Integrating Protein Data Resources through Semantic Web Services

by

Xiong Liu

B.S., Hefei University of Technology, 1996

M.S., Tsinghua University, 1999

M.S., University of Pittsburgh, 2004

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Xiong Liu

It was defended on

November 7, 2006

and approved by

Dr. Ivet Bahar, Professor, Computational Biology, School of Medicine

Dr. Michael Lewis, Professor, School of Information Sciences

Dr. John Vries, Associate Professor, Computational Biology, School of Medicine

Dr. Vladimir Zadorozhny, Assistant Professor, School of Information Sciences

Dr. Hassan Karimi, Associate Professor, School of Information Sciences
Dissertation Director

Integrating Protein Data Resources through Semantic Web Services

by

Xiong Liu, PhD

University of Pittsburgh, 2006

Understanding the function of every protein is a goal of bioinformatics. Currently, a large amount of information (e.g., sequence, structure and dynamics) is being produced by experiments and predictions that are associated with protein function. Integrating these diverse data about protein sequence, structure, dynamics and other protein features allows further exploration and establishment of the relationships between protein sequence, structure, dynamics and function, and thereby controlling the function of target proteins. However, information integration in protein data resources faces challenges at technology level for interfacing heterogeneous data formats and standards and at application level for semantic interpretation of dissimilar data and queries.

In this research, a semantic web services infrastructure, called Web Services for Protein data resources (WSP), for flexible and user-oriented integration of protein data resources, is proposed. This infrastructure includes a method for modeling protein web services, a service publication algorithm, an efficient service discovery (matching) algorithm, and an optimal service chaining algorithm. Rather than relying on syntactic matching, the matching algorithm discovers services based on their similarity to the requested service. Therefore, users can locate services that semantically match their data requirements even if they are syntactically distinctive. Furthermore, WSP supports a workflow-based approach for service integration. The chaining algorithm is used to select and chain services, based on the criteria of service accuracy and data interoperability. The algorithm generates a web services workflow which automatically integrates the results from individual services.

A number of experiments are conducted to evaluate the performance of the matching algorithm. The results reveal that the algorithm can discover services with reasonable performance. Also, a composite service, which integrates protein dynamics and conservation, is experimented using the WSP infrastructure.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Introduction

In the post-genomic era, understanding the function of every protein is viewed as a key step towards effective drug design and a major goal in bioinformatics. Protein function is a dynamic property closely related to conformational changes accessible to the protein structure under physiological conditions. Protein catalytic activity, binding and molecular recognition all involve protein motions (Sinha and Smith-Gill 2002). With the rapid accumulation of protein structures in the Protein Data Bank (PDB) (http://www.rcsb.org/pdb/; Berman et al., 2000), it is now widely recognized that efficient methods and tools for predicting dynamics are needed in order to better understand the function of target proteins.

iGNM is a Web-based system for high throughput analysis and prediction of protein dynamics (http://ignm.ccbb.pitt.edu; Liu et. al, 2004; Yang et al., 2005). It is a joint project between the Department of Computational Biology, School of Medicine, and the Geoinformatics Laboratory, School of Information Sciences, at the University of Pittsburgh. As of August 2006, iGNM Version 1.2 provides protein dynamics information, i.e., conformational changes, for more than 20,000 protein structures. Due to its efficiency and applicability to large structures and assemblies, iGNM is gaining attention of researchers in the scientific community.

Integrating iGNM with other protein data resources (see Section 2.4) allows for further exploration and establishment of the relationships between sequence, structure, dynamics and function, and thereby controlling the function of target proteins. This motivates the author to develop a convenient software environment for integrating various types of protein data, ranging from sequence-derived features (e.g., conservation) to structure-derived features (e.g., dynamics) and to biophysical features (e.g., hydrophobicity and enzyme active sites).

Currently, many protein data resources are accessible to researchers through Web application interfaces, e.g., through a HTTP (Hypertext Transfer Protocol) form and a corresponding Java servlet. Users of these data resources are mainly biomedical researchers and developers. To integrate data through Web applications (current approach), users have to have prior knowledge about data resources and write scripts to parse HTML (Hyper Text Markup Language) code to exact data while ignoring explanatory text and graphics. This approach is labor intensive and fragile for minor changes in the HTML code of a given Web application may cause failure (Stein 2002). Refer to Section 2.5 for details.

Web services, on the other hand, offer an environment for flexible integration of various types of information, including data, programs, files and other Web resources (see Section 2.6). Web services represent underlying information using standard programmatic interfaces so that user applications can obtain explicit results without tedious HTML code parsing. Also, web services provide XML (eXtended Markup Language) based protocols and tools that facilitate the discovery and integration of Web resources developed in different platforms. Since Web services

simplify the information integration process, interest of applying Web services to biological research has grown in recent years (Stein 2002; Foster 2005; Gao et al., 2005).

Current web services standards (see Section 2.6.1) allow user programs to discover services based on syntactic descriptions of services, e.g., WSDL (Web Services Description Language). However, service providers and users may have distinctive perspectives and knowledge about one service resulting in differing descriptions for the service. In this case, syntactic based matching, e.g., UDDI (Universal Description Discovery and Integration), will be unable to locate the service because there are no semantic operations. The term "semantic", as defined by the Semantic Web community, refers to a machine's ability to solve a problem without human direction by performing well-defined operations on existing well-defined data (W3C Semantic Web, 2006). The Semantic Web can provide the ability to tag all content on the web and give semantic meaning to the content item. The potential benefits are that search engines become more effective than they are now by providing the precise information users are looking for.

To make Web services capable of handling semantic interoperation, the Semantic Web community has combined semantic markup languages and ontologies* with current web services standards. This has led to semantic web services, one type of web services that can express not only interfaces among services but also their capabilities (McIIraith et al., 2001; Paolucci, et al., 2002). Since the semantics of web services are explicitly stated, services can be automatically discovered even if the services and the service requested are syntactically distinctive.

* An ontology is a "specification of a conceptualization" (Gruber 1993). In the computer science domain, ontology provides a commonly agreed understanding of domain knowledge for sharing across applications and organizations. Typically, ontology consists of a list of terms and the relationships between those terms.

Semantically annotated web services bring new paradigm shift of computing in scientific research where data are heterogeneous and distributed. This dissertation proposes a semantics-based Web services infrastructure, called Web Services for Protein data resources (WSP), and how it can meet the requirements of optimal (automatic and accurate) integration of distributed protein data. In WSP, data resources (e.g., web databases, computational servers and tools) that provide protein data are modeled as reusable web services. Each service has a programmatic interface as well as a semantic description of its capabilities in terms of inputs, outputs and constraints. The semantic descriptions are published in a service registry and a semantic matchmaker is designed to perform semantic matching between services and requests. To facilitate protein data integration, a workflow-based chaining algorithm is designed to pipe together inputs and outputs of consecutive web services. Potential WSP users include protein data providers, bioinformatics researchers and developers. WSP allows users to conveniently publish, discover and assemble various types of protein data (both existing and yet to come) for their applications (e.g., predicting protein function from dynamics data).

1.2. Contributions

This research yields the following contributions:

- An infrastructure for representing and correlating protein features at a higher semantic level. By exploiting the features of semantic-based web services, this infrastructure allows researchers to conveniently discover and assemble various types of protein data for their applications, e.g., determining the function or other features of proteins.

- Two biological web services that demonstrate the process of developing and using biological web services. The iGNM web service provides protein dynamics data for more than 20,000 protein structures. The N-gram web service provides conservation profiles for more than 50,000 protein sequences.
- A semantic matchmaker service that allows service providers to publish the description of their services and allows users to submit requests and obtain semantically matched services. The matchmaker includes an efficient semantic service matching algorithm.
- An optimal chaining algorithm that considers both accuracy and data interoperability between services.

1.3. Organization

This dissertation first describes research background and related work. Further, the WSP infrastructure for optimal data resource integration is presented. From this foundation the dissertation discusses the major components of the infrastructure, including protein web services, a semantic matchmaker and chaining of services. The outline of the chapters is as follows.

Chapter 2: Background and Related Work. Covers the basics of proteins, the iGNM system and other protein data resources, traditional methods for data resource integration, web services and their applications in bioinformatics.

Chapter 3: A Semantic Web Services Infrastructure for Protein Data Resource Integration. Discusses the methodologies and components of the WSP infrastructure.

Chapter 4: Modeling Protein Web Services. Details the design and development of protein web services.

Chapter 5: Semantic Description of Web Services. Describes the role of ontologies and methods used to semantically describe protein web services' capabilities.

Chapter 6: Semantic Publication and Matching of Web Services. Describes the design of the WSP matchmaker, the service publication algorithm, and the service matching algorithm.

Chapter 7: Chaining of Protein Web Services. Presents the methods and criteria used for chaining protein web services.

Chapter 8: WSP Prototype. Discusses the implementation issues related to WSP components.

Chapter 9: WSP Evaluation. Describes experiments performed to evaluate the service matching and integration processes.

Chapter 10: Conclusion and Future Research. Summarizes the research and also presents topics for future work.

2. BACKGROUND AND RELATED WORK

2.1. Introduction

This chapter first presents the background knowledge about proteins, the iGNM data resource, and other protein data resources. Then it reviews the technologies used to integrate data resources, including traditional methods (e.g., data warehousing) and web services based methods. Finally, applications of web services in bioinformatics are discussed.

2.2. Protein Sequence, Structure, Function and Dynamics

Proteins are the most abundant macromolecules in living cells, constituting more than half of the dry weight of cells (Garrett and Grisham 1999). Proteins consist of amino acids. Figure 2-1(a) shows the basic structure of an amino acid, where the central alpha carbon (Cα) carries a carboxyl end (written as COOH), a hydrogen atom (H), an amino end (written as NH2), and a variable R group. R denotes any one of the 20 possible side chains found in the nature. Two amino acids connected to form a peptide through dehydration, see Figure 2-1(b), and a sequence of amino acids form a peptide chain. By convention, an amino acid in a peptide chain is also called a ***residue***.

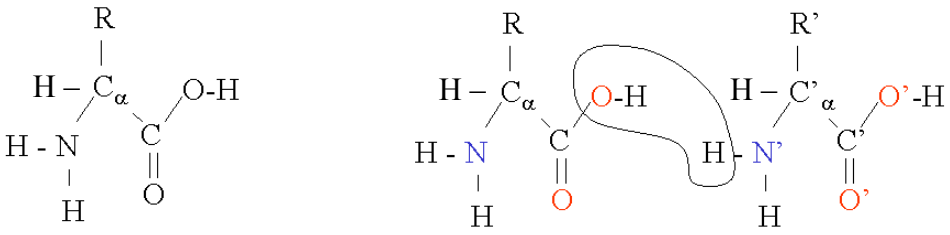


Figure 2-1. (a) The basic structure of amino acid; (b) A peptide chain.

Protein has multiple levels of structure, see Figure 2-2. The most basic level is the primary structure, which is simply the sequence of amino acids. The secondary structure refers to certain common repeating structures found in proteins such as alpha-helix and beta-pleated sheet. The tertiary structure is the full 3-dimensional folded structure of the polypeptide chain. The quaternary structure is the joining together of tertiary units, it is only present if there is more than one polypeptide chain.

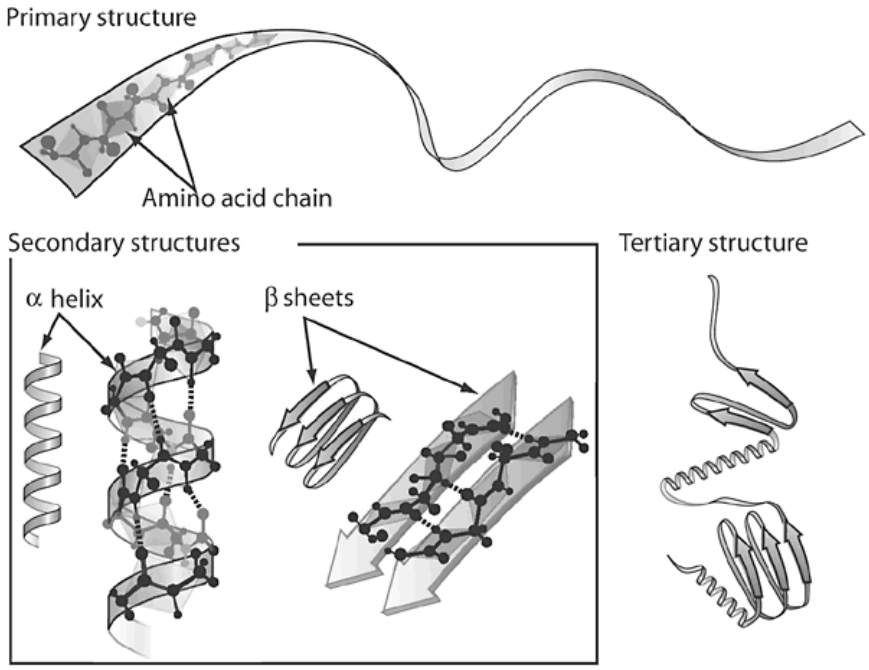


Figure 2-2. Protein multiple-level structures (Brown 2003)

While structural genomics attempts to crystallize or predict the structures of all protein in humans, functional genomics attempts to identify protein's function or what it does. Protein function involves the binding of other molecules called ligands, which can be any kind of molecule, even other proteins. Ligands bind to the protein at a certain site called the binding site. The binding site is particular to the ligand with respect to shape and chemical properties (Gold and Jackson, 2006).

Protein function is well illustrated by enzymes, a special type of proteins. Enzymes are biochemical catalysts that speed up chemical reactions that would occur too slowly for cells to function. The substrates fit into an enzyme's active site. Enzymes do their job of catalysis by providing an optimal chemical environment for bond making or breaking steps. This is usually achieved by lowering the transition state energy during reaction (Griffiths et al. 2002). Figure 2-3 illustrates the action of a hypothetical enzyme in putting two substrate molecules together, where "*" represents an active site.

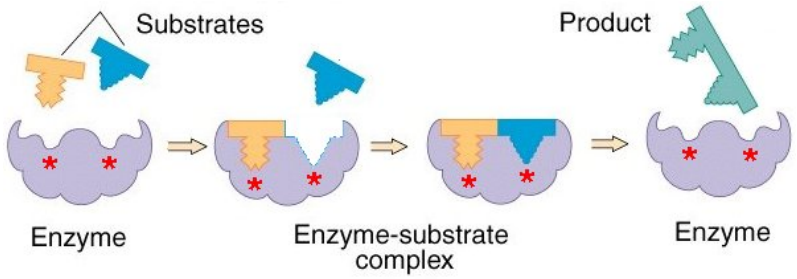


Figure 2-3. In the induced-fit model, binding of substrates induces a conformational change in the enzyme (Griffiths et al. 2002)

Protein function is a property closely related to the conformational mechanics of the structure in its physiological environment. Protein catalytic activity, folding, binding and molecular recognition all involve protein motions or conformational changes (Sinha and Smith-Gill 2002). The connection between structure and function presumably lies in dynamics, suggesting a paradigm shift in structural genomics studies from sequence-structure analysis to structure-dynamics analysis, for gaining a more insightful understanding of sequence-structure-dynamics-function relations (Yang et al., 2005).

Proteins have uniquely defined native structures under physiological conditions. The motions of proteins near native state conditions are confined to a subset of conformations in the neighborhood of the folded state, e.g., the open and closed forms of enzymes. In addition, the equilibrium dynamics of proteins can be viewed as a collection of normal modes that lead to experimentally observed residue fluctuations. Fluctuations involve correlated motions of different structural elements, ranging from atoms, residues to large domains and subunits whose concerted movements underline biological function (Yang et al., 2005).

2.3. iGNM Data Resource

As dynamics gives insightful picture of the mechanism that dictates protein function, the collective effort to predict dynamics in a large database scale is relatively less than that for protein structure information. iGNM is a data resource that provides dynamics information for more than 20,000 protein structures. This section discusses the computational issues of protein dynamics, the Gaussian Network Model, and the iGNM system.

2.3.1. Computational Prediction of Protein Dynamics

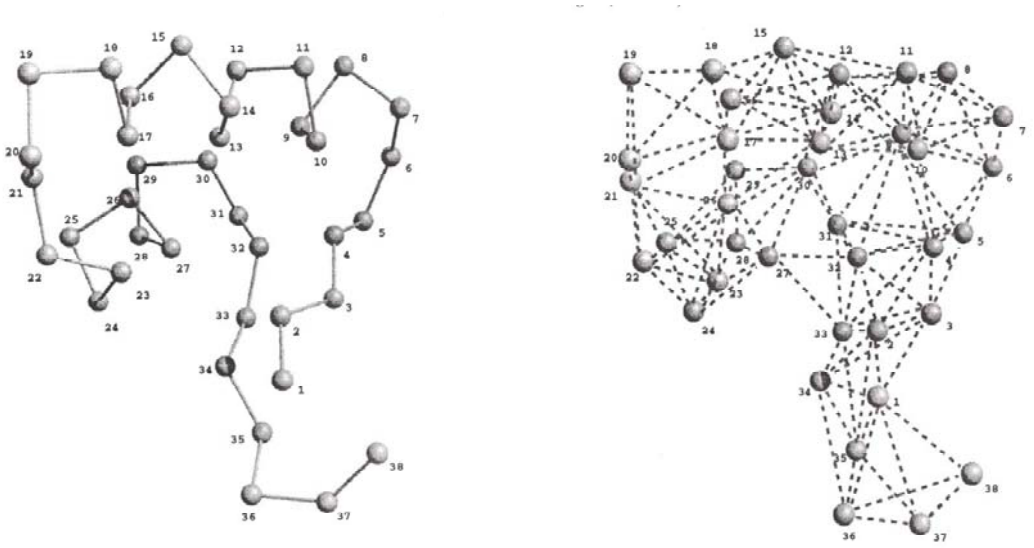
Experimental methods, such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) and Hydrogen/Deuterium (H/D) exchange, reveal atomic level information on protein internal motions. For example, temperature factor or B-factor is used to measure the positional uncertainty associated with each atom in the thermal fluctuations. Due to experimental cost, a major endeavor in recent years has been devoted to developing computational models and methods for simulating protein dynamics using structural data and relating the observed behavior to other experimental data.

Molecular Dynamics (MD) simulations have proven to be a useful approach for generating conformational trajectories of macromolecules in order to visualize the correlation of their dynamics to the biological functions (Brooks and Karplus 1983). However, MD simulations are expensive in terms of both CPU time and memory. An efficient method for identifying function-related conformational changes is Normal Mode Analysis (NMA), a method widely used for characterizing molecular fluctuations near a given equilibrium sate using vibrational modes. The utility of NMA for protein dynamics has been recognized for the last 20 years (Brooks and Karplus 1983; Go et al., 1983) but has been revitalized in recent years with the success of elastic network models used in NMA. In these models, atoms or groups of atoms (e.g., residues or groups of residues) are modeled as point sites (network nodes) connected by springs, which account for the force field that stabilizes the native structure. The utility of these models in NMA was first pointed out by Tirion (Tirion, 1996). Given the insensitivity of the most cooperative

modes to the detailed structure, a large majority of recent analyses have been performed using lower resolution elastic network models. Among the elastic network models of different complexities, the simplest is the Gaussian Network Model (Bahar et al., 1997).

2.3.2. Gaussian Network Model (GNM)

GNM is entirely based on inter-residue contact topology in the folded state. GNM requires no *a priori* knowledge of empirical energy parameters, based on the original proposition of Tirion (Tirion, 1996), and most importantly it lends itself to a closed mathematical solution. An important feature of GNM is the possibility of dissecting the observed motion into a collection of modes. These modes usually provide information on the molecular mechanisms relevant to biological function (Tama and Sanejouand, 2001). Several studies (Jernigan and Bahar, 1998; Bahar et al., 1998; Jernigan and Bahar, 1999; Haliloglu and Bahar, 1999; Rader and Bahar, 2004) have demonstrated the utility of GNM for understanding the machinery of proteins and their complexes.



(a) (b)

Figure 2-4. GNM graphic representation, where there is no distinction between nonbonded and bonded neighbors. (a) Chain of residues; (b) GNM interaction network of residues

In GNM, the alpha carbon atoms of residues are identified as the junctions or nodes of the network, and the pairs of nodes closer than a cutoff distance are connected by harmonic potentials with a uniform spring constant γ (see Figure 2-4). In addition to nonbonded interactions, the effect of chain connectivity is also considered, as the model automatically includes the constraints imposed by the first neighboring alpha carbon atoms along the backbone. Thus, the residues fluctuate under the potentials of their near neighbors. The connectivity (or Kirchhoff) matrix of contacts, Γ, is used to describe the inter-residue contact topology.

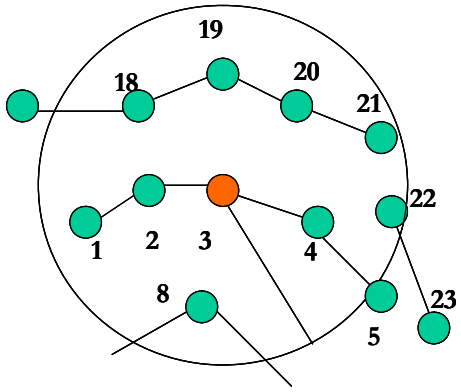


Figure 2-5. An example illustrating residue connectivity

The definition of Γ is given in Equation 2-1, where i and j are residue index, R_{ij} is the distance between residue i and residue j, r_c is the distance cutoff usually around 5 to 7 angstroms (Å), and $\sum \Gamma_{ik}$ is the number of coordination residues within the cutoff. The off diagonal elements of Γ are defined as $\Gamma_{ij} = -1$ if R_{ij} is shorter than r_c, and zero otherwise; and the i^{th} diagonal terms is the degree of node i, or the coordination number of residue i.

$$\Gamma_{ij} = \begin{Bmatrix} -1 & if & i \neq j & and & R_{ij} \leq r_c \\ 0 & if & i \neq j & and & R_{ij} > r_c \\ -\sum_{k,i \neq k} \Gamma_{ik} & if & i \neq j & & \end{Bmatrix} \qquad 2\text{-}1$$

$$\begin{matrix} & 1 & 2 & 3 & 4\ldots & 18 & 19 & 20 & 21\ldots N \\ 3 & & & & & & & & \end{matrix}$$

$$\Gamma(R) = \begin{Bmatrix} & & & & & & & & \\ -1 & -1 & 9 & -1\ldots & -1 & -1 & -1 & -1\ldots & 0 \\ & & & & & & & & \end{Bmatrix} \qquad 2\text{-}2$$

Figure 2-5 shows an example of calculating Γ, where residue "3" is at the center of a cutoff sphere. The 3rd row of Γ, which is the connectivity between residue "3" and other residues is given in Equation 2-2.

The statistical thermodynamics of the network are controlled by the Hamiltonian (Bahar et al., 1998):

$$H = (\gamma/2)\,[\Delta X\,\Gamma\,\Delta X^T + \Delta Y\,\Gamma\,\Delta Z^T + \Delta Z\,\Gamma\,\Delta Z^T] \qquad 2\text{-}3$$

where γ is the spring constant, ΔX, ΔY and ΔZ are the *N*-dimensional vectors of the *X-, Y-* and *Z-*components of the fluctuation vectors $\{\Delta R_1, \Delta R_2, \ldots, \Delta R_N\}$ of the *N* residues in the examined protein. The mean-square fluctuations of residue *i* scale with the i^{th} diagonal element of the inverse of Γ (Bahar et al., 1997; Haliloglu et al., 1997), as

$$\langle(\Delta R_i)^2\rangle = (3kT/\gamma)\,[\Gamma^{-1}]_{ii} \qquad 2\text{-}4$$

and the cross-correlations $\langle\Delta R_i \cdot \Delta R_j\rangle$ scale with the ij^{th} off-diagonal elements of Γ^{-1}.

The fluctuation dynamics of the structure results from *N-1* superposed GNM modes. The modes can be extracted by the eigenvalue decomposition of Γ. The decomposition reads $\Gamma = U\,\Lambda\,U^T$, where U is an orthogonal matrix whose columns u_i ($1 \le i \le N$) are the eigenvectors of Γ, and Λ is the diagonal matrix of the eigenvalues λ_i, usually organized in ascending order. The ith

eigenvector reflects the shape of the ith mode as a function of residue index. The ith eigenvalue represents its frequency (Haliloglu et al., 1997).

The eigenvalue decomposition of the connectivity matrix Γ is the most expensive task in GNM calculations from computational time point of view. Singular value decomposition (SVD) method (Press et al., 1992) is usually used to this aim, the computing time of which scales with N^3 for a network of N residues. When N is less than 1,500, the computations are performed within minutes, while the CPU times increased up to 15 days in the case of the largest structures.

An alternative decomposition algorithm that utilizes the BLZPACK software (Marques, 1995) is based on Block Lanczos Method for large structures. This method evaluates a subset ($1 \le k \le 100$) of dominant (slowest) modes, within a time scale of N^2, i.e. the computing times is more than 3 orders of magnitude shorter than the routine SVD, when structures of more than 10^3 residues are analyzed.

The theoretical temperature factor (B_i) predicted by GNM is proportional to the inverse Kirchhoff matrix and also to the summation of all modes as

$$\boldsymbol{B_i} = (8\pi^2 k_B T/\gamma)\left[\Gamma^{-1}\right]_{ii} = (8\pi^2/3)\sum_{k=1}^{N-1}\lambda_k^{-1}[\boldsymbol{u}_k]_i[\boldsymbol{u}_k]_i \qquad 2\text{-}5$$

Equation 2-5 follows from Equation 2-4 and the definition $B_i = (8\pi^2/3)\langle(\Delta R_i)^2\rangle$. $[\boldsymbol{u}_k]_i$ designates the i^{th} element (corresponding to i^{th} residue) of the k^{th} eigenvector.

2.3.3. Internet Accessible GNM (iGNM)

The accumulating evidence that supports the utility of GNM as an efficient tool for protein dynamics has led to the construction of iGNM, a database of GNM results compiled for more than 20,000 protein structures ranging from small enzymes to large complexes and assemblies. iGNM is based on a client-server architecture for query and visualization of protein dynamics (see Figure 2-6). The client is based on standard Web browsers, where the servers include the iGNM database server and the PDB server. Also, there is an additional online calculation server, called oGNM, for online calculation of PDB structures that are not deposited in the iGNM database server (Yang et al., 2006).

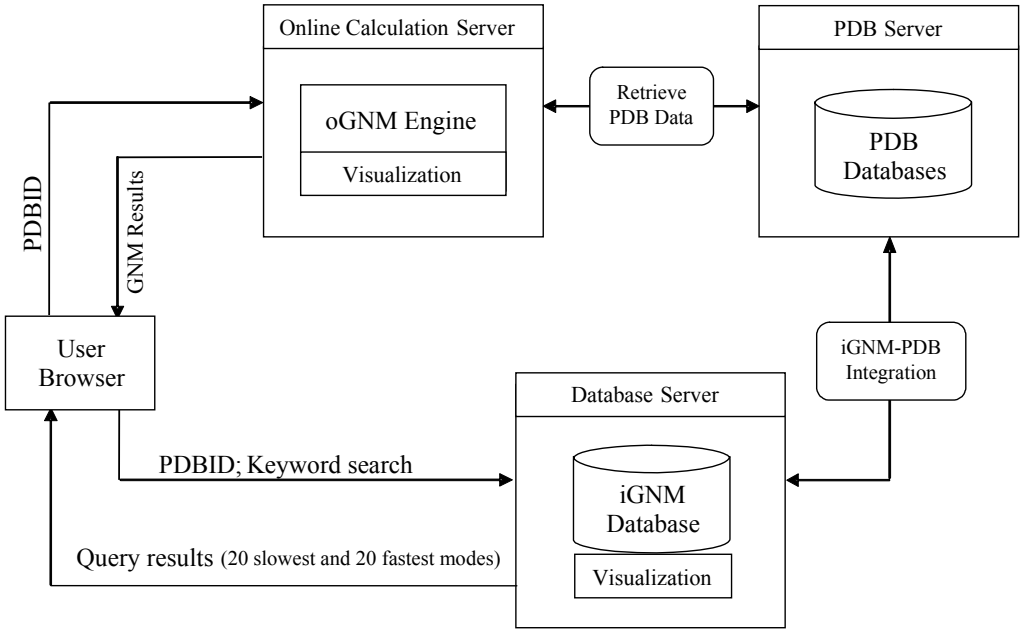


Figure 2-6. iGNM architecture: database server + online calculation server

iGNM Database Server

The goal of the database server is to provide information on the dynamics of all proteins beyond those experimentally provided by B-factors (for X-ray structures), root-mean-square fluctuations (NMR structures), or by interpolation between existing PDB structures. Currently, the iGNM database contains visual and quantitative information on the collective modes predicted by GNM for 20,058 structures deposited in PDB prior to September 15, 2003.

There are five major database entities (tables) in the database server: (1) the GNM entity that stores both structural and dynamics information for each protein structure; (2) the B-factors entity that stores equilibrium fluctuations for each residue of a protein structure; (3) the Slow-modes entity that stores the slowest (lowest frequency) modes; (4) the Fast-modes entity that stores the fastest (highest frequency) modes; and (5) the Crosscorr entity that stores the correlation of fluctuations between different residues. The GNM entity has a one-to-one relationship with the rest of the entities. Table 2-1 shows the iGNM database schema.

Table 2-1. iGNM Database schema

Entity Name	Attributes
GNM	PDBID, protein name, protein class, structure, description, B-factors, Slow-modes, Fast-modes, CrossCorr
B-factors	PDBID, Residue index, theoretical B-factor, experimental B-factor
Slow-modes	PDBID, Residue index, 1st slow mode, 2nd slow mode, …, 10th slow mode
Fast-modes	PDBID, Residue index, 1st fast mode, 2nd fast mode, …, 10th fast mode
Crosscorr	PDBID, Residue index1, Residue index 2, correlation

The GNM entity contains 9 attributes (see Table 2-1). The first one is the PDBID, which is a key to a protein record. Attributes such as protein name, protein class, structure and descriptions are the PDB objects, while attributes such as B-factors, Slow-modes, Fast-modes and Crosscorr are iGNM objects.

The B-factors entity contains three attributes: the residue index, the GNM calculated theoretical B-factors, and the X-ray crystallographic B-factors taken from PDB.

The Slow-modes entity contains eleven attributes. The first one refers to residue index. Attributes 2-11 are slow mode shapes associated with the 10 slowest modes, starting from the slowest (first) mode. Figure 2-7 shows a data instance example of the Slow-modes entity.

There are also 11 attributes in the Fast-modes entity. The first one refers to residue index. Attributes 2-11 are fast mode shapes associated with the 10 fastest modes, starting from the highest mode. Since the last modes reflect localized fast motions in the protein, these modes have few non-zero elements.

The Crosscorr entity contains three attributes. The first two are the residue indices and the last one is the correlation value.

Residue index	Residue Name	1st slowest mode	2nd slowest mode	3rd slowest mode	...
0	MET	0.00632	0.01597	0.00061	
1	VAL	0.00511	0.02603	0.00276	
2	LEU	0.0038	0.02323	0.00279	
3	SER	0.00327	0.02429	0.00328	
4	GLU	0.00277	0.02328	0.00325	
5	GLY	0.00243	0.02234	0.00292	
6	GLU	0.00249	0.02053	0.00193	
7	TRP	0.00204	0.01651	0.00339	
8	GLN	0.00165	0.01939	0.00257	
9	LEU	0.00152	0.01733	0.00114	

Figure 2-7. Data instance example of entity Slow-modes

The design of the iGNM database is customized based on GNM output files. The database has one instance of the GNM table schema for each protein in PDB. The attributes that are iGNM objects (e.g., B-factors) are linked through PDB ID to each corresponding table. As the database schema shows, there is only one-to-one relationship between the GNM table and other tables. Since there are no complicated join operations between the tables, each GNM instance is implemented as a folder and the object attributes (e.g., B-factors, Slow-modes) associated with the instance are stored as text files.

iGNM allows users to retrieve information through a simple search engine by entering the PDB identifier of the protein structure of interest. For example, “2hmg” is the PDB code for influenza virus hemagglutinin A (HA). The output includes: (1) the equilibrium fluctuations of residues and comparison with X-ray crystallographic B-factors; (2) the sizes for residue motions in different collective modes; (3) the cross-correlations between residue fluctuations, or domain motions in the collective modes; and (4) the identity of residues that assume a key mechanical

role (e.g., hinge) in the global dynamics, and thereby function, of the molecule, as well as those potentially participating in folding nuclei/cores (Bahar et al., 1998; Rader and Bahar, 2004).

After a protein structure is retrieved, the fluctuations of each residue are displayed in both 2D mobility graph and 3D ribbon diagram. iGNM allows the visual query of each residue's fluctuation by either interactively clicking a residue's position in the 2D graph or using embedded menus to select residues with desired features in the 3D diagrams.

In addition to queries using PDB IDs, iGNM is integrated with the PDB SearchLite query interface for keyword-based queries (Liu et. al, 2004). By typing keywords related to the biological macromolecules of interest, users can browse PDB records and iGNM output files for a given protein family in an integrated environment (see Figure 2-8). The PDB linkage and the GNM linkage are inserted into each retrieved record for convenient access to both conformational and dynamic information. The retrieved records can be sorted alphabetically according to PDB ID and Title, as well as numerically according to protein resolution.

254 records have been found. Record 0-20 >>next page

QUERY REPORT & GNM CALCULATION

PDBID	Resolution	Title	GNM
1A2A	2.80 Å	Agkistrotoxin, A Phospholipase A2-Type Presynaptic Neurotoxin From Agkistrodon Halys Pallas	1A2A
1A3D	1.80 Å	Phospholipase A2 (Pla2) From Naja Naja Venom	1A3D
1A3F	2.65 Å	Phospholipase A2 (Pla2) From Naja Naja Venom	1A3F
1AE7	2.00 Å	Notexin, A Presynaptic Neurotoxic Phospholipase A2	1AE7
1AH7	1.50 Å	Phospholipase C From Bacillus Cereus	1AH7
1AII	1.95 Å	Annexin III Co-Crystallized With Inositol-2-Phosphate	1AII
1AIN	2.50 Å	Crystal structure of human annexin I at 2.5 A resolution.	1AIN
1AKN	2.80 Å	Structure Of Bile-Salt Activated Lipase	1AKN
1AOD	2.60 Å	Phosphatidylinositol-Specific Phospholipase C From Listeria Monocytogenes	1AOD
1AOK	2.00 Å	Vipoxin Complex	1AOK
1AQL	2.80 Å	Crystal Structure Of Bovine Bile-Salt Activated Lipase Complexed With Taurocholate	1AQL
1AX9	2.80 Å	Acetylcholinesterase Complexed With Edrophonium, Laue Data	1AX9

Figure 2-8. iGNM-PDB integration snapshot: the results using 'phospholipase' as keyword are shown. The GNM information for all the retrieved structures is tabulated in the right column

oGNM Online Calculation Server

When the user performs a search for a PDB structure, the iGNM database is checked first for that structure's GNM results. If the structure's results are not found, an interface to the oGNM online calculation server is automatically provided.

oGNM takes as input a 4-digit PDB ID from the user's browser. It then retrieves the corresponding structure from PDB and performs online calculation. Once the calculation is complete the results are delivered to the visualization engine for visual presentation to the user (see Figure 2-6).

2.4. Other Protein Data Resources

2.4.1. Protein Structure Data Resources

PDB (Berman et. al, 2000) is the single international repository for the distribution of 3D macromolecular structure data primarily determined by X-ray crystallography and Nuclear Magnetic Resonance.

Based on PDB, there are many specialized structure databases for various purposes. For examples, the SCOP (Structural Classification of Proteins) database (Conte et al., 2000) provides information on close relatives of a given protein using keywords and homology searches; the CATH (Class, Architecture, Topology, and Homologous superfamily) database (Pearl et al., 2001) provides a hierarchical classification of protein domain structures at four levels: class, architecture, topology and homologous super-family; MMDB (Molecular Modeling Database) (Chen et. al, 2003) provides graphical summaries of the biological annotation available for each structure, based on automated comparative analysis.

2.4.2. Protein Dynamics Data Resources

In addition to iGNM (see Section 2.3), there are several other online data resources on protein dynamics, including MolMovDB (Echols et al., 2003), ElNémo (Suhre and Sanejouand, 2004), ProMode (Wako et al., 2004), MoViES (Cao et al., 2004), Dynamite (Barrett and Noble, 2005), and WEBnm (Hollup et al., 2005).

Table 2-2. Sample data resources on protein structural dynamics

System Name	Brief Description	Components	Model	Num. of Structures
MolMovDB (USA)	Presenting conformational changes using motion trajectories	Database; Calculation server	Interpolation between known Conformations	4400
ElNémo (France)	Presenting up to 100 slowest modes of studied structures	Calculation server	Simplified NMA	No database available
ProMode (Japan)	Presenting collective motions for 20 slowest modes	Database	All-atom NMA	1,442
iGNM (USA)	Presenting collective motions for 20 slowest modes and 20 fastest modes	Database; Calculation server	GNM	22,549

Table 2-2 shows features of some sample data resources. As can be seen, theses data resources are based on different methods and contain differing information about protein dynamics. For examples: MolMovDB contains protein motions (also referred to as "morphs") generated by interpolation between two known conformations (structures); ElNémo uses a simplified NMA model at the residue level; ProMode uses all-atom NMA and determines the motions in the space of dihedral angles, as opposed to Cartesian coordinates used in all other resources; iGNM uses GNM, which is also at the residue level, but yields N-1 normal modes' amplitudes, and not 3N-6 node vectors conventionally obtained by NMA.

These data resources have different components for providing dynamics. For examples: MolMovDB and iGNM have both database and online calculation server; ProMode contains a database; ElNémo only provides online calculation. In addition, these data resources have differing data coverage. Due to its simplicity and efficiency, the iGNM database contains the results for a significantly larger number of protein structures compared to other databases.

2.4.3. Protein Sequence Data Resources

Protein sequence data resources contain sequence information without detailed 3D structure information. For examples: SwissProt/Trembl (Bairoch et al., 2005) is a database that contains more than 2 million protein sequences; Pfam (Finn et al., 2006) is a database of multiple alignments of protein domains or conservation residue regions. Pfam consists of two sets of protein families: Pfam-A families are based on multiple alignments whereas Pfam-B is an automatic clustering of the SwissProt/Tremble.

