Demonstration and report of data mining and discovery tools and services

Project Number: FP6-2005-IST-026996

Deliverable id: D6.2

Deliverable name: Demonstration and report of data mining and discovery tools and services

Date: 15. December 2007

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 6.2
Document name:	Demonstration and report of data mining and knowledge discovery tools and services
Document type (PU, INT, RE)	INT
Version:	1.0
Date:	15.12.2007
Authors: Organisation: Address:	WP6 Partners

Document type PU = public, INT = internal, RE = restricted

ABSTRACT:

The present document provides an overview of the ACGT tools and services for data mining and knowledge discovery in clinico-genomic data

KEYWORD LIST: Knowledge and data management, discovery tools, data mining.

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	20.11.2007	Draft	Michael Mock
0.4	30.11.2006	Draft	Michael Mock (Ed.)
1.0	15.12.2007	Submitted	Michael Mock (Ed.)

List of Contributors

- Johan Karlson, UMA
- Michael Mock, FhG
- Andréas Persidis, Biovista
- George Potamias, FORTH
- Stefan Rüping, FhG
- Basavenneppa Tallur, INRIA
- Dennis Wegener, FhG

Contents

Executive Summary

The goal of this deliverable is to describe the use of data mining in discovery tools in the ACGT platform. In particular, the description given in this document refers to the ACGT demonstrator setup shown in December 2007. The overall setup and the workflow executed in the demonstration are described in detail. The workflow is built around the well-known clinical research project described by Farmer et. al, for which experimental data is freely available. In the ACGT demonstrator, this workflow will be used to show various interactions of the components of the ACGT platform. In particular, the interaction between knowledge discovery tools, the workflow enactor, the mediator, and the data management and grid management system will be demonstrated. Furthermore, the individual implementations of the knowledge discovery services are described, i.e, the GridR service and client, the Literature Mining service, and the clustering services.

1 Introduction

1.1. Introduction

This deliverable D6.2 presents the state of development of the knowledge discovery tools and their integration into the ACGT platform. The GridR service for statistics in R is described in section 2.1, the GridR client, a tool for accessing the ACGT platform from R, is described in section 2.2. The sections 2.3 and 2.4 deal with the literature mining tools and the clustering tools, respectively. Finally, section 3 presents an outlook discussing the further developments to be undertaken in WP6.

1.2. Knowledge Discovery in the ACGT Demonstrator

This section gives an overview of the use case that builds the exemplary background of the evaluation scenario underlying the deliverable D6.2 and its related deliverables and outlines the practical relevance of the demonstrator (see also D13.1).

Zur Anzeige wird der QuickTime™
Dekompressor „TIFF (Unkomprimiert)“
benötigt.

- Figure 1: Overview of WP6 knowledge discover process in the ACGT Demonstrator

Figure 1depicts the general overview of the KD tools in the ACGT demonstrator and the flow of input data and output data. We refer to the well known scenario described in the article by Farmer [Farmer et al. 2005], a simple clinical research project available from the literature and for which all data were available online [GEO]. This also provides a validation of the KD tools in a realistic usage. In the “Farmer scenario”, microarray data (Affymetrix U133A gene expression microarrays) obtained from breast cancer tumor samples of 49 patients are used to associate subtypes of breast cancer to patterns of gene expression and molecular signatures. R and BioConductor packages are used to load, normalize and analyze the data. In a first step, the ACGT demonstrator will be validated by showing that the results of the original paper can be reproduced using GridR. Secondly, the names of the genes detected as output in the R analysis will serve as input for literature mining tool in the ACGT demonstrator’s knowledge discovery process.

1.3. Overview of interactions between KD components

This section gives a detailed overview on interactions between the knowledge discovery components in the ACGT demonstrator. Figure 2 presents the WP6 tools organization and their interactions.

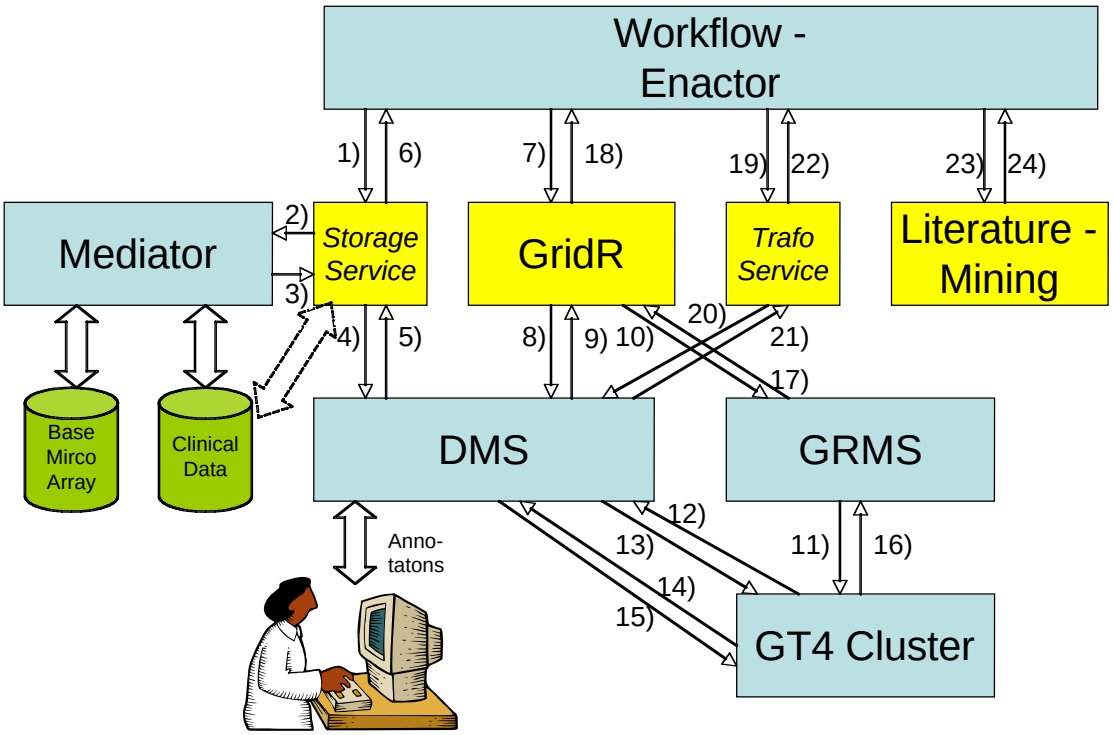

Figure 2: Overview of WP6 tools organization (yellow boxes), in the ACGT demonstrator.

In Figure 2, the yellow boxes denote the knowledge discovery tools used in the ACGT demonstrator whereas the ACGT components interacting with the knowledge discovery tools are depicted as blue boxes. The green items denote the database systems providing the mircoarray data, the clinical data and the annotations which are used in the demonstration scenario. Each black arrow represents a message sent from one component to another.

The workflow enactor builds the central user interface of the ACGT architecture, from which all executions are initiated and controlled. It interacts the rest of the system solely by invoking the knowledge discovery tools. All interactions are implemented in a standardized service oriented software architecture (SOA) relying on approved software engineering standards. In order to get the resources necessary for executing the knowledge discovery task, the knowledge discovery tools invoke other components of ACGT architecture, namely the mediator to retrieve input data from the database systems, the data management system (DMS) for making the input data available to the computations, and the grid management system (GRMS) for getting the computational resources (i.e. computing clusters) for execution the computations. Note that the GRMS may but does not necessarily provide these computational resources itself, instead, it can access computing clusters which are located elsewhere via the Globus Toolkit (GT4).

