D580–D589 *Nucleic Acids Research, 2007, Vol. 35, Database issue*
doi:10.1093/nar/gkl836

SNAPPI-DB: a database and API of Structures, iNterfaces and Alignments for Protein–Protein Interactions

Emily R. Jefferson, Thomas P. Walsh, Timothy J. Roberts[1] and Geoffrey J. Barton*

School of Life Sciences Research, College of Life Sciences, University of Dundee, Dow Street, Dundee, DD1 5EH, Scotland, UK and [1]School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK

Received August 16, 2006; Revised October 4, 2006; Accepted October 6, 2006

ABSTRACT

SNAPPI-DB, a high performance database of Structures, iNterfaces and Alignments of Protein–Protein Interactions, and its associated Java Application Programming Interface (API) is described. SNAPPI-DB contains structural data, down to the level of atom co-ordinates, for each structure in the Protein Data Bank (PDB) together with associated data including SCOP, CATH, Pfam, SWISSPROT, InterPro, GO terms, Protein Quaternary Structures (PQS) and secondary structure information. Domain–domain interactions are stored for multiple domain definitions and are classified by their Superfamily/Family pair and interaction interface. Each set of classified domain–domain interactions has an associated multiple structure alignment for each partner. The API facilitates data access via PDB entries, domains and domain–domain interactions. Rapid development, fast database access and the ability to perform advanced queries without the requirement for complex SQL statements are provided via an object oriented database and the Java Data Objects (JDO) API. SNAPPI-DB contains many features which are not available in other databases of structural protein–protein interactions. It has been applied in three studies on the properties of protein–protein interactions and is currently being employed to train a protein–protein interaction predictor and a functional residue predictor. The database, API and manual are available for download at: http://www.compbio.dundee.ac.uk/SNAPPI/downloads.jsp.

INTRODUCTION

Protein–protein interactions are fundamental to understanding biological networks and cellular processes. Accordingly, many experimental (1–3) and computational (4–10) techniques have been developed to probe and predict interacting protein partners. There are several databases of protein interactions which store the information generated from high throughput experimental methods and literature curation, for example, GRID (11), the IntAct Project (12), BIND (13), MINT (14), DIP (15–17) and the HPRD (18). STRING (19) also contains data derived from database and literature mining and high-throughput experimental data, but in addition contains predictions based on genomic context analysis.

These computational and experimental techniques can yield significant information about possible interactions but they do not provide information about the structure of the interfaces at the atomic level. High-resolution X-ray and NMR structures can provide an atomic level of detail and have therefore been utilised for both investigation and prediction of protein–protein interactions. Analyses of interaction sites from 3-D structures have identified a number of properties that distinguish interaction sites from other areas of protein surfaces, including: residue conservation across species; a tendency to be polar, uncharged and hydrophobic; a planar protruding shape and a higher solvent accessible area (20–25). These properties have been exploited to predict interaction surfaces on protein structures (26–28).

Predictions of protein–protein interactions using structural data have been based on the hypothesis that if two proteins are seen to interact in a known 3-D structure, their homologues will interact in a similar fashion (29,30). A multimeric threading method has been used to extend this approach to distantly related homologous and analogous pairs (31). Structural data for interfaces has also been used to create templates that capture the essential features of interactions sites and which are employed to screen protein structures for the presence of interaction sites (32). Methods of protein–protein interaction prediction have been extensively reviewed by Szilagyi *et al.* (33).

The advantages of structural data have motivated the creation of several databases of protein-protein interactions and interfaces including 3did (database of 3-D interacting domains) (34), PIBASE (structurally defined protein interfaces) (35), SCOPPI (a structural classification of

*To whom correspondence should be addressed. Tel: +44 01382 385860; Fax: +44 01382 385764; Email: geoff@compbio.dundee.ac.uk protein–protein interfaces) (36), PSIBase (Protein Structural Interactome map) (37) and PRISM (PRotein Interactions by Structural Matching) (38).

In this paper, a system is presented which provides a foundation for analysis and prediction of structural data with an emphasis on domain–domain interactions. This system consists of SNAPPI-DB, a database of Structures, iNterfaces and Alignments of Protein–Protein Interactions, and its associated Application Programming Interface (API). SNAPPI-DB, a high performance, object oriented database provides consistent, enhanced quality structural data, enriched with additional data such as multiple domain classifications, quaternary structures and domain-domain interactions. The API facilitates rapid development, is extensible, allows easy access to the data and circumvents the need to write complex SQL queries.

The contents and creation of SNAPPI-DB are discussed, followed by an overview of the API. The system is then compared to other databases of protein–protein interactions observed in structural data. Finally, the unique features of the system and its applications are discussed.

CONTENT AND CREATION

SNAPPI-DB is currently a 38 GB database containing 31 136 Protein Data Bank (PDB) structures and associated data including:

- Atomic level PDB data (39,40).
- SCOP (41–43), CATH (44–46) and Pfam (47–49) domains.
- Domains classified to different levels of similarity based on the SCOP and CATH hierarchical classification system and Family level for Pfam.
- Domain–domain interactions determined for SCOP, CATH and Pfam domains down to the level of which atoms interact.
- Domain–domain interactions classified to different levels of similarity based on the SCOP and CATH hierarchical classification system and Family level for Pfam.
- Domain–domain interactions classified by their interacting interfaces (orientation in which the domains are interacting).
- Multiple structural alignments of domain interactions from each Family/Superfamily pair for each unique orientation for SCOP, CATH and Pfam domain definitions.
- Biological units from Protein Quaternary Structures (PQS) (50).
- Unique identifiers to link to the MSD data warehouse (51,52).
- Interpro (53) regions/domains.
- GO (54) terms.
- SWISSPROT (55) identifiers and numbering.

MSD as the data source

The macromolecular structure database (MSD) (51,52) developed at the European Bioinformatics Institute (EBI) was chosen as the raw data source for SNAPPI-DB as it contains the complete contents of the PDB (39,40) together with substantial complementary data pertaining to domains, functional sites, protein families and sequences from other databases. These data include SCOP (41–43), CATH (44–46), Pfam (47–49), SWISSPROT (55), InterPro (53), GO terms (54), PQS (50), secondary structure information and detailed ligand properties. In addition, a key feature of the MSD is the improved consistency compared to PDB flat files.

