LHSG-CT-2003-503420

BioXHIT

A European integrated project to provide a highly effective technology platform for Structural Genomics.

Life Sciences, Genomics and Biotechnology for Health

WP5.2: De 5.2.2 Evaluation report on the requirements of the database system for storing the data

Due date of deliverable: **31.12.2005**
Actual submission date: **31.12.2005**

Start date of project: **1.1.2004** **Duration: 60 months**

Organisation name of lead contractor for this deliverable: CCP4/CCLRC Daresbury **Author:** Peter Briggs

WP 5.2: Data management and project tracking in structure solution software. Coordinator Peter Briggs (Partner 10), contributing Partners 1C, 10.

De 5.2.2 *Evaluation report on the requirements of the database system for storing the data*

This report is part of *Task 5.2.1* aims to fill the need for project tracking within the BIOXHIT structure solution software pipeline. CCP4 currently provides a project database within its graphical user interface CCP4i, which records basic information about each job run through the interface within a particular project, and is useful for reviewing progress through a structure determination, backtracking and managing files and other data associated with each job. This task will develop the CCP4i project database into a more general project database that can be used for project tracking within the BIOXHIT software pipeline, *Ms 5.2.1* and *D 5.2.1*. In the later stages of the project more tools will be added to increase the usefulness of the project history database, for example search and visualisation mechanisms. Compatibility with other databases and LIMS developments (for example LIMS developed by Partner 1C, WP 5.1) will also be vital, and data exchange mechanisms will be implement based on communications technologies developed elsewhere in WP 5.1, *Ms 5.2.2* and *D 5.2.2*. This task will also result in the following deliverables: *D 5.2.3*, *D 5.2.4*, *D 5.2.5*, *D 5.2.6*.

Evaluation Report on the Requirements of the Database System

Introduction

The propose project tracking and database system being developed in BIOXHIT Work Package 5.2 is made up of a number of components, namely:

- A brokering application (the project database handler) to mediate between client applications and the database
- Visualisation applications to display the database contents to end users
- A database to actually store the data.

A separate deliverable (D 5.2.3) outlines the specification for version 1 of this system, which has been implemented in deliverable D 5.2.1. This report is concerned solely with evaluation of the requirements of the last of the three components listed above, i.e. the database storage system.

The report follows from a meeting on Database Requirements for CCP4 Projects held 17th October 2005 York University (a detailed summary of that meeting is included in this report as appendix A) in conjunction with many discussions with interested parties over the past year.

Database overview

The database structure will inherit from CCP4i the notion of division of the structure determination process into “projects” (one per target structure), and is intended to be a dynamic repository capable of reflecting the current state of knowledge about the structure determination. It should be able to act as both a “source” and a “sink” for data used by crystallographic applications, that is, applications should be able to

acquire the data that they need in order to run and then update the database afterwards with their results (ready to be accessed by other processes).

Within a project the data that will be stored is divided up into three groups:

- **Tracking/project history**
  This keeps records of events which make changes to the database contents. It is partially implemented in CCP4i, where records of jobs (i.e. runs of programs or scripts) are kept along with links to parameter files, log files and input and output files.
  In a full implementation all events would be recorded – not only runs of programs but also information on updates to any item or "data object" stored or referenced by the database.

- **Project Data**
  This is a structured repository for crystallographic data that can be shared between different applications – that is, it is considered to be generally useful and transferable and is not application-specific. This database component has been labelled variously as the "knowledge base" (as it stores what is known about the structure determination at the current time) and as the "exchange database" (as it acts as a way of indirectly exchanging data between applications).
  The state of the project database will change as more information becomes available about the structure being worked upon – that is the dataset is dynamic. The database must therefore be able to accommodate these changes. It must also be able to provide the information required for final deposition.

- **Operational data**
  This is application-specific data, i.e. data in a form that is generally only useful or meaningful to a particular application. Examples of operational data would in the first instance be CCP4i parameter files or XIA pickled Python objects – it is plausible that either could be shared with other applications, but this would not be true generally.