The yellow boxes in Figure 2 mainly refer to the GridR service as execution support for R statistics scripts in the ACGT architecture and the Literature Mining service for detecting results from research and practice which relevant for the actual knowledge discovery process. The *Storage Service* and *Trafo Service* are do not perform actual knowledge discovery algorithms, but serve for retrieving and providing the correctly formatted input for the GridR service and Literature Mining service, respectively. More specifically, the *Storage Service* gets the necessary microrray data, clinical data and annotations data needed for the R scripts that compute the Farmer scenario, and the *Trafo Service* reads the results of the analysis from data management system and provides them as parameters to the Literature Mining service.

Finally, the interaction steps between the individual services in the ACGT demonstrator are depicted as black arrows in Figure 2. The following Table 1 gives and overview on their meaning.

Step	Sender	Receiver	Data Type	Meaning
1)	WF Enactor	Storage Service	Text / Qeury	Specification of search
2)	Storage Service	Mediator	Text / Query	Specification of search
3)	Mediator	Storage Service	Data (mircoarray, clinical, annotations)	Data input for the R scripts
4)	Storage Service	DMS	Data (mircoarray, clinical, annotations)	Store input data for the R script in the DMS
5)	DMS	Storage Service	DMS IDs	IDs for the input data for later identification
6)	Storage Service	WF Enactor	DMS IDs	IDs for the input data for the R script
7a)	WF Enactor	GridR	Text / R script	The R script to be executed
7b)	WF Enactor	GridR	DMS IDs	IDs for the input data for the R script
8)	GridR	DMS	Text / R script	Store the R script in the DMS for later execution in the GRMS

9)	DMS	GridR	DMS IDs	The ID for the R script
10	GridR	GRMS	DMS IDs	The IDs for the R script and for the input data
11)	GRMS	GT4	DMS IDs	The IDs for the R script and for the input data
12)	GT4	DMS	DMS IDs	The IDs for the R script and for the input data
13a)	DMS	GT4	Text / R script	The R script to be executed
13b)	DMS	GT4	Data (mircoarray, clinical, annotations	Data input for the R script
14)	GT4	DMS	Text / Result	Result of the execution of the R script
15)	DMS	GT4	DMS ID	ID of the result of the execution of the R script
16)	GT4	GRMS	DMS ID	ID of the result of the execution of the R script
17)	GRMS	GridR	DMS ID	ID of the result of the execution of the R script
18)	GridR	WF Enactor	DMS ID	ID of the result of the execution of the R script
19)	WF Enactor	Trafo Service	DMS ID	ID of the result of the execution of the R script
20)	Trafo Service	DMS	DMS ID	ID of the result of the execution of the R script
21)	DMS	Trafo Service	Text / Result	Result of the execution of the R script
22)	Trafo Service	WF Enactor	Text / Result	Result of the execution of the R script
23)	WF Enactor	Litratur Minig	Text / Parameters	The result of the execution of the R script serves as input parameters for the

				Literatur Mining service
24)	Literature Mining	WF Enactor	Text / Result	The result of the literature search performed in text format

Table 1: Execution stepts of the knowledge discovery in the ACGT demonstrator.

Most of the steps listed in Table 1 are self-explanatory. For improving the readability of the table, we put steps 7a), 7b) and 13a), 13b) into two lines, respectively. In the actual execution of the workflow, the a) and b) steps are combined in a single message. We did not go into detail when describing the access of the mediator to the different database systems. This is for avoiding unnecessarily confusing details, but also partly due to the fact that only the interaction between the mediator and the clinical database system is currently implemented in the ACGT demonstrator. In the current implementation, the access to the microarray data is directly performed by the storage service. The user annotations are uploaded by the user into the DMS.

1.4. References

[Farmer et. al. 2005] P. Farmer, H. Bonnefoi, V. Becette, M. Tubiana-Hulin, P. Fumoleau, D. Larsimont, G. Macgrogan, J. Bergh, D. Cameron, D. Goldstein, S. Duss, AL. Nicoulaz, C. Brisken, M. Fiche, M., R. Iggo, “Identification of molecular apocrine breast tumours by microarry analysis.” Oncogene, 24, 2005, 4660-4671

[GEO] Gene Expression Omnibus (GEO), accession number GSE1561, http://www.ncbi.nlm.nih.gov

2 KD Tools and Services in the ACGT platform

2.1. Introduction

Figure 3 gives on overview on the ACGT knowledge discovery tools developed in WP6. All off them can be accessed and invoked from the workflow enactor, yielding a comprehensive and suite of possibilities for combinations and interactions between the individual knowledge discovery tools. The tools denoted in the upper line in Figure 3 will be described in this section, whereas the Metadata repository, which will be part of the next deliverable D6.3, is briefly sketch in the outlook in section 3.

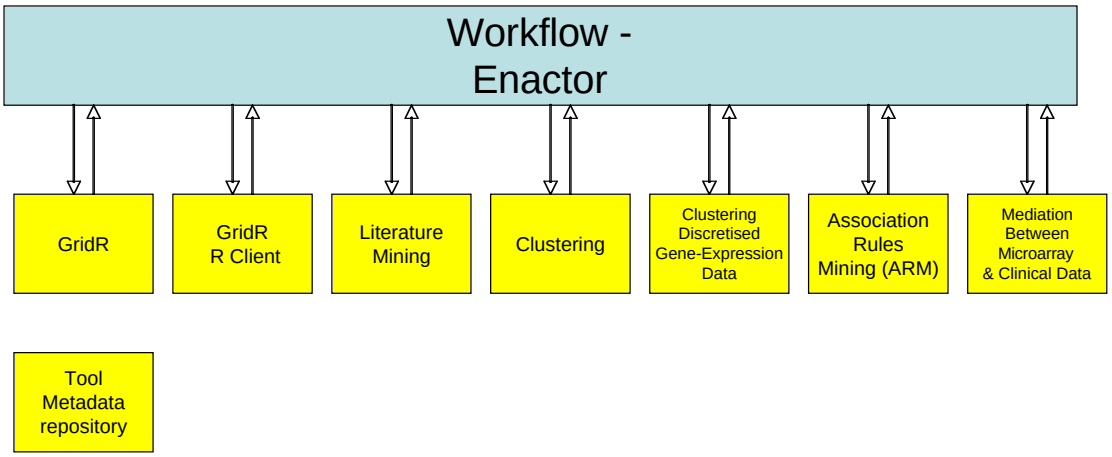

Figure 3: Overview of WP6 tools described in this document.

Not yet all of the knowledge discovery tools from Figure 3 are integrated in the ACGT demonstrator example workflow. Here, we focussed on showing that the GridR service can in fact be used as portal for accessing the ACGT grid infrastructure and can be used as centre of interaction with other knowledge discovery services, namely the literature mining service is used as example.

2.2. GridR – Service for statistics in R

2.2.1 Overview

As depicted in Figure 1, the ACGT demonstrator so far implements and workflow that integrates knowledge discovery tools GridR and Literature Mining service. Figure 4 and Figure 5 provide a user-oriented description of the knowledge discovery secenarios executed by the workflow in the ACGT demonstrator. Both scenarios use the GridR service as entry point into the grid infrastructure and can be executed either sequentially or in parallel.