Protein sequences that belong to the same family have conserved regions. Conservation profiles are a measure of the shared patterns that remain. The conserved regions revealed in profiles are useful for identifying sites that are important for structure and function (Valdar and Thornton, 2001). Traditionally they have been constructed from multiple alignments (MSA) using scoring matrices and weighted averages (Valdar and Thornton, 2001). This approach has been effective, but it also requires a chain of assumptions that may not be valid in all cases. There are many ways to generate scoring matrices and these matrices vary in their sensitivity to remote homologs (Johnson and Overington, 1993). Many proteins contain multiple domains or overlapping and/or nested domains that strongly influence alignment (Raghava et al., 2003). Sequences for multiple alignments often require preprocessing to eliminate low complexity regions (Wootton and Federhen, 1996).

A new algorithm based on n-gram patterns has been developed that avoids the assumptions associated with the MSA approach (Vries et al., 2006a). Due to the advantage of this algorithm,

it has been used to generate conservation profiles for more than 50,000 protein sequences (see Section 4.6 for more details).

2.5. Traditional Methods for Data Resource Integration

Currently, many bioinformatics resources (e.g., PDB and iGNM) are accessible to researchers through web application interfaces, e.g., through a HTTP form and a server-side processing script. There are three main ways to integrate data via web applications: data warehousing, data wrapping, and link-based integration (Stein 2003).

2.5.1. Data Warehousing

Data warehousing method brings all the data from different databases into a single database. There are two steps to construct a data warehouse: the first step is to develop a unified data model that can accommodate all the information contained in various data resources; the second step is to write a set of scripts to fetch the data from the source databases, transform them to the unified data model and then load them into the warehouse. The warehouse serves as a centralized database for answering any of the queries that source databases can handle, as well as cross-database queries that the individual databases cannot handle.

The major limitation of the data warehousing approach is that the warehouse is fragile to source database changes. New information is being continuously added to the source databases, which means that new data must be incorporated into the warehouse or the warehouse will be out of

date. Also, source databases may continuously be modified by adding new data types, changing fields and the relationships between data types. This means that scripts written for one version of a database may no longer work with a later version. For example, the Integrated Genome Database (IGD) project attempted to combine human sequencing data with the multiple genetic and physical maps using data warehousing approach (Ritter et al., 1994). On average, each of the source databases changed its data model twice a year, the IGD data import scripts often broke down and had to be rewritten. Because software maintenance became unmanageable, the IGD project eventually terminated (Stein 2003).

2.5.2. Data Wrapping

Most online biological databases provide HTTP user interfaces and underlying datasets using different data formats and access methods. To integrate these databases, the data wrapping method leaves the information in its source databases, but builds an environment or view on top of the databases that makes them seem to be part of a logic unit. To this end, the database community has developed various cross-database query languages. For example, Kleisli and K2 languages can analyze a given query to discover which databases need to be accessed to satisfy the request, and generates a set of subqueries to fetch data (Davidson et al., 2001). Despite the appeal of this approach, the data wrapping approach introduces the complexity of writing and maintaining the database drivers or wrappers. Therefore, languages such as K2 have not been widely adopted by the bioinformatics community (Stein 2003).

2.5.3. Link-Based Integration

Link-based integration often begins query with one data resource, and then creates hypertext links to related information in other data resources. In this approach, data resources either cooperate to create dependable linking rules or links to external databases can be manually inserted. The sequence retrieval system (SRS) is an example of link-based integration for biological databases. SRS is more sophisticated than general web-based search tools (e.g., Google) because it allows users to explicitly relate a field in one database to a differently named field in another database (Zdobnov et al., 2002). However, link-based integration is problematic because it is vulnerable to naming ambiguities. For example, a user might interpret the name of links differently than the developer and wander into the wrong page. Also, links to external databases may fail if external databases no longer function.

2.6. Web Services and Data Resource Integration

Traditional web application-based methods (e.g., data warehousing) are tightly-coupled with data resources. Users need to have prior knowledge about data resources and write scripts to parse HTML code to extract data while ignoring explanatory text and graphics. This tightly-coupled approach is labor intensive and fragile for minor changes in the HTML code of a given web application may cause failure (Stein 2002). In addition, most methods only focus on the data integration aspect without providing means to integrate computational and visualization tools such as molecule search engines (e.g., PDB) and homology search tools (e.g., BLAST) (Lacroix et al., 2003).

Web services, on the other hand, offer a loosely-coupled environment for dynamic integration of various data resources and software components (see Table 2-3). Web services provide XML-based programmatic interfaces and communication protocols for user applications to obtain explicit results without tedious HTML code parsing. Also, web services provide protocols and tools that facilitate the discovery and aggregation of data resources. This section reviews current web services technologies and web services-based methods for data resource integration.

Table 2-3. Web Applications vs Web Services

	Web Applications	Web Services
Design	User-to-program interaction	Program-to-program interaction
	Workflow is determined by developer	Application determines workflow
	Static integration of components	Possibility of dynamic integration of components
	Monolithic service	Possibility of service aggregation
Developer	Existing web applications are based on either a specific platform	Doesn't assume any specific platform or programming paradigm
	HTML is the primary data format	XML is the foundational enabling technology
	Application is more difficult to maintain	Maintenance of services is easier
User	Application is described in graphic user interface (GUI)	Individual services are self describing via programmatic interfaces
	GUI is important	GUI may not be necessary

2.6.1. Web Services and Service-Oriented Architecture

The term “Web services”, as defined by the World Wide Web Consortium (W3C) (http://www.w3.org/), refers to “programmatic interfaces” used for Web application to application communication. Web services represent an emerging distributed computing paradigm that differs from other approaches such as Common Object Request Broker Architecture

(CORBA), Remote Procedure Call (RPC) and Java Remote Method Invocation (RMI) in that it emphasizes Internet-based standards to address heterogeneous distributed computing (Foster et al., 2002). The aim of Web Services is to exploit XML technology and the Web by integrating applications that can be published, located and invoked over the Web. An example of Web services for biology is myGrid which is a middleware for in-silico experiments (Lord et al., 2004). It contains a number of Web services, such as XEMBL service which displays data from the EMBL (European Molecular Biology Laboratory) Nucleotide Sequence Databank in XML formats (http://www.ebi.ac.uk/xembl/).

Service-Oriented Architecture

Web services interact with each other and user applications using the Service-Oriented Architecture (SOA). There are three entities in any SOA (see Figure 2-9): service providers, service requestors, and service registry. A service provider is responsible for generating a service description, publishing that description to one or more service registries, and responding invocation messages from service requestors. A service registry acts as a central location for registering all services. A service requestor is a customer of a Web service that can be either a human or a software agent. The requestor checks for a service description in a service registry and then binds to the Web service if found. The responsibility of a service registry is receiving service descriptions from service providers and matching them with requestors' service requests.

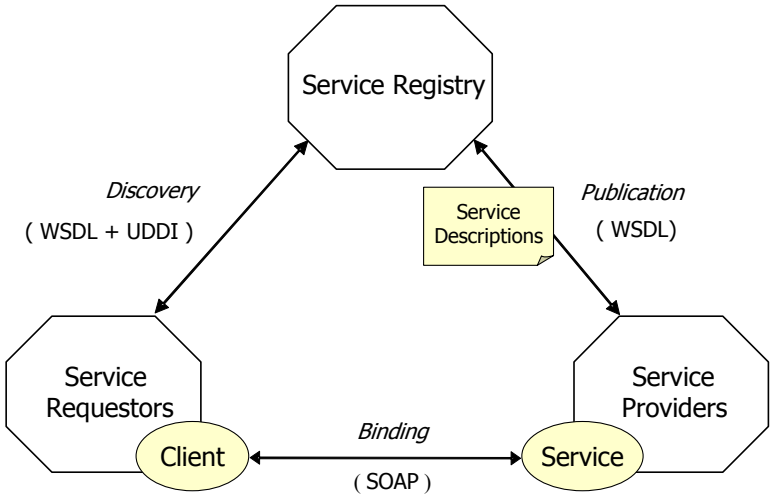


Figure 2-9. Service-Oriented Architecture

Several technologies are involved in enabling interactions within the web services architecture, including SOAP (Simple Object Access Protocol) (W3C SOAP, 2001), WSDL (Web Service Description Language) (Christensen et al., 2001), and UDDI (Universal Description, Discovery and Integration) (UDDI, 2000). A stack diagram of these component technologies is illustrated in Figure 2-10.

The lowest layer in the stack diagram is the communications layer which includes standard Internet protocols such as HTTP. The layer above the communication level is the messages layer, representing how a message is exchanged between providers and requestors. SOAP is the framework for functionality in this layer. The description layer is for creating a common understanding of message structure and data types for both service providers and requestors. WSDL is currently used to describe the invocation syntax of Web services. The processing layer contains high-level tools used for service discovery and composition. UDDI is one of the

techniques which could be used in this layer. In addition, composition languages and standards, such as WSFL (Web Services Flow Language) and BPEL4WS (Business Process Execution Langue for Web Services), are also used in this layer for representing workflows. The details of WSDL, SOAP, UDDI and BPEL4WS are discussed below.

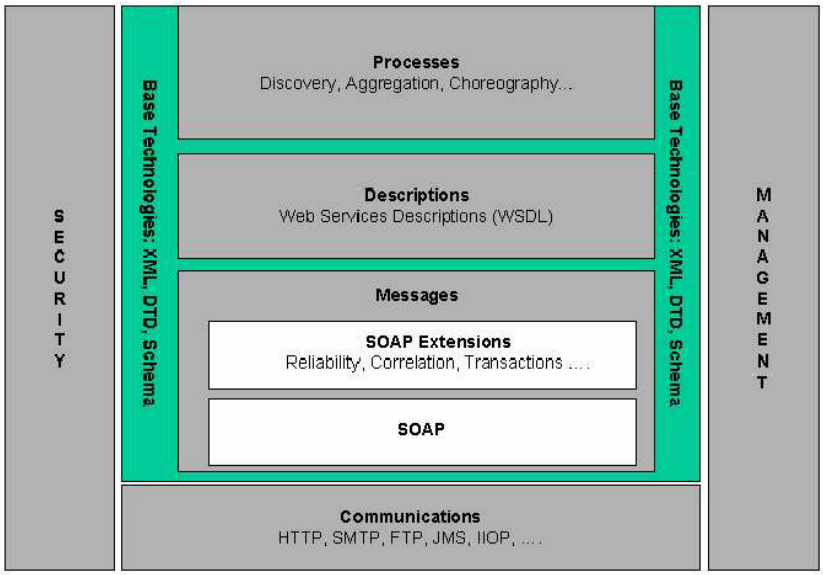


Figure 2-10. Technology stack diagram for Web services (Booth et al., 2003)

Web Services Standards

- ***WSDL***

WSDL is an XML-based standard for describing application services that use a standard messaging layer such as SOAP. A WSDL description is a collection of ports for providing operations. In WSDL, abstract definitions of data for exchange are called messages, and abstract collections of operations are called port types. The protocol and data format specifications for a particular port type form a reusable binding.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="Blast">

  <message name="searchSimple1In">
    <part name="program" type="xsd:string" />
    <part name="database" type="xsd:string" />
    <part name="query" type="xsd:string" />
  </message>
  <message name="searchSimple1Out">
    <part name="Result" type="xsd:string" />
  </message>

  <portType name="Blast">
    <operation name="searchSimple" parameterOrder="program database query">
      <documentation>Execute Blast</documentation>
      <input name="searchSimple1In" message="tns:searchSimple1In" />
      <output name="searchSimple1Out" message="tns:searchSimple1Out" />
    </operation>
  </portType>

  <binding name="Blast" type="tns:Blast">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="searchSimple">
      <soap:operation soapAction="searchSimple" style="rpc" />
      <input name="searchSimple1In">
        <soap:body use="encoded" namespace="http://tempuri.org/Blast"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output name="searchSimple1Out">
        <soap:body use="encoded" namespace=http://tempuri.org/Blast
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>

  <service name="Blast">
    <port name="Blast" binding="tns:Blast">
      <soap:address location="http://xml.nig.ac.jp/xddbj/Blast" />
    </port>
  </service>

