CORE
Metadata, citation and similar papers at core.ac.uk
Provided by e-Prints Soton

The Schools Malaria Project

Robert Gledhill, Sarah Kent, Brian Hudson, Jonathan W. Essex and **Jeremy G. Frey**

School of Chemistry, University of Southampton

Abstract

The Schools Malaria Project (http://emalaria.soton.ac.uk/) brings together school students with university researchers in the hunt for a new anti-malaria drug. Malaria kills a child every thirty seconds, and 40% of the world's population lives in countries where the disease is endemic. Resistance to existing drugs is increasing and there is a growing need for new compounds. This challenge is being offered to school students who will use a distributed drug search and selection system via a web interface to design potential drugs. The project will display the results of the trials in an accessible manner, giving students an opportunity for discussion and debate both with peers and with university contacts.

Fig 1. Female *Anopheles Gambiae* mosquito - a malaria vector

1. Introduction

It is generally acknowledged that the public reputation of science is poor. Science is perceived as boring, hard and irrelevant to people's lives. The numbers of pupils choosing science courses in schools are falling, school science teaching can be uninspiring and the decline is numbers is worrying for the science community and society at large. To address this difficulty, we have, as part of a project jointly supported by EPSRC (CombeChem) and JISC (e-Malaria), developed an educational tool targeted at drug design for malaria. Diseases such as malaria, while being unprofitable to "big pharma", make good choices for academic outreach projects.

The e-Malaria project is about using e-science techniques and technology to make modern chemical ideas and practice available to a much wider audience of less experienced scientific investigators: school students, the investigators of the future. It also serves as a device to connect school students with active researchers; a virtual organisation for scientific research.

A target protein was chosen, DHFR, which is an enzyme involved in the regulation of DNA transcription, and for which there are known differences in action between the malaria parasite and humans. A crystal structure of DHFR provides a guide for the structure of the active site in the enzyme; we wish to assess the ability of a chosen small drug molecule to bind to this. This is known as a docking study.

An integrated software environment combining web design, database development, and distributed computing has been developed. The software is aimed at A-level students of chemistry; the students are asked to design chemical compounds using a sketchpad that they can then submit for docking against a known malaria target. The score from their docked structure may then be used, together with molecular graphics, to further refine their potential drug. This software teaches the elements of molecular structure and intermolecular forces, with the added driver of targeting a serious illness.

Further development of this project through the South Eastern Science Learning Centre based at Southampton is planned. At a system level, software to be used by school students has to be designed differently from one to be used by university researchers. It must also be robust and the project has taken this on board and presents valuable lessons in how to achieve the secure integration of industrial strength programs into a 'free' outreach environment.

We ouselves have learned many important lessons about what is achievable using the new methods of web based systems design, and how to go about doing these things effectively. Most of what we have done boils down to plumbing together already-existing applications using already-existing standards and constructing a robust and intuitive web

based interface to hide the warren of piping behind. Some detailed notes on what we have learned are given at the end of this report, but the main points are:

- Complex systems can be constructed using software on different machines using simple HTML communications protocols.
- Using Javascript, Java applets, CSS and CGI scripting it is possible to construct functional, elegant and intuitive user interfaces to complex systems. The only caveat to this is the need to work around problems in one particular web browser.

1.1 Description of Website Features

On first entering the site the user sees an introduction page containing a picture of a mosquito, a few paragraphs about the disease and this project, a few links and a list of system requirements for taking part in the project (Currently tested on Windows (IE and Firefox), Linux (Firefox and Konqueror) and Apple (Safari)).

The site has been designed so that anyone with a web connection can access the teaching material we have prepared without needing an account. A series of links to the teaching material is shown in the navigation bar, sorted under three headings: the project, malaria, and chemistry.

From the point of view of the pupil the drug design workflow can be thought of as having eight stages, as shown in Figure 2.

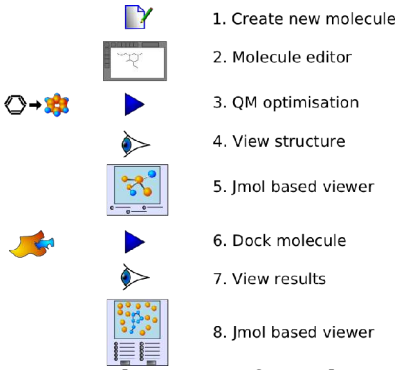

Fig 2. Schematic of Emalaria user workflow.