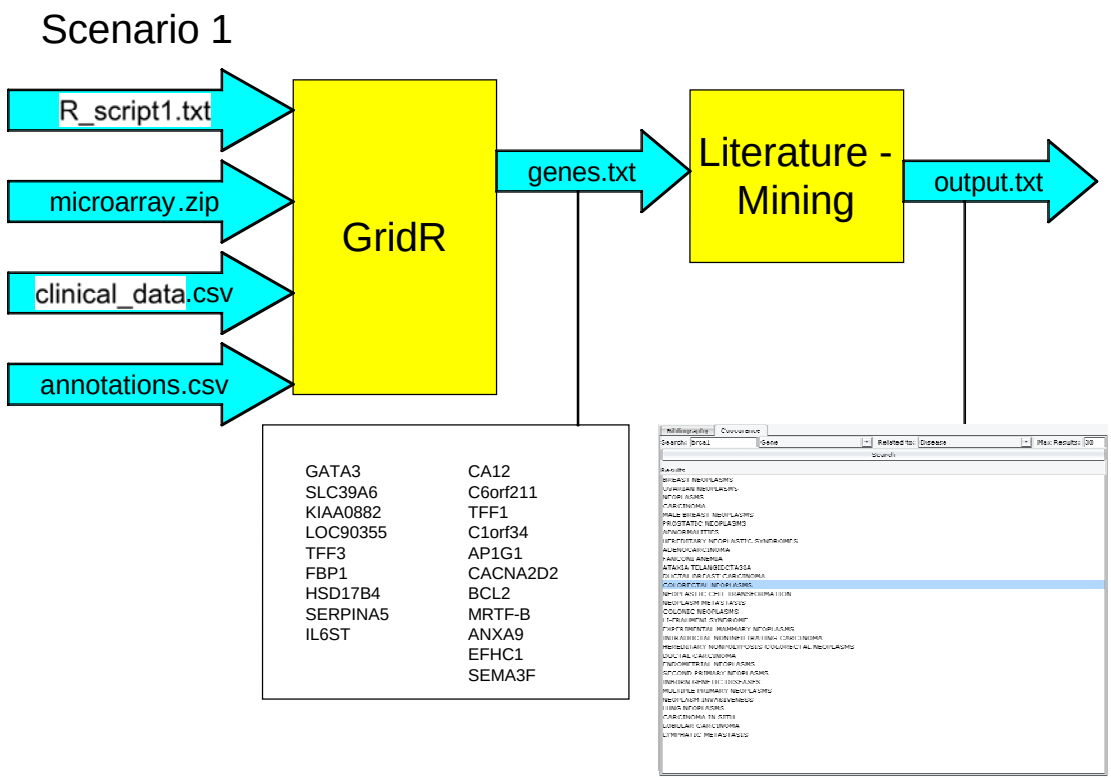

Figure 4: Scenario 1 of the GridR integration.

In the knowledge discovery scenario depicted Figure 4, the GridR service is provided with the R script to be executed and three data inputs, namely the mircoarray data, the clinical data and the user annotations. The mircoarry data is provided in a raw data format as .zip archive and the annotations are provided as comma separated text values. Here, the workflow is assumed to make use of predefined, fixed data formats. As described in the outlook, this will be replaced by allowing to retrieve format information dynamically from the meta data repository in the next version of the ACGT demonstrator. Using this input, GridR initiates the execution of the R script on the grid management system as described in Table 1. As a result, the gene names determined will be passed in a text file as input parameters to the Literature Mining tool, which will retrieve the state-of-the art literature regarding the detected genes.

In the knowledge discovery scenario depicted Figure 5, the input to the GridR execution includes additional input from clinical data in the form of comma separated values and the output consists of a postsript file which provides a visualization of the results computed by the R script.

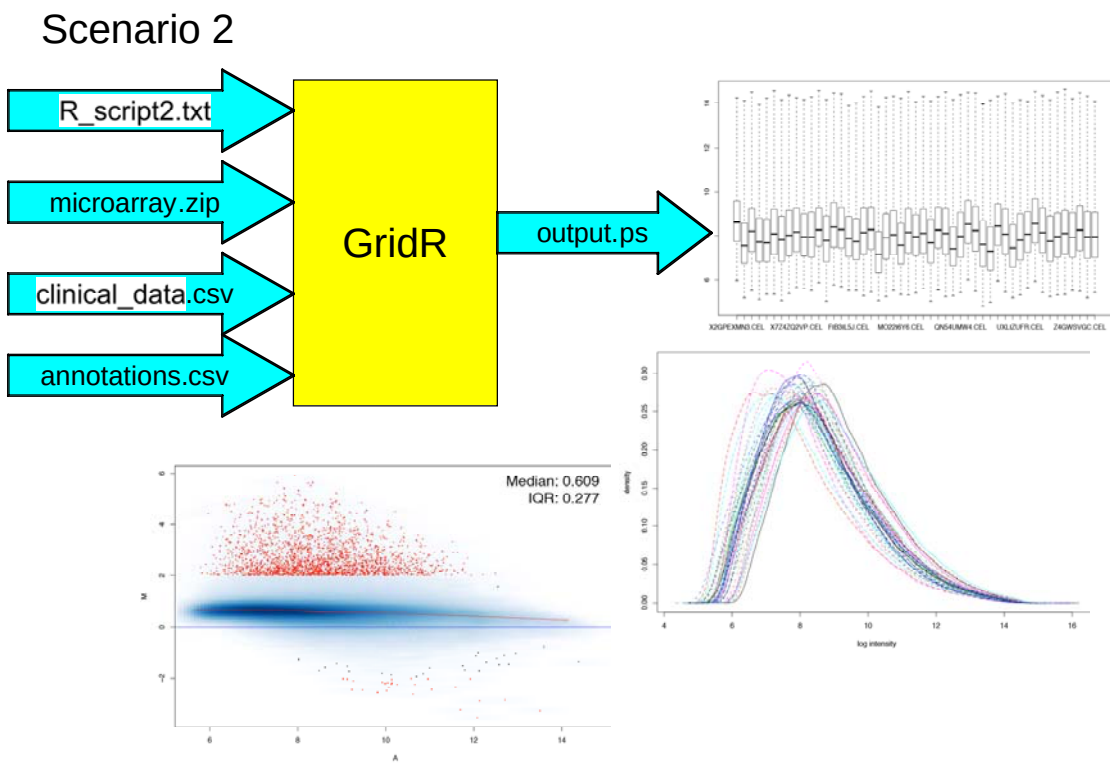

Figure 5: Scenario 2 of the GridR integration.

2.2.2 Technical Details

The GridR service is currently deployed as web service running on Axis 1 under http://grid5.kd-grid.ais.fraunhofer.de:8080/axis-acgt/services/GridRService?wsdl http://grid6.kd-grid.ais.fraunhofer.de:8080/axis-acgt/services/GridRService?wsdl. The GridR Webservice has the following method for interfacing:

executeScript

Executes an R script in the grid environment. A list of integer IDs, referring to the grid data management system (DMS), specifies the input data for the R script. A list of strings defines the names of the output files the script generates. Username and password of a myproxy certificate (on the dedicated myproxy server moss1.man.poznan.pl) have to be provided for accessing other grid services (GRMS and DMS). See the references below for more details.