The following sections outline the requirements identified for each of these components in turn.

Requirements for tracking

Tracking requires that we have a way of defining an event or action and a way of referencing the objects that were changed by the event. Events would not be limited to runs of programs – they might also reflect decision points or the results of manual/interactive processes – so a set of "event types" would have to be defined. A timestamp would also need to be associated with each event.

Other attributes could also be associated with events – for example, CCP4i supports a "notebook" function that allows the user to add annotation to the record of a program run.

Requirements for the knowledge base

The preliminary evaluation of how the knowledge base should be structured and what it should contain suggested that it should consist of a core set of data types. Broadly these should include:

- Crystal data (grouping datasets collected from the same crystal, plus an associated unit cell)
- Dataset (grouping reflection lists collected in the same experiment, plus associated wavelength and f' and f'' values)
- List of reflections of different types (e.g. intensities, structure factor amplitudes)
- Molecule (defining the current knowledge about the target molecule)
    - Unit cell contents (numbers and types of atom)
    - Sequence
- Free reflection set (a mask applied to a set of reflections)
- NCS operations
- Unit Cell (defined as cell parameters plus symmetry information)
- Data processing
    - Sweep (single continuous set of diffraction images)
    - Orientation matrix
    - Experimental geometry information (e.g. rotation axes, beam vectors etc)
    - Scaling parameters and statistics
- Experimental phasing
    - Descriptions of phasing attempts and phasing results
    - Maps (patterson or electron density)
    - Descriptions of heavy atom substructure (heavy atom positions and statistics)
- Molecular replacement
    - List of molecular replacement models and related statistics
- Protein model
- Program list (a set of available programs with associated data such as version, author list and citation text)

Currently a more detailed analysis of these data types is ongoing with the aim of building a more formal data model that also establishes the relationships between the data items. It is also clear that more investigation needs to be performed into the requirements for molecular replacement, model building and refinement.

Requirements for operational data

Being application-specific, by its very nature operational data will be difficult to store in a structured manner. It is probably most realistic to provide only the most basic storage mechanism for operational data. At present the "dingbat" data object implemented in the version 1 database handler would appear to be a good model for the storage primitive of the operational database. This should be linked to the tracking system but otherwise left to the specific applications to manage.

Alternatively an application may provide its own operational database, in which case there would be a requirement that the tracking system is capable of referencing data objects stored externally.

There is also an issue of the "volatility" of operational data objects, namely are they temporary or should they be stored forever?

Requirements for different types of application

We can divide the applications that have been considered into two classes: one that intends to have very close linkage to the database system, such as XIA and HAPPy,

and another that wants only minimal contact with the system, such as Coot or CCP4mg.

For close linkage the applications will expect the database to be the primary store for all their data, and deficiencies in the data model will be very problematic. For other applications, they will most likely want just to leave a small footprint in the tracking database (for example linking to a refined set of coordinates output from an interactive run of Coot).

It is not clear at this point whether this leads to a divergence in the requirements for the database, or whether one is simply a very small subset of the other.

Other requirements

- Other projects have demonstrated that it is vital to build in management of user rights at an early stage
- The database must also store the information that is required at deposition time.
- Links to other databases must be built in, for example a reference to data stored in a LIMS or beamline database.
- Having a "controlled vocabulary" (along the lines of a project glossary or something more formal, for example mmCIF) is vital.

Conclusions and next steps

The evaluation work outlined in this report indicates that there is a large amount of crystallographic data that needs to be stored in order to be useful for crystallographic applications, and the creation of a data model for this purpose is nontrivial. It may also be that the model has to be highly complete before it can be of practical use to application programs. It should also be pointed out that the data model is not intended to act as a general purpose model of protein crystallographic data.

