Notes from a meeting held at the ESRF on 14th June 2001 to discuss interfacing Mosflm with data acquisition

Present:
Els Homan (EH), Olof Svensson (OS), Dave Love (DL), Steve Kinder (SK), Harry Powell (HP), Andrew Leslie (AL), Liz Duke (ED), Darren Spruce (DS) and Sean MacSweeney (SM).

Specific objectives set out at the end of the meeting:

- OS and DL are to put together the expert system based on the individual requirements at each synchrotron. It is planned that OS will visit DL at Daresbury in late July.
- OS will set up a mailing list (name to be determined)
- SK will set up a www site based at DL
- The next meeting is planned to be in Cambridge before the Autostruct meeting to be held in Cambridge around 20/21st Sept.

Meeting Notes:

1. Ultimate aim of the project:

SM started the meeting off with a summary of his hopes for the meeting. He hoped that as a result of the meeting ideas (which in many ways are longstanding ones) would move forward so that some form of useable concept could be created and that effort would be co-ordinated. Hopefully a certain amount of "criticism" and discussion would be generated along the way.

The ultimate aim of the project as a whole is to completely automate data collection and processing right from the mounting of the crystal through to the determination of the actual structure. Along the way a record of all the data associated with the structure would be recorded – i.e. data harvesting. Ultimately we are aiming to a user-free beamline.

In reality our desire for a completely automatic system will produce something which is initially semi-automatic – however even this is progress. AL asked about timescale for producing this semi-automated system linking data collection and processing – ASAP! was the response from SM. Timescales of the order of a year were felt to be too long. A discussion followed on what facilities would be available for testing – would official beamtime be applied for or would use of commissioning suffice. SM felt it would be possible to use some commissioning time at the ESRF plus a few days (1-2) per run from in-house research time could be used. This time would be used to try ideas out. Additionally there is sufficient slack within the system that this would be ok. However if it was felt that beamtime requirements were greater than this amount then an LTP (Long term project) should be submitted to apply for beamtime. The advantage here would be that any travel costs would be paid for (though it was felt possible that Daresbury could fund some travel to the ESRF if required). It might be politically expedient to be seen to apply for time officially – thus raising the profile of the project. It also sets deadlines for applications etc, which focus peoples' minds. If an application is to be made then it has to be submitted by September for beamtime

next year. Certainly, if regular on-line testing were felt necessary then official beamtime would be required. It was felt that beamtime at the level of a maximum of a few days per month would be all that was required. Alternatively willing users could assist in testing things out.
Work would be done to modify data collection software to provide information to the "expert" system – this would be unseen by the users.
One aspect felt to be useful would be the ability to screen (characterise) multiple crystals at the touch of a button. The next step would be to be able to process (integrate and merge) the data as they were being collected – this would show that the data could be processed and provide some form of data quality indicator.
The use of a "simulator" would allow progress to be made in terms of de-bugging etc without the reliance on beamtime. However some level of access to beamtime would be required if only to provide a reality check! It would also be possible to use data that have already been collected – perhaps even a library of test data sets.
DL said that he had already done something with DPS to index images.

2. Expert System Basic Definition and required commands

AL presented his views on the expert system:
In the simplest implementation 4 different functions (these describe processes rather than the single commands one has with mosflm) are required:

i. Characterise Crystal
   - Instruct the beamline to collect 2 images 90degrees apart.
   - Analysis of these images will provide information on cell parameters, symmetry, crystal orientation, mosaic spread and strength of diffraction
   - A table of results should be maintained in order to allow comparison with other samples so that some form of ranking could take place.
   - Some work may be needed to work out how to optimise certain aspects of the experiment such as beamsize, which affects I/σvalues – perhaps an option to collect a series of images at different slit settings would be necessary.

ii. Set up data collection run
   - Select sample from rank ordered samples
   - Obtain data collection strategy (contains aspects of "characterise crystal")
   - Determine resolution limit by examining I/σ values – possibly a series of exposures of different duration would be necessary. Alternatively an empirical approach could be adopted based on hard limits.

