

PUBLIC

SIXTH FRAMEWORK PROGRAMME
PRIORITY 1
LIFE SCIENCES, GENOMICS AND BIOTECHNOLOGY FOR HEALTH



DIAMONDS PROJECT

Contract no: LSHG-CT-2004-512143

EU Deliverable
D5.2

Definition of the essential design characteristics of the toolbox

Date: 30th June 2005

Partner responsible: Noray Bioinformatics S.L.



TABLE OF CONTENTS

1	INTRODUCTION.....	4
2	PLATFORM GENERAL DIAGRAMS	3
2.1	FLOW LIKE DIAGRAM	4
2.1.1	INTRODUCTION	4
2.1.2	NOTATION.....	4
2.1.3	MODULES	5
2.2	BLOCK DIAGRAM	8
2.2.1	INTRODUCTION AND NOTATION	8
2.2.2	BLOCK DIAGRAM MODULES	12
2.2.2.1	UPLOADING DATA.....	12
2.2.2.2	EXPRESSION DATA ANALYSIS.....	13
2.2.2.3	GENE HOMOLOGIES.....	16
2.2.2.4	PHYLOGENETIC TREES	16
2.2.2.5	PROMOTER ANALYSIS	18
2.2.2.6	PROTEIN CLASSIFICATION	19
2.2.2.7	INTERACTION MODES AND KEY INTERACTIVE RESIDUES	21
2.2.2.8	RESIDUE SUBSTITUTIONS.....	22
2.2.2.9	PATHWAYS ANNOTATION.....	24
2.2.2.10	VISUALISATION	26
2.2.2.11	MODELLING AND SIMULATION.....	26
2.2.3	BACKGROUND PROCESSES.....	29
2.2.3.1	ANNOTATION MODULE.....	29
2.2.3.2	TEXT MINING and DATA CURATION	31
2.3	DIAGRAM DEVELOPMENT	32
3	DIAMONDS WORKBENCH.....	33
4	INTEGRATION TECHNOLOGY: WEB SERVICES.....	36
4.1	INTRODUCTION AND ADVANTAGES	36
4.2	WEB SERVICES TECHNOLOGIES	37
4.3	ALTERNATIVES	39
5	PLATFORM IMPLEMENTATION	40
5.1	FUNCTIONALITIES	40
5.2	CORE DATABASE	43
5.3	FUTURE EXTENSIONS.....	46
6	CONCLUSIONS.....	47
	REFERENCES.....	50

TABLE OF FIGURES

Figure 1. Process and Background Process	4
Figure 2. Data	5
Figure 3. Solid Arrow	5
Figure 4. Dotted Arrow	5
Figure 5. Module	5
Figure 6. Flow-Like Diagram	7
Figure 7. Modules, data sets and blocks	8
Figure 8. Dependencies	9
Figure 9. Data dependencies	9
Figure 10. Multiple sources example	10
Figure 11. Block Diagram	11
Figure 12. Uploading data	12
Figure 13. Expression Data Analysis	13
Figure 14. Gene Homologies	16
Figure 15. Phylogenetic trees construction	17
Figure 16. Promoter Analysis	18
Figure 17. Protein classification	20
Figure 18. Interaction modes and Key interactive residues	21
Figure 19. Residue substitutions	23
Figure 20. Pathways annotation	24
Figure 21. Visualisation	26
Figure 22. Modelling and Simulation	27
Figure 23. Annotation module	29
Figure 24. Modules involved in the block diagram	33
Figure 25. Relationship between modules and user options	33
Figure 26. User directory	34
Figure 27. Input and Output structure	35
Figure 28. Workbench example and history file	35
Figure 29. Web services usual operation	38
Figure 30. Block Diagram with tools assignments	42
Figure 31. Database queries centralised	44
Figure 32. GIN-DB structure	45

1 INTRODUCTION

The present paper is Deliverable 5.2 of DIAMONDS Project: Definition of essential design characteristics of the toolbox. The objective of this deliverable is to explain and collect the list of functionalities and characteristics that the DIAMONDS platform may contain.

In this sense, the DIAMONDS platform's main objective is to offer an integrated framework for systems biology, providing a unique access point to several tools and functionalities.

We can distinguish between two kinds of partners involved in the project:

- Wet-lab partners: Will generate data for feeding the platform and will be the users of it.
- Dry-lab partners: Will be involved in the platform development, providing different tools and integration facilities. All these tools will be considered to become integrated in a common workbench.

The main goals and advantages of the DIAMONDS platform are listed below:

- ✓ Tools involved in the DIAMONDS project will cover a great number of functionalities together enabling data integration for systems biology.
- ✓ The users can access these tools using a unique access point.
- ✓ All data and knowledge collected and processed with the different tools will be stored in same framework, and will be available for the user at any moment for further use.
- ✓ The core database of the project will store very high quality data.
- ✓ The Platform is designed to be flexible and extendable

For developing the present deliverable, all partners have answered a questionnaire regarding the specific tools they can offer, in order to evaluate the role and added complexity to fit these tools in the platform.

In a parallel way, we have developed in collaboration with Martin Kuiper (project coordinator) and his team, two diagrams describing the whole system:

Flow-Like diagram: Describes the system by means of data flows, the first step for the system description. It has been developed by Martin Kuiper's team.

Block Diagram: Taking the flow diagram as starting point, the block diagram gathers all the functionalities the platform should ideally perform. It has been developed by NorayBio.

The following step has been to combine the knowledge obtained in the diagrams and questionnaires: In the block diagram we have considered all the potential functionalities of the platform. Based on the tools' questionnaires, we assigned tools to tasks.

The result is the present deliverable, which is structured as follows:

- **Diagrams explanation.** Both diagrams (flow like diagram and block diagram) are explained and detailed, stressing the importance of the functionalities of the platform.
- **Workbench.** The platform design must allow the user to have available at any moment all data and knowledge essential for network building.
- **Web Services.** Technological proposal for platform integration.
- **Platform implementation.** Assignments of *tools-functionalities*, for the complete platform diagram.

On 20th and 21st of June all DIAMONDS partners met in Bilbao and reached a consensus regarding this deliverable's contents.

2 PLATFORM GENERAL DIAGRAMS

For the platform definition, two different describing diagrams have been developed:

- **Flow-like diagram:** Describes the flows of data involved in the project. It has been developed by Martin Kuiper's team and it is shown in Figure 6.
- **Block diagram:** Describes the platform functionality. It is composed of modules and functional blocks, linked by dependencies. It has been developed by NorayBio and it is shown in Figure 11.

The reason for having two diagrams is that the platform is too complex to capture all data in a single one. The two diagrams together make the platform model clearer and easier to understand, as they are complementary. The block diagram will be the starting point to build the software platform, with the challenge to take into account all the data flows managed for every process.

In the present document we will explain and analyze both diagrams.

The flow-like diagram shows a functional viewpoint of the DIAMONDS project. Alternatively, the block diagram represents an abstraction/simplification of the flow-like diagram, trying to map functionality to the software modules of the project. It is necessary to perform this type of knowledge abstraction in order to get a starting point for the integration phase.

2.1 FLOW LIKE DIAGRAM

2.1.1 INTRODUCTION

A flow-like diagram has been created for both having a global vision of the processes and data flow, and communicating between each other among the DIAMONDS partners. This diagram is by no means comprehensive. On the contrary, it is intended to be tuned and continuously updated by each participant according to the emerging needs, as they are going to be identified in the defined Use-Cases (below).

The diagram is the result of internal brainstorming sessions and productive discussions with NORAYBIO. It is based on the description of the project[1] trying to account for all the analyses foreseen at the time the project was drafted.

This chapter first presents a description of the used notation, next a summary of each module is provided, then short description of the model and results are given, and finally some frequently asked questions are listed.

2.1.2 NOTATION

- **Process:** This element represents a time-consuming activity to be performed according to some given input that generates or produces a result (output). Probably some existing processes might be either split (to increase the level of detail of the diagram for instance) or merged (to reduce the redundancy for example).
- **Background process:** It is an activity (similar to the previous described element) that is meant to be performed “behind the screen”, that is, it will not be an active part of the DIAMONDS platform. Annotation/curation and text mining for bulk data uploading are typical background process types.



Figure 1. Process and Background Process

- **Data:** This box represents a data repository or data source. Most of these elements are meant to feed the data warehouse.

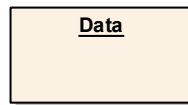


Figure 2. Data

- Solid Arrow: It connects two elements (processes/data) and is usually labelled. Its label specifies the input for the destination element and the output of the origin element. Depending on the required granularity level, some new solid arrows should be added or removed.

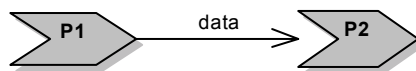


Figure 3. Solid Arrow

- Dotted Arrow: It shows the dependency between two elements. It may have a label describing the dependency relationship. The element at the head of the arrow depends on the one at the tail. Many dependencies are not shown in order to reduce the complexity of the diagram.

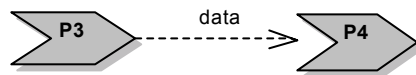


Figure 4. Dotted Arrow

- Module: A module highlights a virtual group of related elements. Modules point out some key functionalities or even workpackages like Modelling and Simulation for example.

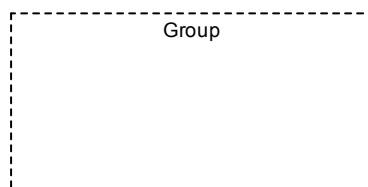


Figure 5. Module

2.1.3 MODULES

Most of the processes have been inserted within a module that groups related elements. Dependencies among modules are also shown in the diagram.

- **Prior and novel data:** Groups most of the principal data repositories or sources. It is highly connected to the Warehouse module since the latter holds all the gathered data.
- **Warehouse:** This module encompasses the data integration process that basically is fed by all the input data sources. It is strongly attached to the main repository (GIN-DB[2]). The data integration process is the neural centre of the system since all processes need data or produce data.
- **Mining proteome data:** Within this module, proteome related processes are grouped mainly showing the interaction.
- **Mining transcriptome data:** Transcriptome related processes and data are assembled in this module. Furthermore, this module contains a sub-module dedicated to promoter-scanning.
- **Modelling and Simulation:** The principal processes in the domain of a modelling and simulation are shown here. This module stresses some of the iterative activities that will be supported by the platform.
- **Cell cycle:** As the DIAMONDS platform will be focused on cell cycle regulation, related activities are grouped within this module. Nevertheless, it is important to mention that the other modules also are designed to deal with the cell cycle.
- **Front-end:** The visualisation process is one of the most important processes within the DIAMONDS platform, the front-end of the platform will provide support for it.

