Data Management in CCP4(i)
Current Status and Future Plans

Peter Briggs
CCP4 Daresbury Laboratory
• Introduction/Overview

• Current status
  • Libraries/MTZ files
  • Data Harvesting
  • Automation in CCP4i
  • CCP4i Project History Database

• Ongoing developments & near-future plans
  • Database handler

• Further ahead
  • Expansion of database content
  • Project history visualisation
CCP4, Data Management and High Through-Put

• In this context HTP = automation

• Data management in automated procedures
  • Acquisition of necessary data on demand (both for running software components and for making decisions)
  • Capture as much information as possible at each step
  • Tracking required for review and diagnostics
  • Information required for validation and deposition

• Requirements are similar (if less formal) for human user

• Automatic transmission of data:
  • allows streamlining of procedures
  • cuts down on manual handling & transcription mistakes
Background: Quick Introduction to the CCP4 Suite

• Collaborative Computational Project No.4
  • software for macromolecular structure determination by X-ray crystallography

• Consists of a suite of ~150 programs plus a set of core software libraries
  • Scope: from data reduction to structure validation/analysis
  • Modular: each program covers small range of functionality
  • Data is transferred via files with standard formats
  • Core libraries used to provide a consistent user and developer environment

• CCP4i (graphical user interface)
  • Sits on top of programs
  • Adds extra functionality (project history database)
CCP4 Libraries: Moving Towards Improved Data Models

• Requirement to support legacy code while moving forward

• Currently: underlying libraries mix Fortran and C
  • “Data models” expressed in Fortran COMMON blocks

• New libraries based on C/C++
  • Data models expressed in C/C++ headers
  • Aim for consistency with community data models
  • Interfaces to Fortran, scripting languages (e.g. Python, Tcl)

• Key components:
  • CMTZ: imposes formal structure on reflection data
  • MMDB: similar for co-ordinate data
MTZ files and Data Hierarchy

• MTZ files
  • MTZ = McCloughlin Terry and Zelinka
  • CCP4 reflection data format

• Next release introduces the concept of /crystal/dataset/column hierarchy

• Explicit relationships between column data enables basic automation:
  • e.g. automatic column selection for consistent scaling
MTZ Hierarchy - Other developments

- e-HTPX is developing XML Schema to describe HTP structure determination
  - XML schema based on CMTZ forms part of description of structure solution (Joel Fillon)
  - CCP4 libraries will be consistent with data model

- Viewer for hierarchical information
  - Selection
  - Editing
Data Harvesting

• Originally introduced into CCP4 in version 4.0 (2000)
  • Historically significant: introduced the concept of datasets into CCP4/MTZ files

• Key programs automatically capture data which are written to harvesting files:
  • MOSFLM, SCALA, TRUNCATE, MLPHARE, REFMAC
  • Harvest files written in mmCIF format
  • Use Project-Dataset name pairs in MTZ header
  • Can be uploaded to AutoDep and merged into current deposition

• Harvesting integrated into CCP4i
  • Turned on by default
Data Harvesting: Harvesting Management Tool (Pryank Patel)

- Raise profile of harvesting within user community
- Graphical tool to assist users in managing harvest files prior to uploading at deposition site

Functions:
- Select and view harvest files
- (Cross) validate files
- Convert mmCIF to XML
**CCP4i - CCP4 graphical user interface**


- Introduced an integrated environment for running CCP4 programs, particularly suitable for novice users

- Easier to run programs

- Data management features:
  - Basic file management
  - Job database
  - Electronic notebook
  - Retrieval of parameters
**CCP4i: Automating Tasks**

CCP4i build around interfaces to **tasks**

- Task = script running one or more programs

Two approaches to automation piloted within CCP4i:

- Add automated procedures within existing tasks
- Automated transfer of data between tasks

Use XML files as transport mechanism for data
CCP4i Automated Parameter Passing within Tasks

• Example 1: CAD Auto-reindexing
  
  • CAD is program for merging two or more reflection files
  • Merging files which are inconsistently indexed can be disastrous
  • Can be checked and corrected automatically

• CAD task runs ALMN program:
  • performs pair-wise comparison of MTZ files
  • generates XML file with reindexing instructions
  • file is parsed by CAD script
  • runs REINDEX with appropriate parameters if necessary

• Automation of a “trivial” but convoluted operation
CCP4i Automated Parameter Passing between Tasks

• Example 2: Molecular Replacement

  • “Cell content” and “MR Analyse” tasks are be run prior to “Molrep”
    • “Cell content” calculates # of monomers in asu
    • “MR Analyse” identifies pseudo translations from Patterson
  • User must transfer data manually to input of “Molrep”

  • Instead: programs write information to XML files:
    • MATTHEWS_COEF for “Cell content”
    • PEAKMAX for “Analyse MR”
  • “Molrep” task parses XML files on startup and reads in data automatically