</definitions>
```

Figure 2-11. Sample BLAST WSDL description, adapted from http://xml.nig.ac.jp/wsdl/Blast.wsdl, which is provided by XML Central of DNA Data Bank of Japan (DDBJ).

Figure 2-11 shows a sample WSDL description for Basic Local Alignment Search Tool (BLAST) (Altschul et al., 1990; Altschul et al., 1997), which is the tool most frequently used for calculating biological sequence similarity. In the WSDL description, the port type contains an operation (Java class method) called “searchSimple”, the input message “searchSimpleIn” contains three parameters called “program”, “database” and “query”, the output message

"searchSimpleOut" contains one parameter called "Result", and the binding uses SOAP messages on top of the HTTP protocol.

- ***SOAP***

SOAP is an XML-based protocol for exchanging information in a distributed environment using typed message and remote invocation. SOAP supports a framework for describing what is in a message and how to process it, and there is a set of rules for encoding instances of data types. This protocol is specific mostly to Web services over HTTP. A Web service could interact with remote machines through HTTP's post and get methods, but SOAP is more robust and flexible.

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. The SOAP envelope defines an overall framework for expressing what the message contains, who should deal with it and whether it is optional or mandatory. The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types. The SOAP RPC representation refines a convention that can be used to represent remote procedure calls and response.

- ***UDDI***

UDDI is a set of services that support the description and discovery of web services. It is the technical interface, or "yellow page", to access available services. UDDI contains four core elements:

(1) Service provider (business) information. Service provider information is described using the "BusinessEntity" element representing a physical organization. It contains information, such as name, description and contacts, about the organization.

(2) Service information. Service information is described using the "BusinessService" element, which groups together a set of services provided by an organization.

(3) Binding information. Binding information is described using the "BindingTemplate" element, which contains information relevant for application programs that need to connect to and then communicate with a remote Web service. The instructions can be in the form of WSDL or a text-based document.

(4) Service specification. Specification for services is described using the "tModel" element, which supports the registration of attributes of services. In general tModels have two functions: tagging the type of service advertised and providing abstract keys to be associated with a service specific value. For example, in myGrid service registry, the myGrid ontology is represented as a tModel (myGrid, 2002).

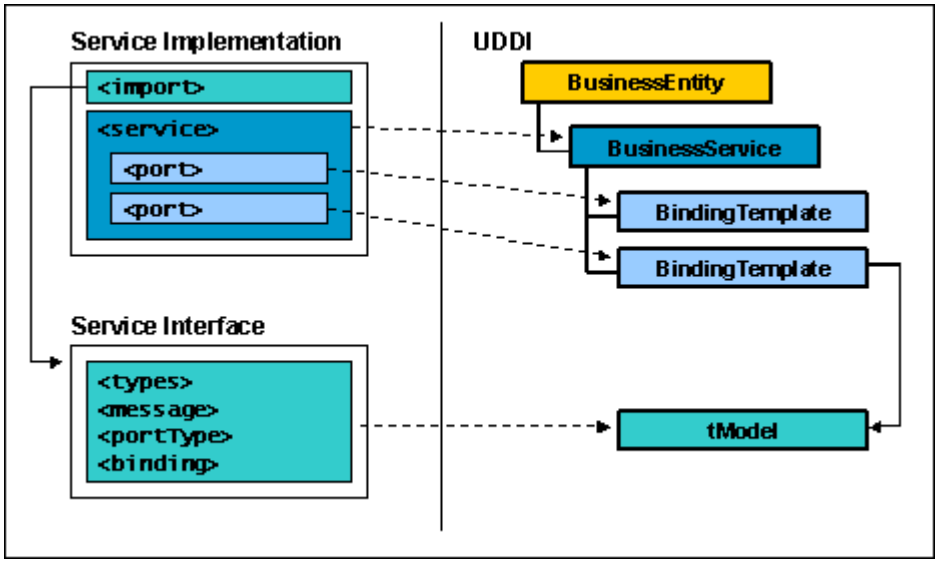


Figure 2-12. WSDL to UDDI mapping (Brittenham et al., 2001)

WSDL descriptions are published in UDDI through WSDL-UDDI mapping (see Figure 2-12). For example, each service element in a service implementation document is used to publish a UDDI "BusinessService", and a service interface is published as a tModel in a UDDI registry (Brittenham et al., 2001).

Workflow

Workflows are representation of structured activities or processes (Singh and Vouk, 1996). A biological process typically involves the invocation of a series of activities that are invoked in a routine manner. In workflow management, output from one task is fed as input to the next task with additional parameters, if necessary. The intermediate results are checked for consistency and validated to ensure that the computation as a whole remains on track.

A workflow task must be explicitly represented (e.g., as Web services) to enable effective intervention. In the Web services architecture, executable processes are used to execute workflows by referencing port types contained in WSDL documents. There are several languages for specification of executable processes. Examples are WSFL (Web Services Flow Language) which provides a process model to glue WSDL services together and specify the order in which operations execute (Staab et al., 2003; Foster et al., 2002) and BPEL4WS (Business Process Execution Langue for Web Services) which enables specification of executable processes (composed of Web services) in terms of execution logic or control flow (Mandell and McIlraith, 2003).

2.6.2. Semantic Web Services

The Semantic Web, coined by Berners-Lee (1998), provides a common interoperable framework in which information is given a well-defined meaning such that data and applications can be understood and reasoned by machines for more accurate and automatic discovery and integration of information resources across organizational boundaries (Berners-Lee, 1998). To make Web services capable of handling semantic interoperation, the semantic Web community has combined semantic markup languages grounded in Description Logics with current Web service standards. This has led to semantic Web services, one type of Web services that can express not only interfaces among services but also their capabilities (i.e., inputs, outputs, preconditions and effects) using ontologies (McIIraith et al., 2001; Paolucci, et al., 2002).

Ontologies describe concepts and their relationships for a domain (Gruber, 1995). Several ontology languages have been developed for describing ontolgoies. For example, OWL (Web Ontology Language) (W3C OWL, 2004) is a Web ontology language that is widely used to formally describe various types of concepts and relationships. Based on OWL, OWL-S (OWL-based Web Service Ontology) (W3C OWL-S, 2004) is a set of interrelated OWL ontologies that provide a set of well-defined terms for describing Web services (see Section 2.5.1). By having service descriptions and requests refer to the same ontological concepts, semantic matching algorithms can be designed to reason about similarities between services and requests in an unambiguous and machine-interpretable form (Paolucci et al., 2002). Thus services that provide

solutions to a request can be automatically discovered even if the services and the service requested are syntactically distinctive.

A typical semantic Web services architecture contains five major components: (1) service interfaces using some form of programmatic access; (2) semantic descriptions about service capabilities; (3) a domain ontology which provides terms or key concepts used with semantic descriptions of Web services; (4) a registry service that advertises the availability of services and searches over the semantic descriptions made available to it; and (5) messaging that enables service requestors to treat data from different service providers in a uniform fashion (Lord et al., 2004).

OWL-S Framework

The most prominent semantic Web services framework is OWL-S. OWL-S is an OWL-based upper ontology that describes three key aspects of a service: "service profile" which states abstract description of service capabilities; "service model/process" which states how requestors interact with services; and "service grounding" which states actual messages that are exchanged among services (see Figure 2-13) (Paolucci et al., 2002). OWL-S supports semantic representation of services through its tight connection with OWL. OWL supports subsumption reasoning on taxonomies of concepts. It also facilitates relations definition between concepts.

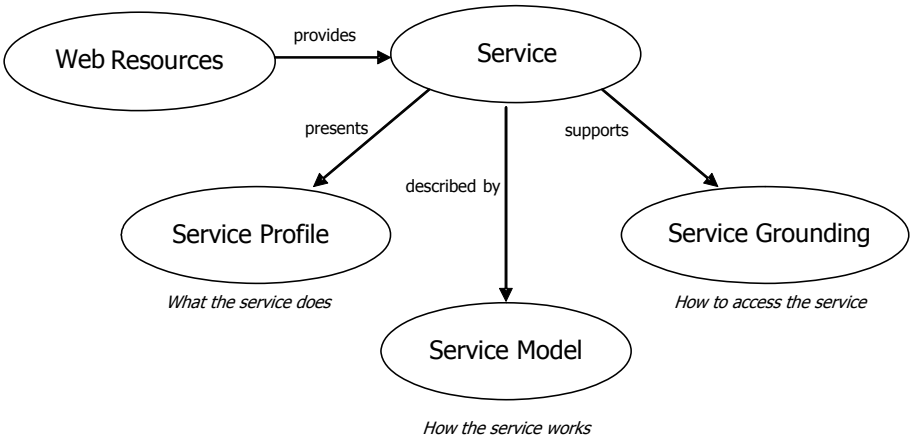


Figure 2-13. OWL-S upper ontology

2.6.3. Web Services Based Methods for Data Resource Integration

Several methods have been proposed to enable automatic or semi-automatic integration or composition of web services. These methods usually fall in the realm of workflow composition or AI planning (Rao and Su, 2004), see Figure 2-14.

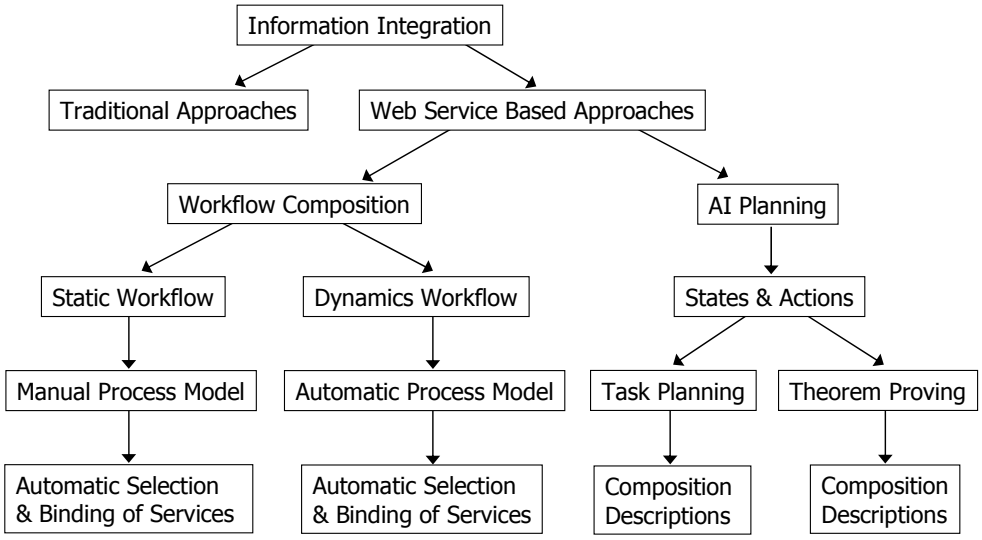


Figure 2-14. Hierarchy of information integration problems

The goal of workflow composition is to create a composite service which is a set of atomic services together with the control and data flow among the services. For the workflow based methods, the workflow can be generated either statically or dynamically. In the static workflow category, users have to manually create an abstract process model, while the selection and binding of services is done automatically. In the dynamic workflow category, both the abstract process model and the section and binding of services are done automatically.

The AI planning based methods usually assume relevant service descriptions are already loaded into the reasoning engine. Traditional AI planning techniques such as theorem proving (McIlraith and Son, 2002; Ponnekanti and Fox, 2002) and hierarchical task planning (Wu et al., 2003) are applied to derive the final composition descriptions.

Workflow composition requires service discovery or searching all services to find most relevant services for workflow tasks. The major issue is that when the number of all the services is large, the discovery process may not be efficient (Constantinescu et al., 2004). AI planning does not require service discovery. However, it requires users to have prior knowledge about all the services, which may not be feasible.

As discussed in Sections 2.6.1 and 2.6.2, several initiatives have been conducted to provide platforms and languages that will allow web services based integration. In particular, WSDL, OWL-S, UDDI, and SOAP define standards ways for service discovery, description and invocation; BPEL4WS and OWL-S service model provide means for representing web service integrations or compositions.

Despite all these efforts, web service composition still is a complex task. First, the number of available services has increased dramatically in the recent years. Finding suitable services that provide a solution to the problem at hand becomes more complex than ever. Second, services can be created and updated on the fly. The composition system needs to detect the updating at runtime and the decision should be based on up-to-date information. Third, services are developed by different organizations, which use different concept models to describe services; and different data schema for service parameters (Rao and Su, 2004).

2.7. Web Services in Bioinformatics

Recently, interest in applying Web services and semantic Web technologies to bioinformatics has grown (see Table 2-4 for example projects). For example, PDB is currently in the early implementation stage of providing Web services that will allow users to use XML and SOAP to perform queries and retrieve results programmatically from the PDB Beta Web site (Deshpande et al., 2005); BSML (Bioinformatics Sequence Markup Language (http://www.bsml.org/) is an XML that encodes biological sequence information and includes graphic representations of sequences, genes, electrophoresis gels, and multiple alignments (Spitzner, J.); SBML (Systems Biology markup language) is a language for representing biochemical reaction networks serving as a standard exchange format for computational models of biochemical networks (Hucka et al., 2003); BioSimGrid is a project for archiving and analyzing MD trajectories and making them accessible to the biological community (Tai et al., 2004); the myGrid (Lord et al., 2004) project aims to provide high-level middleware to support personalized in-silico bio experiments using

semantic Web services technologies, namely WSDL and UDDI; the MOBY services (Lord et al., 2004) use user created and centrally stored Gene Ontology (GO) style ontologies for semantic descriptions; the Semantic-MOBY (Lord et al., 2004) services use semantic Web technology to improve automation; the N-Glycosylation Process (NGP) project (Sahoo et al., 2005) uses Web services to expose computational tools for different Web process phases and uses two glycomics domain ontologies to annotate data with different formats such as image and raw data.

Table 2-4. Sample data resources using Web services/semantic Web technologies

Data Resources	**Brief Description**	**Web Services**	**Semantic Web**
PDB	A public data resource of protein structures	Yes	No
BSML	Bioinformatics Sequence Markup Language	No	Yes
SBML	Systems Biology markup language	No	Yes
BioSimGrid	Data resource for archiving and analyzing MD trajectories	No	Yes
myGrid	A high-level middleware for personalized in silico bio experiments	Yes	Yes
MOBY	A prototype semantic web service for the investigation of biological problems in different organisms	Yes	Yes
NGP	A semantic web service for N-glycosylation process	Yes	Yes

3. A SEMANTIC WEB SERVICES INFRASTRUCTURE FOR DISTRIBUTED PROTEIN DATA INTEGRATION

3.1. Introduction

Bioinformatics research relies on techniques for describing and performing experiments that retrieve data from distributed resources. Such computational techniques, or *in silico* experiments, are an important tool for researchers because they can validate and motivate experiments performed in conventional laboratories or "wet" laboratories (Hull et al., 2005).

Understanding the function of every protein is one of the major objectives of bioinformatics. Currently, there is a lot of information (e.g., sequence, structure and dynamics) being produced by experiments and predictions that are associated with protein function. Integrating these diverse data resources about protein sequence, structure, dynamics and other protein features allows further exploration and establishment of the relationships between protein sequence, structure, dynamics and function, and thereby controlling the function of target proteins (Marcotte and Date, 2001). However, researchers from different organizations develop algorithms, tools and resources on protein data—often with no thought on how other researchers are doing the same tasks. Consequently, an integration problem may require interactions with many incompatible data resources with many different interfaces (Gao et al., 2005).

Currently, researchers use *ad hoc* scripts and programs to retrieve data from these distributed and heterogeneous data resources (see Section 2.5 for more details). There are no systematic methods

for accessing and integrating those heterogeneous data resources for collaborative research and large-scale knowledge discovery. Also, there are no available architectures and tools that allow automatic discovery, selection and invocation of accurate data resources that meet specific research requirements from users' perspectives. To facilitate knowledge discovery and stimulate collaboration in the research community, architectures that provide efficient and automatic solutions for biological data resource integration are critical.

In this research, a Web services infrastructure (WSP) for flexible integration of various protein data resources, is presented. The main features of this architecture include component-based design of service functionalities and semantics-based description and matching of web services using ontologies (i.e., domain and upper service ontologies). The architecture allows researchers to conveniently discover and assemble various types of protein data (both existing and yet to come) for their applications (e.g., determining the function or other features of proteins).

This chapter discusses the challenges for distribute protein data integration, WSP's strategies for optimal integration, WSP's computing platform and major components, and the WSP deployment process. The details of WSP's major components are discussed in the Chapters 4–7.

3.2. Protein Data Resource Integration: Challenges

Stein (2002) envisioned a "bioinformatics nation" in which previously fragmented organizations provide standard web service interfaces to their resources so that performing *in silico* experiments is potentially quicker and easier than the *ad hoc* methods (e.g., data warehousing). Recently, the interest of applying web services in bioinformatics has grown. For example, as of 2005, the myGrid project has a registry of more than 1000 web services provided by a wide range of third parties (Hull et al., 2006). The number of available web services will continue growing as the impetus behind the technology grows (Greenbaum et al., 2005).

Despite the advantageous features of web services (see Section 2.6), there are a series of challenges to apply web services in protein data resource integration:

(1) Unlike traditional web applications, the purpose of developing web services is to wrap reusable software components with standard programmatic interfaces so that application programs can automatically obtain data/results. However, current research only focuses on delivering the final results to end users without exposing some intermediate data that can benefit other users. To maximize the utilization of available resources that provide protein data, the first challenge is how to define common elements in protein data resources that can be shared by different applications.

(2) Users usually have specific requirements about the data used for their applications. However, service providers and users may have distinctive perspectives and knowledge about one service resulting in differing descriptions for the service. In this case, syntactic-based matching cannot locate the service. In order to allow semantic interpretation of dissimilar

data and query expressions, common concept models (i.e., ontologies) have to be adopted. The challenge is what ontologies provide the necessary terminologies for expressing various perspectives/requirements related to protein data and data integration.

(3) As number of biological Web services increases, finding suitable services that provide a solution to the problem at hand becomes more important than ever. The XML-based web services standards (i.e., WSDL and UDDI) do not support operations at semantic level, leaving the promise of automatic discovery and integration of services incomplete. The challenge is how to design service matching algorithms that can efficiently retrieve the most accurate service required to perform a given task.

(4) An *in silico* experiment or integration problem usually involves many data resources or services. For an integration problem, bioinformatics researchers usually have a high-level workflow without knowing a concrete and executable web services workflow. Because services are developed by different organizations, they may have different concept models and data schema for service parameters (i.e., inputs and outputs). Semantic interoperability (interoperable concepts) and data interoperability (interoperable data schema) become major issues. The challenge is how to design automatic means of integrating heterogeneous data resources with little or no user interventions.

3.3. Methodologies for Optimal Integration

WSP exploits the potential of semantic web services to address the aforementioned challenges. The goal of WSP is to provide optimal solutions for protein data resource integration. In this research, optimal solutions include three major components: (1) web services granularity, (2) matching accuracy, and (3) automation.

(1) ***Granularity***. Web services granularity means the identification of common atomic components which can be published as web services (Yang 2003). The goal is to maximize the utilization of available data resources. In WSP, atomic components are identified as protein features, which are functions of a protein sequence. This is based on the observation that the core of most protein research is related to some biological sequences. Whatever features proteins have, they are functions of some operations of a position in a sequence. Examples are: conservation is a function which tells how each residue in a sequence is repeated across the protein family; enzyme active site is a function which tells whether a residue in a sequence is an active site or not; slow mode is a function of fluctuation of each residue in a sequence. Therefore, protein features represent common atomic components in protein data resources that can be shared by many different studies. For example, enzyme active site data can be used for correlating with other protein features and also for developing prediction models. Refer to Chapter 4 for details.

(2) ***Accuracy***. Matching accuracy is a measure of how a service request semantically matches a service. Unlike syntactic matching which operates on unannotated descriptions, WSP relies on semantic annotation and matching to discover services. By having service parameters (i.e., inputs and outputs) in requests and service descriptions refer to the concepts in the same

domain ontology, a semantic matching algorithm is designed to reason about similarities between requests and service descriptions in an unambiguous and machine-interpretable form. WSP ranks matched services based on their similarity score to the request. Services with the highest similarity score represent the most accurate solutions. Refer to Chapters 5 and 6 for details.

(3) ***Automation***. A set of methodologies are designed to enhance the level of automation for integration problems. First, protein feature data are modeled as web services with programmatic interfaces so that user applications can automatically call exposed functionalities and obtain explicit results without tedious HTML code parsing. Secondly, a semantic matching mechanism is used to locate most accurate services so that users do not need to have prior knowledge about data resources. Especially, services that provide solutions to a request can be automatically discovered even if the services registered and the services requested are syntactically distinctive. Thirdly, a service chaining strategy is adopted to automate the process of data integration for *in silico* experiments. One way of performing *in silico* experiments is to pipe together inputs and outputs of consecutive web services in a workflow environment. Usually, scientists would like to create a high-level "abstract" workflow and not bother about low-level details of web service, e.g., urls, parameter passing, data transformations, and control flow (Ludascher et al., 2003). Due to the nature of scientific workflows, a static-workflow based approach is chosen for service integration (see Figure 3-1). In this approach, a higher-level workflow is created by the user, and the system is responsible for mapping the higher-level workflow to an invokable workflow which consists of real web services. Refer to Chapter 7 for details.

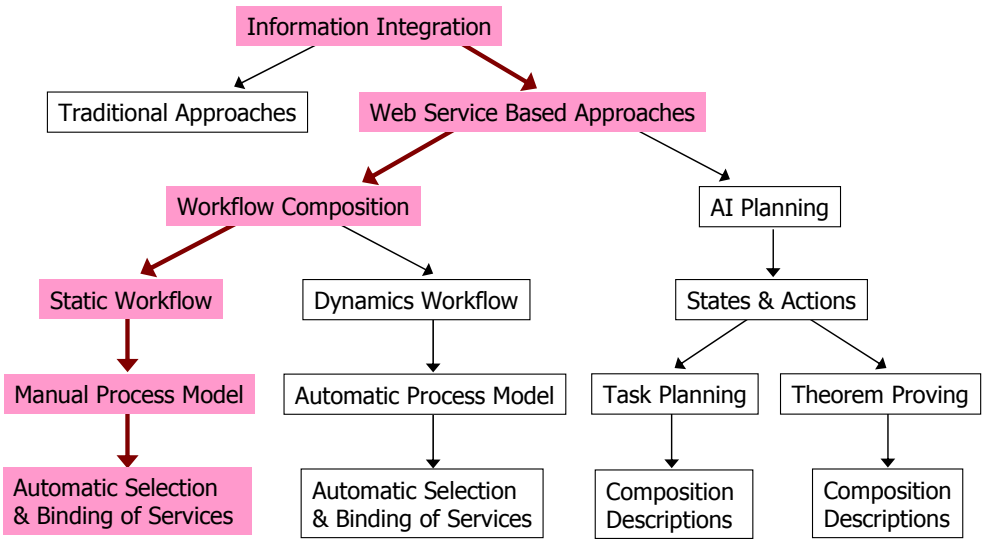


Figure 3-1. A hierarchy of information integration problem. The shaded paths and blocks represent WSP's integration approach

3.4. Computing Platforms

WSP is a web services environment where protein data resources are wrapped with uniform WSDL programmatic interfaces and SOAP is used as the protocol for communications between these data resources. Unlike generic service oriented architectures, UDDI is not used as the registry for locating services because it can only perform syntactic but not semantic matching between user requests and services. Instead, a semantic registry is designed to meet the requirement of providing accurate solutions from user's perspectives. This matchmaker registry uses current Semantic Web standards such as OWL and OWL-S to capture the capabilities of web services and the data requirements of user applications.

3.5. Architecture, Components and Tools

As any Web services architecture (see Section 2.6.1), WSP has three entities: the service provider, the service requestor, and the service registry (see Figure 3-2). The provider refers to an organization that provides protein data and Web services. The requestor refers to users (researchers) that require protein Web services. The registry refers to a middle registry service that contains registered services and facilitates discovery of services requested.

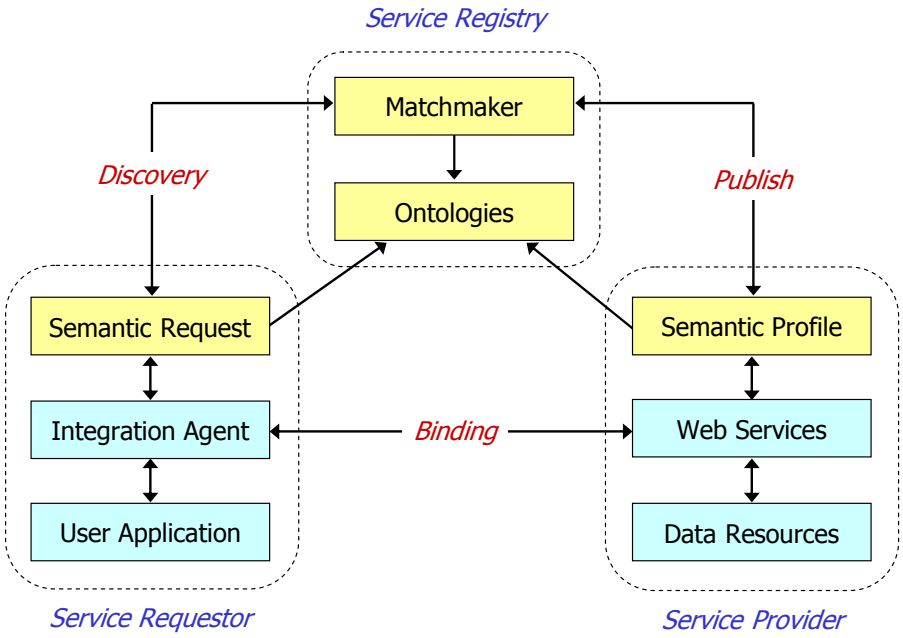


Figure 3-2. Mapping of Components to WSP Architecture

On the service provider side, there is a set of components, including data resources which provide protein feature data, tools for developing web services, tools for semantic description and publication of services. On the service registry side, there is a matchmaker service and

community-agreed domain ontologies. On the service requestor side, there are tools for representing integration problems (*in silico* experiments), tools for semantic description of service requests and an integration agent which interacts with the service registry and the service provider to obtain results.

These components and tools can be classified into four categories: tools for biological data resources and web services; tools for semantic description of web services; tools for semantic publication and matching of web services; and tools for chaining of web services. The following sections provide a brief review for each category.

3.5.1. Biological Data Resources and Web Services

In WSP, protein data resources are modeled as Web services by providers. Web services represent reusable components with certain query functionalities. The query functionalities (operations) are data and application specific. For sharing of protein data, it is expected that a service's basic (atomic) operation is a functionality that provides a certain protein feature. A complex service may contain several such functionalities.

In this research, two methods for developing biological web services are considered. One method is based on the existing Web applications of protein data. The other method is about building web services from scratch. Despite their differences, the common goal is to identify appropriate query functionalities and implement them as WSDL programmatic interfaces.

For example, iGNM's functionality for providing dynamics information (mode shape) is modeled as a Mode Shape Service with a WSDL interface. To invoke the Mode Shape Service, a SOAP client is written to acquire the service's WSDL description and generate a SOAP request to the service. Similarly, any protein data resources that provide protein features can be modeled as web services. These resources include other protein dynamics data resources besides iGNM, protein sequences data resources, structure and function data resources.

Web services can be implemented in many different platforms (e.g., Microsoft .NET, IBM WebSphere/J2EE) according to the web services standards. In WSP, the Apache Axis (http://ws.apache.org/axis/java/) is adopted as the development platform because it has a SOAP engine for processing service requests and toolkits for generating WSDL interfaces.

3.5.2. Semantic Description of Services

The purpose of semantic description of web services is to describe service capabilities (what a service can do) and user requests (what a user wants) in machine-understandable format so that algorithms can be designed to automatically discover semantically matched services.

There are two ways to semantically describe service capabilities. The first one assumes onotlogies that provide an explicit representation of the operations (tasks) performed by web services. In those ontologies, each operation is described by a different concept. The second one describes web services by the state transformation and the information transfer that they produce. In this case, the operation is implicitly represented by the state transformation. Since web

services can perform many different operations, the first way will lead to very large ontologies that is unmanageable and may not scale up when new capabilities become available. The implicit representation does not suffer from those shortcomings since they only require concepts that describe the domain of the web services (Sycara et al., 2003). Therefore, implicit representation of service operations is used in this research.

WSP provides an extended version of the Protein Ontology (PO) (Sidhu et al., 2005) and a semantic profile generator that allows a wide range of semantic descriptions of web services on complex protein data related issues. PO is implemented in OWL language using the Protégé ontology editor (Protégé, 2002). The service descriptions are implemented in OWL-S data format using the profile generator. While the protein ontology represents common perspectives of the research community, the descriptions of web services are based on individual researchers' perspectives and data requirements.

3.5.3. Semantic Publication and Matching of Services

A semantic matchmaker is used to identify accurate services for service requests. The matchmaker includes a client-server architecture. The server is a centralized registry service. It implements two algorithms, one for publication and one for matching of services. There are two SOAP clients for interacting with the registry services. One client is designed for service providers to publish their service descriptions to the registry. The other client is designed for users to send their requests and obtain information (i.e., URL) about semantically matched services.

The matchmaker is capable of locating the most accurate services among all registered services. That is when there are exactly matched services, it will be automatically discovered. When there are no exact matches, the most semantically accurate services will be provided.

3.5.4. Chaining of Services

Scientific workflows are usually subject-specific and exploratory-based (require an iterative procedure) (Rygg et al., 2005; Ludascher et al., 2003). It is important for users to interact with the integration program during the generation and execution of the web service workflow.

Due to the nature of scientific workflows, an integration problem is represented as an abstract workflow of service requests (tasks). The abstract workflow defines data dependencies between services and the control flow (i.e., order of execution). Each request is expected to be implemented by a web service.

An integration agent (a program that performs integration tasks) is used to select and chain services. First, requests are generated using the profile generator. Then the agent will send the requests to the matchmaker and determine a concrete web services workflow. Since each request may return several semantically matched services (candidate services), the agent needs to perform service selection. WSP adopts two selection criteria: (1) select the service with the highest score (most accurate service) for each request and (2) when there are multiple services with the highest score (equivalent services) for a request, select a service that is compatible with

the previously selected service. Here "compatible" means that the input data schema of the current service is the same or similar to the output schema of the previous service. WSP performs three types of comparisons between the output and the input of adjacent services, including all-or-nothing compatibility, subset compatibility and arbitrary compatibility (Spillner et al., 2006).

By combining the two selection criteria, WSP handles both semantic interoperability (interoperable concepts) and data interoperability (interoperable data schema) at design time. However, due to the nature of scientific workflows, users are expected to interact with the integration agent to produce a production web service workflow (e.g., modifying requests in the abstract workflow). A production workflow is fully automatic. It works as a logic unit or a global data resource so that automatic integration of various types of protein data can be achieved.

3.6. The Deployment Process

The purpose of this section is to compare the current approach and the WSP approach for protein data integration.

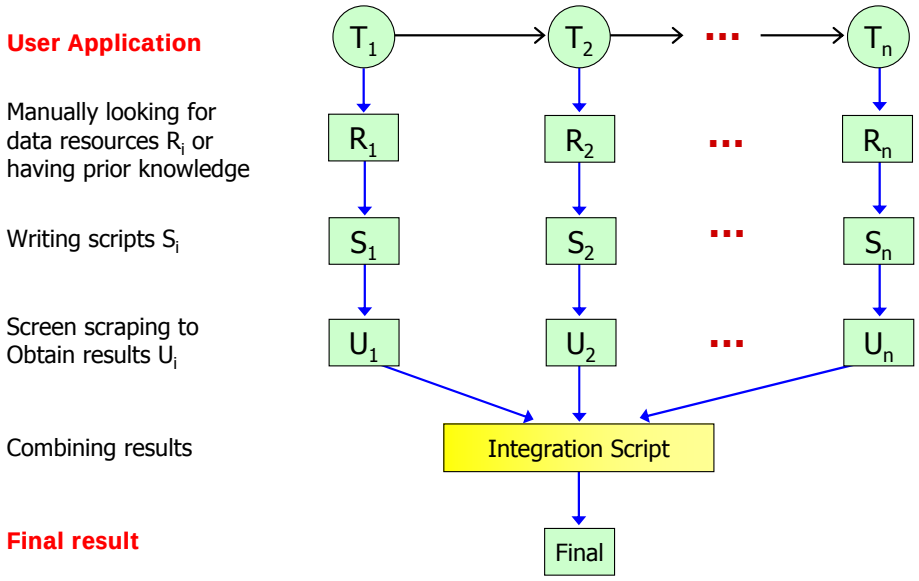


Figure 3-3. The current ad hoc approach for data integration

With the current approach for integrating data resources (see Figure 3-3), researchers are required to either have prior knowledge or manually look for data resources (i.e., Web applications) for workflow tasks. Then they have to write a screen-scraping script for each selected data resource to obtain a semi result. These screen-scraping scripts are non-reusable because different Web applications have different HTML coding and a script written for a given Web application cannot be used for other Web applications. The intermediate results from all screen-scarping scripts are then combined to form the final result. The integration script is also non-reusable because it is designed according to the specific data formats of intermediate results and cannot handle new results. The iGNM-PDB integration (see Section 2.3.3) shows an example of screen-scraping based data integration.

Given an integration problem (i.e., integrating protein features), WSP uses the OWL-S profile generator to generate service requests. It then uses the matchmaker to discover data resources (i.e., web services) that semantically match the user's requests. Finally, the integration agent is used to select and chain services to obtain the final result (see Figure 3-4). The profile generator, the matchmaker and the integration agent are all reusable components because they can handle different service requests by exploiting standards such as OWL-S upper ontology and WSDL interface.

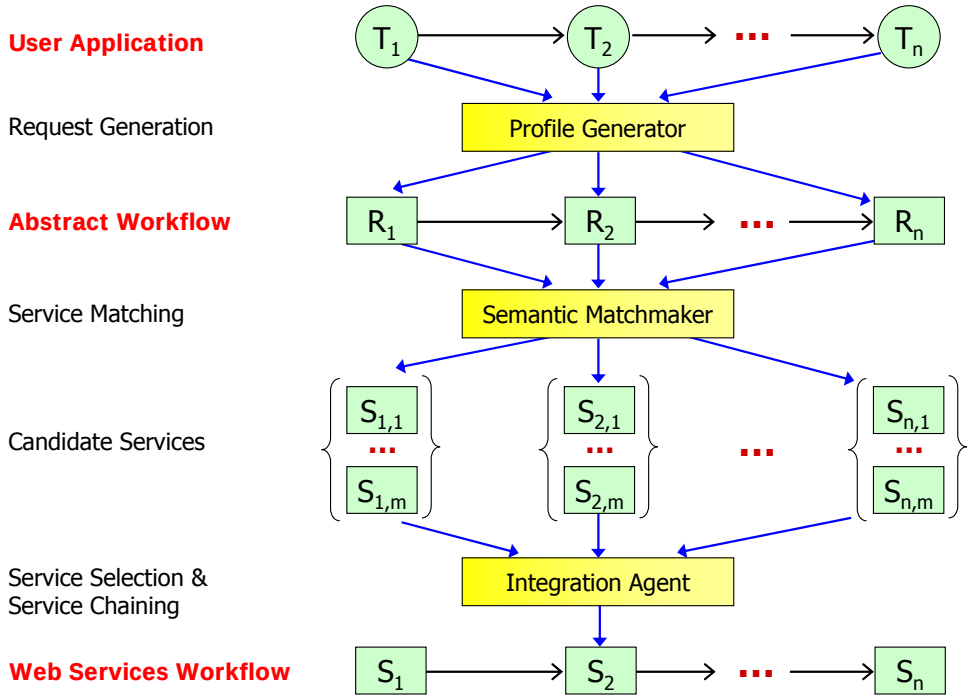


Figure 3-4. The deployment process of WSP

The WSP approach has several advantages over the current web application-based approach for protein data resource integration:

- *Desired solutions from user's perspectives*. By exploiting the potential of semantic web and ontologies, WSP allows users to discover desired services among all registered services.

- *Higher level of automation*. With the WSP approach, researchers only need to specify their service requirements (i.e., desired output) without having to manually look for data resources. The system will automatically discover, select and integrate appropriate services.

- *No duplication of effort*. With the current approach, researchers are required to perform such tedious tasks as writing scripts to parse the HTML code from every data resource (i.e., Web application). The scripts are non-reusable and new scripts have to be developed when new data resources are incorporated. With the WSP approach, reusable components such as the matchmaker and the integration agent are used to handle any data resources (i.e., Web services). There is no development overhead when new data resources are incorporated.

- *Less fragile to Web site changes*. In the current approach, the screen-scraping scripts are fragile for a minor change in the HTML code of a given Web application may cause failure. With the WSP approach, data resources/services are discovered and integrated dynamically. Whenever a service changes or fails, alternative services will be selected and invoked without modifying existing software components.

4. MODELING PROTEIN WEB SERVICES

4.1. Introduction

Web services are self-contained, self-describing, modular applications that can be published, located, and dynamically invoked across the Web (Gao et al., 2005). By modeling protein information resources (e.g., iGNM) as Web services, different types of biological data can be dynamically assembled from multiple network-enabled Web services for a variety of user applications.

This research contributes two biological web services: the iGNM web service that provides protein dynamics data for more than 20,000 protein structures; and the N-gram web service that provides conservation profiles for more than 50,000 protein sequences. These two services demonstrate the process of developing and utilizing biological web services.

This chapter discusses the issues for modeling protein feature web services, the approaches for developing web services, the iGNM web service, the N-gram web service, and the category of WSP web services.

4.2. Protein Features

Understanding protein function is the key to understand health and diseases. In many cases, however, it is difficult to directly determine the function of a specific protein. In experimental context, a protein has many attributes or features (e.g., the binding sites, enzymatic activities, and ability to fold). These diverse features are associated with protein function and can be integrated to determine the function of proteins (Greenbaum, 2004).

The core of most protein research is related to some biological sequences. Whatever features proteins have, they are function of sequences. A fundamental way is to relate protein features to sequences. Each feature is a function of some operation of a position in a sequence. For instances, conservation is a function which tells how each residue in a sequence is presented across the protein family; enzyme active site is a function which tells whether a residue in a sequence is an active site or not; slow mode is a function of fluctuation of each residue in a sequence. Figure 4-1 shows an example of protein conservation feature, where the x-axis represents the residue position and the y-axis represents the percentage of occurrence across the protein family.

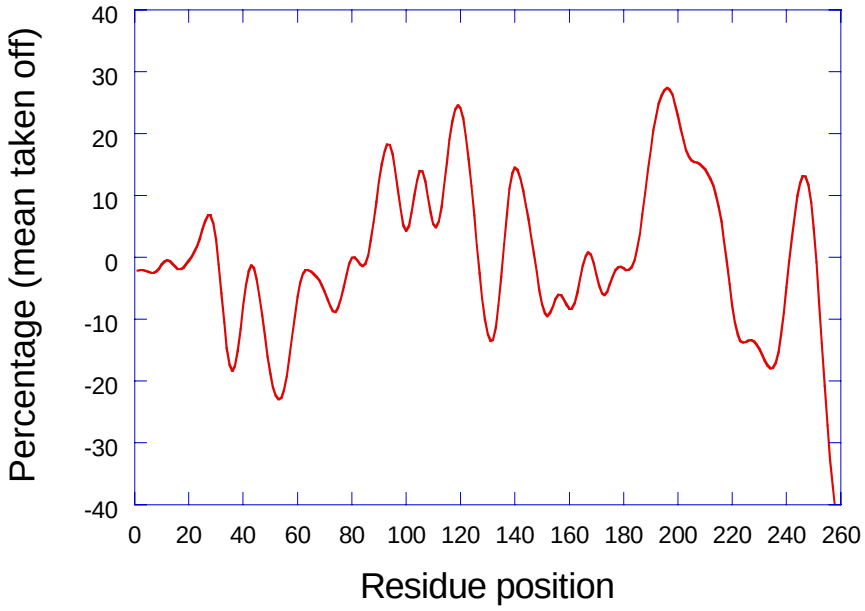


Figure 4-1. A conservation profile for carbonic anhydrase (PDB ID 1ca2)

4.3. Modeling Protein Feature Services

Many research questions can be asked for protein features, e.g., how does one feature relate to another feature, and how does one do that? These are no easy problems because features provided by different organizations have heterogeneous format and semantic quality (capabilities) associated with them. Web services are widely regarded as a way to solve this integration problem (Stein 2002; Foster 2005; Gao et al., 2005). Web services provide a higher level layer of abstraction that hides implementation details from applications so that each organization can concentrate on its own competence and still leverage the services provided by other research groups (Gao et al., 2005). By developing web services which provide protein features, it is possible to build high throughput features maps and correlations in theory.

A Web service is a set of application operations or functionalities that can be programmatically invoked over the Internet (IBM 2006). A functionality may take one or more input parameters and return one or more output parameters (Spillner et al., 2006). The definition of service functionalities and parameters are dependent on specific requirements. For information integration, it is valid to assume that the functionality of a service represents *an atomic operation* that can be assembled into many different applications (Rao and Su, 2004).

To model protein feature web services, we define a protein feature F_i as a function f_i of coordinate system (x, y, z, t), see Equation 4-1. The coordinate is the residue position (x, y, z) and/or time t. F_i can be any protein feature such as motions and conservation. Function f_i refers to some model or algorithm that has certain operations of residues to produce the feature. For example, the GNM model is a function that generates fluctuation for each residue.

$$F_i = f_i(x, y, z, t) \qquad 4\text{-}1$$

We define a service operation O_j as a function which maps a protein identity I_i into a protein feature F_i, see Equation 4-2. I_i refers to an identifier that distinguishes a protein. It can be protein name, PDB ID, protein sequence or structure. O_j represent a query functionality (operation) that takes a protein and returns the protein's feature. For example, the PDB SearchLite query interface is an operation that takes a protein's ID (PDB ID) and returns the protein's structure.

$$O_j(I_i) \rightarrow F_i \qquad 4\text{-}2$$

We define a feature web service S_k as a set of operations O_j for providing feature data, see Equation 4-3. A simple service only contains one operation (j = 1); while a complex service many contain multiple operations (j >1). For example, N-gram service can be designed to just provide one feature, e.g., ICP profile, or it can be designed to provide both ICP profile and Valdar profile (see Section 4.6).

$$S_k = \{O_1, O_2, ..., O_j\} \quad (j \geq 1) \qquad 4\text{-}3$$

Protein feature web services allow users to develop a wide range of applications. The service operations that provide protein features constitute basic building blocks out of which new applications are created. For example, enzymatic sites feature can be used in many different applications. One application can be using enzymatic feature to develop prediction models for protein function. Other applications can be correlating enzymatic feature to conservation, hydrophobicity or dynamics.

4.4. Developing Protein Feature Services

There are two approaches to design service operations or functionalities: (a) the Web application approach and (b) the bootstrapping approach. The Web application approach is based on existing functionalities provided by Web applications. For example, Gao et al. (2005) built Web services based on existing Web applications such as IBM's Genes@Work (Califano et al., 2000) and the National Center for Biotechnology Information's Entrez Databases (Wheeler et al., 2003). An

example of existing functionality in iGNM is mode shape query that can be used for developing Web services (see Section 2.3.3). The bootstrapping approach is a from-scratch design of Web service's functionality. For example, N-gram conservation service is build on top of ICP profiles without reference to existing systems (see Section 4.6).

4.5. iGNM Protein Dynamics Web Service

While iGNM web service could be built upon iGNM existing components (i.e., program for processing protein dynamics data), it is important to analyze the granularity of functionalities for web services. For example, iGNM's mobility query extracts a user selected mode from the slow mode or fast mode file and inserts it to the PDB file for visualization. The selected mode, however, is not returned to the user explicitly. A finer functionality for returning a selected mode is desired for user applications that process the mode for other purposes rather than visualization.

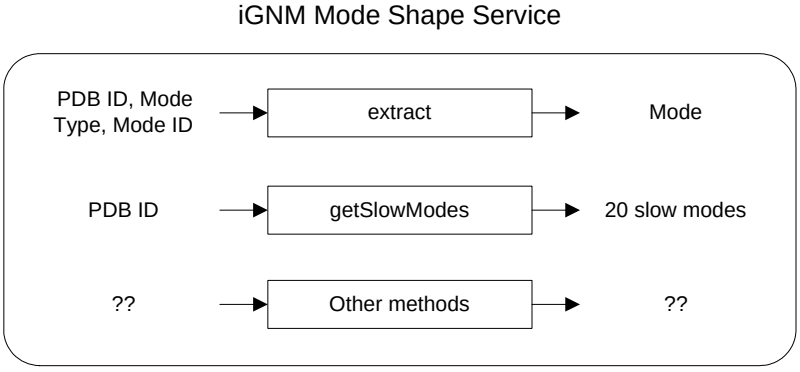


Figure 4-2. Design of the iGNM web service

Figure 4-2 shows the iGNM mode shape service with a set of operations. For example, the "extract" operation takes a PDB ID, mode type and mode ID, and returns the PDB structure's given mode. The PDB ID is a 4-digit string. The mode type is a string which takes a value of "slow" or "fast". The mode ID is an integer ranging from 1 to 20 (because iGNM provides up to 20 slow modes and 20 fast modes). The output mode is a dynamics feature which is represented as a vector of fluctuations (functional motions) associated with each residue position. Similarly, the service provides a "getSlowModes" operation which takes a PDB ID and returns all 20 slow modes. Also, the iGNM web service is extensible by adding new operations.

4.6. Protein N-gram Web Service

4.6.1. Protein N-gram Patterns

N-gram patterns (NP{*n,m*}) are sets of *n* residues and *m* gaps in windows of size *n*+*m* that start with a residue. Figure 4-3 shows an example of NP{4,2} patterns, where a pattern consists of 4 residues and 2 gaps in a window of 6. For each residue position, there are 10 patterns based on combinatorics.

Interest in these patterns was sparked by the success of an alignment-independent protein classification algorithm based on the distribution of NP{4,2} patterns (Vries et al, 2004). Features of interest in NP{4,2} patterns included: (1) the inclusion of all possible n-gram combinations for $1 \leq n \leq 4$; (2) a window wide enough to capture $n+k$ periodicities for $2 \leq k \leq 5$; (3) an implied scoring matrix due to the presence of gaps at variable positions; (4) a low

probability for finding redundant n-gram patterns in the same sequence; (5) a high probability of family membership for two sequences that contain the same pair of non-overlapping NP{4,2} patterns; and (6) the existence of all theoretically possible NP{4,2} patterns in nature.

These features, together with our recent studies on protein conservation and secondary structure prediction (Vries et al., 2006a; Vries et al., 2006b), have lead support to the utility of n-gram patterns for characterizing protein structure propensities, conservation, dynamics and other protein features. In this research, we focus on n-gram's utility for capturing protein invariant conservation profiles and how n-gram conservation profiles can be developed into a web service.

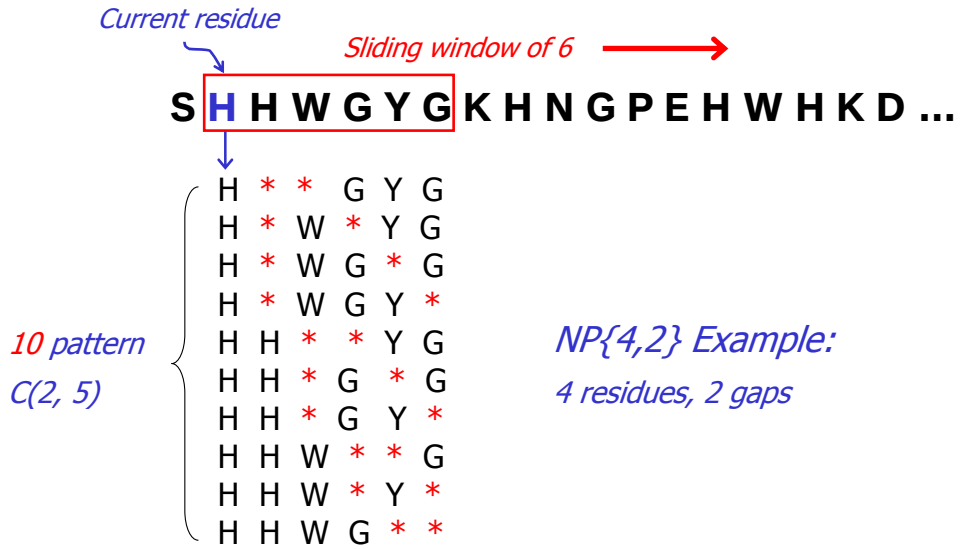


Figure 4-3. A example of NP{4,2} pattern, where residue H has 10 patterns.

4.6.2. Protein Conservation Profile

Protein homologs are amino acid sequences with a common evolutionary ancestor. Substitutions, insertions and deletions over the course of evolutionary time cause the patterns of residues and gaps in homologs to drift away from each other (Dayhoff, 1976;Henikoff and Henikoff, 1992). Conservation profiles are a measure of the shared patterns that remain. The conserved regions revealed in profiles are useful for identifying sites that are important for structure and function (Valdar and Thornton, 2001). Traditionally they have been constructed from multiple alignments (MSA) using scoring matrices and weighted averages (Valdar and Thornton, 2001). This approach has been effective, but it also requires a chain of assumptions that may not be valid in all cases. There are many ways to generate scoring matrices and these matrices vary in their sensitivity to remote homologs (Johnson and Overington, 1993). Many proteins contain multiple domains or overlapping and/or nested domains that strongly influence alignment (Raghava et al., 2003). Sequences for multiple alignments often require preprocessing to eliminate low complexity regions (Wootton and Federhen, 1996). The protein sequence samples available for multiple alignment are frequently skewed requiring the application of weighting algorithms (Karchin and Hughey, 1998). Finally, multiple alignment requires a parameterized gap penalty (Altschul et al., 1997).

A new algorithm based on n-gram patterns, called n-gram pattern local alignment (NPLA), has been developed that avoids the assumptions associated with the MSA approach. The goal of the NPLA algorithm is to generate a conservation profile that is specific to a given query sequence when the family membership of the sequence is unknown. The first step is to identify and count the non-wildcard positions in the NPLAs shared by the query sequence and a representative set of 2.1 million target PDB chains (sequences). This process is illustrated in Figure 4-4(a). A

collection sequence equal in length to the query sequence is initialized to zero for each target sequence. The non-wildcard position in the collection sequence for each common element in shared NPLAs is set to 1. The combinatorics associated with NP{4,2} patterns generates 10 different patterns for each position in the query sequence. These patterns are tested in an order that favors the longest contiguous residue runs. The algorithm stops when the first pair of NP{4,2} patterns is found. This provides an implicit substitution matrix and it insures that each position is counted only once. Summing the 1s for each collection sequence also provides a measure of similarity with respect to the query sequence for each target sequence.

The similarity threshold with respect to the target sequence is used as the basis for separating the 2.1 million target sequences into 20 samples with increasing levels of identity. This process is illustrated in Figure 4-4(b). The 95% bin for example represents all target sequences with 95% or greater similarity. The 0% bin represents all sequences in the target set. The collection sequences in each subset are then summed and normalized with respect to sample size to provide 20 raw conservation profiles. This is process is shown in Figure 4-4(c).

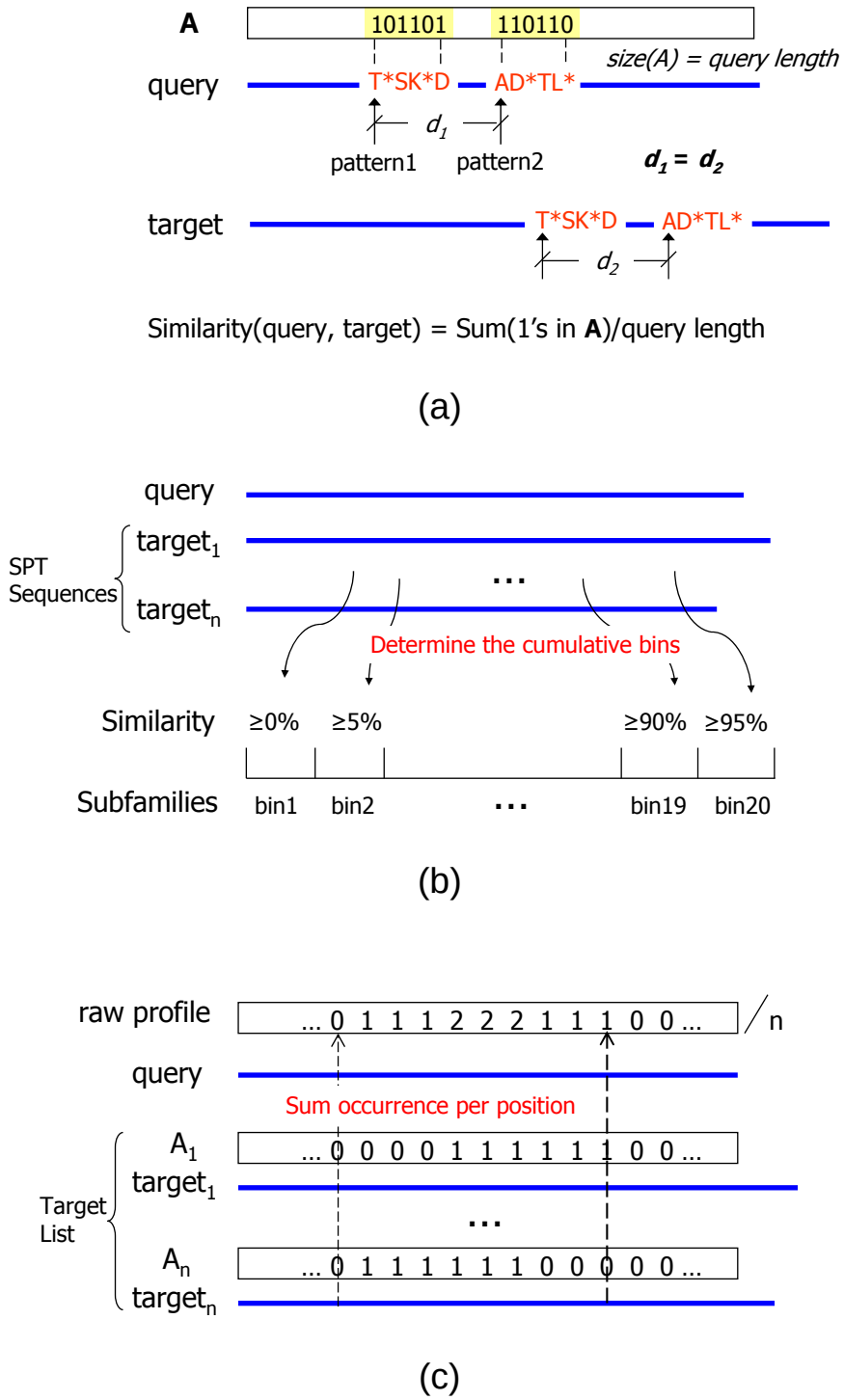
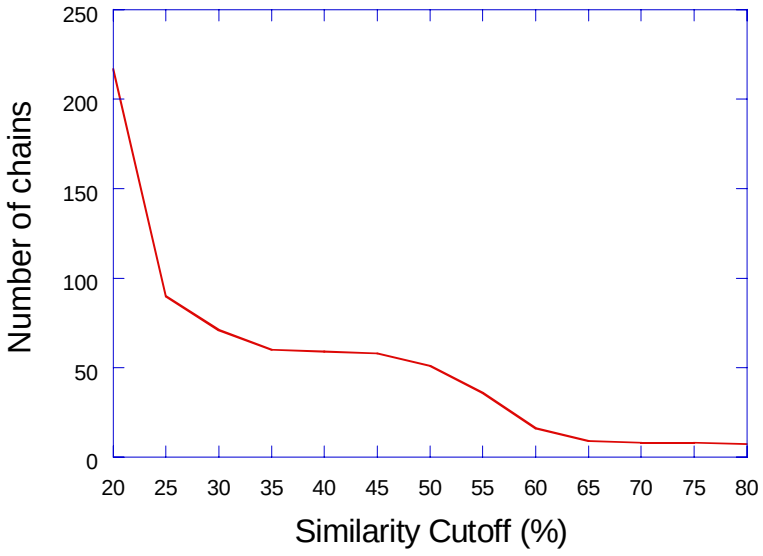


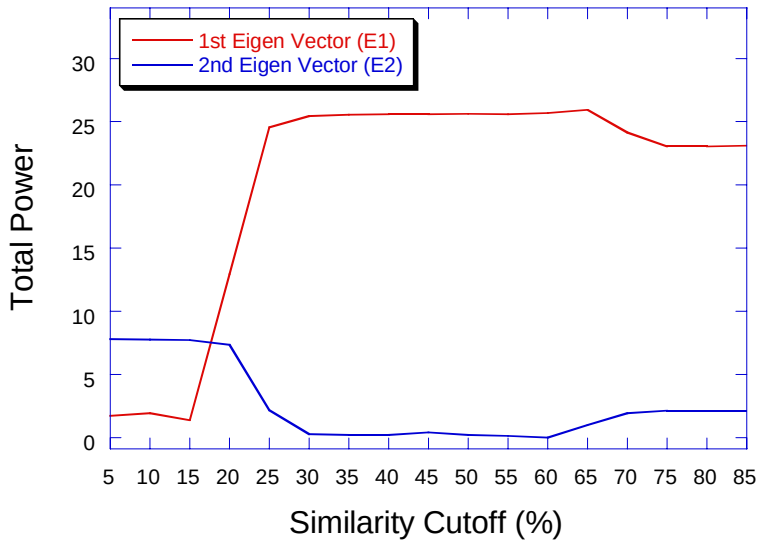
Figure 4-4. Initial steps in the NPLA algorithm: (a) Identifying and counting the non-wildcard positions in the n-gram patterns shared by the query sequence and the target sequences; (b) Dividing the target sequences into 20 bins; (3) Generating raw conservation profiles.

A covariance matrix is generated from the 20 raw profiles and subjected to singular value decomposition (SVD) (Fogolari et al, 2002). Reconstructions of the 20 raw profiles are generated for each individual eigenvector with a significant eigenvalue (> 0.01). The applicability of the algorithm is then assessed based on the sample and eigenvalue distribution and the amplitude profiles of the reconstructions (Vries et al., 2006a).

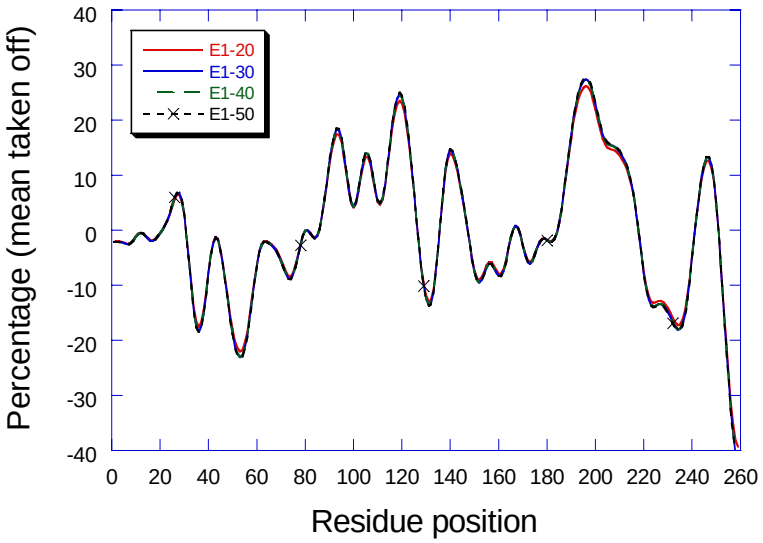
The carbonic anhydrase (P00918) from Pfam-A family PF00194 is characteristic of the sequences that meet these criteria. The sample distribution from a similarity thresholds ranging from 15% to 95% is shown in Figure 4-5(a). The sample is distributed over the range and large enough for statistical analysis. The percentage of variance associated with the first two eigenvalues is 0.75 and 0.20. A plot of the amplitude of the samples reconstructed from the first two eigenvectors is shown in Figure 4-5(b). The average amplitude of the reconstructions from the first eigenvector goes from low to high as the percentage of family members in the sample increases. The reconstructions from the second eigenvector go in the opposite direction. It can also be seen that the amplitude of the reconstruction from the first eigenvector is invariant over the central part of the the similarity range. Reconstructions of the raw profiles using the first eigenvector are shown in Figure 4-5(c) for similarity thresholds varying from 20-60%. The final plots have been subjected to 16 iterations of nearest-neighbor smoothing to eliminate high frequency noise. The conservation profiles over this range are nearly invariant. The final conservation profile (ICP) is selected from this set by identifying the profile with the least rmsd difference with its neighbors. The ICP trace representing the 40% similarity level is shown in Figure 4-5(d) with and without smoothing.


Number of chains
250
200
150
100
50
0
20 25 30 35 40 45 50 55 60 65 70 75 80
Similarity Cutoff (%)

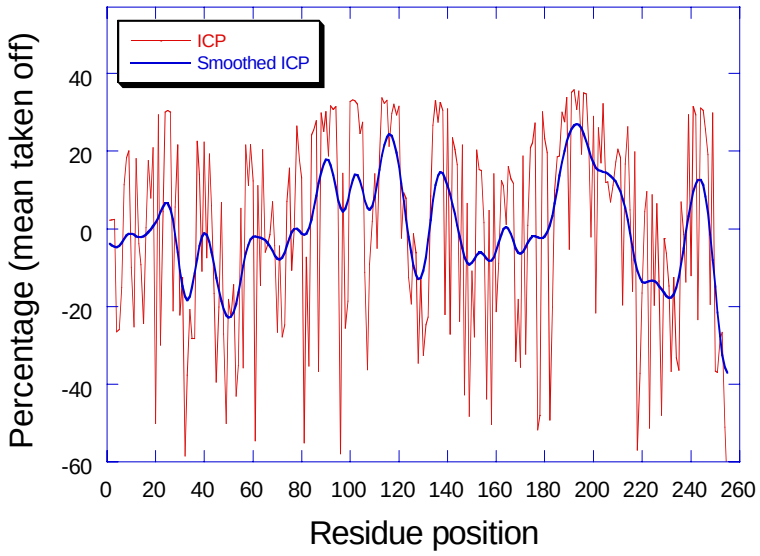
(a)


1st Eigen Vector (E1)
2nd Eigen Vector (E2)
Total Power
30
25
20
15
10
5
0
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85
Similarity Cutoff (%)

(b)



(c)



(d)

Figure 4-5. (a) Distribution of the similarity threshold samples for carbonic anhydrase (P00918) over the range from 20-80%; (b) The average amplitude of reconstructions from the first and second eigenvectors for P00918; (c) The reconstructions using the first eigenvector for similarity ranges from 20-60%. (d) Invariant conservation profile (ICP) for P00918 reconstructed from the 40% similarity level.

4.6.3. N-gram Conservation Profile Web Service

Due to the utility of n-gram patterns in characterizing protein features, several web service functionalities can be designed to provide reusable protein features such as conservation profile, slow mode profile, and hydrophobicity profile (see Figure 4-6).

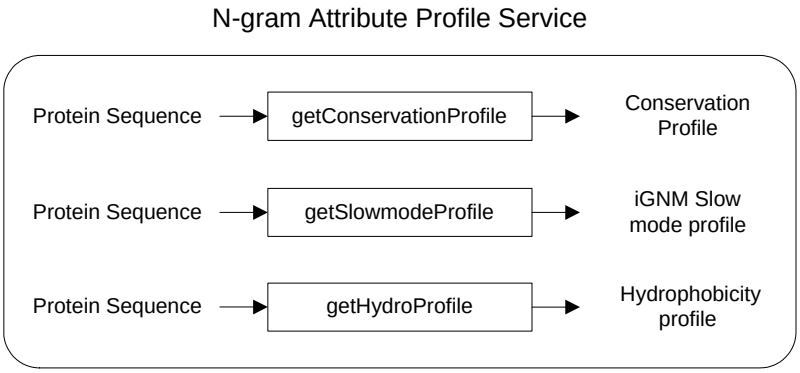


Figure 4-6. Sample n-gram service functionalities

To demonstrate how features derived by n-gram patterns can be modeled as web services using the boot strapping approach, a conservation profile web service was developed in this research. Figure 4-7 shows the n-gram conservation service with a set of operations. For example, the "extrct_ICP" operation takes a PDB ID and a chain ID, and returns the PDB chain's ICP profile. The PDB ID is a 4-digit string. The chain ID is a string which represents the chain (sequence) name of a PDB structure. The output ICP profile is a conservation feature which is represented as a vector of conservation percentage associated with each residue position. The n-gram conservation service is extensible by adding new operations. For example, an operation called

"extract_Valdar" can be added to provide MSA-based conservation profile using Valdar and Thornton's method (Valdar and Thornton).

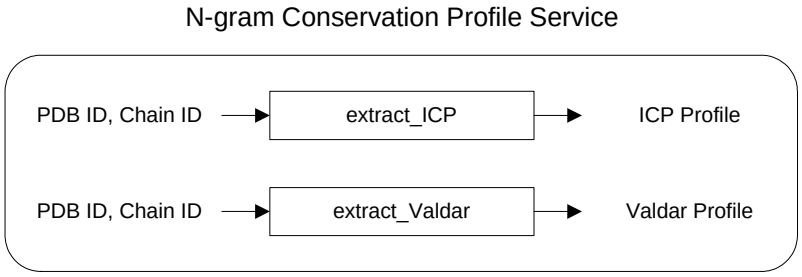


Figure 4-7. Design of the n-gram conservation web service

4.7. Category of WSP Web Services

In addition to protein feature web services, other types of services can also be introduced in WSP, including tool services and analysis services (see Figure 4-8). Feature services are the core services that provide protein feature data. For example, the iGNM Mode Service provides protein functional motion data. Tool services are used to facilitate the processing or presentation of protein feature data. For example, the PDB Replacer Service is used to present feature data in standard PDB format. Analysis services are used to integrate or analyze protein features. For example, the Feature Overlay Service is used to overlay two (or many) features in the same normalization scale and compare the values at each residue position.

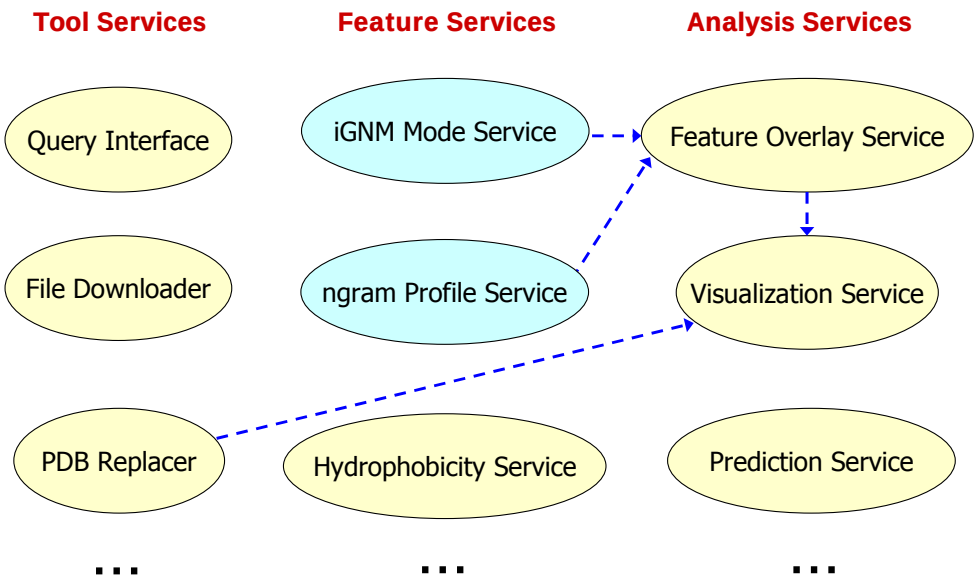


Figure 4-8. Category of WSP services

Protein feature services represent the common elements or basic building blocks that can be shared by different user applications. Once a desired service is located, users can submit queries to obtain features for specific proteins. Once the feature data is obtained, users can either use tool services and analysis services (if available) for further analysis, or develop their own standalone applications.

WSP services can also be dynamically assembled to perform *in silico* experiments. As shown in Figure 4-8, the iGNM mode service and the n-gram profile service can be linked with the feature overlay service to study the correlation between protein motions and conservation. Further, the feature overlay service and the PDB replacer service can be linked with the visualization service to visualize the motion-conservation correlation.

5. SEMANTIC DESCRIPTION OF WEB SERVICES

5.1. Introduction

There are many types of service descriptions which are used for different purposes. For example, WSDL is used as the programmatic interface to bind to the service. Also, there are quality of service (QoS) parameters or non-functional parameters (e.g., availability, reliability and reputation) that can be used as the criteria to select and chain services (Cardoso and Sheth, 2003; Zeng et al., 2004; Rao and Su, 2004).

To enable effective and user-oriented discovery of web services, it is now widely recognized that ontologies are needed for semantically describing the capabilities of web services (i.e., what a service can do) (Paolucci et al., 2002; Hull et al., 2005). To semantically describe protein web services, domain ontologies used by the biological community and service ontologies used by the web services community need to be considered.

This chapter first reviews the role of ontologies. It then describes the ontologies adopted by WSP and a semantic profile generator that generates semantic service descriptions and semantic service requests.

5.2. The Role of Ontologies

Ontology originated from philosophy as a reference to the nature and the organization of reality. In general, an ontology is a "specification of a conceptualization" (Gruber 1993). In the computer science domain, ontology provides a commonly agreed understanding of domain knowledge in a generic way for sharing across applications and organizations (Chandrasekaran et al., 1999). Typically, ontology consists of a list of terms and the relationships between those terms. The terms denote the domain concepts (or classes of objects). The relationships indicate hierarchies of concepts. They also provide property information, value restrictions and specifications of logical relationships between objects.

An ontology may take a variety of forms (e.g., object-oriented design), but it will necessarily include a vocabulary of terms and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constraint the possible interpretations of terms (Uschold and Jasper, 1999).

The database community as well as the object-oriented design community also build domain models using concepts, relations, properties, etc., but in general both communities impose less semantic constraints that those imposed in heavyweight ontologies (Gómez-Pérez et al., 2004).

Ontologies are largely used for representing domain knowledge. A common use of ontologies is data standardization and conceptualization via a machine-understandable language. Existing

ontology languages include RDF (Resource Description Framework), DAML-OIL (DARPA Agent Markup Language-Ontology Interface Language), OWL and OWL-S.

Typical ontologies include taxonomies on the web (e.g., Amazon's product catalog), domain specific standard vocabularies (e.g., Gene Ontology) and ontologies for data integration (e.g., semantic interoperation between XML schemas) (Cruz, I.F. and Xiao, H., 2005).

There are three approaches to use ontologies for data integration (Cruz, I.F. and Xiao, H., 2005; Wache et al., 2001):

- Single ontology approach. The schemas used by data resources are directly related to a shared global ontology that provides a uniform interface to the user. This approach requires that all resources have the same view on a domain.

- Multiple ontology approach. Each data resource has its own local ontology. Instead of using a common ontology, local ontologies are mapped to each other. Therefore, additional representation formalization is required for inter-ontology mapping.

- Hybrid ontology approach. This approach combines the above two approaches. First, each local resource has its own local ontology. Then, each local ontology is mapped to a global shared ontology. New data resources can be added without modifying existing mappings.

5.3. WSP Ontologies

Much progress has been made in proteomics ontologies to semantically integrate heterogeneous protein data resources, including approaches to systematic structural and functional classification and initial work towards developing standardized and unified descriptions for protein properties. For examples, CATH (Pearl et al., 2003) describes protein folds; SPINE (Bertone et al., 2001) focuses on biophysical characteristics; Gene Ontology (GO) (Gene Ontology Consortium, 2004) is a controlled vocabulary of nearly 17,500 terms for describing function of gene products; Protein Ontology (PO) (Sidhu et al., 2005) is a recent effort in building a common structured vocabulary in OWL for sharing knowledge in proteomics domain, including concepts for proteomics data and the relations among these concepts. The details of the protein ontology are reviewed in the following section.

5.3.1. Protein Ontology

The Protein Ontology or PO (Sidhu et al., 2005) is a recent effort in building a common structured vocabulary in OWL for sharing knowledge in proteomics domain, including concepts for proteomics data and the relations among them. Currently, PO contains 92 concepts implemented as OWL classes.

The Protein Ontology shows the value of hierarchy and relationships present in proteomics data. The creation of a Protein Ontology provides understanding of diverse types of data: (1) Protein Entry Details, (2) 3D Structural Representations of Proteins, (3) Structural Folds and domains

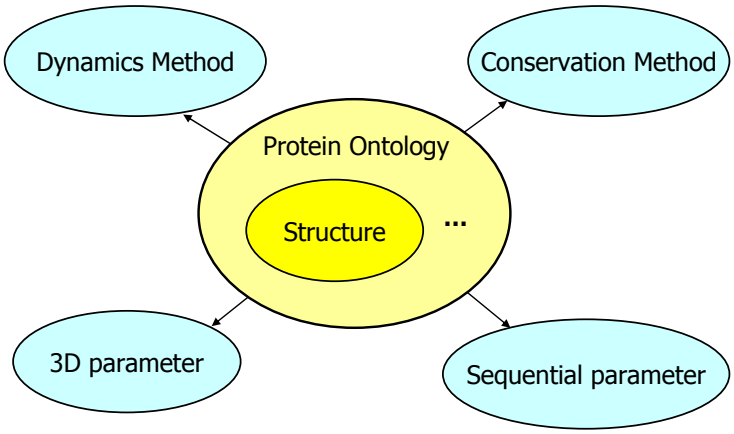
conserved in proteins, (4) Functional Domains and Families created based on Physiological and Pathological Functions of Proteins, and (5) Various Constraints like Genetic Defects and Chemical Properties of Cell that affect Final Stable Molecular Structure of Protein. Protein Ontology describes the concepts of interest in protein complex mechanisms and proteomics process (Sidhu et al., 2005).

5.3.2. Extended Protein Ontology

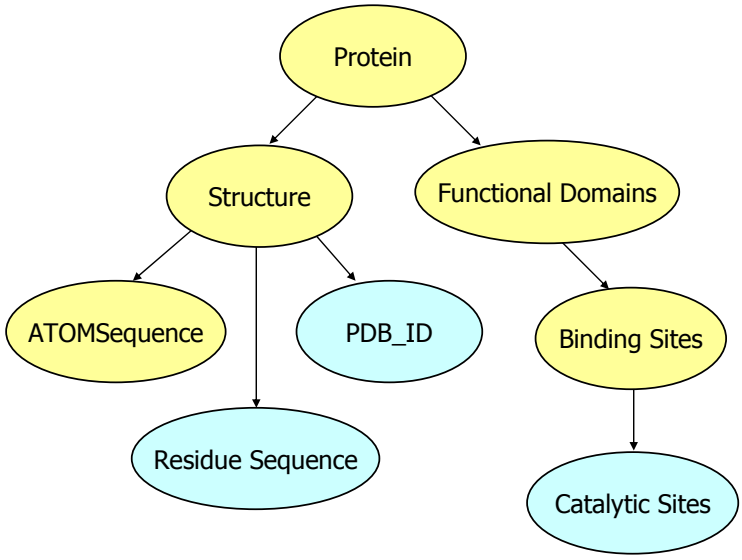
For semantic description of protein web services, existing ontologies that provide terminologies for describing protein identities and features need to be considered. For protein features that cannot be described by existing ontologies, existing ontologies need to be expanded or new ontologies need to be developed. Based on literature survey, we identified that PO already provides a set of standard terminologies that can be used to describe protein identities and features (e.g., “structure”, “sequence”, and “active binding sites”). However, there are also many protein features that cannot be described by PO, e.g., dynamics and conservation. Therefore, we try to use the established PO conceptualization as much as possible, and try to enhance PO’s expression power by adding more terminologies regarding protein features. We call the enhanced version of the protein ontology as the Extended Protein Ontology (EPO).

Figure 5-1 shows the design of EPO, where the yellow ellipses represent the existing concepts in the protein ontology and the blue ellipses represent new concepts about protein features (e.g., dynamics, n-gram pattern and conservation). The new concepts are either inserted as leave

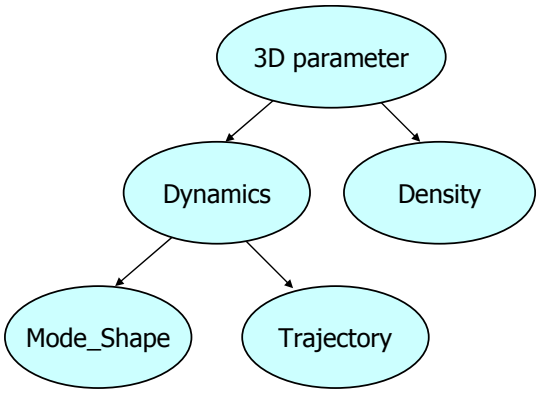
nodes of existing concepts (see Figure 5-1(a)) or organized as an independent hierarchy—a branch that is directly connected to the root node "Protein Ontology".



(a)



(b)



(c)

Figure 5-1. (a) Overall design of the extended protein ontology (EPO), where new concepts (blue ellipses) are added to the existing protein ontology (PO). (b) a fragment showing that new concepts are added as leaves of existing concepts (yellow ellipses). (c) a fragment showing an independent hierarchy.

5.3.3. Upper Service Ontology

OWL-S is an OWL-based upper ontology that describes key aspects of a service: "service profile" which states abstract description of service capabilities; "service model" which states how services interact with each other; and "service grounding" which states actual message that are exchanged among services (W3C OWL-S, 2004). The OWL-S Profile ontology is adopted to describe WSP web services. In an OWL-S profile, the service input and output are explicitly labeled with concepts in a domain ontology (e.g., EPO).

OWL-S and WSDL have a complementary relationship (see Figure 5-2). OWL-S defines message types (inputs and outputs) in terms of OWL classes (ontological concepts), which allows for a richer semantic foundations underlying the type specifications. However, OWL-S

profile is an abstract specification; it does not specify the details of message formats, protocols, and network addresses by which a web service is instantiated. Since WSDL provides a well developed means of specifying these kinds of details, it is used as the protocol to access the service (Ankolekar et al., 2004).

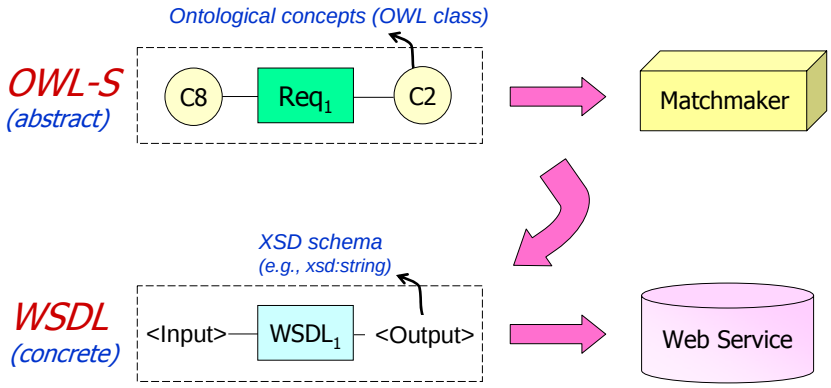


Figure 5-2. The complementary relationships between OWL-S and WSDL

5.4. Semantic Description of Protein Web Services

OWL-S profile provides the data structure for describing capabilities of services. As defined in OWL-S (see Section 2.6.2), the upper ontology for service profile includes three parts: the actor, the functional attributes, and the functional description. The actor class records information about service providers. The functional attributes include parameters such as service category, the rating assigned to the services and the geographic constraints to the service. The functional description describes the capabilities of services in terms of inputs, outputs, preconditions, and effects.

In WSP, OWL-S service profile is used for describing both service descriptions (capabilities) and service requests. The service description of a request contains 3 categories of functional parameters: input, output and constraint. The input refers to the value passed to the service operation, the output refers to the value returned by the service operation, and the constraint refers to quality requirements on the output. For example, for a service that provides protein dynamics, the input could be a protein identity (e.g., a PDB ID), the output could be the mode shape, and the constraint could be the requirement that the mode shape should provide motion data (fluctuations) at atomic level.

In theory, the input can contain multiple values/parameters, so are the output and the constraint. To semantically annotate these values, multiple concepts are needed. However, it is valid to assume a single parameter for input, output and constraint each, if the parameter is considered to be complex structure on their own (Spillner et al., 2006). Based on this assumption, the input is annotated using only one concept. The output and the constraint are also annotated using one concept respectively.

Figure 5-3 shows an example of how service capabilities can be described by incorporating OWL ontologies and OWL-S profile. By having service descriptions and requests dynamically refer to the OWL concepts defined in the same ontology (i.e., EPO), semantic matching algorithms can be designed to reason about similarities between services and requests in an unambiguous and machine interpretable form (Paolucci et al., 2002).

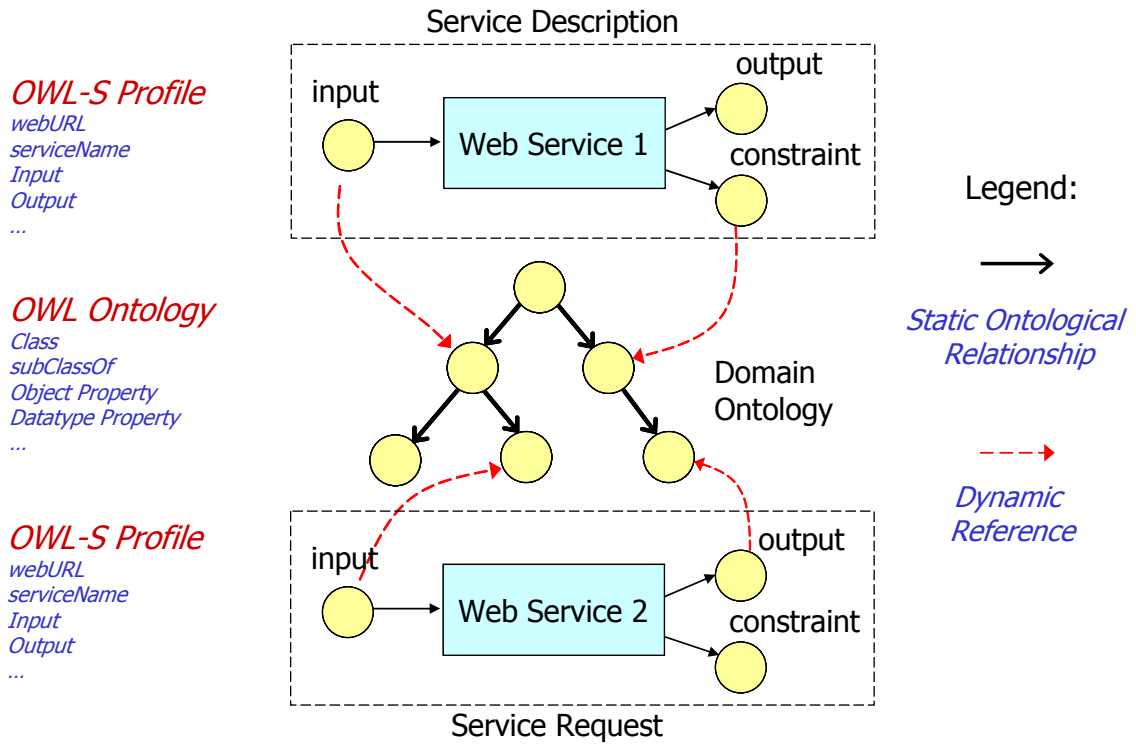


Figure 5-3. Methodology for generating semantic descriptions of web services

An OWL-S profile generator is developed to create OWL-S service profile. Given a domain ontology (i.e., EPO), the profile generator parses the ontology into a list of concepts. To generate a profile about service capabilities, the generator selects a concept as service input, a concept as service output and another concept as service constraint. For service descriptions, input, output and constraint are based on the functionalities provided by Web services. For example, "getAllSlowModes" provided by iGNM Mode Shape Service has one "PDB ID" as input and 20 "mode shape" as output, where "PDB ID" and "mode shape" are both EPO concepts. For service requests, input, output and constraint are based on user requirements. For example, a user who is looking for a service that provides protein dynamics information may specify the input as "protein" and the output as "dynamics", where "protein" and "dynamics" are both EPO concepts.

After generating the inputs and outputs for a service, the generator creates an OWL-S profile according to the OWL-S profile specification.

6. SEMANTIC PUBLICATION AND MATCHING OF WEB SERVICES

6.1. Introduction

Web services are an emerging technology for bioinformatics research. As number of biological web services increases, finding suitable services that provide a solution to the problem at hand becomes more important than ever. Therefore, service discovery is a critical component in web services integration.

Current web services standards provide a standard means of interoperating between different software components (see Section 2.6). However, they do not support operations at semantics level, leaving the promise of automatic and user-oriented discovery of Web services incomplete. For example, service providers and requestors may have distinctive perspectives and knowledge about one service resulting in differing descriptions for the service. In this case, UDDI will be unable to locate the service because it can only perform syntactic, and not semantic, matching between the service requested and the services advertised.

By having service advertisements and requests refer to the concepts defined in the same ontology, semantic matching algorithms can be designed to reason about similarities between service descriptions and requests in an unambiguous and machine-interpretable form (Paolucci et al., 2002). Thus services that provide solutions to a request can be automatically discovered even if the services registered and the service requested are syntactically distinctive.

This research contributes a semantic matchmaker service that allows service providers to publish the description of their services and allows users to submit requests and obtain semantically matched services. The matchmaker implements a service publication algorithm (see Section 6.4) and an efficient service discovery (matching) algorithm (see Section 6.5). By adopting an ontology-based service indexing strategy during the publication phase, the service matching algorithm has lower time complexity than existing algorithms.

6.2. Background

In this section, we will give an overview of data structures and matchmaking operations that are relevant to the WSP matchmaker's design and implementation.

6.2.1. Data Structures

In this research, we assume each web service has both an OWL-S semantic description and a WSDL description. These two descriptions are coupled with each other and have one-to-one relationship. While the OWL-S description is used for service discovery, the WSDL description is used for service binding.

Table 6-1 shows that a service functionality contains four basic parameters: operation, input, output, and constraint. These parameters are represented differently in OWL-S and WSDL. In OWL-S, an operation is referred as an atomic process. The input, output and constraint

parameters associated with an atomic process are annotated using ontological concepts. In WSDL, the operation name is explicitly presented. The input and output are presented as messages. There is no presentation for constraint.

Table 6-1. Comparison of OWL-S and WSDL data structure.

Service Functionality	Semantic Description (OWL-S)	Programmatic Interface (WSDL)
Operation	Atomic process	Operation name
Input	Ontological concept	Message name, data schema
Output	Ontological concept	Message name, data schema
Constraint (optional)	Ontological concept	N/A

OWL-S profiles contain more explicit semantics than WSDL descriptions and act as the main data structure for service matching/discovery. Formally, an OWL-S web service description *S* is described by a tuple:

$$S =< I, O, C > \qquad 6\text{-}1$$

where *I* is an concept (OWL class) that specifies the service's input, *O* is an concept (OWL class) that specifies the service's output, and *C* is an optional concept (OWL class) that specifies the service's output constraint.

The tuple representation is isomorphic to the more common XML serialization of OWL-S, but more explicit for processing. The discovery of service is performed at the semantic level by comparing the service's tuple and the request's tuple. More specifically, the service input concept is compared with the request input concept, the service output concept is compared with the request output concept, and the service constraint concept is compared with the request

constraint concept (if available). Therefore, to reason the similarity between two semantically annotated service descriptions, we need to know the degree of match between concepts.

6.2.2. Matchmaking Operations

Since providers and users may have differing service descriptions, the output concept of a service may not match that of a request exactly. For example, iGNM Mode Shape Service which provides "mode shape" for a "PDB ID" does not exactly match a request that looks for "dynamics". However, the iGNM service can be used for such request and should appear in the result list of the service request. Such implicit relationships between service descriptions and requests can be derived through reasoning the degree of match between two output (or input and constraint) concepts.

There are four degrees of similarity match between the output of a service (O^S) and the output of request (O^R):

- **Exact**. If O^R and O^S are the same, the match gets a score of 3 (highest similarity)
- **Plug-in**. If O^R subsumes O^S, then O^S can be plugged instead of O^R. The match gets a score of 2.
- **Subsumption**. If O^S subsumes O^R, then the service may not completely satisfy the request. The match gets a score of 1.
- **Fail**. If O^R and O^S do not have either plug-in or subsumption relations, then the match fails and gets a score of 0.

These definitions of match degrees are based on previous studies on semantic matching (e.g., Paolucci et al., 2002; Li and Horrocks, 2003) but with different scoring matrix. With these definitions of match degrees, the similarity between a service request and a service description will be realized. For example, in Figure 6-1, if a request has O^R as "Dynamics", then the match between the request and a service whose O^S is the "Dynamics" class is exact. If a service has O^S as "Mode Shape", then the match is plug-in. If a service has O^S as "3D Parameter", then the match is subsumption.

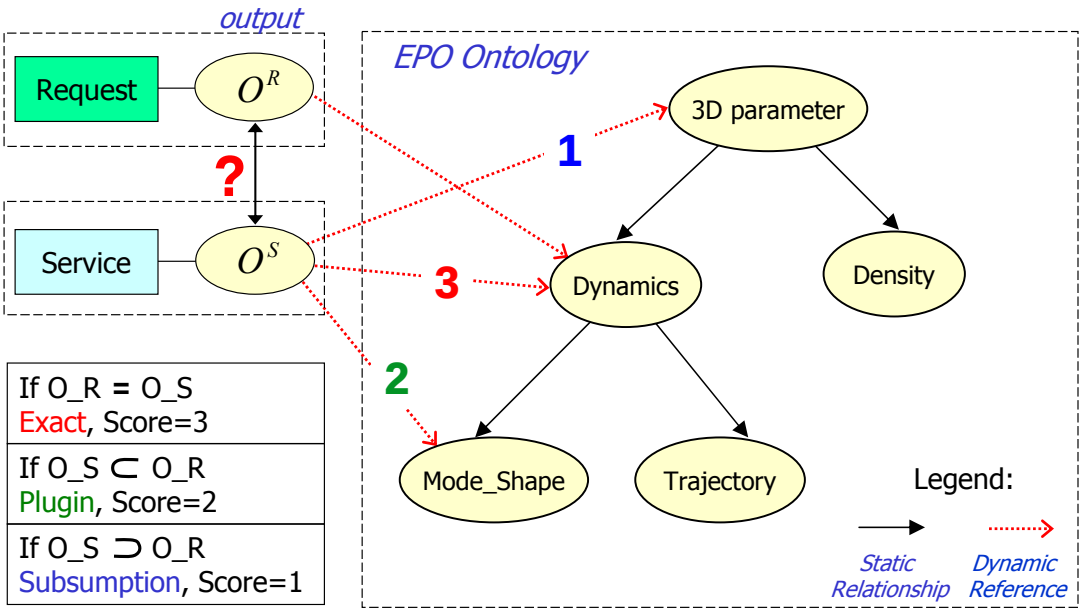


Figure 6-1. Semantic matching of service descriptions and a service request whose output concept O^R is "Dynamics". Each dashed line (red) represents a specific type of matching between the request and a service.

The matching between inputs and constraints is computed following the same procedure. Equation 6-2 generalize the comparison between a service concept C_i^S and a corresponding request concept C_i^R:

$$match(C_i^R, C_i^S) = \begin{cases} 3 & if\ C_i^R = C_i^S \\ 2 & if\ C_i^S \subset C_i^R \\ 1 & if\ C_i^S \supset C_i^R \\ 0 & else \end{cases} \qquad 6\text{-}2$$

Assuming there are *m* concepts in a service description and there are *m* corresponding concepts in a service request, the similarity or global match between the request *R* and the service *S* can be derived by summing up the match scores between the a concept pair:

$$similarity(R,S) = \sum_{i=1}^{m} match(C_i^R, C_i^S) \qquad 6\text{-}3$$

Therefore, the matching between a request and a set of services can be quantitatively measured. A service with the highest similarity score represents the most accurate service for the request. There may be more than one most accurate service. Besides the most accurate service(s), those services with a similarity greater than zero are still useful as backup services.

6.3. A Semantic Matchmaker Service

In this research, we developed a semantic matchmaker for user-oriented discovery. The matchmaker performs service publication and service discovery by interacting with a matchmaker client. Two algorithms are implemented in the matchmaker, one for publication and one for discovery. During the publication phase, the publication algorithm creates a hash index table (or registration table) of services by traversing the ontology. Then the discovery algorithm operates on the registration table without traversing the ontology (as in existing algorithms). By introducing service indexing, the time complexity associated with service discovery is reduced.

6.3.1. Architecture

Figure 6-2 shows the client-server architecture of the matchmaker service. The client has two modules, one for publication and one for discovery. The publication client is used by service providers to publish the OWL-S descriptions of their services. The discovery client is used by requestors (users) to submit OWL-S requests and obtain matched services.

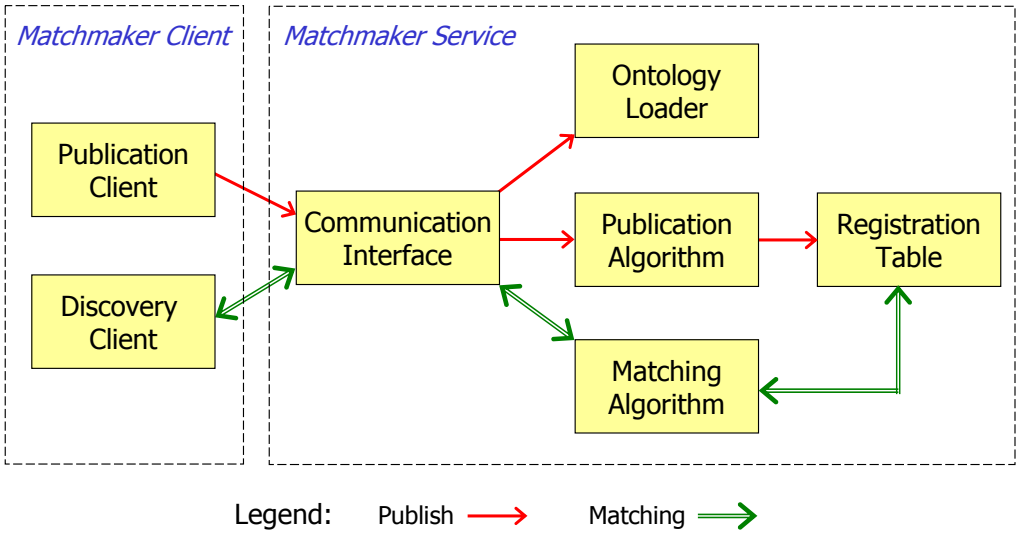


Figure 6-2. Overall Semantic Service Discovery Architecture

There matchmaker service has four components: the communication interface, the ontology loader, the publication module, the registration table and the matching module. The matchmaker receives messages from the client through the communication interface. When a message is a service description, the interface sends it to the ontology loader and the publication module for publication. The result of the publication is stored in the registration table. When a message is service request, the communication interface sends it to the matching module for service discovery. The result of the matching is set of semantically matched services, which will be sent to the discovery client.

The WSP matchmaker assumes there is only one centralized domain ontology (i.e., EPO), which is referenced by the OWL-S service description and the OWL-S service request. The ontology loader preprocesses the domain ontology and creates data structures necessary for semantic service publication and matching. The ontology loader parses the OWL file of the ontology and creates a hierarchical data structure to store all the OWL classes (concepts) based on their relationships defined in the OWL file. For each concept of the ontology, there is a list which stores the entire super and sub classes of the concept. Figure 6-3 shows an example of the hierarchical structure, where concept C2 has a super class C1 and a sub class C4. Based on the list associated with C2, we can infer that C1 subsumes C2, C2 subsumes C4, and C2 has no relation with C3.

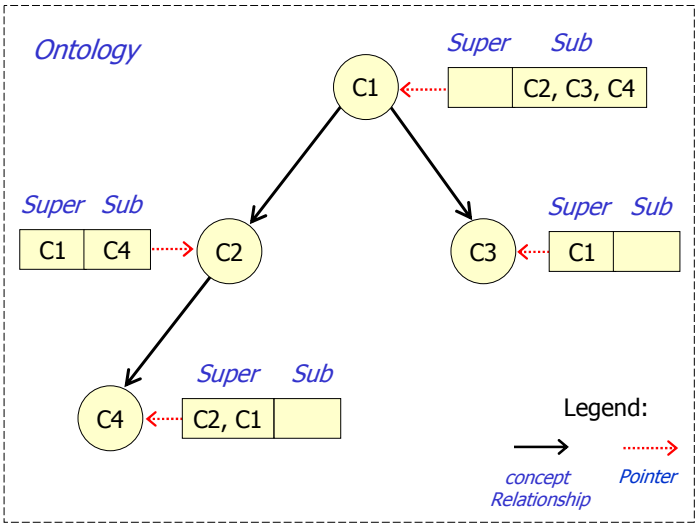


Figure 6-3. Preprocessing of the domain ontology

6.4. Semantic Publication Algorithm

The purpose of the publication algorithm is to store and index service descriptions. Each service description is assigned a unique ID and store in a local database. Also, each service description is indexed using the domain ontology. The indexing result is stored in the registration table, which is used as the main data structure for service matching.

6.4.1. Algorithm Description and Analysis

Figure 6-4 shows the pseudo code of the publication algorithm. For a concept *C* in a service description, the publication algorithm performs a graph traversal to identify *C*'s relations with each node (concept) of the graph (ontology). At a node *u*, the algorithm compares *C* with *u* and *u*'s super and sub classes. If *C* is the same as *u*, then the service gets an "exact" match score at *u*.

If C is a subclass of *u*, then the service gets a "plug" match score at *u*. If C is a super class of *u*, then the service gets a "subsumption" match score. Otherwise, the service does not have a score at *u*, which means the service cannot provide the capability represented by *u*. If the service has a score at *u*, it is indexed by *u* and the match result is inserted into a hash table called service registration table.

The index table is a survey of all services' capabilities. It associates a service's concept (capability) at relevant concepts (nodes) in the ontology. The benefit is that when we need to search for a capability (concept), we can search the index table and find all services that can provide the capabilities.

A time complexity analysis was performed for the algorithm (see Appendix A). The analysis shows the time complexity is O(m(|V| + |E|)), where |V| is the number of vertices in G, |E| is the number of edges, and m is number of concepts in service.

