Interoperability of ACGT knowledge discovery services with existing bioinformatics tools / Prototype and report of the final ACGT analysis interface

Project Number:	FP6-2005-IST-026996
Deliverable id:	D6.6 / D6.7
Deliverable name:	Interoperability of ACGT knowledge discovery services with existing bioinformatics tools / Prototype and report of the final ACGT analysis environment
Date:	August 30th, 2010

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 6.6 / 6.7
Document name:	Interoperability of ACGT knowledge discovery services with existing bioinformatics tools / prototype and report of the final ACGT data analysis environment.
Document type (PU, INT, RE)	PU
Version:	1.0
Date:	30/08/2010
Editors: Organisation: Address:	Stefan Rüping Fraunhofer IAIS stefan.rueping@iais.fraunhofer.de

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

ABSTRACT:

The purpose of this deliverable is to describe the final ACGT data analysis environment. In a nutshell, data analysis in the ACGT system is performed by accessing the data to analyse following the ACGT multi-level data integration approach (cf. Deliverable D7.10) and orchestrating a pipeline of data analysis services in the form of an analysis workflow in the ACGT Workflow Environment (cf. Deliverable D9.6), all enabled by appropriate service metadata (cf. Deliverable D9.5). Hence, the ACGT data analysis environment can more correctly be described as the data analyst's view of the integrated ACGT system. To complete the picture of the ACGT data analysis environment, this deliverable will focus on the actual knowledge discovery services that perform the actual data analysis. In ACGT, the open-source statistical toolkit / language R, the open-source suite of bioinformatics web-services BIOMOBY, and generic command-line tools have been identified as the most relevant tools for integration.

KEYWORD LIST: Knowledge Discovery Services & Architecture

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	22/03/2010	Draft	Max García (D6.6)
0.2	24/03/2010	Draft	O.Trelles (D6.6)
0.3	17/05/2010	Draft	Stelios Sfakianakis (D6.6)
0.4	15/08/2010	Draft	Stefan Rüping (joint version)
1.0	30/08/2010	Final	Stefan Rüping

List of Contributors

- Stefan Rüping, FHG
- Stelios Sfakianakis, FORTH-ICS
- Oswaldo Trelles, UMA
- Johan Karlsson, UMA
- Max García, UMA

Contents

Executive Summary

The purpose of this deliverable is to describe the final ACGT data analysis environment. In a nutshell, data analysis in the ACGT system is performed by accessing the data to analyse following the ACGT multi-level data integration approach (cf. Deliverable D7.10) and orchestrating a pipeline of data analysis services in the form of an analysis workflow in the ACGT Workflow Environment (cf. Deliverable D9.6), all enabled by appropriate service metadata (cf. Deliverable D9.5). Hence, the ACGT data analysis environment can more correctly be described as the data analyst's view of the integrated ACGT system.

To complete the picture of the ACGT data analysis environment, this deliverable will focus on the actual knowledge discovery services that perform the actual data analysis. A considerable number of knowledge discovery services - such as GridR, Literature Mining, Association Rule Mining, Gene Clustering, and Visualization - have already been described in Deliverables D6.2 and D6.4. However, in response to the plethora of freely available data mining and bioinformatics tools that are available today, the main objective of the knowledge discovery work package in ACGT is not the development of yet another set of tools, but the integration of (quasi-)standard modules. The important role that the integration of existing tools play in the definition of the ACGT data analysis approach is the reason why we chose to join the two deliverables D6.6 "Interoperability of ACGT knowledge discovery services with existing bioinformatics tools" and D6.7 "Prototype and report of the final ACGT analysis environment" into this single joint deliverable.

In ACGT, the open-source statistical toolkit / language R and the open-source suite of bioinformatics web-services BIOMOBY have been identified as the most relevant tools for integration, both because of their wide-spread popularity among the researchers, and because of the complementarity of the approaches (interactive programming language vs. web-services). In addition, many researchers develop new tools outside of established frameworks. To simplify the integration of these tools, and to lower the threshold for new users to add tools to ACGT, a way to integrate tools that work on the most simple of all interfaces, namely the command line, has been developed. In summary, these three, largely complementary ways of developing and integrating new ACGT services covers the state of the art in software engineering in scientific disciplines.

1. Introduction

The purpose of this deliverable is to describe the ACGT data analysis services. As all ACGT services are orchestrated by the central ACGT workflow editor and enactor, and are interfaced by the integrated ACGT portal, this already describes the complete ACGT data analysis environment as a whole. The document is structured as follows: first, we will give an overview of bioinformatics as a whole, in order to describe the different kinds of tools and methods that a analysis platform for clinic-genomic data has to support. In the following, the three approaches to integrate existing, standard data analysis software - namely BioMoby, Command Line Services and R - are described.

Biology and Bioinformatics at a glance

As in any other application domain, in the case of computational biology is quite important a minimal understanding of biological data, both to develop and to exploit efficiently bioinformatics applications. In these few paragraphs we will try to summarize part of this knowledge.

All living organisms are endowed with genetic material that carries out the instructions to built-up all the other constituents of the cell. This information is mainly stored in long strands of DNA grouped into X shaped structures called chromosomes. DNA instructions are written in a four letter alphabet represented by A, C. G, T that correspond to the Adenine, Cytosine, Guanine and Thymine nucleotides. All of the genetic information of an organism is referred to as its *genome*. The genome size varies from few hundreds in some bacteria to more than 10^{10} nucleotides in salamander (i.e. Human genome is approximate 3,100 million nucleotides length).

But not all the DNA in eukaryotes codes for proteins. Just around a few 5% of the human genome is formed by coding regions called exons, separated by long strains of non-coding regions named introns. Even more, the instructions for producing a particular protein -called *genes*- are normally composed by several exons separated by introns inserted into them. These introns are spliced out before the sequence is translated into aminoacids (the constituent of proteins).

The cell machinery is able to interpret the gene instructions following the rules of the genetic code. Each non-overlapping triplet of nucleotides, called a *codon*, codes for one of the twenty different amino acids. Observe that four nucleotides can code $4^3 = 64$ possible triplets, which is more than the 20 needed to code for each amino acid. Three of these codons designate the end of a protein sequence (*stop* codons). That means that most amino acids are encoded by more than one codon, which is explained as the *degeneracy* of the code.

⇒ DNA sequencing

Many technological breakdowns have made possible to obtain the DNA sequence of whole organisms. A genomic project is the effort to obtain the sequence of the DNA of a given organism. Being able to divide the genome into moderate sized chunks is a prerequisite to

determining its sequence. The sequencing method developed by Sanger allows a resolution of a few thousand base-pairs at a time. Thus, in order to determine the sequence of large pieces of DNA, many different overlapping chunks must be sequenced, and then these sequences must be assembled.

In similar way, modern NGS sequencing technologies based on the pyro-sequencing principle also split randomly the DNA in a large number of released fragments, but in this case, ranging from 25 to 300 bases or 'reads'. The resolution and coverage of the intended sequencing determines which sequencing technique to use. Either the task is to assemble a genome, to determine the transcripts and their concentrations, to detect nuclesome positions, to identify single nucleotide polymorphisms, or to estimate copy number variations. All these algorithms have to handle huge amounts of data. For example, the NGS-Solexa technique can be used to achieve very high coverage of the genome because it yields more than 50 million reads which map back to some 1000 hits on the genome; compared to Roche-454 technique which is better suited for genome assembly because the reads are longer and support the assembly with larger overlaps. SOLiD (Applied Biosystems) is comparable with Solexa with 10-20 million reads per run.

Efforts have to be made in order to ensure the quality and reliability of sequence data to remove contaminant sequences [1,2], vectors [3]; and in general filtering of low-quality sequences, identification of specific features (such as poly-A or poly-T tails, terminal transferase tails, and adaptors), removal of contaminant sequences (from vector to any other artefacts) and trimming the undesired segments. There are some bioinformatic tools that can accomplish individual pre-processing aspects (e.g. TrimSeq, TrimEST, VectorStrip, VecScreen, ESTPrep [4], crossmatch, Figaro [5]), and other programs that cope with the complete pre-processing pipeline such as PreGap4 [6] or the broadly used tools Lucy [7,8] and SeqClean [9]; or Web-based applications for annotation or assembling. There are Web implementations (ESTAnnotator [10], ESTpass [11] or ESTExplorer [12]) start with pre-processing and end with assembling and/or annotating ESTs. On the specific task of DNA assembly, the most used applications (until recently) have been the software package formed by Phread [13], Phrap [14] and Consed [15]. Phred is well known as the “base-caller” since the main tasks y to identify the bases from the chromatogram assigning quality values. Phrap is the assembler and finally Consend is a graphical tool for finishing the full process.

Using these assembly methods, a first draft of the human genome was obtained at the end of the last century, several tens of high eukaryotes genomes are now available (see http://www.ebi.ac.uk/genomes), and close to thousand bacteria and archaea (prokaryote organisms) have also been completed (www.ncbi.nlm.nih.gov/**genomes**/static/a.html).

⇒ Genes and proteins: Sequence analysis

Although knowing genome composition of a given organism is an important achievement, it is only the first step in understanding the biological process that underlies life. The second step is to identify the genes hidden in the mountains of DNA information. But the process is not as easy as it seems. The basic process of synthesizing proteins maps from a sequence of codons to a sequence of amino acids. However, there are some important complications: since codons come in triples, there are three possible places to start parsing a segment of DNA, and it is also possible to read off either strand of the double helix, and finally there are

well known examples of DNA sequences that code for proteins in both directions with several overlapping reading frames.

The central feature of life organisms is their ability to reproduce it and became different through an accumulative process named *evolution.* In order to evolve, there must be a source of *variation* such as random changes or mutations (*insert* a new nucleotide, to *delete* an existing one, or to change one nucleotide into another), sexual recombination and various other kinds of genetic rearrangements. These changes modify the genetic information passed from parent to offspring. The analysis of such variations allows determine the way in which organisms were diverging (phylogenetic analysis). This is mostly performed by similarity comparison of molecular sequences. The similarities and differences among molecules that are closely related provide important information about the structure and function of those molecules.

Large scale sequence analysis provides a basic way of understanding the whole genome structure in terms of protein coding-non coding regions as well as regulatory elements. Gene-discovering [16] and protein function assignment [17], are a first order requirement in this process.

Searching for similar sequences using global or local strategies, including the statistical evaluation of their significance, are by far the most used computational tasks in biosequence analysis area. Several tools are routinely used to this end, from heuristics ones such as FASTA [18] and BLAST [19] families of programs, to more exhaustive algorithms based on dynamic programming approaches [20, 21]. High assurance and accuracy are obtained from these methodologies when sequences similar to a query sequence clearly exist on the database [22, 23].

