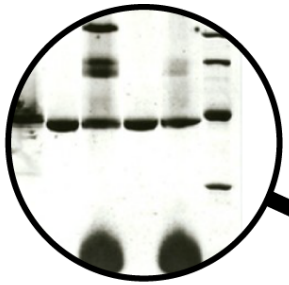
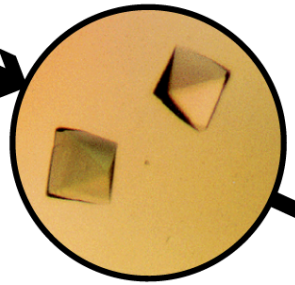
High-throughput Crystallography

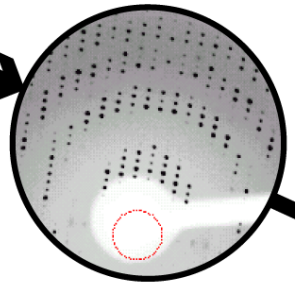
- The structural database must expand
- NIH funding centers to collectively solve several 1000 structures in next 5 years



Automated Protein Purification



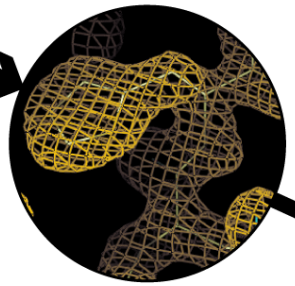
Automated Crystallization



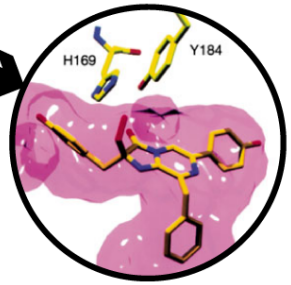
Automated Data Collection

PHENIX

***P**ython-based*
***H**ierarchical*
***EN**vironment for*
***I**ntegrated*
***X**tallography*



Automated Structure Solution



Interpretation

Structure solution will take hours - days

Technology benefits all crystallographers

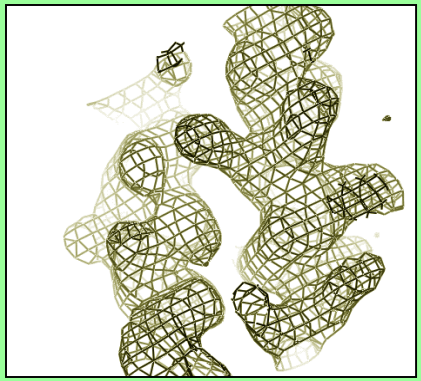
http://www.phenix-online.org

The *PHENIX* project

- **Goal: develop new crystallographic software for automated structure determination, using modern algorithms and advances in computer science**

- **The new system will be comprehensive and flexible enough to automate structure solution, but allow user interaction, input and control**

- **Automated crystallography at medium to low-resolution (3Å)**
  - **Maximum Likelihood methods**
    - **Phasing**
    - **Molecular replacement**
    - **Density modification**
    - **Refinement**
  - **Pattern matching for automated building**
  - **Simulated annealing refinement**



The *PHENIX* project

A collaboration between several groups

Computational Crystallography Initiative (LBNL)
-***Paul Adams***, **Ralf Grosse-Kunstleve**
-**Nigel Moriarty, Nicholas Sauter, DK Smith**