```
procedure register(service, G)          //service is a service description
                                        //G is an ontology; G = <V, E>
1.  registration = empty hash table
2.  parse service into concepts c[m]    //m = number of concepts
3.  for i = 1 to m do
4.     u0 = the root vertex in G
5.     DFS(u0, c[i])                    //depth-first traversal of G
6.  return registration

procedure DFS(u, c)                     //u, c are concepts
1.  degreeOfMatch(u, c)
2.  status[u] = "processed"
3.  for each neighbor v of u do
4.     if status[v] != "processed" then
5.        DFS(v, c)
6.  return

procedure degreeOfMatch(u, c)           //measure match between u and c
1.  if c = u then
2.     service.score = "exact" or 3
```

```
3.  if c is subclass of u then
4.      service.score = “plugin” or 2
5.  if c is superclass of u then
6.      service.score = “subsumption” or 1
7.  if service.score != null then
8.      registration.add(u, service)
9.  return
```

Figure 6-4. Pseudo code for the WSP service publication algorithm

6.4.2. A Service Registration Example

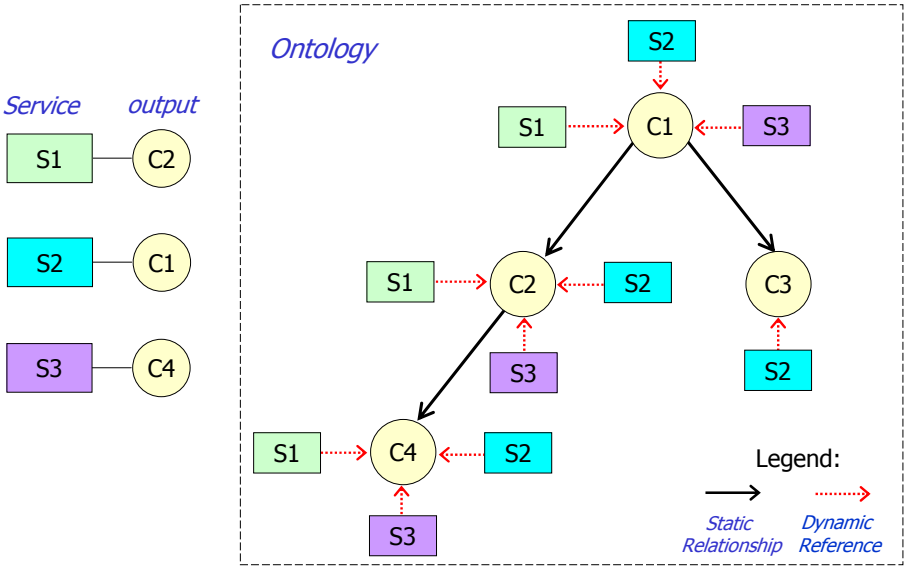


Figure 6-5. An example of service registration, where a service is pointed to the conceptual nodes which have relations with the concept in the service.

Figure 6-5 shows an example of registering three semantic service descriptions S1, S2 and S3 with the domain ontology, where S1 has an output concept C2, S2 has an output concept C1, and S3 has an output concept C4.

For S1, the algorithm traverses the ontology graph and compares C2 with each node. At node C1, the service gets a match score of 2 because C2 has a "plugin" match with C1; at node C2, the service gets a match score of 3 because C2 has an "exact" match with C2; at node C4, the service gets a match score of 1 because C2 has an "subsumption" match with C4; at node C3, the service gets a match score of 0 because C2 has no match with C3. Similarly, S2 gets a score of 3 at node C1; a score of 1 at node C2; a score of 1 at node 4; and a score of 1 at node C3. S3 gets a score of 2 at node C1; a score of 2 at node C2; a score of 3 at node C4; and a score of 0 at node C3. The registration result is shown in Figure 6-6.

Concept	Service	Mapping Score
C1	S1	Score (c1, S1) = 2
	S2	Score (c1, S2) = 3
	S3	Score (c1, S3) = 2
C2	S1	Score (c2, S1) = 3
	S2	Score (c2, S2) = 1
	S3	Score (c2, S3) = 2
C4	S1	Score (c4, S1) = 1
	S2	Score (c4, S2) = 1
	S3	Score (c4, S3) = 3

Figure 6-6. Service registration result is a service registration table (or an index table).

6.5. Semantic Service Matching Algorithm

The matching algorithm compares the concepts (requirements) in an OWL-S request with the concepts (capabilities) in an OWL-S service description. The global match between the request and the service description is a summation of the degree of match between concepts, see Equation 6-3.

6.5.1. Algorithm Description and Analysis

Figure 6-7 shows the pseudo code of the matching algorithm. For each concept Ci in a service request, the algorithm looks up the registration table for a record associated with it. This record contains a list of services that could provide the capability represented by Ci. This list is called a candidate list for Ci. After all candidate lists are retrieved, the algorithm tries to find the intersection of those lists. Services that appear in all the candidate lists represent the final matched services—services that can provide all the concepts (capabilities) in the request. Finally, the matched services are sorted based on their global match scores.

A time complexity analysis was performed for the algorithm (see Appendix A). The best case time complexity is O((m)(N)), the worst case time complexity is $O((m)(N) + n^2)$, where m is the number of concepts in request, N is the total number of registered services, and n is the number of matched services.