Another practical issue is that of moving from the abstract representation of the data model to a concrete implementation. This will involve not only the generation of schemas for some database language e.g. mySQL but also the creation of a set of API functions to allow the content of the schema to be accessed in some controlled manner. Clearly there is a terrific amount of work required to do this manually, and so the use of tools that can automatically generate some or all of the API functions should be investigated.

Work to construct the data model is ongoing and will ultimately form deliverable D5.2.5 ("specification for version 1 of the project tracking database design"). The evaluation work in this report has provided a valuable foundation for this next step.

Appendix A: Summary of a meeting on Database Requirements for CCP4 Projects 17th October 2005, York University.

Abstract for this meeting

This one-day meeting is a first attempt to gather information on the current and future needs and concerns of CCP4-related projects with regard to data management and the use of databases within CCP4 for structure determination.
Issues to be considered will include:

- what are the data being stored?
- what are the needs for data models?
- consideration of possible database technologies

Participants

Peter Briggs, Keith Wilson, Susy Griffiths, Chris Morris, Wanjuan Yang, Ronan Keegan, Graeme Winter, Kevin Cowtan, Eleanor Dodson, Paul Emsley, Charles Ballard, Dan Rolfe, Garib Murshudov, Liz Potterton, Fei Long, Steven Ness.

Representatives from each of the interested CCP4 and CCP4-related projects are invited to give 20-25 minute presentations on their projects and the data management issues therein i.e. to answer the question "what does this project need a database to do?". More specifically the talk for each project should address the following points:

- What sort of "database(s)" do you want/need in your project e.g. databases of structures, ligands etc.
- Do you want to be able to track operations that the user might perform (and if so at what level)? (Tracking means keeping a record of what has been done in order to visualise, backtrack, restart etc.)
- Do you want to store crystallographic data in it? Do you want to retrieve crystallographic data from other applications from it?
- Are there any technical issues, e.g. preferred languages for APIs?

The projects considered as part of this meeting are listed below, along with their primary contacts):

- CCP4 Automation (Charles Ballard)
- e-HTPX (XIA and BMP) (Graeme Winter)
- York Refmac/Molrep pipeline (Garib Murshudov)
- CRANK (Steven Ness)
- CCP4i (Peter Briggs)
- CCP4mg (Liz Potterton)
- Coot (Paul Emsley)

The primary contacts should each nominate one of their group to speak.
Programme

Time	Talk	Speaker
10:20	Welcome and introduction	
10:30	Data modelling: some lessons from experience *plus discussion*	Chris Morris
11:30	CCP4(i)/Bioxhit Database Project	Peter Briggs
12:00	Lunch	
13:00	CCP4 Automation	Charles Ballard

13:30	e-HTPX	Graeme Winter
14:00	York Refmac/Molrep pipeline	Garib Murshudov
14:30	CRANK	Steven Ness
15:00	Coffee break	
15:30	CCP4i	Peter Briggs
15:40	CCP4mg	Liz Potterton
15:50	Coot	Paul Emsley
16:00	Discussion: contents of DB for automation of structure solution using existing knowledge and information	Chaired by Garib Murshudov
17:00	Close	

Presentations

The following presentations are available for download as Powerpoint files:

- Introduction (Peter Briggs) [CCP4_DBS_MeetingIntro.ppt]
- CCP4(i)/BIOXHIT Database (Peter Briggs & Wanjuan Yang) [CCP4_DB_BIOXHIT_db.ppt]
- "Database" usage and requirements for CCP4 Automation (Charles Ballard) [CCP4_DB_HAPPy.ppt]
- Databases in e-HTPX (Graeme Winter) [CCP4_DB_eHTPX.ppt]
- Crank and Databases (Steven Ness) [CCP4_DB_Crank.ppt]
- CCP4i database overview (Peter Briggs) [CCP4_DB_CCP4i.ppt]

A presentation by Chris Morris is also available from a password protected area:

- The Map Is Not The Territory: Some experience of software development using a data model [data_modelling.ppt]

Thanks to all the speakers for providing their presentations.