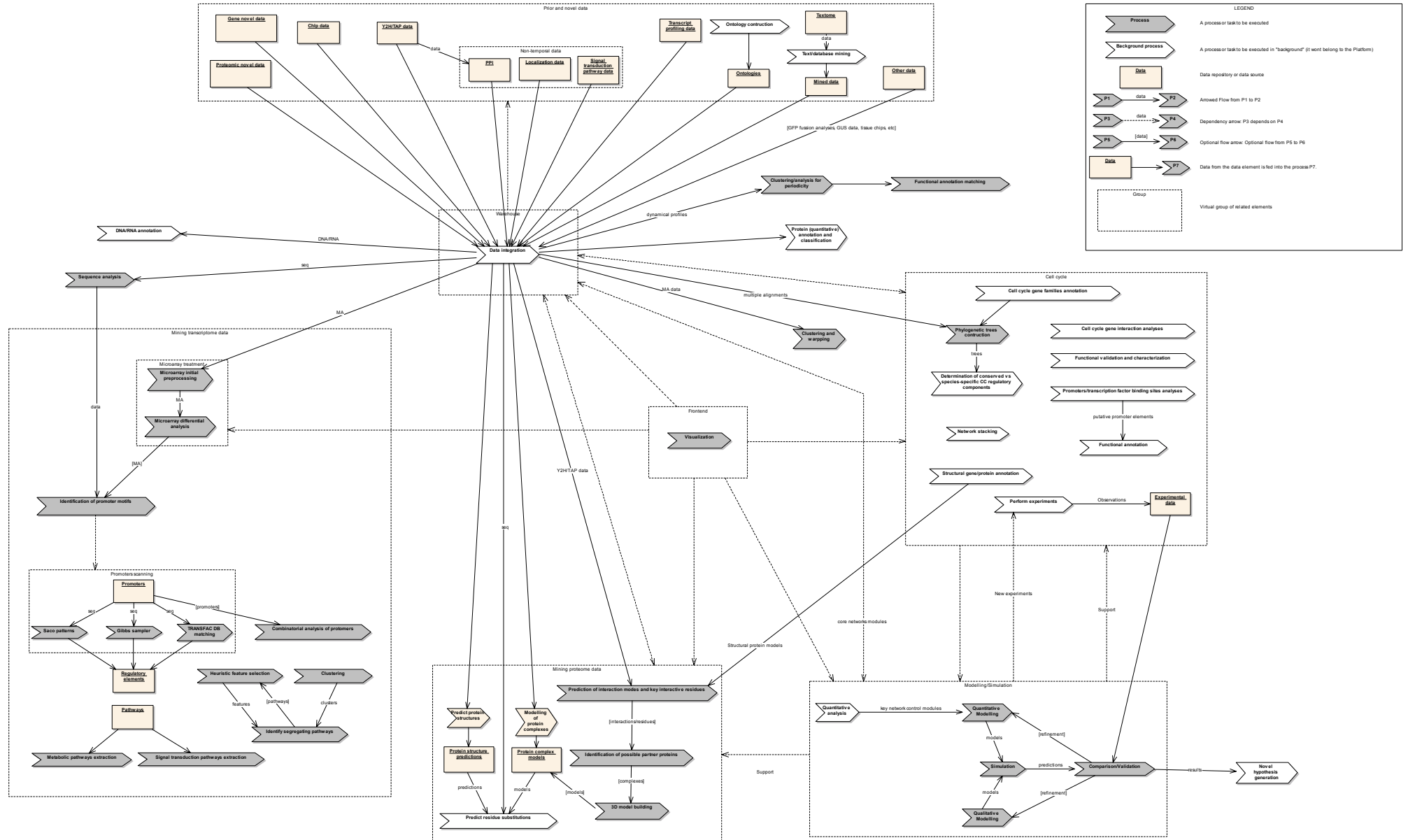


Figure 6. Flow-Like Diagram

2.2 BLOCK DIAGRAM

2.2.1 INTRODUCTION AND NOTATION

The block diagram has been structured using blocks, data and modules, represented with different shapes.

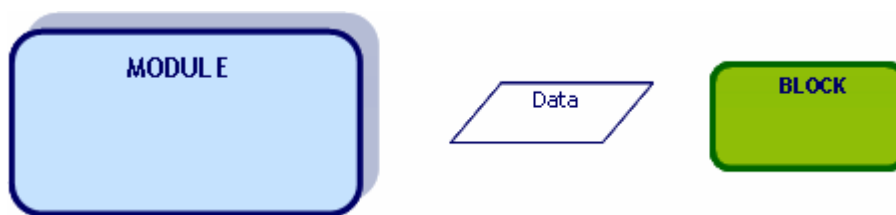


Figure 7. Modules, data sets and blocks

Block: Process to change existing data or to obtain new data. Each block highlights some functionality that, depending on their importance for supporting use-cases, will be incorporated into the DIAMONDS platform. So, each block relates to one particular DIAMONDS tool, preferably by means of web services. This matter is explained in detail in the chapter regarding “Implementation”.

Data: Any type of data managed during the process.

Module: Group of blocks and data. Each module depicts some functionality for the user platform: an option that can be physically chosen and executed in the user interface.

The basic modules defined at the moment are the following ones:

- Uploading wet lab data
- Microarray data analysis
- Promoter Analysis
- Gene Homologies
- Phylogenetic trees
- Protein classification
- Residue Substitutions
- Interaction modes and key interactive residues
- Modelling and simulation

During the project implementation probably more modules will be added, so the platform must be designed to be modular and extendable.

To have a **modular design** is important to make each module independent. So, the user can execute any option of the user interface in any order.

For any option (module) the user can use as input data any data generated previously and stored in GIN-DB (the core DB of the project) or new external data introduced on-line, from external databases or from their personal workspace.

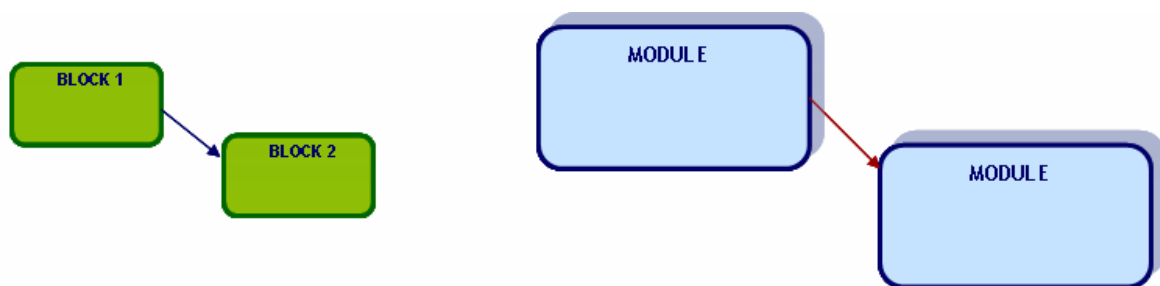


Figure 8. Dependencies

In the block diagram, we also see links between blocks and links between modules, where all the links represent dependencies.

The dependencies existing among the blocks and modules limit their execution order. For example, in figure 9 below, it would be necessary to finish the execution of “block1” before starting the execution of “block2”.

Also the links represent data dependencies for some blocks, and data generation (as a result of the processing developed in the block). For example:



Figure 9. Data dependencies

Each block generates some information, which is going to be stored in the core DB of the system. Also, most of the modules can process data coming from the core DB of the system, as well as external user data. So, any of the modules can act as an entry door for the platform processing.

Alternatively, for many blocks there are different possibilities of input data. As an example:

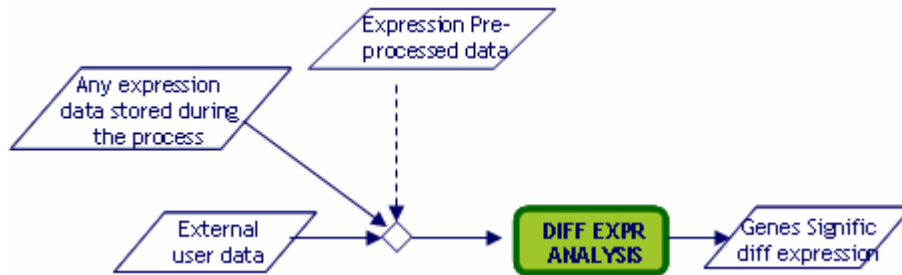


Figure 10. Multiple sources example

In the example of figure 10, the block “Differential Expression Analysis” takes as an input 3 different types of data:

- Pre-processed expression data: obtained in wet-lab experiments.
- Any expression data stored in the DB during the process. For example, we could have executed first the block “Detection of genes periodically expressed”. We obtain as a result a list of genes periodically expressed that we store in GIN-DB or elsewhere and we can extract this list later for further processing.
- External user data. Many blocks of the system accept external user data as an input. This is important to give freedom to the users to execute any option they want with any data they want.

We do not have to have the three types of information to execute the “Differential Expression Analysis” but it is necessary to have at least one.

In fact, the symbol



used in the diagram represents the collection of several alternatives: it means that any of the inputs can exist, but at any moment we will be using just one.

This is extendable to all the “multiple input data” cases in the diagram.

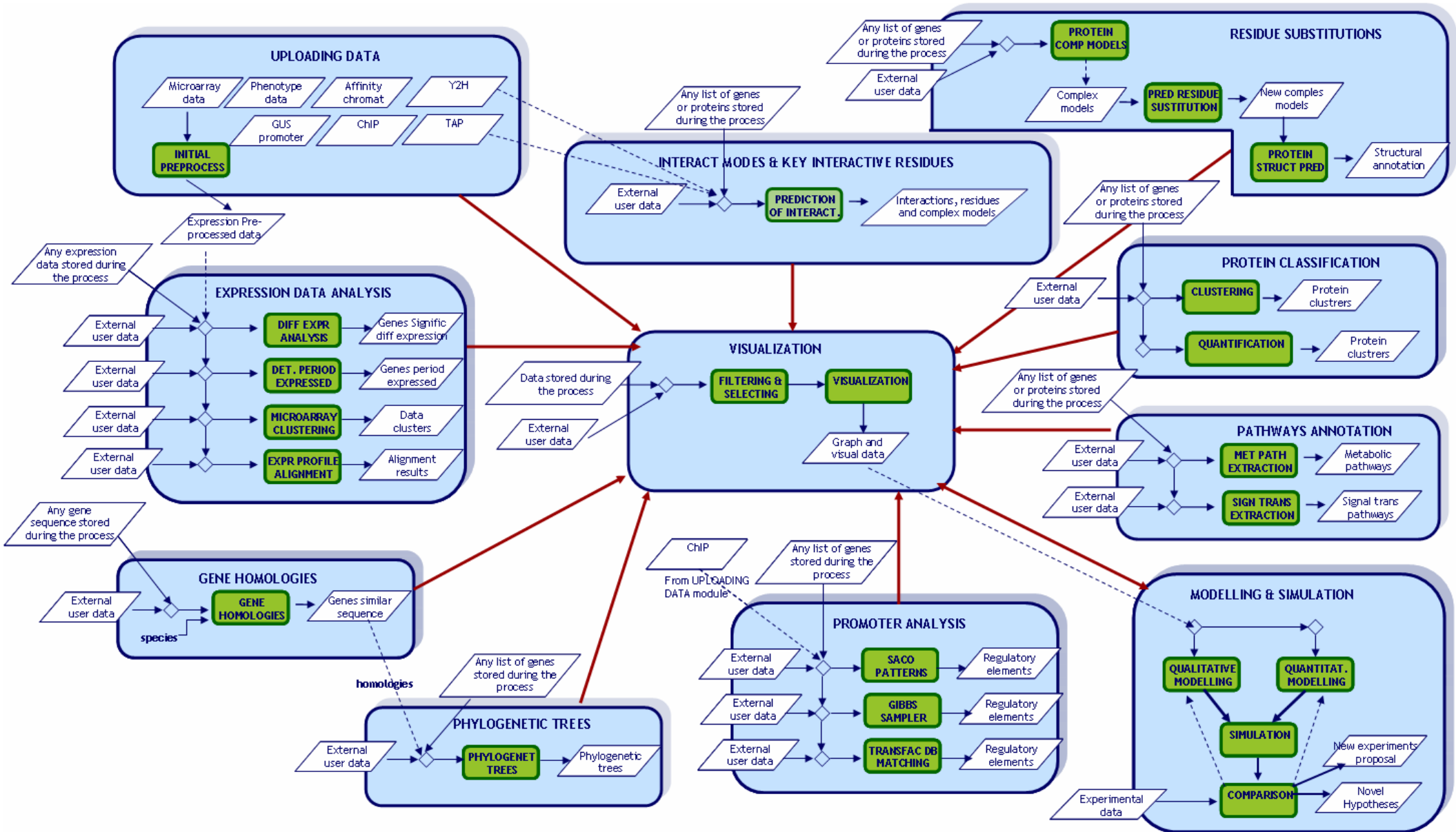


Figure 11. Block Diagram

2.2.2 BLOCK DIAGRAM MODULES

2.2.2.1 UPLOADING DATA

This option lets the user upload data obtained in wet lab experiments. First of all, the user selects the type of data he/she is going to upload from:

- Microarray data
- Y2H
- TAP tag
- Affinity Chromatography
- CHIP on chip
- Phenotype information
- GUS promoter fusions
- Others

In all cases, the user has to indicate the path of the file containing the data. If the user is going to upload some microarray data, he/she also has to indicate the type of chip used: Affymetrix, CATMA, and so forth.

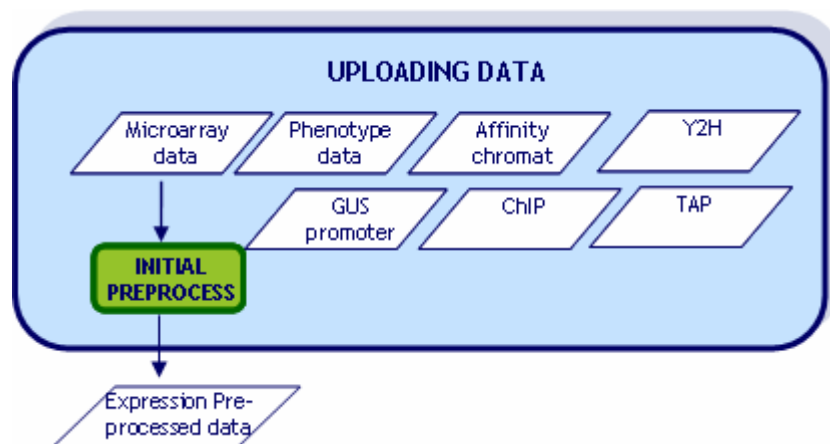


Figure 12. Uploading data

2.2.2.1.1 Initial pre-processing

Objectives

When microarray data are uploaded an initial pre-processing is (semi)automatically performed. The objectives are to accomplish the following tasks:

- a) Normalisation: should use signal-dependent transformation of data (thereby minimising bias effects of e.g. labelling, hybridisation and chip manufacturing)
- b) Hybridisation: may be background corrected
- c) Expression index calculation
- d) Visualisation of chip to chip variation.