```
procedure match (request, registration)          //request is a service description
                                                 //registration is an index table
1.  parse request  into concepts c[m]
2.  var service[], candidateList[m][]
3.  for i = 1 to m do                            //m = number of concepts
4.     candidateList[i] = services indexed by c[i]   //hash table search
5.  service = candidateList[1]
6.  for i = 2 to m do
7.     for j = 1 to N do                         //N = all registered services
8.        if service[j] != candidateList[i][j] then
9.              remove service[j]
10.  for k = 1 to length(services) do
11.     for i = 1 to m do                        //n = matched services (n < N)
12.        service[k].match += service[k].score(c[i])
13. insertion_sort(service)                      //calculate total match score
14. return service                               //sort matched services
```

```
procedure insertion_sort(A)
1.   for j = 2 to length(A) do                    //insert A[j] into the sorted list
2.      key = A[j]                                A[1...j-1]
3.      i = j – 1
4.      while i > 0 and A[i] > key do
5.         A[i+1] = A[i]
6.         i = i – 1
7.      A[i+1] = key
8.   return
```

Figure 6-7. Pseudo code for the WSP service matching algorithm.

6.5.2. A Service Matching Example

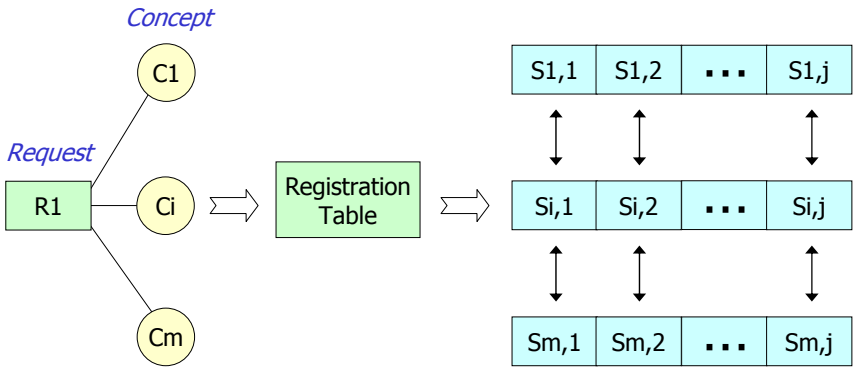


Figure 6-8. Protocol for service matching. Each concept in a service request is matched against a record in the registration table. The final matched services will be the intersections of all candidate lists.

Figure 6-8 shows an example of the service matching process, where the service request R1 has m required concepts {C1, …, Cm}. For C1, there is a list of matched services L1 = {S1,1, …, S1,j}. Similarly, Cm has a list of matched services Lm = {Sm,1, …, Sm,j}. The m lists are joined to find the common services—the final results.

6.5.3. Comparison with Typical Matching Algorithm

Existing service matching algorithms usually perform a pair-wise comparison between a service request and all the registered services (e.g., Paolucci et al., 2002; Li and Horrocks, 2003). A match between a request and a service consists of the match of the request concept C_i^R and the service concept C_i^S. In order to find the similarity between a request concept and a service concept, a matching algorithm needs to search the ontology graph to find the locations of the two concepts and then determines if they have any ontological relationship using the scoring matrix defined in Equation 6-2 (Section 6.2.2).

Assuming the ontology is preprocessed such that each concept has a record of its subclasses and superclasses (see Section 6.3.1), then either C_i^R or C_i^S needs to be searched but not both. For example, to determine the match between a request output O^R and a service output O^S, an algorithm can search the ontology graph to find the location of O^R and determine and degree of match between the two concepts by checking if O^S appears in O^R's list of subclasses and superclasses. Similarly, the match between a service input and a request input and the match between a service constraint and a request constraint all require traversal of the ontology graph.

```
procedure match(request, All, G)          //G is an ontology, G = <V, E>
                                          //All is a list of all registered services
1.  var service[ ]
2.  for i = 1 to length(All) do           //N = number of all services
3.     match = serviceMatch(request, All[i])
4.     if match != null then
5.        add All[i] to service
6.  sort(service)
7.  return service
```

```
procedure serviceMatch(request, service)          //compare a request with a service
1.  parse request into concepts c1[m]
2.  parse service into concepts c2[m]
3.  for i = 1 to m do
4.     u0 = the root vertex in G
5.     score[i] = DFS'(u0, c1[i], c2[i])
6.     service.match += score[i]                  //depth-first search of service concept
7.  return service.match                          //calculate match score

procedure DFS'(u, x, y)                           //x is request concept
                                                  //y is corresponding service concept
1.  if u = y then
2.     score = degreeOfMatch(y, x)
3.     return score
4.  else
5.     status[u] = "processed"
6.     for each neighbor v of u do
7.        if status[v] != "processed" then
8.           DFS'(v, x, y)

procedure degreeOfMatch(u, c)
1.  if c = u then
2.     score = "exact" or 3
3.  if c is subclass of u then
4.     score = "plugin" or 2
5.  if c is superclass of u then
6.     score = "subsumption" or 1
7.  return score
```

Figure 6-9. Pseudo code for a typical service matching algorithm

Figure 6-9 shows the pseudo code of a typical service matching algorithm. In the main control procedure "match" of the matching algorithm, a service request is matched against all the registered services. Whenever a match between the request and any of the services is found, it is recorded and scored to find the services with the highest similarity.

A match between a request and a service consists of the match of all the request concepts and the service concepts (see procedure "serviceMatch"). A match is recognized if and only if for each request concept, there is a matching service concept. To determine if there is a match, the algorithm first searches for the service concept in the ontology and then calls the scoring matrix

(see procedure "degreeOfMatch") to calculate the degree of match (or match score). The match scores for all concepts are summed up as the global match score (or similarity score) between the request and the service.

The last piece of the algorithm is to sort the resulting matches. The sorting is based on the similarities scores of all matched services. Traditional sorting algorithms (e.g., insertion sort) can be applied here.

A time complexity analysis was performed for the algorithm (see Appendix A). The best case time complexity is O((N)(m)(|V|+|E|)), and the worst case time complexity is O((N)(m)(|V|+|E|) + n^2), where N is the total of registered services, m is the number of concepts in service request, |V| is the number of vertices in G, |E| is the number of edges in G, and n is the number of matched services.

Table 6-2. Comparing WSP matching algorithm with typical matching algorithm

	Typical	**WSP**
Service Publication	No service indexing	O(m(|V|+|E|))
Service Matching (best case)	O((N)(m)(|V|+|E|))	O((m)(N))
Service Matching (worst case)	O((N)(m)(|V|+|E|) + n^2)	O((m)(N) + n^2)

Table 6-2 compares the time complexity for typical service matching and WSP service matching. The typical matching algorithm does not have service indexing strategy. The matching between a service request and all the registered services is dependent on the size of the ontology and the number the registered services. When the number of the registered services increases or (and) the size of the ontology increases, the matching time also increases significantly. By introducing

service indexing, the time complexity associated with WSP matching is reduced. It is not dependent on the size of the ontology.

6.6. WSP Scenario: Discovery of Protein Dynamics Data Resource

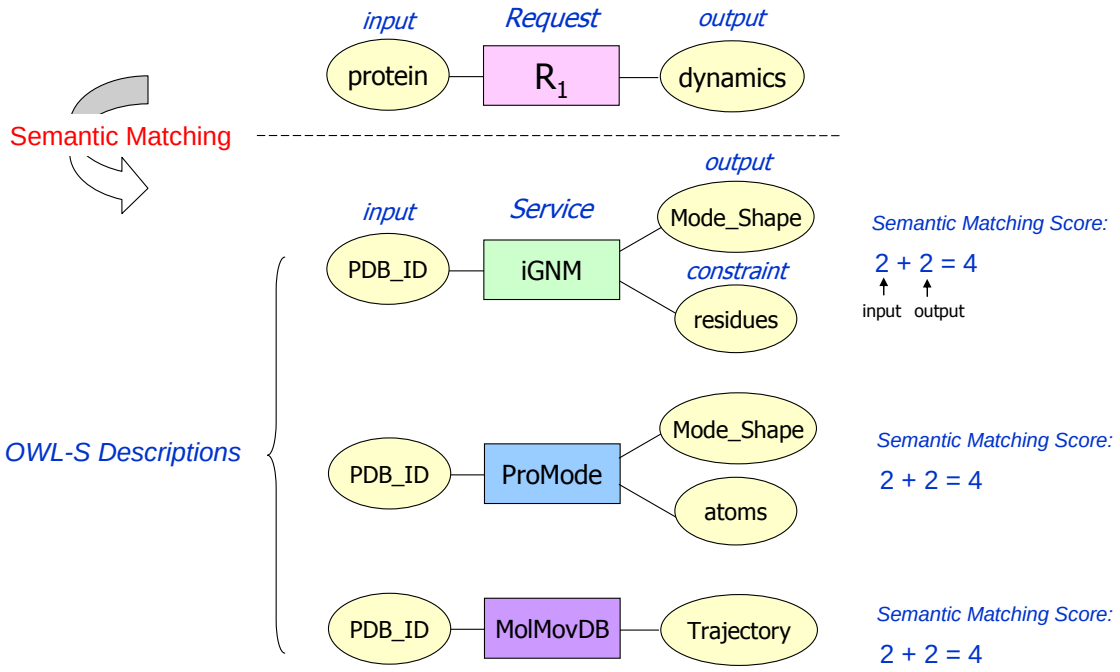
With the rapid accumulation of protein structures in PDB, it now widely recognized that efficient methods and tools are needed for understanding the protein dynamics, and thereby controlling the function of target proteins (Yang et al., 2005). Indeed, researchers have built a wide range of Web servers and databases on protein structural dynamics, including MolMovDB (Echols et al., 2003), DynDom (Lee et al., 2003), ElNemo (Suhre and Sanejouand, 2004), ProMode (Wako et al., 2004), MoViES (Cao et al., 2004), Dynamite (Barrett and Noble, 2005), WEBnm (Hollup et al., 2005), and iGNM (Yang et al., 2005), see Section 2.4.2.

Although these data resources all provide protein dynamics data, they have different capabilities and semantic meanings associated with them. This is because there are many types of dynamics data derived from different methods. For example, the data provided by iGNM is the mode shapes predicted by the GNM model at the residue level; the data provided by ProMode is the mode shapes predicted by NMA at the atom level; the data provided by MolMovDB is the trajectories derived by interpolations between two conformations. Despite their heterogeneity, OWL-S semantic descriptions can be generated to precisely capture the capabilities of these data resources (see Section 8.3.2).

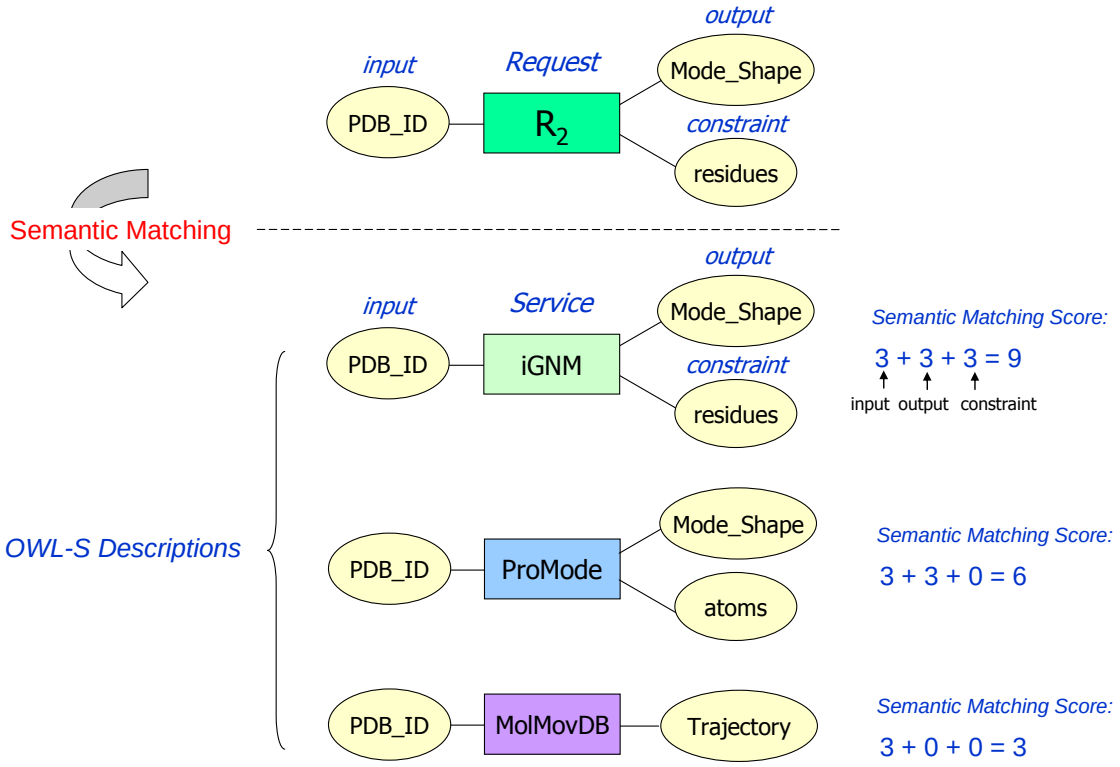
By having service descriptions and service requests refer to the EPO ontology, the semantic matching algorithm can quantitatively measure the similarity between a service and a request. Therefore, for a given request, the WSP matchmaker can perform user-oriented discovery by retrieving services that semantically meet the request.

Figure 6-10 shows two service discovery examples. In the first example, the input of iGNM gets a score "2" (because "PDB ID" is a subclass of "protein"), and the output of iGNM gets a score "2" (because "mode shape" is a subclass of "dynamics"). So iGNM gets a global match score of "4". Similarly, ProMode and MolMovDB also gets a score of "4". In this case, all three services have equal capabilities for the request R1, which looks for general (any type) protein dynamics data.

In the second example, the input of iGNM gets a score "3", the output of iGNM gets a score "3" and the constraint gets a score "3". So iGNM get a global match score of "9". The input of ProMode gets a score "3", the output of ProMode gets a score "3", and the constraint gets a score "0" (because "atoms" is different from residues"). So ProMode gets a global match score of "6". The input of MolMovDB gets a score "3", but the output and the constraint both get a score "0" (because "trajectory" is different from "mode shape" and there is no constraint concept in the description). So MolMovDB gets a global match score of "3". In this case, iGNM is the most accurate service because it provides the same capability that is being looked for—providing mode shape data for PDB structures at the residue level.



(a)



(b)

Figure 6-10. WSP service matching examples

7. CHAINING OF PROTEIN WEB SERVICES

7.1. Introduction

Protein feature web services and associated analysis web services (see Chapter 4) allow users to develop a wide range of applications. The service operations that provide protein features constitute basic building blocks out of which new applications are created. For example, enzymatic sites feature can be used in many different applications. One application can be using enzymatic feature to develop prediction models for protein function. Other applications can be correlating enzymatic feature to conservation, hydrophobicity or dynamics.

This chapter presents a workflow-based approach for integrating protein feature web services. In this approach, a WSP application (i.e., an integration task) is represented as an abstract workflow of service requests, where each request is expected to be implemented by a web service. An integration agent is used to select and chain services, based on the criteria of service accuracy and data interoperability. The agent finally generates a concrete workflow of web services, which automatically integrates the results from individual services.

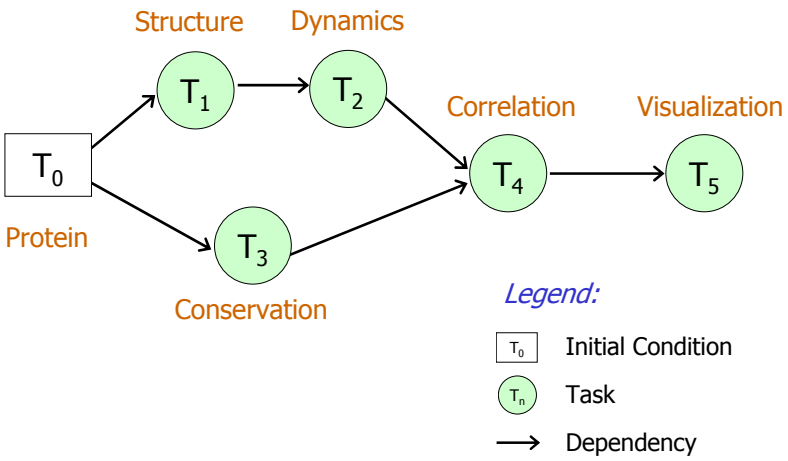
7.2. Workflow-Based Service Integration

WSP allows users to integrate protein features through protein web services. As discussed in Section 2.6.3 and Section 3.3, one way of performing service integration is to pipe together inputs and outputs of consecutive web services in a workflow environment.

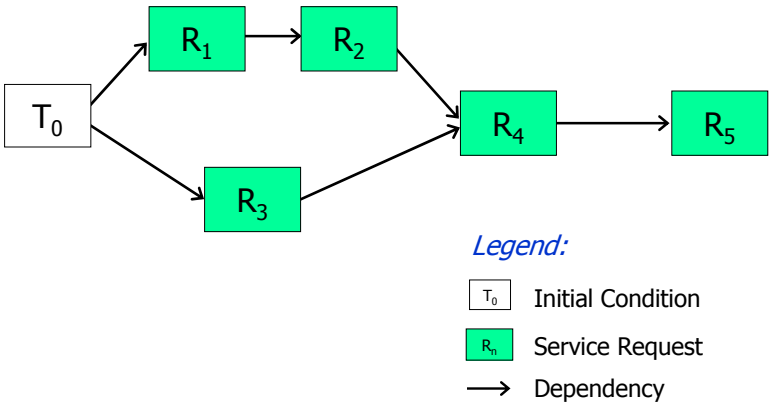
In WSP, there are three types of workflows, namely *application or higher level workflow*, *abstract workflow* and concrete *web services workflow*. The higher level workflow is used to represent a user application (i.e., integration task). It is a directed acyclic graph (DAG) of tasks, where each task is to obtain a protein feature or correlate two protein features. Figure 7-1(a) shows a higher level workflow for protein feature integration/correlation, where the initial condition (input) T_0 is a protein, the first task T_1 is to obtain a structure of the protein, the second task T_2 is to obtain dynamics for the structure, the third task T_3 is to obtain the conservation profile of the protein, the four task T_4 is to correlate the dynamics feature with the conservation feature, and the fifth task T_5 is to visualize the correlation map.

Given a higher-level workflow, the OWL-S profile generator (see Section 5.4) is used to generate a semantic service request for each task. The resulting workflow is called an abstract workflow which consists of service requests. The abstract workflow has the same topology (data dependency) as the higher-level workflow. Figure 7-1(b) shows an example of abstract workflow. The first request R_1 is to find a service that provide protein structure, it is formulated according to the first task T_1 in the high-level workflow by the user. Similarly, the second request R_2 is to find a service that provide protein dynamics, and so on.

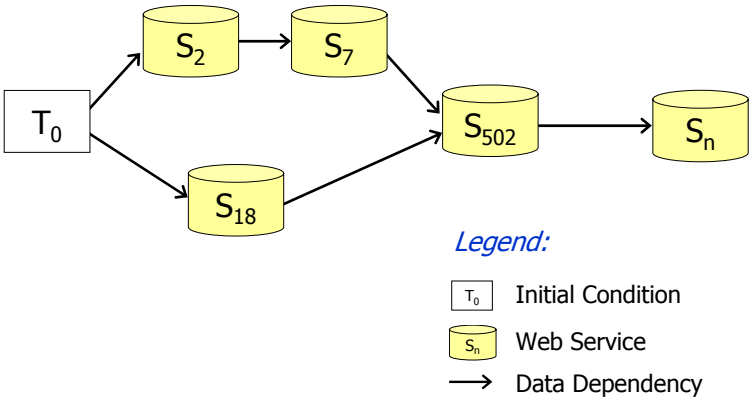
Given an abstract workflow, WSP generates a web services workflow which consists of web services. The web services workflow has the same topology as the higher-level workflow and the abstract workflow. Figure 7-1(c) shows a web services workflow, where each node is a service that implements a task. For example, service S_2 is used to implement T_1, S_7 is used to implement T_2, and so on.



(a)



(b)



(c)

Figure 7-1. (a) An example of higher-level workflow; (b) an example of abstract workflow; (c) an example of concrete web services workflow.

As discussed in Section 3.3, WSP adopts a static-workflow based integration approach (see Figure 3-1), where the higher-level workflow is provided by the user. Based on a higher-level workflow, users can generate semantic service requests using the OWL-S profile generator (see Section 5.4). Once an abstract workflow is generated, the integration agent is responsible for generating the web services workflow. The details of service integration are presented in the following section.

7.3. WSP Service Integration Process

Figure 7-2 shows the WSP service integration process. Given a higher-level workflow, the OWL-S profile generator (see Section 5.4) is used to generate a semantic service request for each task. The result is an abstract workflow. The semantic matchmaker (see Chapter 6) is then used to discover services for all requests. The result is a list of candidate services for each request.

After that an integration agent is used to select and chain services. The resulting workflow is a web services workflow which consists of web services.

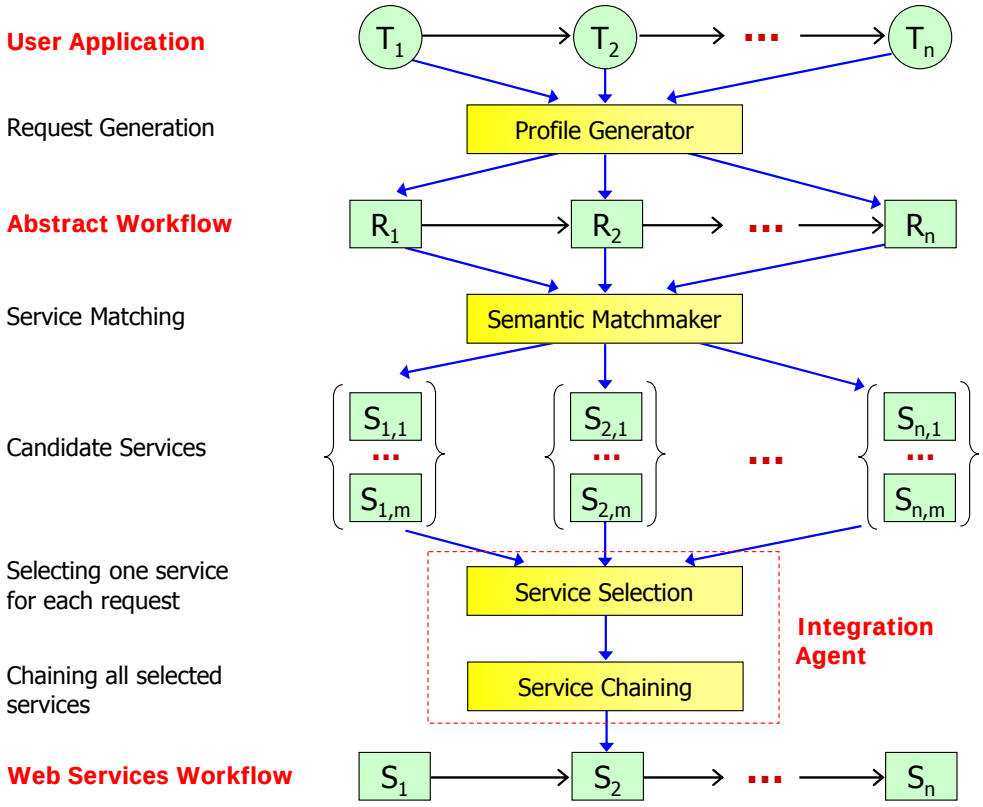


Figure 7-2. WSP integration process.

The integration agent contains two modules, the first is service selection module and the second is service chaining module. Service selection is to select a service for each request using the WSP selection criteria: service accuracy and data interoperability (see Section 7.4 for details). Service chaining is to generate SOAP requests to all selected services and chain the service operations (functionalities) according to the data dependencies (links) defined in the abstract workflow. A service chaining algorithm is developed based on the WSP service selection criteria (see Section 7.5).

It should be noted that the mapping from an abstract workflow to a concrete web services workflow may require a number of user interactions (e.g., providing additional input parameters or modifying the abstract workflow). This is because adjacent services may have dissimilar service parameters (i.e., input and output data schema) that prevent a web services workflow from being fully automatic. The problem of user involvement in service chaining is not unique to WSP, scientists using workflows of web services have experienced similar problems (Hull et al., 2005; Kim et al., 2004; Cardoso and Sheth, 2003; Sirin et al., 2003), so it seems that this problem is general rather than specific to WSP.

7.4. Service Selection Criteria

7.4.1. Literature Review

Given a workflow of service requests (abstract work), the integration problem is to discover, select and chain services to generate an invokeable workflow of web services. Service selection is a critical step wherein a specific service instance is chosen based on user requirements. These requirements represent a wide range of quality expectations (e.g., accuracy, response time and reputation) for the services.

Several service composition systems have been proposed to select services based on nonfunctional attributes such as quality of services (QoS) and trust. For example, Maximilien and Singh (2004) proposed a framework for service selection based on a QoS ontology. In their

framework, both users and providers can specify QoS policies using notations in the QoS ontology. A matching algorithm is used to match user policies to provider service policies and thereby select services according to user QoS requirements. Using a QoS model, Zeng et al. (2004) proposed two service selection approaches, namely local optimization and global planning. In the local optimization approach, the system selects the web service which has the maximum QoS score for a given task. In the global planning approach, all possible execution paths are generated and the one which maximizes the user's QoS requirements is selected.

In addition to QoS-based service selection, there are service selection methods based on the interoperability at the WSDL interface level. For example, Spillner et al. (2006) proposed an approach to select services based on data interoperability between consecutive services. This approach checks for the compatibility of the output message of one service operation with the input message of the other operation. For each check, there are three possible outcomes: (1) all-or-nothing compatibility: the output message and the input message are either fully compatible (i.e., identical) or not at all; this can be easily checked for simple data types and recursively checked for complex data types; (2) subset compatibility: the input of the second operation is a subset of the output of the first one, in this case some output values of the first operation will have to discarded; or the output of the first operation is a subset of the input of the second operation, in this case some additional values will have to be provided in order to invoke the second operation; (3) arbitrary compatibility: there is a certain percentage of compatibility between the output message and the input message, e.g., two complex XML schema may have some compatible or overlapping parts. Similar to Spillner's approach, Cardoso and Sheth (2003) proposed a method that checks the structural properties of service inputs and outputs. When this

is no perfect match between two services, the system will prompt the user to manually establish the connections among web service interfaces, e.g., manually connect the outputs of one service with inputs of the next service.

7.4.2. WSP Service Selection Criteria

WSP adopts two criteria to select services: *service accuracy* and *data interoperability*. Service accuracy means that the most accurate service is selected for each request (task). The accuracy is measured by the semantic similarity between the request and the service. For each request, the WSP matchmaker returns a list of semantically matched services sorted based on their similarity score to the request (see Section 6.5). Therefore, services with the highest score are selected. These services represent the services that can best meet the data requirements for each task.

The second criterion is to consider data interoperability between adjacent services. More specifically, the output schema of one service is compared with the input schema with the next services. Figure 7-3 shows an example of selecting services based on data interoperability, where the first request R_1 is to find protein members for a given protein family, and the second request R_2 is to find dynamics data for protein members. Through semantic matching, the system discovers a list of candidate services for each request. For R_1, there is only one service called "Pfam" that semantically matches R_1. Therefore, "Pfam" is selected. For R_2, there are two services "oGNM" and "iGNM" that semantically match R_2. As can be seen, "oGNM" and "iGNM" have the same matching score and thereby the same capabilities to provide dynamics data (i.e., slow mode). Since "oGNM" and "iGNM" have the same capabilities, the system will

consider their interoperability with "Pfam". By checking the WSDL interfaces of "oGNM" and "iGNM", the system selects "iGNM" because its input data schema (XSD:String) is the same as the output schema of "Pfam".

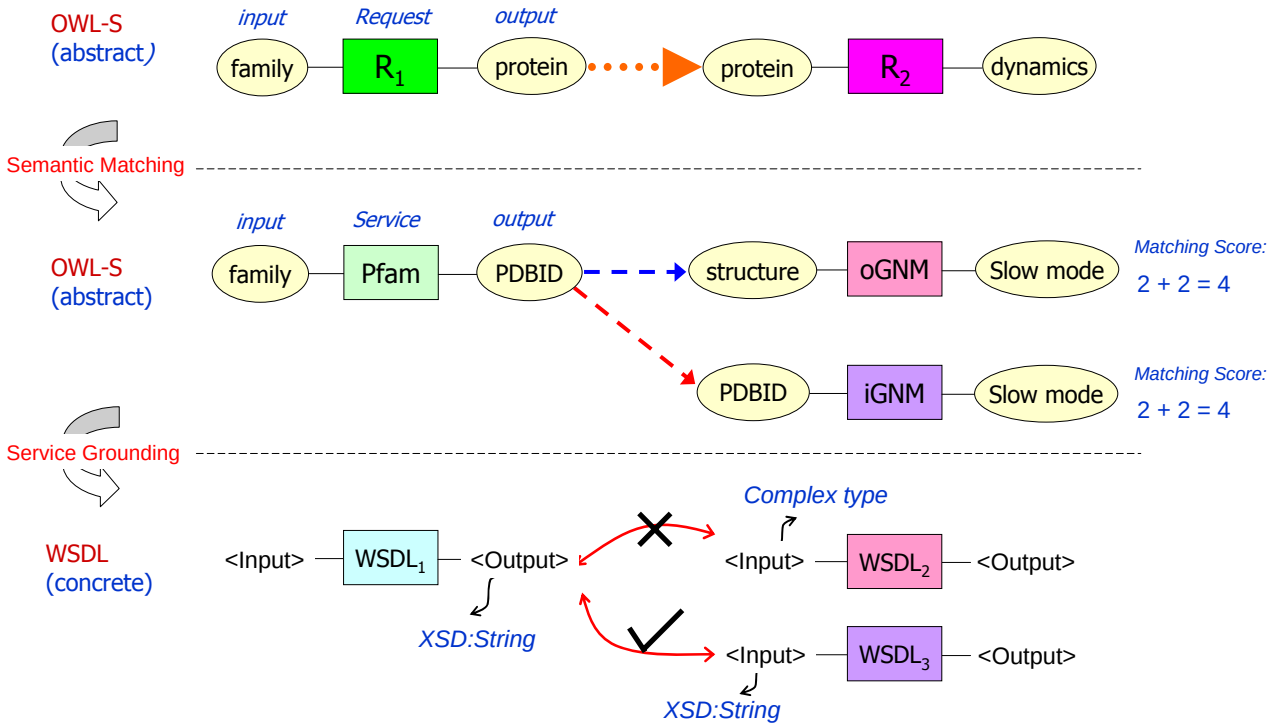


Figure 7-3. An example showing the process of service selection

7.5. WSP Service Chaining Algorithm

Service chaining is a complex problem. Assuming there are N requests (tasks) and each request has M candidate services, then there are M^N possible solution paths (i.e., chain of services), see

Figure 7-4. As discussed in Section 7.4.1, many algorithms have been proposed to select and chain services. These algorithms generally fall into two categories: local optimization algorithms and global optimization algorithms. The main feature for a local optimization algorithm is to select a service for each task without considering the relationships between selected services. The main feature of a global optimization algorithm is to select a solution path that maximizes the optimization parameter (e.g., QoS or interoperability).

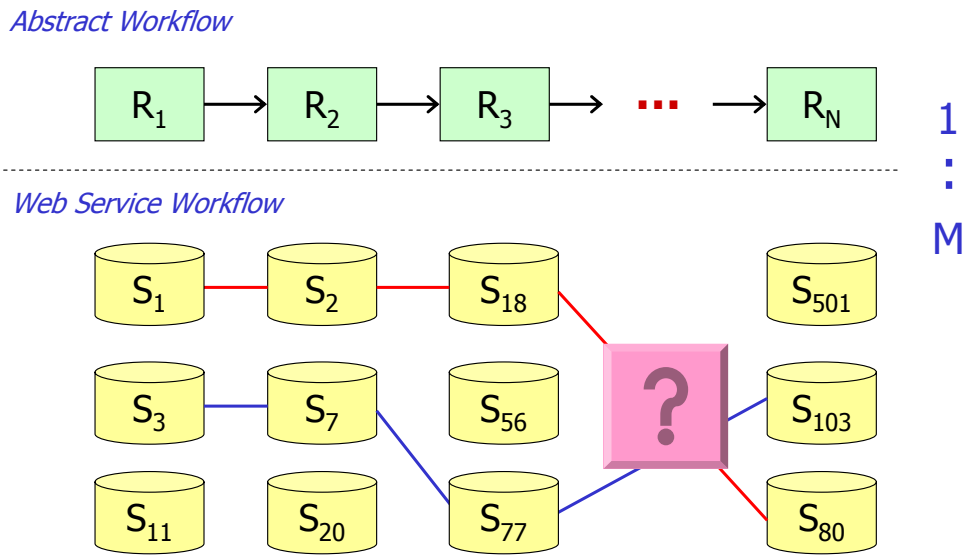


Figure 7-4. An illustration of the service chaining problem

Based on the WSP service selection criteria (see Section 7.4.2), we designed a service chaining algorithm that considers both semantics and data interoperability. This algorithm is a hybrid of local optimization and global optimization, because service accuracy is used for local optimization (vertical) and data interoperability is used for global optimization (horizontal).

Figure 7-5 shows the pseudo code of the chaining algorithm. There are three major steps in the main procedure “serviceIntegration”. The first step is to perform service discovery for each request by traversing the abstract workflow G. After service discovery, each request has a list of candidate services, which are matched services sorted based on their matching score. If there is no candidate service for a request, the algorithm will prompt the user to modify the request. The second step is to perform initial service selection based on the matching score of candidate services. For each request, the service with the highest match score is selected. If there is more than one service with the highest score, all such services are selected. The selected service(s) for each request is stored in “semiList”. The third step is to perform final service selection by choosing a service from the “semiList” for each request. The selection is based on the data interoperability. That is a service is selected if its input schema is compatible with the output schema of the previous selected service.

A time complexity analysis was performed for the algorithm (see Appendix A). The analysis shows the time complexity is O(|V| + |E|), where |V| is the number of vertices (requests) in G, and |E| is the number of edges.