The design of SNAPPI-DB

The MSD is an extremely useful database that has been key to the development of SNAPPI-DB; however, it was found that the speed of the MSD was not sufficient for high performance analysis at the atom level. In addition, the structure of the MSD is not optimised for analysis at the level of domains and does not contain the additional information stored by SNAPPI-DB on domain–domain interactions and structural multiple alignments of domain–domain interactions classified by their interface.

In order to increase performance and to allow complex analysis with a high degree of abstraction the data relevant to SNAPPI-DB were migrated to an object-oriented database developed with Java Data Objects (JDO) technology. JDO is a persistence framework for the Java language that allows the storage, retrieval and querying of objects. SNAPPI-DB employs the FastObjects community edition JDO implementation (http://www.versant.net/index.html). The application of JDO to storing biological has been described by Srdanovic *et al*. (56).

In essence, JDO provides an automatic mapping between a data-store and Java objects. This approach has many benefits. Firstly, JDO reduces development time as performing complex queries using this technology is easier than accessing a relational database directly via SQL. Secondly, employing JDO removes the complications inherent in mapping objects to a relational database, a difficulty commonly known as the 'object-relational impedance mismatch' problem (57). This feature greatly facilitates handling of biological data which often fits the object model. Finally, the JDO specification is intentionally data-store agnostic and so the JDO interface is the same regardless of the database back-end. Possible data-stores include relational databases, object databases and XML files. The choice of data-store will depend upon the user requirements. For example, a relational database is preferable if queries are to be performed by another application. In the case of high performance data mining an object-oriented data-store has many advantages over other data-store mechanisms such as lack of SQL overhead, speed, easy storage of polymorphic entities and direct two way references (56) and hence was choosen for SNAPPI-DB. However, if required the data could be ported to a relational database and the same API used.

Generation of domain–domain Interactions

Multiple domain definitions. Analysis of protein–protein interactions is usually performed at the level of domains rather than proteins since domains are often considered to be the fundamental functional and structural units. Domains can be assigned differently depending on the domain definition. As some domains in one domain classification do not have corresponding domains in another classification, some domain–domain interactions may not be found if only one of the classifications is used. SNAPPI-DB was employed to analyse the increase in the number of non-redundant domain–domain interactions provided by employing both SCOP and CATH domain definitions. It was observed that the use of both CATH and SCOP domain assignments increases the number of non-redundant interacting domains by between 23.6 and 37.3%. Therefore, it is advantageous to employ both sets of domain classifications simultaneously to investigate interactions at the domain level. Accordingly, both SCOP and CATH domain definitions are included in SNAPPI-DB. Pfam (47–49) was also included as this is a widely used sequence based domain definition which may be utilised to link to proteins which do not have a solved structure.

In addition to these three domain definitions, InterPro (53) domains, GO terms (54) and SWISSPROT (55) indexes are all stored.

Probable quaternary structures. The coordinates that appear in PDB files are those of the asymmetric unit (ASU) which is the fraction of the crystallographic unit cell that has no crystallographic symmetry. This may not be the biologically relevant unit of structure, and so may lack some key protein–protein interactions. In addition, some of the interactions seen in the ASU of the crystal may be artefacts of crystallisation and may not be biologically relevant (58,59).

There are two main sources of quaternary states: the state suggested by the authors of the structure (for all structures deposited since 1999) and computer predictions which apply all relevant symmetry operations and then discriminate between crystal packing artefacts and likely functional protein–protein interactions. The true quaternary state of a complex is not always straightforward to determine and errors are made by both the authors of the structure and *in silico* predictive methods. PQS (50) was chosen as the source of quaternary structure since although the initial assignments of biological units made by PQS are done by a computer program, they are hand-curated for each structure and errors and inconsistencies in the PQS database are corrected and updates made continuously.

Previously, SNAPPI-DB was used to investigate the additional non-redundant domain–domain interaction interfaces which are observed in the PQS predicted biological units in comparison to the ASUs seen in PDB files (60). It was determined that using PQS instead of the PDB increases the number of additional non-redundant interaction interfaces observed in structural data by 34.5% (1455 interfaces). PQS also removes 2981 interactions from the data set which it classifies as crystal packing artefacts. Therefore, the domain–domain interactions used in SNAPPI-DB are those observed in PQS biological units instead of the ASUs. The interactions which are seen in ASUs and not considered valid by PQS are also available to search in the database if required.

Defining an interaction. Interactions between domains are determined based on distance. Atoms are considered to interact if the distance between them was less than the sum of their van der Waals radii (61) +0.5 Å. Two domains are considered to be interacting if there are $\geqslant 10$ interacting residue pairs between the domains. The threshold of 10 residues was chosen based on inspection of interaction sites and study of relevant literature. However, since an object-oriented design is adopted this behaviour can be over-ridden by users. For example, the distance based measure could be replaced by solvent accessible area.

Clustering of interactions by Family/Superfamily and interface

When analysing domain–domain interactions it is often necessary to classify them into pairwise Families/Superfamilies (e.g. in order to deal with redundancy in structural data). The term ‘pairwise Family’ is used to describe the classification of a domain–domain interaction based on the Family classification of each of the interacting domains. Similarly, the term ‘pairwise Superfamily’ is used to describe the classification of a domain–domain interaction based on the Superfamily classification of each of the interacting domains. The database contains domain–domain interactions classified to different levels of similarity (from Class through to Family) based on the SCOP and CATH hierarchical classification system and Family level for Pfam. The first step in Figure 1 shows an example of this process for the SCOP domain classification system to the Superfamily level of similarity.

SNAPPI-DB not only classifies interactions by their domain classification but also by the interface with which they are contacting. The second step in Figure 1 shows an example of classification of domain–domain interactions by their interface.