Arguments

`String rCode`	the R script to be executed
`int[] inputIDs`	these are the IDs of the input files for the R script stored in the DMS
`String[] outputNames`	these are the names of the output files the R code creates

`String username`	username of myproxy hard coded on moss1.man.poznan.pl
`String pwd`	password of myproxy hard coded on moss1.man.poznan.pl

Return Value

`int[] outputIDs`	the DMS-IDs of the output files the R code generates

Details

The following text describes details on the execution of the executeScript method:

- According to the username and password provides as arguments, a credential object is received from the myproxy machine for getting access to the gridge and gridFTP services.
- The GridR service acts as client of the gridge services GRMS and DMS. The connection to these services is initialized and a unique directory in the user's home dir inside the DMS is created for storing data of the current GridR execution.
- The R code provided as argument is customization in a way that it can be executed in the grid, including e.g. a "matching" of the input data to data objects in the R code.
- A temporary local directory is created that is used for storing the customized R code that will be executed as file as well as an executable file (which is a shell script containing a special command for starting R on the execution machine in the grid).
- The files contained in the local temporary directory are uploaded physically to the DMS via gridFTP.
- The GRMS job description is generated, specifying which files to stage in to the execution machine, which command to execute and which output files to transfer back.
- The job is submitted and some monitoring info is stored in the log files of the tomcat container the GridR service runs in.
- After the execution finished, the result files of the GridR execution are automatically transferred to the DMS under the given names provided as arguments by the GRMS service. The GridR service performs a cleanup of the local temporary directory and returns a list containing the DMS IDs of the output files.

Future outlook:

The service will be fully integrated with the ACGT security infrastructure and will make use of credential delegation. The part of the implementation related to security will be customized and the arguments username and password will be obsolete.

There will be a similar method called “executeFunction” that executes a single R function in the grid.

2.2.3 Interaction with other components

The service is internally interfacing with the gridge services GRMS and DMS.

Externally, the webservice will be called by the workflow enactor and, in the future, by the GridR client.

2.2.4 References

Dennis Wegener, Thierry Sengstag, Stelios Sfakianakis, Stefan Rüping and Anthony Assi. GridR: An R-based grid-enabled tool for data analysis in ACGT clinico-genomic trials. In: Proceedings of the 3rd International Conference on e-Science and Grid Computing (eScience 2007), Bangalore, India.

2.3. GridR – Client for accessing ACGT from R

2.3.1 Overview

The R client for the GridR service enables the statistician to make use of the complete ACGT platform and services by using the (familiar) R language. R scripts can source out the execution of sub R scripts into the ACGT infrastructure in a transparent manner. The current implementation of the GridR client reflects the state of integration achieved in the ACGT review of April 2007 and still has to be adapted to the currect WebService-based implementation structure of the ACGT demonstrator.

2.3.2 Technical Details

As described above, GridR is used as a tool for the remote execution of R code in the grid. More specifically, the task of the execution of the R code is submitted as a grid job to a remote grid machine.

The client side part is structured around the components “RemoteExecution” (JobSubmission and JobDescription Generator) and “Locking”. The RemoteExecution component is responsible for the execution of R code as a job in the grid environment by transforming the R code to execute into a set of files, creating a job description file in the respective job description language and submitting the job to the resource management system by the GRMS-client. During this process, the locking component takes care of the consistency of files/variables.

These components are based on R functions (see Table 2 and Table 3), so no changes in the core R implementation are necessary. The functions are based, among other, on the following R functionalities:

- callbacks - functions that are executed automatically by R after the user has issued a command (this is used when checking for results)
- active bindings - a variable is replaced by a function call which is handling the locking system and allows working interactively with that variable. When the variable is read, the predefined function is called and returns the value associated to the variable (or an error code if the variable is locked). When a value is assigned to the variable the function is called with the value as parameter for storage in an internal structure.
- parsing of error code - checks for missing values, variables and functions in the code which is executed remotely

Name	Action
`grid.init`	Initializaion of all variables necessary for the grid execution
`grid.exit`	Unlocks all variables (grid.unlockAll) and removes all callbacks
`grid.printJobs`	Prints a summary of running jobs
`grid.getLibraries`	Returns the remote list of libraries (through grid.apply)
`grid.getVersion`	Returns the remote R version (through grid.apply)
`grid.apply`	Performs a remote execution of R functions; waits (callback) or sets a lock (grid.writeLock)

Table 2: List of functions for the users

Name	Action
`grid.callback`	Performs a callback
`grid.waitForAll`	Performs grid.callback for all jobs
`grid.readLock`	Read lock a variable
`grid.writeLock`	Write lock a variable
`grid.unLock`	Removes the lock from grid variable
`grid.unlockAll`	Unlocking of all grid variables
`grid.catchObjectNotFou`	Error handling by parsing of error messages

ndError	

Table 3: List of internal used functions

R as client

An R programming language interface that supports the access to the ACGT services is provided in the ACGT environment. This means that R users and developers will have access to distributed resources in a transparent fashion, as if those resources were local. The complexity of the grid is thus hidden from the user.

Again, accessing the ACGT grid environment requires no changes in the core R implementation. In practice grid access is performed through the call of predefined R functions loaded from a package. R users can make use of the grid technology in a transparent way by passing the functions to be executed in the grid as input to one of those predefined functions (grid.apply) in their local code.

Figure 6 shows the execution of a simple sum function with the GridR client side components. Details on the steps of execution will be described in the following section.

```
>
> source("gridR.R")
> grid.init(localDir="/home/dwegener/acgt/",
+             remoteHost="grid5.kd-grid.ais.fraunhofer.de",
+             remoteDir=":2811//home/dwegener/grmsjobs/test/",
+             grmsDir="/usr/local/grms/",
+             cogDir="/home/dwegener/cog-kit/cog-4_1_4/")
> grid.apply("y",sum,1:100)
> y
Fehler in function (...)  : Object y has write lock from grid function
> y
Grid job finished, result written to variable y
Fehler in function (...)  : Object y has write lock from grid function
> y
[1] 5050
>
```

Figure 6: Simple GridR Example

2.3.3 Interaction with other components

The R client for the GridR service can be used as part of the Workflow enactor or, in simple workflows, can be used as stand-alone tool for accessing the GridR service. Through the GridR services, the R client gets access to potentially all other knowledge discovery services in the ACGT environment as well as to all data maintained in the data management system. Furthermore, all computing resources managed by the grid management service are (inderictly) accessible by this to the R client.

2.3.4 References

[Wegener et. al 2007] Wegener, Dennis, and Sengstag, Thierry, and Sfakianakis, Stelios and Rüping, and Assi, Anthony. GridR: An R-based grid-enabled tool for data analysis in ACGT clinico-genomic trials. In: Proceedings of the 3rd International Conference on e-Science and Grid Computing (eScience 2007), Bangalore, India.

2.4. Literature Mining

2.4.1 Overview

The literature mining functionality consists of a set of individual capabilities supporting literature analysis and knowledge discovery services that will be directly available to ACGT end users via the ACGT Portal.

These services have no data anonymization requirements and only require the end user to have the proper authorization certificates that are managed by the respective ACGT service. Since they depend on Biovista's commercially available solutions, work to date has focused on 'gridifying' the services and making them compatible with the various other ACGT services, as these come online over time.

To date 2 basic services have been implemented and made compatible with the following ACGT modules:

1. The Workflow Editor module
2. The Certification Authority module
3. The Ontology Viewer module

The services that are presently deployed are the "Article Finder" service and the "Link Finder" service, with additional services planned for the future.