```
procedure serviceIntegration(G, registration)        //G is an abstract workflow
                                                     //G = <V, E>, where V is requests
                                                     //and E is request dependencies
1.  reqest0 = the root vertex in G
2.  discovery(request0, registration)                //perform service discovery
3.  initialSelection(request0)                       //select services based on matching
                                                     //score
4.  finalSelection(request0)                         //further select services based on data
5.  return                                           //interoperability

procedure discovery(u, registration)                 //depth-first traversal of G
                                                     //u is a service request
1.  status[u] = "processed"
2.  candidates[u] = match(u, registration)
3.  for each neighbor v of u do
4.     if status[v] != "processed" then
5.        discovery(v, registration)
6.  return

procedure initialSelection(u)                        //depth-first traversal of G
                                                     //u is a service request
1.  status[u] = "processed"
2.  if candidates[u] = null then                     //check if there are candidate services
3.     return
4.  semiList[u] = top services in candidates[u]      //select services with highest score
5.  for each neighbor v of u do
6.     if status[v] != "processed" then
7.        initialSelection(v)
8.  return

procedure finalSelection(u)                          //depth-first traversal of G
                                                     //u is a service request
1.  status[u] = "processed"
2.  if u is root vertex in G then
3.     service[u] = first service in semiList[u]
4.     lastS = service[u]
5.  for i = 1 to m do                                //m is the number of services in
6.     S = semiList[u][i]                            //semiList[u], m ranges from 1 to 5
7.     if S.inputSchema = lastS.outputSchema then
8.        service[u] = S break
9.  if service[u] = null then
10.    service[u] = first service in semiList[u]
11. for each neighbor v of u do
12.    if status[v] != "processed" then
13.       lastS = service[u]
14.       finalSelection(v)
15. return
```

Figure 7-5. Pseudo code for the WSP service chaining algorithm.

8. WSP PROTOTYPE

8.1. Introduction

A WSP prototype was developed using the methodologies proposed in Chapters 3–7. This chapter discusses the implementation issues related to protein web services, ontologies, semantic service descriptions, and the WSP matchmaker.

8.2. Protein Feature Web Services

A protein feature web service contains operations that process protein features (see Chapter 4). Once a protein feature web service is appropriately modeled, it will be implemented as a reusable component wrapped with standard interfaces for invocation and discovery.

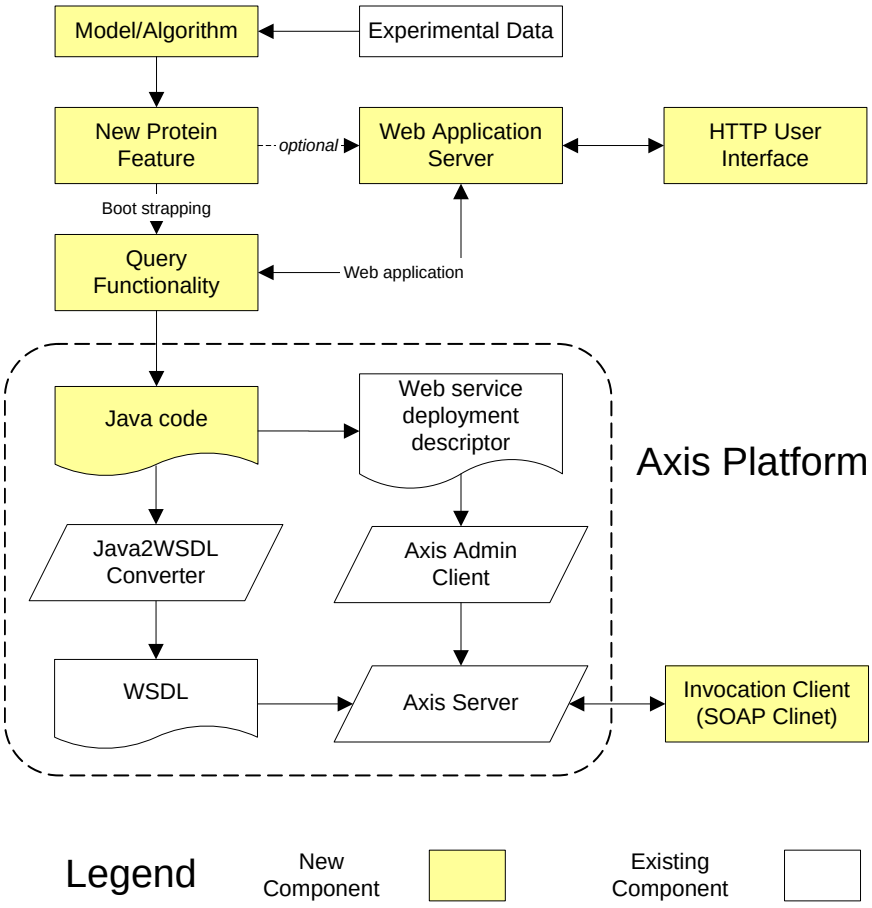


Figure 8-1. Life cycle to develop biological Web services using AXIS platform

Web services can be implemented in many different platforms (e.g., Microsoft .NET, IBM WebSphere/J2EE) according to the web services standards. In WSP, the Apache Axis (http://ws.apache.org/axis/java/) is adopted as the development platform because it has a SOAP engine for processing service requests and toolkits for generating WSDL interfaces. Figure 8-1 shows the life cycle of Web services development. The boxes in the figure represent activities that the developer needs to perform. The parallelograms in the figure represent tools provided by Axis. The document symbols represent code or file that the developer needs to generate, possibly with the assistance of tools.

For cross-platform interoperability, we implement Web services using Java. The Java code is converted into WSDL automatically with the Java2WSDL Converter. A Web Service Deployment Descriptor (WSDD) contains information to be deployed into Axis, i.e., service name, Java class name, and allowed methods. Once a WSDD file is generated, it is sent to the Axis Server by the Axis AdminClient in order to actually deploy the described service. After that, a binding agent based on Axis Client is used to invoke the service through the WSDL description.

8.2.1. iGNM Protein Dynamics Web Service

The iGNM service operations are implemented in Java. The implementations are then published to the Apache Axis engine using a publish client and a deployment file. The deployment file explicitly lists all the allowed operations (methods) and their Java class names:

```
<deployment name="test" xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="ignm-mode-service" provider="java:RPC">
    <parameter name="className" value="ignm.ignmModeService"/>
    <parameter name="allowedMethods" value="extract test2"/>
    <parameter name="allowedRoles" value="user1,user2"/>
    <parameter name="wsdlServicePort" value="GetQuote"/>
    <requestFlow name="checks">
      <handler type="java:org.apache.axis.handlers.SimpleAuthenticationHandler"/>
      <handler type="java:org.apache.axis.handlers.SimpleAuthorizationHandler"/>
    </requestFlow>
  </service>
</deployment>
```

Once a service and its operations are published, Apache Axis automatically generates a WSDL programmatic interface based on the definitions of iGNM service operations. Figure 8-2 shows a portion of the iGNM service's WSDL interface, where the operation name is called "extract", the input message is called "extractRequest" and the output message is called "extractResponse".

The data schemas for both input message and output message are explicitly defined. For example, the input data schema consists of two strings and one integer, and the output data schema consists of a double array.

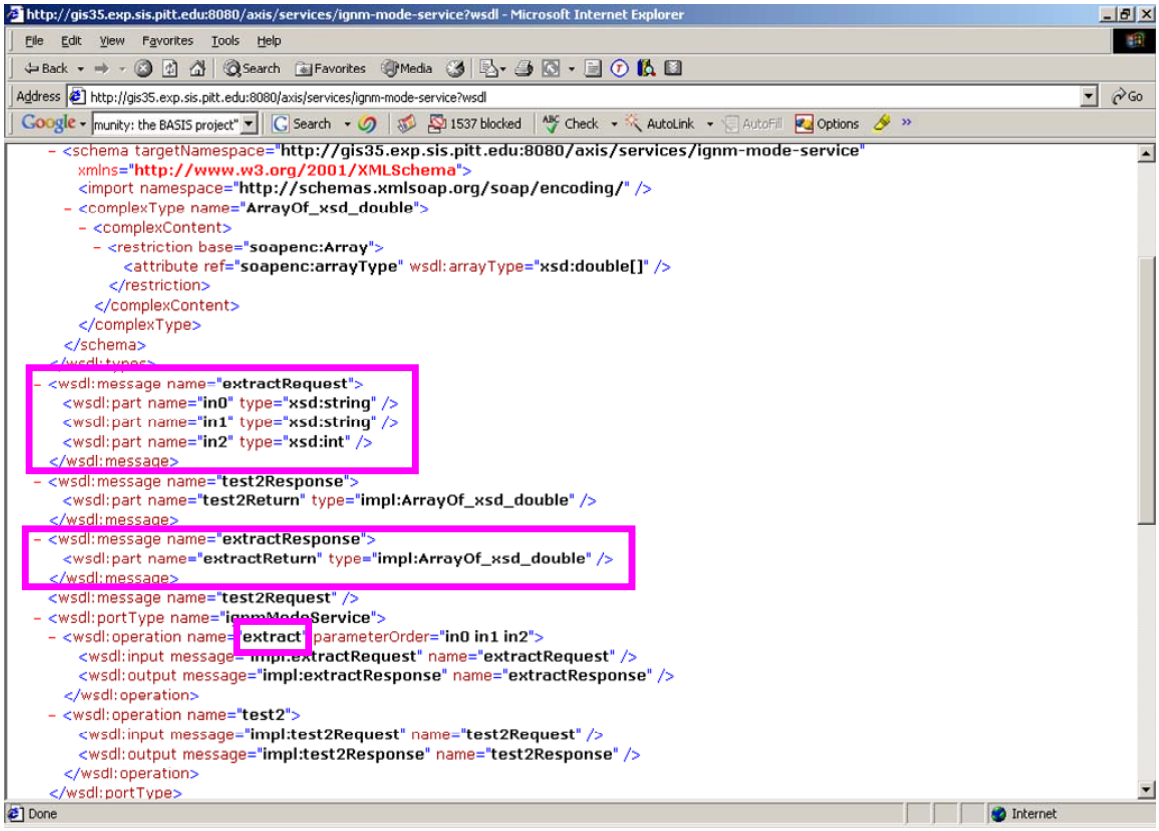


Figure 8-2. Fragment of iGNM web service's WSDL interface

A SOAP client is used to invoke the iGNM web service. The client first reads the definition of the selected operation and its input and output schema. Then it passes in the input values and makes binds to the iGNM service to obtain the output values. For example, to obtain the 1st slow mode for the PDB structure 101m, the following service binding command is used:

```
java ignm.ignmClient -lhttp://gis35.exp.sis.pitt.edu:8080/axis/services/ignm-mode-service -uuser1
-wpass1 101m slowmode 1
```

where the SOAP client is called “ignm.ignmClient”, the iGNM service is called “ignm-modeshape-service”, and the default account and password are called “user1” and “pass1” respectively. This command returns an array of residue fluctuations, where each fluctuation is a double value.

8.2.2. N-gram Conservation Profile Web Service

The n-gram conservation operations are implemented in Java. The implementations are then published to the Apache Axis engine using a publish client and deployment file. The deployment file lists the allowed operations (methods) and their Java class names:

```
<deployment name="test" xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="ngram-profile-service" provider="java:RPC">
    <parameter name="className" value="ngram.ngramService"/>
    <parameter name="allowedMethods" value="extract"/>
    <parameter name="allowedRoles" value="user1,user2"/>
    <parameter name="wsdlServicePort" value="GetNgram"/>
    <requestFlow name="checks">
      <handler type="java:org.apache.axis.handlers.SimpleAuthenticationHandler"/>
      <handler type="java:org.apache.axis.handlers.SimpleAuthorizationHandler"/>
    </requestFlow>
  </service>
</deployment>
```

Once n-gram service is published, Apache Axis automatically generates a WSDL interface based on its definitions. Figure 8-3 shows a portion of the n-gram conservation service’s WSDL.

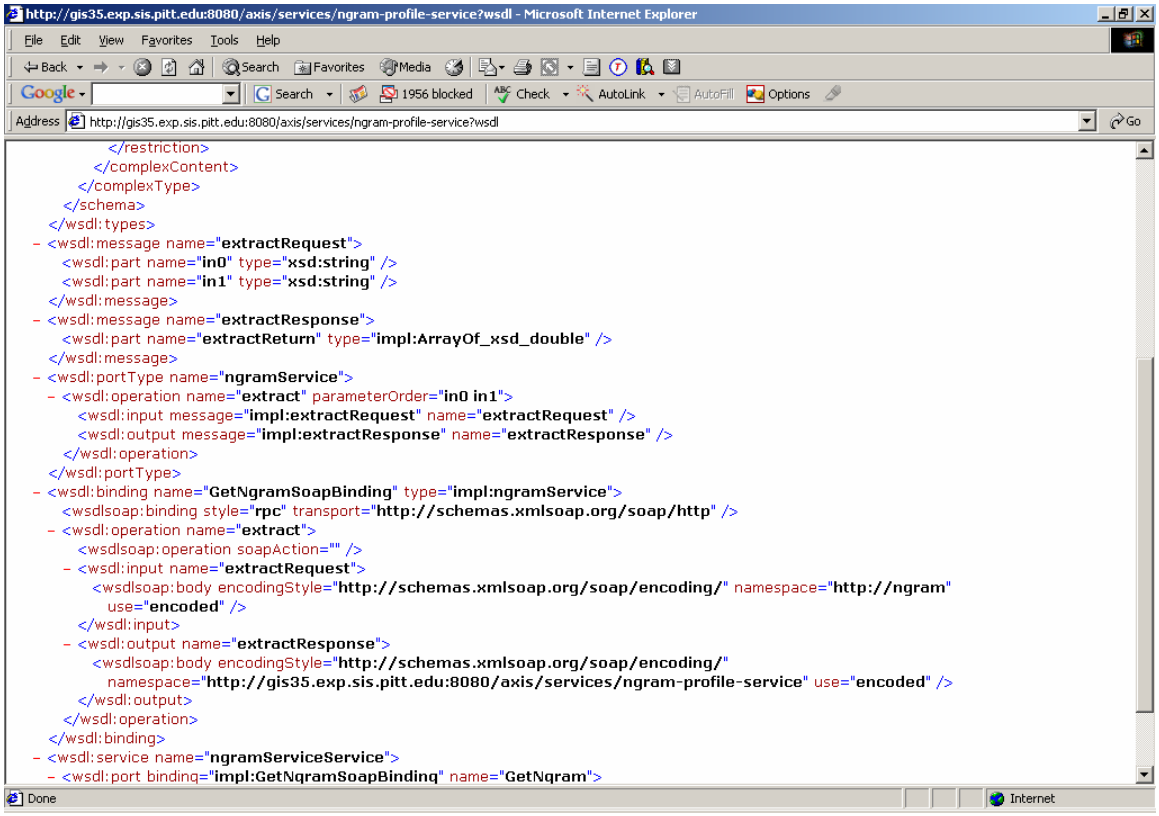


Figure 8-3. N-gram conservation service's WSDL interface

A SOAP client is used to invoke the n-gram conservation service. For example, to obtain the PDB structure 101m's chain "0", the following service binding command is used:

```
java ngram.ngramClient -lhttp://gis35.exp.sis.pitt.edu:8080/axis/services/ngram-profile-service -
uuser1 -wpass1 101m 0
```

where the SOAP client is called "ngram.ngramClient", the service is called "ngram-profile-service", and the default account and password are called "user1" and "pass1" respectively. This operation call returns an array of conservation percentages for residues in chain "0".

8.3. Semantic Descriptions of Services

8.3.1. Implementation of the EPO Ontology

EPO (see Section 5.3.2) is implemented in OWL using the Protégé OWL Editor (Protégé, 2002), see Figure 8-4. The current implementation of EPO contains 122 concepts, including 92 PO concepts and 30 additional concepts. Figure 8-5 shows a fragment of EPO OWL ontology, where each class element represents a concept, and the "subClassOf" attribute provides the hierarchy relationship between concepts. For example, "Protein" is a subclass of "ExtendedProteinOntologyConcept", and "Dynamics" is a subclass of "ThreeD_Parameters".

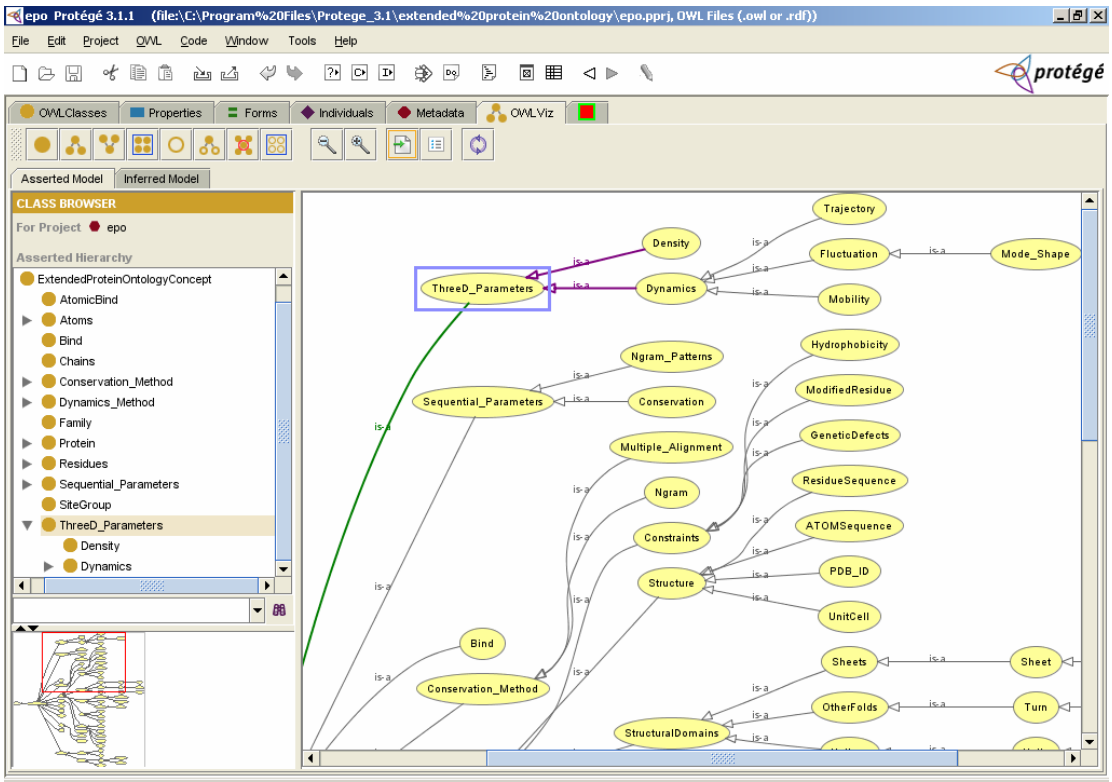


Figure 8-4. Development of EPO in Protégé

```
<owl:Class rdf:about="#Protein">
  <rdfs:subClassOf rdf:resource="#ExtendedProteinOntologyConcept"/>
</owl:Class>
<owl:Class rdf:about="#Dynamics">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ThreeD_Parameters"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PDB_ID">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Structure"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PhysiologicalFunctions">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BiologicalFunction"/>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 8-5. Fragment of the EPO OWL ontology

8.3.2. Generating OWL-S Service Descriptions

The OWL-S profile generator (see Section 5.4) is implemented in Java. It incorporates EPO OWL ontology and OWL-S profile ontology to generate a wide range of semantic descriptions of web services on complex protein data related issues.

Figure 8-6 shows the sample OWL-S service descriptions for iGNM, ProMode, and MolMovDB, using the EPO ontology. The description of iGNM specifies the input as “PDB ID”, the output as “mode shape” and the constraint as “residues”, which means that iGNM provides mode shape at the residue level for a given PDB ID. The description of ProMode specifies the input as “PDB ID”, the output as “mode shape” and the constraint as “atoms”, which means that ProMode provides mode shape at the atom level for a PDB structure. The description of MolMovDB specifies the input as “PDB ID” and the output as “Trajectory”, which means that MolMovDB provides protein motion trajectories for PDB structures.

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=”mode_shape”>
  <parameterType>http://ontologyURL/EPO.owl#Mode_Shape
  </parameterType>
</Output>
<Constraint rdf:ID=”Residues”>
  <parameterType>http://ontologyURL/EPO.owl#Residues
  </parameterType>
</Constraint>
<profile rdf:ID="dynamics service">
  <serviceName>iGNM_Mode_Shape</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#mode_shape"/>
  <hasConstraint rdf:resource=”#Residues”/>
</profile>
```

(a)

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=”mode_shape”>
  <parameterType>http://ontologyURL/EPO.owl#Mode_Shape
  </parameterType>
</Output>
<Constraint rdf:ID=”Atoms”>
  <parameterType>http://ontologyURL/EPO.owl#atoms
  </parameterType>
</Constraint>
<profile rdf:ID="dynamics service">
  <serviceName>ProMode</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#mode_shape"/>
  <hasConstraint rdf:resource=”#Atoms”/>
</profile>
```

(b)

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=”trajectory”>
  <parameterType>http://ontologyURL/EPO.owl#Trajectory
  </parameterType>
</Output>
<profile rdf:ID="dynamics service">
  <serviceName>MolMovDB</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#trajectory"/>
</profile>
```

(c)

Figure 8-6. Sample OWL-S service descriptions for services that provide protein dynamics data. (a) iGNM mode shape service description; (b) sample ProMode service description; (c) sample MolMovDB service description.

Similarly, users looking for protein dynamics data may have different requirements. In this case, OWL-S semantic descriptions of expected services can be generated to precisely capture the user’s data requirements. For example, if a user is looking for general protein dynamics data,

he/she can submit a request which specifies the input as “protein” and the output as “dynamics”, see Figure 8-7(a). However, if a user is looking for specific protein dynamics data, e.g., the mode shape for a PDB structure at the residue level, he/she can submit a request which specifies the input as “PDB ID”, the output as “mode shape” and the constraint as “residues”, see Figure 8-7(b).

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#Protein
  </parameterType>
</Input>
<Output rdf:ID=”Dynamics”>
  <parameterType>http://ontologyURL/EPO.owl#Dynamics
  </parameterType>
</Output>
<profile rdf:ID="dynamics service">
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#Dynamics"/>
</profile>
```

(a)

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=”mode_shape”>
  <parameterType>http://ontologyURL/EPO.owl#Mode_Shape
  </parameterType>
</Output>
<Constraint rdf:ID=”Residue”>
  <parameterType>http://ontologyURL/EPO.owl#Residues
  </parameterType>
</Constraint>
<profile rdf:ID=”dynamics service”>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#mode_shape"/>
  <hasConstraint rdf:resource=”#Residue”/>
</profile>
```

(b)

Figure 8-7. Two sample OWL-S service requests that look for protein dynamics data.

Figure 8-8 shows a portion of a sample service description used to describe the N-gram Conservation Profile Service, which takes a PDB structure as input and produces a conservation profile as output. The description contains information about the input, output and constraint. More specifically, the input refers to the EPO concept of “PDB ID”, the output refers to the EPO concept of “conservation”, and the constraint refers to the EPO concept of “Ngram”.