Method of clustering by different interaction interfaces. In order to classify interactions by their interaction interface, the relative orientation of the interacting pair was determined using an implementation of the iRMSD (interaction root–mean–square deviation) method described by Aloy *et al.* (62). This method determines if two interacting domains are at the same orientation as another pair of interacting domains and thus if they are interacting with the same interfaces. An iRMSD cut-off of <10 Å was applied to distinguish interactions between pairs that have a similar orientation and those that do not.

Structural alignments and positions of interacting residues. Once the interactions are classified by Superfamily/Family and by orientation, structural alignments are generated by STAMP (63). The structural alignments are used to generate a pair of alignments for each set of classified interactions as shown in the final step in Figure 1. SNAPPI-DB contains each of these alignments, the positions of the interacting residues at the surface as well as the transformation matrix required for superposition.

Generation of the data

The downloadable version of SNAPPI-DB contains the data generated after all of these steps have been performed. However, should the user have a local copy of the MSD Oracle relational database they may wish to generate SNAPPI-DB themselves. The system is designed so that this can be done in four easy steps each of which can be customised to allow flexibility.

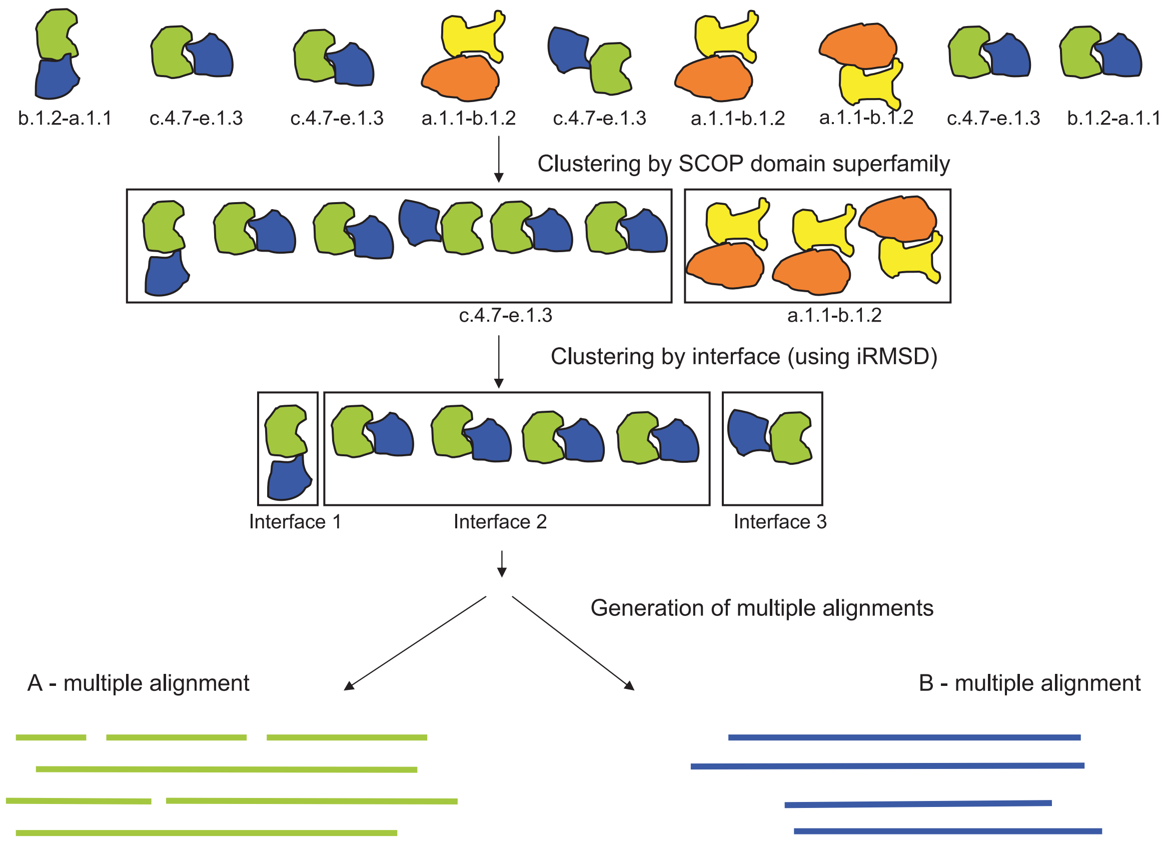

Figure 1. The classification of domain–domain interactions. The first step shows clustering by pairwise SCOP domain classification system at the Superfamily level of similarity. Clustering is also performed for the CATH and Pfam classification systems and at all levels of similarity. The second step shows an example of classification of domain–domain interactions by their interface. This classification is determined by the relative orientation of the interacting pair using an implementation of the iRMSD (interaction root-mean square deviation) method described by Aloy *et al.* (62). The final step shows the generation of a pair of multiple structure alignments, one from each partner of the interaction. These alignments are generated by STAMP (63).

THE APPLICATION PROGRAMMING INTERFACE

To efficiently deal with the complex and varied nature of the data contained in SNAPPI-DB an easy to use Java API has been developed. The API enables rapid development at a high level of abstraction without any requirement for complex SQL queries. In particular, specific design attention was paid in providing a natural model of protein–protein interaction data and the way bioinformaticians analyse such data. For example, as most analysis is done on the level of domains rather than chains, navigation via domains and domain–domain interactions is as seamless as via PDB entries, and dealing with redundancy in structural data is a core component of the API.

Java 5 is employed since it provides many features that are not available in previous versions of Java such as generics, enhanced 'for' loops and autoboxing/unboxing. The same naming convention as the MSD is employed so that users familiar with the MSD can rapidly learn the SNAPPI API. The same unique identifiers are also used so that structures can be mapped back to the MSD.

Figure 2 shows a simplified UML diagram of the structure of the API. As the database is object-oriented there is no difference between the relationships of the objects seen in the UML diagram of the API and the way that the objects are stored in the database. Although there are many different ways of navigating through the data in SNAPPI-DB, the database is optimised for searching either from (i) 'Domains' class: single domains classified by Family, (ii) 'DomainInteractions' class: domain–domain interactions classified by Family pair, (iii) 'OrientationSimilarInteractions' class: domain–domain interactions classified by orientation of interaction and (iv) 'Entries' class: collection of structures. These four key ways to access the data can be seen at the top of the UML diagram and are summarised below along with pseudo-code.