2.4.2 Technical Details

The webservices currently offered are:

getBibliography()

Description: This method returns the specified maximum number of PubMed ids. The method searches the most frequent occurrences of the desired search term (entity name). The caller must also specify the type of the search term i.e (name = cancer, type = disease). See below for a list of available types. In case of Invalid type or wrong input the method returns zero hits

- Input parameter: search_term, the entity name to look for
- Input parameter: the type of the search_term
- Input parameter: The max_results output is the upper bound of the size of the returning list
- Output: Returns a list of pubmed_ids containing the specified entity

getCoocurance()

Description: This method returns the specified maximum number of most frequent co-occurring terms with the specified search term. The caller must specify both the search term, and the destination type. See below for a list of available types. In case of non-matching or invalid input the method returns zero hits

- Input parameter: search_term, the entity name to look for
- Input parameter: the type of the search_term
- Input parameter: the type of the of co-occurring terms
- Input parameter: The max_results output is the upper bound of the size of the returning list
- Output: Returns a list of entities that co-occur with the specified search term.

getAvailableTypes()

Description: This is a convenient method for the caller in order to retrieve the valid types, the services getBibliography and getCoocurance can handle.

- Input parameter: None
- Output: Returns a list of all the available types the methods getBibliography and getCoocurance can handle

2.4.3 Interaction with other components

Currently the literature mining services are used / integrated with the following modules:

- Workflow editor from ITE
- Ontology Viewer from Biovista
- LiteratureMininingServiceDemoClient from Biovista

2.4.4 References

- http://wiki.healthgrid.org/index.php/ACGT:Biovista_Services
- http://wiki.healthgrid.org/index.php/ACGT_Web_Service_reference_implementation

2.5. Clustering

Clustering is a reference implementation for including R libraries for clinico-genomic data analysis

2.5.1 Overview

The goal of clustering (also known as unsupervised classification) is to organize a set of objects or individuals (eg. patients, genes, proteins etc..) characterized by a set of 'attributes' or 'variables', into classes and sub-classes of similarity. In microarray

experiments genes are characterized by their expression levels across samples. The attributes describing the objects may be of different types such as numeric (quantitative), nominal (categorical) or ordinal, just to mention some most frequent types. The notion of similarity or dissimilarity between individuals is crucial in clustering since it aims at putting together, in a same cluster, the individuals that 'resemble' to each other. Clustering algorithms divide a set of objects into groups so that the objects within a same group are more similar than those across groups. In the particular case of gene expression microarray experiments, clustering algorithms can be used to discover the groups of genes having similar profiles i.e. similar pattern of expression across samples. There are two approaches to clustering techniques: hierarchical and non-hierarchical. The former provide a series of nested clusters whereas the latter find a single partition into a predetermined number of classes. There are two kinds of hierarchical approaches: agglomerative and divisive. Agglomerative (bottom-up) algorithms begin with every object in a distinct cluster. At the first step, two most similar objects are joined. Then, at every step, the two most similar objects or clusters are joined until all the objects are joined in a single cluster. Detailed description of the algorithms, choice of metrics (distance measures) and linkage methods can be found in Gordon (1999) and Quakenbush (2001).

In the Clinico Genomic Trials, gene expression data as well as clinical trials data are available for a set of patients. Clustering the set of genes based on the expression data can produce groups of genes with similar patterns on the one hand, and clusters of patients can be obtained from clinical data, on the other. The link between the clusters of genes and those of patients can be studied with the help of statistical measures of association.

The crucial role played by cluster analysis in genomics was first demonstrated by Eisen et al (1998) and Brown P.O., Botstein D. (1999). They proposed to use hierarchical clustering strategy to group genes on the basis of similarity in their patterns of expression and organize the data to find inherent orderly features. Then the ordered tables of data produced by these methods are displayed graphically in ‘functional maps’. The clustering methods most commonly used linkage methods in hierarchical clustering (Single Linkage, Complete Linkage, Average Linkage, Centroid, and Ward’s criteria) and they make use of Euclidean distance or distance measures based on correlations between genes or samples most often for numerical data.

The clustering tools that will be developed for use of clinicians and researchers in ACGT are based on the Likelihood Link Analysis (LLA) methodology. Here a measure of similarity is defined as the ‘likelihood’ of the resemblance between the objects to be clustered (eg genes or patients) under some non-link hypothesis, giving rise to a probabilistic measure of similarity devoid of bias. The cluster joining strategy is itself defined as the likelihood of the similarity between clusters, a kind of probabilistic aggregation criterion. This methodology provides two statistics of utmost practical importance: the ‘global index’ that measures the quality of the partition obtained at different levels of hierarchy, and enables the user to choose the most ‘coherent’ partition, and the ‘local index’ which allows to detect the ‘significant nodes’ associated with the ‘strong clusters’.

The program CHAVLH which integrates the LLA methodology is the result of a long series of research articles and thesis. It is written in FORTRAN77 and in its latest version it can be used to cluster either a set of objects or a set of variables. Five

different types of variables are considered in CHAVLH: quantitative or numerical variables, logical or Boolean variables, qualitatitive nominal or categorical variables, qualitatives ordinal variables and qualitative pre-ordinance variables. This tool (CHAVLH) is integrated in the R package called LLAhclust which, at this stage, does not include pre-ordinance variables.

2.5.2 Technical Details

The package LLAhclust contains routines for computing various types of probabilistic similarity measures between objects (LLAsimobj) or variables (LLAsimvar). Once the similarity values between objects/variables are computed, a hierarchical agglomerative clustering can be performed using several probabilistic and non-probabilistic linkage criteria, and indices measuring the quality of the partitions compatible with the resulting hierarchy can be computed. R version 2.1.0 or higher is needed. The R functions plot and cuttree can be used to produce the dendrogram and to to obtain a partition by cutting the tree at a given leve of the hierarchy. The package is developed in collaboration of Ecole Polytchnique de Nantes, and is available under the licence CeCILL2 (GNU GPL 2 compatible). The authors are Ivan Kojadinovic, I. C. Lerman, P. Peter.

Description of routines.

LLAsimobj: Computes similarities among objects

Description

Computes similarities among objects using the likelihood linkage analysis approach proposed by Lerman. The likelihood linkage analysis method mainly consists in replacing the value of the similarity coefficient between two objects by the probability of finding a lower value under the hypothesis of *absence of link*. See the references below for more details.

Usage

```
LLAsimobj(x, method = "LLAeuclidean", upper = FALSE)
```

Arguments

`x`	a numeric matrix or data frame.
`method`	Can be one of `LLAeuclidean`, `LLAcosinus`, `LLAcategorical`, `LLAordinal`, or `LLAboolean`. The two first methods can be used to compute similiarty coefficients between objects described by numerical variables.
`upper`	logical value indicating whether the upper triangle of the similarity matrix should be printed by `print.LLAsim`.

Details

The following functions are also defined for objects of class `LLAsim`: `names.LLAsim`, `format.LLAsim`, `as.matrix.LLAsim` and `print.LLAsim`.

Value

Returns an object of class `LLAsim` whose attributes are very similar to those of objects of class `dist`. See `dist` for more details.

LLAsimvar : computes similarities among variables using the likelihood linkage analysis approach

Description

Computes similarities among variables using the likelihood linkage analysis approach proposed by Lerman. The likelihood linkage analysis method mainly consists in replacing the value of the estimated similarity coefficient between two variables by the probability of finding a lower value under the hypothesis of stochastic independence, called *absence of link* in that context. Nine similarity coefficients can be computed using the `LLAsimvar` function.