Outcomes

- **CCP4 data model:** we will now work on incorporating the requirements of the various participants into the CCP4/BIOXHIT data model.
- **EBI data model:** this has been suggested as a potential starting point for CCP4 data model. See a version here http://www.ccp4.ac.uk/peter/private/MSD-SQL/sql.html (password protected)
- **Project glossary:** it was suggested that we should maintain a project glossary which defines the common concepts. An example which could be used as a starting point for this project is here: example_glossary.html Another example is that for the PIMS project: http://www.pims-lims.org/project/glossary.html

Meeting Notes Notes prepared by Wanjuan Yang and Peter Briggs

Peter Briggs started the meeting by talking about the meeting's theme and agenda. The meeting was intended to gather information on the current and future needs and concerns of CCP4-related projects with regard to data management and the use of databases within CCP4 for structure determination.

The issues to be considered were:

- What are the data being stored?

- What are the needs for data models?
- Consideration of possible database technologies.

This meeting wasn't intended to resolve these issues but to serve as a starting point for discussion and to decide the steps needed to progress.

Peter also reported on the database meeting held at Daresbury on October 13th 2005. That previous meeting examined the scope and interface/overlaps/data exchange issues between projects. The key points were:

- The scope of PIMS is expression/crystallisation.
- ISPyB is installed on the beamline and deals with information from the data collection experiment. ISPyB "wraps" DNA on the beamline.
- CCP4/BIOXHIT database covers the process post data collection
- e-HTPX is seen as the "glue" which can hold these together. The interface between PIMS and ISPyB is an ongoing issue which needs to be resolved within e-HTPX.
- The location of the databases is also an issue, for example it might be possible to have ISPyB installed at the home lab, or the CCP4 database installed on the beamlines. An action from the earlier meeting was for Peter to follow this up with Richard Wollinscroft at DIAMOND (*nb a meeting has since taken place – PJB*). It is also recognised that there is a requirement to be able to export/import data, in order for the end user to be able to move it around e.g. from beamline to laptop to home lab.

Chris Morris gave a talk about data modelling in PIMS: some experience of software development using a data model.

Some key points were:

1. In a large system it is vital to build in management of user rights at an early stage.
2. A number of technologies can be considered: Ontology, UML, XML, SQL, Object-Role modelling, controlled vocabulary. These each have advantages and drawbacks, some of which are summarised as:
    1. UML: is hard to read, hard to generate data models from, and very hard to share development of.
    2. XML: drawbacks for use as a data model, but suitable for data exchange.
    3. SQL: good for sharing very large amounts of data.
    4. Object role modelling: E-R model and SQL.
    5. Controlled vocabulary (e.g. mmCIF): without an agreed vocabulary, all other data modelling will fail.
3. The data model is reflected in the user interface (UI). If you can't use the interface easily, then it's an indication that you'd better fix the model.

The remaining presentations focused on projects currently being developed. Specific questions to be answered were:

1. What sort of "database" do you want/need/use now in your project, e.g databases of structures, ligands etc.
2. Do you want to be able to track operations that the user might perform(and if so at what level)?
3. Do you want to store crystallographic data in it? Do you want to retrieve crystallographic data from other applications from it? What data in particular?
4. Are there any technical issues, e.g preferred languages for APIs?
5. What do we already have that we can use now?

Peter Briggs gave a talk on CCP4(i)/BIOXHIT Database Project: Scope, Aims, Plans, Status and all that jazz.

First Peter gave the background of BIOXHIT (Bioxtallography on a Highly Integrated Technology Platform for European Structural Genomics) which is an EU framework programme 6 integrated project with more than 20 partner institutions. The aim is to provide platform for high-throughput structure determination from crystallisation to structure solution. Although the project started on 1st January 2004 for 4 years, CCP4's start was delayed due to a recruitment problem. Background information on BIOXHIT can be found in the website http://www.bioxhit.org.