Navigation via Domains

The 'Domains' singleton (sole instance of a class) contains a list of domains classified by their domain Family. Domains can be easily accessed by their domain classification to any level of the domain hierarchy for SCOP and CATH and at the Family level for Pfam. For example, for the SCOP domain definition at the Family level of similarity there is a map which stores the name of the SCOP Family (e.g. a.1.1.1) as the key and a list of all of the domains with this classification as the value. In the pseudo-code in Figure 3, the (Scop.-class, 4) is used to denote the Family level of similarity in the SCOP hierarchy. If this analysis was performed at the superfamily level (e.g. a.1.1) then this number would be 3.

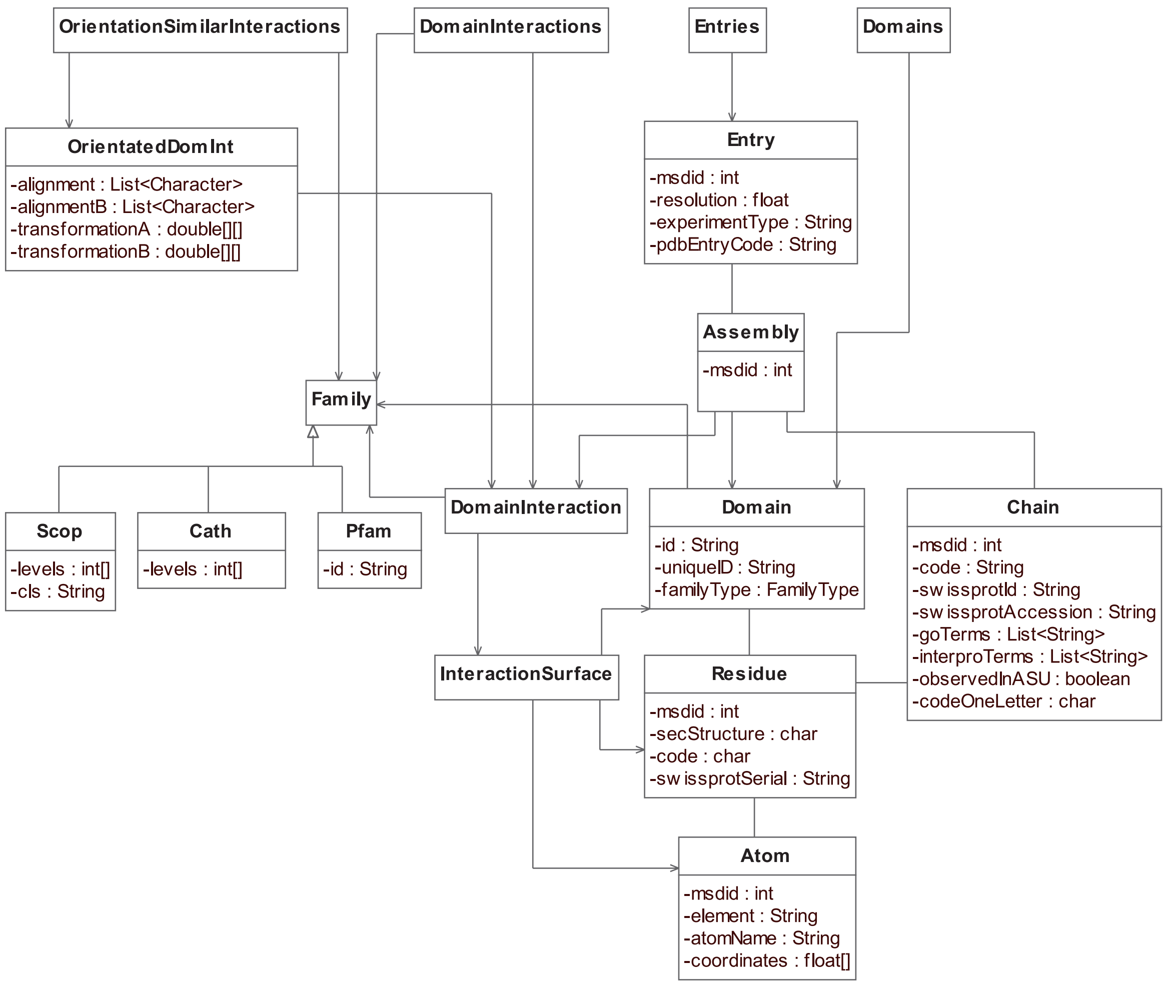

Figure 2. Simplified UML diagram of the structure of the SNAPPI-DB API. There are four main points of entry to the data in SNAPPI-DB: (i) 'Domains' class: single domains classified by Family, (ii) 'DomainInteractions' class: domain–domain interactions classified by Family pair, (iii) 'OrientationSimilarInteractions' class: domain–domain interactions classified by orientation of interaction and (iv) 'Entries' class: collection of structures.

One of the key aspects when working with structural data is the problem of redundancy within the data set. The API deals with this explicitly by having the option to select only one domain from a set of domains classified to any level of the domain hierarchy. For example, if the similarity of the domains was set to be the SCOP Family level of similarity then the API would return one domain to represent each Family. If a more stringent level of similarity was set, for example, at the Superfamily level of similarity, then the API would return one domain to represent each Superfamily. Which domain is selected to represent the group is determined by a (user overridable) strategy object passed in. Implementations could, for example, return the highest resolution domain or a randomly selected domain.

Navigation via DomainInteractions

The 'DomainInteractions' singleton contains a list of domain–domain interactions grouped by their pairwise domain classification. Each pair of interacting domains can be accessed by their pairwise domain classification to any level of the domain hierarchy for SCOP and CATH and at the Family level for Pfam. For example, for the SCOP domain definition at the Family level of similarity there is a map which stores the name of the pairwise SCOP Family (e.g. a.1.1.1–b.1.2.3) as the key and a list of all of the domain interactions with this classification as the value. In a similar way to Domains, a non-redundant set of domain–domain interactions can easily be generated. Example psuedo-code is shown in Figure 4.