```
<Input rdf:ID=”Protein”>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=”conservation”>
  <parameterType>http://ontologyURL/EPO.owl#Conservation
  </parameterType>
</Output>
<constraint rdf:ID=”method”>
  <parameterType>http://ontologyURL/EPO.owl#Ngram
  </parameterType>
</constraint>
<profile rdf:ID="conservation service">
  <serviceName>ProMode</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#conservation"/>
  <hasConstraint rdf:resource=”#method”/>
</profile>
```

Figure 8-8. Sample OWL-S description of n-gram conservation service

8.4. WSP Matchmaker

The Apache JUDDI (http://ws.apache.org/juddi/) is selected as the development platform for the WSP matchmaker (see Chapter 6). The advantage of JUDDI is that it is an open source Java implementation of the UDDI specification for Web Services. The WSP publication algorithm (see Section 6.4) and the WSP service matching algorithm (see Section 6.5) are integrated with JUDDI for semantic publication and discovery. Also, Jena toolkit (http://www.hpl.hp.com/sem web/jena.htm) is used to parse OWL ontologies for model representation and service matching.

9. WSP EVALUATION

9.1. Introduction

A number of experiments were performed to evaluate the performance of the WSP matchmaker. The parameters tested include: (1) solution space, (2) accuracy and (3) time efficiency. The results show that the matchmaker can efficiently match services from registered services. Also, a composite service, which integrates protein dynamics and conservation, is developed to demonstrate the effectiveness of the integration agent.

9.2. Evaluation of the WSP Matchmaker

9.2.1. Experimental Setups

Two hundred OWL-S service descriptions were generated using the OWL-S profile generator (see Section 5.4 and Section 8.3). These service descriptions reflect a wide range of possible protein web services, from protein dynamics services to protein conservation services and to functional domains services (see Appendix B).

Twenty OWL-S descriptions were randomly selected from the 200 OWL-S files to represent possible service requests. The 20 requests include 10 requests without constraints (2 concepts:

input and output) and 10 with constraints (3 concepts: input, output and constraint), see Table 9-1.

Table 9-1. Sample WSP service requests

Request Index	Service Input	Service Output	Constraints
1	Protein	Dynamics	
2	Protein	Conservation	
3	Protein	Catalytic Sites	
4	Protein	Active Binding Sites	
5	Structure	Hydrophobicity	
6	Structure	Ngram Patterns	
7	Structure	Residue Sequence	
8	PDB_ID	Mobility	
9	PDB_ID	3D Parameters	
10	PDB_ID	Physiological Functions	
11	Protein	Dynamics	Experimental
12	Protein	Dynamics	NMA
13	Protein	Conservation	Multiple Alignment
14	Protein	Conservation	Ngram
15	Structure	Dynamics	Computational
16	Structure	Catalytic Sites	Experimental
17	Structure	Catalytic Sites	Computational
18	PDB_ID	Hydrophobicity	Residues
19	PDB_ID	Hydrophobicity	Atoms
20	Chains	Functional Domains	Computational

Figure 9-1 shows the process to perform service matching. The 200 services are randomly divided into 20 sets, each containing 10 services. The 20 sets are published to the WSP matchmaker sequentially. After a set is published, service matching is performed using the 20 requests (see Table 9-1). The matching results (e.g., matched services for each request) are recorded for further analysis.

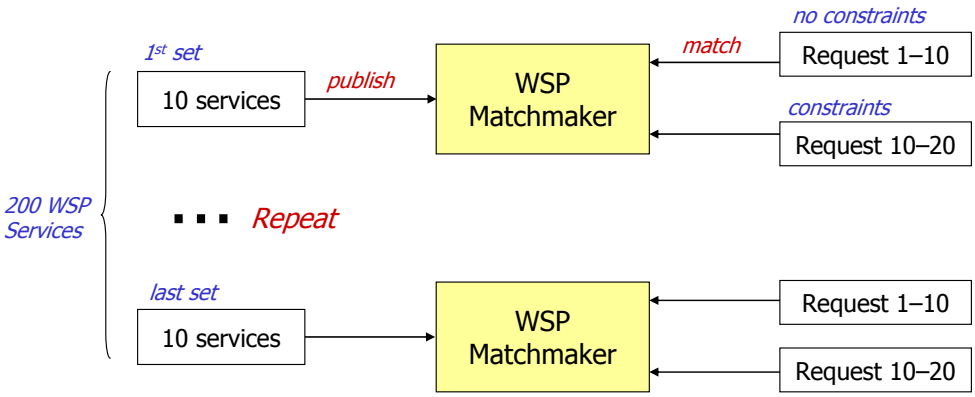


Figure 9-1. Illustration of the matchmaker evaluation process

Currently, EPO and PO have limited number of concepts, which in turn leads to limited number of semantic service descriptions about protein web services. As the ontology development in the protein domain continues to grow, the size of EPO/PO is expected to grow. To test the matching algorithm's scalability with respect to the ontology size, a 1000-concept random ontology was generated. To test the algorithm's scalability with respect to the number of registered services, 1000 service descriptions (500 with constraints and 500 without constraints) were generated using the 1000-concept random ontology.

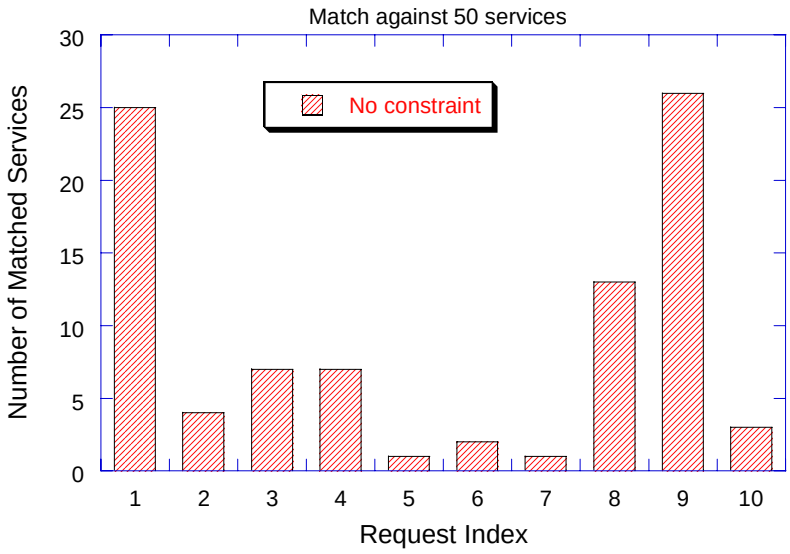
To generate the random ontologies, concept names were created in the form of "conceptX" where X varies from 1 to N (e.g., N = 1000 for the 1000-concept ontology). The concept names are then randomly linked through a subclass relation. The services and requests are generated by randomly selecting concepts for inputs, outputs and constraints from the randomly generated ontology.

The experimental process for the 1000 random service descriptions is the same as that for the 200 WSP service descriptions. 20 service requests (10 with constraints and 10 without constraints) were randomly selected and used to match against the 1000 random services.

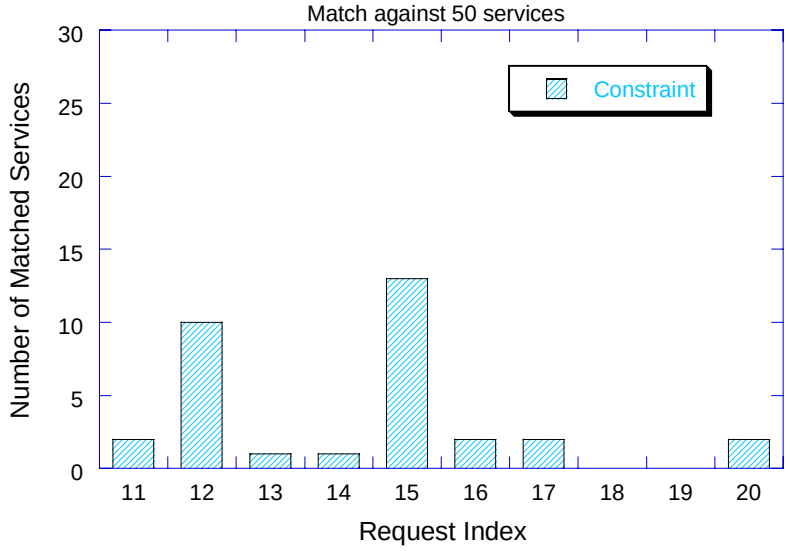
9.2.2. Analysis of Solution Space

For a give request, the matched services constitute its solution space. The solution space is dependent on the available services registered in the matchmaker. Figure 9-2 shows the number of matched services for the 20 requests (see Table 9-1) when there are only 50 WSP services. As can be seen, the number of matched services varies from request to request. For example, "Request 1" has 25 matched services in its solution space while "Request 18" does not have matched services and thereby a null solution space.

Also, it can be seen that the solution space for requests without constraints (see Figure 9-2(a)) is in general larger than that for requests with constraints (see Figure 9-2(b)). This can be explained by the number of concepts that need to be matched. For requests without constraints, there are only two concepts (input and output) that need to be matched, while for requests with constraints, there are three concepts (input, output and constraint) that need to be matched. Therefore, requests with constraints require stricter matching and thereby a smaller solution space (see Section 6.6).



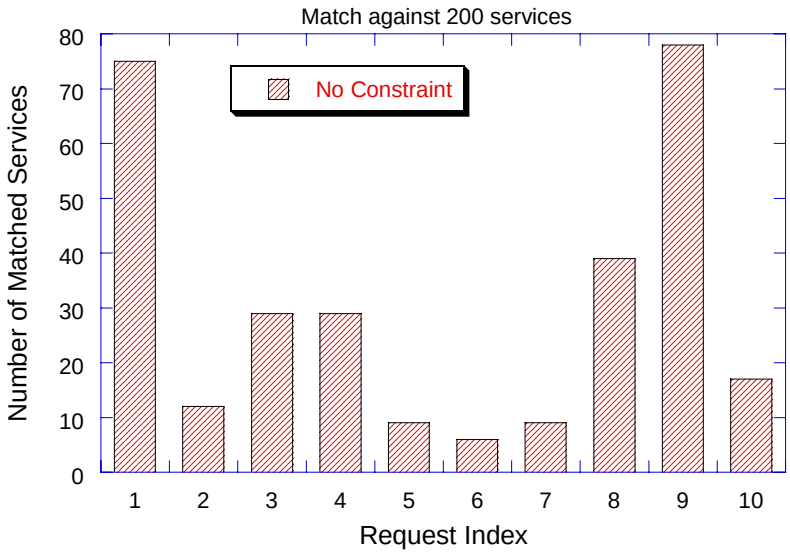
(a)



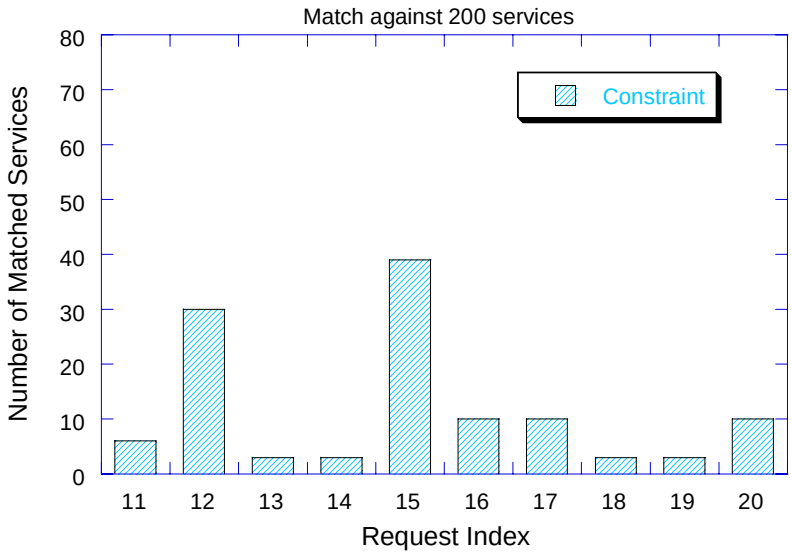
(b)

Figure 9-2. Number of matched services when there are 50 WSP services: (a) using 10 requests without constraints; (b) using 10 requests with constraints.

Figure 9-3 shows the number of matched services for 20 requests when all the 200 WSP services are registered in the matchmaker. As it is obvious (compared with the 50 services situation shown in Figure 9-2), the solution space for each request in the case of 200 services increases. This is because there are more services for matching, i.e., with a larger search space, there is a larger solution space.



(a)



(b)

Figure 9-3. Number of matched services when there are 200 WSP services: (a) using 10 requests without constraints; (b) using 10 requests with constraints.

Figure 9-4 illustrates the increase of solution space when the search space (registered services) increases from 10 to 200. It can be seen that there is a linear increase of solution space for both types of requests (without or with constraints). When the search space is creased 300% (from 50 to 200), there is a 1000% increase (from 2.7 to 30.3) in solution space for the 10 requests without constraints, and there is a 255% increase (from 3.3 to 11.7) for the 10 requests with constraints.

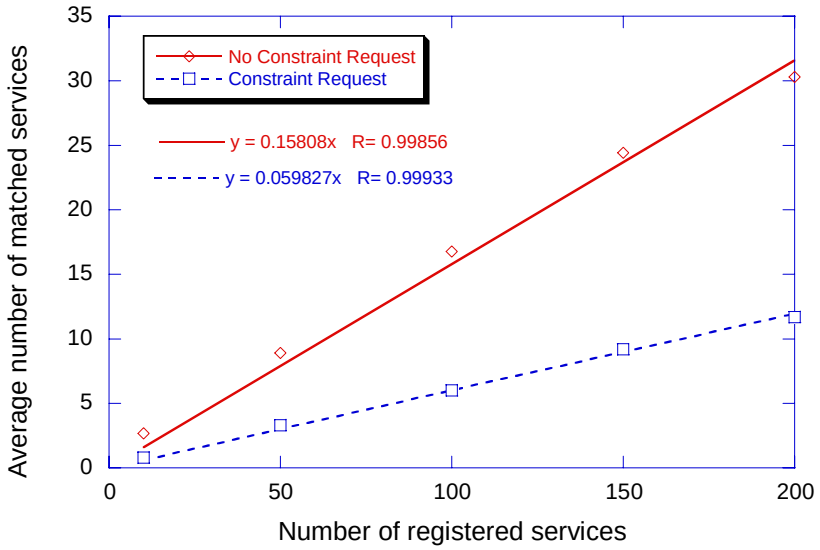


Figure 9-4. Average number of matched services as a function of search space.

The services in a solution space have differing match scores and thereby differing capabilities to meet the requirements of the request. There is a distribution of match scores associated with a solution space. Figure 9-5 shows four examples of score distributions, where (a) is the score distribution for “Request 1” with 50 services, (b) is the score distribution for “Request 1” with 200 services, (c) is the score distribution for “Request 15” with 50 services, and (d) is the score distribution for “Request 15” with 200 services. These distributions show that there are only a few services with the highest match score.

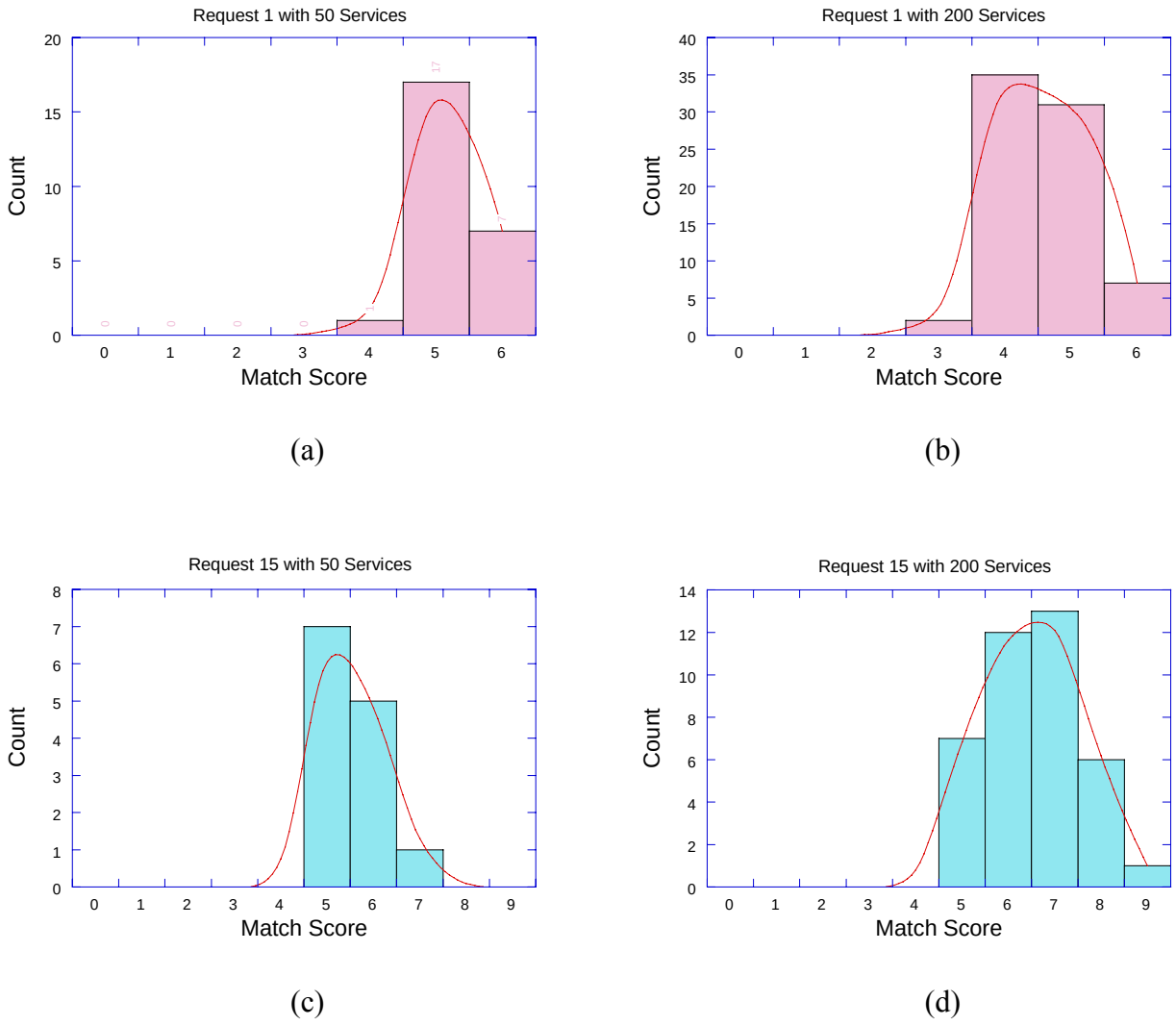


Figure 9-5. Sample match score distributions.

Figure 9-6 shows the average match score for the solution space (each for a request) when there are 200 services. It can be seen that the average score for a solution space varies from request to request. Figure 9-7 shows the average match score as a function of the search space (registered services). It can be seen that the average match score for both types of requests (without or with constraints) does not scale with the search space.

(a)

(b)

Figure 9-6. Average match score when there are 200 WSP services: (a) using 10 requests without constraints; (b) using the 10 requests with constraints.

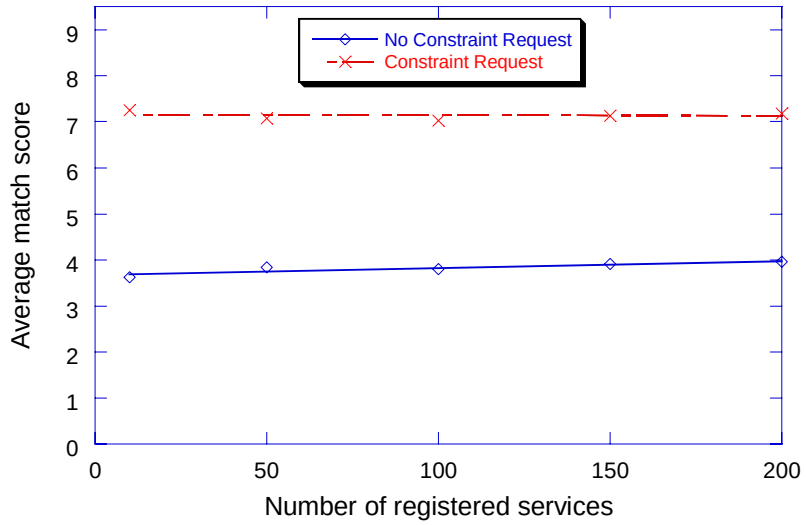


Figure 9-7. Average match score as a function of search space.

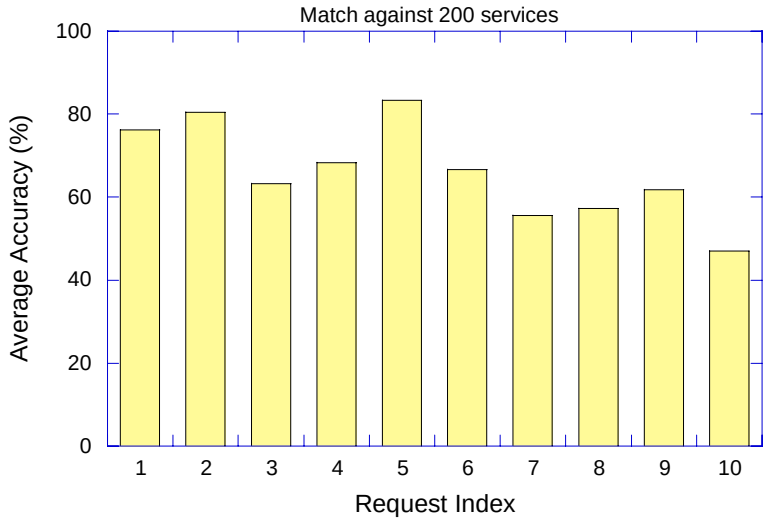
9.2.3. Analysis of Matching Accuracy

In this research, accuracy *acc* is a measure of how a service request semantically matches a service. It is defined as the match score *x* divided by the largest possible match score *max*, see Equation 9-1. For request without constraints, the largest possible match score is 6 (3 for input and 3 for output). For requests with constraints, the largest possible match score is 9 (3 for input, 3 for output and 3 for constraint).

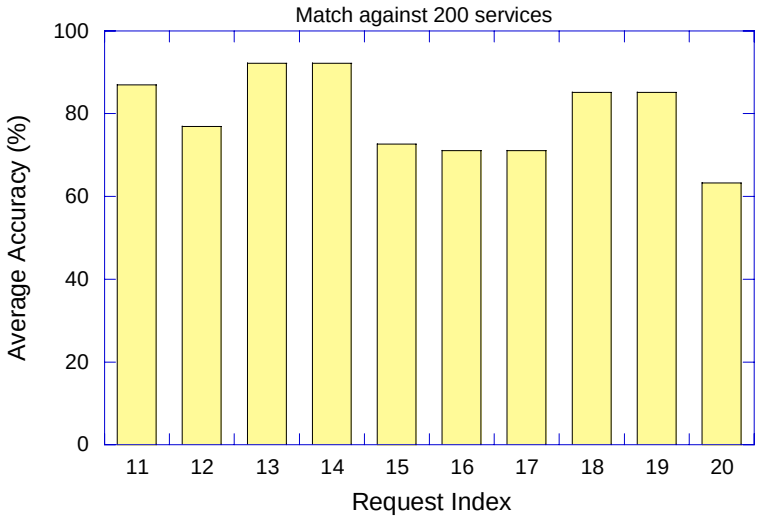
$$acc = \frac{x}{max} \quad (0 \le x \le max, \;\; 0 \le acc \le 1)$$

$$= \begin{cases} \frac{x}{6} & no\ constra\ int \\ \frac{x}{9} & constra\ int \end{cases} \qquad 9\text{-}1$$

The accuracy of each matched service was calculated using the math score *x*. The average accuracy for each request's solution space is also calculated. For requests without constraints, the average accuracy is within 40% ~ 85%, see Figure 9-8(a). For requests with constraints, the average accuracy falls within 60% ~ 95%, see Figure 9-8(b).



(a)



(b)

Figure 9-8. Matching accuracy with 200 services: (a) using 10 requests without constraints; (b) using 10 requests with constraints.

Figure 9-9 shows the average accuracy as a function of the search space. It can be seen that the average accuracy for both types of requests (without or with constraints) does not scale with the search space. Also, the average accuracy for requests with constraints appears to be higher than the accuracy for requests without constraints. This suggests that formulating requests with constraints could improve accuracy.

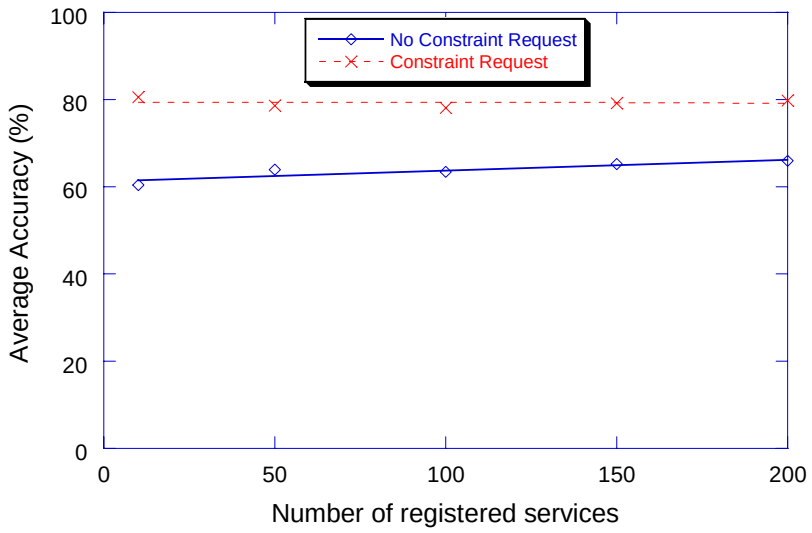
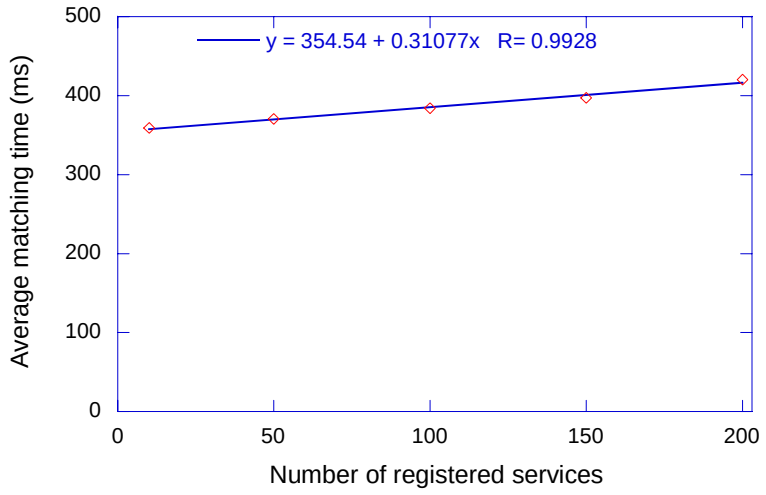


Figure 9-9. Average accuracy as a function of search space.

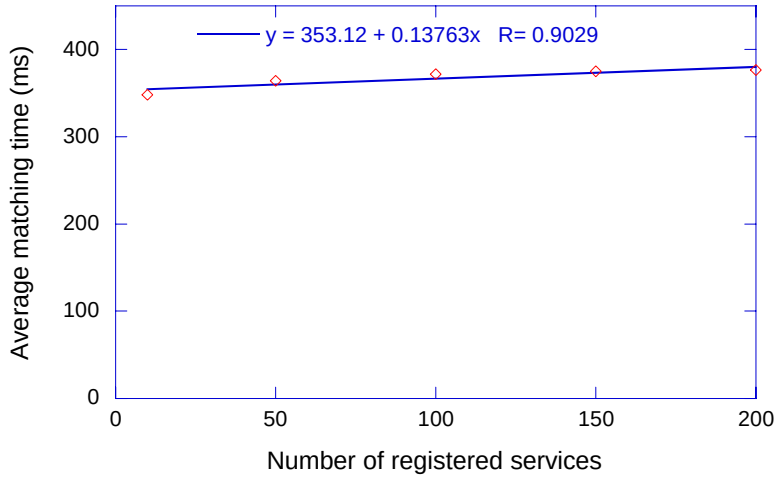
9.2.4. Analysis of Service Matching Time

Figure 9-10(a) shows the average matching time for the 10 WSP requests without constraints. The value varies from 360ms to 420ms. Figure 9-10(b) shows the average matching time for the 10 WSP requests with constraints. The value varies from 350ms to 380ms. Figure 9-10(c) shows the average matching time using the 1000-cocnetp random dataset. The value varies from 465ms to 530ms. These time ranges show that the matchmaker can discover services within seconds.

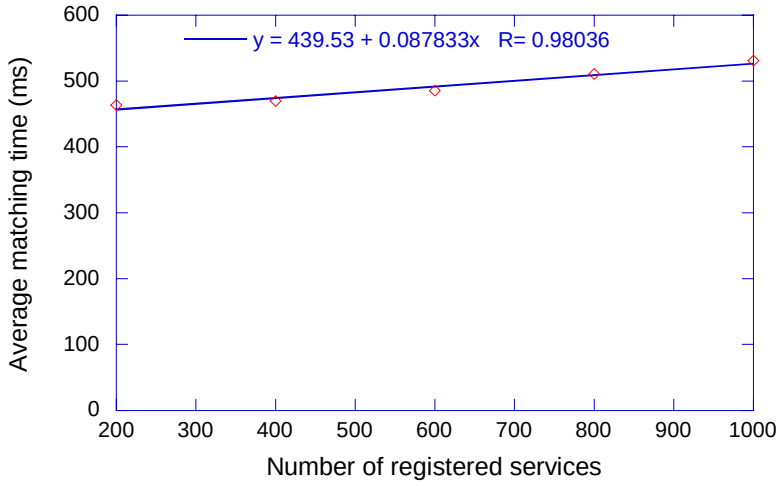
The data also shows that the matching time increases as the number of registered services increases, which is consistent with the matching time complexity, see Section 6.5.1. In terms of trend, the data shows that the percentage of time increase is small as the number of registered services increases. Therefore, the overhead of service matching is small.


y = 354.54 + 0.31077x R= 0.9928
Average matching time (ms)
Number of registered services
0
100
200
300
400
500
0
50
100
150
200

(a)


y = 353.12 + 0.13763x R= 0.9029
Average matching time (ms)
Number of registered services
0
100
200
300
400
0
50
100
150
200

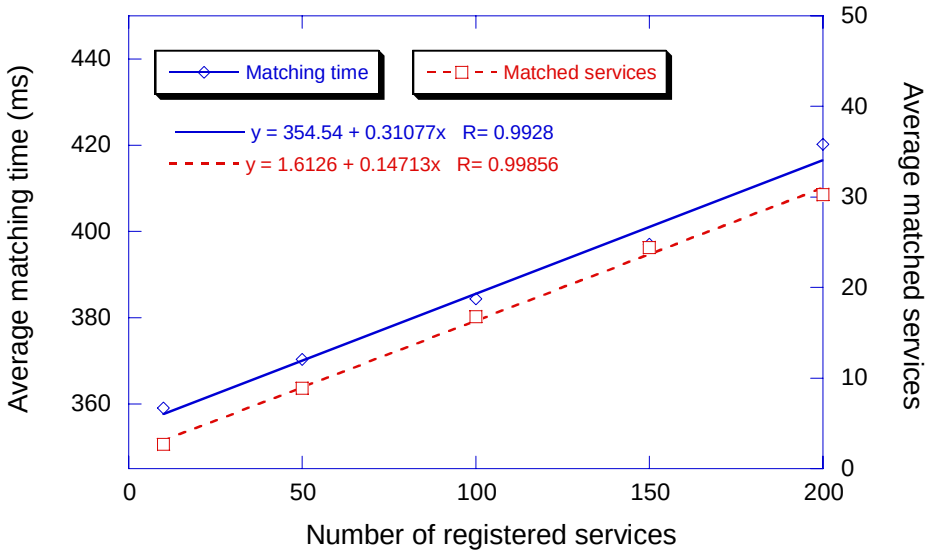
(b)



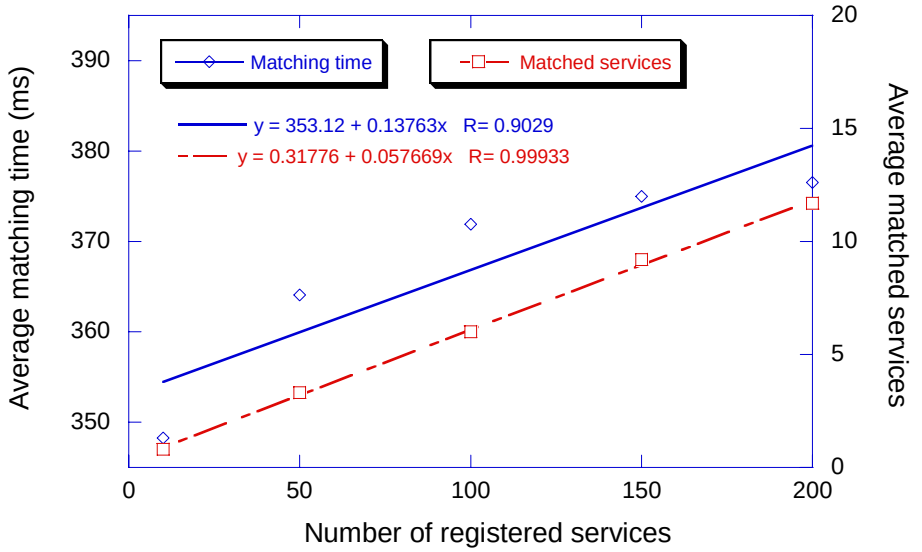
(c)

Figure 9-10. (a) Service matching time for 10 service requests without constraint; (b) service matching time for 10 requests with constraint; (c) service matching time for random requests.

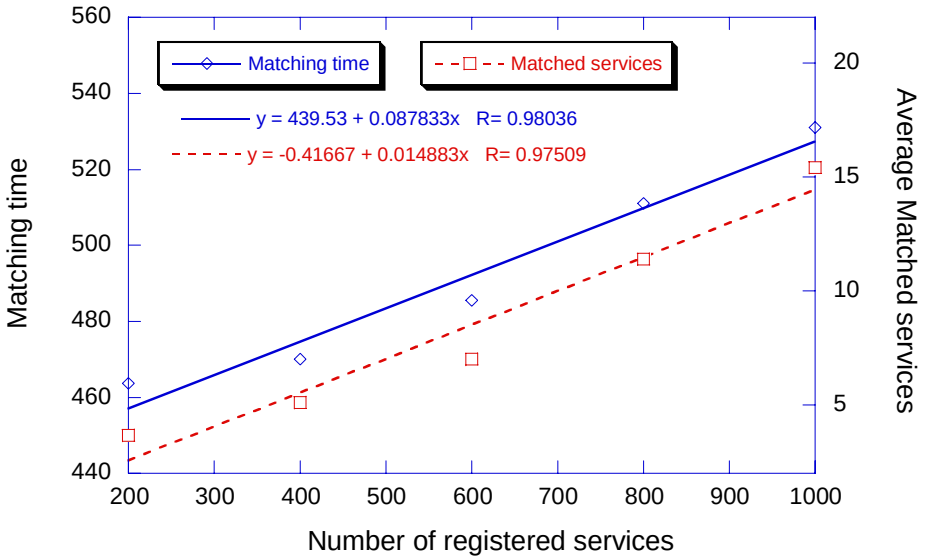
Figure 9-11 correlates the average matching time with the average number of matched services (size of solution space). It can be seen that the number of matched services has a strong correlation with the average matching time. This is consistent with the matching process and the time complexity discussed in Section 6.5.1.


Matching time
Matched services
y = 354.54 + 0.31077x R= 0.9928
y = 1.6126 + 0.14713x R= 0.99856
Average matching time (ms)
Average matched services
Number of registered services
440
420
400
380
360
50
40
30
20
10
0
0
50
100
150
200

(a)


Matching time
Matched services
y = 353.12 + 0.13763x R= 0.9029
y = 0.31776 + 0.057669x R= 0.99933
Average matching time (ms)
Average matched services
Number of registered services
390
380
370
360
350
20
15
10
5
0
0
50
100
150
200

(b)



(c)

Figure 9-11. Correlation between matching time and solution space: (a) using 10 service requests without constraint; (b) using 10 requests with constraint; (c) using random requests.

9.3. Evaluation of the WSP Integration Agent

In this section, we present a WSP scenario describing the integration of protein dynamics data with sequence data. Figure 9-12 shows the higher-level workflow. The initial condition is a protein name or ID. The first task is to obtain the protein's dynamics information. The second task is to obtain the protein's conservation profile. The third task is to correlate the dynamics information with the conservation profile.

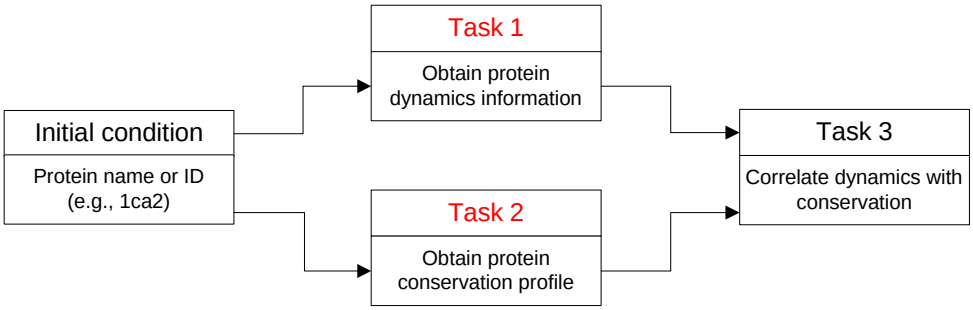


Figure 9-12. A sample workflow for integrating dynamics data with conservation data

Figure 9-13 shows the process of integrating dynamics data with sequence data, using the workflow shown in Figure 9-12. First, the integration agent takes the workflow and generates semantic service requests for each task (Step 1 in Figure 9-13). For Task 1, the request is to find a service which provides specific protein dynamics data (e.g., the mode shape at residue level) for a given protein (e.g., PDB ID). For Task 2, the request is to find a service which provides conservation profile (e.g., n-gram based conservation) for the protein. For Task 3, the request is to find a service which correlates two profiles at each residue position.

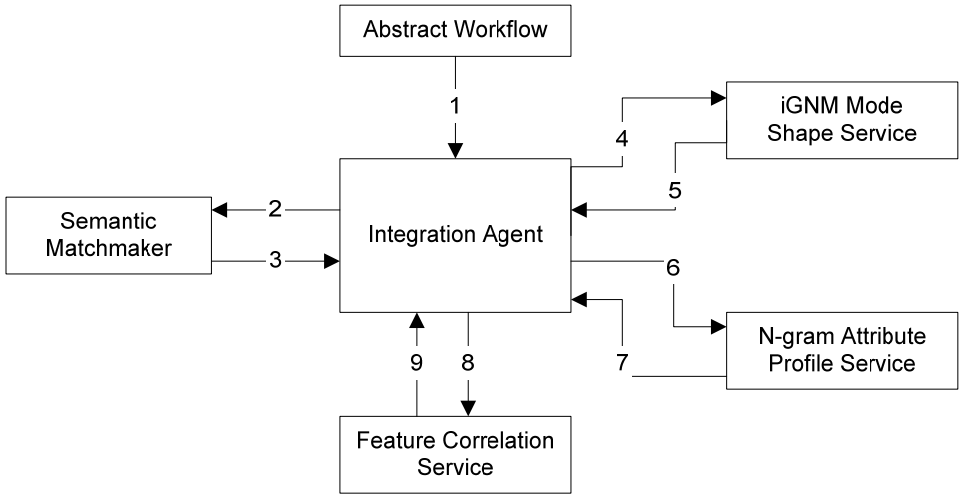


Figure 9-13. Integrating dynamics data with sequence data

Figure 9-14 shows a portion of an OWL-S request that is used to describe the task. The input refers to the EPO concept of "PDB ID", the output refers to the EPO concept of "Mode Shape" and the constraint refers to the EPO concept of "Residues".

```
<Input rdf:ID=″Protein″>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=″mode_shape″>
  <parameterType>http://ontologyURL/EPO.owl#Mode_Shape
  </parameterType>
</Output>
<Constraint rdf:ID=″Residues″>
  <parameterType>http://ontologyURL/EPO.owl#Residues
  </parameterType>
</Constraint>
<profile rdf:ID="dynamics service">
  <serviceName>iGNM_Mode_Shape</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#mode_shape"/>
  <hasConstraint rdf:resource=″#Residues″/>
</profile>
```

Figure 9-14. Sample OWL-S request for protein dynamics (mode shapes)

Figure 9-15 shows a portion of an OWL-S request that is used to describe Task 2. The input refers to the EPO concept of "PDB ID", and the output refers to the EPO concept of "conservation".

```
<Input rdf:ID=″Protein″>
  <parameterType>http://ontologyURL/EPO.owl#PDB_ID
  </parameterType>
</Input>
<Output rdf:ID=″conservation″>
  <parameterType>http://ontologyURL/EPO.owl#Conservation
  </parameterType>
</Output>
<profile rdf:ID="conservation service">
  <serviceName>ProMode</serviceName>
  <hasInput rdf:resource="#Protein"/>
  <hasOutput rdf:resource="#conservation"/>
</profile>
```

Figure 9-15. Sample OWL-S request for protein conservation

The matchmaker is used to perform semantic matching between each service request and registered service descriptions (Step 2 in Figure 9-13), and returns a set of matched services for each request (Step 3 in Figure 9-13). After obtaining all matched services (candidate services) from the matchmaker, the chaining algorithm is applied to select and chain services. For example, the chaining algorithm selects iGNM Mode Shape Service for Task 1 based on the high match score between the iGNM service description and the service request (see Section 6.6). Similarly, the algorithm selects the N-gram Conservation Profile Service for Task 2 and the feature overlay service (see Section 4.7) for Task 3. The integration agent then invokes the selected services as a group by sending SOAP requests (Steps 4-7 in Figure 9-13). The output of the iGNM Mode Shape Service (e.g., 1st slow mode) and the output of the N-gram Conservation Profile Service are used as the input to the feature overlay service (Step 8 in Figure 9-13). The output (a correlation map) of the feature overlay service constitutes the final result (Step 9 in Figure 9-13).

Figure 9-16 shows a sample correlation map between dynamics and conservation for carbonic anhydrase (PDB ID: 1ca2). It can be seen that conserved regions tend to have small motions (e.g., position 100) while non-conserved regions tend to have large motions (e.g., position 20). This kind of statistical relationships could potentially lead to prediction models that predict dynamics directly from sequence data instead of 3D structure data.

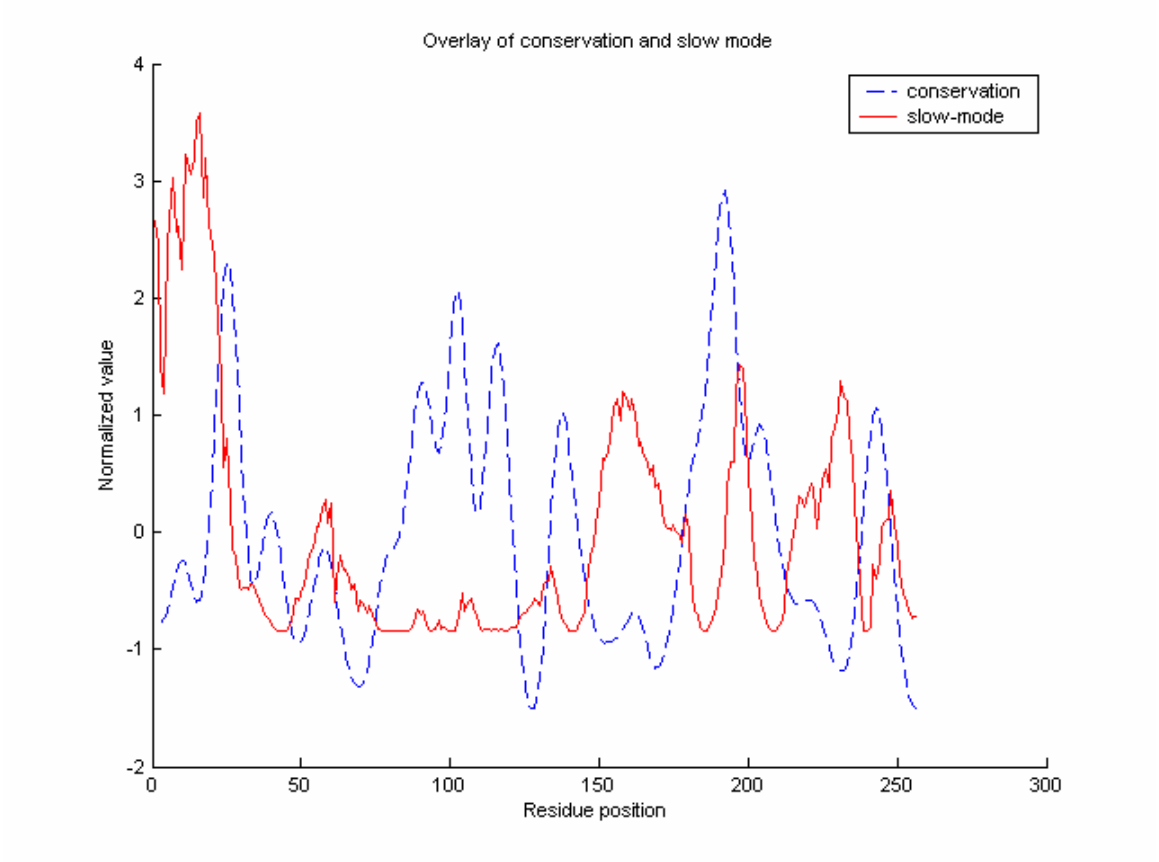


Figure 9-16. Correlation between protein dynamics and conservation, using the carbonic anhydrase (PDB ID: 1ca2) as an example.

10. CONCLUSION AND FUTURE RESEARCH

10.1. Summary of the Research

Understanding the function of every protein is one of the major objectives of bioinformatics. Currently, there is a lot of information (e.g., sequence, structure and dynamics) being produced by experiments and predictions that are associated with protein function. Integrating these diverse data about protein sequence, structure, dynamics and other protein features allows further exploration and establishment of the relationships between protein sequence, structure, dynamics and function, and thereby controlling the function of target proteins.

Currently, many protein data resources are accessible to researchers through Web application interfaces, e.g., through a HTTP form and a corresponding Java servlet. To integrate data through Web application interfaces (current approach), users (researchers) need to perform such tedious tasks as screen scraping or writing scripts to extract data while ignoring explanatory text and graphics associated with the HTML code. Since different web applications have different HTML code, this approach is labor intensive and not scalable.

Web services, on the other hand, offer an environment for flexile integration of various data resources, including databases, web servers and software tools. Web services provide standard programmatic interfaces (e.g., WSDL interface) for user applications to obtain explicit results without tedious screen scraping tasks. Also, web services provide protocols and tools that

facilitate the discovery and integration of data resources. However, current web service standards (e.g., WSDL and UDDI) do not support operations at semantic level, leaving the promise of automatic discovery and integration of web services incomplete. For example, service providers and requestors may have distinctive perspectives and knowledge about one service resulting in differing descriptions for the service. In this case, a UDDI registry will be unable to locate the service because it can only perform syntactic, and not semantic, matching between the service requested and the services registered.

In this research, a semantics-based web services infrastructure, called WSP, for semantic and user-oriented integration of protein data resources, is proposed. In WSP, protein data resources are modeled as reusable web services that provide protein features. In addition to the WSDL programmatic interface, each service has a semantic description which precisely describes the service's capabilities in terms of inputs, outputs and constraints. The semantic descriptions of web services are generated using the OWL-S upper service ontology and the EPO domain ontology. These semantic descriptions are then published in the WSP matchmaker for service discovery and integration. Rather than relying on syntactic matching (e.g., UDDI service discovery), the matchmaker discovers services based on their similarity to the service request. Therefore, users can locate services that semantically match their data requirements even if the services and the service requested are syntactically distinctive. In addition to service discovery, WSP supports a workflow-based approach for service integration. In this approach, an integration problem is represented as an abstract workflow of service requests, where each request is expected to be implemented by a web service. An integration agent is used to select and chain services, based on the criteria of service accuracy and data interoperability. The agent

finally generates an invokeable web services workflow, which automatically integrates the results from individual services.

A number of experiments were conducted to evaluate the performance of the WSP matchmaker. The parameters tested included solution space, accuracy and time efficiency. The results show that the size of solution space (number of matched services) is dependent on the size of the search space (number of registered services); the larger the search space, the larger the solution space. The solution space is larger for (service) requests without constraints (i.e., 2 concepts: input and output) than for (service) requests with constraints (i.e., 3 concepts: input, output and constraint). However, the average match score for both types of requests, without or with constraints, does not scale with the search space. This is also true for the accuracy of the matched services; the matching accuracy does not scale with the search space. The matchmaker can efficiently discover semantically matched services, and the matching time increases as the search space becomes larger. However, the percentage of time increase is relatively small which means the overheard of service matching is small.

A composite service, which integrates protein dynamics and conservation, is experimented using the WSP integration agent (i.e., service chaining algorithm). The experiment shows that the agent can select the most desirable (accurate) services and integrate the results (i.e., protein features) from selected services.

The contributions of this research are:

- An infrastructure for representing and correlating protein features at a higher semantic level. By exploiting the features of semantic-based web services, this infrastructure allows researchers to conveniently discover and assemble various types of protein data for their studies, e.g., determining the function or other features of proteins.
- Two biological web services that demonstrate the process of developing and using biological web services. The iGNM web service provides protein dynamics data for more than 20,000 protein structures. The N-gram web service provides conservation profiles for more than 50,000 protein sequences.
- A semantic matchmaker service that allows service providers to publish the description of their services and allows users to submit requests and obtain semantically matched services. The matchmaker includes an efficient semantic service matching algorithm.
- A service chaining algorithm that considers both service accuracy and data interoperability between services.

10.2. Conclusions

This research explores a semantic web services approach for protein data integration. Several conclusions can be drawn:

- It is feasible to integrate protein data resources (e.g., online databases and web servers) through web services. By modeling protein features (e.g., dynamics and conservation) as web

services, WSP allows users to conveniently share and retrieve protein data for many different applications.

- Using the OWL-S upper service ontology and the EPO domain ontology, it is feasible to generate a wide range of semantic descriptions of web services on complex protein data related issues. The semantic descriptions of protein web services precisely describe service capabilities (i.e., what a service can do) and user requests (i.e., what a user wants) in machine-understandable format so that algorithms can be designed to automatically discover and integrate semantically matched services.
- It is feasible to apply semantics-based service discovery mechanism for protein data resource integration. Semantic matching of web services' capabilities is an effective way to discover semantically matched services from users' own perspectives. Semantic matching is more robust than syntactic matching (e.g., UDDI service discovery), because the most accurate services can be discovered from all registered services even if the services and the service requested are syntactically distinctive.
- Workflow-based web services integration is an effective way to integrate protein data resources. An abstract workflow allows users to precisely define individual service requests and their data dependencies. Given an abstract workflow, service chaining algorithms can be designed to discover, select and chain services. Since web services are developed by different organizations and have heterogeneous interfaces, it is important to consider both semantics and data interoperability for service selection and integration.
- The proposed semantic web services approach (WSP) provides a more convenient environment for protein data resources integration than the current approach (i.e., web applications based integration): (1) With WSP, researchers only need to specify their service

requirements (i.e., desired output) without having to manually look for data resources. Using the web application approach, users have to write ad hoc scripts for different data resources. However, there is no duplication of effort with the WSP approach because the semantic matchmaker and the chaining algorithm can be applied to many different data resources; (2) The web application approach is fragile for a minor change in the HTML code of a given web application may cause failure. The WSP approach is less fragile to web site changes because data resources/services can be discovered and integrated dynamically.

10.3. Future work

Using semantic web services for bioinformatics is an emerging area for research. Many problems in this field remain to be studied. In light of this research, some topics that need further investigation are:

- *Granularity of web services*. In this research, protein features are identified as the atomic elements that can be shared by many protein-related problems. However, other bioinformatics topics (e.g., microarray data analysis) may require different methods to define the atomic elements. A valid design of granularity needs further investigation.

- *Ontology development*. In this research, the EPO ontology is developed by extending the Protein Ontology. To allow a wider range of applications, it is necessary to further evaluate the construct of the EPO ontology. Also, other proteomics ontologies need to be investigated and incorporated for richer descriptions of complex protein data related issues.

- *Semantic matching*. The semantic matching of web services' capabilities is dependent on the comparison of concepts in domain ontology. In this research, four relationships between concepts are adopted: "exact match", "plug-in match", "subsumption match" and "no match". However, other scoring methods could be studied and compared with the current scoring method.

- *Service chaining*. Service selection and integration is a complex problem. In this research, a workflow-based chaining algorithm is developed based on the service accuracy (semantics) and data interoperability. However, more complex algorithms can be designed to support various user requirements. For example, it is useful to develop service selection algorithms based on a combination of semantics, data interoperability and QoS parameters. In addition to workflow based approaches, it is useful to explore AI planning based approaches for service integration.

APPENDIX A

Time Complexity Analysis

WSP Publication Algorithm

Cost		Times	
	procedure register(service, G)		//service is a service description //G is an ontology; G = <V, E>
C1	1. registration = empty hash table	1	
C2	2. parse service into concepts c[m]	1	//m = number of concepts
C3	3. **for** i = 1 to m **do**	m+1	
C4	4. u_0 = the root vertex in G	m	
C5	5. DFS(u_0, c[i])	m	//depth-first traversal of G
C6	6. **return** registration	1	
	procedure DFS(u, c)		//u, c are concepts
C7	1. degreeOfMatch(u, c)	|V|	
C8	2. status[u] = "processed"	|V|	
C9	3. **for each** neighbor v of u **do**	|E|+|V|	
C10	4. **if** status[v] != "processed" **then**	|E|	
C11	5. DFS(v, c)	|V|	
C12	6. **return**	|V|	
	procedure degreeOfMatch(u, c)		//measure match between u and c
C13	1. **if** c = u **then**	1	
C14	2. service.score = "exact" or 3	1	
C15	3. **if** c is subclass of u **then**	1	
C16	4. service.score = "plugin" or 2	1	
C17	5. **if** c is superclass of u **then**	1	
C18	6. service.score = "subsumption" or 1	1	
C19	7. **if** service.score != null **then**	1	
C20	8. registration.add(u, service)	1	
C21	9. **return**	1	

Figure A-1. Pseudo code for the WSP service publication algorithm

A time complexity analysis for the algorithm was performed as follows:

Time = C1(1) + C2(1) + C3(m+1) + C4(m) + C5(m) + C6(1)

Cost = C1 + C2 + C3 + C3(m) + C4(m) + [C7(|V|) + C8(|V|) + C9(|E|+|V|) + C10(|E|) + C11(|V|) + C12(|V|)](m) + C6

= C1 + C2 + C3 + C3(m) + C4(m) + {[C13(1) + C14(1) + C15(1) + C16(1) + C17(1) + C18(1) + C19(1) + C20(1) + C21(1)]|V| + C8(|V|) + C9(|E|+|V|) + C10(|E|) + C11(|V|) + C12(|V|)}(m) + C6

= (C1+ C2 + C3 + C6) + (m)(C3 + C4) + m(|V|)(C13 + C14 + C15 + C16 + C17 + C18 + C19 + C20 + C21 + C8 + C11 + C12) + (m)(|E|+|V|)(C9) + (m)(|E|)(C10)

= C1'm + C2'(m)(|V|) + C3'(m)(|E|+|V|) + C4'(m)(|E|)

= O(m(|V|+|E|))

Note that "DFS" is a recursive function, which is called at each node of G. Therefore, line 1 and line 2 are visited |V| times. Line 3 is visited $\sum_{i=1}^{|V|}(neighbors(u_i)+1) = |E| + |V|$ times. Similarly, line 4 is visited $\sum_{i=1}^{|V|} neighbors(u_i) = |E|$ times. Line 5 and line 6 are visited |V| times each. Therefore, time complexity is O(m(|V| + |E|)), where |V| is the number of vertices in G, |E| is the number of edges, and m is number of concepts in service.

WSP Matching Algorithm

cost		Times	
	procedure match (request, registration)		//request is a service description //registration is an index table
C1	1. parse request into concepts c[m]	1	
C2	2. **var** service[], candidateList[m][]	1	
C3	3. **for** i = 1 to m **do**	m+1	//m = number of concepts
C4	4. candidateList[i] = services indexed by c[i]	m	//hash table search
C5	5. service = candidateList[1]	1	
C6	6. **for** i = 2 to m **do**	m	
C7	7. **for** j = 1 to N **do**	(m-1)(N+1)	//N = all registered services
C8	8. **if** service[j] != candidateList[i][j] **then**	(m-1)(N)	
C9	9. remove service[j]	(m-1)(N)	
C10	10. **for** k = 1 to length(services) **do**	n+1	
C11	11. **for** i = 1 to m **do**	n(m+1)	//n = matched services (n < N)
C12	12. service[k].match += service[k].score(c[i])	n(m)	
C13	13. insertion_sort(service)	1	//calculate total match score
C14	14. **return** service	1	//sort matched services
	procedure insertion_sort(A)		
C15	1. **for** j = 2 to length(A) **do**	n	//insert A[j] into the sorted list
C16	2. key = A[j]	n-1	A[1...j-1]
C17	3. i = j – 1	n-1	
C18	4. **while** i > 0 and A[i] > key **do**	$\sum_{j=2}^{n} t_j$	//tj = the number of times the while loop test for that value of j
C19	5. A[i+1] = A[i]	$\sum_{j=2}^{n}(t_j - 1)$	
C20	6. i = i – 1	$\sum_{j=2}^{n}(t_j - 1)$	
C21	7. A[i+1] = key	n-1	
C22	8. **return**	1	

Figure A-2. Pseudo code for the WSP service publication algorithm.

A time complexity analysis for the algorithm was performed as follows:

Time cost = C1(1) + C2(1) + C3(m+1) + C4(m) + C5(1) + C6(m) + C7(m-1)(N+1) + C8(m-1)(N) + C9(m-1)(N) + C10(n+1) + C11(n)(m+1) + C12(n)(m) + C13(1) + C14(1)

= (C1 + C2 + C3 + C5 + C14 – C7 + C10 + C14) + (m)(N)(C7 + C8 + C9 + C11 + C12) + N(–C7 – C8 – C9) + m(C3 + C4 + C6 + C7) + n(C10 + C11) + nm(C11 + C12) + C13

= C1'(m)(N) + C2'(N) + C3'(m) + C4'(n) + C5'(nm) + [C15(n) + C16(n-1) + C17(n-1) +

$$C18(\sum_{j=2}^{n} t_j) + C19(\sum_{j=2}^{n}(t_j - 1)) + C20(\sum_{j=2}^{n}(t_j - 1)) + C21(n-1) + C22(1) \ ]$$

$$= C1'(m)(N) + C2'(N) + C3'(m) + C4'(n) + C5'(nm) + C6'(\sum_{j=2}^{n} t_j) + C7'(\sum_{j=2}^{n}(t_j - 1))$$

Best case $= (m)(N) + [\ (n-1) + 0\ ]$

$= O(\ (m)(N)\ )$

Worst Case $= (m)(N) + [\ (\frac{n(n+1)}{2} - 1) + \frac{n(n-1)}{2}\ ]$

$= O(\ (m)(N) + n^2)$

As can be seen, the best case time complexity is O((m)(N)), the worst case time complexity is $O((m)(N) + n^2)$, where m is the number of concepts in request, N is the total number of registered services, and n is the number of matched services. The time complexity for this algorithm is lower than existing algorithms (see Section 6.2.3).

Typical Matching Algorithm