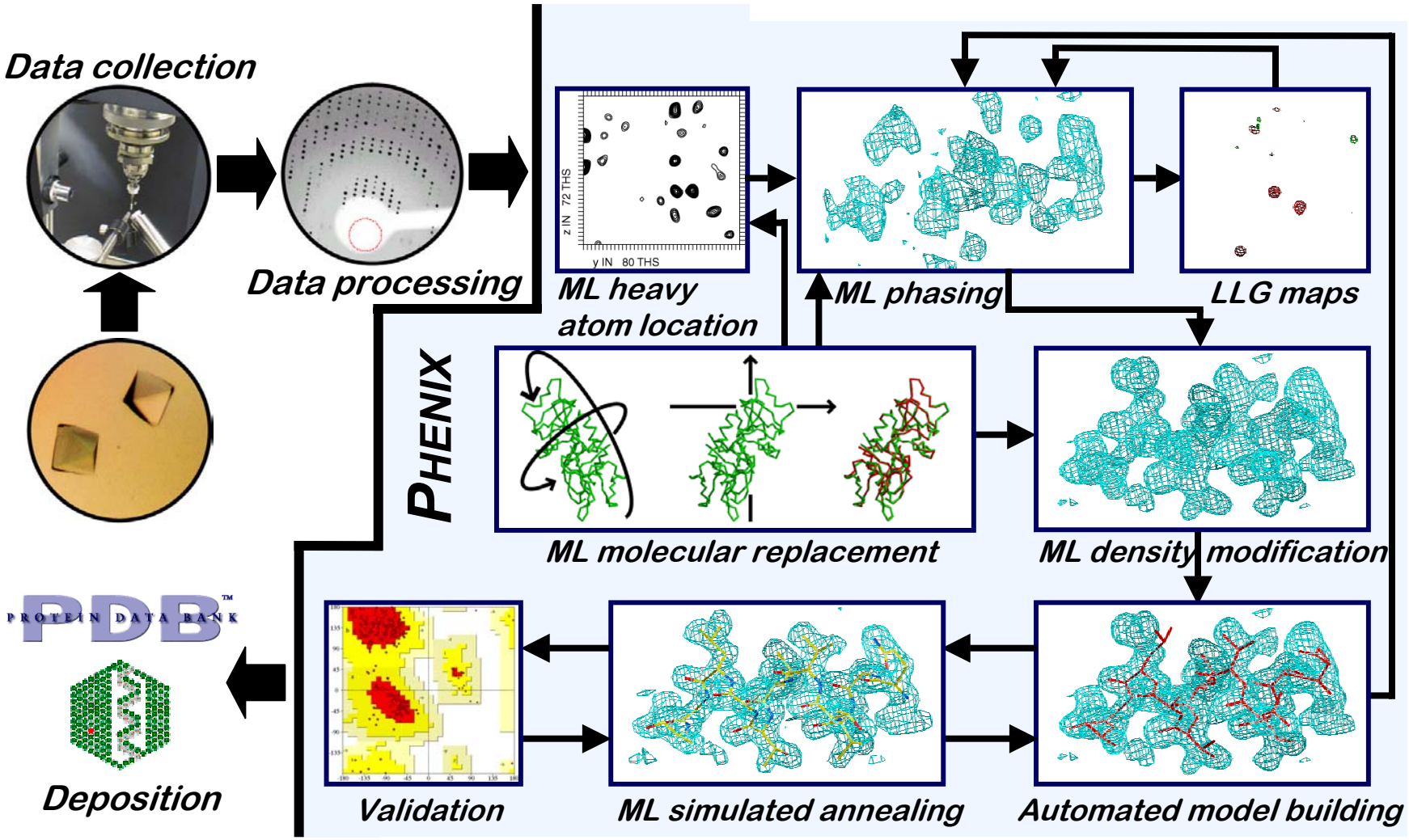
Los Alamos National Lab (LANL)
-***Tom Terwilliger***, **Li-Wei Hung**

Cambridge University
-***Randy Read***, **Airlie McCoy, Laurent Storoni**