Navigation via OrientatedDomInts

The 'OrientationSimilarInteractions' singleton contains a list of domain–domain interactions classified by their interface orientation. In a similar way to the DomainInteractions above, each domain–domain interaction is classified by their Family pair but in addition to this they are then further classified by the orientation of the interaction giving a collection of lists of domain–domain interactions for each pairwise Family. For example, there is a map with pairwise SCOP Family (e.g. a.1.1.1–b.1.4.7) as the key and a collection of lists of all of the domain–domain interactions with this pairwise Family classification and classified by orientation as the value. Rather than storing DomainInteraction objects in these lists OrientatedDomInt objects are stored. An OrientatedDomInt

Pseudo-code

```
//This gets the Domains classified by their Scop family (denoted by 4) class
Map<Family,Collection<Domain>>domainsHashedByFamily
                    =Domains.getDomainsHashedByFamily(Scop.class, 4);
for (Map.Entry<Family, Collection<Domain>> map : domainsHashedByFamily.entrySet())
{
   //The Domain Family Classification is obtained by map.getKey() e.g. scop family a.1.2.3,
   System.out.println("Scop Family = " + map.getKey());

   //This iterates through all the Domains with the same family classification
   for (Domain domain : map.getValue())
   {
      //This iterates through all of the Residues within a Domain
      for (Residue r : c.getResidues())
      {
         //This iterates through all of the Atoms within a Residue
         for (Atom a : r.getAtoms())
         {
            //This gets the coordinates for an Atom
            float[] coordinates = a.getCoordinates();
            //This prints out the coordinates for an Atom
            System.out.println(coordinates[0] + "," + coordinates[1] + "," + coordinates[2]);
         }
      }
   }
}
```

Figure 3. Psuedo-code for navigation via Domains.

Pseudo-code

```
//This gets the Domain-Domain Interactions classified by their Scop family pair class
Map<Pair<Family>,Collection<DomainInteraction>>domIntsHashedByFamilyPair
                    =DomainInteractions.getDomainInteractionsHashedByFamilyPair(Scop.class, 4)
for (Map.Entry<Pair<Family>,Collection<DomainInteraction>> map : domIntsHashedByFamilyPair.entrySet())
{
   /*
   The domain interaction family classification is obtained by map.getKey()
   e.g. scop pairwise family a.1.2.3 interacting
   with b.1.4.7. The print statement below would give "Scop pairwise family = a.1.2.3,b.1.4.7"
   */
   System.out.println("Scop pairwise family = " + map.getKey());

   //This iterates through all the Domain Interactions with the same family classification
   for (DomainInteraction domainInteraction : map.getValue())
   {
      //Do Something
   }
}
```

Figure 4. Psuedo-code for navigation via DomainInteractions.

contains a DomainInteraction and additional information regarding the transform and alignment of the DomainInteraction. Example psuedo-code is shown in Figure 5.

Navigation via Entries

The 'Entries' singleton contains a list of PDB Entries. Navigation through each PDB Entry is straightforward as the data are stored in a hierarchal structure as shown in Figure 2. Each Entry contains one or more Assemblies (PQS predicted structures), each Assembly contains one or more Chains. Each Chain contains one or more Residues and each Residue contains one or more Atoms. Each level of the hierarchy also contains other information relevant to the item. For example, each Atom contains the co-ordinate positions of the Atom. The Assemblies also contain domains and domain interactions of SCOP, CATH and Pfam. Example psuedo-code is shown in Figure 6.

Pseudo-code

```
//This gets the Domain-Domain Interactions classified by their pairwise Scop Family class and orientation
Map<Pair<Family>, Collection<Collection<OrientatedDomInt>>> domainInteractionsHashedByFamilyPair
                = OrientationSimilarInteractions.getDomainInteractionsHashedByFamilyPair(Scop.class, 4);
for (Map.Entry<Pair<Family>, Collection<Collection<OrientatedDomInt>>> mapEntry :
domainInteractionsHashedByFamilyPair.entrySet())
{
    System.out.println("mapEntry.getKey() = " + mapEntry.getKey());
    for (Collection<OrientatedDomInt> orientatedDomInts : mapEntry.getValue())
    {
        for (OrientatedDomInt orientatedDomInt : orientatedDomInts)
        {
            System.out.println("");
            for (Character character : orientatedDomInt.getAlignmentA())
            {
                System.out.print(character);
            }
            System.out.println("");
            for (Character character : orientatedDomInt.getAlignmentB())
            {
                System.out.print(character);
            }
            System.out.println("");
        }
    }
}
```

Figure 5. Psuedo-code for navigation via OrientatedDomInts.

Pseudo-code

```
//This iterates through all of the PDB entries stored in the database
for (Entry e : Entries.getEntries())
{
    //This iterates through all of the PQS Assemblies within an Entry
    for (Assembly ass : e.getAssemblies())
    {
        //This iterates through all of the Chains within an Assembly
        for (Chain c : ass.getChains())
        {
            //This iterates through all of the Residues within a Chain
            for (Residue r : c.getResidues())
            {
                //This iterates through all of the Atoms within a Residue
                for (Atom a : r.getAtoms())
                {
                    //This gets the coordinates for an Atom
                    float[] coordinates = a.getCoordinates();
                    //This prints out the coordinates for an Atom
                    System.out.println(coordinates[0] + "," + coordinates[1] + "," + coordinates[2]);
                }
            }
        }
    }
}
```

Figure 6. Psuedo-code for navigation via Entries.

UTILITY AND DISCUSSION

Databases of structural domain–domain interactions

When the development of SNAPPI-DB began there were no extensive domain–domain interaction databases based on structural data; however, recently, there have been several databases made available. The fact that so many have been developed shows the timely aspect of investigation of domain–domain interactions using structural data. SNAPPI-DB and its associated API has several features which set it apart from other databases.

SNAPPI-DB contains different forms of derived data, including multiple domain definitions, PQS, GO terms, Interpro, and SWISSPROT and secondary structure information. The database uses the same unique identifiers as the full MSD data warehouse and so extra information required which is contained within the MSD but is not in SNAPPI-DB can easily be obtained. SNAPPI-DB classifies interactions based on the different interfaces with which they interact and provides information about which residues and atoms are in contact. A key advantage of SNAPPI-DB is that each set of classified domain–domain interactions has an associated multiple structural alignment for each partner. These alignments can be used for many tasks such as analysis of conservation patterns for domain–domain interactions or to train protein–protein interaction predictors.

