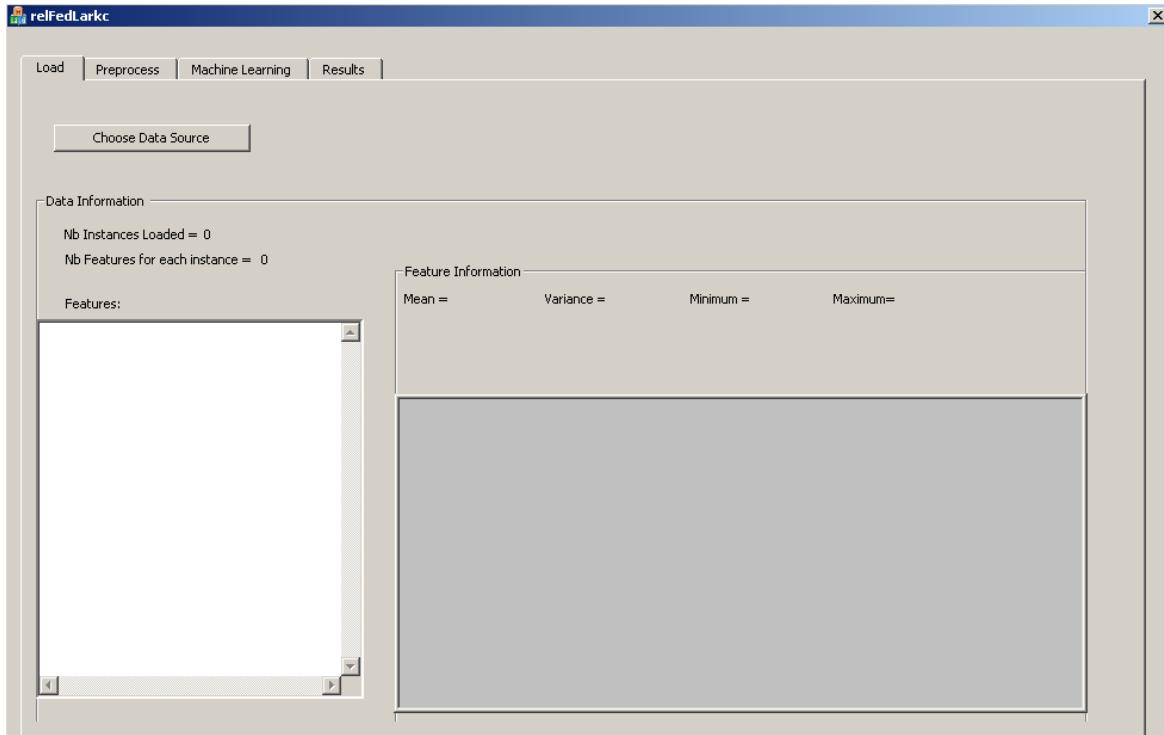


Figure 7.1: Main window of the training module

- Load
- Pre-process
- Machine learning
- Results

The first step you need to do is to have some data for training. The application demo comes with a data file ('data.csv'). Select the 'Load' tab and press the 'Choose data source' button. It will open a file chooser dialog. You should select a .csv file. The structure of the .csv file should be the following:

- on the first line the names of the features separated by comma.
- on the second line the types of the features separated by comma (NUMERIC, DATE, CATEGORICAL, STRING).
- on the next lines the values of the features for an instance (feature values are separated by comma).

After loading the .csv file a short summary on features and their values is displayed. That is we display the list of features encountered in the data file and their type. For each feature, if it is numeric, you can obtain its mean and standard deviation. Figure 7.2 depicts a screen shot.