Other systematically used tools in this area are pattern-based procedures. Most of the pattern discovery algorithms proceed by collecting a set of related sequences using systematic database searches [24, 25], or by computing all-all distance matrices and clustering procedures [26], followed by a semi-automatic multiple sequence alignment [27] to highlight the common feature (if any) shared by the set of sequences. The key element in these strategies is to detect the presence of strong alignments shared by all sequences, from which it is possible to infer that those sequences share a common feature (named “signature”). In this way, when the same pattern is found in a new sequence it is assumed that the new sequence also shares the feature.

Linkage analysis is another tipical task in sequence analysis. Genetic linkage analysis is a statistical technique used for rapid and largely automated construction of genetics maps from gene linkage data. For example, one key application of linkage analysis aims to map human genes and locate disease genes. The basic computational goal in genetic linkage analysis is to compute the probability that a recombination occurs between two loci. Most frequently used programs estimate this *recombination function* by using a maximum likelihood approach [28].

The champions in Multiple Sequence Analysis (MSA) are Clustal [29, 30, 31] and Philyps packages both used as starting points for building phylogenetic trees [32, 33]. The term "phylogeny" studies the evolutionary history of a group of organisms, predicted pedigree of their ancestors and the genotype of the individuals in the lineage which is discovered through molecular sequencing data. Phylogeny reconstruction is mostly based on maximum

likelihood estimation or Bayesian analysis, both showing exponential computing time for calculations. The original Felsenstein version of the method is implemented in the PHYLIP package (available at evolution.genetics.washington.edu). In very simple terms, the maximum likelihood method searches for a tree and a branch length that have the greatest probability of being produced from the current sequences that form that tree. The algorithm proceeds by adding sequences into a given tree topology in such a way that maximizes the likelihood topology. Once the new sequence is inserted, a local-optimization step is performed to look for minor rearrangements that could lead to a higher likelihood.

Reduction of the computational CPU-time and memory demand have been in the main motivations for the development of new strategies and algorithms. An exemplary case in the study of the genome of one organism is the so called 'Spectral Analysis of DNA sequences' where the periodicities of the genome can be revealed. DNA spectral analysis can be applied to systematically investigate DNA patterns, which may correspond to relevant biological features in very long sequences that are not identifiable by sequence alignment. This technique can be used to design useful classifiers and predictors in genomic biomarker discovery efforts - both applicable in biomedicine and green biology. In order to enable the phylogenic and biological comparison of a large number of long sequences in frequency domain, complex models are applied that involve expensive calculations. New implementations of both algorithms and programs are required to make effective use of computational parallelisation. New research may be required to speed up specific use cases.

⇒ 3D folding of proteins

Once the genes have been identified, and thus, the protein it encodes, next step is to disclose the role of the protein in the organism. The sequence of amino acid residues that make up a protein is called the *primary structure* of the protein, but the function the protein holds is more related to its tri-dimensional conformation and one of the major unsolved problems in molecular biology is to be able to predict the structure and function of a protein from its amino acid sequence. In raw terms, the folding problem involves finding the mapping from primary sequence (a sequence of from dozens to several thousand symbols, drawn from a 20 letter alphabet) to the real-numbered locations of the thousands of constituent atoms in 3D space.

An approach to protein folding problem starts with the prediction of the *secondary* structure of the protein which refers to local arrangements of a few to a few dozen amino acid residues that take on particular conformations that are seen repeatedly in many different proteins: corkscrew-shaped conformations called α-*helices*, and long flat sheets called a β- *strand*, and a variety of small structures that link other structures: *turns.* Next problem is to determine the position of the atoms in a folded protein –known as its *tertiary* structure- and finally, some proteins only become functional when assembled with other molecules, termed the *quaternary* structure of the protein.

3D protein folding, analysis of protein or DNA flexibility and prediction and optimization of structures, is based on molecular dynamics (MD). Because of the computationally intensive nature of structural analysis, it has been used an early OSS implementor of parallelisation techniques, with a number of popularised libraries like Gromacs and NAMD. Currently,

simulation time for medium to large sized protein systems require roughly 3-4 weeks of CPU time to achieve 1 nanosecond of simulation [34]. The most efficient MD codes only scale reasonably well up to 128 or 256 cores, depending on the communication protocol and the size of the simulated system, giving an effective simulation rate of 2-3 ns per day. As biological processes rank in the microsecond to millisecond time scale, state-of-the-art simulation allows only scratching the surface of processes being analyzed. Several approaches are being used to overcome the problem, from designing specialized hardware to, most usually, using coarse-grained techniques like discrete dynamics, Brownian dynamics (motion of molecules in molecular simulation), restricted Monte Carlo, etc. where a significant insight is sacrificed to allow longer simulations. A more specialized kind of biomolecular simulations is QMMM (Quantum mechanics molecular mechanics), where quantum mechanics is used to obtain potential energies instead of classical force-fields. This allow a much better insight in processes involving chemical transformations, like enzyme catalysis. Unfortunately QMMM is extremely demanding and, in contrast to regular MD, requires huge amounts of memory.

Another important issue in the structural biology area is integrative image processing. A useful example of this latter approach, which represents one of the most innovative ways to combine heterogeneous sets of information, can be found in the EMDB [35] and the Bioimage [36] initiatives, an example of European global leadership in research. This initiative stores 3D density maps of molecules and sub-cellular structures acquired by electron microscopy. All structural analysis research is computationally intensive and benefits from HPC strategies.
As the number of protein structures known in atomic detail increases, the demand for searching by similar 3D structures also grows. Several computer algorithms helps in this task: (a) extending dynamic programming algorithms [37]; (b) importing strategies from computer vision areas [38]; (c) using intra-molecular geometrical information, as distances, to describe protein structures [39]; and (d) finding potential alignments based on octomeric C alpha structure fragments, and determining the best path between these fragments using a final dynamic programming step followed by least squares superposition [40].

⇒ Gene expression

Although every cell has the same DNA, at any particular time, a given cell is producing only a small fraction of the proteins coded for in its DNA. The amount of each protein is precisely regulated in order for the cell to function properly in a given particular environment. Thus, the cell machinery modifies the level of proteins as response to changes in the environmental conditions or other changes. Although the amount of protein produced is also important, genes are generally said to be *expressed* or *inhibited*. Recently advances in gene monitoring microarray technology [41] has enabled the simultaneous analysis of thousands of gene transcriptions in different developmental stages, tissue types, clinical conditions, organisms, etc. The availability of such expression data affords insight into the functions of genes as well as their interactions, assisting in the diagnosis of disease conditions and monitoring the effects of medical treatments.

Bioinformatics transcriptomic analysis allows monitoring the concentration of expressed messenger RNA (mRNA) in the cell, drawing a global picture of cellular function. This allows

testing for differential expression of genes, gene network elucidation, and, for example, research in cell development, epigenentics and environment interactions. Currently microarray technology is popular, but next generation sequencing is also becoming viable for, so called, deep sequencing mRNA. Transcriptomic analysis is central to biomedical research and can be used for genotyping, detecting single nucleotide polymorphisms (SNPs) and copy number variations (CNVs).

A key task to derive biological knowledge from gene expression data is to detect the presence of sets of genes that share a similar expression pattern and common biological properties, such as function or regulatory mechanism. Current approaches to analyze microarray data includes some pre-processing steps to remove the experimental and acquisition noise [42] and for data normalization, frequently followed by the application of clustering algorithms [43] in order to establish sets of co-expressed genes. These algorithms can incorporate available information about genes and gene products to infer knowledge. Each set of co-clustered genes has to be further examined with the aim of discovering common biological connections among them. In this way, biological information is incorporated as a subsequent process to the analysis of expression data.

Transcriptomic analysis involves, often complex, statistical analysis of variations of tens of thousands expression levels in many individuals and/or environments. The current crop of software solutions (mostly with the statistical language R) were designed for analysis on a single computer and finding computational solutions is an immediate concern. Typical areas of research using are eQTL / mQTL analysis (expression / methylation Quantitative Trait Locus), gene-expression data processing, clustering and classification methods.

⇒ Pathways

The revolution in biology becomes from the knowledge of the basic transformations of intermediary metabolism that can involve dozens or hundreds of catalyzed reactions. These combinations of reactions, which accomplish tasks like turning foods into useable energy or compounds, are called metabolic *pathways.* Because of the many steps in these pathways and the widespread presence of direct and indirect feedback loops, they can exhibit much counterintuitive behaviour.

When Doolittle [44] used the nascent genetic sequence database to prove that a cancer causing gene was a close relative of a normal growth factor, molecular biology labs all over the world began installing computers or linking up to networks to do database searches. Since then, a bewildering variety of computational resources for biology have arisen.

⇒*System biology : integrating the knowledge*

Technological breakthroughs such as next generation sequencing and gene-expression monitoring technology have nurture the ‘omics’ revolution enabling the massive production of data. Unfortunately, this valuable information is often dumped in proprietary data models and specific services are developed for data access and analysis, without forethought to the

potential external exploitation and integration of such data. Dealing with the exponential growing rates of biological data was a simple problem when compared with the problem posed by diversity, heterogeneity and dispersion of data [45]. Nowadays, the accumulated biological knowledge needed to produce a more complete view of any biological process is disseminated around the world in the form of molecular sequences and structure databases, frequently as flat files, as well as image/scheme-based libraries, web-based information with particular and specific query systems, etc. More than one thousand databases [46] and more than 130 servers linking around 1200 services [47] form the "computational compounds" in nowadays bioinformatics.

Forcing a simile, the Internet browser has become the "in-silico" pipette of traditional wet-labs, where compounds are combined and recombined to produce the desired result. In a similar way, bioinformatics strongly relies in the universal availability of web resources that often need to be combined –drawing what is called a *workflow*- to produce a useful outcome. Despite the popularity of web services, however, it is still difficult to locate the right services to combine, understand their interfaces and required parameters, and especially tedious to copy and paste partial results to interconnect different services in pipeline fashion.

Since most web services do not offer an interactive interface to deal with these matters, the solutions are usually relegated to the client software that has come to govern such things as invocation, and input/output visualization. Several clients have been developed to assist in the exploitation of web services, most of them oriented to specific data models and protocols. Gbrowse [48], MOWServ [49], Dashboard (http://biomoby.open-bio.org/CVS_CONTENT/moby-live/Java/docs/Dashboard.html), Seahawk [50], Remora [51] and Taverna [52] are well-used clients, but most of them are limited to working with BioMOBY web services. Taverna and Remora are exceptions because they specialize in the composition and execution of Workflows, while Seawhawk specializes in the data-driven execution of BioMOBY web services.

However, the idyllic vision of the Web as an endless source of resources comes with a challenge: the proliferation of protocols for data interchange, representation and transfer are a barrier to making services work together (interoperability) [53-57]. New tools that can help break down this barrier introduce the potential to combine web services into complex workflows[1], allowing them to address new research problems.