The use of the JDO technology has several advantages (as discussed in depth in the 'Construction' Section). One important advantage is that the JDO interface is data store agnostic and so the database could be stored as either a relational database or an object-oriented database. Another key advantage is the high performance nature of SNAPPI-DB. Srdanovic *et al.* (56) found that the Fast Object Community Edition implementation of JDO had faster performance than the relational database implementation. The authors of the PSIBase system (37) developed a new algorithm (64) for determining interacting domains at the atomic level on the grounds that this task would take months using existing methods. In contrast, determining the interacting domains at the atomic level for 31 136 structures takes $\sim$3 h on a standard desktop machine (3 GHz PIV, 1 GB RAM) using SNAPPI-DB.

The main advantage of the system comes when the database is used in conjunction with the API. As far as we are aware no other databases come with an associated API and therefore they can not be used at the higher level of abstraction that is provided by this system.

Applicability

SNAPPI-DB has been employed in three different investigations already: an investigation into biological units and their effect upon the properties and prediction of protein–protein interactions (60), a comparison of comparison of SCOP and CATH with respect to domain–domain interactions and investigation into the orientation at which proteins interact. In addition SNAPPI-DB is currently being used to train both a functional residue predictor and a protein–protein interaction predictor. These methods demonstrate the wide applicability of this system in investigations and predictions of protein–protein interactions using structural data.

FUTURE DEVELOPMENTS

It is intended to expand SNAPPI-DB in several ways. At the moment it is not easy to update SNAPPI-DB to take into account new entries being added to the MSD. This problem will be resolved by providing update scripts and by offering multiple versions of SNAPPI-DB and the API for download. As JDO allows storage of the data in any form a relational database form of SNAPPI-DB will also be created so that users that prefer not to access the data via the API can do so. In addition, HMM profiles of the multiple structural alignments will be generated for matching to sequences of putative interacting proteins.

A web interface to SNAPPI-DB, SNAPPI-View, is under development and is currently available to perform simple searches of the database (www.compbio.dundee.co.uk/SNAPPI/search.jsp). SNAPPI-View will be extended to provide functions such as viewing the domain-domain interaction alignments, the structures of the interacting domains and protein interaction networks. As the functionality of the full SNAPPI-View interface is extensive, the web interface will be presented elsewhere.

CONCLUSIONS

In summary, a database of Structures, iNterfaces and Alignments of Protein–Protein Interactions (SNAPPI-DB) and corresponding API has been created. The main features of SNAPPI-DB are:

- The API is specifically designed for analysis at the level of domains and domain–domain interactions in addition to PDB entries.
- The core data are derived from a consistent and high quality data source, the MSD data warehouse.
- The JDO technology provides abstraction from complex SQL queries and allows fast development time.
- The object-oriented data store allows high performance and provides a more appropriate model for biological data than does a relational database schema.
- SNAPPI-DB uses multiple domain definitions and PQS-generated biological structures.
- SNAPPI-DB uses the same unique identifiers as the MSD to facilitate interoperability with the MSD warehouse.
- SNAPPI-DB contains many forms of derived data such as SCOP, CATH, Pfam, InterPro, SWISSPROT, GO terms, PQS and secondary structure information.
- The domain–domain interfaces are classified at every level of the CATH and SCOP hierarchies and and by interaction interface type.
- Multiple structural alignments are provided for domain–domain interactions classified by interface orientation.

AVAILABILITY AND REQUIREMENTS

The SNAPPI package includes the Java 5 API, Ant tasks to generate the compiled code, XML files which contain the details of the objects to be stored, a properties file which stores the file locations and connection details specific to the user, a manual, source code and documentation and of course

SNAPPI-DB. The documentation comes in the form of annotated JavaDocs and an in-depth manual. The source code contains many classes of example code.

The SNAPPI-DB package is available for download from www.compbio.dundee.ac.uk/SNAPPI/downloads.jsp. For any help or queries contact emily@compbio.dundee.ac.uk.

SNAPPI-DB and the API are available for both Linux and Windows operating systems and the database will work in parallel for read access. The system will be updated approximately every 6 months, while changes to the derived data such as SCOP and CATH releases will parallel the changes made to the MSD.

The system is distributed under the GPL licence. For alternative non-exclusive licensing options email geoff@compbio.dundee.ac.uk.

ACKNOWLEDGEMENTS

The authors thank the MSD group at EBI for discussions and information, Dr Jonathan Monk, Mr Eduardo Damato and Dr Jon Barber for network and systems support. Emily Jefferson is supported by a BBSRC (UK Biotechnology and Biological Sciences Research Council) studentship and Thomas Walsh is funded by TEMBLOR, European Community Contract No. QLRI-CT-2001-00015 and Scottish Funding Council, Scottish Bioinformatics Research Network (7030-355-064105-35OF). Funding to pay the Open Access publication charges for this article was provided by BBSRC.

Conflict of interest statement. None declared.

