Modelling a Protein Structure Comparison Application on the Grid Using PROTEUS

Mario Cannataro[1], Matteo Comin[2], Carlo Ferrari[2], Concettina Guerra[2], Antonella Guzzo[3], and Pierangelo Veltri[1]

[1] University of Catanzaro
{cannataro, veltri}@unicz.it
[2] DEI - University of Padova
{comin, carlo, guerra}@dei.unipd.it
[3] DEIS - University of Calabria
guzzo@deis.unical.it

Abstract. Bioinformatics applications manage complex biological data stored into distributed and often heterogeneous databases and require large computing power. Among these, protein structure comparison applications exhibit complex workflow structure, access different databases, require high computing power. Thus they could benefit of semantic modelling and Grid infrastructure. We present the modelling and development of the PROuST structure comparison application on the Grid using PROTEUS, a Grid-based Problem Solving Environment.

1 Introduction

Research in biological and medical areas (also known as *biomedicine*), requires high performance computing power and sophisticated software tools to treat the increasing amount of data derived by always more accurate experiments in biomedicine [1]. The emerging bioinformatics area involves an increasing number of computer scientists studying new algorithms and designing powerful computational platforms to bring computer science in biomedical research.

Among the different interests, bioinformatics is focusing on the study of proteins and their biological functions. Proteins are sequences of *amino acids*, represented by strings. Amino acids sequences fold in three dimensional (3D) space assuming a variety of 3D structures. Since the structure of a protein is highly related to its functionality, knowing the amino acids sequence as well as its 3D space conformation helps biologist in predicting protein functionalities [11]. The high number of possible combinations of amino acids composing proteins, as well as the huge number of possible cell-mutations, require a huge effort in designing software environments and architectures able to manage the huge amount of data and to support protein studies. Proteins spatial structure prediction and folding are important issues for studying pathologies and to design new drugs.

For such reasons research communities are interested in studying existing proteins functionalities and in discovering new ones. Databases accessible to such communities have been designed and populated (see Protein Database, PDB [8]) and algorithms for analyzing and comparing proteins have been designed. Such algorithms have to deal both with string representations, i.e. amino acids sequences, and with their 3D structures. In particular, the structural comparison problem plays an important role in the functional classification of known proteins and in the prediction of the function of new ones. Recently, a new approach, named PROuST [4] has been proposed. It combines and integrates different techniques for structure comparison operating at different levels of protein representation with different degrees of accuracy. Comparison techniques need to interact with huge amount of data, requiring high computational efforts. PROuST compares an input query protein with a data set of known proteins, to obtain the 3D protein shapes most similar to the query protein. It works in two phases. First it stores information about the existing proteins in a hash table indexed by invariant properties of the protein structures. These properties are the angles and distances of triplets of segments associated to the secondary structures of the proteins. Then, for a given query protein, the algorithm computes the same invariant features and uses them to access the hash table and retrieve similarity information with the existing proteins. This fist step of the processing generates a list of candidate similar proteins. Next a dynamic programming approach is used to align the query protein with each candidate protein of the obtained list. A snapshot of the protein structural comparison workflow is reported in Figure 1. The protein structures are obtained from publicly available databases, i.e. from the Protein Data Bank that currently contains over 27,000 different structures. Thus, building indexes and evaluating a set of candidate proteins is a computationally intensive problem.

Grid community [9] recognized bioinformatics as an opportunity for distributed high performance computing and collaborative applications. Computational Grids (or simply Grids) are geographically distributed environments for high performance computation [10]. In a Grid environment is possible to manage heterogeneous and independent computational resources offering powerful services able to manage huge volumes of data. Managing heterogeneous datasets (e.g., protein databases) or creating new datasets (e.g., mass spectrometry proteomic data [7]), may take advantages by Grid environment [12].

In this paper we present the modelling and the implementation of the PROuST protein structure comparison application on the Grid, using the PROTEUS [2] Grid-based Problem Solving Environment. Migrating PROuST on Grid platform has been proposed in [6]. PROTEUS allows to design and model bioinformatics applications on Grid, using ontologies for modelling, and workflow techniques for designing and scheduling. In particular PROTEUS embeds an ontology based workflow designer allowing ontology-based design of the application. Moreover, a set of workflow engines allows controlling and enacting different phases of activities.