Usage

```
LLAsimvar(x, method = "LLAnumerical", upper = FALSE,
                  simulated.distribution = NULL)
```

Arguments

`x`	a numeric matrix or data frame.
`method`	Can be one of `LLAnumerical`, `LLAcategorical`, `LLAordinal`, `LLAboolean`, `chi.square`, `pearson.abs`, `spearman.abs`, `kendall.abs` or `empirical.copula`. The methods `LLA*` were initially defined by Lerman (see references below). The four remaining methods compute the similarity between two variables as one minus the p-value obtained from a test of independence
`upper`	logical value indicating whether the upper triangle of the similarity matrix should be printed by `print.LLAsim`.
`simulated.distribution`	Object of class `empcopula.simulation`. Should be set only if the method `empirical.copula` is selected.

Details

The following functions are also defined for objects of class `LLAsim`: `names.LLAsim`, `format.LLAsim`, `as.matrix.LLAsim` and `print.LLAsim`.

Value

Returns an object of class `LLAsim` whose attributes are very similar to those of objects of class `dist`. See `dist` for more details.

LLAhclust: Hirarchical clustering using likelihood linkage criterion

Description

Builds a hierarchy from similarity coefficients among objects or variables as returned by `LLAsimvar`, `LLAsimobj` or `as.LLAsim`. The default aggregation criteria, called `lla`, can be regarded as a probabilistic version of the single linkage.

Usage

```
LLAhclust(s, method = "lla", epsilon = 1, members = NULL)
```

Arguments

`s`	Similarity coefficients as returned by `LLAsimvar`, `LLAsimobj` or `as.LLAsim`.
`method`	Linkage method (i.e. aggregation criterion). Can be one of `lla` (default), `tippett` (Tippett's p-value combination method), `average`, `complete`, `fisher` (Fisher's p-value combination method), `uniform` (uniform p-value combination method; can be regarded as a probabilistic version of the average linkage), `normal` (normal p-value combination method) or `maximum` (maximum p-value combination method; can be regarded as a probabilistic version of the complete linkage).
`epsilon`	Coefficient used in the `lla` linkage. Should lie in [0,1]: `epsilon=0` corresponds to the single linkage, `epsilon=1` (default) yields a probabilistic version of the single linkage.
`members`	"Weights" of the objects to be clustered if not of equal "weight". See `hclust` for more details.

Value

An object of class `hclust` with the corresponding attributes. See `hclust` for more details.

LLAparteval: evaluates the quality of each partition compatible with the hierarchy (local and global indices)

Description

Evalutes the quality of each partition compatible with the hierarchy returned by `LLAhclust` . If the hierarchy is obtained from similarity coefficients computed using `LLA*` methods, the global and local statistics proposed by Lerman are calculated. Otherwise, for similarity coefficients obtained from independence tests (see `LLAsimvar`), for each partition, the inter-class p-values are combined using Tippett's and Fisher's rules. Furthermore, the minimum inter-class p-value and the maximum intra-class p-value are given.

Usage

```
LLAparteval(tree, s, m=NULL)
```

Arguments

`tree`	An object of class `hclust` as returned by `LLAhclust`.
`s`	An object of class `LLAsim` as returned by `LLAsimvar`, `LLAsimobj` or `as.LLAsim`.
`m`	Integer. If set, the quality of the m coarsest partitions only is evaluated.

Value

Returns a `data.frame` whose columns are: `global.stat` and `local.stat` if the hierarchy is obtained from similarity coefficients computed using `LLA*` methods, and `tippett.inter`, `fisher.inter`, `min.inter` and `max.intra` in case of similarity coefficients obtained from independence tests.

Other useful functions are: *as.LLAsim* (converts the lower triangle of a square matrix of similarities into a *LLAsim* object) and *as.matrix.LLAsim* (converts a *LLAsim* object into a square symmetrical matrix. The usual R functions *format*, *print* and *names* have also been extended to deal with *LLAsim* objects.

The complete documentation is available in the help files.

2.5.3 .Interaction with other components

Through the use of the GridR component we will be able to execute our code on the cluster using the Globus Toolkit.

2.5.4 References

[Eisen et al] Cluster analysis and display of genome-wide expression patterns, Eisen MB, Spellman PT, Brown PO, Botstein D., 1998, PNAS USA, 95(25)

[Brown and Botstein] Exploring the new world of the genome with DNA microarrays, Brown P.O., Botstein D.,1991, Nature Genetics, 21

[Gordon 1999] Classification, Gordon AD, New York: Chapman and Hall/CRC

[Quakenbush 2001] Computational analysis of microarray data, Nature Review Genetics 2, pp 418-427

[Lerman, 1981] Classification et analyse ordinale des données, Dunod, Paris, 1981

[Lerman, 1991] Foundations of the likelihood linkage analysis classification method, I. C. Lerman, 1991, Applied Stochastic Models and Data Analysis, 7, pp 63-76

[Lerman, 1993] Likelihood linkage analysis classification method: An example treated by hand, I. C. Lerman, 1993, Biochimie, 75, pp 379-397

[Lerman et al 1993] Principes et calculs de la méthode implantée dans le programme CHAVL (Classification Hiérarchique par Analyse de la Vraisemblance des Liens), I. C. Lerman, Ph. Peter, H. Leredde, Modulad, 12, pp 33-101

[Kojadinovic 2007] Hierarchical clustering of continuous variables on the empirical copula process, submitted

LLAhclusr web site http://www.polytech.univ-nantes.fr/LLAhclust/

R project web site http://cran.r-project.org/

2.6. Clustering Discretised Gene-Expression Data

2.6.1 Overview

A novel clustering algorithm based on the special revision and customization of the k-means clustering algorithm, named ***discr_kmeans***. The algorithm identifies clusters of co-regulated genes from respective microarray data. With a subsequent filtering

operation are retained the clusters that exhibit - in an *adequate number of samples*, *strong gene-expression profiles*. A sample with a strong gene-expression profile for a specific cluster of genes is one that exhibits, in an *adequate percentage* of the current clusters' genes, '*high*' (i.e., '*up-regulated*') or, '*low*' (i.e., '*down-regulated*') gene expression levels. The adequacy of the number of samples covered by a retained cluster, as well as the adequacy of the percentage for considering a sample's expression profile as strong, is determined by user specified threshold. The retained clusters of genes are referred as ***Metagenes***.

2.6.2 Technical Details

The discr_kmeans algorithm is part of a stand-alone tool suited for gene-selection from microarray data, named MineGene. The basic functionality of discr_kmeans has been also developed and deployed as a *Web-service* in the context of the ACGT infrastructure. The service is called and run within *ACGT's workflow environment*.

The *MineGene/discr_kmeans* Web-service interface and execution steps:

Methods: **`executeMineGeneDiscr_kmeans`**

Input parameters of `executeMineGeneDiscr_kmeans` method:

- Read Input Samples' Microarray/Gene-Expression and Clinico-Histopathology Data (URL/XML-file): `String`
- Threshold for the adequacy number of samples covered by clusters: `Real`
- Threshold for the minimum low/high gene-expression percentages to consider a sample as strong: `Real`

- Output: A file (URL/XML-file) that stores the final clusters/metagenes as well as the original samples' clinico-histopathology data): `String`

2.6.3 Interaction with other components

The MineGene/discr_kmeans Web-service does not interact (during its execution) with any other Web-service. Just its output (URL/XML-file) is the input to the HealthObs/AssociationRulesMining Web-service (see below).