REFERENCES

1. Fields,S. and Song,O. (1989) A novel genetic system to detect protein–protein interactions. *Nature*, **340**, 245–246.
2. Zhu,H., Bilgin,M., Bangham,R., Hall,D., Casamayor,A., Bertone,P., Lan,N., Jansen,R., Bidlingmaier,S., Houfek,T. *et al.* (2001) Global analysis of protein activities using proteome chips. *Science*, **293**, 2101–2105.
3. Gavin,A.C., Bosche,M., Krause,R., Grandi,P., Marzioch,M., Bauer,A., Schultz,J., Rick,J.M., Michon,A.M., Cruciat,C.M. *et al.* (2002) Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, **415**, 141–147.
4. Sprinzak,E. and Margalit,H. (2001) Correlated sequence-signatures as markers of protein-protein interaction. *J. Mol. Biol.*, **311**, 681–692.
5. Marcotte,E.M., Pellegrini,M., Ng,H.L., Rice,D.W., Yeates,T.O. and Eisenberg,D. (1999) Detecting protein function and protein-protein interactions from genome sequences. *Science*, **285**, 751–753.
6. Ng,S.K., Zhang,Z. and Tan,S.H. (2003) Integrative approach for computationally inferring protein domain interactions. *Bioinformatics*, **19**, 923–929.
7. Gomez,S.M., Noble,W.S. and Rzhetsky,A. (2003) Learning to predict protein–protein interactions from protein sequences. *Bioinformatics*, **19**, 1875–1881.
8. Marcotte,E.M., Xenarios,I. and Eisenberg,D. (2001) Mining literature for protein–protein interactions. *Bioinformatics*, **17**, 359–363.
9. Bock,J.R. and Gough,D.A. (2001) Predicting protein–protein interactions from primary structure. *Bioinformatics*, **17**, 455–460.
10. Kolesov,G., Mewes,H.W. and Frishman,D. (2002) Snapper: gene order predicts gene function. *Bioinformatics*, **18**, 1017–1019.
11. Breitkreutz,B.J., Stark,C. and Tyers,M. (2003) The grid: the general repository for interaction datasets. *Genome Biol.*, **4**, R23.
12. Hermjakob,H., Montecchi-Palazzi,L., Lewington,C., Mudali,S., Kerrien,S., Orchard,S., Vingron,M., Roechert,B., Roepstorff,P., Valencia,A. *et al.* (2004) IntAct: an open source molecular interaction database. *Nucleic Acids Res.*, **32**, D452–D455.
13. Bader,G.D. and Hogue,C.W. (2000) Bind—a data specification for storing and describing biomolecular interactions, molecular complexes and pathways. *Bioinformatics*, **16**, 465–477.
14. Zanzoni,A., Montecchi-Palazzi,L., Quondam,M., Ausiello,G., Helmer-Citterich,M. and Cesareni,G. (2002) MINT: a molecular INTeraction database. *FEBS Lett.*, **513**, 135–140.
15. Xenarios,I., Salwinski,L., Duan,X.J., Higney,P., Kim,S.M. and Eisenberg,D. (2002) DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, **30**, 303–305.
16. Xenarios,I., Rice,D.W., Salwinski,L., Baron,M.K., Marcotte,E.M. and Eisenberg,D. (2000) DIP: the database of interacting proteins. *Nucleic Acids Res.*, **28**, 289–291.
17. Xenarios,I., Fernandez,E., Salwinski,L., Duan,X.J., Thompson,M.J., Marcotte,E.M. and Eisenberg,D. (2001) DIP: the database of interacting proteins: 2001 update. *Nucleic Acids Res.*, **29**, 239–241.
18. Peri,S., Navarro,J.D., Amanchy,R., Kristiansen,T.Z., Jonnalagadda,C.K., Surendranath,V., Niranjan,V., Muthusamy,B., Gandhi,T.K., Gronborg,M. *et al.* (2003) Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res.*, **13**, 2363–2371.
19. von Mering,C., Huynen,M., Jaeggi,D., Schmidt,S., Bork,P. and Snel,B. (2003) String: a database of predicted functional associations between proteins. *Nucleic Acids Res.*, **31**, 258–261.
20. Jones,S. and Thornton,J.M. (1996) Principles of protein-protein interactions. *Proc. Natl Acad. Sci. USA*, **93**, 13–20.
21. Teichmann,S.A. (2002) Principles of protein–protein interactions. *Bioinformatics*, **18** (Suppl. 2), S249.
22. Janin,J. and Chothia,C. (1990) The structure of protein–protein recognition sites. *J. Biol. Chem.*, **265**, 16027–16030.
23. Nooren,I.M. and Thornton,J.M. (2003) Diversity of protein–protein interactions. *EMBO J.*, **22**, 3486–3492.
24. Jones,S., Marin,A. and Thornton,J.M. (2000) Protein domain interfaces: characterization and comparison with oligomeric protein interfaces. *Protein Eng.*, **13**, 77–82.
25. Lo Conte,L., Chothia,C. and Janin,J. (1999) The atomic structure of protein–protein recognition sites. *J. Mol. Biol.*, **285**, 2177–2198.
26. Jones,S. and Thornton,J.M. (1997) Prediction of protein–protein interaction sites using patch analysis. *J. Mol. Biol.*, **272**, 133–143.
27. Landgraf,R., Xenarios,I. and Eisenberg,D. (2001) Three-dimensional cluster analysis identifies interfaces and functional residue clusters in proteins. *J. Mol. Biol.*, **307**, 1487–1502.
28. Koike,A. and Takagi,T. (2004) Prediction of protein-protein interaction sites using support vector machines. *Protein Eng. Des. Sel.*, **17**, 165–173.
29. Aloy,P. and Russell,R.B. (2002) Interrogating protein interaction networks through structural biology. *Proc. Natl Acad. Sci. USA*, **99**, 5896–5901.
30. Aloy,P. and Russell,R.B. (2003) Interprets: protein interaction prediction through tertiary structure. *Bioinformatics*, **19**, 161–162.
31. Lu,L., Lu,H. and Skolnick,J. (2002) Multiprospector: an algorithm for the prediction of protein-protein interactions by multimeric threading. *Proteins*, **49**, 350–364.
32. Aytuna,A.S., Gursoy,A. and Keskin,O. (2005) Prediction of protein-protein interactions by combining structure and sequence conservation in protein interfaces. *Bioinformatics*, **21**, 2850–2855.
33. Szilagyi,A., Grimm,V., Arakaki,A. and Skolnick,J. (2005) Prediction of physical protein-protein interactions. *Phys. Biol.*, **2**, S1–S16.
34. Stein,A., Russell,R. and Aloy,P. (2005) 3did: interacting protein domains of known three-dimensional structure. *Nucleic Acids Res.*, **33**, D413–D417.
35. Davis,F.P. and Sali,A. (2005) Pibase: a comprehensive database of structurally defined protein interfaces. *Bioinformatics*, **21**, 1901–1907.
36. Winter,C., Henschel,A., Kim,W.K. and Schroeder,M. (2006) SCOPPI: a structural classification of protein–protein interfaces. *Nucleic Acids Res.*, **34**, D310–D314.
37. Gong,S., Yoon,G., Jang,I., Bolser,D., Dafas,P., Schroeder,M., Choi,H., Cho,Y., Han,K., Lee,S. *et al.* (2005) Psibase: a database of protein structural interactome map (psimap). *Bioinformatics*, **21**, 2541–2543.
38. Ogmen,U., Keskin,O., Aytuna,A.S., Nussinov,R. and Gursoy,A. (2005) PRISM: protein interactions by structural matching. *Nucleic Acids Res.*, **33**, W331–W336.
39. Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.

