

Strategy behind the new generation CCP4 software library

Author: Martyn Winn
Revision: 0.2 Date: 23/09/2002

Aims

C-level access to crystallographic functionality.

Access to Fortran libraries from C is possible (see e.g. Solomon code), but preferably avoided.

Support for scripting.

Scripting is central to ccp4i, ccp4mg and automation efforts.

Support for legacy code.

Essential to retain existing users and collaborating developers.

Conformance with community-accepted data model.

CCP4 library implements an implicit data model. This model needs to be made explicit, and should conform as closely as is practical to the EBI-led data model.

Major components

Current:

1. CMTZ
2. CMAP
3. MMDB
4. CSYM
5. Unit cell conversions
6. CCP4 style
7. System-dependent functions.

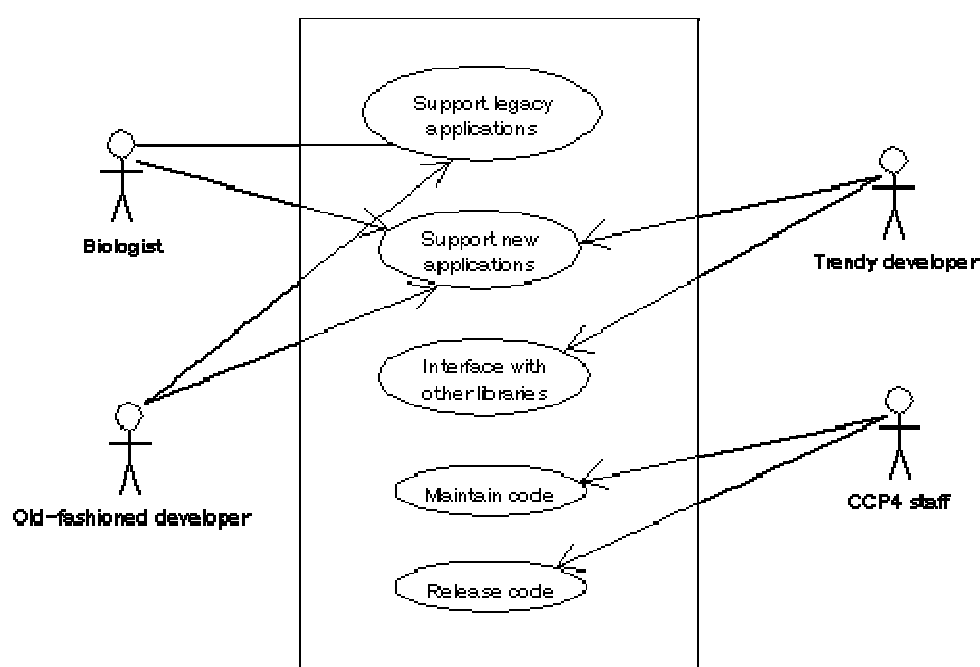
Planned:

1. FFT
2. Sorting
3. HTML/XML
4. Harvesting

Target Uses

1. CMTZ, CMAP used by 3rd parties to access CCP4 formats
2. MMDB used in ccp4mg etc.
3. Support of CCP4 suite
4. Optional back-ends to clipper
5. Provide CCP4 environment to clipper applications

Here is the corresponding Use Case diagram:



Form of library

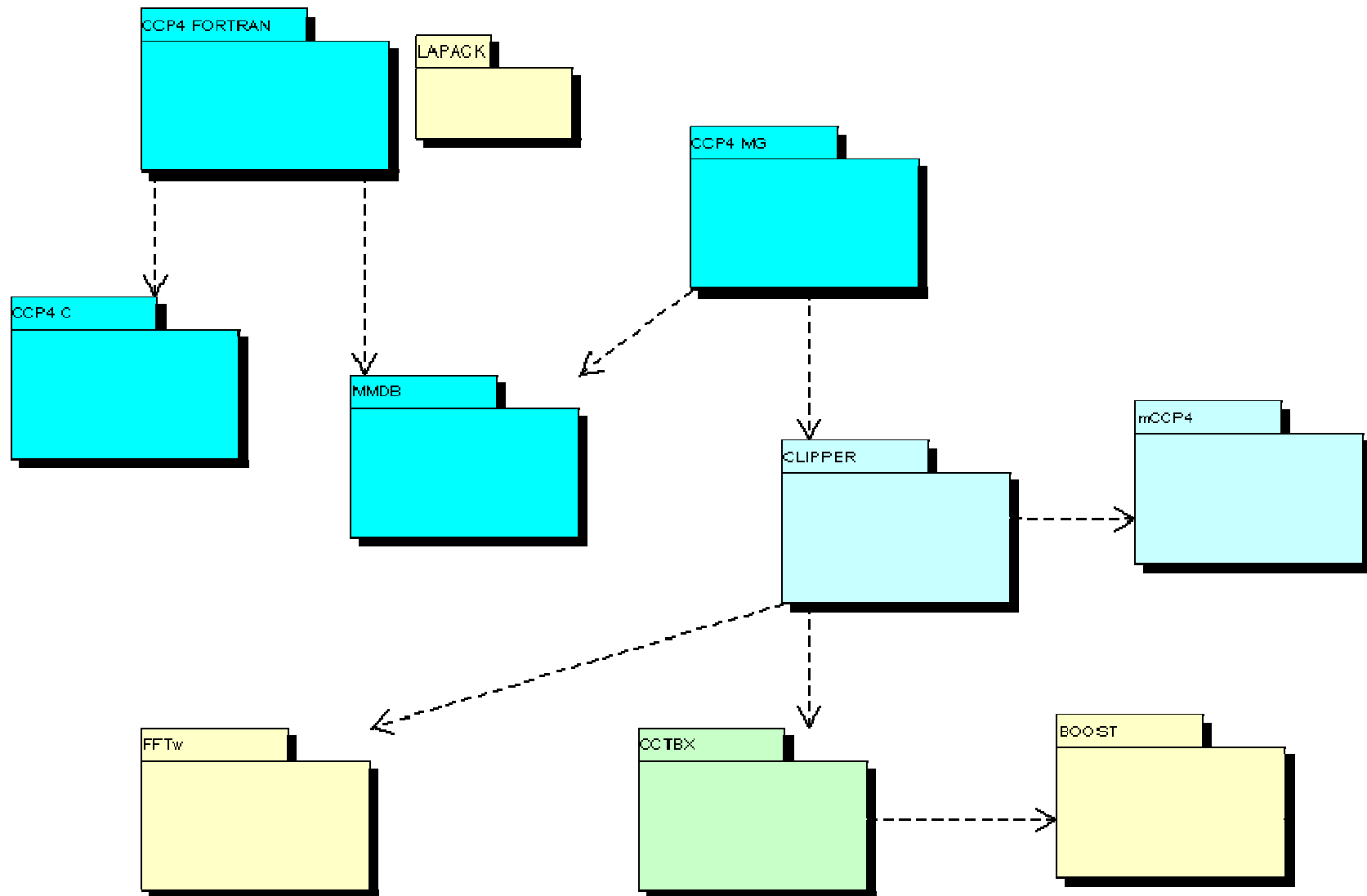
- The library should compile as a whole to give a `libccp4` which supports the current CCP4 release. To support other target uses listed above, it is necessary to allow partial compilation of subsets of the library. For example, compilation of CMTZ, CSYM and the low-level libraries should be sufficient to support 3rd party applications wishing to access MTZ files.
- Along with the source files, the library depends on a number of data files:
 - `syminfo.lib` holds spacegroup information
 - `atomsf.lib` contains form factors

- o `cif_mm.dic` is the mmCIF dictionary
- o `crosssec.lib` currently used by CROSSEC - move to library?
- The library has the following dependencies:
 - o `libccif` - low-level i/o for mmCIF

Future dependencies might include:

- o `fftw`
- o `clipper`
- o `cctbx`

Dependencies



Python

Python will be used in:

- CCP4 Molecular Graphics project
- CCP4i run scripts, replacing current tcl run scripts in suitable cases
- `cctbx` if this is to be a dependency

CCP4 is committed to the use of python, but the question of which version is still open.

Arguments in favour of 1.5.2

1. Increased likelihood of user systems having a suitable version of python.
2. CCP4mg aims to run with Python 1.5 (Stuart reports that CCP4mg also runs with 2.1 and 2.2).
3. We can migrate to Python2 at a later date.

Arguments in favour of 2.2.1

1. Integration with `cctbx`.
2. Some features of 1.5 are deprecated in Python 2 (but will presumably still work).
3. Nick Sauter lists the following useful features of Python 2:

C-speed support for regular expressions
 comparison operators can work on array types
 cross-platform support for pipes
 much better install procedures for third party add-ons
 new language features like iterators and generators

Applications

All legacy applications should work with the new libraries via the Fortran interface. New and/or converted applications include:

1. Pete's `ccp4mapwish` has been converted to use the `cmap` part of the new library. The source is currently in `$CPROG/ccp4mapwish_` under the CVS tag "newlib-dev".
2. Pete's `mtzdiff`, `mapdiff` and `pdbdiff`.
3. Eugene's `pdbcur` and `ncont` use CCP4 style functions as well as MMDB.
4. Martyn's `reindex` uses/will use `cmtz` and `csym` functionality.

Workplan

1. Test support of existing CCP4 suite.
2. Release CCP4 based on new library.
3. Lock APIs at some point.
4. Convert other sections of library (see "planned" above).
5. Write C applications.
6. Write Tcl applications.
7. Integrate with clipper applications.

Future directions

Possible future inclusions:

1. Reading of `atomsf.lib` and `crosssec.lib` (or updated versions of these files).