The paper is organized as follows. Section 2 presents PROTEUS architecture focusing on workflow management and modelling. Section 3 describes the PROuST structure comparison method and Section 4 presents the definition of PROuST application on PROTEUS through workflow modeling. Section 5 concludes the paper and outlines future works.

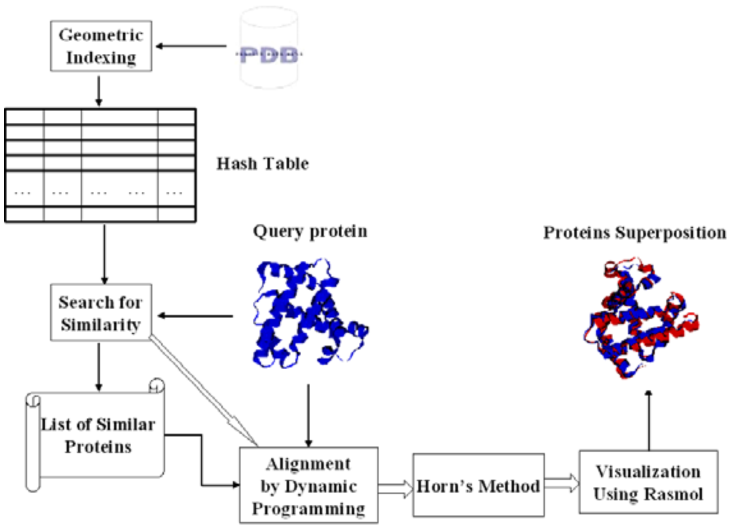

Fig. 1. PROuST Overall Workflow

2 Workflow Management in Proteus

Semantic modelling of Grid resources and workflow-based Grid programming are emergent trends in Grid community [3]. Along this direction, we developed PROTEUS, a Grid-based Problem Solving Environment allowing to model and execute Grid-aware bioinformatics applications through ontologies and workflows. Figure 2 shows main components of PROTEUS architecture.

The *Component and Application Library* contains software tools, databases, data sources, and user-defined bioinformatics applications, whose metadata are contained into the *Metadata Repository*. The *Ontology Repository* contains ontologies describing, respectively, biological concepts, bioinformatics tasks, and user-defined bioinformatics applications, represented as workflows. The *Ontology-based Workflow Designer* allows the design of a bioinformatics application as a workflow of software and data components selected by searching PROTEUS ontologies. It comprises the *Ontology-based Assistant*, that suggests available tools for a given bioinformatics problem, and the *Workflow User Interface*, used to produce workflow schema, stored into the *Workflow Metadata Repository*. Finally, the *WF-model Wrapper* maps an abstract workflow schema into a schedulable workflow, that in turn is scheduled (i.e. *enacted*) on the Grid by the *Workflow Engine*.

While deploying bioinformatics applications, particular attention should be devoted to the modelling phase; in this phase, in fact, the actors of the application as well as the way in which they operate to reach their goals must be described. From a conceptual point of view, such a description is equivalent to build a *workflow model*, i.e., a formal description of the tasks to be carried out, the dependencies/relationships among them (e.g. data flow, temporal prece-

