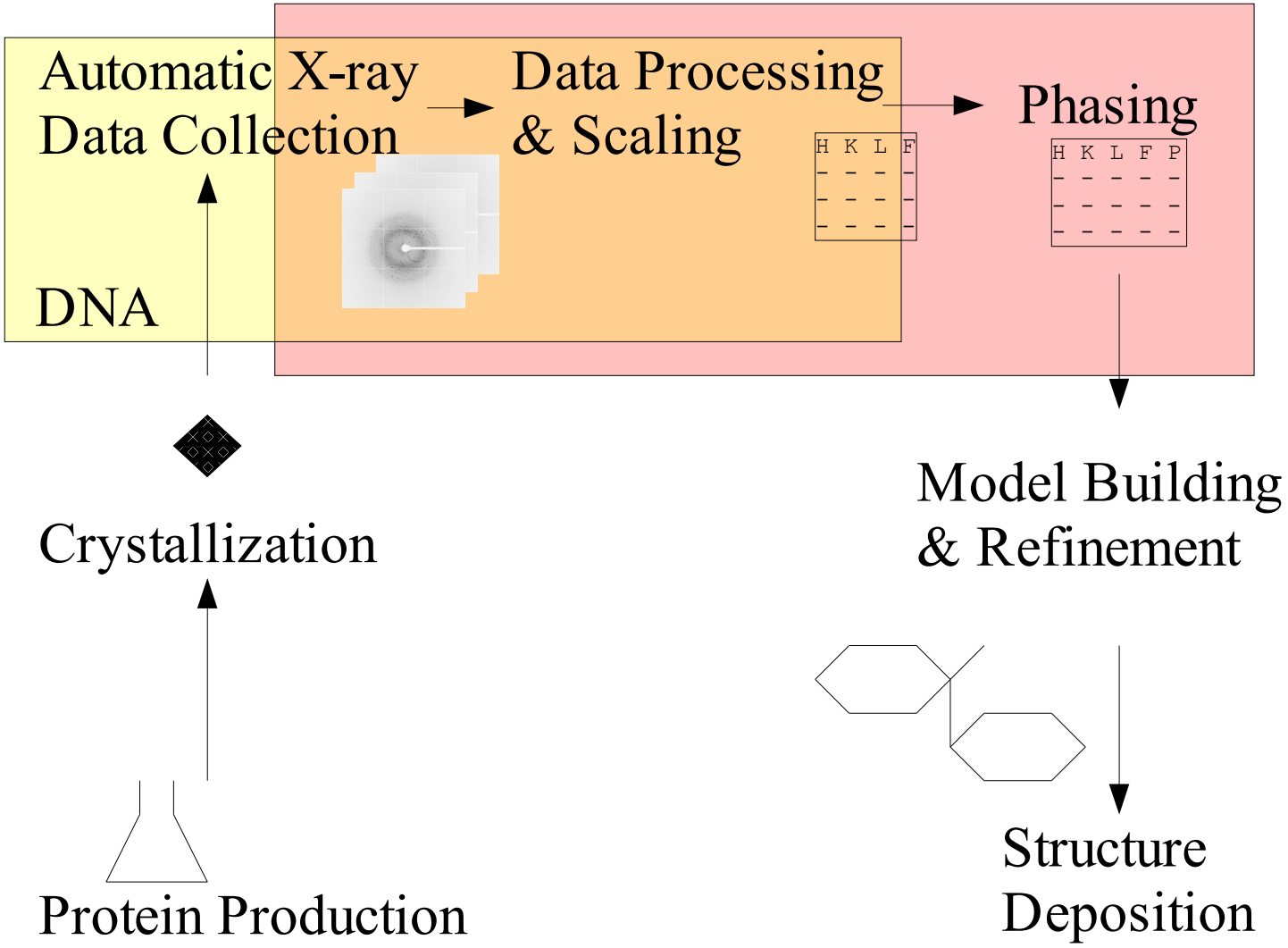
Software Automation in e-HTPX

CCP4 Developer Meeting
York University
2-3 March 2004

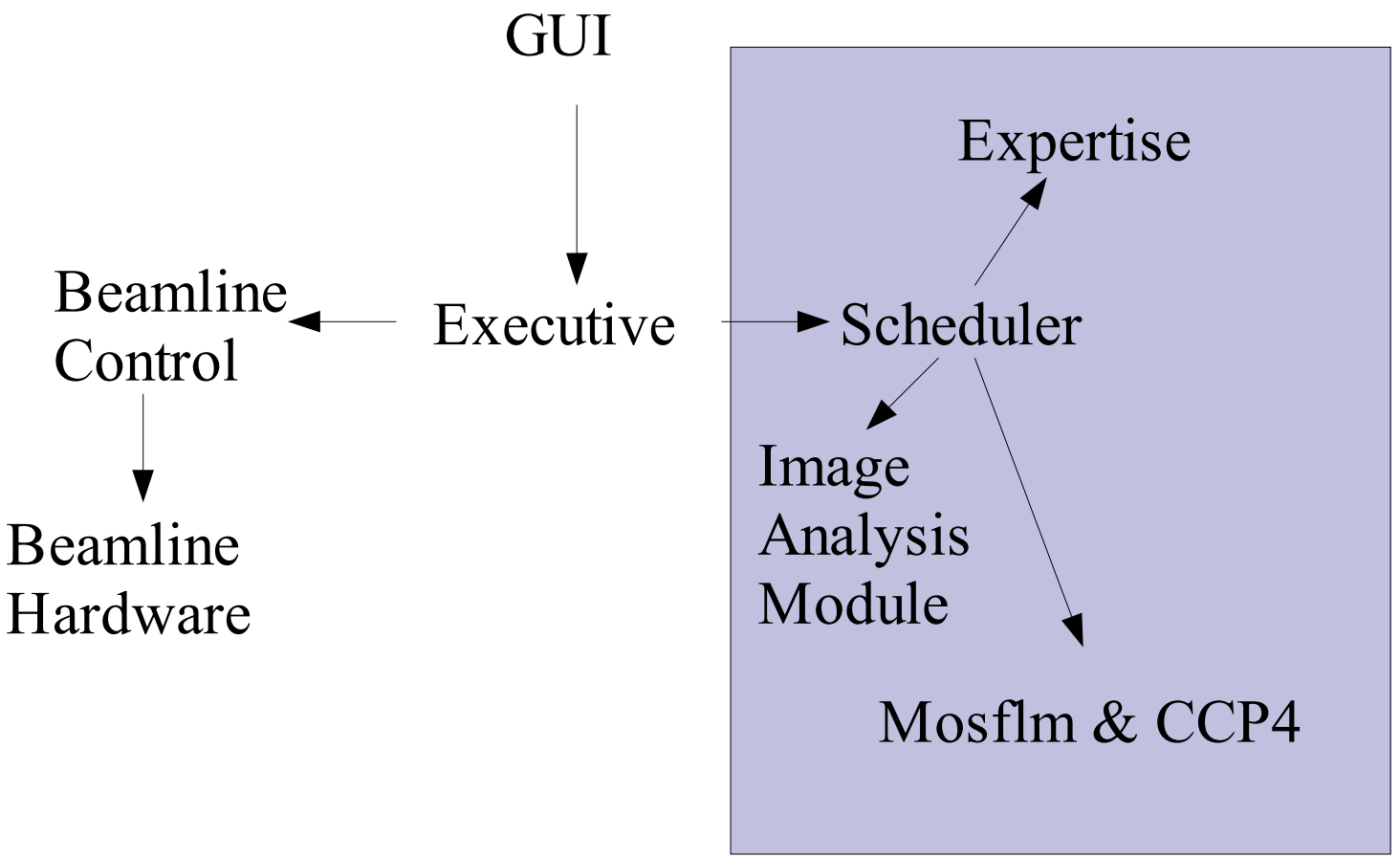
Contents

- e-HTPX Automation Work & DNA
- Relevant Components
- Architecture & Languages
- Implementation
- Progress to Date
- Future Plans
- Other Requirements
- Conclusions

e-HTPX


Automatic X-ray
Data Collection
Data Processing
& Scaling
Phasing
H K L F
H K L F P
DNA
Crystallization
Model Building
& Refinement
Protein Production
Structure
Deposition

DNA


GUI
Expertise
Beamline
Control
Executive
Scheduler
Image
Analysis
Module
Beamline
Hardware
Mosflm & CCP4

Relevant Components

The components which I believe are relevant to CCP4 automation are the scheduler & friends, in particular:

- Image analysis module
- Mosflm.py & so on.
- The Scheduler:
  - Driver.py
  - Scheduler.py
  - Background.py .. etc ...

```
import os

if not os.environ.has_key('DNAHOME'):
    raise RuntimeError, 'DNA not configured'
if not os.environ.has_key('CCP4'):
    raise RuntimeError, 'CCP4 not configured'

dna = os.environ['DNAHOME']
import sys

sys.path.append(dna + '/scheduler/Scheduler/Mosflm')
sys.path.append(dna + '/xsd/python')
sys.path.append(dna + '/scheduler/Scheduler/PxWeb')

import LocalDB, Crystal, Protein

import Mosflm
import XSD

m = Mosflm.Mosflm()
index_request = XSD.Index_request()

document =
"<?xml version='1.0'?>
 <!DOCTYPE index_request>
 <index_request> ... </index_request>"

index_request.unmarshal(document)
index_response = m.autoindex(index_request)
```

Architecture & Languages

Why Python?

