The School is intended for people with a knowledge of crystallography who may want to become involved in software development, but lack experience in this.

This is an outline programme for the 2005 IUCr Computing School, scheduled to be held near Siena. Nothing here is carved in stone (as of 12th November 2003); it is important, however, to get something on paper which can then be modified to suit the needs of the school as seen by the members of the IUCr Commission on Computing. It is also probably important to have a list of topics/study areas decided before picking speakers; most topics "suggest" lecturers quite readily. It is, of course, quite possible that there will be a few "obvious" lecturers who may be omitted by this process, and they should be accommodated somehow in the timetable.

My feeling is that there is more to be gained from not dividing up so much into different types of crystallographers. While the details of their crystallographic fields may differ considerably, the approaches to software developments are actually much closer. Also, of course, many of the algorithms employed are similar, as is (are?) the mathematics.

However, splitting into topic groups during days 3 and 4 (21st and 22nd August) is certainly possible, and might appear more natural.

The first draft is a list of ideas which have come up during e-mail discussions and have been given priorities according to my own personal preujudices.

Harry Powell, November 2003
### Siena Crystallographic Computing School 18th – 22nd August 2005

<table>
<thead>
<tr>
<th>18th August</th>
<th>19th August</th>
<th>20th August</th>
<th>21st August</th>
<th>22nd August</th>
<th>23rd August</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;principles &amp; methods&quot;</td>
<td>&quot;joining things together&quot;</td>
<td>&quot;crystallographic implementations&quot;</td>
<td>&quot;selected topics in crystallography&quot;</td>
<td>&quot;special methods&quot;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>07:30 − 09:00</th>
<th>breakfast</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 − 09:45</td>
<td>procedural programming</td>
</tr>
<tr>
<td>09:45 − 10:30</td>
<td>scripting languages</td>
</tr>
<tr>
<td>10:30 − 11:00</td>
<td>coffee</td>
</tr>
<tr>
<td>11:00 − 12:00</td>
<td>OOP</td>
</tr>
<tr>
<td>12:00 − 13:00</td>
<td>maintenance &amp; development of legacy code</td>
</tr>
<tr>
<td>13:00 − 15:00</td>
<td>lunch</td>
</tr>
<tr>
<td>15:00 − 16:00</td>
<td>complete rewrites – when and why?</td>
</tr>
<tr>
<td>16:00 − 17:00</td>
<td>introductory talk</td>
</tr>
<tr>
<td>17:00 − 18:00</td>
<td>Porting to and from Linux</td>
</tr>
<tr>
<td>18:00 − 19:00</td>
<td>Numerical Recipes</td>
</tr>
<tr>
<td>19:00 − late</td>
<td>case study 2</td>
</tr>
</tbody>
</table>

### Other ideas:

- using available tools, e.g. fftw, Lapack, cctbx, ccp4 libraries
- internal models
- platform dependent issues – porting, file formats, languages
- testing, debugging, feedback from users
- comparison of different methods – 2D vs 3D integration, OOP vs PP, etc...
- sources of funding
**Details of time slots**

**Workshops/projects**
Several hours each day will be devoted to software projects for the students. My inclination is to give a student the chance to develop their skills in applying one (or more) of the main methods discussed in the lectures to a particular crystallographic problem. This needs more thought...

**Case studies**
An opportunity for some participants to present either some work in progress or a completed project with relevance to the objectives of the school. Should this be primarily intended for teaching staff presentations or those by students?

**Day 0**
Essentially a day for travel and arrival, starting the work towards the late afternoon and evening.

- Some kind of introductory talk to the School – giving background to the site, teaching room arrangements, computing/networking facilities, general goals of the School.

- **Porting to and from Linux** can be a major issue, particularly for academic developers. By and large (but certainly not exclusively), small molecule crystallographers develop under MS−Windows using commercial compilers, but macromolecular crystallographers tend to use UNIX – in particular Linux, using the "free" GNU compilers. The development environments used differ tremendously – the Visual Studio is often used on MS−Windows, but Makefiles and plain text editors (e.g. xEmacs) are preferred under UNIX.

- "**Numerical Recipes**" are books available in several commonly used programming languages (FORTRAN, C, C++, Pascal, BASIC, Lisp, etc...) which contain coded routines for many common mathematical methods. A familiarity with the contents of these books means that much "re−inventing of the wheel" can be avoided.

**Day 1**
The general plan for the day at present is to introduce current methods of software development with the emphasis on crystallographic computing and its requirements.

- **Procedural programming** is the approach adopted by more "established" members of the community. Although it is currently not very fashionable in software development circles, programs written using these techniques (eg Structured Programming) form the backbone of our output. For a crystallographer who is a novice programmer, the method is much easier to get started with and to appreciate than more modern methods. It is important to be aware of the methods and issues involved.

  example languages: FORTRAN, C

- **Scripting languages** (eg Python) have been developed over the past decade or so to provide a useful tool for rapid prototyping of many pieces of code. Recent implementations of these languages produce code which is so fast that the resulting code need never be translated to traditional compiled languages.

  example languages: Perl, Python, csh, bash

- **Object−oriented programming** developed in the late 1980's as a formalized way to develop modular code. Although it has taken some time to be adopted by the crystallographic community, it is now
used in a number of major developments from both the academic and commercial software providers.

eample languages: Python, Java, C++

- **Maintenance & Legacy Software.** The bulk of crystallographic software running today originated from code originally written many years ago; there are examples of widely used programs which still have valid FORTRAN−66 code! Many of the programmers involved in the development and current maintenance are nearing retirement age. If the use of these programs is to continue, it is necessary for the code to be maintained and developed, probably by newcomers. The issues involved with this process are not only those of being able to program; there is also a depth of crystallographic knowledge ("tips and tricks") contained in the code which needs to be deciphered.