Input data

Expression data generated from microarray experiments.

Output data

Pre-processed expression data.

Tools proposed

Gene Publisher, for Affymetrix arrays (<http://www.cbs.dtu.dk/services/GenePublisher/>)
CAGE preprocessing pipeline, for CATMA arrays

2.2.2.2 EXPRESSION DATA ANALYSIS

For expression data obtained in wet lab experiments (Microarray data) different types of “Expression Data analysis” can be performed. Also, we can use as an input to this module pre-processed expression data introduced online by the user.

The user can choose between the different options or execute all of them:

- Differential Expression Analysis
- Detection of genes periodically expressed
- Microarray clustering
- Expression Profile Alignment

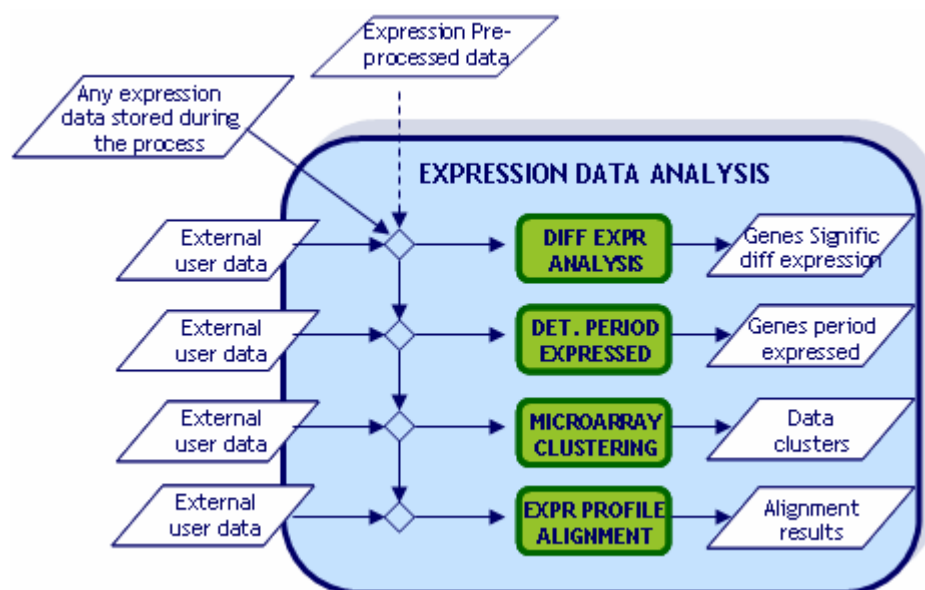


Figure 13. Expression Data Analysis

2.2.2.2.1 Differential Expression Analysis

Objectives:

This process should take into account replicate variation and the Hochberg-Benjamini-corrected **t-test – based p-values**. We finally obtain a list of genes with significant differential expression, accompanied by p-values for a given fold change.

Input data

Pre-processed expression data obtained in “Initial pre-processing” process or introduced on-line by the user; fold change.

Output data

List of genes with significant differential expression, accompanied by p-values.

Tools proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

2.2.2.2.2 Detection of genes periodically expressed

Objectives:

We will detect periodically expressed genes in the time series (transcriptome, proteome) in order to produce a list of genes with significant, periodic expression. Possible methods:

- a) Fourier transformation
- b) Principle component analysis
- c) Single pulse models

Input data

Pre-processed expression data obtained via the “Initial pre-processing” process or introduced on-line by the user.

Output data

List of genes periodically expressed

Tools proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

2.2.2.2.3 Microarray clustering

Objectives:

Different clustering algorithms can be implemented and applied to expression data: similarity search, hierarchical clustering, k-means/k-medoids clustering, flat-flat clustering comparison, hier-flat clustering comparison, signature algorithm.

Input data

Pre-processed expression data obtained via the “Initial pre-processing” process or introduced on-line by the user.

Output data
Gene Clusters

Tools proposed:
Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

2.2.2.2.4 Expression Profile Alignment

Objectives:
We can choose between two tasks:

a) Template Matching

The *template matching mode* allows mining gene expression time series for patterns that fit best a template expression profile provided by the user. A gene expression time series file is supplied as an input having the gene names in the leftmost column. The interface of the template matching mode enables the user to select the expression profile of a gene of interest as a template (by simply scrolling up and down the list of gene names given in the left part of the window). The expression profile of the selected gene is then plotted.

b) Alignment mode

The *alignment mode* allows finding the best time alignment between two sets of gene expression time series. The purpose of the alignment mode is twofold:

1) to study the behaviour of a particular gene set in different experimental conditions, as for instance cell cycle expression data generated with different synchronisation techniques;

2) to compare the time trajectories of different groups of potentially interacting genes in the same time series data.

Input data

Pre-processed expression data obtained via the “Initial pre-processing” process or introduced on-line by the user. It is specifically suited for periodically expressed genes in time-series data.

Output data

For the template matching mode: a list of genes with an expression profile matching the expression profile of a gene of interest (both visually as well as written to file).

For the alignment mode: the optimal alignment of the two given time series is outputted with respect to a newly generated common time axis (both visually as well as written to file).

Tool proposed:

GenTxWarper (<http://www.psb.ugent.be/cbd/papers/gentxwarper>)

2.2.2.3 GENE HOMOLOGIES

From a gene sequence we could infer homologies in other species or in some other genes in the same species. This module is structured in a single block.

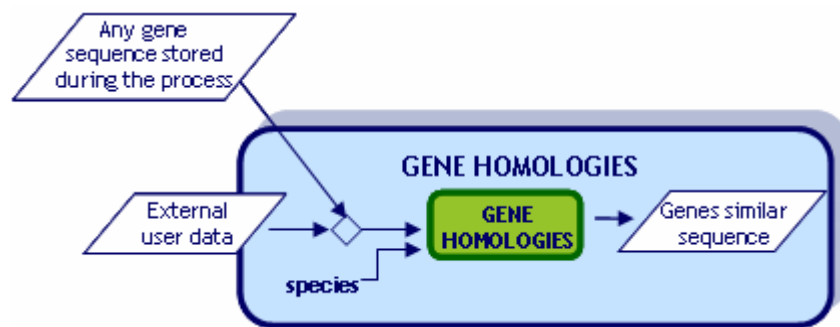


Figure 14. Gene Homologies

2.2.2.3.1 Gene Homologies

Objectives

Mentioned above.

Input data

Any gene sequence stored during the process, or external user data

Output data

Genes with similar sequence

Tools proposed:

Blast.

Tools from the BEG division of PSB

2.2.2.4 PHYLOGENETIC TREES

Since phylogenetic inference is complicated and often far from self-evident, different approaches for inferring molecular phylogenies will be applied to reconstruct the evolutionary relationships between cell cycle genes.

Trees obtained are used for determination of conserved vs species-specific CC regulatory components. If tools for phylogenetic tree inference will be made publicly available via the platform, then of course the same analyses can be performed for any set of genes (and thus not only cell cycle related genes).

This module is composed of a single process, called the same way.

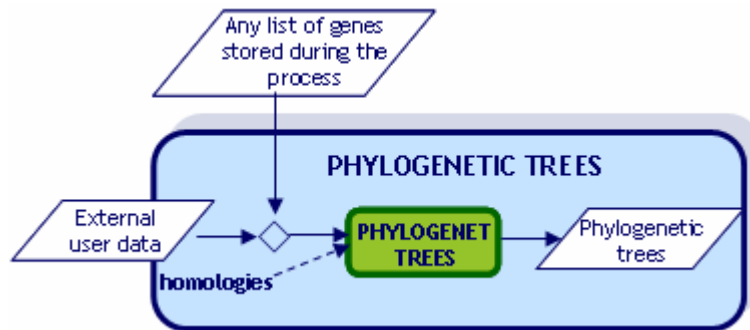


Figure 15. Phylogenetic trees construction

2.2.2.4.1 Phylogenetic trees

Objectives

Several well established approaches for inferring molecular phylogenies will be applied:

- a) Pairwise Distance methods
- b) Maximum Likelihood
- c) Maximum Parsimony
- d) Bayesian statistics

Novel methods **accounting for saturation of the amino acid levels** will also be applied, as well as methods that **correct for unequal rates of evolution** in different branches of the tree.

Input data

This process needs as necessary input a list of genes, which can be provided by GIN-DB (with genes stored at any moment of the platform use) or can be introduced online by the user.

Also, this process can accept gene homologies as an optional input, obtained in the process with the same name that may help in the construction of phylogenetic trees.

Output data

Molecular phylogenies.

Tools proposed

None, at the moment

2.2.2.5 PROMOTER ANALYSIS

Upstream sequences will be scanned for known and unknown **regulatory elements** using three different methods that use different strategies:

- a) Saco Patterns
- b) Gibbs Sampler
- c) TRANSFAC database

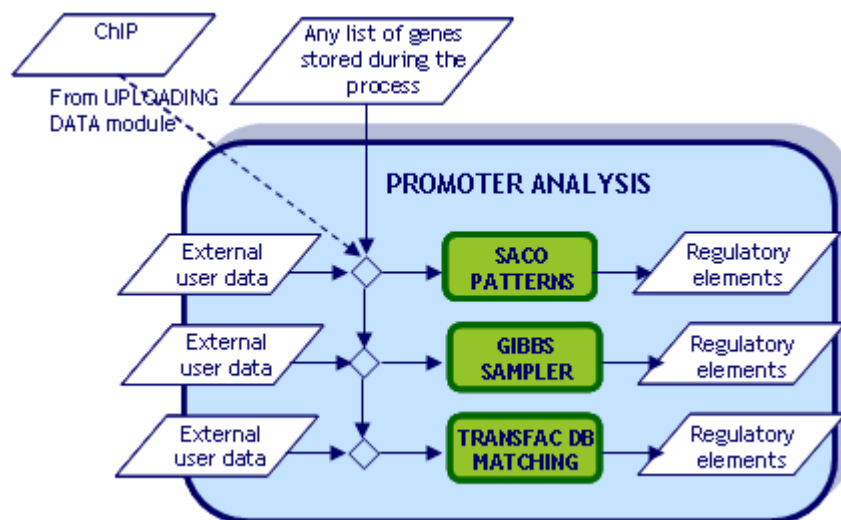


Figure 16. Promoter Analysis

These methods are complementary, so the user could choose all of them and not only one.

2.2.2.5.1 Saco Patterns

Objectives

Saco Patterns method identifies patterns significantly over-represented in the upstream regions relative to a background set of upstream regions from the same organism.

Input data

Any gene or list of genes stored in GIN-DB previously or genes/list of genes introduced by the user as an external input.

Output data

Regulatory elements

Tools proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

2.2.2.5.2 Gibbs Sampler

Objectives

Gibbs Sampler: looks for over-representation of degenerate patterns.

Input data

Any gene or list of genes stored in GIN-DB previously or genes/list of genes introduced by the user as an external input.

Output data

Regulatory elements

Tools proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

2.2.2.5.3 TRANSFAC DB matching

Objectives

The known transcription factor binding sites in the TRANSFAC database will be matched against the same upstream regions.

Input data

Any gene or list of genes stored in GIN-DB previously or genes/list of genes introduced by the user as an external input.

Output data

Regulatory elements

Tools proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

2.2.2.6 PROTEIN CLASSIFICATION

Protein annotation provides a basis for protein classification and function inference techniques. This will allow to classify proteins into sets based on function similarity.

The considerations used for classification are diverse: protein homology, domain identification, phylogenetic considerations, structural similarity, etc...