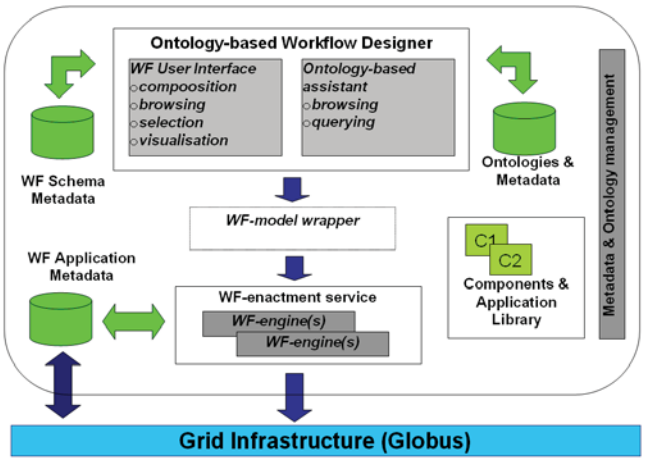

Fig. 2. Proteus Architecture

dences) and the entities involved in the application/process. Our proposal is to use workflow capabilities of PROTEUS in order to support users in the design of complex applications and in deploying experiments in an automatic manner. There are many reasons supporting this choice. First, as the design phase serves as the basis for the deployment, it is clear that correctness of the experiment specifications should be guarantied before the deployment phase, unless to bear the costs of doing so at implementation level. Workflow technology offers *Process definition tools* that allow the user to specify a process/application in a formal and unambiguous manner, according to some formal specification language.

Workflow technology offers several more intuitive graphical user-interfaces to specify bioinformatics applications, thus allowing the users to encode their knowledge without caring of implementation details.

Finally, it is generally recognized that supporting the design phase of an applications is a prerequisite for achieving the benefits with respect to maintainability, comprehensibility and reusability of the applications, which are crucial issues in the bioinformatic domain.

2.1 Basic Workflow Concepts

A *workflow* is a partial or total automation of a business/scientific process, in which a collection of *activities* must be executed by some *entities* (humans or machines), according to certain *procedural rules*. In this context, *Workflow Management Systems* (WfMSs) are well established technological infrastructures, aiming at facilitating the design of any workflow, and supporting its enactments, by scheduling different activities on available entities. According to the Workflow Management Coalition (WfMC) Reference Model (see http://www.wfmc.org), the two most relevant components of WfMSs are: *Buildtime component* and *Runtime component*.

Buildtime Component allows the definition of the workflow by means of some formal description such as the workflow schema, and ensures its persistent storage. It includes two level of specification: (i) *control flow* level, specifying the dependencies among tasks and their execution requirements, through language constructs (e.g. sequencing, synchronization, choice, etc); (ii) *data flow* level, specifying the information about processing entities, such as activity assignment, input and output parameters, etc.

Runtime Component consists of a workflow engine (often called workflow scheduler) responsible of the enactments, by controlling and coordinating execution of activities. Moreover, it stores log files about workflow executions and provides monitoring tools that keep track of execution progress.

2.2 Conceptual Workflow Modelling by Using UML

Many research works deal with the modeling of workflow schemes and currently there are many existing workflow languages, such as Xlang, WSFL, and BPEL from Microsoft and IBM; XPDL from the workflow management coalition; UML extensions and EDOC from the OMG; and WSCI, which is under the umbrella of the W3C, since no such languages is considered the "best" standard. In PROTEUS we use the UML activity diagrams as a workflow language specification. The Unified Modelling Language (UML) is a de-facto industry standard consisting of several graphical languages for representing software system designs and it is frequently used to illustrate processes in software applications. Recently, the activity diagrams are useful for modelling workflow specifications [5]. In particular, several works have demonstrated that UML supports the majority of the control flow constructs and is suitable to modelling the most of recurring situations related to the workflow execution.

Activity diagrams notation describes *activities* and the flow between them, which is determined by *transitions*, *forking*, *synchronization* elements, and flow directions notations, such *decision diamonds*. Figure 3 shows the basic notation for activities nodes; solid arrows represent control flow transitions; decisions are diamonds and forks and synchronization are expressed by solid bars.