2.6.4 References

1. Potamias, G., and Kanterakis, A. (2005). Supporting Clinico-Genomic Knowledge Discovery: A Multi-strategy Data Mining Process. *LECT NOTES ARTIF INT – LNAI* 3955, 520-524.
2. Potamias, G., May, M., and Ruping, S. (2006). Grid-based Knowledge Discovery in Clinico-Genomic Data. *LECT NOTES BIOINFORMATICS* (LNBI) 4345: 219-230.
3. Potamias, G., *et al.* (2007). Knowledge Discovery Scientific Workflows in Clinico-Genomics. *Annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI'07)* (to appear in conference Procs).

2.7. Association Rules Mining (ARM)

2.7.1 Overview

Mining of *interesting clinico-genomic asso*ciations. The approach is based on utilization and customization of standard ***association rules mining*** (ARM) algorithms (e.g., Apriori, prefix-tree data structures etc). The quest is to explore and discover (potential) interesting and ***evidential*** (i.e., of high confidence, and/or user qualification) associations between *genomic samples' profiles*, i.e., metagenes (the

clusters of genes with dominantly low/high profiles, as discovered from the discr_kmeans clustering operation/service), and respective *clinico-histopathological sampless' profiles* (e.g., disease type/class or, outcome, survival etc.

2.7.2 Technical Details

The ARM operations are part of a stand-alone application suited for clinical data mining, named HealthObs. The basic functionality of HealthObs/ARM has been also developed and deployed as a *Web-service* in the context of the ACGT infrastructure. The service is called and run within *ACGT's workflow environment*.

The *HealthObs/ARM* Web-service interface and execution steps:

- Methods: `executeHealthObsARM`
- Input parameters of `executeHealthObsARM` method:
  - Input File with Metagenes and Clinico-Histopathology Data (output from MineGene/discr_kmeans service (URL/XML-file): `String`
  - List of attributes (metagenes and clinico-histopathology features) to focus and their inclusion in the 'IF' or 'THEN' part of association rules: `List of Strings`
  - Minimum-Support (minsup) for association rules: `Real`
  - Minimum-Confidence (minconf) for association rules: `Real`
- Output: A file (URL/XML-file) with association rules: `String`

2.7.3 Interaction with other components

The MineGene/discr_kmeans Web-service does not interact (during its execution) with any other Web-service. Just its input is the output URL/XML-file of the MineGene/discr_kmeans service. Its output (URL/XML-file with retained association rules) is passed as input to a stand-alone application (on the user's site) that ***visualises*** association rules (part of HealthObs tool).

2.7.4 References

1. Potamias G., Koumakis L., and Moustakis V. (2004). Mining XML Clinical Data: The HealthObs System. *Ingenierie des systems d'information* 10(1): 59-79.
2. Potamias G., and Koumakis, L. (2004). HealthObs: An Integrated System for Mining Clinical XML-formatted Data. *In Proceedings 2nd ICICTH International Conference on Information Communication Technologies in Health*, July 8-10, 2004 Samos Island, Greece, pp 359-364.
3. Potamias, G., and Kanterakis, A. (2005). Supporting Clinico-Genomic Knowledge Discovery: A Multi-strategy Data Mining Process. *LECT NOTES ARTIF INT – LNAI* 3955, 520-524.
4. Potamias G., Tsiknakis M., Papoutsidis, V., Kanterakis A., Marias K., and Kafetzopoulos D. (2006). Advancing Clinico-genomic Research Trials via Integrated Knowledge Discovery Operations (extended paper + poster). *Medical Informatics in Europe* conference (*MIE 2006*), 27-30 August 2006, Maastricht, the Netherlands.
5. Potamias, G., *et al.* (2007). Mining Interesting Clinico-Genomic Associations: The HealthObs Approach. *IFIP International Federation for Information Processing, Volume 247, Artificial Intelligence and Innovations 2007: From Theory to Applications, eds. Boukis, C., Pnevmatikanis, L., Polymenakos, L., (Boston: Springer),* pp. 137-145.

2.8. Mediation Between Microarray & Clinical Data

2.8.1 Overview

A ***data mediation*** service (Mediator) for the *seamless* retrieval and integration of samples' microarray/gene-expression data, and respective clinico-histopathological patients'/samples' data. The service supports data-mediation operations between two clinical information systems (OncoSurgury and HistoPathology), and a microarray/gene-expression information system (GenIS – based on enhancement and customisation of the BASE[1] system). It supports, via special *GUI*-based forms, the formulation of complex clinico-genomic queries. In its current deployment supports breast-cancer protocols and respective standard information and data models. For example, the biomedical investigator can form *combined clinico-genomic queries* (e.g., *"get the gene-expression profiles for 'microarray experiments meeting specific criteria' of all breast cancer diseased women meeting a clinical profile of: 'age over 40', 'ER+', 'lymph-node –' and 'followd-up for over 5 years' "*). Note: Currently all references to patients', and to respective samples' data is done completely **anonymously**.

2.8.2 Technical Details

The clinico-genomic data mediator (CGMediator) is implemented as a Web-application and it has also been deployed as a Web-service in the context of the ACGT infrastructure. It is called and executed by the ACGT workflow environment.

The *CGMediator* Web-service interface and execution steps:

- Methods: **`executeCGMediator`**
- Input parameters of **`executeCGMediator`** method:
  - URL of Clinical Information Systems data-base: `String`
  - URL of Microarray/BASE information system data-base: `String`
- Output: A folder (URL-folder) that stores: (a) An XML-file with the clinico-histopathological data, and the genomic (microarray) fields retrieved for respective set of samples'/patients'; (b) A set of files with the exact gene-expression profiles of the retrieved samples.

2.8.3 Interaction with other components

The *CGMediator* Web-service does not interact (during its execution) with any other Web-service. Just its output (URL-file) is the input to the MineGene/discr_kmeans Web-service.

2.8.4 References

1. Tsiknakis, M., Kafetzopoulos, D., Potamias, G., Analyti, A., Marias, K., and Manganas, A. (2006). Building a European biomedical grid on cancer: the ACGT Integrated Project. *Stud Health Technol Inform.* 120: 247-258 [PubMed Id - PMID: 16823143].
2. Analyti, A., Kondylakis, H., Manakanatas, D., Kalaitzakis, M., Plexousakis, D., and Potamias, G. (2006). Integrating Clinical and Genomic Information through the PrognoChip Mediator. *LECT NOTES BIOINFORMATICS* (LNBI) 4345: 250-261.

[1] http://base.thep.lu.se/

3. Tsiknakis, M., Kafetzopoulos , D., Potamias, G., Analyti, A., Marias, M., and Manganas, A. (2006). Building a European Biomedical Grid on Cancer: The ACGT Integrated Project. *Stud Health Technol Inform.* 120: 247-258. [PubMed ID: 16823143]
4. Manolis Tsiknakis, Dimitris Kafetzopoulos, Giorgos Potamias, Anastasia Analyti, K. Marias, A. Maganas, *Building a European Biomedical Grid on Cancer: The ACGT Integrated Project.* (2006). In *Procs. of the HealthGrid 2006 conference*, Valencia, Spain, June 2006, Studies in Health Technology and Informatics, Vol. 120, pp. 247-258.

3 Outlook

This section gives an outlook on the further developments to be undertaken and integrated in WP6.