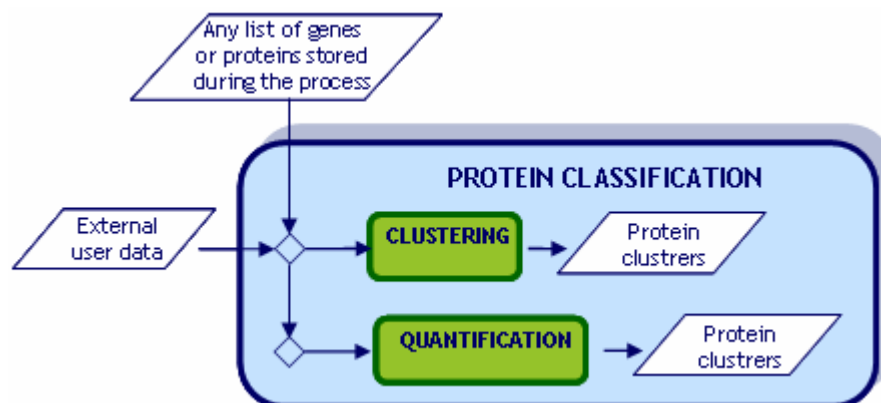


Figure 17. Protein classification

2.2.2.6.1 Clustering

Objectives

Clustering can be performed using different techniques:

Graph-based clustering: The input is a protein set and a selection of one or more annotation types. The system displays the full protein-keyword relations between the proteins of the set and the keywords of the selected types. This is displayed as an intersection-inclusion Directed Acyclic Graph (DAG). An intersection-inclusion DAG is a hierarchical graph that describes all intersection and inclusion relationships between given sets. In our case, these sets would be protein sets, each protein set sharing a unique combination of keywords. This allows presentation of the whole collection of protein-keyword relations without loss of the initial information.

Hierarchical clustering: The clustering process is based on an all-against-all BLAST similarity search. The similarities' *E*-score is used to perform a continuous bottom-up clustering process by joining the two most similar protein clusters at each step, resulting in a hierarchy of protein clusters at various degrees of biological granularity. This hierarchy is structured as a collection of trees, in which the root clusters contain all the proteins of the tree and the rest of the clusters represent subdivisions of the proteins into smaller groups.

Input data

Any list of proteins obtained during the process, or external user data.

Optionally, some data generated before referred to as homologies, domain identification, phylogenetic considerations, etc.

Output data

Protein clusters (grouped by functionality).

Tools proposed:

Pandora (<http://www.pandora.cs.huji.ac.il>)

Protonet (<http://www.protonet.cs.huji.ac.il>)

2.2.2.6.2 Quantification

Objectives

In some cases, you might want to inspect biological properties that may be quantitative, not binary, so it is important to have a tool which allows the introduction of "data properties".

If the data property is binary, it is just treated as an extra annotation. However, if it is quantitative, it is dealt with differently. After constructing the graph based on the binary annotations, each node will have a distribution of the quantitative property on it (recall that each node represents a set of proteins). The distribution of the property on the nodes is visualised by using colour histograms for each node. This allows you to easily detect nodes whose proteins have an interesting distribution. The execution of this block is optional (not necessary after executing Clustering), and it can only be executed if the technique used for clustering has been "graph-based".

Input data

Graph-based cluster

Data properties

Output data

Data quantified

Tools proposed:

Pandora (<http://www.pandora.cs.huji.ac.il>)

2.2.2.7 INTERACTION MODES AND KEY INTERACTIVE RESIDUES

It is also necessary to have available a process for the prediction of interaction modes and key interactive residues, using the Y2H and TAP data generated within the project, complemented with public data on protein interactions.

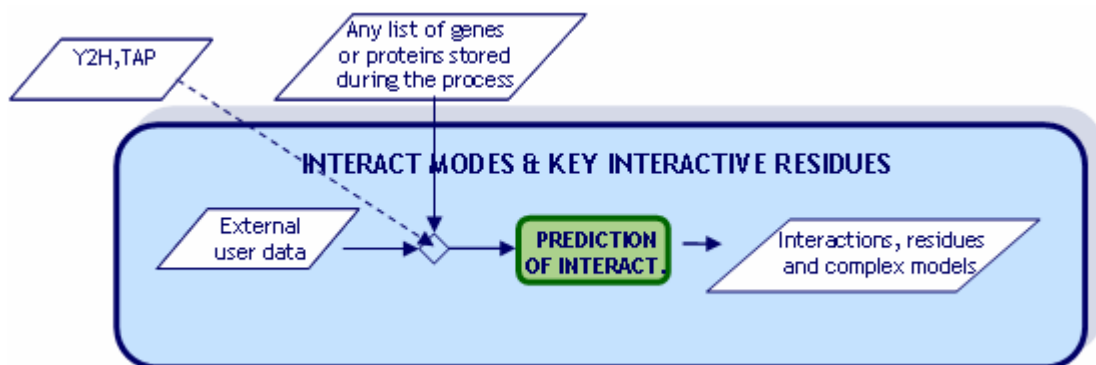


Figure 18. Interaction modes and Key interactive residues

For that purposes, we will use the well-organised European database on protein interactions -INTACT- built on the HUPO standard. We will combine modelling, docking and sequence analysis and propose models of interaction that can be manipulated and challenged in subsequent experiments

This module is composed by a single process, called the same way.

2.2.2.7.1 Prediction of interaction modes and key interactive residues

Objectives

Identification of possible partner proteins and the building of three-dimensional models of the corresponding complexes, their domain decomposition, and the careful construction of corresponding alignments between target and sequences to model.

For this, we will use structural models of key CC proteins to map the binding characteristics differentiating 'binders' from 'non-binders'. Subsequently, we will identify the regions of interaction and the key residues responsible for the specificity of interaction, as a prelude to targeted deregulation (or therapeutic perturbation) of the cell cycle.

If this tool will be made publicly available via the platform, then of course the same analysis can be performed for any set of genes (and thus not only cell cycle related genes).

Input data

Interaction data from Y2H and TAP experiments, any list of genes or proteins generated during the process, or external user data. Interaction data will also be extracted from the INTACT database.

Output data

Interactions, residues and complex models

Tools proposed:

None, at the moment.

2.2.2.8 RESIDUE SUBSTITUTIONS

A module will be available to predict residue substitutions to modulate (or change) the binding specificity and therefore change the regulation of the interaction network. This type of work will support our network validation effort.



Figure 19. Residue substitutions

2.2.2.8.1 Residue substitution effect prediction

Objectives

Changing some residues in protein complex models can change the protein function or structure.

This block will generate a new complex model changing some residues. The new complex model will be tested for structure prediction in block “Protein Structure prediction”.

Input data

- Protein complex models. We can get those models from different sources:
 - Complex models introduced on-line by the user
 - Complex models obtained from block “Protein complex models”. In this block the user must have introduced previously a list of proteins, to obtain protein complex models.
 - Complex models from module “Interaction Modes and key interactive residues”.
- Residues (residues to change and residues to add).

Output data

New protein complex models

Tools proposed:

None, at the moment. In fact, it is not really clear whether this should be provided as a tool for wet-lab biologists or used solely by expert users.

2.2.2.8.2 Protein complex models

Objectives

Generate protein complex models. If the user wants to change some residue in a complex model but doesn't have a model, but just a list of proteins, he/she could introduce the list in this block to obtain a protein complex model. This complex model will be used as an input for “Residue substitutions effect prediction”.

Input data

Any list of proteins generated during the process, or external user data. Also, it could be recommendable (but not necessary) to have available models obtained as a result of the processing done in “Prediction of interaction modes and key interactive residues”.

Output data

Protein complex models

Tools proposed:

None, at the moment. In fact, it is not really clear whether this should be provided as a tool.

2.2.2.8.3 Protein structure predictionObjectives

Predict proteins structure, from a protein complex model.

Input data

Complex models generated in “Residue substitutions effect prediction”.

Output data

Protein structure predictions: structural annotation for the new complex

Tools proposed:

None, at the moment. In fact, it is not really clear whether this should be provided as a tool.

2.2.2.9 PATHWAYS ANNOTATION

This module looks for the annotation of information related to pathways. It accepts as an input any gene or list of genes (obtained during the project or introduced online by the user).

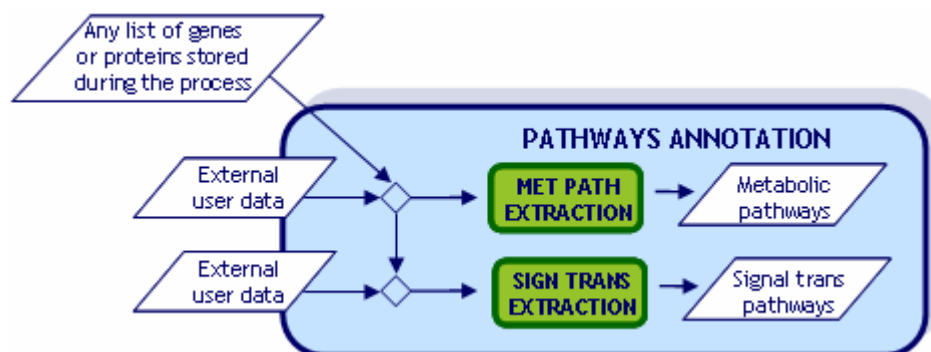


Figure 20. Pathways annotation

2.2.2.9.1 Metabolic Pathways Extraction

Objectives

Genes are matched to the KEGG (Kyoto Encyclopedia of Genes and Genomes) description of known cellular pathways (<http://www.genome.ad.jp>).

Input data

The input data must be a gene or list of genes, but as source we can consider two different possibilities:

- Any gene or list of genes generated during the process
- Gene or list of genes introduced on-line by the user

In any case, it is recommended to use genes periodically expressed, like those obtained in the process "Detection of genes periodically expressed".

Output data

Metabolic pathways where the genes are involved. In case Gene Publisher would be used, then only one pathways will be shown for genes belonging to multiple metabolic pathways.

Tool proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

2.2.2.9.2 Signal Transduction Pathways Extraction

Objectives

Genes are matched to the TRANSPATH database for signal transduction. If genes match more than one pathway, only one pathway is shown.

Input data

The input data must be a gene or list of genes, but as source we can consider two different possibilities:

- Any gene or list of genes generated during the process
- Gene or list of genes introduced on-line by the user

In any case, it is recommended to use genes periodically expressed, like those obtained in the process "Detection of genes periodically expressed".

Output data

Signal transduction pathways where the genes are involved. In case Gene Publisher would be used, then only one pathway will be shown for genes belonging to multiple signal transduction pathways.

Tool proposed:

Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

2.2.2.10 VISUALISATION

This module should allow the visualisation of:

- Any data stored by the user during the whole process
- Any other data (not user's property) stored in GIN-DB
- Some new external user data

The amount of information stored in GIN-DB will be vast, therefore there should be some mechanisms for filtering to make it easy for the user to select the information to visualise.

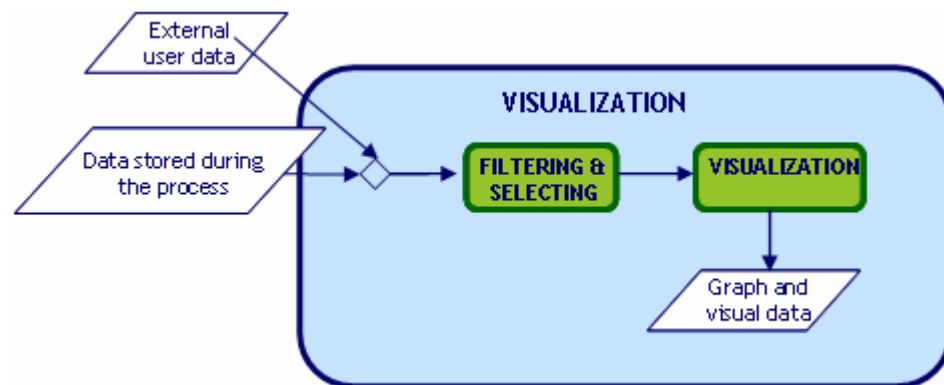


Figure 21. Visualisation

2.2.2.11 MODELLING AND SIMULATION

The user can insert gene network models and simulate their dynamical behaviour using different techniques/mathematical frameworks: Boolean modelling, ODE, PLDE, stochastic equations, hybrid modelling, PetriNets, etc.

As a first step the user will visualise all data (stored in GIN-DB during the process) relevant for the designing the model. All the information and knowledge available will be used by the scientists to create the appropriate gene network model and simulate it.

After simulating, the obtained results will be compared with the experimental results:

- if simulation results match experimental data, we can infer that the proposed model was correct.
- If not, some modifications must be made in the model design. In this case new hypotheses must be generated leading to new experiments to validate those hypotheses.

All this processing should make the platform not only descriptive but also predictive.