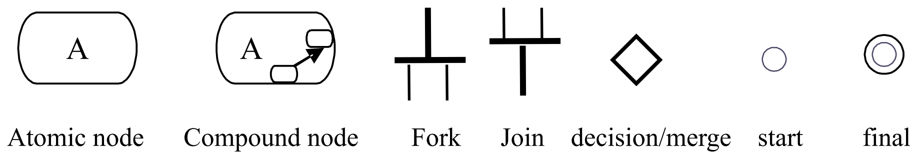

Fig. 3. Graphical Notation for UML Activity Diagrams

3 Protein Structure Comparison: The PRoUST Approach

The structural comparison problem plays an important role in the functional classification of known proteins and in the prediction of the function of new ones. This problem has been studied by several research groups using a variety of techniques including dynamic programming, graph algorithms, minimization of distance matrices, etc. Moreover some approaches have led to the design and implementation of web servers, such as DALI, CE, SSAP and VAST (see http://www.ebi.ac.uk/dali, http://cl.sdc.ede/ce.html, http://www.biochem.ucl.ac.uk/orengo/ssap.html, and www. ncbi.nlm.nih.gov). Recently, a new approach, namely PROuST, has been proposed [4], that combines and integrates different techniques to structure comparison operating at different levels of protein representation with different degrees of accuracy. PROuST consists of many computational components. The computational modules that can be arranged in various ways depending on the specific type of the requested task: a protein can be matched against all the proteins in PDB, or against a list of representative proteins selected from PDB (for instance, choosing only proteins with low degree of sequence similarity), or it can be compared with another protein to obtain an alignment of their structural elements. Moreover, a display of the aligned proteins can be obtained at the level of the secondary structures only, or extended to a subset of the atoms, the Cα backbone atoms or to all atoms.

Basically, PROuST design relies on two main techniques: it uses indexing for a fast retrieval of similarity information from a database of protein substructure features, followed by dynamic programming to obtain an accurate comparison and alignment between the query protein and each of the proteins extracted from the database by the fast index-based search. Indexes are derived from the segments associated to the secondary structure of Proteins, i.e. α-helices and β-strands. Recent comparisons of PROuST with stand alone procedures have demonstrated its efficiency. Moreover in [6] a possible immersion of PROuST on a Grid based environment has been proposed. Since PROTEUS offers a workflow management platform for workflow design and execution on Grid (see Figure 2), after presenting the overall PROuST workflow, we describe its design on PROTEUS.

We now review how PROuST works. More details can be found in [4]. Besides its atomic representation (as a list of 3D coordinates of all its atoms), a protein can be described in terms of its secondary structures (α-helices and β-strands). Our approach represents each protein as a set of vectors associated to secondary structures; the vectors are the best fit line segments for β-strands and the axes of α-helices. PROuST is based on indexing techniques for database access and fast similarity search. It computes angular features of triplets of segments associated to secondary structures. These features generate triplets of numbers that provide indexes to specific locations in an Hash Table (HT). Each table cell (bucket) consists of a list of records corresponding to proteins with one or more triplets of secondary structures that index into that cell. The Hash Table is built in a **pre-processing** phase that inserts all proteins and takes $O(n^3)$ time for the insertion of a protein with n secondary structures.

The **similarity search** problem involves a query protein Q and all the other proteins represented in the hash table. The search procedure accesses the database looking for triplets of secondary structures that are similar to those of Q, that is triplets with similar angles and distances between their vectors.

Proteins similar to Q are selected according to a similarity measure that takes into account the number of similar triplets between the two proteins. For each triplet of segments associated to the secondary structures of Q, the related three dihedral angles are computed and used as indexes to a table cell. All similar triplets of all stored proteins are stored in either that same cell or in adjacent cells. For each protein in the database, the search procedure keeps track of the number of triplets that are found similar to triplets of the query protein Q by incrementing for each access in a given cell a proper counter associated to that protein. After all triplets of Q have been examined, the proteins with the largest value of such counter are selected as the ones most similar to Q. The indexing method described above returns a ranking of candidate similar proteins but does not generate an alignment of secondary structures and atoms of the query protein with each of the candidate proteins.

The **structural alignment** procedure based on dynamic programming generates pairs of corresponding secondary structures and atoms of the two proteins that satisfy the continuity constraint given by the order of secondary structures along the sequence. The alignment optimizes a function based on the score between two secondary structures defined in terms of the number of similar triplets. The score is derived from the Hash Table (HT).

The final stage of the protein structure comparison is the superposition of the two proteins, that is the determination of the rigid transformation that results in the "best" overlap of the two proteins. Horn's algorithm is used to determine the optimal transformation that minimizes the Root Mean Square deviation (RMSD) distance between sets of atoms (pairs of corresponding points of two proteins).