To this end, service providers must define relations of dependence or compatibility between resources for which it is necessary to provide additional meta-information about the computational resources. In such cases, registering resource information into metadata repositories helps to solve the problem. A repository [58] is a system for storing and querying information, using a data model to provide a structure and usually an application programming interface (API) to access the information. Having all the information stored in a common and known format simplifies the task of combining resources provided by different entities, and allows homogeneous access to different kinds of information. For example, MyExperiment [59] is a repository of workflows and related metadata and, like other systems

[1]A workflow or workflow model is the complete or partial automation of a process in which information or tasks are passed from a participant to another, according to a defined set of procedural rules. (The Workflow Management Consortium - WfMC).

(UDDI [60], FETA [61] or BioMOBY [62]) provides an API to facilitate the development of remote clients. BioMOBY also includes a taxonomy of data types, shared by all the services registered on the same server, which supports the identification of compatible services and their interoperability. These solutions provide a mechanism for the rapid discovery [63], invocation [64] and integration [65] of services.

A final comment about HPC.

Noteworthy to observe is the computational demands in the calculation of All-All matrix of similarity scores which grow with the square of the number of sequences participating in the alignment, generating a dynamics of the use of computational resources that is clearly faster than the CPU-power growth of the different generations of microprocessors.
The huge amount of information generated collectively by next generation sequencers imposes new challenges on computational analysis. The petabyte scale of this 'big data' generation requires introducing parallelised computing and other high performance computing strategies to effectively analyse the wealth of generated information. At this point, bioinformatics groups are taking initiative to work on parallelising existing and new software tools and libraries. To prevent duplication of effort, and sub-optimal solutions, a concerted cross-project inter-disciplinary approach is required from the domains of bioinformatics and high performance computing.

2. Interoperability with Biomoby Services

The BioMOBY project [1] offers an impressive number of bioinformatics tools and services. It builds upon an registry where the Biomoby compliant tools are registered and discovered by interesting clients based on their metadata and semantic based descriptions. In particular, as described in Deliverable 9.4, the Biomoby framework is based on a set of end-user-extensible ontologies as its framework to describe data semantics, data structure, and classes of bioinformatics services. These ontologies are shared through a Web Service registry system, MOBY Central, which uses the ontologies to semantically bind incoming service requests to service providers capable of executing them [3].

In order to foster interoperability the Biomoby tools should be created according to the following guidelines:

- Use of the Biomoby object ontology (Figure 1) to define the inputs and outputs
- Use of the Biomoby service classification ontology (**Fehler! Verweisquelle konnte nicht gefunden werden.**) to define functionality and tools capabilities. This is a simple subclass hierarchy which defines a (generic) set of data manipulation and bioinformatics analysis types such as “Retrieval”, for retrieval of records from a database, “Parsing”, for the extraction of information from various flat-file formats, “Conversion”, for data-type syntax changes, etc.
- Implement the tool programmatic interface according to Biomoby web access protocol and serialization formats. In particular the Biomoby tools use a SOAP based (i.e. Web Service) communication protocol that is described below.
- Register to the Moby Central registry(-ies) with the appropriate tool description so that categorization and semantics-based discovery of the tool is possible.

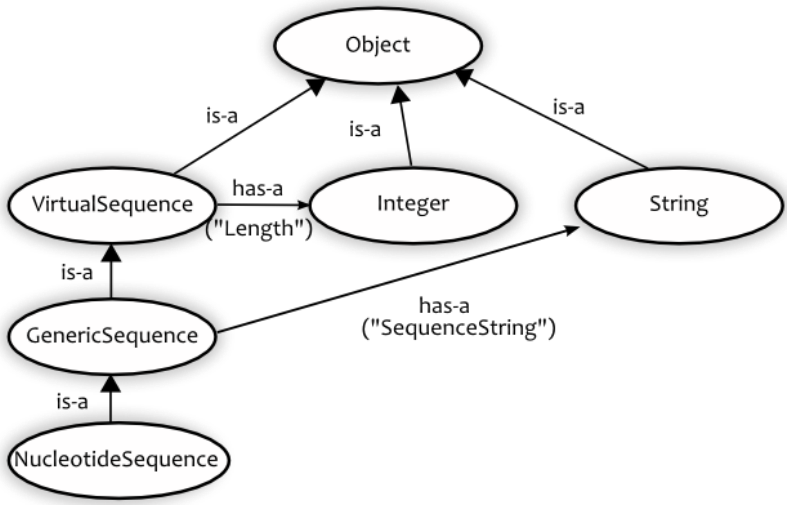


Figure 1 A small part of the BioMOBY object ontology. The Biomoby object ontology is a graph where the nodes represent object classes and supports two types of relationships: ISA, representing subclasses and HAS-A/HAS representing containment. The root (the more general) data type is the Object type and every other class is a direct or indirect subclass of it.

Therefore in order for the ACGT platform to gain access to the Biomoby based bioinformatics tools a “gateway” infrastructure needs to be in place that give access to the MOBY central

registry and "understands" the Biomoby specific protocols and conventions in order for the Biomoby services to be invoked and accessed from inside the ACGT knowledge discovery platform. The details of the implementation of this gateway are given in the following sub sections.

MOBY Central integration

In order to offer an efficient integration mechanism the ACGT-Biomoby gateway uses a local repository of the contents of the MOBY central. The rational is that the contents of the Biomoby registries do not change very often and having a local "cache" of the registry will provide additional opportunities for making optimizations. In general there are a few Biomoby registries available worldwide with different contents but currently the most comprehensive one seems to be that at the University of Calgary, Canada (http://moby.ucalgary.ca/). The contents of this registry were retrieved in RDF format and stored in a local RDF (Sesame[2]-based) repository. This local repository is also augmented by the contents of the Biomoby object, classification (service), and data types (Namespace) ontologies that were also downloaded from the Calgary registry. The relevant web addresses for directly accessing these ontologies are shown in the table below.

Object ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Objects
Service Ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Services
Namespace Ontology	http://moby.ucalgary.ca/RESOURCES/MOBY-S/Namespaces
Services (instances)	http://moby.ucalgary.ca/RESOURCES/MOBY-S/ServiceInstances

Table 1 Web addresses for accessing the BioMOBY service registry in Canada

It's evident that the content of this local repository is not fully synchronized with that of the original MOBY Central. It can be though periodically updated by downloading the content of the MOBY Central in an offline mode and rebuilding the local database.

Invocation of the Biomoby tools

The ACGT-Biomoby gateway is a software component that is also in charge of making the actual communication with the Biomoby tools, i.e. for submitting requests to them and retrieving the results from these invocations. The need for such a "middleman" in the invocation of the Biomoby services is imposed by the Biomoby specific messaging principles.

Normally after retrieving a tool description from the MOBY Central, clients interact with the actual tool by sending a request in the form of an *input* message, and then receiving the response in an *output* message. This request response communication is based on Web Services technologies [2], i.e. the messages are serialized in XML and the communication protocol is SOAP. But SOAP and XML define the envelope of the input and output messages: the "payload" of the message is formatted in compliance with the Biomoby conventions. These conventions explicitly define the syntax of the input messages and the output messages in a way that is strongly connected with the data types of the input parameters and the results as defined in the Biomoby Object ontology.

[2] http://www.openrdf.org/

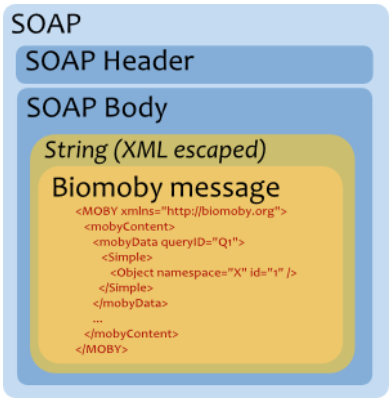


Figure 2 The "Russian doll"-type format of the Biomoby messages

A high level view of a Biomoby message "on the wire" can be seen in Figure 2. The SOAP message contains, in its body, a string encoded Biomoby message. In general this Biomoby message consists of (Figure 3):

- An outermost XML tag "MOBY", identifying the message as a Biomoby formatted message.
- A child XML element "mobyContent" within which formatted service provision information can be placed such as database version, software name and version, etc. In addition, this element is the container for any error reporting and exception information that the service provider wishes to pass back to the client.
- One or more "mobyData" XML elements within "mobyContent", which contain:
- Zero or more instances of a BioMoby Object being passed into or out of the service. These are the input parameters submitted in an input message or the results in an output message.

```
<MOBY xmlns="http://biomoby.org">
  <ProvisionInformation>
    ...
  </ProvisionInformation>
  <mobyContent>
    <mobyData queryID="Q1">
      <Simple>
        <Object namespace="X" id="1" />
      </Simple>
    </mobyData>
    <mobyData queryID="Q2">
      <Simple>
        <Object namespace="X" id="2" />
      </Simple>
    </mobyData>
  </mobyContent>
</MOBY>
```

Figure 3 The serialization of the Biomoby input message

We have identified a number of problems with the Biomoby messaging formats and protocols when used in a state of the art service oriented software architecture:

- The whole message as shown in Figure 2 presents an “onion” layered mix of encodings. The Biomoby specific message with its parameters etc. is encoded as a String that is further encapsulated into a SOAP call.

- Some of the existing functionality of SOAP and web services is duplicated. For example, errors are sent inside the Biomoby message instead of declaring the possible error conditions in the WSDL and using the standard fault mechanisms of SOAP.

- Interaction with the Biomoby services requires knowledge of these serialization conventions and the Biomoby specific ontologies so no generic Web Service clients can be used.

- The web service description (WSDL) of a Biomoby service does not provide enough type information for making possible the interoperability with modern Web Service platforms. In particular the WSDLs of all Biomoby tools declare operations that accept a single string input but of course this cannot be any string: it should be a Biomoby message encoded as a string.