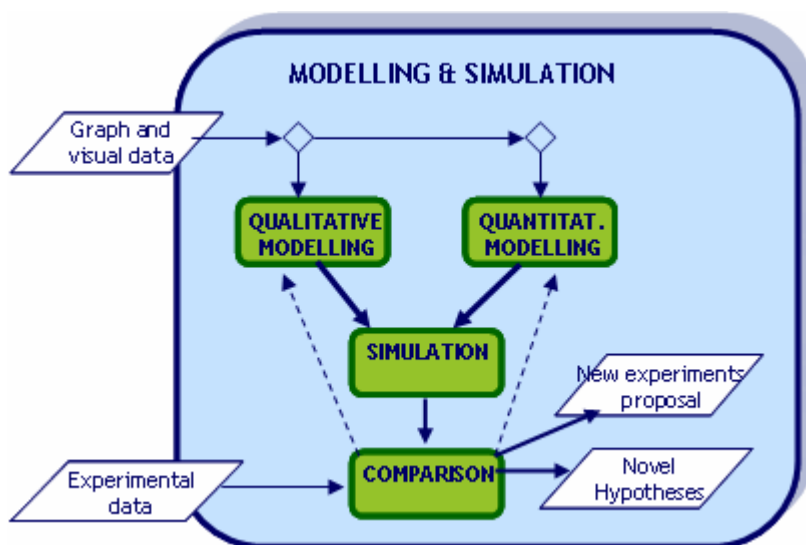


Figure 22. Modelling and Simulation

2.2.2.11.1 Qualitative modelling, quantitative modelling and simulation

Objectives

- Construct qualitative models: translate all relevant interactions into qualitative effects such as activation and inhibition of the different regulatory components. Already at this stage, the application of graph-theoretic algorithms will generate novel insight into specific network properties, either in the context of a model referring to a single species, or in the context of interspecies comparisons.
- Construct quantitative models: the most interesting qualitative models will be further processed to generate quantitative properties of these regulatory models by using methods from the field of non-linear science (including bifurcation analysis, focusing on dynamical properties such as simple or complex periodic behaviour, multi-stability, hysteresis, etc...)

We will develop qualitative and/or quantitative dynamical models for the most important regulatory modules. For the SIM-Plex tool, one of the proposed tools, an approach is currently implemented that would allow the scientist to design and **parameterise** the models, as well as to **simulate** their behaviour for normal or perturbed conditions.

Input data for the simulation tools

Qualitative and quantitative models will be built on-line by the user, based on the knowledge and information visualised by the platform. Eventually the platform database should store validated models, which can then be retrieved for simulation purposes.

Output data for the simulation tools

Both visual and textual simulation results, format to be defined (GXL, VSG).

Tools proposed:GIN-SIM (<http://www.esil.univ-mrs.fr/~chaouiya/GINsim>)SIM-Plex Simulator (<http://www.psb.ugent.be/cbd/papers/sim-plex>)**2.2.2.11.2 Comparison**Objectives

As mentioned at the introduction of this module, the results obtained in the simulation will be **compared** with previous experimental results:

- if simulation results and experimental results match, we can infer that the proposed model was correct.
- If not, some modifications must be made in the model design. In this case new hypotheses should be generated and new experiments should be proposed to validate those hypotheses.

Input data

Simulation results and experimental results. The experimental results could also be fetched from the database by the tool itself.

Output data

Novel hypotheses and new experiment proposals.

Tools proposed:

None, at the moment. It is still unclear whether it is actually feasible to provide this type of functionality as a tool. In case this is possible, it will have to be decided if the functionality must be provided by a separate tool or become integrated within the simulation tools.

2.2.3 BACKGROUND PROCESSES

2.2.3.1 ANNOTATION MODULE

The annotation module is going to be executed in the background. Once new data is entered in the system, the annotation task automatically is going to be performed in a transparent way for the user. The entered data could be:

- all wet lab data uploaded,
- all sequences, genes, proteins, interactions, etc. introduced online by the user.

This module is not shown in the blocks diagram since it will not be part of the user-interactive DIAMONDS platform.

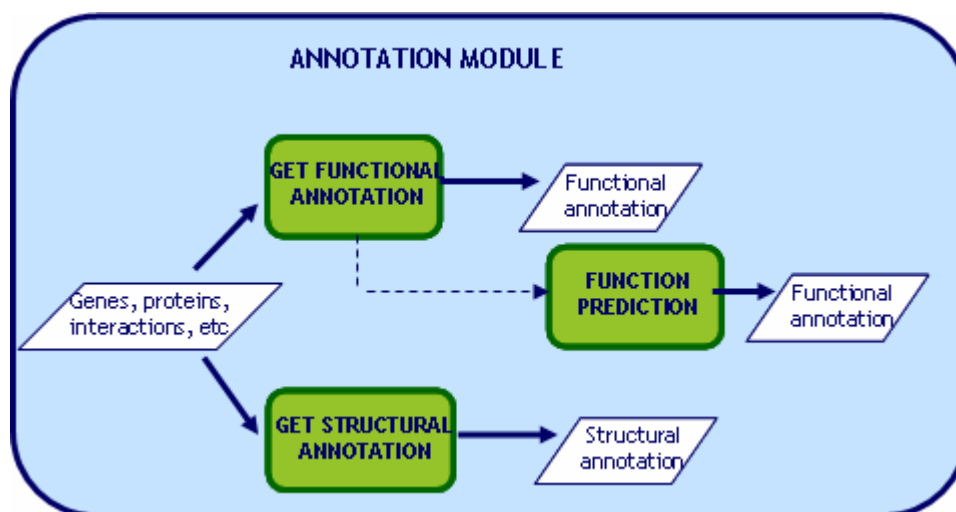


Figure 23. Annotation module

2.2.3.1.1 Functional annotation

Objectives

Gene Ontology annotation. Gene Ontology provides a basic structured description of a gene's function. GO is not a database of gene sequences, nor a catalog of gene products. The three organizing principles of GO are **molecular function**, **biological process** and **cellular component**.

Input data

All new data introduced in the system

Output data

Functional GO annotations in GIN-DB

Tools proposed:Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)Expression Profiler (<http://www.ebi.ac.uk/expressionprofiler>)

Other that may be developed in the context of DIAMONDS, or in collaboration with the GO consortium

2.2.3.1.2 Structural annotationObjectives

Comprises the correct determination of a gene's structure and the elements describing the overall gene model (exon, intron, coding sequence, UTR, promoter, etc.). Probably we can get those data from TIGR, GeneBank or NCB. Actually NCBI also provides codes that allow the integration of genes from multiple organisms.

Input data

Genome sequence data, protein, transcript information

Output data

Structural annotations in GIN-DB

Tools proposed:

Glimmer, EuGene

2.2.3.1.3 Function PredictionObjectives

For those genes where a gene ontology number has not been assigned and the function has not been inferred by homology to another protein, an attempt is made at predicting the function using the **ProtFun** method.

The ProtFun methods predicts the function not based on homology, but based on properties of the protein sequence as well as predicted features such as post-translational modification.

Input data

Genes (input data for "Functional Annotation" block) lacking Gene Ontology annotation and significant sequence homology.

Output data

New functional annotations

Tools proposed:Gene Publisher (<http://www.cbs.dtu.dk/services/GenePublisher>)

2.2.3.2 TEXT MINING and DATA CURATION

TEXT MINING and uploading/curation are going to be facilitated by PubGene and CSIC.

PubGene is going to extract relevant information related to cell cycle mined from relevant literature, PubGene and/or CSIC will provide functionality to link expression clusters to literature data.

As a part of the text mining functionality they will have to develop cell cycle filters / indexing for the various organisms involved in the DIAMONDS project. The extracted literature data sets will contain information about genes, proteins, chemical and compounds, Mesh-terms, GO-terms and mutation data. The information will be mined from a great variety data sources: Medline, Entrez gene, Homologene, Locuslink, Unigene, Ensembl, TIGR, GoldenPath, RefSeq/GenBank, UniProt, PIR/NREF, ENZYME, etc.

In addition, Pub Gene will make a curation tool that the various participants with expert knowledge within the different research fields should use to upload the first corpus of literature data.

2.3 *DIAGRAM DEVELOPMENT*

The models/diagrams (flow-like diagram and block diagram) have been developed using Enterprise Architect™.

Two XMI[3] files were obtained from the diagrams in order to be distributed to all the DIAMONDS partners and to be adapted or modified so that a general consensus could be reached. However, the diagrams will likely be iteratively adapted according to evolving needs, novel use-cases and practicality of implementation. New and complementary diagrams will appear as the project unfolds.

XMI (XML Metadata Interchange) is an interchange format for sharing formats using XML.

All diagrams are available in several formats in this URL:

<http://www.sbcellcycle.org/partners/>

- FlowLikeDiagram.png, only the diagram in PNG format.
- FlowLikeDiagram.xml, the complete model in XMI format.
- FlowLikeDiagram.zip, the complete Enterprise Architect™ project.

- Block Diagram.jpeg, only the diagram in JPEG format.
- Block Diagram.xml, the complete model in XMI format.
- Block Diagram.zip, the complete Enterprise Architect™ project.

3 DIAMONDS WORKBENCH

The platform will support many or all of the functionalities stated in the block diagram.

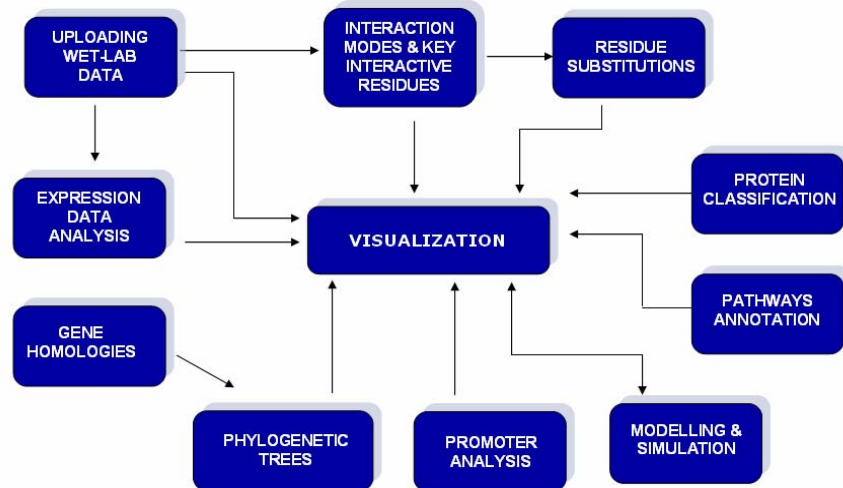


Figure 24. Modules involved in the block diagram

In this sense it is important to link the modules in the block diagram with the graphical user interface: we have to understand each one of the modules included in the diagram as a user option in the graphical user interface.

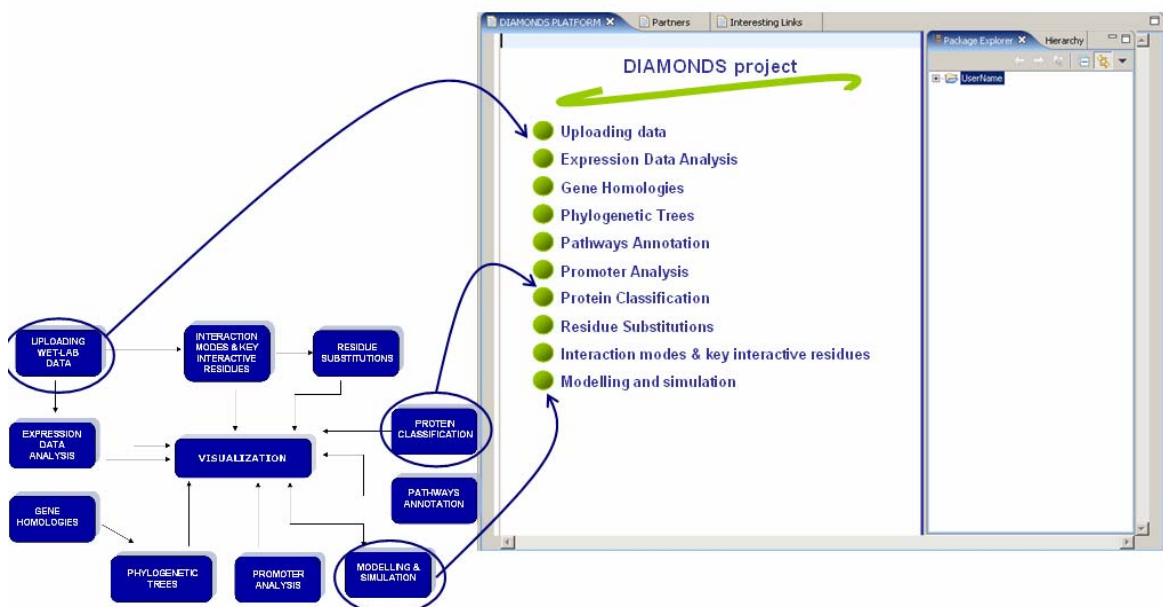


Figure 25. Relationship between modules and user options

The graphical user interface design showed in figure 25 is just provided as illustration, the platform GUI will be assessed as a 9-month milestone. The figure only pretends to give an idea of the user options that may appear in the final graphical user interface.