Texas A&M University
-***Tom Ioerger***, **Jim Sacchettini, Kreshna Gopal, Tod Romo**

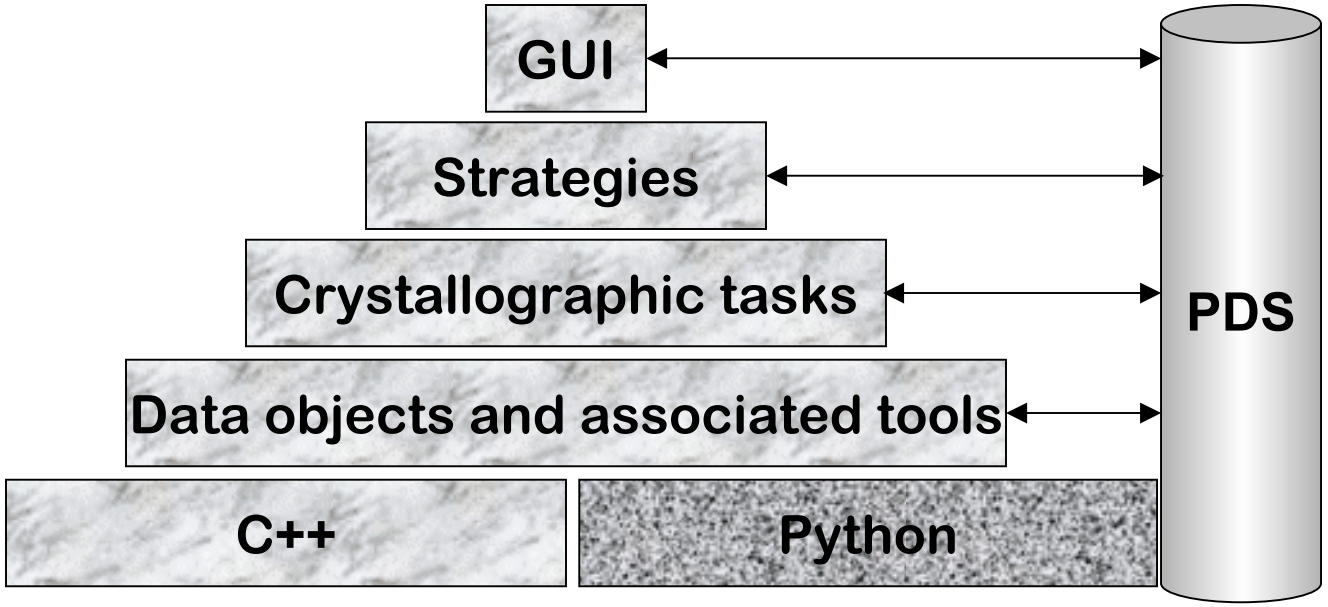
Funding: *NIH (NIGMS, PSI) Program Project (LBNL lead, PDA director)*