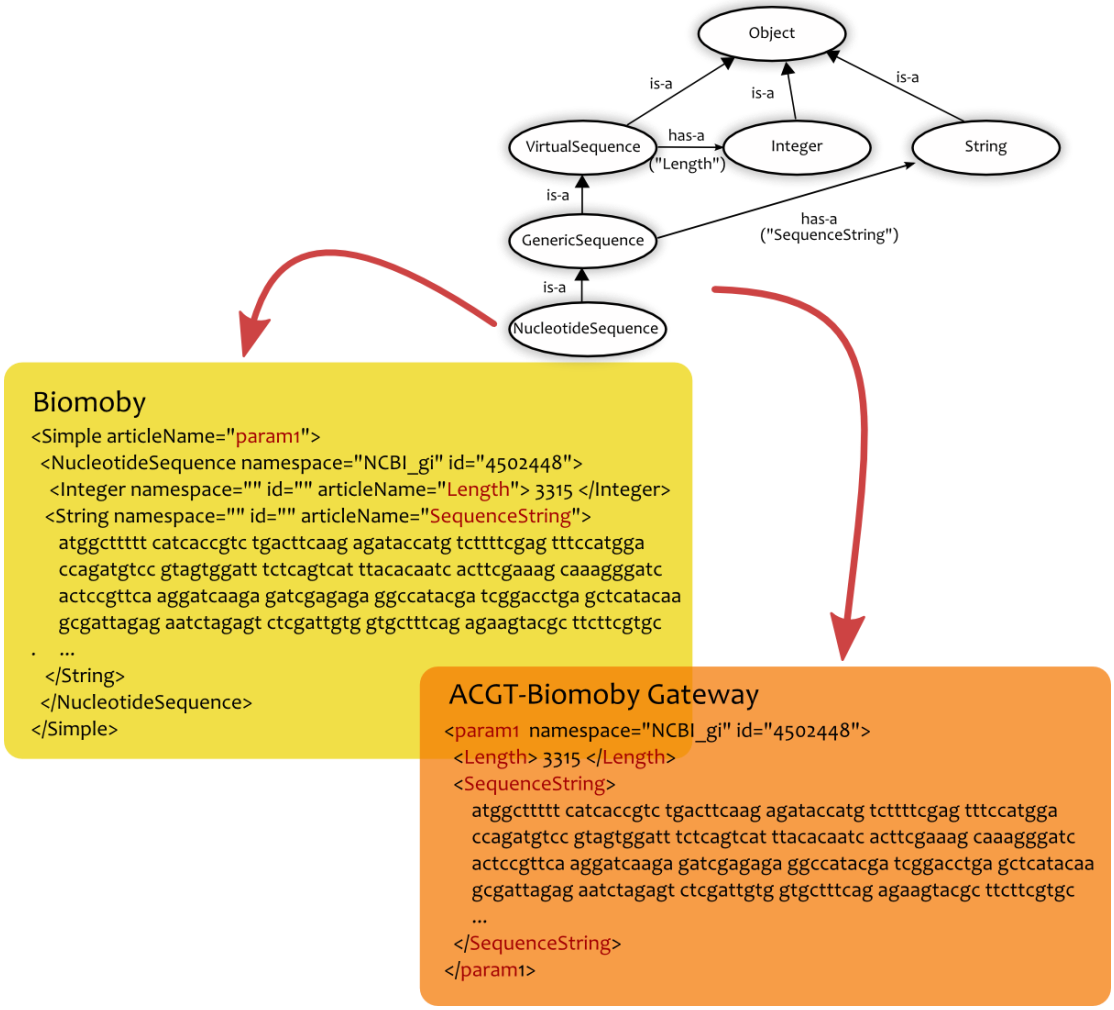
In conclusion, the main problem introduced by the Biomoby API in a Service Oriented Architecture such as the ACGT platform is that the machine readable descriptions of the Biomoby services are not type safe and compliant with the existing genetic Web Services tools and infrastructures. So in order to make the Biomoby services accessible as normal web services through the ACGT knowledge discovery and workflow execution environment. The objective is to build an ACGT-Biomoby gateway that presents a standard web service interface to the rest of the ACGT platform. In order to do so all the necessary information about the parameters and data types of each Biomoby service are explicitly defined in dynamically created WSDL documents. Therefore the mapping to WSDL that is performed by the gateway includes the following

- Each parameter of the Biomoby service, be it “primary” or “secondary”, is explicitly defined as an input parameter of the SOAP call. This is also the case for the results of the invocation.

- Each input and output data type is fully described in an XML schema that is included in the WSDL. Biomoby defines how each type (node) in the Object ontology should be serialized in a transitive way based on its super types and the “contains” relationships that itself or its super types may have. The XML schema produced is also compliant to the semantics of the Object ontology (i.e. there’s mo loss of information) but the actual syntax slightly differs. Each Biomoby object type is represented by an XML Schema *elementType* while the containment properties (e.g. “SequenceString” in the example of Figure 1) are serialized as XML elements. This arrangement makes possible to use a more specific data type in place of one of its super types because “content-wise” they have the same structure (with the possibility of the more specific type to have more properties, but not less). Figure 4 shows an example of the differences between the two syntaxes. It is worth mentioning that in the case of the Gateway the same XML fragment can be used for both the NucleotideSequence and its super type GenericSequence because they have the same “Length” and “SequenceString” properties.

- Due to the way the mapping of the Object ontology to the XML Schema is performed the explicit semantic information is lost. For example the Gateway's XML fragment in Figure 4 there's no indication that the "param1" object is a NucleotideSequence. Nonetheless, this semantic information still exists in the Gateway's WSDL. Every input parameter and result is annotated with its Biomoby Object type through the means of Semantic Annotations for WSDL and XML Schema (SAWSDL [4]).

Following the above design decisions the ACGT-Biomoby Gateway presents a Biomoby service-specific WSDL and makes the appropriate transformations and mappings between the two messaging paradigms. For the rest of the ACGT platform each Biomoby service appears to be a standard Web Service that is also semantically described using the SAWSDL annotations inside the WSDLs. On the other hand in the Biomoby world the



Gateway presents itself a Biomoby compliant client application.

Figure 4 The Biomoby original data type serialization compared to the one of the ACGT-Biomoby gateway. Both represent the XML syntax for a parameter called "param1" of the NucleotideSequence type.

References

[1] M.D. Wilkinson and M. Links. BioMOBY: An open source biological web services proposal. Briefings in Bioinformatics, 3(4):331-341, 2002.

[2] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: An introduction to Soap, WSDL, and UDDI. IEEE Internet Computing, 6(2):86-93, 2002.

[3] Wilkinson M.D. et. al Interoperability with Moby 1.0 - it's better than sharing your toothbrush!, Brief Bioinform. 2008 May;9(3):220-31

[4] Semantic Annotations for WSDL and XML Schema, W3C Recommendation 28 August 2007, http://www.w3.org/TR/sawsdl/

3. Interoperability with Command Line Programs

Considering technological advances in high-throughput sequencing [66] performing clinical trials require processing and analysis of massive data sizes. Distributed and advanced parallel computing has been suggested as an effective solution to process such kind of data. Grid-computing is, in particular, suitable for these tasks because of the inherent ability to effectively use computational resources.

Proliferation and profusion of services in the field of bioinformatics [67-69] and the use of Web services technology as the option to deploy of these tools causes new challenges for service discovery and composition (using results of service invocations as input to other services).

As a result, on one hand, command line programs (CLP) are one the favorite style for the deployment of computational services inside a grid infrastructure. On the other hand, the standard definition of Web-Services allows automatic intercommunication between services, augmenting with important advances his interoperability from a semantic point of view. To better exploit both, the high number of CLP available and the advantages of standard definitions, we develop a general interface using Axis [70] as SOAP engine and Tomcat [71] as a Servlet container. Once wrapped in this interface, the tools are registered in the ACGT Metadata Repository. The use of these new tools inside the secure ACGT Grid environment meant that some issues had to be taken into account.

Integration in ACGT architecture

According to a layering design pattern, ACGT architecture is distributed in:

- User access, the highest level layer, including editors, portals, dedicated clients, etc.
- Business process services, covering knowledge discovering, ontology access, data mediation and whatever high level service required by clients.
- Advanced Grid middleware [72].
- Common Grid infrastructure, using Globus Toolkit [73].
- Hardware layer, computational resources, networks, databases, etc.

Among these layers, Generic Axis Web Service makes use of the advanced Grid middleware, thus enabling the access to the secure environment, data management and execution inside the Grid.

Secure access

Available resources in the Grid (hosts, data, instruments etc.) are accessed by many different users from many different organizations and places. In the low level of the Grid, Globus Toolkit provides a Grid Security Infrastructure (GSI [74] to overcome those issues, providing authentication, credential delegation and transport/message-level security.

ACGT make use of the Gridge Authorisation Service (GAS) to support authorisation operations in Grid space. GAS is used in our CLPs to request credentials of users who want use them. These credentials are used to control the delegation of rights among different components of the architecture.

Using file system

User's credentials can be delegated in the secure environment to request Data in the Management Suite (DMS). This mechanism centralizes all the data, providing fast access, management of large amount of data. DMS also provides the possibility of managing metadata about data directly from tools (annotation of execution times, author, dates, etc) making use of different metadata schemas, like Dublin Core or new schemas defined by the user.

Job execution

To manage the whole process of remote job submission, our tools use Gridge Resource Management System (GRMS), which has an interface to launch, resume and monitor jobs.

Registering in ACGT Metadata Repository

Generic Web Service allows execution of command line programs correctly registered in ACGT Metadata Repository. Register must be carry out in the ACGT portal using Metadata registration Portlets.

To register new CLPs, we need to:

- Previously register data types of inputs/outputs (if needed).
- Select functional category of the tool (register new functional categories if needed)
- Register tool metadata, including documentation, and Author info.
- Register tool-specific information (Tool Location). This includes type of tool, host and other info necessary for execution.

To add a new tool, we must select a functional category in the portal (left tree in metadata registration Portlet at http://rd.siveco.ro/acgt, see figure 1)

Figure 1. Adding new tools

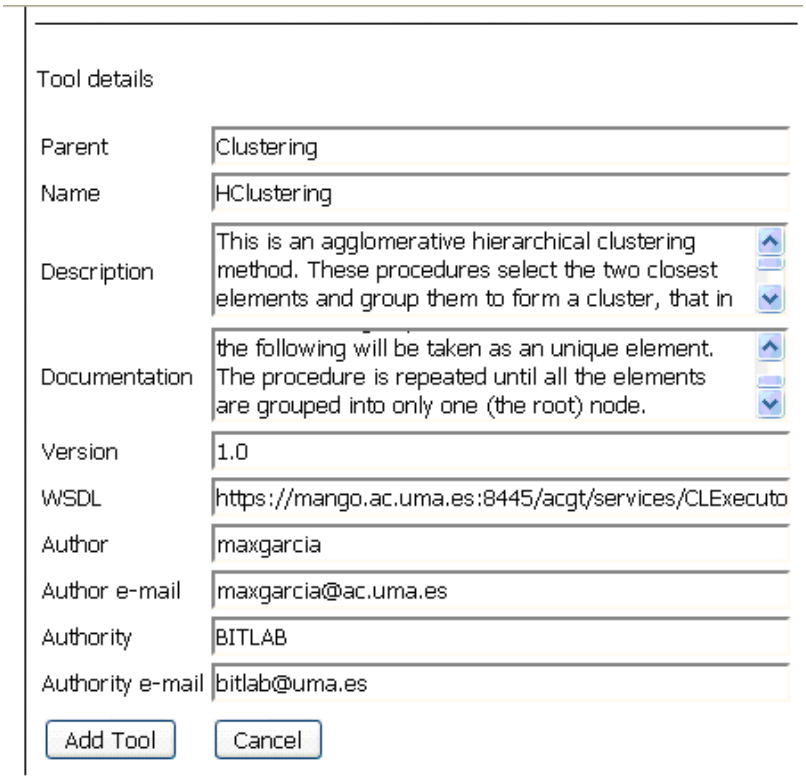


Figure 2. Tool metadata registering

- Name: Name to identify the tool. Will be used to generate a unique URI for this tool.
- Description: Short comment about the tool.
- Documentation: Long description of the tool.
- Version: Version of registered tool.
- WSDL: WSDL text can be used in this field. Alternatively a link to WSDL can be used.
- Author: Tool author name. This field is automatic generated in the Portlet with user credential info.
- Author E-Mail: Mail of author.
- Authority: Institution of author.
- Authority E-mail: E-mail of institution.