All modules are independent, and this is a very important characteristic to make the platform flexible and extendable. So, if in any moment of the development cycle the consortium decides to add or remove any user option in the graphical user interface, we will only have to add or remove a specific module in the block diagram.

Another important characteristic of the platform is that all user information should be available at any moment. For that purpose, when a user gets into the platform introducing login and password, then a folder structure will be automatically created where the father folder should have the same name as the login user.

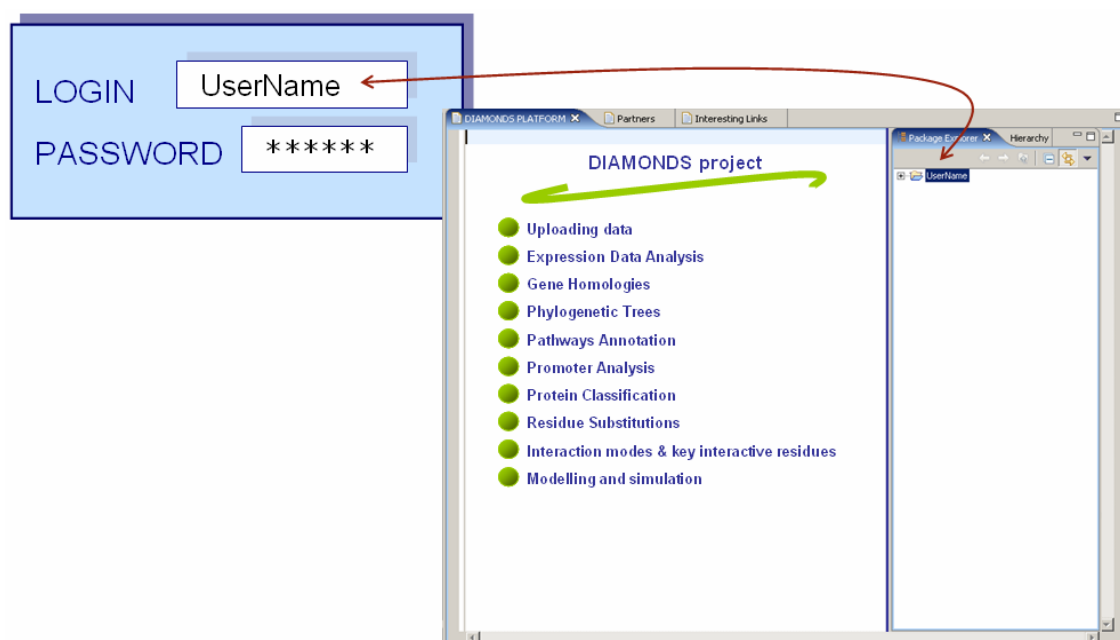


Figure 26. User directory

In the stated folder structure, the user will have visible and available all the input files uploaded to the workbench, all the processes performed on those files, as well as the results obtained.

The complete set of files must be organised to make it easy for the user to realise which files are inputs and which are results. The structure should also allow to group sets of processes.

In this sense, we make the following proposal for workbench operation:

- a) For each input file we will have available all the processes done and the output files obtained with those processes

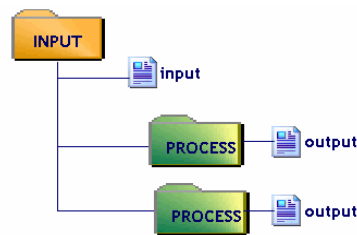


Figure 27. Input and Output structure

- b) The general structure could be based on projects and experiments. Each experiment could be composed by a set of processes grouped by their input file.

The platform will generate a history file which includes the process flow followed by a certain file. The history file is really significant when we reuse an output file as an input for another process.

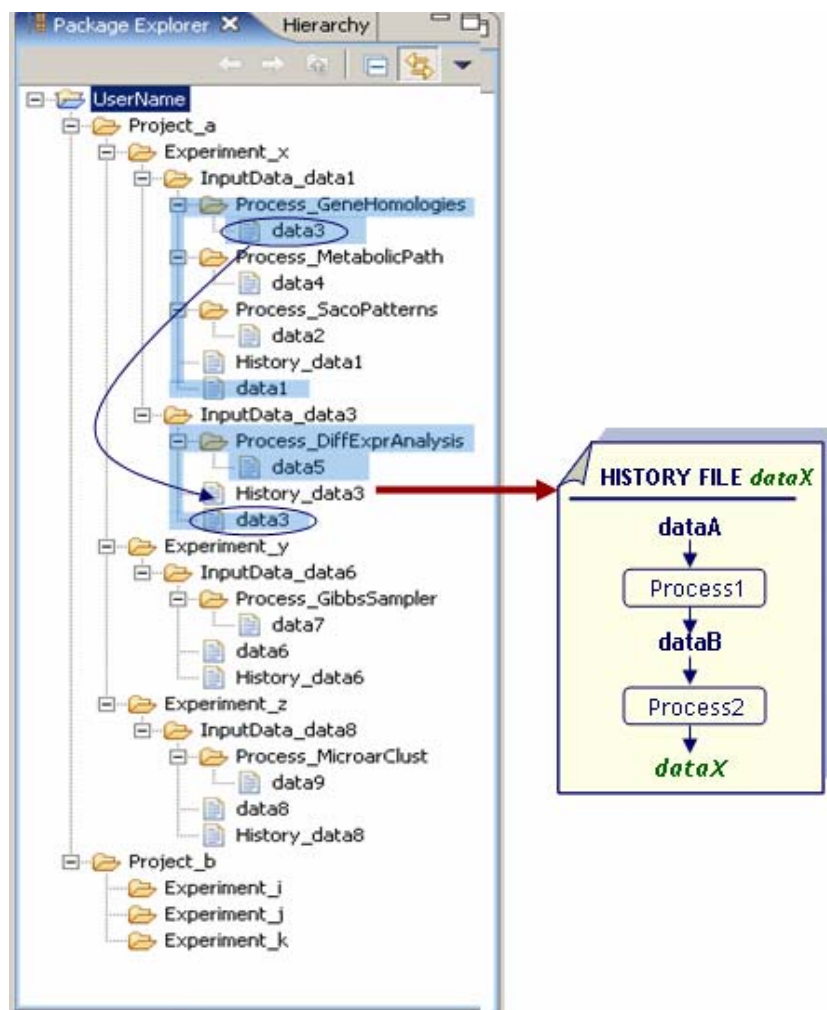


Figure 28. Workbench example and history file

4 INTEGRATION TECHNOLOGY: WEB SERVICES

4.1 INTRODUCTION AND ADVANTAGES

A Web Service[11] is a programmable application logic accessible using standard Internet protocols. Web Services offer a single uniform method for application integration through the Internet. They provide a model for accessing SW systems over the web by pointing to their web address, while their public interfaces and bindings are defined and described using an XML standard format.

Web Service is the basic technology proposed for integration in this project because it allows the communication between systems with no common design aspects. In DIAMONDS platform we will have several tools designed and implemented using different programming languages and different deployment modes.

Web Services represent functionality that can be easily reused without taking into account how the service is implemented, so in this heterogeneous environment, web services could be used as a the basic technology for tools integration.

It is also important to emphasise that every tool has been developed in a different location. Using web services as integration technology, the services could be stored and supported by their own authors in their own location, because the services are accessed via ubiquitous web protocols (like HTTP) and using universally accepted data formats (like XML).

To sum up, the principal advantages to use web services as integration technology for DIAMONDS are:

- A web service is an application or information resource accessible over the web. Web services communicate using platform-independent and language-neutral web protocols. These web protocols ensure easy integration of heterogeneous environments.
- A web service provides an interface that can be called from another program. This application-to-application programming interface can be invoked from any type of application client or service.
- A web service is registered and can be located through a web service registry. The registry enables service consumers to find services that match their needs.
- Web services support loosely coupled connections between systems. Web services communicate by passing messages to each other. The web service interface adds a layer of abstraction to the environment that makes the connections flexible and adaptable.

- Any type of application can be offered as a Web service.
- Authors themselves are people in charge of supporting the tools. This fact allows a continuous improvement of the service offered by those tools.

However, it will be left to the discretion of the tool builders to provide their tools as a web service, or to consider different alternatives in order to choose the integration technology which fits best with every tool / partner's competence.

4.2 WEB SERVICES TECHNOLOGIES

Web Services can be developed using any programming language and can be deployed on any platform. Web Services can communicate because they all speak the same language: the Extensible Markup Language (XML).

Web Services use XML to describe their interfaces and to encode their messages. XML-based Web Services communicate over standard Web protocols using XML interfaces and XML messages, which any application can interpret.

But XML by itself does not ensure effortless communication. The applications need standard formats and protocols that allow them to properly interpret the XML. Three XML-based technologies have emerged as standards for Web Services:

- Simple Object Access Protocol (SOAP) defines a standard communications protocol for Web Services.
- Web Services Description Language (WSDL) defines a standard mechanism to describe a Web Service.
- Universal Description, Discovery and Integration (UDDI) provides a standard mechanism to register and discover Web Services.

Figure 29 shows how these technologies relate to each other.

When a service provider wants to make the service available to service consumers, he describes the service using WSDL and registers the service in a UDDI registry. The UDDI registry will then maintain pointers to the WSDL description and to the service. When a service consumer wants to use a service, he queries the UDDI registry to find a service that matches his needs and obtains the WSDL description of the service, as well as the access point of the service. The service consumer uses the WSDL description to construct a SOAP message with which to communicate with the service.

SOAP is an extensible XML messaging protocol that forms the foundation for Web Services. SOAP provides a simple and consistent mechanism that allows one application to send an XML message to another application.

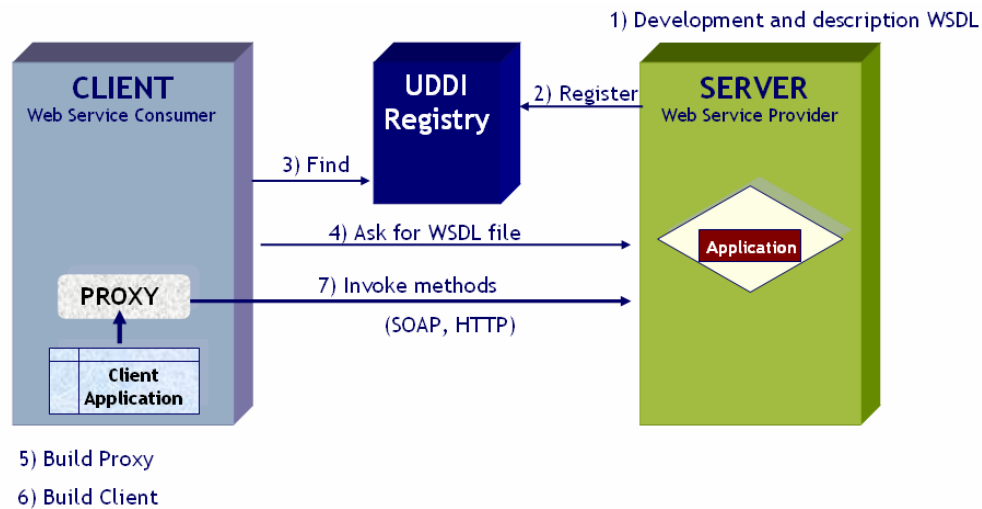


Figure 29. Web services usual operation

Fundamentally, SOAP supports peer-to-peer communications. A SOAP message is a one-way transmission from a SOAP sender to a SOAP receiver, and any application can participate in an exchange as either a SOAP sender or a SOAP receiver. SOAP messages may be combined to support many communication behaviours, including request/response, solicit response, and notification.

SOAP was first developed in late 1999 by DevelopMentor, Microsoft, and UserLand as a Windows-specific XML-based remote procedure call (RPC) protocol. In early 2000 Lotus and IBM joined the effort and helped produce an open, extensible version of the specification that is both platform- and language-neutral.

SOAP consists of four parts:

- **SOAP Envelope.** The SOAP envelope provides a mechanism to identify the contents of a message and to explain how the message should be processed. A SOAP envelope includes a SOAP header and a SOAP body. The SOAP header provides an extensible mechanism to supply directive or control information about the message.
- **SOAP Body.** The SOAP body contains the payload that is being sent in the SOAP message.
- **SOAP Transport Binding Framework.** SOAP 1.1 defines bindings for HTTP and the HTTP Extension Framework. SOAP 1.2 defines a default binding for HTTP (not the extension framework) and provides for other bindings such as SMTP, JMS and others.
- **SOAP Serialisation Framework.** All data passed through SOAP messages are encoded using XML. Data can be passed as a literal XML document that validates against some XML Schema.