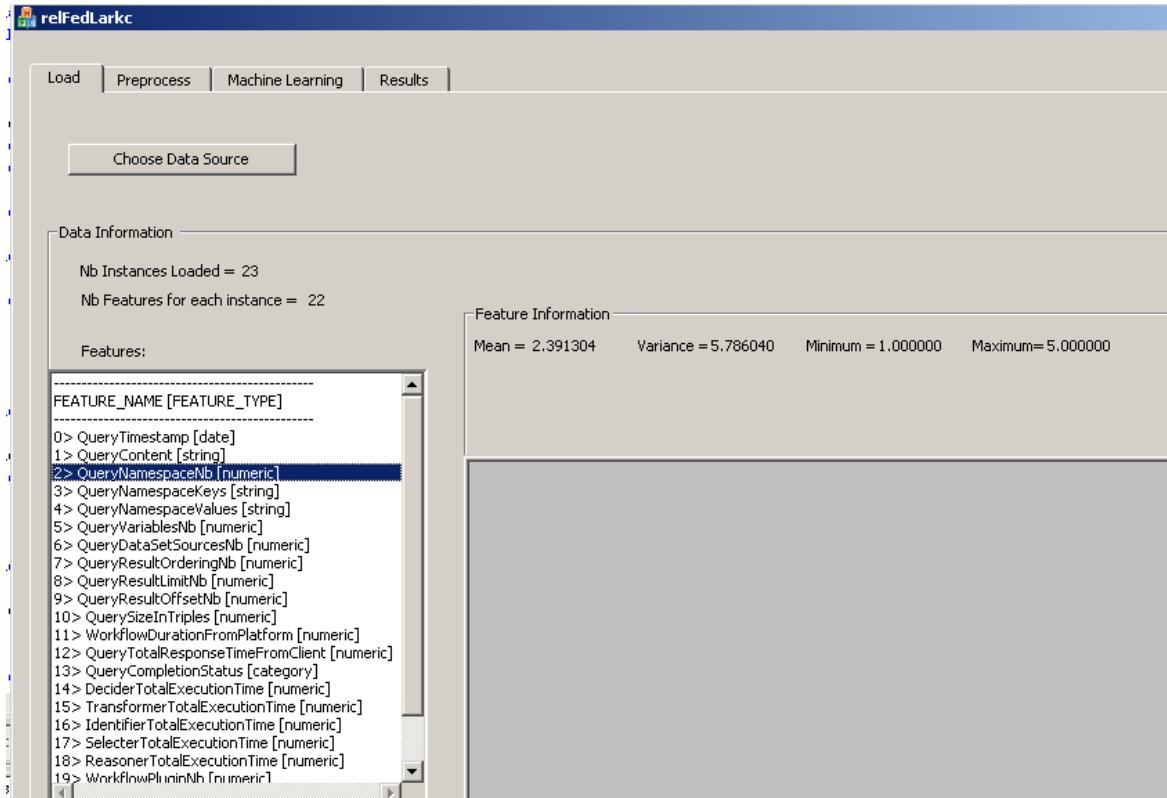


Figure 7.2: Summary of sample training instances that were loaded by RF training module

The second tab-dialog contains two radio buttons: PCA and CFS. In the demo none of these is enabled because the number of training instances and features is too small to allow such an analysis. For our next release we will have more training data (we hope that we can capture data from two LarKC use-cases) and these algorithms will have more relevance than.

The third tab contains the 'Machine Learning' algorithms we have implemented. We have enabled the 'Clustering and regression' algorithm, as it has best results on the sample data set. The first step you need to do in order to train the algorithm is to choose the feature indexes. You should choose features that have a numeric type. For example you could put in the *training features* edit box the following indexes: "2,5,6,11,12". In the *Predicted feature* you should input the index of the feature for which you want the prediction. Be careful to choose a 'numeric' feature. Then check the *Clustering and Regression* box. Next press the 'Load test data and predict' button and choose the file on which you want to make the prediction. We have tested the prediction on the training set and on some instances that we have removed from the training set before the actual training was done. The results are pretty good, but we strike for obtaining more relevant results after the data set is enlarged. Figure 7.3 marks the steps needed for obtaining prediction results (follow the order of steps marked in red). Currently the prediction results are depicted in the same tab dialog but we plan to move them in the 'Results' tab for our future release. The model has