3.1. Tool metadata repository

Introduction

Many bioinformatics tools are available on-line but because of rapid developments of new and improved tools, it is necessary to be able to publish and maintain metadata about such tools in a public catalogue. The tool metadata repository is responsible for providing this functionality in ACGT.

Available metadata

An initial proposal for tool metadata was given in Deliverable 6.1 (section 3.1.9 “Services Metadata Initial Proposal”). This proposal is still valid with some minor modifications. In the first version of the tool metadata repository, metadata for the following entities are supported:

- Service definition including functional description
- Parameter (variable) definition
- Data type
- Admin data definition
- Provider
- Workflow metadata definition

The next version will support the remaining metadata:

- Workflow application metadata definition. This metadata is used to describe actions and results from workflow enactments.
- Visualization definition

In deliverable 6.3, the metadata repository and model will be presented with more detail.

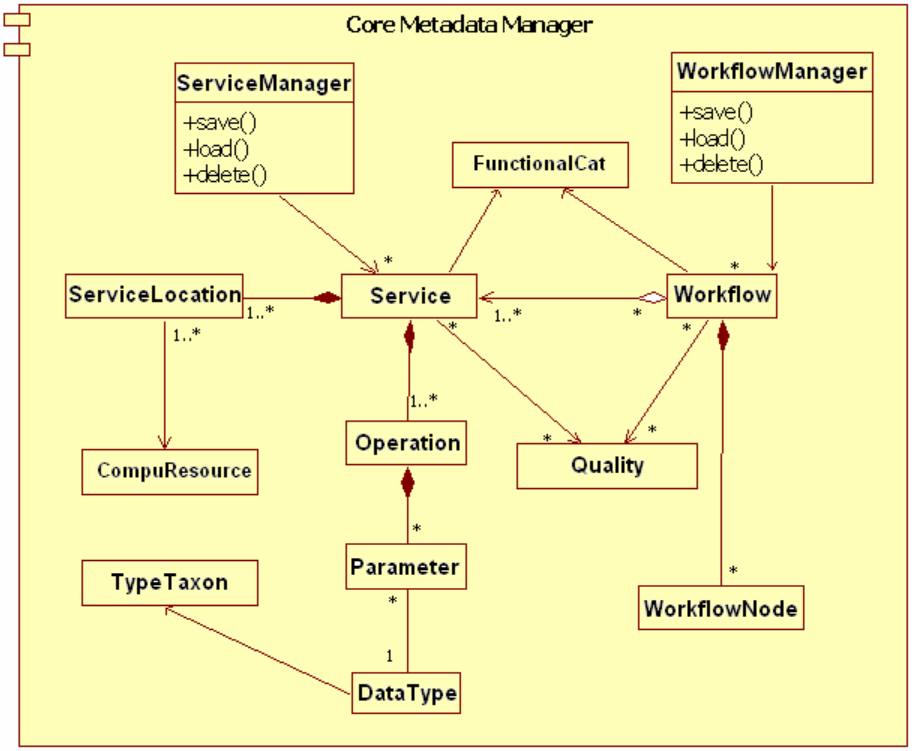

Figure 7 Overview of the current metadata model used in the repository. It shows available entities and their relations.

Implementation description

To access information about available tools in ACGT, an application programming interface (RepoServices API) has been developed as a SOAP web service. This API is used by the Java portlets of the ACGT portal for registering and browsing the metadata. The current version of the workflow editor (AWE) uses the same metadata model and will also use the RepoServices API in future versions.

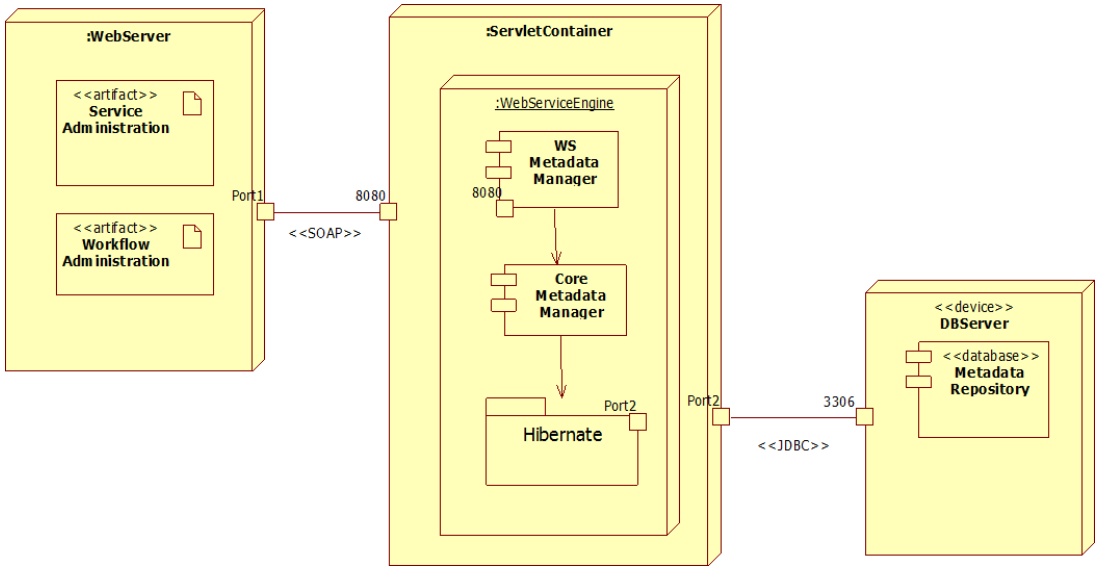

Figure 8 Repository deployment. Data storage layer corresponds to the DBServer, Data access layer with object persistence is made available with the Core Metadata Manager. The WS Metadata manager exposes the same functionality as in the Core Metadata Manager but as web services.

The RepoServices API provides the following functionality for the repository:

- Registering new entities
- Editing existing entities
- Removing existing entities
- Retrieving existing services and workflows based on
    - Input data type
    - Output data type
    - Functional category

The metadata repository is implemented in several layers (see also Figure 8):

- Data storage layer: The model in Figure 7 is implemented as MySQL database tables with the relations indicated in the figure. Scripts have been created that allow the construction of the database instance.
- Data access layer: Hibernate mappings are used to create persistent Java objects. The properties of the Java objects are mapped to columns in the database tables corresponding to the model in Figure 7
- Web service layer: Axis2 is the web service engine and Tomcat is used as the servlet container of the deployment. JiBX is also used as a framework in the development of the web service, binding XML to Java code.

3.2. Additional Tools and Services

Further tools to be developed include:

Support for massively parallel data mining

The analysis of the huge amount of clinico-genomic data in ACGT requires the use of massively parallel computing resources for achieving an acceptable performance for the end user. Execution and resource management support will be provided for enabling the data mining algorithms to exploit parallelism.

- Implementation of support for massively parallel data mining
- Parameter sweep
- Input file sweep
- Feature selection
- Model selection
- Profiling and runtime optimization of typical data mining workflows

Implementation of specialized tools and services for clinico-genomic data

Specialized data-mining tools to clinico-genomic data will adapt approved data-mining algorithms to the ACGT application domain and will be made available as services including:

- Frequent Itemsets
- Visualization
- GridR client to access the ACGT infrastructure from within R

Implementation and test of reference workflows for ACGT clinical trials

Representative example workflows for patient data analysis requiring and integration and variety of knowledge discovery tools will identified and implemented.