4.3 ALTERNATIVES

Even though web services provide a good integration mechanism for platform development, we have to consider some disadvantages of web services:

- To work using web services some stages may have been developed: we have to encapsulate the tool/application into a web service, generate the WSDL description, as well as publishing it in the UDDI registry. Also is necessary to manage SOAP messages for data interchange. These tasks could not be trivial for some applications.
- Another disadvantage is that web services are usually *stateless* (even though, in theory, there is nothing in the Web Services Architecture that says they can't be stateful). This means that the Web service can't "remember" information, or *keep state*, from one invocation to another.

Even though web services are stateless, we could manage them to generate *statefulness*, in some cases, but is not the natural operating way. So, for applications or services that imply an important interoperation between the tool and the user, maybe web services are not the best option for integration. This fact, complemented with the complexity of generating the web service environment, could induce us to use another integration technology.

One alternative to web services for tools integration is the generation and use of **plug-ins**.

A plug-in is an application that adds features to an existing program. The idea behind plug-ins is that a small piece of software is loaded into memory by the larger program, adding a new feature, and that users need only install the few plug-ins that they need, out of a much larger pool of possibilities. A plug-in must be downloaded by the user and the functionality provided must be optional, not critical for the rest of the platform operation.

Some tools/functionalities should be included in DIAMONDS platform as plug-ins, when they imply such interaction with the user, and when the use of those functionalities is optional for the rest of the platform functionalities.

Another possibility for tools integration should be the use of **extensions**. Plug-ins are slightly different from extensions, which modify or add to existing functionality. The main difference is that plug-ins generally run within a sandbox¹, rely on the main program's user interface and have a well-defined boundary to their possible set of actions. Extensions generally have fewer restrictions to their actions, and may provide their own user interfaces.

¹ In computer security, a sandbox is a safe place for running semi-trusted programs or scripts, often originated from a third party. The sandbox security model provides a tightly-controlled set of resources for foreign programs to run in, such as a small "scratch-space" on the disk and a section of memory to carry out instructions. The sandbox may allow some user interaction, and the user may be prompted to allow or disallow certain actions as the program runs.

5 PLATFORM IMPLEMENTATION

5.1 FUNCTIONALITIES

The diagrams detailed in the first section describe the complete scope of the platform: all the functionalities the platform could perform. In figure 30 we can see the complete block diagram and the tools assigned for the implementation of every one of the functionalities of the platform.

In an initial stage of the platform development, only some of those functionalities are going to be implemented, namely those that constitute the basic core of DIAMONDS platform.

Functionalities regarding to expression data analysis will be implemented using Expression Profiler, from EBI. Initially, we will try to use Expression Profiler not only as a tool included in the platform, but as the skeleton base. In this sense, in the near future much effort is going to be done in the comprehension of Expression Profiler's source code in order to determine if this tool suits or not to the platform skeleton.

In case we validate that Expression Profiler can function as a seed for future development, the following steps will be based on it. So, all the rest of the tools will be added as modules connected to the functionalities provided by Expression Profiler.

Expression Profiler will provide also a workbench for DIAMONDS, as well as direct access to Array Express [5] public repository for microarray gene expression.

In a second stage of platform implementation it will be expanded with additional functionalities, like:

- Detection of genes periodically expressed
- Promoter Analysis
- Pathways annotation
- Modelling and simulation

Those functionalities, except the one related to modelling and simulation, are going to be provided by Gene Publisher [6], from CBS. Probably, the integration method applied in those cases will be *Web Services*, but some other ways are going to be considered as well.

The last one, "Modelling and Simulation" is also an essential functionality of the platform design. This one is going to be provided, in the first stage, by GIN-SIM [7], from Université de la Méditerranée. In this case *web service* does not seem to be the best integration method because *web service* fits better with stateless approaches. So, people in charge of GIN-SIM must provide another type of integration way, like for example a *plug-in*.

In parallel, within the project we need to develop mechanisms for data visualisation. For that purpose we are going to consider the integration in the system of specific tools like, for example, Cytoscape [8].

In a third stage, some more functionality should be added:

- For protein classification, we may implement some (or all) functionalities of Protonet [9] and Pandora [10], both provided by The Hebrew University of Jerusalem.
- For gene homologies, the platform will include also a blast processing, or tools developed by PSB (TREECON, i-ADHoRe, other: <http://bioinformatics.psb.ugent.be/software.php>).

Each one of the stages listed should last approximately 6 months, so we should have a beta version of the platform in December of 2006.

During 2007 year, the whole set of partners will have time to try and test the platform in order to improve it.

We have to stress the importance of wet-lab partners in the testing phase because they are going to be the main users of the platform. Also, in the near future, all wet-lab experts are going to be asked for feedback in platform design, to develop the platform GUI that best fits the scientists' needs.

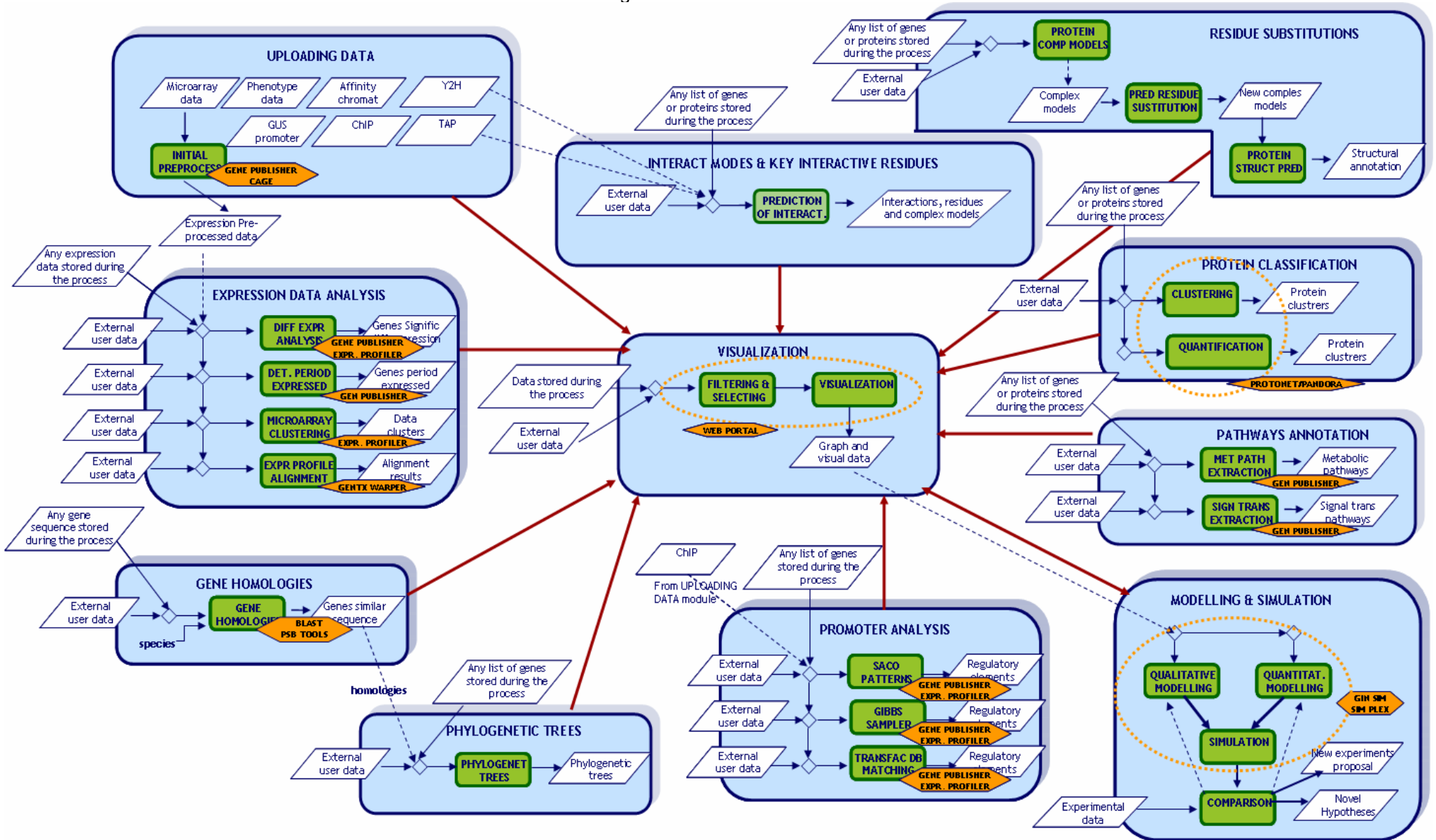


Figure 30. Block Diagram with tools assignments

5.2 CORE DATABASE

GIN-DB will serve as core database for highly validated data. It will collect a vast amount of information regarding the core cell cycle genes studied in the project, including also annotations and text mining. For all the (raw) high throughput experimental data we will first identify the established public or local data repositories in order to plan the necessary links from and within the platform (e.g. to GIN-DB).

As mentioned before, text mining and data curation processes will be performed prior to the platform launching. For that purpose, specific interfaces will be developed to upload and curate such data.

In general terms, GIN-DB [2] (Gene Interaction Network Database), from Université de la Méditerranée, is a repository for different types of data, broadly divided in three types of relationships:

- Physical, macromolecular relationships: complexes or associations
- Genetic interactions
- Regulatory effects

GIN-DB general structure and terms considered is shown in figure 32.

During the next months, GIN-DB will be improved in order to get a semi-definitive structure to allow storage of essential, deep information for core cell cycle genes. As a second step, an xml schema will be extracted from the database design to be considered as a uniform interchange language for database's requests and responses.

Text miners and people in charge of wet-lab experiments, are going to feed GIN-DB, and they will have available a graphical interface to upload data and curate it. To develop appropriate interfaces, it first is necessary to have examples of the files that will be provided by text miners and wet lab experts.

To accommodate high throughput data , most importantly microarray data, we will try not replicate in the efforts put into public databases, but rather provide links to them.

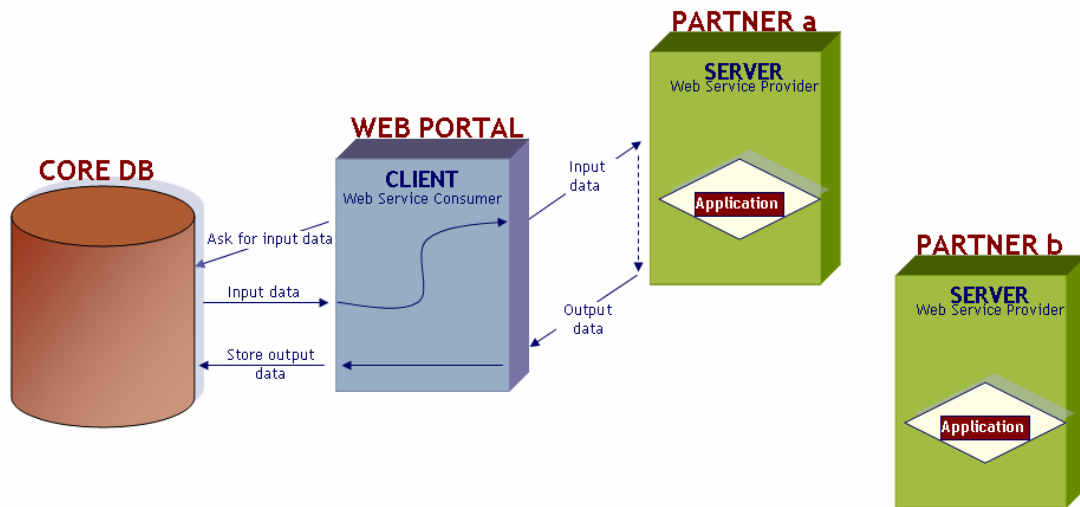


Figure 31. Database queries centralised

The final characteristic specified for GIN-DB is that all the queries to that database are going to be centralised in the web portal. Getting data from the data base and storing data in the database are tasks that will only be performed by the web portal, and not by the different tools involved in the platform.

In that sense, a general database interchange is detailed in the following example:

When the web portal decides to use the functionality provided by one of the partners, for example partner a, first of all, the web portal should extract from the database the input file necessary for the web service execution. The web portal takes that file and offers it to the partner tool.

After processing, the tool generates an output file which is returned to the web portal. The web portal will then store the new data generated into the database.