CCP4's contribution of CCP4 to BIOXHIT is in work package 5.2 ("Data management & project tracking in structure solution"). The aim is to fill the need for project tracking within the BIOXHIT structure solution software pipeline (the pipeline covers software components post-data processing i.e. scaling and merging, phasing, model building, refinement) and so the project is complementary to PIMS and DNA. The staff for the CCP4 effort at Daresbury is Peter Briggs (project coordinator) and Wanjuan (Wendy) Yang (full time programmer).

Peter then talked about the history of CCP4 database. CCP4i job database records details of tasks run including associated files, parameters, date, status etc. There is no additional information e.g. relationship between jobs or crystallographic data, and the job database is only accessible via CCP4i. Within BIOXHIT the remit of the database has been expanded to include extended tracking (i.e. storing the relationships between jobs) and storage of crystallographic data. Although this means that a CCP4 data model will be required, CCP4 is not committed to providing a general data model for structure solution.

The scope of the project is to deal with inputs to and output from software components post data processing up to structure validation and deposition. Within this we provide

1. Tracking information for steps taken, ie programs run, decision made, associated input/output files or other "data objects"
2. Crystallographic data, both application-specific and generic.

The target users are single users performing manual/automated/mixed procedures. Other modes of operation have not been requested or investigated so far.

The aim of the project is to implement system for both manual and automated structure determination. It will use CCP4i as a starting point and accommodate non-CCP4(i) applications, providing a small/lightweight database system to support single applications. Multiple database backends will be implemented. Information will be gathered as much as possible automatically. The project recognises that structure determination will most likely not be performed exclusively within a single software package and data will most likely not be stored in a single database. Exchange of data between systems requires standards for transfer such as standards developed in BIOXHIT WP 5.1.

The project deliverables are:

- Project database handler: This is a broker application to mediate interactions between database and client applications, aim to provide client APIs to handler from different languages. It will deal with multiple users/clients within/outside CCP4 system.
- Project database: The database will include:
    - tracking/project history (steps in the determination process): stores project history, contains links to the data in internal and external DBs, show relationships between data items/steps taken.

    - project data ("knowledge base"/"exchange database") & data history: this contains common crystallographic data items used within the software pipeline which are shared between different applications, and will require relevant info from earlier stages (crystallisation/data collection). It will also provide information for final deposition.
    - application-specific ("operational") data: these are the application specific data & representations, e.g. CCP4i parameter files, XIA python objects, not intended to be shared between different applications. The aim is to provide database schema and multiple implementations.
- Visualisation tools: these will provide views of the data to facilitate review and analysis.

Current status of the project: so far the project has focused on prototyping the tracking database, and this has revealed the need for the operational and knowledge databases. Little work has been done on the exchange database; this will be informed by this meeting. What we can say now is that the contents of exchange database will be dictated by the interfaces between applications, and by the data needed for deposition. The needs of the applications will also dictate what information is required from downstream/upstream databases, i.e. development should be driven by applications' requirements.

The prototype handler is written in python, with APIs provided in Python and Tcl. The database backend is MySQL, which is robust and easy to develop with (e.g. it allows us to duck concurrency issues) but may not be good long-term choice. XML is used as a messaging technology within CCP4 db, and socket communications are used between the server and the client applications.

The current collaborators are e-HTPX/XIA, HAPPy and CCP4i. Other relevant projects are PIMS (this provides the requirement to be able to exchange data) and projects that wish to interface with CCP4i (CCP4MG & Coot, Crank, Mosflm).

Peter summarised the talk by emphasising that: CCP4/BIOXHIT DB aims to provide small/lightweight db system to support applications, but not to provide either a data model for all structure solution software applications or a central database implementation. Next stage of the project is to revise and expand db schema, and the development is dependent on input from application developers.

Some discussion:

- Paul asked why and where do you want to store space group?
  Graeme said during data processing, there is a need to store space group. Kevin said need to store estimated space group. Kevin also mentioned that MySQL hasn't got foreign key constrain (by default?), need to turn it on.