Next step is to create operations for the tool (see figure 3)

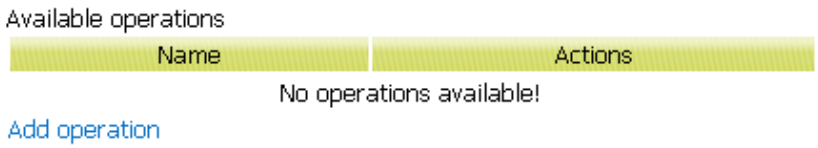


Figure 3. Adding operations

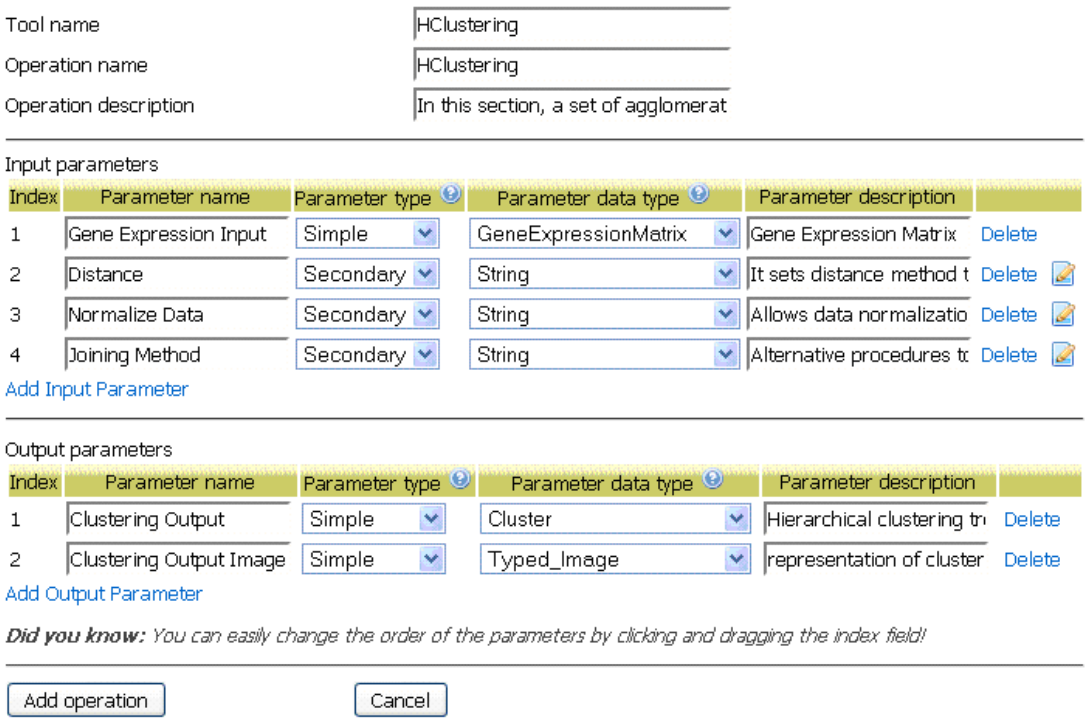


Figure 4. Adding operation info and parameters

For operation, the user must enter:

- Operation name: Suitable name for operation.
- Description. Short description.

Input and output parameters can be added. In the case of input parameters, we take into account the case of secondary type parameters, case of thresholds, options, etc. In these secondary parameters we can include default and allowed values. In the case of input and output files, we can use Simple or Array type, to use single files or collection of files. For each file, data type and description must be specified.

In the case of command line tools, is quite important the order of parameters, which is displayed in the first column of figure 4.

Finally, we must include tool location information for a correct execution.

In tool location information must be specified the host where command line is available. We select *GRIDCLSERVICE* as a type and for extra values we add the URL for generic Web Service which will execute the command line. A relative path is also necessary (see figure 5).

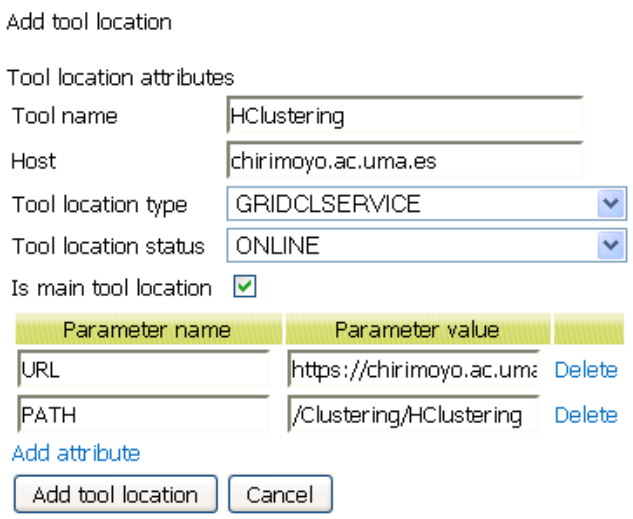


Figure 5. Adding Tool Location info

Web Service Interface

The Generic Axis Web Service allows execution of whichever CLP correctly registered in ACGT Metadata Repository (AMR). This Web service can be invoked from any client making use of metadata recovered from AMR. There are three operations available:

runAsyncDMS

This operation launches the execution. Necessary parameters are:

- **Host**. With this server name we decide where the CLP is executed. The command Line program must be available in the server.
- **Path.** This parameter contains the relative path of the program.
- **InputFileIDs.** We include the DMS identifiers of files to be used during the execution. This is an array of arrays which allows the use of collections of data as an input.
- **Parameters.** Parameters are included in an array of strings.
- **OutputInfo.** Array of arrays of Strings. For each output file must be provided:
    - Output file name
    - Mime type of file
    - Semantic Data Type
    - Type of output (Simple or Array)
- **FolderDMS**. A destination folder for output. If null, user home folder in DMS is used.

This operation returns a string with the current job identifier. Example of the method in Java code:

```
String runAsyncDMS(String host,
                   String path,
```

```
int[][] inputFileIDs,
String[] parameters,
String[][] outputInfo,
String folderDMS)
```

getJobStatus

This operation returns the current status of job in the Grid. Necessary parameters are:

- **JobID**. This parameter is provided as output of the **runAsyncDMS** operation.

This operation returns a string with the current status of job. Returned values are *Running*, *Finished* or *Failed*. Example of the method in Java code:

```
String getJobStatus(String jobId)
```

getResultDMS

This operation returns DMS identifiers of results when job is finished. Necessary parameters are:

- **JobID**. This parameter is provided as output of the **runAsyncDMS** operation.

Returning value is an array of arrays, allowing the use of collections as output. Example of the method in Java code:

```
String getResultDMS(String jobId)
```

CLPs also have similar operations to *runAsync* and *getResult* where the data can be directly sent as part of the calls. However, in ACGT it is mandatory to use DMS as file storage.

CLP Life cycle

Including new CLPs in the Grid is carried-out in two main steps: First, install the CLP in available servers in the Grid and register service metadata in ACGT Metadata Repository. Once registered, service metadata is used to discover and invocate the CLP in the ACGT environment.

Registering. Registering is performed in the ACGT portal at http://rd.siveco.ro/acgt. Metadata details are given in section 3. Providers first need to check that necessary data types and functional categories are available, otherwise they must be registered first. Metadata related to invocation include parameter and tool location metadata. The parameter information is later used by Clients to automatically build-up service interfaces. Details such us input/output files, optional parameters; tool-location including endpoint information are also provided in this registration step.

Discovering. Client applications with the correct user credentials may access the metadata repository to discover tools based on descriptions, input/output datatypes, functional categories etc. Service discovery is typically performed using the Magallanes application (available via the ACGT portal).

Generic Web service for CLP. This Web service is the default Web service for executing CLPs. The CLPs themselves are registered as abstract tools, each with the endpoint to the actual web-service (the generic Web service). This service has been developed using Axis as SOAP [75] engine and use Tomcat as a Servlet container. Servlet container must compliant with secure access restrictions of ACGT. Currently, this web-service is installed in two servers in the ACGT Grid. The client can launch and control execution using three available operations in the generic Web service (see figure 6).