Automated Structure Solution



A Collaborative Framework

- **Flexible: can automate the whole structure solution process but also allow user input and control**
- **Built around a scripting language: efficient development of new ideas**



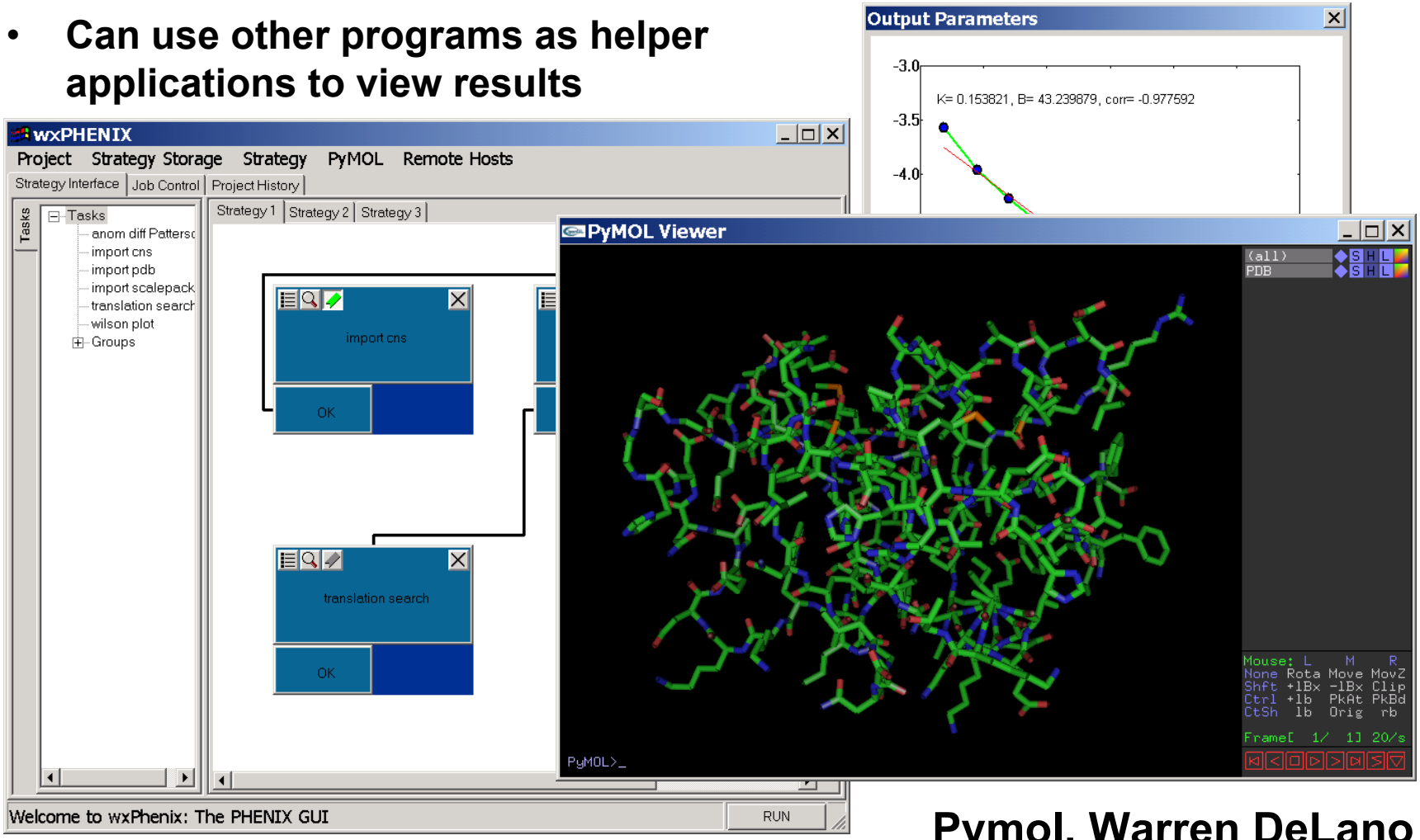
- **Project Data Storage (PDS): for each structure maintains a history of the structure solution and all the data generated.**