Charles Ballard talked about "Database usage and requirements for CCP4 automation". The loose storage requirements include tracking jobs, data types, knowledge base, object persistence, templates and protocols. Current database usage in HAPPy is for job tracking – it is a workflow view to track progress. Tracking can use either a database or file system – the current CCP4i is a good example.

The HAPPy view of tracking is designed to work with the CCP4i database, and contains nodes (actions) with one of five types (job, fork, decision, start, end/container) and one of four statuses (success, failed, killed, running). Each of the nodes also has a timestamp and human readable notes. For data tracking, it is nice to store metadata: for example, to describe an MTZ file using URI (Universal

Resource Identifier) plus wavelength, cell, and column names. It is also nice to store data source and usage.

For a knowledge base, it would be a small amount of data stored in XML files or in a database. This data should be project and pipeline independent and map onto mmCIF definitions at the deposition end. Data here includes:

- HA positions and statistics
- Sequence
- solvent fraction
- MR models.

For persistent objects, also viewed as a "data bucket", this is project specific data, for example store the local state in object orientated database (e.g. ZODB), as a dingbat, or in an XML file. For example: HAPPy has a "HAPPy state object" which is always in memory and which stores the current state and the history. Persistence is required for restart.

For storage of protocols and templates, the database would need to store user preferences and pathways or workflow.

Graeme Winter talked about database usage in e-HTPX. He said in the process of solving a structure, project management is important because operations might be performed at a different site (probably a remote site). Pipeline projects within the e-HTPX domain include:

- XIA-DPA (automated data processing)
  Database is required for data exchange & internal data management
- BMP (bulk molecular replacement)
  Database of model structures is required (currently this is an external PDB database at the EBI) & internal job management
- XIA-HA (heavy atom/experimental phasing pipeline)
  Database for obtaining program input, or (in the case of a suspended job e.g. to run at a different site or on another computer) in order to perform a restart.
- Deposition

Graeme said we need to know the relationship between data, which the MTZ file doesn't completely reflect. The database also needs to have import /export mechanism. He concluded saying that "When we learn something, we need to store it".

Garib talked about an Automated Molecular Replacement system being developed at York. The "entry" requirements are all the experimental data; on exit the system produces coordinates and a set of diagnostics. Internally the system consists of a script plus a database of structures – essentially a "reorganised PDB" which stores information at different structural levels (domains, multimers, monomers etc). Garib's interests are in improving the structure of the molecular replacement database.

Steven Ness talked about database usage in CRANK. He started with an overview of CRANK, which currently uses the file system to store all the data needed by each step. It organises the data in hierarchical directories and uses filenames to encode program step and the type of data. MTZ column labels also encode information through the use of symbolic column names, and all CCP4i user input column labels are renamed to avoid known problems with manipulation programs.

Other types of input data are:

- Sequence

- Substructure
- List of substructures
- Protein model
- List of protein models
- Map
- Rfree column etc.

CRANK XML is generated either directly by programs which have been modified to output XML markup, or by wrappers which convert log files to XML. It stores all information generated by programs. The main usage of the XML file is for decisions, and using these allow the user to direct program and information flow in their pipeline. (A secondary purpose is to allow for the possibility of data mining.)

Steven outlined CRANK's requirements for any future database implementation. In CRANK there needs to be a way to access any given column in an MTZ file. It also needs to be able to store sequence, substructure, protein models, maps, Rfree columns, and many more types. It also needs to be able to access the data via an API (Python , Tcl, C, C++) and the file system.

Peter gave a talk about the current CCP4i database. A CCP4i project is a directory containing a set of 'databases': a job database, job data such as parameter files for each job, log files from each job, notebook entry etc, Amore MR model database and experimental XML files. CCP4i also stores a list of projects and aliases that "belong to" each user.