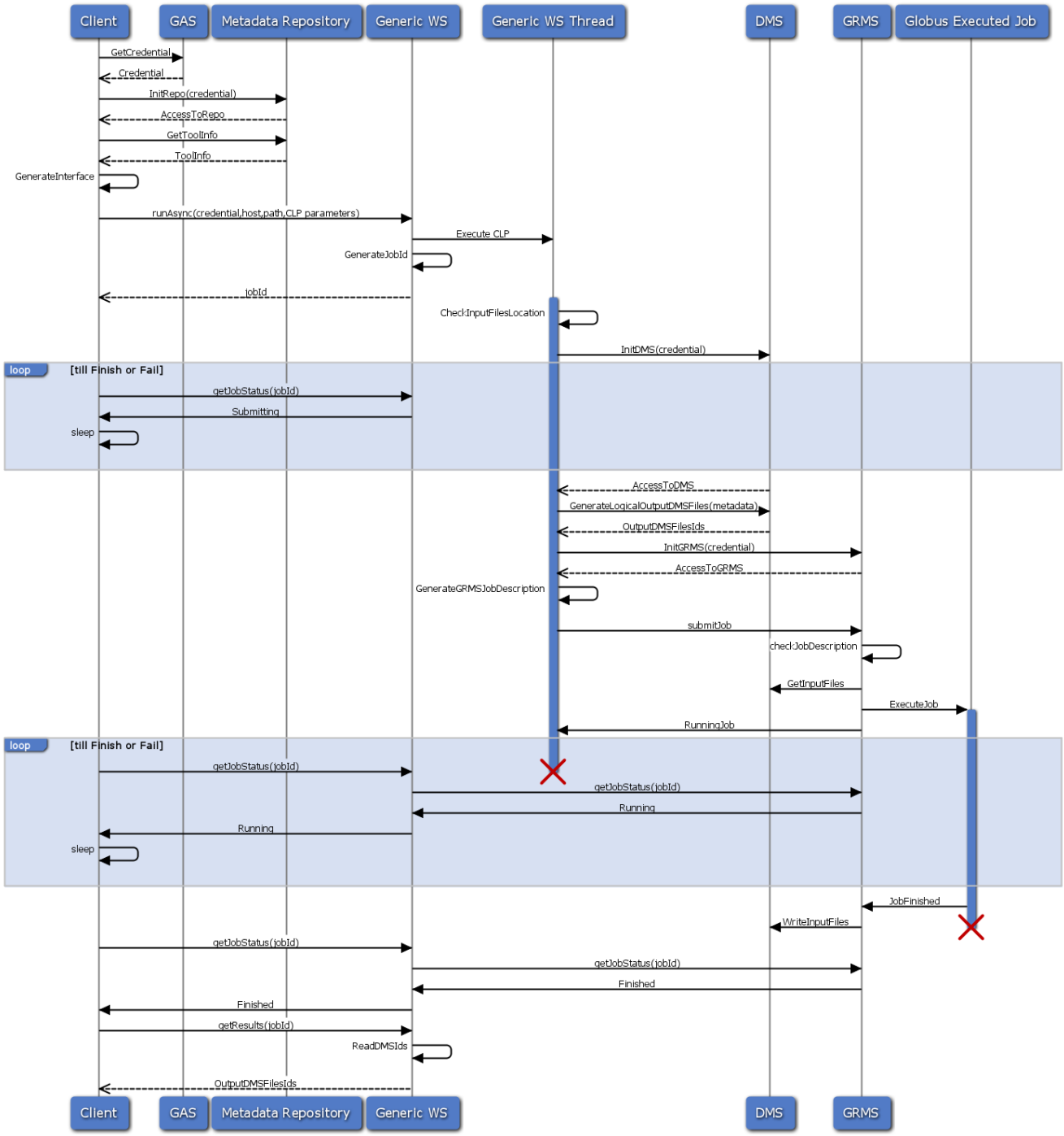
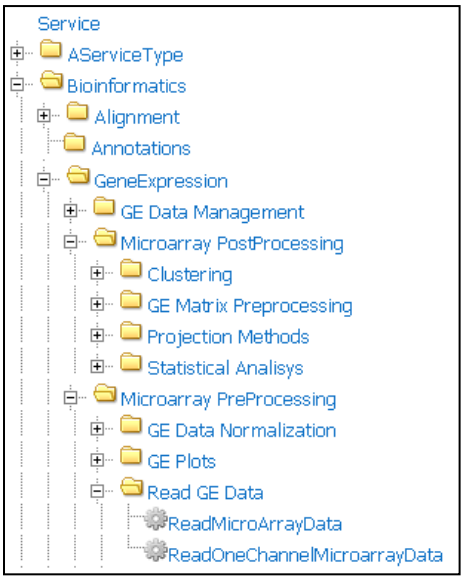


Figure 6. Shows sequence diagram of events between the software components. Assuming a scenario where user is correctly identified with his credentials, client program gets tools metadata from ACGT Metadata Repository. With this info, client has enough information to build a user interface in the application. Client program requests user data in the interface, and creates runAsync call to generic Web service. In this operation, a jobId is generated to be used in next operations. The generic WS then retrieves the necessary info of DMS files and creates a Job description to be submitted with GRMS software in the Grid. Client application can monitor the current status of the Job using getJobStatus operation. When status is *Finished*, client can retrieve results using GetResults operation.

Available command Lines for Gene Expression data analysis



We have deployed and registered in the repository about 30 services in the field of gene expression data processing. We have organised these tools in three main functional categories:

- Microarray Preprocessing. Covering gene expression data read and analysis for different data formats.
- Gene expression (GE) data management. Allowing creation of GE matrix, data uploading, etc.
- Microarray Postprocessing. Once GE matrix is created, we can perform different methods of analysis in the set of experiments.

Pre-processing group contains methods to spot filtration, inter and intra slide normalization, duplicate resolution, dye-swapping, error removal and statistical analyses. Additionally, it contains two unique implementation of the procedures – double scan and Supervised Lowess-, a complete set of graphical representations – MA plot, RG plot, QQ plot, PP plot, PN plot – and can deal with many data formats, such as tabulated text, GenePix GPR and ArrayPRO. The “Create matrix” CLP has been implemented to create different kind of matrixes, the format determines which values to select in the in microarray source files [37].

Create matrix method has been implemented with the possibility of create different kind of matrixes, depending on a format which selects the correct values in microarray source files.

Postprocessing branch integrates a variety of analysis tools for visualizing, pre-processing and clustering expression profiles. Preprocess matrix method allows multiple operations to the data in order to accommodate it for further analysis. For example, filters to select genes of interest, normalization, logarithm transformation, treatment of missing data and others. Once we have GE matrix data properly transformed, several clustering (hierarchical, k-means, fuzzy methods, etc.) and projection (PCA and non-linear techniques like Sammon mapping and Self-Organizing Maps) techniques can be applied [38].

Workflows

Services of gene expression data analysis can be combined to generate different workflows. An example line of execution is showed in figure 7 and 8.

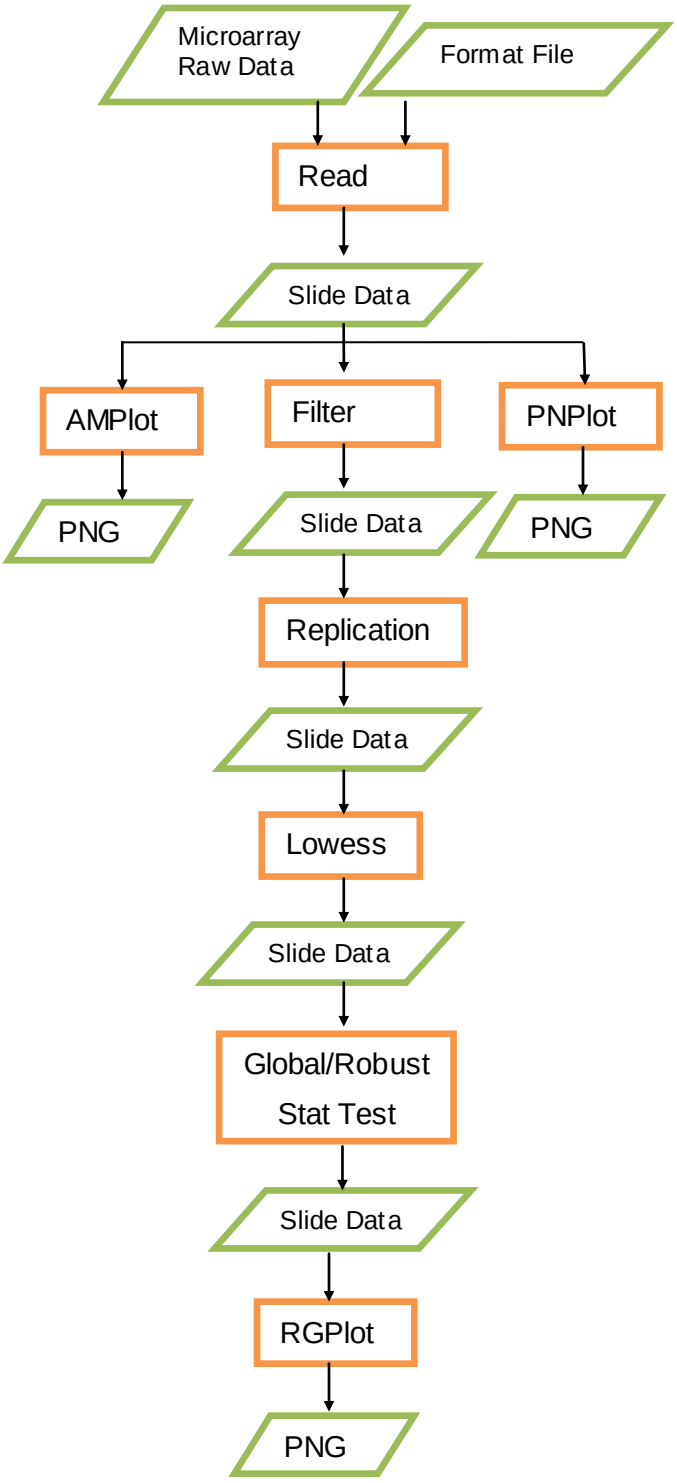


Figure 7. Possible gene expression workflow using preprocessing methods.

Once the first workflow generates different slides files, we can use them for GR matrix workflows.

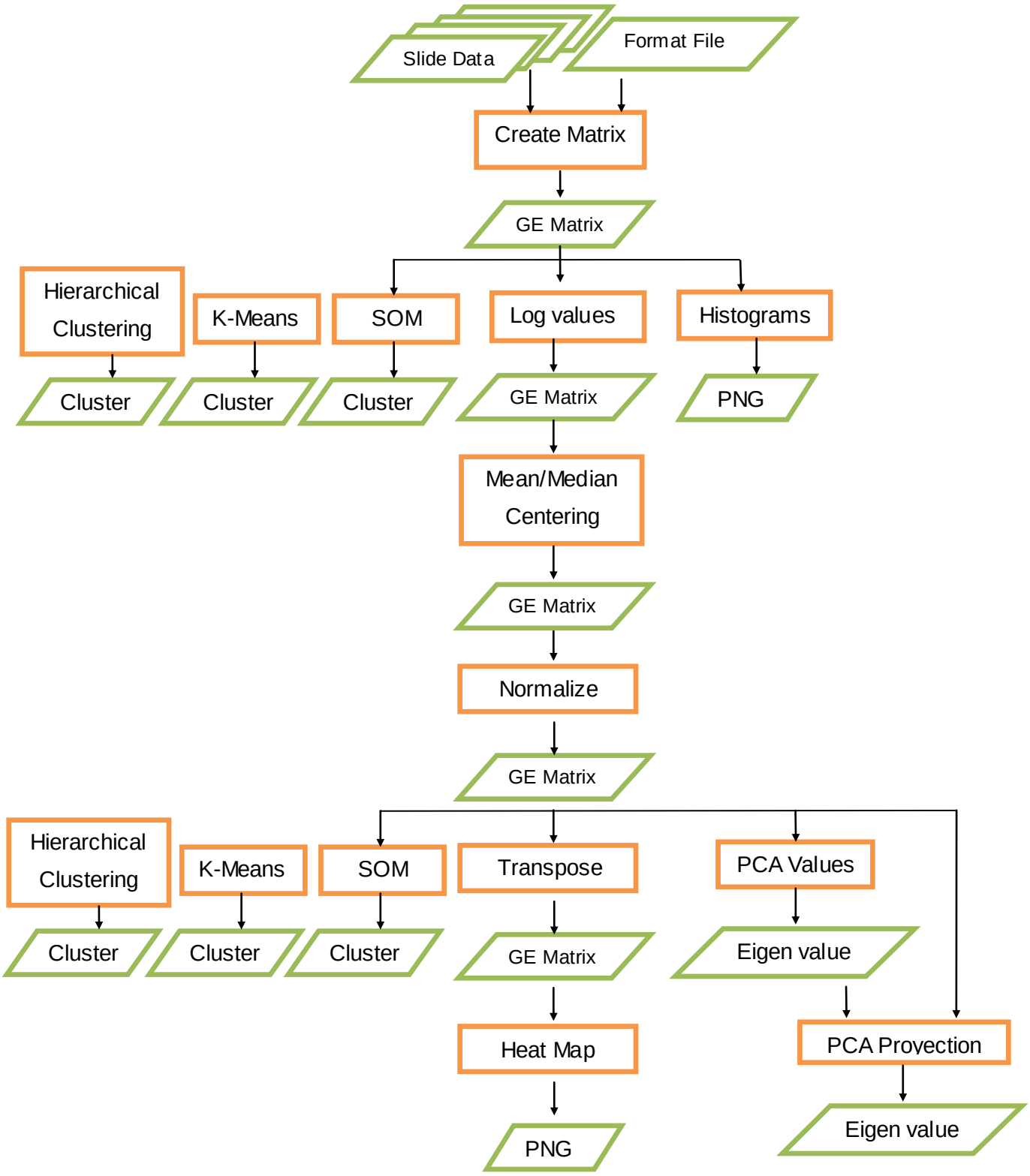


Figure 8. Example of possible steps in postprocessing of GE Matrix.