A discussion then ensued over potential problems with "auto-strategy" however overall it was felt not to be too great a problem as literally there is just a requirement to pass information to mosflm and a strategy would be returned. It was pointed out that some level of project information would be required at the start – such as the resolution required and also crystal information such as the minimum crystal size which would be useful for setting the slits.

- A choice of wavelength would need to be made – using a wavelength scan – possibly this would actually be done first.
- All parameters would be set/determined and then the beamline would be instructed to collect the data.

iii. Process a data set
   - Integrate a series of images – there are different ways of doing this – one could either collect all the data and then process them or one could do it as a series (i.e. in blocks)
   - Information needs to be fed back - is the data quality ok? Was the correct choice of symmetry made? If no, then the strategy would need to be re-evaluated. It is not really possibly just to collect 180degrees of data due to issues over radiation damage.

iv. Perform beamline alignment – though this is a requirement of the beamline control software.

Commands required of the data processing software would be:

- Index
- Find mosaicity
- Integrate (single image)
- Strategy
- Refine cell (collect and process data to get accurate cell – probably requiring more than 2 images 90 degrees apart)
- Process (multiple images)

Commands to go to the beamline control software would be:

- Collect (one or more images)
- Align (optimise beam)

For the index command, information to be passed would be:

- Project id
- Directory
- Filename template
- Combination of images used
- ϕ values used
- Cell and symmetry (if known)

Beamline parameters would be:

- Image header information: detector type, gain, pixel size, direct beam, beam divergence, polarisation, wavelength and distance.

Mosflm would return the following information:

- Project id
- Solution id
- Status flag (ie has it worked?)
- Cell
- Symmetry
- Orientation matrix
- Number of reflections used / rejected
- RMS deviation on spot position
- Updated direct beam

For the command to find the mosaic spread all the previous information would be passed to Mosflm plus also the cell, symmetry and orientation. Mosflm would then return an estimate of the mosaic spread.

The "integrate" command would similarly also require all the previous information and mosflm would return values of $I/\sigma I$ as a function of resolution.

The "Collect data" command would require:

- Resolution
- Exposure times
- ϕ angle(s)
- Oscillation angle(s)
- Number of passes
- Slit settings
- Wavelength
- Data directory
- Filename template
- First image number

Discussions ensued as to what the program might look like – it was highlighted that the program at the development stages would probably look different from the final version which is likely to be "black box" in style. For example, at the development stages in the GUI one would have a button for each stage. Information should be given on the status of the program and it should be possible for the user to accept or reject each stage and have the option of going backwards if necessary. This will be complex though it was felt not to be as complex as the mosflm GUI is going to be. However during the development it is important that the future is thought about – how the "black box" situation will be dealt with – perhaps it should be possible to see what is going on within the program but not be able to interact with it.
HP and AL said that they do not have time to do this "Expert system" but will ensure that Mosflm can cope with the information required via some form of interpreter. Ultimately we are working towards the situation where the program just works. At the basic level all this is is just a program - however there are difficulties with the precise definition of an "expert system" - is this just a program or does it acquire knowledge. Technically speaking an "expert system " is just a program plus knowledge – it does not learn, it just applies knowledge to reach a solution. What we are aiming for in the long term is some form of artificial intelligence but this cannot be even contemplated until the expert system is in place.
The "expert" system will have to be able to interpret commands such as "index" so that mosflm will understand. Ultimately other programs will also have to be catered for as well – such as denzo. To a certain extent in the long-term we wont "care" how everything is done as long as one gets hklF out at the end! At this stage we will focus solely on mosflm but we should (in principal) allow other programs to be bolted on.
SM asked how everything would be co-ordinated – who would do it, how would it come together and what level of co-ordination would there be with the different beamlines.