- OS access is good
- “Clean” language
- OO structure, but not as design critical as Java

Architecture?

- Distributed Computing + Distributed Development = Modular Architecture

Architecture ...

- All components coded in same language (e.g. Java) can make communicating easy. Between languages can be difficult if not designed in from the start.
- Deciding on a common XML framework is painful but a necessary step.
- Languages are fashionable. Any solution must be updatable to the next fashion – like converting Tcl to Python.

Progress to Date

- Have added many xml writing statements to Mosflm, so that I can get all of the into I want. Note – this is **not** the dna XML!
- Mosflm to Mosflm.py binding is very tight, and very reliable.
- Have coded C++ modules to perform specific tasks, for example analying diffraction images.
- “DNA” expertise is coming along nicely.

Future Plans

Short term:

- Scala, truncate & feedback to data collection.
- Much more expertise ;o)

Longer term:

- See where the CCP4 automation goes!

Other Requirements

- It takes time to do this work.
- It takes a fair understanding of the process to develop expertise.
- A common framework is critical.
- Making use of computing resources – multi-CPU computers are common place, and wrapping code allows the use of this without recoding applications in some instances.

Conclusions

- Have already made some progress in automation. Most important aspect is probably reliability.
- Lots of people working in similar fields.
- Lots of expertise available.
- Shouldn't be hard developing automatic CCP4's.