- **Complete Rewrites.** There are occasions when a piece of software cannot be maintained and developed any further; this often happens after many features have been added and the code has turned into "spaghetti", if compilers for the language used become unavailable or prohibitively expensive, or if it is realized that the original choice of language is in some way limiting. Under these circumstances, a complete rewrite may be the only option. It is vital to understand the issues involved before undertaking this process.

**Day 2**
There are many programs and routines which can be joined or linked together in a variety of ways. This day's topics cover some of the ways in which this can be done under different environments and for differing needs.

- **Program suites.** Some projects combine many individual programs together. Fundamentally, there are two ways of achieving this. The programs can use common io formats to allow the easy exchange of information, or some kind of parsing feature can be introduced which uses each of the programs' original syntax. Both methods have been adopted, and each has its own advantages and disadvantages.

- **Remote operations.** The remote operation of crystallographic hardware or software requires reliable communications using some kind of network protocol.

- **GUIs.** Designing Graphical User Interfaces (GUIs) introduces a design element into Crystallographic Computing. A well−designed GUI is not something that happens by accident. Also, the method of communication with the underlying software is important – does the GUI run as a separate program, is it an integral part of the main software, etc? Also, there are more platform dependent issues with GUIs than with other programming problems – *eg* MS Windows does not provide an easy way to use a GUI written using X−Windows protocols.

- **Automation.** The trend in many large laboratories is towards automating the entire process of crystallography. This subject is multi−facetted and requires some attention (issues of reliability, validation, deposition of results, etc, etc...)

**Day 3**
In my current timetable, no crystallographic problems as such are introduced until the third day of the School; this is quite deliberate – as noted above, I don't see that it is necessarily a good idea to split the School into powder/chemical/macromolecular crystallography prematurely. Even the topics I have pencilled in for today are inclusive.

- **Machine control.** How exactly do you control a diffractometer? How does the software for an image
plate with a single phi–axis goniostat differ from that for a CCD with a kappa goniostat? Or a chi
circle? Or a six– or eight circle machine? What are the underlying considerations and how do
developers treat them?

- **Integration of intensities.** There are basically two ways of integrating diffraction images, *ie* 2D and
3D methods. How do these differ, and what are the computational requirements of each? Other items
which could be considered here are autoindexing methods (difference vector, FFT...), refinement of
parameters, optimization of measurement "boxes", etc.

- **cctbx.** The cctbx ("Crystallographic Toolb") is an open–source initiative which aims to provide many
common crystallographic tools in an easy–to–use OOP package. The idea behind this is that if a
function (*eg* symmetry operator geneartion/tabulation) has been encoded once in an efficient way,
there should be no reason for countless other crystallographer programmers to do it again – they
should concentrate on developing new methods and algorithms.

- **LIMS.** Laboratory Information and Management Systems (LIMS) become important as the
throughput of crystallographic laboratories increases to unprecedented levels. LIMS couple
equipment to computers and databases to give seamless recording of (and access to) essential
information about an experiment.

- **Databases & Query Languages.** Hand in hand with LIMS go databases. What types of databases
exist? How do they differ from databanks? Use of SQLs (Structured Query Languages) – pros and
cons of Oracle, MySQL, etc... Access to databases on the Internet via PHP protocol.

---

**Day 4**

For the first time, topics which might be considered to be specific to different areas will be dealt with. This
gives the opportunity to have parallel sessions, perhaps, but even so, much of the underlying themes should
have broad appeal.

In particular, if we have parallel sessions, they should be designed so that they will necessarily appeal to
different constituencies – there is little worse than the thought of a partipicant that they have gone to the
"wrong" session.

The observant will notice that this is more of a mélange than the previous days!

- **Direct methods.** How do you begin coding something like direct methods? Perhaps something on
probability, invariants, etc. What about random start methods?

- **Refinement.** L.S. versus maximum likelihood – when are the two methods different, when the same?
Why use one rather than the other? What about Fourier methods? It may be that methods of coding
some statistics here might be useful.

- **CIF.** The Crystallographic Interchange Format was conceived about 15 years ago, and has been
adopted across the science in one form or another (CIF, mmCIF, powderCIF, imgCIF...); how do
programs deal with the files? What are they best used for, what are the limitations? Are there better
modern alternatives (XML?).

- **Crystallographic graphics.** The use of appropriate computer graphics forms a vital part of any
structural analysis – whether it's a polyhedral representation of a Zeolite, a thermal ellipsoid plot of an
organometallic, a ribbon diagram of a protein, a contoured density plot from a charge density study,
it's the part of a structural publication that everyone will look at and (probably) be able to understand.
It's certainly worthwhile spending a little time explaining how it's done.
Siena Crystallographic Computing School 18th – 22nd August 2005

• Analysis of Results. Structure validation and analysis session – see CCG Autumn Meeting 2003.

Day 5
A light morning’s session on some special topics that don't necessarily fit in anywhere else.

• Artificial Intelligence and Expert Systems. Applications to automation, validation, etc etc.

• something good here to finish the School with a bang.