40. Westbrook,J., Feng,Z., Jain,S., Bhat,T.N., Thanki,N., Ravichandran,V., Gilliland,G.L., Bluhm,W., Weissig,H., Greer,D.S. *et al.* (2002) The protein data bank: unifying the archive. *Nucleic Acids Res.*, **30**, 245–248.
41. Murzin,A., Brenner,S., Hubbard,T. and Chothia,C. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.
42. Lo Conte,L., Brenner,S., Hubbard,T., Chothia,C. and Murzin,A. (2002) SCOP database in 2002: refinements accommodate structural genomics. *Nucleic Acids Res.*, **30**, 264–267.
43. Andreeva,A., Howorth,D., Brenner,S., Hubbard,T., Chothia,C. and Murzin,A. (2004) SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Res.*, **32**, D226–D229.
44. Orengo,C., Michie,A., Jones,S., Jones,D., Swindells,M. and Thornton,J. (1997) CATH—a hierarchic classification of protein domain structures. *Structure*, **5**, 1093–1108.
45. Pearl,F., Lee,D., Bray,J., Sillitoe,I., Todd,A., Harrison,A., Thornton,J. and Orengo,C. (2000) Assigning genomic sequences to CATH. *Nucleic Acids Res.*, **28**, 277–282.
46. Pearl,F., Todd,A., Sillitoe,I., Dibley,M., Redfern,O., Lewis,T., Bennett,C., Marsden,R., Grant,A., Lee,D. *et al.* (2005) The CATH domain structure database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis. *Nucleic Acids Res.*, **33**, D247–D251.
47. Sonnhammer,E.L., Eddy,S.R. and Durbin,R. (1997) Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, **28**, 405–420.
48. Bateman,A., Birney,E., Cerruti,L., Durbin,R., Etwiller,L., Eddy,S.R., Griffiths-Jones,S., Howe,K.L., Marshall,M. and Sonnhammer,E.L. (2002) The Pfam protein families database. *Nucleic Acids Res.*, **30**, 276–280.
49. Bateman,A., Coin,L., Durbin,R., Finn,R.D., Hollich,V., Griffiths-Jones,S., Khanna,A., Marshall,M., Moxon,S., Sonnhammer,E.L. *et al.* (2004) The Pfam protein families database. *Nucleic Acids Res.*, **32**, D138–D141.
50. Henrick,K. and Thornton,J.M. (1998) PQS: a protein quaternary structure file server. *Trends Biochem. Sci.*, **23**, 358–361.
51. Golovin,A., Oldfield,T.J., Tate,J.G., Velankar,S., Barton,G.J., Boutselakis,H., Dimitropoulos,D., Fillon,J., Hussain,A., Ionides,J.M. *et al.* (2004) E-MSD: an integrated data resource for bioinformatics. *Nucleic Acids Res.*, **32**, D211–D216.
52. Velankar,S., McNeil,P., Mittard-Runte,V., Suarez,A., Barrell,D., Apweiler,R. and Henrick,K. (2005) E-MSD: an integrated data resource for bioinformatics. *Nucleic Acids Res.*, **33**, D262–D265.
53. Mulder,N.J., Apweiler,R., Attwood,T.K., Bairoch,A., Bateman,A., Binns,D., Bradley,P., Bork,P., Bucher,P., Cerutti,L. *et al.* (2005) Interpro, progress and status in 2005. *Nucleic Acids Res.*, **33**, D201–D205.
54. Harris,M.A., Clark,J., Ireland,A., Lomax,J., Ashburner,M., Foulger,R., Eilbeck,K., Lewis,S., Marshall,B., Mungall,C. *et al.* (2004) The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res.*, **32**, D258–D261.
55. Boeckmann,B., Bairoch,A., Apweiler,R., Blatter,M.-C., Estreicher,A., Gasteiger,E., Martin,M.J., Michoud,K., O'Donovan,C., Phan,I. *et al.* (2003) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, **31**, 365–370.
56. Srdanovic,M., Schenk,U., Schwieger,M. and Campagne,F. (2005) Critical evaluation of the JDO API for the persistence and portability requirements of complex biological databases. *BMC Bioinformatics*, **6**, 5.
57. Ambler,S. (2003) *Agile Database Techniques*. John Wiley & Sons.
58. Carugo,O. and Argos,P. (1997) Protein–protein crystal-packing contacts. *Protein Sci.*, **6**, 2261–2263.
59. Ponstingl,H., Henrick,K. and Thornton,J.M. (2000) Discriminating between homodimeric and monomeric proteins in the crystalline state. *Proteins*, **41**, 47–57.
60. Jefferson,E.R., Walsh,T.P. and Barton,G.J. (2006) Biological units and their effect upon the properties and prediction of protein–protein interactions. *J. Mol. Biol.*, http://dx.doi.org/10.1016/j.jmb.2006.09.042.
61. Chothia,C. (1976) The nature of the accessible and buried surfaces in proteins. *J. Mol. Biol.*, **105**, 1–12.
62. Aloy,P., Ceulemans,H., Stark,A. and Russell,R. (2003) The relationship between sequence and domain-domain interaction divergence in proteins. *J. Mol. Biol.*, **332**, 989–998.
63. Russell,R.B. and Barton,G.J. (1992) Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. *Proteins*, **14**, 309–323.
64. Dafas,P., Bolser,D., Gomoluch,J., Park,J. and Schroeder,M. (2004) Using convex hulls to extract interaction interfaces from known structures. *Bioinformatics*, **20**, 1486–1490.