relFedLarkc

Load | Preprocess | Machine Learning | Results |

Machine Learning / Data Mining

1 Training features: Predicted feature: 2

3 Clustering and Regression
 KCCA

4 IdentifierTotalExecutionTime

5

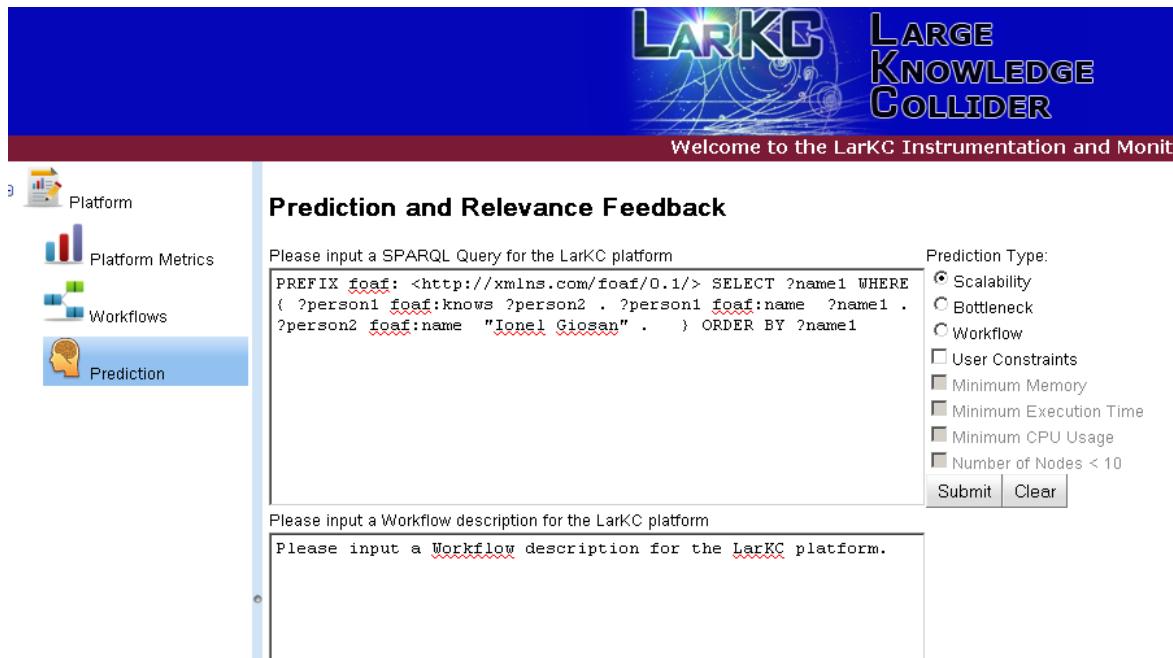
```
**Instance no 0** Measured: 11468.00 | Predicted: 11457.14
**Instance no 1** Measured: -1.00 | Predicted: -1.00
**Instance no 2** Measured: 21156.00 | Predicted: 21134.87
**Instance no 3** Measured: 6688.00 | Predicted: 6681.32
**Instance no 4** Measured: -1.00 | Predicted: -1.00
**Instance no 5** Measured: -1.00 | Predicted: -1.00
**Instance no 6** Measured: -1.00 | Predicted: -1.00
**Instance no 7** Measured: -1.00 | Predicted: -1.00
**Instance no 8** Measured: 19047.00 | Predicted: 19027.97
**Instance no 9** Measured: 3704.00 | Predicted: 3700.30
**Instance no 10** Measured: -1.00 | Predicted: -1.00
**Instance no 11** Measured: 24110.00 | Predicted: 24085.91
**Instance no 12** Measured: -1.00 | Predicted: -1.00
**Instance no 13** Measured: 9250.00 | Predicted: 9240.76
**Instance no 14** Measured: -1.00 | Predicted: -1.00
**Instance no 15** Measured: -1.00 | Predicted: -1.00
**Instance no 16** Measured: -1.00 | Predicted: -1.00
**Instance no 17** Measured: 12703.00 | Predicted: 12690.84
**Instance no 18** Measured: -1.00 | Predicted: -1.00
**Instance no 19** Measured: -1.00 | Predicted: -1.00
**Instance no 20** Measured: 29438.00 | Predicted: 29408.59
**Instance no 21** Measured: -1.00 | Predicted: -1.00
```