```
cost    procedure match(request, All, G)                    times   //G is an ontology, G = <V, E>
                                                                    //All is a list of all registered services
C1   1.   var service[ ]                                    1
C2   2.   for i = 1 to length(All) do                       N+1     //N = number of all services
C3   3.      match = serviceMatch(request, All[i])          N
C4   4.      if match != null then                          N
C5   5.         add All[i] to service                       N
C6   6.   insertion_sort(service)                           1
C7   7.   return service                                    1

         procedure serviceMatch(request, service)                   //compare a request with a service
C8   1.   parse request into concepts c1[m]                 1
C9   2.   parse service into concepts c2[m]                 1
C10  3.   for i = 1 to m do                                 m+1
C11  4.      u0 = the root vertex in G                      m
C12  5.      score[i] = DFS'(u0, c1[i], c2[i])              m
C13  6.      service.match += score[i]                      m       //depth-first search of service concept
C14  7.   return service.match                              1       //calculate match score

         procedure DFS'(u, x, y)                                    //x is request concept
                                                                    //y is corresponding service concept
C15  1.   if u = y then                                     |V|
C16  2.      score = degreeOfMatch(y, x)                    |V|
C17  3.      return score                                   |V|
C18  4.   else                                              |V|
C19  5.      status[u] = "processed"                        |V|
C20  6.      for each neighbor v of u do                    |E|+|V|
C21  7.         if status[v] != "processed" then            |E|
C22  8.            DFS'(v, x, y)                           |V|

         procedure degreeOfMatch(u, c)
C23  1.   if c = u then                                     1
C24  2.      score = "exact" or 3                           1
```

C25	3.	**if** c is subclass of u **then**	1	
C26	4.	score = "plugin" or 2	1	
C27	5.	**if** c is superclass of u **then**	1	
C28	6.	score = "subsumption" or 1	1	
C29	7.	**return** score	1	
		procedure insertion_sort(A)		
C30	1.	**for** j = 2 to length(A) **do**	n	//insert A[j] into the sorted list
C31	2.	key = A[j]	n-1	A[1...j-1]
C32	3.	i = j – 1	n-1	
C33	4.	**while** i > 0 and A[i] > key **do**	$\sum_{j=2}^{n} t_j$	//tj = the number of times the while loop test for that value of j
C34	5.	A[i+1] = A[i]	$\sum_{j=2}^{n} (t_j - 1)$	
C35	6.	i = i – 1	$\sum_{j=2}^{n} (t_j - 1)$	
C36	7.	A[i+1] = key	n-1	
C37	8.	**return**	1	

Figure A-3. Pseudo code for a typical service matching algorithm

A time complexity analysis for the algorithm was performed as follows:

Time Cost = $C1(1) + C2(N+1) + C3(N) + C4(N) + C5(N) + C6(1) + C7(1)$

$= (C1 + C2 + C7) + N(C2 + C4 + C5) + C3(N) + C6$

$= C1'(N) + (N)C3 + C6$

$= N + (N)[C8(1) + C9(1) + C10(m+1) + C11(m) + C12(m) + C13(m) + C14(1)] + [C30(n) + C31(n-1) + C32(n-1) + C33(\sum_{j=2}^{n} t_j) + C34(\sum_{j=2}^{n} (t_j - 1)) + C35(\sum_{j=2}^{n} (t_j - 1)) + C36(n-1) + C37(1)]$

$= N + (N)[(C8 + C9 + C10 + C14) + m(C10 + C11 + C13) + (m)C12] + [n + \sum_{j=2}^{n} t_j + \sum_{j=2}^{n} (t_j - 1)]$

$= N + (N)(m) + (N)(m)C12 + [\sum_{j=2}^{n} t_j + \sum_{j=2}^{n} (t_j - 1)]$

$= N + (N)(m) + (N)(m)[C15(|V|) + C16(|V|)[C23(1) + C24(1) + C25(1) + C26(1) + C27(1) + C28(1) + C29(1)] + C17(|V|) + C18(|V|) + C19(|V|) + C20(|E|+|V|) + C21(|E|) + C22(|V|)] + [\sum_{j=2}^{n} t_j + \sum_{j=2}^{n} (t_j - 1)]$

$= (N)(m) + (N)(m)(|V|+|E|) + [\sum_{j=2}^{n} t_j + \sum_{j=2}^{n} (t_j - 1)]$

Best case = $(N)(m)(|V|+|E|) + [n + (n-1) + 0]$

$= O((N)(m)(|V|+|E|))$

Worst case = $N + (N)(m)(|V|+|E|) + [n + (\frac{n(n+1)}{2} - 1) + \frac{n(n-1)}{2}]$

$= O((N)(m)(|V|+|E|) + n^2)$

The best case time complexity is O((N)(m)(|V|+|E|)), and the worst case time complexity is O((N)(m)(|V|+|E|) + n^2), where N is the total of registered services, m is the number of concepts in

service request, |V| is the number of vertices in G, |E| is the number of edges in G, and n is the number of matched services.

As can be seen, the matching between a service request and all the registered services is dependent on the size of the ontology and the number the registered services. When the number of the registered services increases or (and) the size of the ontology increases, the matching time also increases significantly.

WSP Service Chaining Algorithm

cost			times	

	procedure serviceIntegration(G, registration)			//G is an abstract workflow
				//G = <V, E>, where V is requests
				//and E is request dependencies

C1	1.	reqest$_0$ = the root vertex in G	1	
C2	2.	discovery(request$_0$, registration)	1	//perform service discovery
C3	3.	initialSelection(request$_0$)	1	//select services based on matching //score
C4	4.	finalSelection(request$_0$)	1	//further select services based on data
C5	5.	**return**	1	//interoperability
		procedure discovery(u, registration)		//depth-first traversal of G //u is a service request
C6	1.	status[u] = "processed"	|V|	
C7	2.	candidates[u] = match(u, registration)	|V|	
C8	3.	**for each** neighbor v of u **do**	|E|+|V|	
C9	4.	**if** status[v] != "processed" **then**	|E|	
C10	5.	discovery(v, registration)	|V|	
C11	6.	**return**	|V|	
		procedure initialSelection(u)		//depth-first traversal of G //u is a service request
C12	1.	status[u] = "processed"	|V|	
C13	2.	**if** candidates[u] = null **then**	|V|	//check if there are candidate services
C14	3.	**return**	|V|	
C15	4.	semiList[u] = top services in candidates[u]	|V|	//select services with highest score
C16	5.	**for each** neighbor v of u **do**	|E|+|V|	
C17	6.	**if** status[v] != "processed" **then**	|E|	
C18	7.	initialSelection(v)	|V|	
C19	8.	**return**	|V|	
		procedure finalSelection(u)		//depth-first traversal of G //u is a service request
C20	1.	status[u] = "processed"	|V|	
C21	2.	**if** u is root vertex in G **then**	|V|	
C22	3.	service[u] = first service in semiList[u]	|V|	
C23	4.	lastS = service[u]	|V|	
C24	5.	**for** i = 1 to m **do**	(m+1)(|V|)	//m is the number of services in
C25	6.	S = semiList[u][i]	(m)(|V|)	//semiList[u], m ranges from 1 to 5
C26	7.	**if** S.inputSchema = lastS.outputSchema **then**	(m)(|V|)	
C27	8.	service[u] = S break	(m)(|V|)	
C28	9.	**if** service[u] = null **then**	|V|	
C29	10.	service[u] = first service in semiList[u]	|V|	

C30	11.	**for each** neighbor v of u **do**	|E|+|V|
C31	12.	**if** status[v] != "processed" **then**	|E|
C32	13.	lastS = service[u]	|V|
C33	14.	finalSelection(v)	|V|
C34	15.	**return**	|V|

Figure A 4. Pseudo code for WSP service matching algorithm

A time complexity analysis for the algorithm was performed as follows:

Time = C1(1) + C2(1) + C3(1) + C4(1) + C5(1)

Cost = C6(|V|) + C7(|V|) + C8(|E|+|V|) + C9(|E|) + C10(|V|) + C11(|V|) + C12(|V|) + C13(|V|) + C14(|V|) + C15(|V|) + C16(|E|+|V|) + C17(|E|) + C18(|V|) + C19(|V|) + C20(|V|) + C21(|V|) + C22(V|) + C23(|V|) + C24((m+1)|V|) + C25(m|V|) + C26(m|V|) + C27(m|V|) + C28(|V|) + C29(|V|) + C30(|E|+|V|) + C31(|E|) + C32(|V|) + C33(|V|) + C34(|V|)

= O(|V| + |E|)

Time complexity is O(|V| + |E|), where |V| is the number of vertices in G, and |E| is the number of edges.

APPENDIX B

WSP Service Descriptions without Constraints

Service ID	Service Input	Service Output	Constraints
1	Protein	Dynamics	
2	Protein	Mode_Shape	
3	Protein	Fluctuation	
4	Protein	Trajectory	
5	Protein	Mobility	
6	Protein	Conservation	
7	Protein	Hydrophobicity	
8	Protein	Catalytic Sites	
9	Protein	Structure	
10	Protein	Sequential Parameters	
11	Protein	3D Parameters	
12	Protein	Ngram Patterns	
13	Protein	Family	
14	Protein	Chains	
15	Protein	Functional Domains	
16	Protein	Biological Function	
17	Protein	Structural Domains	
18	Protein	Source Cell	
19	Protein	Active Binding Sites	
20	Protein	Physiological Functions	
21	Protein	Pathological Functions	
22	Protein	Density	
23	Protein	Residues	
24	Protein	Atomic Bind	
25	Protein	Molecule	
26	Protein	Residue Link	
27	Protein	CISPeptide	
28	Protein	Description	
29	Protein	Reference	
30	Protein	Helix Structure	
31	Protein	Modified Residue	
32	Protein	Residue Sequence	
33	Protein	ATOM Sequence	
34	Structure	Dynamics	
35	Structure	Mode_Shape	
36	Structure	Fluctuation	
37	Structure	Trajectory	
38	Structure	Mobility	
39	Structure	Conservation	
40	Structure	Hydrophobicity	
41	Structure	Catalytic Sites	
42	Structure	Sequential Parameters	
43	Structure	3D Parameters	
44	Structure	Ngram Patterns	
45	Structure	Family	
46	Structure	Chains	
47	Structure	Functional Domains	

48	Structure	Biological Function	
49	Structure	Structural Domains	
50	Structure	Source Cell	
51	Structure	Active Binding Sites	
52	Structure	Physiological Functions	
53	Structure	Pathological Functions	
54	Structure	Density	
55	Structure	Residues	
56	Structure	Atomic Bind	
57	Structure	Molecule	
58	Structure	Residue Link	
59	Structure	CISPeptide	
60	Structure	Description	
61	Structure	Reference	
62	Structure	Helix Structure	
63	Structure	Modified Residue	
64	Structure	Residue Sequence	
65	Structure	ATOM Sequence	
66	PDB_ID	Dynamics	
67	PDB_ID	Mode_Shape	
68	PDB_ID	Fluctuation	
69	PDB_ID	Trajectory	
70	PDB_ID	Mobility	
71	PDB_ID	Conservation	
72	PDB_ID	Hydrophobicity	
73	PDB_ID	Catalytic Sites	
74	PDB_ID	Structure	
75	PDB_ID	Sequential Parameters	
76	PDB_ID	3D Parameters	
77	PDB_ID	Ngram Patterns	
78	PDB_ID	Family	
79	PDB_ID	Chains	
80	PDB_ID	Functional Domains	
81	PDB_ID	Biological Function	
82	PDB_ID	Structural Domains	
83	PDB_ID	Source Cell	
84	PDB_ID	Active Binding Sites	
85	PDB_ID	Physiological Functions	
86	PDB_ID	Pathological Functions	
87	PDB_ID	Density	
88	PDB_ID	Residues	
89	PDB_ID	Atomic Bind	
90	PDB_ID	Molecule	
91	PDB_ID	Residue Link	
92	PDB_ID	CISPeptide	
93	PDB_ID	Description	
94	PDB_ID	Reference	
95	PDB_ID	Helix Structure	
96	PDB_ID	Modified Residue	
97	PDB_ID	Residue Sequence	
98	PDB_ID	ATOM Sequence	
99	PDB_ID	Chemical Bonds	
100	PDB_ID	UnitCell	

WSP Service Descriptions with Constraints

Service ID	Service Input	Service Output	Constraints
101	Protein	Dynamics	Experimental
102	Protein	Dynamics	Computational
103	Protein	Dynamics	MD Simulation
104	Protein	Dynamics	NMA
105	Protein	Dynamics	Simplified NMA
106	Protein	Dynamics	Full-Atomic NMA
107	Protein	Mode_Shape	Residues
108	Protein	Mode_Shape	Atoms
109	Protein	Mode_Shape	NMA
110	Protein	Mode_Shape	Simplified NMA
111	Protein	Mode_Shape	Full-Atomic NMA
112	Protein	Fluctuation	Residues
113	Protein	Fluctuation	Atoms
114	Protein	Fluctuation	NMA
115	Protein	Fluctuation	MD Simulation
116	Protein	Mobility	Experimental
117	Protein	Mobility	Computational
118	Protein	Mobility	NMA
119	Protein	Mobility	MD Simulation
120	Protein	Conservation	Multiple Alignment
121	Protein	Conservation	Ngram
122	Protein	Catalytic Sites	Experimental
123	Protein	Catalytic Sites	Computational
124	Protein	Active Binding Sites	Experimental
125	Protein	Active Binding Sites	Computational
126	Protein	Hydrophobicity	Residues
127	Protein	Hydrophobicity	Atoms
128	Protein	Structure	Xray
129	Protein	Structure	NMR
130	Protein	Functional Domains	Experimental
131	Protein	Functional Domains	Computational
132	Protein	Structural Domains	Experimental
133	Protein	Structural Domains	Computational
134	Structure	Dynamics	Experimental
135	Structure	Dynamics	Computational
136	Structure	Dynamics	MD Simulation
137	Structure	Dynamics	NMA
138	Structure	Dynamics	Simplified NMA
139	Structure	Dynamics	Full-Atomic NMA
140	Structure	Mode_Shape	Residues
141	Structure	Mode_Shape	Atoms
142	Structure	Mode_Shape	NMA
143	Structure	Mode_Shape	Simplified NMA
144	Structure	Mode_Shape	Full-Atomic NMA
145	Structure	Fluctuation	Residues
146	Structure	Fluctuation	Atoms
147	Structure	Fluctuation	NMA
148	Structure	Fluctuation	MD Simulation
149	Structure	Mobility	Experimental

150	Structure	Mobility	Computational
151	Structure	Mobility	NMA
152	Structure	Mobility	MD Simulation
153	Structure	Catalytic Sites	Experimental
154	Structure	Catalytic Sites	Computational
155	Structure	Active Binding Sites	Experimental
156	Structure	Active Binding Sites	Computational
157	Structure	Hydrophobicity	Residues
158	Structure	Hydrophobicity	Atoms
159	Structure	Conservation	Multiple Alignment
160	Structure	Conservation	Ngram
161	Structure	Functional Domains	Experimental
162	Structure	Functional Domains	Computational
163	Structure	Structural Domains	Experimental
164	Structure	Structural Domains	Computational
165	PDB_ID	Dynamics	Experimental
166	PDB_ID	Dynamics	Computational
167	PDB_ID	Dynamics	MD Simulation
168	PDB_ID	Dynamics	NMA
169	PDB_ID	Dynamics	Simplified NMA
170	PDB_ID	Dynamics	Full-Atomic NMA
171	PDB_ID	Mode_Shape	Residues
172	PDB_ID	Mode_Shape	Atoms
173	PDB_ID	Mode_Shape	NMA
174	PDB_ID	Mode_Shape	Simplified NMA
175	PDB_ID	Mode_Shape	Full-Atomic NMA
176	PDB_ID	Fluctuation	Residues
177	PDB_ID	Fluctuation	Atoms
178	PDB_ID	Fluctuation	NMA
179	PDB_ID	Fluctuation	MD Simulation
180	PDB_ID	Mobility	Experimental
181	PDB_ID	Mobility	Computational
182	PDB_ID	Mobility	NMA
183	PDB_ID	Mobility	MD Simulation
184	PDB_ID	Catalytic Sites	Experimental
185	PDB_ID	Catalytic Sites	Computational
186	PDB_ID	Active Binding Sites	Experimental
187	PDB_ID	Active Binding Sites	Computational
188	PDB_ID	Hydrophobicity	Residues
189	PDB_ID	Hydrophobicity	Atoms
190	PDB_ID	Conservation	Multiple Alignment
191	PDB_ID	Conservation	Ngram
192	PDB_ID	Structure	Xray
193	PDB_ID	Structure	NMR
194	PDB_ID	Functional Domains	Experimental
195	PDB_ID	Functional Domains	Computational
196	PDB_ID	Structural Domains	Experimental
197	Chains	Structural Domains	Computational
198	Chains	Functional Domains	Experimental
199	Chains	Functional Domains	Computational
200	Chains	Structural Domains	Experimental

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J. (1990). Basic local alignment search tool. *J Mol Biol*, 215:403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25:3389-3402.

Ankolekar, A., Martin, D., McGuinness, D., McIlraith, S., Paolucci, M., and Parsia, B. (2004). OWL-S' Relationship to Selected Other Technologies, W3C Member Submission 22 November 2004. Accessed at http://www.w3.org/Submission/2004/SUBM-OWL-S-related-20041122/.

Bahar, I., Atilgan, A.R. and Erman, B. (1997). Direct evaluation of thermal fluctuations in protein using a single parameter harmonic potential. *Folding &* Design 2, 173-181.

Bahar, I., Wallqvist ,A., Covell, D.G., and Jernigan, R.L. (1998). Correlation between native state hydrogen exchange and cooperative residue fluctuations from a simple model. *Biochemistry* 37, 1067-1075.

Bairoch, A., et al. (2005) "The Universal Protein Resource (UniProt)." *Nucleic Acids Res*. 33, D154-D159.

Barrett, C.P. and Noble, M.E. (2005). Molecular motions of human cyclin dependent kinase 2. *J. Biol. Chem*., 280, 13993–14005.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2002). The Protein Data Bank. *Nucleic Acids Res.* 28, 235-242.

Berners-Lee, T. (1998). Semantic Web road map. Retrieved March 21, 2006 from http://www.w3.org/DesignIssues/semantic.html

Bertone, P., Kluger, Y., Lan, N., Zheng, D., Christendat, D., Yee, A., Edwards, A.M., Arrowsmith, C.H., Montelione, G.T. and GersteinM. (2001). SPINE: an integrated tracking database and data mining approach for identifying feasible targets in high-throughput structural proteomics, *Nucleic Acid Res.*, 29, (13), 2884-98.

Brittenham, P., Cubera, F., Ehnebuske, D., and Graham, S. (2001). Understanding WSDL in a UDDI registry: how to publish and find WSDL service descriptions. IBM developerWorks, http://www-128.ibm.com/developerworks/webservices/library/ws-wsdl/ .

Brooks,B. and Karplus,M. (1983) Harmonic dynamics of proteins: normal modes and fluctuations in bovine pancreatic trypsin inhibitor. *Proc. Natl. Acad. Sci. U.S.A*., 80, 6571–6575.

Califano, A., Stolovitzky, G., and Tu, Y. (2000). Analysis of gene expression microarrays for phenotype classification. *Proc. Int'l Conf. Intelligent Systems for Molecular Biology*, vol. 8, AAAI Press, 2000, pp.75-85.

Cao,Z.W. et al. (2004) MoViES: molecular vibrations evaluation server for analysis of fluctuational dynamics of proteins and nucleic acids. *Nucleic Acids Res*., 32, W679–W685.

Cardoso, J. and A. Sheth (2003). Semantic e-Workflow Composition. *Journal of Intelligent Information Systems* (JIIS).

Chandrasekaran, B., Johnson, T., and Benjamins, V. (1999). Ontologies: what are they? Why do we need them? *IEEE Intelligent Systems and Their Applications*, 14(1), 20-26.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web services description language 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

Constantinescu, I., Faltings, B., and Binder, W. (2004). Large scale, type-compatible service composition. In *Proceedings of the International Conference on Web Services (ICWS 2004),* 506-513.

Cruz, I.F. and Xiao, H. (2005). The role of ontologies in data integration. *Journal of Engineering Intelligent Systems*, 13(4), December 2005.

Dayhoff, M. O. (1976). The origin and evolution of protein superfamilies. *Fed.Proc*. 35.10: 2132-38.

Davidson, SB, Crabtree, J, Brunk, BP, Schug, J, Tannen, V, Overton, GC, Stoeckert, CJ. (2001). K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. IBM Systems Journal, 40(2).

Deshpande, N., Addess, K.J., Bluhm, W.F., Merino-Ott, J.C., Townsend-Merino, W., Zhang, Q., Knezevich, C., Chen, L., Feng, Z., Kramer Green, R., Flippen-Anderson, J.L., Westbrook, J., Berman H.M., and Bourne, P.E. (2005). The RCSB Protein Data Bank: A Redesigned Query System and Relational Database Based on the mmCIF Schema *Nucleic Acids Research*. 33: D233-D237.

Echols, N., Milburn, D., Gerstein, M. (2003). MolMovDB : analysis and visualization of conformational change and structural flexibility. *Nucleic Acids Res*. 31, 478-482.

Eckman,B.A., Lacroix,Z. and Raschid,L. (2001) Optimized seamless integration of biomolecular data. IEEE symposium on Bio-Informatics and Biomedical Engineering (BIBE'2001), Washington DC, nov 2001, p.23-32.

Finn, R. D., et al. (2006). "Pfam: clans, web tools and services." *Nucleic Acids Res*. 34, D247-D251.

Fogolari, F., Tessari,S. and Molinari, H. (2002). Singular value decomposition analysis of protein sequence alignment score data. *Proteins* 46.2: 161-70

Foster, I. (2005). Service-Oriented Science. *Science*, 308, pp 814-17.

Foster, I., Kesselman, C., Nick, J., Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Report.

Gao, H.T., Hayes, J.H., and Cai, H. (2005). Integrating biological research through Web services. *IEEE Computer*, 38(3):26–31, March 2005.

Garrett, R.H. and Grisham, C.M. (1999). Biochemistry, 2nd ed. Harcourt Brace College Publishing, Fort Worth.

Gene Ontology Consortium, http://www.geneontology.org/, 2004.

Go, N., Noguti, T. and Nishikawa, T. (1983) Dynamics of a small globular protein in terms of low-frequency vibrational modes, Proc. Natl. Acad. Sci. USA, 80, 3696.

Gold, N.D. and Jackson, R.M. (2006). SitesBase: a database for structure-based protein ligand binding site comparisons. *Nucleic Acids Research*, 34, D231-234.

Gómez-Pérez, A., González-Cabero, R., and Lama, M. (2004). A Framework for Design and Composing Semantic Web Services. *IEEE Intelligent Systems*, vol. 16, pp. 24–32.

Greenbaum, D., Smith, A. and Gerstein, M. (2005). Impediments to database interoperation: legal issues and security concerns. *Nucleic Acids Research*, 33:D3-D4, 2005.

Greenbaum, D.S. (2004) Dictionary of Bioinformatics and Computational Biology. Edited by Hancock and Zvelebil. Page 108.

Griffiths, A.J.F., Miller, J.H., Suzuki, D.T., Lewontin, R.C., Gelbart, W.M. (2002). An introduction to Genetic Analysis 7th ed., W.H. Freeman and Company, New York.

Gruber, T. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2), 199-220.

Gruber, T.R. (1995). Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies, 43(5-6):907-928.

Haarslev, V., Moller, R. (2001). RACER System Description. In: *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'2001),* Siena, Italy, Springer Verlag (2001) 701–705.

Haliloglu, T. and Bahar, I. (1999). Structure-based analysis of protein dynamics: Comparison of theoretical results for hen lysozyme with X-ray diffraction and NMR relaxation data. *Proteins: Structure, Function, & Genetics* 37, 654-667, 1999.

Hollup SM, Salensminde G, Reuter N. WEBnm@: a web application for normal mode analyses of proteins. BMC Bioinformatics. 2005 Mar 11;6(1):52.

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novere N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J; SBML Forum. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models Bioinformatics. 19, 524-31.

Hull, D., Stevens, R. and Lord, P. (2005). Describing web services for user-oriented retrieval. In *W3C Workshop on Frameworks for Semantics in Web Services, Digital Enterprise Research Institute (DERI), Innsbruck, Austria. June 9-10, 2005*, 2005.

IBM (2006). Web Services Overview. http://www.ibm.com/developerworks/webservices.

Jernigan, R.L. and Bahar, I. (1998). Coarse Grained Searches over Protein Conformations. *Encyclopedia of Computational Chemistry*. Schleyer, P. v.R.; Allinger, N. L.; Clark, T.; Gasteiger, J.; Kollman, P. A.; Schaefer III, H. F.; Schreiner, P. R. (Eds.) John Wiley & Sons: Chichester, 1998.

Johnson, M. S. and J. P. Overington. (1993). A structural basis for sequence comparisons. An evaluation of scoring methodologies. *J.Mol.Biol.* 233.4: 716-38.

Karchin, R. and R. Hughey. (1998). Weighting hidden Markov models for maximum discrimination. *Bioinformatics* 14.9: 772-82.

Kim, J., Gil, Y., and Spraragen, M. (2004). A knowledge-based approach to interactive workflow composition. In *Planning and Scheduling for Web and Grid Sciences workshop at the 14th International Conference on Automatic Planning and Scheduling (ICAPS 04),* Whistler, Canada.

Lacroix, Z, Boucelma, O, Essid, M. (2003) The biological integration system. In *Proceedings of the Fifth ACM CIKM International Workshop on Web Information and Data Management* (WIDM 2003), 45-49, New Orleans, Louisiana, USA.

Lan N, Montelione GT, Gerstein M. Ontologies for proteomics: towards a systematic definition of structure and function that scales to the genome level. Curr Opin Chem Biol. 2003 Feb; 7:44-54.

Lee,R.A., Razaz, M., Hayward, S. (2003). The DynDom Database of protein domain motions. *Bioinformatics* 19(10): 1290-1291.

Li, G., and Cui, Q. (2002). A coarse-grained normal mode approach for macromolecules: an efficient implementation and application to Ca(2+)-ATPase, *Biophys. J.*.83, 2457-2474.

Li, L., and Horrocks, I. (2003). Matchmarking Using an Instance Store: Some Preliminary Results. *Description Logics 2003*.

Liu, X., Karimi, H., Yang, L-.W. and Bahar, I. (2004). Protein Functional Motion Query and Visualization. *In IEEE 28th Annual International Computer Software and Applications Conference*.

Lord, P., Bechhofer, S., Wilkinson, M.D., Schiltz, G., Gessler, D., Hull, D., Goble, C., and Stein, L. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *International Semantic Web Conference*, pages 350-364, 2004.

Ludascher, B., Altintas, I., and Gupta, A. (2003). Compiling Abstract Scientific Workflows into Web Service Workflows. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*.

Mandell, D.J. and McIlraith, S.A. (2003). Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. *Proceedings of the Second International Semantic Web Conference* (ISWC2003), Sanibel Island, Florida.

Marcotte E. and Date S. (2001) Exploiting big biology: Integrating large-scale biological data for function inference. *Brief Bioinform.* 2: 363-374.

Marques, O. (1995). BLZPACK: Description and User's Guide, TR/PA/95/30. CERFACS, Toulouse, France.

Maximilien, E.M. and Singh, M.P. (2004). A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

McIlraith, S. and Son, T.C. (2002). Adapting golog for composition of semantic web services. In *Proceedings of the 8th International Conference on Knowledge Representation and Reasoning* (KR2002), 482-493.

McIlraith, S., Son, T.C. and Zeng, H. (2001). Semantic Web services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*. 16(2):46-53, March/April.

myGrid (2002). Comments on registry personalization and metadata.

Paolucci, M., T. Kawamura, T. R. Payne & K. Sycara (2002): *Semantic Matching of Web Service Capabilities*, in: Horrocks, I. & J. Hendler (eds.): 1st International Semantic Web Conference (ISWC2002).

Pearl, F. M., C. F. Bennett, et al. (2003). "The CATH database: an extended protein family resource for structural and functional genomics." Nucleic Acids Res 31(1): 452-455.

Press, W.H., et al. Numerical Recipes in Fortran: 2nd Ed., (1992) Cambridge University Press Chapter 2.6, pp 51–62.

Protégé (2002). *Protege Frequently Asked Question*. Accessed at http://portege.stanford.edu/faq.html

Rader, A.J. and Bahar, I. (2004) Folding core predictions from network models of proteins. *Polymer*, 45, 659–668.

Raghava, G. P., et al. (2003). OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics* 4 (2003): 47.

Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *Proceedings of First International Workshop on Semantic Web Services and Web Process Composition*.

Ritter, O., Kocab, P., Senger, M., Wolf, D. and Suhai, S. (1994). Prototype implementation of the integrated genomic database. *Comput. Biomed. Res.* 27, 97-115.

Rygg, A., Mann, S., Roe, P., and Wong, O. (2005) Bio-workflows with BizTalk: using a commercial workflow engine for eScience. In *Proceedings of the First International Conference on e-Science and Grid Computing* (e-Science'05).

Sahoo, S.S, Sheth, A.P., York, W.S., Miller, J.A. (2005). Semantic Web Services for N-glycosylation Process. *International Symposium on Web Services for Computational Biology and Bioinformatics*, VBI, Blacksburg, VA, May 26-27, 2005.

Sidhu, A.S., Dillon, T.S., and Chang, E. (2005). Creating a protein ontology resource. IEEE Computational Systems Bioinformatics Conference. Stanford University.

Singh, M.P. and Vouk, M.A. (1996). Scientific Workflows. Position paper in *Reference Papers of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions*, May 1996.

Sinha N, Smith-Gill SJ (2002). Protein structure to function via dynamics. *Protein and Peptide Letters*, 9: 367–377.

Sirin, E., Hendler, J. and Parsia, B. (2003). Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure* workshop in conjunction with ICEIS2003.

Snell, J. (2002). Implementing Web services with the WSTK v3.3. Part 1. *IBM Developer Works*, Dec. 2002, pp.5-6.

Spillner, J., Braun, I., and Schill, A. (2006). WSInterConnect: dynamic composition of web services through web services. In *Proceedings of the 6th International Conference on Distributed Applications and Interoperable Systems (DAIS 2006)*: 181-186.

Spitzner, J.H. Bioinformatic Sequence Markup Language, on-line, available at http://www.visualgenomics.com/bsml.

Staab, S., van der Aalst, W., Benjamins, V.R., Sheth, A., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., Gannon, D. (2003). Web services: been there, done that? *IEEE Intelligent Systems*, Volume: 18 , Issue: 1, Pages:72 – 85.

Stein, L. (2002). Creating a bioinformatics nation. *Nature*, 417, 119 - 120.

Stein, L. (2003). Integrating biological databases. *Nature Rev. Genet.*, 4: 337-345.

Suhre K. and Sanejouand, Y.H. (2004). ElNemo: a normal mode web-server for protein movement analysis and the generation of templates for molecular replacement. *Nucleic Acids Research*, 32, W610-W614.

Sycara, K., Paolucci, M., Ankolekar A. and Srinivasan, N. (2003). Automated Discovery, Interaction and Composition of Semantic Web services, Journal of Web Semantics, Volume 1, Issue 1, September 2003, pp. 27-46.

Tai, K., Murdock, S., Wu, B., Hong Ng, M., Johnston, S., Fangohr, H., Cox, S.J., Jeffreys, P., Essex, J.W., Sansom, M.S.P. (2004). BioSimGrid: towards a worldwide repository for biomolecular simulations. *Org. Biomol. Chem.* 2:3219–3221 DOI: 10.1039/b411352g.

Tai, S., Khalaf, R., and Mikalsen, T. (2004) Composition of coordinated web services. In *Middleware 2004*, pp. 294-310.

Tama, F., Gadea, F.X., Marques, O. & Sanejouand, Y.H. (2000) Building-block approach for determining low-frequency normal modes of macromolecules, *Proteins*, 41, 1.

Tirion, M.M. (1996) Large amplitude elastic motions in proteins from a single-parameter, atomic analysis, *Phys. Rev. Lett.,* **77**, 1905.

UDDI. The UDDI technical white paper. http://www.uddi.org, 2000.

US Census Bureau. North American industry classification system. http://www.census.gov/epcd/www/naics.html, 1997.

Uschold, M. and Jasper, R. (1999). A Framework for Understanding and Classifying Ontology Applications. In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving

Methods. V.R. Benjamins, B. Chandrasekaran, A. Gomez-Perez, N. Guarino, and M. Uschold (Ed.).

Valdar, W.S. and Thornton, J.M. (2001). Conservation helps to identify biologically relevant crystal contacts. *J.Mol.Biol.* 313.2: 399-416.

Vries, J.K., Munshi, R., Tobi, D., Klein-Seetharaman, J., Benos, P.V., and Bahar, I. (2004). A sequence algnment-independent method for protein classification. Applied Bioinformatics, 2004, 3(2-3): 137-148.

Vries, J.K., Liu, X., and Bahar, I. (2006a). Invariant Conservation Signatures for Proteins based on N-gram Patterns", to be submitted to a referred journal.

Vries, J.K., Liu, X., and Bahar, I. (2006b). The Relationship between N-Gram Patterns and Secondary Structure, submitted to *Proteins: Structure, Function and Bioinformatics*.

W3C OWL. (2004). http://www.w3.org/TR/owl-guide/.

W3C OWL-S (2004). http://www.w3.org/Submission/2004/07/, 2004

W3C RDF. (2004). http://www.w3.org/TR/rdf-schema/.

W3C Semantic Web. (2006). http://www.w3.org/2001/sw/

W3C SOAP. (2001). http://www.w3.org/TR/soap/

Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S. (2001). Ontology-Based Integration of Information - A Survey of Existing Approaches. In *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing*, 2001.

Wako, H., Kato, M and Endo, S. (2004). ProMode: a database of normal mode analyses on protein molecules with a full-atom model. *Bioinformatics*, 20, 2035-2043.

Wheeler, D.L., et al. (2003). Databases resources of the National Center for Biotechnology. *Nucleic Acids Research*, vol. 31, no.1, 2003, pp. 28-33.

Wootton, J. C. and S. Federhen. (1996). Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.* 266: 554-71.

Yang, J. (2003). Web Service Componentization: Towards Service Reuse and Specialization. *Communications of ACM*, October, 2003.

Yang, L.W., Liu, X., Jursa, C.J., Holliman, M., Karimi, H.A., and Bahar, I. iGNM: A Database of Protein Functional Motions Based on Gaussian Network Model. *Bioinformatics*, pp. 2978-2987, Vol. 21, No. 13, 2005.

Yang, L.W., Rader, AJ, Liu, X., Jursa, C.J., Chen, S.C., Karimi, H.A., and Bahar, I. (2006). oGNM: Online Computation of Structural Dynamics Using the Gaussian Network Model, *Nucleic Acids Research*, 34, W24-31, 2006.

Zdobnov, E.M., Lopez, R., Apweiler, R. and Etzold, T. (2002). The EBI SRS server – new features. *Bioinformatics* 18, 1149-1150.

Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transaction on Software Engineering*, Vol. 30, No. 5, May 2004.