4 Designing PROuST Application on PROTEUS

Currently PROuST is implemented as a stand alone application, so we wish to implement it by using a service-oriented approach. The main phases of the application have to be made independent by each others and implemented as autonomous software components, able to fulfill requests coming by different users, or triggered by external events.

Taking a bottom-up approach, we first model with PROTEUS the inner workflow schema of each PROuST phase (that represents a service), such sub-workflows are then combined to obtain the overall application. The description of workflows is carried out by means of the UML syntax introduced before. Note that activity diagrams specify not only the control flow, but also the data flow. This is an important feature because to enact a process, a WfMS needs to know which activity to call next and what data the activity needs. UML class diagrams can be used to describe the internal structure of data objects. In the following, the main phases of the PROuST application (Pre-processing, Similarity search and Structural alignment) are described as UML activity diagrams.

Pre-processing. This phase is represented as the activity diagram of Figure 4 where UML data flow (dashed arrows) is the connection of data objects with activities that require them as input and/or produce them as output. The input of this phase, the *PDB* file, is processed to obtain an internal protein representation allowing an accurate and efficient protein comparison. In some cases the *DSSP* database can be queried to obtain information about the secondary structures of a protein if it is not present in the PDB file. The output of this phase is the *Hash Table* (*HT*) introduced above. This phase is executed once, when the system starts up and whenever updates affect the PDB. From a computational point of view, Pre-processing is triggered by relevant updates in PDB, or by timeout expiration (e.g. each month), or by user action. Hash Table updating can be obtained by (incrementally) applying the Pre-processing phase on a local copy of the updated PDB file, or by using an agent-based system to periodically report PDB updating. A structured relational database allows to enhance the Pre-processing phase. In summary the Pre-processing phase has INPUT=$\{PDB, DSSP\}$, OUTPUT=$\{HT\}$.

As reported in the workflow of Figure 4, the preprocessing phase starts by accessing the PDB file ("*PDB access*" task). If PDB contains the secondary structure of a protein the task "*Compute SS from PDB*" is executed; otherwise, the DSSP file is used for its extraction ("*Compute SS from DSSP*" task). However, if the secondary structure is not available in any databases, the current PDB file is no more processed and the workflow returns in the starting activity. The secondary structure of a protein results in a couple of files representing its starting and ending residua (file .sec) and the coordinates of the carbon atoms (file .ca). They are the input of the task "*Vectorial representation*" that computes a representation (file .fit) of the secondary structure stored in the Hash Table (task "*Hash table update*"). Specifically, two possible updates might occur: (1) the insertion of a new protein, and (2) the insertion of a new version of an existing protein.

Similarity Search. In this phase (Figure 5), a target protein P, identified through its PDB identifier (e.g. 1tim is the <pdbID> of the protein Tim barrel), is compared against **all** the proteins contained in HT to obtain a list of similar proteins L_S, according to a similarity measure S.