Figure 7.3: RF training - the steps needed for performing the training and prediction

been integrated in the visualization module and there it can be tested on completely new test instances.

7.2 RF application

The results provided by the RF training module are integrated in the visualization component. An operational demo can be seen at: <http://larkc.utcluj.ro/visual/>. Figure 7.4 offers a screen shot of the on-line demo. The models have been trained on a small amount of data (about 25 queries, one workflow).



The screenshot shows the LarkC visualization component's user interface. At the top, the LarkC logo and the text "Welcome to the LarKC Instrumentation and Monit" are visible. On the left, a sidebar menu includes "Platform", "Platform Metrics", "Workflows", and "Prediction", with "Prediction" being the active tab. The main area is titled "Prediction and Relevance Feedback". It contains two input fields: "Please input a SPARQL Query for the LarkC platform" containing the query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT ?name1 WHERE
{ ?person1 foaf:knows ?person2 . ?person1 foaf:name ?name1 .
?person2 foaf:name "Ionel Giosan" . } ORDER BY ?name1
```

and "Please input a Workflow description for the LarkC platform" containing the text "Please input a Workflow description for the LarkC platform.". To the right of these fields is a "Prediction Type" section with several radio buttons and checkboxes:

- Scalability
- Bottleneck
- Workflow
- User Constraints
- Minimum Memory
- Minimum Execution Time
- Minimum CPU Usage
- Number of Nodes < 10

At the bottom right of the prediction type section are "Submit" and "Clear" buttons.

Figure 7.4: Relevance feedback integrated in the visualization component

8. Conclusion

In this deliverable we reported about the advance of two components of Semantic Instrumentation and Monitoring (SIM), the instrumentation and monitoring solution in LarKC, namely visualization and relevance feedback. The deliverable contains updates of the architecture of the two components presenting final design decision and status of the technical implementation. The deliverable contains also a revision and refinement of the scenarios that are supported by the relevance feedback component. The set of metrics that are relevant for this task was refined. This deliverable defines also a mathematical model for the relevance feedback. The implementation of this model is available as part of the relevance feedback component. To support the interested end users in using the instrumentation and monitoring tools, this deliverable contains basic guides on how to use the visualization and relevance feedback tools. As the this deliverable defines the final architecture and design of the visuzalitaion and relevance feedback, further updates of these componets will include the advance of their implementation to cover the promised functionalities. Next steps include also the testing and validation of instrumentation and monitoring tools, including visuzalitaion and relevance feedback, an real monitoring data coming from the instrumented version of LarKC platform, workflows and plugins.

REFERENCES

- [1] I. Toma, R. Brehar, S. Nedevschi, M. Chezan, S. Bota, I. Giosan, M. Negru, and A. Vatavu, "Instrumentation and monitoring platform - design, architecture specification and first prototype," LarKC Project Deliverable, Tech. Rep. D11.2, 2011.
- [2] R. Brehar, I. Toma, S. Nedevschi, M. Negru, S. Bota, I. Giosan, A. Vatavu, C. Viccas, and M. Chezan, "State of the art and requirements analysis," LarKC Project Deliverable, Tech. Rep. D11.1.1, 2010.
- [3] S. Akaho, "A kernel method for canonical correlation analysis," in *In Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*. Springer-Verlag, 2001.
- [4] F. R. Bach, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2002.
- [5] A. S. Ganapathi, "Predicting and optimizing system utilization and performance via statistical machine learning," Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2009. [Online]. Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-181.html>