A discussion ensued concerning the details of what is going on. The command "collect" is sufficiently well defined by all beamline software – but beamline control software is different at each synchrotron. However sensible defaults could be set to make things easier – such as always collect the images for indexing at certain distance and exposure time would be one possibility. Alternatively one could have a database of set parameters associated with each project e.g. expected resolution etc. An area that could pose difficulties would be determining cut-offs - maybe they could be associated with each project id. E.g. if one had three autoindexing solutions, which one would one chose, where is the cut-off of acceptability? What completeness is required – 100% completeness isn't entirely necessary, but what is acceptable? Also, what about the basic assumption that the information the user has supplied is always correct, invariably it isn't! – The crystals could be mounted in the dewar the wrong way round, the derivative could be the wrong one (thus affecting choice of wavelength). In many ways dealing with these issues is something to aspire to rather than to sort out at the start.
It was felt that 50% of projects would be amenable to being tackled in this fashion – certainly the system should work on straightforward cases. It is possible that one way of reducing mistakes would be to have set parameters – such as the distance set to a certain value, wavelength fixed etc. This would allow greater confidence than if it didn't work it was due to the sample rather than the beamline or software.

3. Script-based system, prototypes and ideas

DL presented some thoughts and ideas that he had had. He started with the quotation " Plan to throw one away, you will anyhow" Fred Brookes.

- A malleable system should be ensured
- Appropriate components should be used to create the system
- It should be user hackable
- Its basis is scientific problems - procedures and decisions are required.

DL put something together in shell scripts in order to get something working. The rationale behind use of shell scripts is that they work and no GUI is required. They are generally understandable – at least as commands and it is possible to plumb together major program components with the addition of helper programs as necessary. For example a program has been written to extract header information from Q4 images and to convert the beam centre co-ordinates. Pipes have been used to slot things together – again because they work. An attempt has been made to minimise the number of parameter files so as to avoid confusion.
So for autoindexing the following has been created:
Autoindex [option to override parameters] <image>
The solution is output on stdout. A solution is assumed to be reported.
DPS was used on Q4 images rather than MOSFLM because at that time the DPS style indexing could not be used when MOSFLM was running as a background job. (However, Harry reported that this was now possible).

The DPS indexing required:

- Image header info

- DPS beam centre
- Spots (obtained by hacking dps_display)
- Dps_index (twice)
- Dps_bravais (symmetry / penalty)

DL then posed the question – what is a plausible solution, which one do you pick? AL replied the solution one would chose is the highest symmetry with the lowest penalty. The next stage is to get from the autoindexing solution to a strategy.
It was highlighted that if header information is to be used then work must done to ensure that the header information is ALWAYS correct. However with the different image formats – not all have the ability to be altered – the mar ccd image format causes problems – which is one of the reasons that the ESRF have gone to a database system.
A discussion took place regarding image headers and the information in them compared to what one could put into a database. At the ESRF ALL parameters go into a database in order that other programs can pick them up. All header information is ignored.
The "program" "elves" was discussed. This has been installed and tried out at the ESRF – it ran ok up to the solve stage and the formats of all the files appear to be ok. In elves autoindexing is carried out followed by integration and then a run of scala. The output is examined and if it thinks that the symmetry chosen earlier was incorrect then it will re-do earlier stages.
Also Marion Szebenyi is looking at increased automation of processing though no one appeared to know precisely what was meant by "increased automation".
The question was posed – do people keep up with processing the images as they appear. The response was that at the ESRF it is virtually impossible as exposures are so quick – 8 frames per minute was quoted as a usual rate. Even with distributed data processing (as used at the ESRF) it is difficult to keep up though this is with user intervention – when an integration is set going it is possible at the ESRF to integrate 120 images in 2 minutes. Certain aspects of profile fitting slow mosflm down – perhaps integrated intensities should be used instead of profile fitting, as this is quicker though is likely to lead to an unacceptable reduction in data quality. Given this effect, profile fitting will be the method of choice. The limits really are due to computer architecture and disk I/O.
Issues concerning radiation damage were discussed – how to monitor it. It is certainly not possible just to automatically collect 180degrees. Reference images would be one way to keep track of any damage – recollect these images periodically. Perhaps even in the first instance only integrate these ones and then, purely for diagnostic purposes.

4. Socket based communications – issues and questions

HP presented ideas to be developed into the new mosflm gui.

i. Introduction to sockets and why they are to be used:

- A socket can be viewed simply as an i/o channel
- They can be used to communicate between processes on the a local machine, LAN, WAN, internet
- It is easy to set up simple TCP/IP sockets in C, Tcl/Tk, Java etc etc

ii. Why use sockets with mosflm?