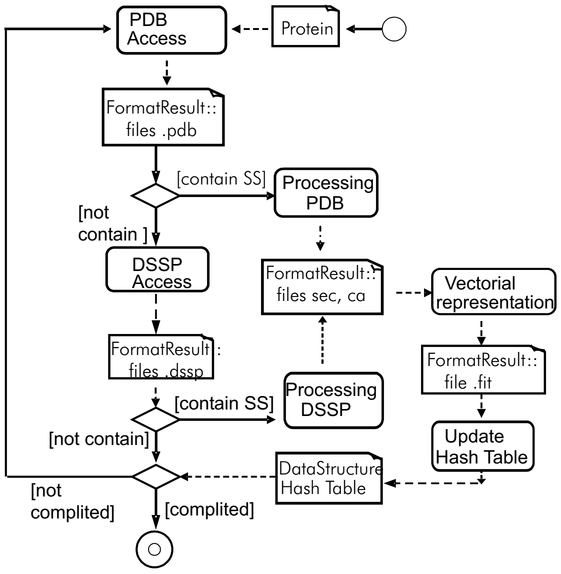

Fig. 4. Activity Diagram for the Pre-processing Phase

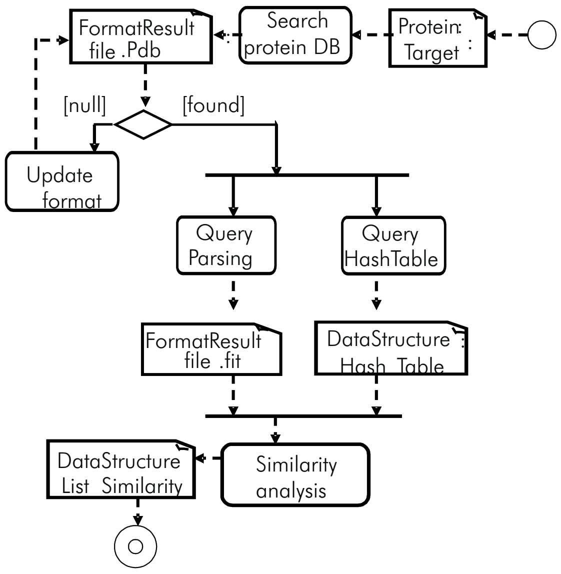

Fig. 5. Activity Diagram for the Similarity Search Phase

Each element of L_S contains a similar protein identified through its <pdbID>, and a value representing the similarity measure S with respect to the target protein P. Without loss of generality we can order the list L_S according to the similarity measure, and choose the sub list L_S^k containing the first k similar proteins, where k is a parameter provided by the user on the basis of his/her experience. Notice that similarity search is conducted against all the proteins stored in HT, so the parameter k is only a way to select the useful output for this phase but does not affect complexity or efficiency of the similarity search phase. The value of k may eventually be determined dynamically on the basis of a required minimum similarity threshold t, i.e. we could search for the first $k = k(t)$ similar proteins whose similarity measure is greater than t. Finally, since similarity search is conducted comparing vector-based representation of proteins, the target protein P has to be pre-processed by a parser module. In summary, Similarity search phase has INPUT=$\{P, HT\}$, PARAMETERS=$\{k\}$, OUTPUT=$\{L_S^K\}$.

As reported in the workflow of Figure 5, this phase starts by supplying a target protein P= <pdbID> used to query the PDB file to obtain the secondary structure protein information (task "*Search protein*"). In case the target protein is not stored in a PDB format, the task "*Update format*" is responsible of deriving the PDB information. Then, in the "*Query parsing*" task the file .fit is generated. Such file is needed for the "*Similarity analysis*" task. Which, in fact, computes a list of proteins sorted according to their degree of secondary structural similarity with the target protein.

Structural Alignment. In this phase (Figure 6) a detailed similarity analysis is performed by considering the position of atoms of target and similar proteins. The user chooses a protein L_i (*i=1,...,k*) from the similarity list L_S^k, then a

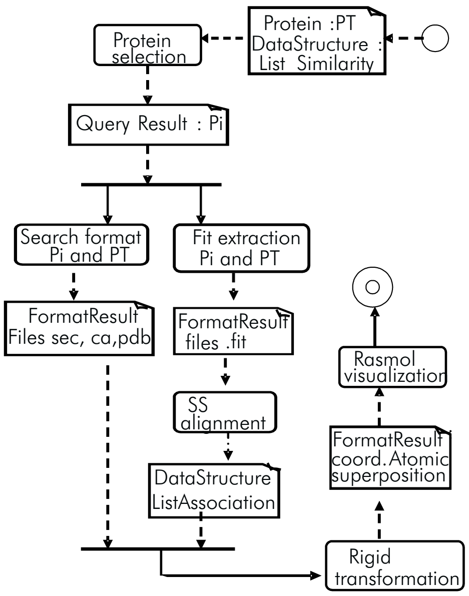

Fig. 6. Activity Diagram for the Structural Alignment Phase

structural alignment between L_i and the target P is performed. Next an *atomic superimposition* of these two proteins, based on a rigid transformation composed by roto-translation movements, is performed.