CCP4i is runs in a "single user" mode of operation, i.e.:

- each project is owned by a single user
- each user runs a single instance of CCP4i

CCP4mg also uses CCP4i projects. CCP4i acts as a visualiser and provides an interface to manipulate the job database. CCP4i spawns running jobs as independent processes and these also interact with job database using limited "write-only" operations.

Issues for CCP4i include:

- speed of access to data is crucial because users request data in real time
- sharing projects/data between applications and users is ad hoc
- there is issue of access permissions and questions of whether multiple users want to access the same job database?
- tracking information needs to be expanded to allow the concepts of "subjobs" and "subprojects".

CCP4i could store common (crystallographic) project data accessible to all tasks e.g. project.def file. Initially the data is populated by hand when project is started, then updated from output of task, e.g. from XML files generated by programs, or updated by hand (via an appropriate interface). Tasks could query project.def to automatically populate field. Possible data items for project.def are:

1. sequence data
2. molecular weight(theoretical and experimental)
3. experimental details:
    - type of experiment (MAD, SAD, MIR etc)
    - crystal identifiers: associated cell information, native or derivative, heavy atom data(type, expected number of sites, coordinates, …), datasets derived from each crystal(wavelength, f' and f'', pointer to MTZ columns with intensities/sf amplitudes, pointer to scalepack

intensities. Derived quantities: e.g. nono-crystallographic translation, results from twining analysis, solvent content, number of molecules in asu…..

The question is: what data are useful for input into CCP4i tasks? What data are useful for input into other applications?

Paul talked about the requirements for Coot. His requirements concern interactions with CCP4i specifically – the ability to launch Coot from CCP4i and have it pick up the relevant data from a refinement run (coordinates, MTZ file), and the ability to store the parameters and files associated with a run of Refmac which has been performed within Coot.

Liz talked about the requirements for CCP4mg. Her interest is post the refinement stage (i.e. analysis etc). She would like to be able to record details of what has been done, and save the inputs and outputs from CCP4mg (including user annotation) for example, in order to be able to regenerate views when making pictures. She would also be interested in being able to store user-defined ligand dictionaries for later retrieval.

Some discussion:

- Charles suggested database need to store registration of available program. Keith said EBI already started structure solution progress data modelling, it is worth looking at it and decide whether use it or not. But history tracking is missing.
- Paul asked for raw API of SQL.
- Eleanor suggested also storing sequence alignments
- Garib suggested storing information on twinning
- Garib suggested that we should be able to store data in any format (e.g. MTZ, Scalepack etc) – the database should support some kind of "import/export" feature.
- Paul said storing dictionary is crucial.
- Regarding the content of the Knowledge Base, Charles suggested to divide two groups: MR (he suggested Ronan, Charlie Bond and Garib/Alexei) and experimental phasing (he suggested himself, Dan and Steven). Each group discusses what data they need to store.

Summary and Actions

The main outcome of the meeting was that there is now a better understanding of the requirements that each project has from the CCP4/BIOXHIT database system. Wendy is now working with the developers to produce SQL schemas that capture these requirements, this is an ongoing action.

Other specific actions:

- PJB: to investigate the EBI data model and assess for suitability
- PJB/WY: to produce a glossary/dictionary to try and standardise communications between the different developers (cf "controlled vocabulary" point in Chris Morris' presentation)
- PJB/WY: need to talk more to MR pipeline developers in order to also accommodate their requirements

Appendix

The following is some relevant discussion about database taken from automation meeting note on June7-8 in York.

There was often confusion about what sort of database was being discussed. There are:
- the project history,
- the set of useful facts which need to be accessible, e.g. sequence, or the list of possible SGs,
- the project report which will not include dead ends (probably overlaps with data harvesting),
- and quite separate, the knowledge repository. Eleanor: I see this as a modern $CLIBD. It would include basic crystallographic material (symops, atomsf.lib), the monomer descriptions, histograms, maybe building fragments, etc etc