- The current mosflm GUI is integrated within the main code but uses xdl-view libraries that are obsolescent. It is inflexible and difficult to maintain.
- Sockets provide a convenient way to communicate between mosflm and an external GUI and they also add extra flexibility.
- They also allow processes to interact seamlessly.

iii. Sockets in Mosflm

- The new mosflm will follow this general pattern:

i. The server runs as a daemon, listening on Port X
ii. The User Interface (UI) client connects on Port X from host *hostname*
iii. Server creates a new thread (NT) of itself with a new port A.
iv. UI connects to NT on port A
v. NT starts a new mosflm process which listens on Port B
vi. NT ensures that mosflm is ready to receive commands
vii. NT listens on Port A, letting UI communicate with mosflm

In more detail:

- The server sends commands to mosflm as mosflm commands and sends messages to the UI marked up as XML
- The UI client sends commands to the server preceded by an XML tag such as:
  - <moscmd> is a "raw" mosflm command
  - <command> is a generic command – the server generates mosflm commands from this
  - <server> are server maintenance commands

This could serve as the translator module

It will always be possible to run mosflm from the command line and from the GUI. A long discussion ensued as to whether this was adding an extra layer of complexity.

The piping of images in jpg format rather than the image itself was also discussed. A query was posed about problems with reverse client server - but with not all processes running on the same machine it was felt to be better this way rather than spawning processes.

There was a long and unresolved discussion initiated by DL concerning whether security and reliability would be a problem as upgrades would need to be repeated in multiple places – sockets and commands lines are not the same thing therefore 2 different methods of communication would have to be maintained. The question was posed as to whether the same facilities are available in mosflm if you start it from a command line than if it is started from a socket. Currently if one starts mosflm as the command line it won't have sockets but ultimately it would have access to sockets.

Concerns were expressed over the fact that there were 2 ways of doing things – would one be able to do the same thing with each way? HP responded that it would be identical to the end user. It was added that there have always been 2 routes into mosflm and so far this has not caused a problem.

The daemon will have to run on every platform – however it has not been tried on VMS or NT (and NT sockets are done via a different method).

HP stated that it was hoped to have a working version by Christmas.

Mosflm sends messages to the UI either in mosflm format OR marked up in XML such as:

- <table cell> a b c α β γ </>
- OR images as jpegs – possibly b64 encoded - mosflm will generate the jpgs as opposed to reading the jpgs from an image server. A discussion ensued as to who or what would create the jpg files.

Display aspects are important in the first instance of automation due to the overheads involved. There are resource issues anyway associated with data processing and it was felt that trying to see what was going on should not interfere with data processing.
There is a need to learn to talk to the daemon in order to do things automatically. Any new client (user interface, expert system etc) will use this protocol:

- Connect to Port X on server
- Reconnect to port A (information supplied by server)
- Send commands to mosflm via server port A
- Read information from server and parse according to XML tags

5. The image server

OS presented the work that has been done at the ESRF using the image server. The idea itself came from SSRL where it is up and running. The server runs close to the data, it caches a number of images and serves the images back to the client in jpg or png format. IT is possible to zoom in and change the contrast on these jpg images. SSRL gave the ESRF the code, which is in C and makes use of threads. A student successfully wrote a python image server based on the server built in http in a short time.
Ipread is used which is based on ipdisp as a python module and reads in images into python with their associated spdfiles. It has been integrated into PXWEB so that now images in the database can be visualised. The next stage is to work on displaying overlaps, resolution rings and predicted spots and creating a java applet for zooming in and controlling the contrast. For the overlapping spots it is planned to look at the way mosflm treats them. (At SSRL they go back to adxv if they want to do anything more complex with an image such as looking at histograms). It is planned that people would use this as a standalone tool to monitor data on the beamline or to allow images to be examined from home. Ultimately this may well replace ipdisp. It will also be possible to link it in with information contained within the database.
This project has a wider remit that purely the automatic data processing.