Finally, this superimposition can be eventually visualized using a 3D *visualization* tool such as Rasmol (see http://www.umass.edu/microbio/rasmol/). It should be noted that both Li and P are visualized with respect to the same point of reference. After visualization the user can choose another similar protein Lj, to conduct a new Similarity analysis, or he/she can stop the process. In summary the Similarity analysis phase has INPUT=$\{P, L_i\}$, OUTPUT=$\{$*superimposition (P,* L_i)$\}$.

As reported in the workflow of Figure 6, a protein occurring in the similarity list is selected (task "*Protein selection*") for testing its actual structural similarity with the target protein on the basis of the degree of atoms overlapping. This measure is obtained by computing the rigid transformation of the proteins that makes their structures overlap as much as possible.

This task can be performed by analyzing not only the PDB, .sec, .ca files (task "*Files extraction*") associated to both proteins, but also an association list between the secondary structures. This list is computed from the .fit files (obtained by means of the task "*Fit extraction*"), by means of the task ("*SS alignment*").

Notice that the tasks "*SS alignment*" and "*Files extraction*" are synchronized in a way that the "*Rigid Transformation*" task can be executed only after their proper termination. Finally, the overlapping can be visualized by means of a visualization tool, such as Rasmol.

After the application modelling phase, the workflows designed so far, stored into the Workflow Metadata Repository of PROTEUS, are combined together to form the overall Grid-aware PROuST application. Using the PROTEUS workflow-enactment service, the application is then executed on the Grid. On the other hand, some of the designed workflows represent self-contained services that can be reused for further applications.

5 Conclusions and Future Work

Bioinformatics applications, such as structure comparison, present complex workflows that involve different data sources and software components, and often require high computing power. The deployment of such applications on the Grid can benefit from semantic modelling of both the elementary tasks and the overall application through workflow. We described the modelling and implementation of the PROuST structure comparison application through PROTEUS, a Grid-based Problem Solving Environment. Detailed descriptions of PROuST phases and related PROTEUS workflows have been presented.

Future work regards the completion of the PROTEUS workflow-enactment service and its use to evaluate the Grid-aware PROuST application. Moreover, PROuST workflow shows various sources of parallelism that can further benefit of Grid deployment, such as Hash Table construction and querying, and parallel execution of PROuST phases in a multi-user setting.

References

1. P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* MIT Press, 2001.
2. M. Cannataro, C. Comito, F. Lo Schiavo, and P. Veltri. Proteus, a grid-based problem solving environment for bioinformatics: Architecture and experiments. *IEEE Computational Intelligence Bulletin*, 3(1):7–18, February 2004.
3. M. Cannataro and D. Talia. Semantic and Knowledge Grids: Building the Next-Generation Grid. *IEEE Intelligent Systems*, 19(1):56–63, January-February 2004.
4. M. Comin, C. Guerra, and G. Zanotti. PROuST: A comparison method of three-dimensional structures of proteins using indexing techniques. *J. of Computational Biology*, 11:1061–1072, 2004.
5. Marlon Dumas and Arthur H. M. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. *UML01, Lecture Notes in Computer Science*, 2185, 2001.
6. C. Ferrari, C. Guerra, and G. Zanotti. A grid-aware approach to protein structure comparison. *Jounal of. Parallel and Distributed Computing Special issue on High Performance Bionformatics*, 63, 2003.
7. NCBI-National Cancer for Biotechnology Information. Genbank dna sequences. http://www.ncbi.nlm.nih.gov/.
8. Research Collaboratory for Structural Bioinformatics (RCSB). The protein data bank. www.rcsb.org/pdb.
9. Global Grid Forum. Life science grid - research group. http://www.ggf.org/7_APM/LSG_b.htm.
10. I. Foster and C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufmann Publishers, 1999.
11. C. Guerra and S. Istrail. *Mathematical Methods for Protein Structure Analysis and Design.* LNBI, Springer, 2000.
12. University of Manchester. Mygrid. http://mygrid.man.ac.uk/.