4. Interoperability with R Scripts

The GridR tool of ACGT has been extensively covered in deliverables D6.2 and D6.4. Hence, in the following we will only shortly summarize the main features of GridR and refer the reader to these deliverables for the technical details. Main features of GridR include:

- **Execution of R scripts through the generic GridR Service**. The GridR service is an ACGT-compliant service that allows to transparently view R scripts as ACGT services. This is implemented by passing the R code to the generic GridR service as a parameter. The GridR service automatically takes care of data input and output to and from the ACGT Data Management System (DMS) in the Grid. It executes R code by generating dedicated Grid jobs and submitting them to the Grid Resource Management System (GRMS) which is on a lower level of the ACGT architecture (cf. D3.4). The GRMS takes care of the distribution, execution, and monitoring of the actual data processing jobs. Security is also guaranteed by the Grid system.
- **Parallel execution of R scripts:** The GridR Service offers the feature to parallelize R scripts. Using specially formatted comment strings in the R code, the user specifies which code to execute in parallel. From this information, the GridR service automatically generated multiple Grid jobs which are submitted to the GRMS in parallel, thus making full use of the cluster architectures upon which ACGT system is set up. This feature avoids having to implement parallelism on the level of the workflow, which significantly simplifies the construction and re-use of data analysis workflows.
- **Accessing the ACGT environment from within R using the GridR Package:** This package, implemented in the R language itself, allows the R programmer to access ACGT services, e.g. the mediator, from inside an R script. This feature has been implemented in reaction to the observation that often bioinformaticians are skilled R developers and prefer to interface the ACGT system in the language they are used to. This simplifies data analysis in ACGT for R programmers, as they do not have to switch between two levels of programming (the workflow level and the script level)
- **Developing R scripts inside the ACGT portal using the GridR Portlet:** The GridR Portlet allows to interactively develop new R scripts inside the ACGT portal. The user can open multiple R sessions, read and write data from the DMS, and directly access other ACGT components using the GridR Package. The increased interactivity from the GridR Portlet simplifies the development of new scripts in GridR, as the time-consuming scripts of deploying R scripts for the GridR Service, executing the analysis workflow and waiting for the results can be circumvented.

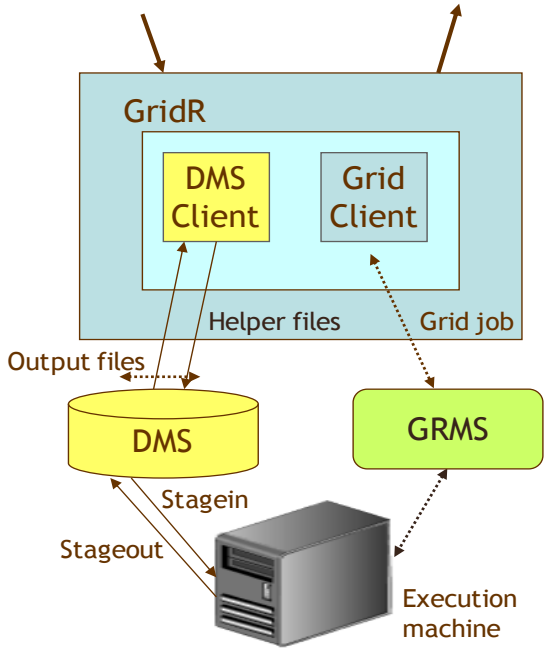


Figure 5: Architecture of the GridR Service

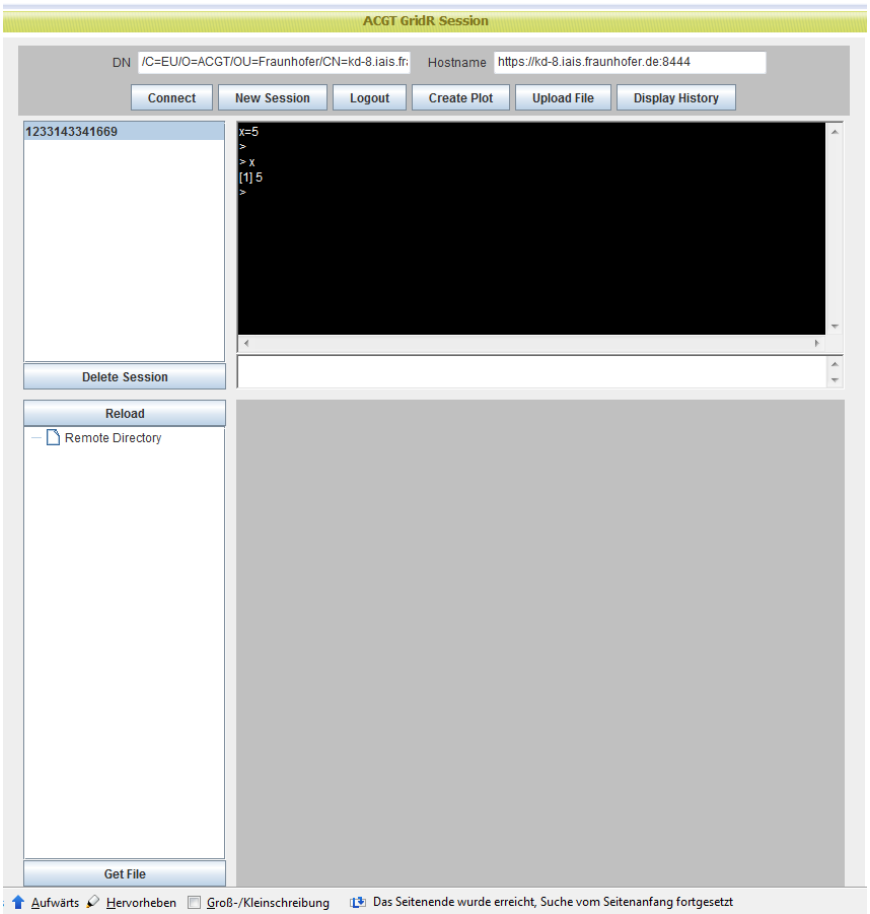


Figure 6: Screenshot of GridR Applet

5. Conclusions

The ACGT data analysis environment consists of the integration of both specialized and generic data analysis services in the overall architecture and user interface of the ACGT system. In ACGT, the open-source statistical toolkit / language R and the open-source suite of bioinformatics web-services BIOMOBY have been identified as the most relevant tools for integration, both because of their wide-spread popularity among the researchers, and because of the complementarity of the approaches (interactive programming language vs. web-services). In addition, many researchers develop new tools outside of established frameworks. To simplify the integration of these tools, and to lower the threshold for new users to add tools to ACGT, a way to integrate tools that work on the most simple of all interfaces, namely the command line, has been developed. In summary, these three, largely complementary ways of developing and integrating new ACGT services covers the state of the art in software engineering in scientific disciplines.

6. References

[1] Seluja GA, Farmer A, McLeod M, Harger C, Schad PA: Establishing a method of vector contamination identification in database sequences. Bioinformatics 1999, 15:106-110.
[2] Coker JS, Davies E: Identifying adaptor contamination when mining DNA sequence data. Biotechniques 2004, 37:194-198.
[3] Chen YA, Lin CC, Wang CD, Wu HB, Hwang PI: An optimized procedure greatly improves EST vector contamination removal. BMC Ge nomics 2007, 8:416.
[4] Scheetz TE, Trivedi N, Roberts CA, Kucaba T, Berger B, Robinson NL, Birkett CL, Gavin AJ, O'Leary B, Braun TA, Bonaldo MF, Robinson JP, Sheffeld VC, Casavant MBSTL: ESTprep: preprocessing cDNA sequence reads. Bioinformatics 2003, 19:1318-1324.
[5] White JR, Roberts M, Yorke JA, M P: Figaro: a novel statistical method for vector sequence removal. Bioinformatics 2008, 24:462-467.
[6] Bonfield JK, Smith K, Staden R: A new DNA sequence assembly program. Nucleic Acids Res 1995, 23:4992-4999.
[7] Chou HH, Holmes MH: DNA sequence quality trimming and vector removal. Bioinformatics 2001, 17:194-198.
[8] Li S, Chou HH: LUCY2: an interactive DNA sequence quality trimming and vector removal tool. Bioinformatics 2004, 20:3657-3665.
[9] TIGR: SeqClean. [http://compbio.dfci.harvard.edu/tgi/software/]
[10] Hotz-Wagenblatt A, Hankeln T, Ernst P, Glatting KH, Schmidt ER, Suhai S: ESTAnnotator: A tool for high throughput EST annotation. Nucleid Acids Res 2003, 31:3716-3719.
[11] Lee B, Hong T, Byun SJ, Woo T, Choi YJ: ESTpass: a web-based server for processing and annotating expressed sequence tag (EST) sequences. Nucleic Acids Res 2007, 35:W159-162.
[12] Nagaraj SH, Deshpande N, Gasser RB, Ranganathan S: ESTExplorer: an expressed sequence tag (EST) assembly and annotation platform. Nucleic Acids Res 2007, 35:W143-147.
[13] Ewing; Hillier, Wendl and Green; (1998) “Base-Calling of Automated Sequencer Traces using Phred. I. Accuracy Assessment”; Genome Research 8-175
[14] Phrap, The assembler: www.phrap.org
[15] Gordon, D.; Abajian, Ch. And Green Phil. (1998), “Consed: a graphical tool for sequencing finishing. Genome Research 8, 195-202
[16] Guigo, R., and Fickett, J. W. (1995). Distinctive sequence features in protein coding, genic noncoding, and intergenic human DNA. J. Mol. Biol. 253: 51–60.
[17] Pérez,AJ.; Thode G., and Trelles, O.; "AnaGram: protein function assignment"; Bioinformatics Vol. 20 no. 2 2004, pages 291-292; ISBN 1367-4803
[18] Pearson WR, Lipman DJ: Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences, 1988, 85(8):2444–2448
[19] Altschul, S. F., et. al. (1997); Gapped BLAST and PSI-BLAST: a new generation of protein database search programs» Nucleic Acid Res. Vol. 25. pp. 3389-402. PMID 9254694.
[20] Needleman, S. B. & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443-453
[21] Smith, T. F. & Waterman, M. S. (1981). Identification of common molecular subsequences. J. Mol. Biol. 147, 195-197.
[22] Agarwal, P and States, D.J. (1998) Comparative accuracy of methods for protein-sequence similarity search; Bioinformatics, 14, 40-47.
[23] Brenner, S.E. et al. (1998) Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. roc. Natl. Acad. Sci. U. S. A. 95, 6073–6078
[24] Pearson, W.R. (1995). Comparison of Methods for Searching Protein Sequence Databases. Protein Science 4, 1145-1160.