Centralising the queries lowers the chances to run into inconsistencies.

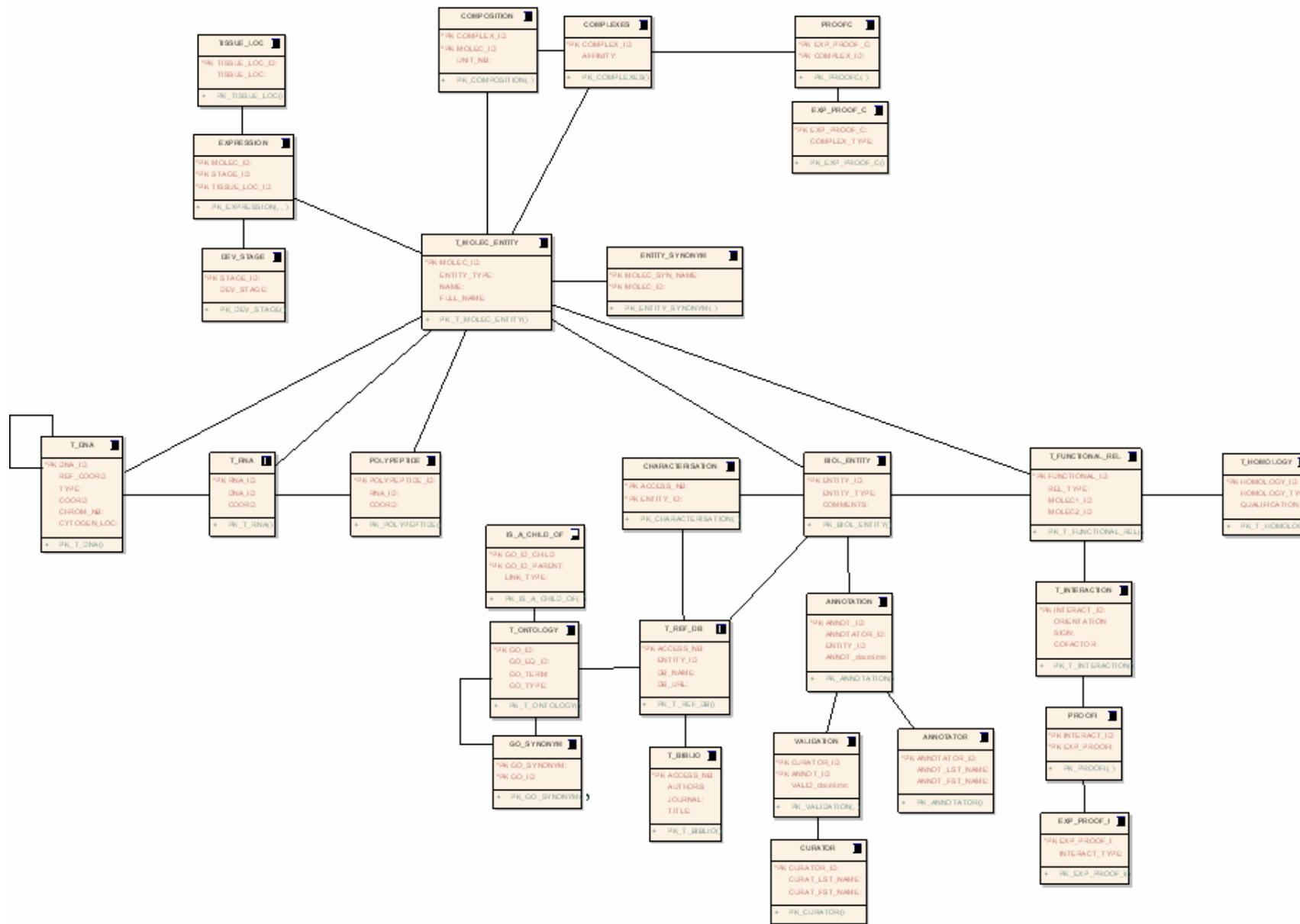


Figure 32. GIN-DB structure

5.3 *FUTURE EXTENSIONS*

Once the core development with the basic functionalities is finished, also other tools and functionalities are going to be integrated, as listed in the general scope of the platform.

Some examples could be:

- GenTxWarper, provided by PSB for expression profile alignment
- SIM-PLEX, provided by PSB, as another type of modelling and simulation tool.
- Links to Ensemble, Ensmart, other external databases linked to GIN-DB, possibly facilitated by BioMoby.

Moreover, during the development of the platform new tools and functionalities (not considered at present) may become available / can be deemed necessary from a use-case perspective making it necessary to include them as new modules of the platform. In this sense, as stated before, the platform must be flexible and extendable.

6 CONCLUSIONS

1) For DIAMONDS design and development, the first step has been the complete description of requirements as listed in the project description using two kinds of description diagrams:

- FLOW-LIKE DIAGRAM
- BLOCK DIAGRAM

The development of the flow-like diagram is one of the first steps towards the description of DIAMONDS platform. It allows the partners to get a better understanding of the entire system and will also trigger discussions between each other and with wet-lab users. The flow-like diagram represents an abstract view of the entire system.

The development of the Block diagram is the second phase in the description of the DIAMONDS platform, and it provides another viewpoint of the whole system and a next design step for platform programming. It has been developed taking the flow-like diagram as a starting point.

In the block diagram we can identify the user options proposed for the general implementation of the platform (modules) and the basic functionalities (blocks) that should be implemented by one or several tools to offer the user data processing options.

To sum up, in table 1, we have the complete set of options and functionalities, as well as the tools proposed to implement the functionalities..

There are also two modules not considered as user options because the execution is done in background:

- Annotation module: Executed automatically and in a transparent way for the user every time new data gets into the system.
- Text mining and data curation: Is considered a background module because it is going to be executed prior to the platform launching.

All *modules/user options* are independent, and this is a very important characteristic to make the platform flexible and extendable. So, if at any moment in the development cycle the consortium decides to add or remove any user option in the graphical user interface, it only implies to add or remove a specific module in the block diagram.

USER OPTIONS	FUNCTIONALITIES	TOOLS PROPOSED
UPLOADING DATA	Initial pre-processing	Expression Profiler CAGE
EXPRESSION DATA ANALYSIS	Differential Expression Analysis	Expression Profiler / Gene Publisher
	Detection genes periodically expressed	Gene Publisher
	Microarray clustering	Expression Profiler
	Expression Profile Alignment	GenTxWarper
GENE HOMOLOGIES	Gene Homologies	Blast Tools developed by PSB
PHYLOGENETIC TREES	Phylogenetic trees construction	None, at the moment
PROMOTER ANALYSIS	Saco Patterns	Gene Publisher Expression Profiler
	Gibbs Sampler	
	TRANSFAC DB matching	
PROTEIN CLASSIFICATION	Clustering	Protonet / Pandora
	Quantification	Protonet / Pandora
INTERACTION MODES AND KEY INTERACTIVE RESIDUES	Prediction of interaction modes and key interactive residues	None, at the moment
RESIDUE SUBSTITUTIONS	Residue substitution effect prediction	None, at the moment
	Protein complex models	None, at the moment
	Protein structure prediction	None, at the moment
PATHWAYS ANNOTATION	Metabolic Pathways Extraction	Gene Publisher
	Signal Transduction Pathways Extraction	Gene Publisher
MODELLING AND SIMULATION	Qualitative modelling	GIN-SIM/ SIM-PLEX
	Quantitative modelling	GIN-SIM/ SIM-PLEX
	Simulation	GIN-SIM/ SIM-PLEX
	Comparison	None, at the moment

Table 1. User options, functionalities and tools proposed

2) Another important characteristic of the platform is that all information under study by a user should be available at any moment. For that purpose, when a user gets into the platform introducing a login and password, a folder structure will automatically be created where the father folder should have the same name as the login user. Inside this folder structure the user may have visible and available all files uploaded to the workbench, as well as the processes done and the output files generated.

3) For platform implementation, the set of functionalities provided by the different tools should be integrated into a web portal. Web Services is the basic technology proposed for

integration in this project because it allows the communication between systems with no common design aspects. In the DIAMONDS platform we will have several tools designed and implemented using different programming languages and different deployment modes, and in this kind of heterogeneous environment Web Services provide many facilities for integration and collaboration.

4) However, for every particular case we will consider the use of either web services or other alternatives, choosing the one that fits best. For example, for applications or services that imply an important interoperation between the tool and the user, maybe web services are not the best option for integration. This fact, complemented with the complexity of generating the web service environment, could induce us to use another integration technology, like plug-ins or extensions.

5) Platform implementation will be progressive. Initially, platform functionalities will spawn from expression data analysis as facilitated by Expression Profiler. For that purpose, the first efforts will be done in trying to use Expression Profiler, from EBI, not only as a tool included in the platform, but as the skeleton base for further development.

Next, we will integrate additional essential functionalities including "Detection of genes periodically expressed", "Promoter analysis", "Pathways annotation" and "Modeling and simulation". The remaining functionalities will be integrated later, in each case considering the integration technology that best fits depending on the characteristics of the tool, the level of interactions with the user, and the particular competence of the tool builder.

6) Another essential part for the complete design of the platform is the core database of the system. GIN-DB, provided and developed by Umed, is going to be used for core cell cycle data, and it will collect a vast amount of information regarding the genes studied in the project, including also annotations and text mining. During the next months, GIN-DB's responsible will improve the database in order to move to a first stable phase. Depending on the evolving needs it is foreseen that later in the project additional requirements may be implemented in the database schema.

An important characteristic specified for GIN-DB operation is that all the queries to that database are going to be centralised in the web portal. Centralising the queries lowers the problem of query inconsistency.

7) In summary, the development phase will include: the development of integration components from tools (web services, plug-ins, etc.), the integration of them in the platform, the improvement of the database and the addition of it to the platform as a core repository of high value data. Also, in the development phase, especially the design of the platform GUI, we will always take into account the wet-lab scientists' opinion and feedback, because they are going to be the main users of DIAMONDS.

8) DIAMONDS platform will be flexible and extendable in order to allow the integration of new tools and functionality not considered at present, but deemed necessary during platform development.

9) The beta version of the project will be available in December of 2006, allowing the consortium partners to test, use and improve the platform during 2007.

REFERENCES

- [1] **“DIAMONDS, Annex I - Description of Work”**,
SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT,
SIXTH FRAMEWORK PROGRAMME (2004).
- [2] **“GIN-DB: a relational scheme for the integration of multi-species gene interaction data”**, Sabatier C. et al, ECCB Poster (2003).
- [3] **“XML Metadata Interchange (XMI)”**,
<http://www.omg.org/technology/documents/formal/xmi.htm>,
Object Management Group web site (2005).
- [4] **“DIAMONDS - Dedicated Integration And Modelling Of Novel Data and prior knowledge to enable Systems biology”**,
<http://www.sbcellcycle.org>,
DIAMONDS main web site (2005).
- [5] **“ArrayExpress—a public repository for microarray gene expression data at the EBI “**
H. Parkinson*, U. Sarkans, M. Shojatalab, N. Abeygunawardena, S. Contrino, R. Coulson, A. Farne, G. Garcia Lara, E. Holloway, M. Kapushesky, P. Lilja, G. Mukherjee, A. Oezcimen, T. Rayner, P. Rocca-Serra, A. Sharma, S. Sansone and A. Brazma
European Bioinformatics Institute, EMBL-EBI Wellcome Trust Genome Campus, Hinxton CB10 1SD, UK
http://nar.oxfordjournals.org/cgi/reprint/33/suppl_1/D553.pdf
- [6] **“GenePublisher: automated análisis of DNA microarray data”**
Steen Knudsen Christopher Workman, Thomas Sicheritz-Ponten, and Carsten Friis
<http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&pubmedid=12824347>
- [7] **“GINsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks”**
A.Larrinaga, L.Sánchez, D.Thieffry, C.Chaouiya (2005)
- [8] **“Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks”**
Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski and Trey Ideker
<http://www.genome.org/cgi/content/full/13/11/2498>

[9] "**ProtoNet 4.0: A hierarchical classification of one million protein sequences**"

Noam Kaplan, Ori Sasson, Uri Inbar, Moriah Friedlich, Menachem Fromer,
Hillel Fleischer, Elon Portugaly, Nathan Linial and Michal Linial

<http://www.protonet.cs.huji.ac.il/papers/ProtonetNAR2005.pdf>

[10] "**PANDORA: keyword-based analysis of protein sets by integration of annotation sources**"

Noam Kaplan, Avishay Vaaknin, and Michal Linial

<http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&pubmedid=14500825>

[11] "**Web Services: a practical introduction**"

Systinet Corp.TM. White Paper, 2003