6. Higher level communication

EH presented work that has been done on ZOPE, which is a framework for building web applications. It is being used to develop a web interface to data processing. It is possible to have both static and dynamic images and is object oriented.
Advantages are:

- Open source
- Object oriented
- There is a gui for the content manager
- It is robust, flexible and platform independent
- It is extendable by python

Zope has a lanaguage, DTML, to get information out of databases. DTML can be used in conjunction with, for example, python, to create objects. Additionally, because it is object oriented it is possible to add more things.
It is planned to use it for data processing via the web. It will be able to:

- Display data
- See what is taking place on the beamline
- There is a database of users folders for each experiment
- There is a database of statistics which is useful for the administrators
- Documents can be stored
- There are possibilities for scripting

Each user has a folder with a direct link to their data. It is possible to remotely process data, display images and have access to experimental information from the database – such as run information, parameters used and the actual data stored. This is done using MySQL database as the experimental database. It is a secure system where one can input a proposal or look up an existing experiment. Parameters are available for each beamline. It is possible to create new runs or look at old ones. Folders will contain parameters, lists of images (click to display) and processing information. The experimental parameters, which were initially created by proDC, can be displayed and modified.
For data processing there are script versions of the programs. The function "create" will create a new folder in the experiment database for each programme. There are agent folders containing unix scripts with the parameters filled in based on information in the user folder. It is possible to start a script singly or one after the other.
It will also be possible to do statistics of the beamline. Currently the database is purely for data collection, it doesn't contain any history.
A question was proposed as to how efficient the system is – what were the overheads of writing to the database and potentially serving several clients simultaneously. MySQL is a fast and high performance database. There are two other (ESRF specific) databases – one concerned with user office information and the other with beamline information. The beamline database is ESRF wide and contains a list of beamline components for diagnostic purposes. It was originally intended purely for the machine and then vacuum was included. Everything is done at the taco interface (TACO is the object oriented control system originally developed at the ESRF to control accelerators and beamlines and data acquisition systems.)to oracle, the taco interface is used to write the experimental parameters. Assistance was provided from the machine people who had experience of adding to databases and adding dictionary definitions.

Information is recorded automatically so that it can be extracted when required. Effusion is used to spread out jobs over a number of different machines.
There are 5 steps from creating a script through to the analysis. Python is used to open an mtz file and extract header information to put it into the next stage of analysis. It is unclear at present as to how far it will be extended into structure solution. Also there is a desire to increase flexibility so that non-default parameters can be used in data processing – the editing of scripts.

There are plans to automate various stages – for example in a MAD experiment, taking it through the various stages automatically.
Query: is this ultimately what we are trying to do – automate the complete structure determination?
At the minute nothing is actually done with the output from each stage. It is possible to examine the output but ultimately it could be parsing the information – extracting information to keep track of the project – compare this with the harvesting which goes on in CCP4. However SM admitted to being unsure of what the harvesting is actually doing in CCP4. However it could be used here.
Ultimately are we working towards automatic pdb deposition?
In many ways this is similar to what Jan Zelinka did a few years ago with zlbd but it didn't get anywhere – possibly because people didn't understand it.
Ultimately the ESRF are setting up to run automatically – to effectively have a continuously evolving notebook.
In some ways this is also similar to autostruct. Autostruct is a project to get different programmes to pass information between each other. IT includes non-CCP4 programmes. There plan to be higher level definitions of pathways between programmes. It is intended to do this in the context of a CCP4 gui look-a-like. However it was pointed out that by definition anything in a GUI is not automatic as at least 1 button must be pressed.
Basically at the end of the data we want to co-ordinate information flow and information gathering.

7. Knowledge-based system

OS presented some ideas on a knowledge-based system.

In the automation of the learning process one area which needs to be examined is the cut-offs for acceptance. In the past OS has had some experience with implementing "rules" using "fuzzy clips" in a project joint with Roberto Pugliese on beamline alignment. In 1997 the idea of using fuzzy logic was abandoned and a move made towards an expert system – JESS (Java Expert System Shell). In 1999 python was employed as it was felt to be more widely understood.