PHENIX Infrastructure

- Basic objects:
  - Structure factors
  - Map
  - Coordinates
- Visual programming
  - Strategies
- Project Data Storage
  - Type based data storage
- Project history browser
  - Linked to visualization tools
- Distributed computing
- Modularity
  - Object oriented programming
  - Dynamically linked objects
- Scripting language
  - Python
- Open Source cctbx project
  - Available to all developers
- Source code distribution to academic groups
- Other developers can build "plug-ins" that make use of *PHENIX*

Interfacing with the GUI

- **Can use other programs as helper applications to view results**



Pymol, Warren DeLano

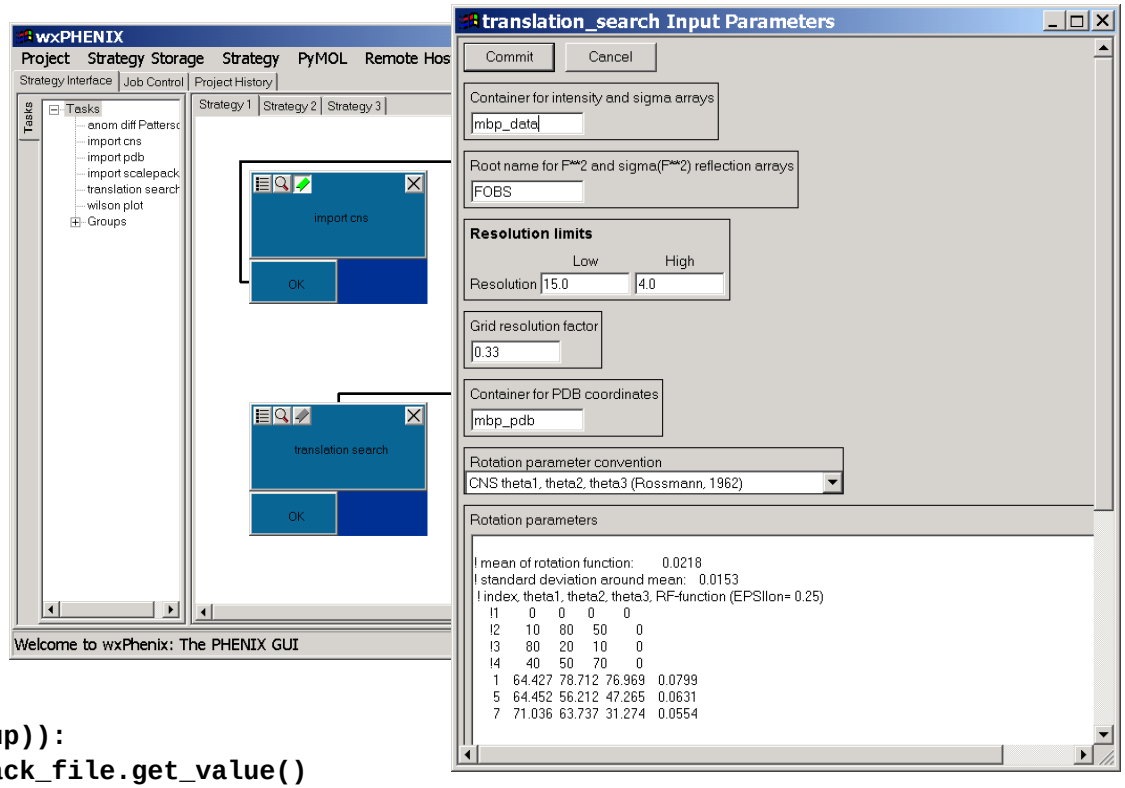
Developing new Strategies and Tasks

- Visual programming allows easy creation of complex algorithms
- GUI elements are constructed from Python objects (created in text files)
- Python task files could call other programs

```
def __init__(self):
  ...
  task_parameters = parameters(
    group(
      ...
      name = 'file_group',
      ...
      parameters = parameters(
        define(
          ...
          name = 'scalepack_file',
          ...
        ),
      ),
    )
  )
  task_branches = {...}

 def Run(self):
   inp = self.get_parameters()

   for i in xrange(len(inp.file_group)):
     print inp.file_group[i].scalepack_file.get_value()
```



Implementing algorithms - Python

```
#READ INTENSITY DATA
 S = xarray.ScalepackReader(scalepackfile)
 A = S.newIspace()
 S.loadIspace(A,"Iobs")
#CALCULATE AMPLITUDES
 Iobs = A.use_IntensityArray("Iobs")
 F    = Iobs.F()
 sigF = Iobs.sigF()
#SELECTION ON SIGMA (F)
 S1 = xarray.greater(F, SigCut*sigF)
#CALCULATE ANOMALOUS DIFFERENCES
 anom_diff = F.plus() - F.minus()
 sigdiff = xarray.sqrt (xarray.pow2 (sigF.plus())+ xarray.pow2 (sigF.minus()) )
#SELECTION ON SIGMA (delta F)
 S3 = xarray.greater(xarray.abs(anom_diff), DifCut * sigdiff)
#CALCULATE E VALUES
 ebin = A.Binner(8, 4.0)
 amp = xarray.sqrt(dfsquare)
 E = amp / ebin.average(amp)
 Esquare = E * E
#REMOVE ORIGIN
 Esq_rem = Esquare - ebin.saverage(Esquare)
#CALCULATE AND OUTPUT PATTERSON MAP
 proform = xarray.MapForm(A,0,4.0,0.25)  #0=HERMITIAN, 1=NONHERMITIAN
 pro = xarray.Xarray(proform,1)          #0=RealSpace, 1=ReciprocalSpace
 pro.loadArray(Esq_rem)
 pro.toRealSpace()
 pro.real2CNSfile(outputfile)
```

Python Script

- **The *PHENIX* scripting language uses Python, extended with C++**
- **Provides the convenience of a scripting language with the performance of a compiled language**
- **Many ideas can be tested first using Python, then optimized as needed**
- **The *PHENIX* scripting language provides access to crystallographic data objects, and tools (e.g. structure factors and structure factor algebra)**
- **No compilation is required to run this script**