[25] A Bairoch, P Bucher, K Hofmann - Nucleic Acids Research, 1996 - ncbi.nlm.nih.gov
[26] [6] Trelles O., Andrade M.A., Valencia A., Zapata E.L., and Carazo J.M.; “Computational Space Reduction and Parallelization of a new Clustering Approach for Large Groups of Sequences”; Bioinformatics vol.14 no.5 1998 (pp.439-451)
[27] Jyun-Sheng Chang, Andrew Chang, Tsuey-Fen Lin, Sur-Jin Ker, August 1992; Proceedings of the 14th conference on Computational linguistics - Volume 3 , Volume 3
[28] Fann CSJ, Ott J (1995) Parsimonious estimation of sex-specific map distances by stepwise maximum likelihood regression. Genomics 29, 571-575
[29] Corpet,F. (1988) Multiple sequence alignments with hierarchical clustering. Nucleic Acids Res., 16, 10881–10890.
[30] Gotoh,O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. Comput. Applic.Biosci., 9, 361–370.
[31] Miller,W. (1993) Building multiple alignments from pairwise alignments.Comput. Applic. Biosci., 9 169–176.
[32] Cavalli-Sforza, L.L. and A.W.F. Edwards. 1967. Phylogenetic analysis: models and estimation procedures. American Journal of Human Genetics 19:233–257.
[33] Felsenstein, J. 1981a. Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Molecular Evolution 17: 368-376.
[34] Kaminski S, et al., Molecular Dynamics Simulations of the Chromophore Binding Site of Deinococcus radiodurans Bacteriophytochrome Using New Force Field Parameters for the Phytochromobilin Chromophore. J Phys Chem B. 2009
[35] http://www.ebi.ac.uk/pdbe/emdb/
[36] Carazo, J. M.; et al.(1999); "Organising multi-dimensional biological image information: The BioImage Database"; Nucleic Acids Research Volume 27, Number 1 Pp. 280-283
[37] Orengo, C.A., Brown, N.P., Taylor, W.T. (1992), "Fast structure alignment for protein databank searching", Proteins, 14:139-167
[38] Fisher, D., Bachar, O., Nussinov, R. and Wolfson, H. (1992), "An efficient automated computer vision based technique for detection of three dimensional structural motifs in proteins", J.Biomol.Struct.Dyn. 9:769-789
[39] Holm, L. and Sander Ch. (1993), "Protein structure comparison by alignment of distance matrices", J.Mol.Biol. 233:123-138
[40] Shindyalov, I.N, and Bourne, P.E. (1998), "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path", Protein Engineering 11 (9) 739-747
[41] Schena M, Shalon D, Davis RW, Brown PO. (1995); "Quantitative monitoring of gene expression patterns with a complementary DNA microarray"; Science. 1995 Oct 20;270(5235):467-70.
[42] Victoria Martin-Requena, Antonio Muñoz-Merida, M Gonzalo Claros and Oswaldo Trelles; PreP+07: improvements of a user friendly tool to preprocess and analyse microarray data; BMC Bioinformatics 2009, 10:16doi:10.1186/1471-2105-10-16
[43] Garcia de la Nava J, Santaella DF, Cuenca Alba J, Maria Carazo J, Trelles O, Pascual-Montano A: Engene: the processing and exploratory analysis of gene expression data. Bioinformatics 2003, 19:657-658. (http://www.engene.cnb.uam.es)
[44] R.F. Doolittle, M.W. Hunkapiller, L.E. Hood, S.G. Devare, K.C. Robbins, S.A. Aaronson and H.N. Antoniades, Simian sarcoma virus onc gene, v-*sis*, is derived from the gene (or genes) encoding a platelet-derived growth factor, *Science* 221 (1983), pp. 275–277.
[45] Rechenmann, F. (2000), “From Data to Knowledge”, Bioinformatics, v.16.n5 pp 411
[46] Galperin MY., Cochrane GR. (2009) Nucleic Acids Research annual Database Issue and the NAR online Molecular Biology Database Collection in 2009. *Nucleid Acids Research* (Database issue):D1-4.
[47] Fox J., Brown T., McMillan S. and Ouellette F. (2008) 2008 update on the Bioinformatics Links Directory. *Nucleic Acids Research*, Vol. 36, No. suppl_2 W2-W4
[48] Wilkinson M.D. (2006) Gbrowse moby: a web-based browser for BioMoby services. *Source Code for Biology and Medicine* 1: 4.

[49] Navas-Delgado I, et al. (2006) Intelligent client for integrating bioinformatics services. *Bioinformatics*, 22, 106–111.
[50] Gordon P.M.K., Sensen C.W. (2007) Seahawk: Moving Beyond HTML in Web-based Bioinformatics Analysis. *BMC Bioinformatics* 8:208.
[51] Carrere S, Gouzy J. (2006) Remora: a pilot in the ocean of BioMoby web-services. *Bioinformatics*, 22, 900–901.
[52] Oinn T, et al. (2004) Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20 : 3045–3054.
[53] K. Karasavvas, R. Baldock, and A. Burger, "Bioinformatics integration and agent technology," Journal of Biomedical Informatics, vol. 37, no. 3, pp. 205–219, 2004.
[54] F. Achard, G. Vaysseix, and E. Barillot, "Xml, bioinformatics and data integration," Bioinformatics, vol. 17, no. 2, pp. 115–125, 2001.
[55] I. Letunic, R. Copley, S. Schmidt, F. Ciccarelli, T. Doerks, J. Schultz, C. Ponting, and P. Bork, "Smart 4.0: towards genomic data integration," Nucleic Acids Research, vol. 32, no. Database Issue, p. D142, 2004.
[56] A. Siepel, A. Farmer, A. Tolopko, M. Zhuang, P. Mendes, W. Beavis, and B. Sobral, "Isys: a decentralized, component-based approach to the integration of heterogeneous bioinformatics resources," Bioinformatics, vol. 17, no. 1, pp. 83–94, 2001.
[57] Z. Lacroix, "Biological data integration: wrapping data and tools," IEEE Transactions on information technology in biomedicine, vol. 6, no. 2, pp. 123–128, 2002.
[58] J. Li, J. Gao, J. Dong, W. Wu, and Y. Hou, "A metadata registry for metadata interoperability," Data Science Journal, vol. 6, no. 0, pp. 379–384, 2007.
[59] C. Goble and D. De Roure, "myexperiment: social networking for workflow-using e-scientists," Proceedings of the 2nd Workflows in support of large-scale science.1em plus 0.5em minus 0.4emACM New York, NY, USA, 2007, pp. 1–2.
[60] T. Bellwood, L. Clement, C. Von Riegen et al., "Uddi version 3.0. 1: Uddi spec technical committee specification," Retrieved November, vol. 15, p. 2004, 2003.
[61] C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood, "A suite of daml+ oil ontologies to describe bioinformatics web services and data," International Journal of Cooperative Information Systems, vol. 12, no. 2, pp. 197–224, 2003.
[62] M. Wilkinson, M. Senger, E. Kawas, R. Bruskiewich, J. Gouzy, C. Noirot, P. Bardou, A. Ng, D. Haase, A. Saiz Ede et al., "Interoperability with moby 1.0 -it's better than sharing your toothbrush," Brief Bioinform, vol. 9, no. 3, pp. 220–31, 2008.
[63] Javier Ríos, Johan Karlsson and Oswaldo Trelles; Magallanes: a web services discovery and automatic workflow composition tool; BMC Bioinformatics 2009, 10:334 doi:10.1186/1471-2105-10-334
[64] Victoria Martín-Requena , Javier Ríos , Maximiliano García , Sergio Ramírez and Oswaldo Trelles; jORCA: easily integrating bioinformatics Web Services; Bioinformatics 2010 26(4):553-559; doi:10.1093/bioinformatics/btp709
[65] Ramirez, et al. A flexible framework for the design of knowledge-discovery clients. In: International Conference on Telecommunications and Multimedia. (2008) Ierapetra, Crete, Greece.
[66] Shendure, J. and Ji, H. (2008) Next-generation DNA sequencing. Nature Biotechnology. Vol 26, Nº 10, pages 1135-1145.
[67] Wilkinson, M.D., Gessler, D., Farmer, A., Stein, L. (2003). The Bio-MOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability. Proceedings Virtual Conference Genomic and Bioinformatics (3):16-26. (ISSN 1547-383X).
[68] Stevens, R.D., et al. (2003) myGrid: personalised bioinformatics on the information grid. Bioinformatics, 19, (Suppl. 1), i302–i304
[69] Oster S, Langella S, Hastings S, Ervin D, Madduri R, Phillips J, Kurc T, Siebenlist F, Covitz P, Shanbhag K, *et al*. (2007): caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research. *J Am Med Inform Assoc* 2007

[70] Axis, an implementation of the Simple Object Access Protocol (SOAP) http://ws.apache.org/axis/
[71] Apache Tomcat, an open source software implementation of Java Servlet and JavaServer Pages technologies http://tomcat.apache.org/
[72] Juliusz Pukacki et al. (2006) Programming Grid Applications with Gridge. Computational Methods in Science and Thecnology; 12(1), 47-68
[73] I. Foster. (2006), Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13.
[74] Giovanni Aloisio, Massimo Cafaro, Italo Epicoco (2002); Early experiences with the GridFTP protocol using the GRB-GSIFTP library; Future Generation Computer Systems; Volume 18, Pages 1053-1059; ISSN:0167-739X
[75] W3C - SOAP specifications (http://www.w3.org/TR/soap/)