Clips are used when one has knowledge and an expert system. It resembles lisp – with lots of facts and lots of rules and many if/then statements. One ends up with a huge rule list that could only be used by the people involved in setting it up (in this case OS). A better situation is lots of facts and few rules – though this is not the case in the beamline situation.

In many ways PX automation is similar to beamline alignment and OS was reluctant to re-use this method mostly due to the poor relationship between the number of facts to rules. Also one would need to define all the if/then statements. HP pointed out that a system already exists within mosflm to chose a single indexing solution. However it was pointed out that denzo treats this problem differently – looking instead at the 14 Bravais lattices. It was pointed out that the intelligence is within the expert system and not in the data processing program. However in the more complex situations the filtering of a solution might not be too good an idea as one could end up filtering out

the correct one. In the first instance it might be sensible to produce a semi-automatic system so that one can work out when the wrong decision is being made so that further rules can be made to prevent this occurring in the future.

A system that just says index and waits for information to be returned is extremely powerful. The server would have the general command "index" which would be converted to something like "autoindex_dps ---- maxcell------" which is much more detailed. – ie what is required is either a generic command or something that is mosflm specific.

One could state the criteria that if indexing fails then junk the crystal. However if indexing does appear to fail then there are other aspects that should be considered such as the number of reflections used and rejected, and the rms. spot deviations. One could have a weighting system to assist in working out whether indexing has worked. This led onto considering what one should do with exceptions and then on to how one should report errors in a sensible way. If a sample is flagged as rejected should one just go on to the next sample or should the system automatically return to a standard sample so that one can be sure that the problem is related to the crystal and not to the beamline. It was felt that exceptions should be thought through however it is important to get something that will work now – perhaps there should be hooks in place so that one could allow exceptions to be put in at a later date.

8. Generalisation and Future Work

It was suggested that the immediate future work would be focussed on getting something up and running quickly which would give us something to build upon. One would take AL's suggestions for the initial functions of the expert system (Section 2) (eg. Collect data), add XML tags and put it into an XML type server so that "collect" can be recognised. The program would then instruct the data to be collected and wait for it to report the status back.

HP will put commands into mosflm with xml tags such as cell. The system "in the middle" would be able to read and interpret the information.

However what needs to be established is who does what now – jobs need to be split up with each demarcation between each job. There will be beamline specific items that can only be done by people at the respective synchrotrons however the in-between section should be transparent.

What is required is to take the output from mosflm and turn it into a command to collect data. One should assume that the image exists, tell mosflm to index the image – this will require talking to the server and issuing the index command.

HP and AL will provide cut-off values for various parameters. As the mosflm server is currently under major development at this stage this will probably be done by grepping information from the log file. It was pointed out that sockets would be easier and more consistent. The question was asked – in what form will the server provide information. HP and AL said that in approximately one weeks time it should be possible for mosflm to produce the information and read information back.

OS felt that it would be wise to have the expert system in python due to it being easy to use and there is experience of using it. DL pointed out that in python there is a limit on serious calculations – was that an issue? IT was felt not to be an issue in this case and in any case converters exist to convert python into C. Other alternatives were suggested by DL such as lisp as he has experience in it. However the final conclusion was to use python.

There are plans to simplify the output of mosflm and how it will come out of the program. There are issues over parameters and acceptance/rejection criteria as well as all the beamline related issues:

- Indexing – has it worked?
- Send strategy command – how are the matrices defined – the amat matrix will be passed back
- Integrate a single image –values for $I/\sigma I$ vs resolution will be passed back to assist in deciding on a strategy. In mosflm cut-off are done with $I/\sigma I$ of 3 and in scala it drops down to 1.5-2. It was decided that the effective resolution is where the value for $I/\sigma I$ drops down to 3.

Specific Aims:

- OS and DL are to put together the expert system based on the individual requirements at each synchrotron. It is planned that OS will visit DL at Daresbury in late July.
- OS will set up a mailing list (name to be determined)
- SK will set up a www site based at DL
- The next meeting is planned to be in Cambridge before the Autostruct meeting to be held in Cambridge around 20/21st Sept.