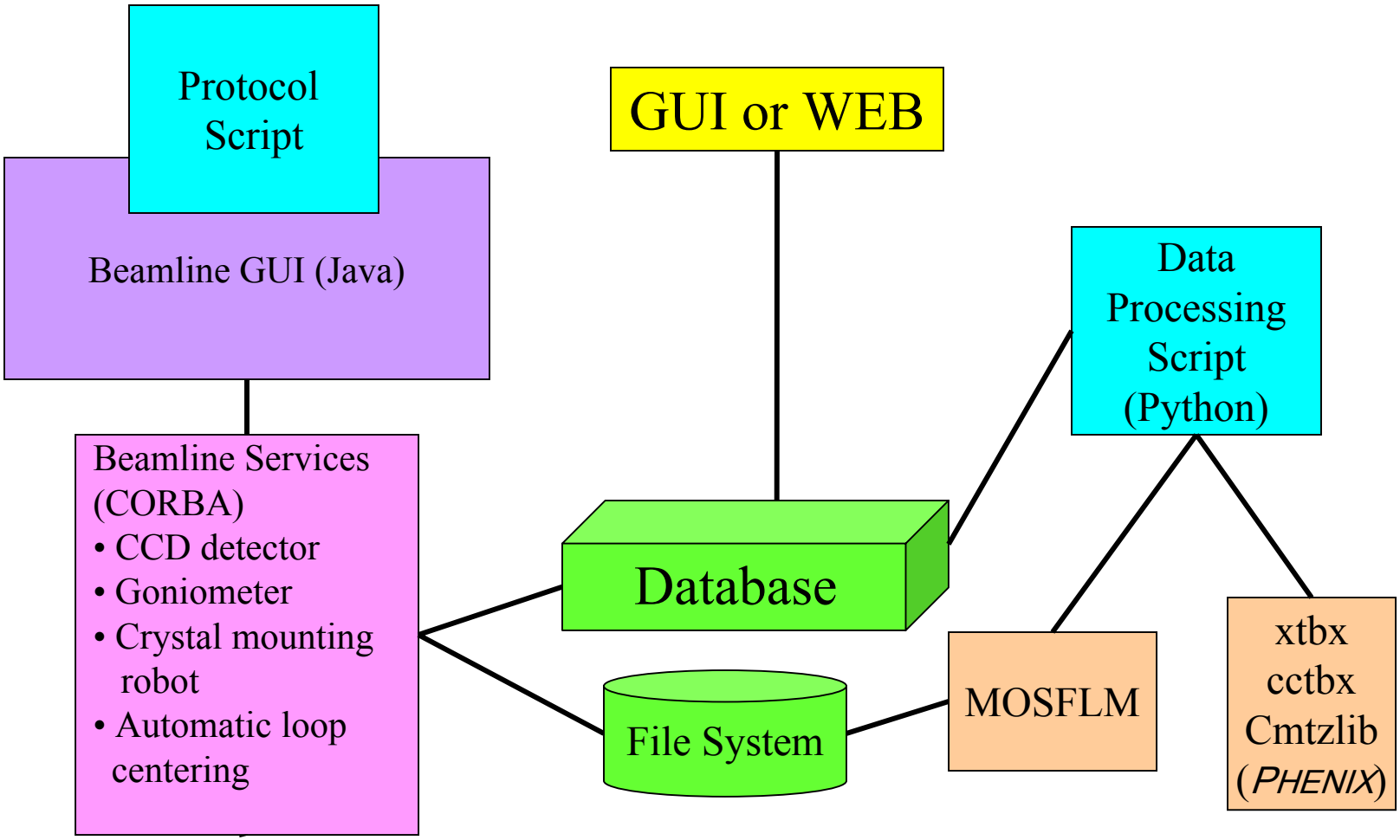
Implementing algorithms – C++

```
//READ INTENSITY DATA
 ScalepackReader S = ScalepackReader(scalepackfile);
 Ispace A = S.newIspace();
 S.loadIspace(A,"Iobs");
//CALCULATE AMPLITUDES
 IntensityArray Iobs = A.use<IntensityArray>("Iobs");
 IspaceArray F     = Iobs.F();
 IspaceArray sigF = Iobs.sigF();
//SELECTION ON SIGMA (F)
 IspaceArray S1 = Xalgebra::greater(F, SigCut*sigF);
//CALCULATE ANOMALOUS DIFFERENCES
 IspaceArray anom_diff = F.plus() - F.minus();
 IspaceArray sigdiff = sqrt (pow2(sigF.plus()) + pow2(sigF.minus()));
//SELECTION ON SIGMA (delta F)
 IspaceArray S3 = Xalgebra::greater(abs(anom_diff), DifCut * sigdiff);
//CALCULATE E VALUES
 Binner ebin = A.temporary<Binner>(8, 4.0);
 IspaceArray amp = sqrt(dfsquare);
 IspaceArray E = amp / ebin.average(amp);
 IspaceArray Esquare = E * E;
//REMOVE ORIGIN
 IspaceArray Esq_rem = Esquare - ebin.saverage(Esquare);
//CALCULATE AND OUTPUT PATTERSON MAP
 MapForm proform(A, HERMITIAN, 4.0, 0.25);
 Xarray pro(proform, ReciprocalSpace);
 pro.loadArray(Esq_rem);
 pro.toRealSpace();
 pro.real2CNSfile(outputfile);
```

C++ Program

- **C++ libraries implement crystallographic data objects and tools (xtbx, cctbx)**
- **C++ code can be used outside of the *PHENIX* framework**
- **The Boost.Python Library can be used to make the classes written in C++ available in Python**
- **Very similar coding with both C++ and Python**

Using *PHENIX* as an application



Nick Sauter and Thomas Earnest's group at the ALS

Summary

- The scope of the *PHENIX* software development is very broad:
  - New algorithms for structure solution
  - High level, generic tools (Strategy interface)
  - Crystallographic libraries (cctbx at SourceForge)
  - Low level "computer science" tools (Python/C++ interface)
- There are many opportunities for other developers to make use of the *PHENIX* infrastructure