• Reduce manual handling by users
Example XML Parameter File Fragment

• MR_ANALYSE.xml (output from PEAKMAX)

```xml
<?xml version="1.0"?>
<peakmax_run>
  <PEAKMAX ccp4_version="4.2" date="7/17/02" />
  <peakmax_keyword>
    </peakmax_keyword>
  <peakmax_result
    order_number="1"
    site_number="1"
    height_over_rms="38.66371"
    peak_frac_x="0.5000000"
    peak_frac_y="0.5000000"
    peak_frac_z="0.0000000E+00"
  />
...
```
**CCP4i XML Parsing**

- **Programs:**
  - Use PXXML library to generate XML files

- **CCP4i:** use xml.tcl/sgml.tcl
  - Write custom code based on xml.tcl to parse XML files
  - Currently need to write specific code for each file

- **Generally:** XML is not defined in a global context
  - i.e. there is no ccp4.xml schema defining the tags
  - would be useful to standardise output from programs
  - should be consistent with schema being developed by e-HTPX

---

**Data Management Aspects of High Throughput Structure Determination**

13 - 14th March 2003
CCP4i Project History Database

• CCP4i uses the concept of Projects and Project Directories
  • CCP4i creates a database subdirectory in the Project directory which
    stores database file and job parameter files
  • Definition of a Project is left up to the user
  • One Project per Project Directory

• Job Database (Project History Database)
  • Keeps record of each instance of a task (=job) run in Project
CCP4i Project History Database

Perform clean up

Review parameters

Job Database

Utilities

View Log File in Web Browser
View Log Summary in Web Browser
View Log File
View Log Graphs
View Command Scripts
Input files...
gere_nat.mtz
Output files...
gere_nat_truncate.mtz
WORKSHOP_2_falloff.plt
native.truncate

Data Management Aspects of High Throughput Structure Determination
13 - 14th March 2003
CCP4i Project History Database - technical details

• Flat file storage
  • CCP4i .def file format
  • Indexed parameter-value pairs
  • Associated parameter files (.def files) are identified via standard naming scheme

• Example entry:

  STATUS,55                  FINISHED
  DATE,55                    985023073
  LOGFILE,55                 55_superpose.log
  TASKNAME,55                superpose
  TITLE,55                   "overlaying reduced on oxidised wdmsor"
  INPUT_FILES,55             "/dl/sr/homes/px/slj/1e18.pdb ...
  INPUT_FILES_DIR,55         "FULL_PATH FULL_PATH"
  INPUT_FILES_STATUS,55      ""
  OUTPUT_FILES,55            "overlaid_1.pdb overlaid_2.pdb ...
  OUTPUT_FILES_DIR,55        "PROJECT PROJECT PROJECT PROJECT ...
  OUTPUT_FILES_STATUS,55     ""
CCP4i Database: Limitations

Technical:
- Tcl API to interact with the database embedded in core CCP4i
- Not accessible from outside of CCP4i
- Ambiguous status for many-user access
- Flat file format doesn’t scale well

Content:
- Relationships between jobs can only be inferred from file names
- Focused on history content
CCP4i Database: Database Handler (dbCCP4i)

- Server process running independently of main CCP4i
- Current prototype version:
  - Uses Tcl sockets (allow many processes/users to connect across network boundaries)
  - Need to define communication protocol (request/response)
  - Issues with security/authentication/authorisation

- Manages interactions with Project Database
  - Hides database implementation details
  - Easier to migrate database backend e.g. mySQL

- Alpha version of CCP4i using dbCCP4i available summer 03
  - See http://www.ccp4.ac.uk/peter/dbhandler_specification.html
CCP4i Database: Expanding Scope/Content

- Improving project tracking facilities
  - Capture more information about each job:
    - Logical flow
    - Data flow
  - Fill in gaps from other dbs?

- Expand scope of Project Database
  - e.g. sequence information, MR trial models, HA derivative preparation
  - Could be obtained from external databases e.g. LIMS

- Use community standards (XML schema) to facilitate Data Exchange (both ways)

- Requires redesign/re-implementation of the database
Future Plans: Project History Visualisation Tools

• Currently in CCP4i: just a list

• Investigate other ways of presenting history to emphasise particular aspects:
  • Flow of data - where data comes from
  • Logical flow
  • Paths (“How did I get here?”)
  • Searching/Querying

• Facilitate understanding of the structure solution
  • Increasingly important for reviewing automated procedures
  • Trend towards data-driven interfaces?
Acknowledgements

**CCP4 Daresbury**
Core developers: Martyn Winn, Alun Ashton, Charles Ballard, Peter Briggs
TEMBLOR/Data Harvesting: Pryank Patel

**CCP4 University of York**
Liz Potterton

**e-HTPX (EBI)**
Joel Fillon

... and countless other contributors to the CCP4 Project
The End.