Project Name: CS^2 (Control System for Crystallographic Software)
Name of "pipeline": PyWarp has been developed using the CS^2 architecture
Author information: Marouane Ben Jelloul
Current Status: PyWarp works, but is not released to general public. A demonstration version of PyWarp+Phaser also works.

Purpose

CS^2 is a simple architecture that has been developed to help development of PyWarp, the new high-level control of the ARP/wARP 'pipeline'.
The aim is to integrate different programs in a decision driven process, in a general and transparent way.

High Level Description

CS^2 is made of Decision Functions, Controllers (wrappers around the programs) and the Step Manager (communication and information storage).

Decision functions decide which Controllers to execute and which Decision function should be called after a Controller has been executed. Decision function consult the Step Manager to harvest information they need.

The Step Manager thus keeps all information of the pipeline history. The Step Manager also keeps a 'dictionary' of control parameters that are initially input by the user but can be modified on the fly by Decision Functions.

When a Controller has been executed the output is stored in the Step Manager as a 'step'. We also plan that Decision functions also store their data in the Step Manager. Currently, Decision function can modify the dictionary of the Step Manager, but cannot add 'step' data to it.

When a developer wants to add a program to PyWarp only the Controller around it needs to be implemented.

Decision functions can be modified or added accordingly to the CS^2.

PyWarp Controllers include: Refmac, FFT, MapMask, ARP-update, MirBuild, Snow, Cubes, Pept_Hmain

PyWarp+Phaser also includes BLAST, Phaser Controllers

Jiffies

No Jiffies for PyWarp.

For PyWarp+Phaser two programs have been made; one to run BLAST and one to Get a PDB file from the Internet database.

A Python/Tk visualiser of the control system flow that present graphically selected contents of the Step Manager.

Decision Making

A simple scenario involves the decisions of either adding and removing free atoms and refining or to proceed with model auto-building. For that we compare two successive Rfactors.

```
# I know I come from Refmac step
previousStep = stepmanager.cStepList[-1]
# I want the Refmac step just before the last ARP
oldStep = stepmanager.cStepList[stepmanager.indexOfLastStep('ARP')-1]
# calculation of the RARPdiff
RARPdiff = oldStep['Rfact'][-1] - previousStep['Rfact'][-1]
# compare with RARPThresh
if RARPdiff > stepmanager.vdict['RARPThresh']:
    return (CFFTMapMaskController,CARPController),decisionEnterRefmac
# otherwise we go to CubesPept and we save the NbArpCycles
outOfARPCycles()
return (CFFTMapMaskController,),decisionEnterCubesPept
```

Data Standards and Management

After running a program (= a Controller) we store the information of the run as a step in the Step Manager. The Step Manager has some methods used in the decisions to use these information. The Step Manager is output as an XML file. If a problem occurred during execution it is possible to restart PyWarp using as input the Step Manager (and do not rerun all the programs up to the point that the problem occurred).

Languages

PyWarp is in Python. Every single program is executed as a sub process.

External dependencies

Python module Tk-inter for the viewer.

Context/Audience/Environment

The audience for the Control System tools we propose is developers.
The audience for the product (ie pyWARP) is any user.

Links to Supporting Documents

-

References

-