When a student starts, the molecule table screen will show the 'create new molecule' icon (1), along with a dialogue explaining what is going on and prompting them to click on the icon. Doing this will take them to the molecule editing page (2), where they can construct a molecule through freehand drawing, or by using a library of pre-made molecule fragments. On submitting their molecule from this page, the molecule is entered into the system along with a geometry optimisation/energy minimisation job. Clicking the activate button (3) puts the job into a queue for execution. When compute resource becomes available a two stage minimisation process is run to obtain a reasonable three dimensional structure for the molecule along with electronic point charges. Once finished, the results are entered into the molecule database alongside the initial 2d structure entered by the student. The user is then prompted to examine the calculated 3d structure and charge distribution (4,5), and then activate a docking job (6). Docking in this context means finding an energetically good geometric fit for the molecule in the active site of the target enzyme. After docking has finished, the best structure obtained is inserted into a results table on the same page as the molecule table. The user is encouraged to look at the energy numbers obtained from the docking calculation and then click on the view icon (7) to examine the results in a 3d display application (8).

On our present hardware, a QM optimisation job takes from a few seconds to 2 minutes to complete, and docking jobs generally take up to around 5 minutes.

The 2d molecule editing application was created from client side Javascript and the SDA labs Java molecular editor applet [ACD2005]. It contains just enough tools to get the job done with little extraneous content, and is also free for us to use on the site. It interacts well with the Javascript code in place on the page, and creating the application was reasonably straightforward.

A pair of 3d molecule viewing applications, for 2d/3d minimised and 3d docked structures, was created based on the open source Jmol [Sourceforge2005] molecular viewing Java applet. This proved clean and simple to work with and it was quick and straightforward to construct the applications.

2. System Description

2.1 Hardware Used

e-Malaria depends on a number of components sited on different machines working together in a seamless manner. The web server and the primary database are sited on a dual Xeon machine, 'Green', with approximately 1 terabyte of RAID disk space; a second similar machine, 'Purple', has been set aside for the United Devices subsystem which manages the multiple remote machines that perform the docking jobs. A

further pair of machines have also been set aside for local execution of docking and quantum mechanics jobs. Finally, (at the time of writing), a low-powered workstation PC has been put in place to act as a firewall. Further computing power for running docking jobs will be made available from the University's Science Learning Centre [Southampton2004]. This system is shown in Figure 3.

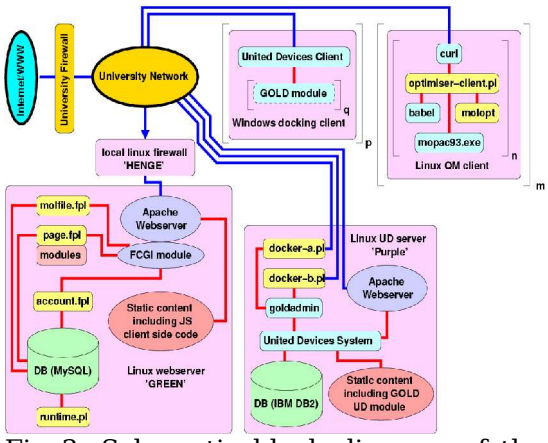

Fig 3. Schematic block diagram of the Emalaria system. Labelled square brackets indicate opportunities for application- and machine-level parallelisation.

There are two main problems with the above architecture:

1. I/O bandwidth is limited to 100Mbit/sec by the network cards in the firewall
2. Only one machine (the webserver) is made externally visible; the UD server cannot be accessed by external machines.

There are plans to upgrade the firewall and make the UD server visible once security concerns have been satisfied.

2.2 Firewall

The e-Malaria project requires a server that is visible outside Southampton University's firewall. This obviously creates a potential security hazard. To make any machine available to the outside, University regulations require its managers to take responsibility for any security issues. To protect against certain types of breaches it was decided to implement an internal firewall solely for the projects' server.

While the University's firewall will protect us from the bulk of the service scanning attacks taking place over the internet, what it cannot protect us from is errors or omissions in the security of the publicly exposed software running on the server machine. Of particular worry are the Apache web server, the database and, critically, the CGI programs operating the various public services provided on the server. Even with the most dedicated patching regime, the possibility of the machine being compromised remotely cannot be ignored. If this were to happen, not only would the data and programs on the machine be at risk (hazardous to the projects), but much more importantly, the machine could be used as a base for further attacks on other machines within the University firewall. To mitigate the risk of a compromised machine being used as a springboard for launching attacks on other University machines it was decided to put a firewall into place to block connections and traffic *outbound* from the server. The specified traffic is allowed through this firewall in either direction.

2.3 Database

MySQL was chosen for the database, largely because it was already required for another part of the Combechem project.

3. Quantum Mechanics Jobs

Users of e-Malaria construct their molecules in a 2d graphical editor. Unfortunately, the editor has no means for constraining the bond lengths, element types or bond-orders of the molecules entered, or deriving three dimensional structures from it. Molecules created thus need considerable checking and modification before they can be used for docking purposes. This preparatory phase, though in reality consisting of many separate stages, is referred to as the 'QM optimiser' throughout this paper.

The QM optimiser is implemented in a client-server fashion; one or more instances of the client script running on a remote Linux machine poll the webserver machine with requests for jobs. All network traffic is performed using the HTTP protocol as implemented in the 'curl' application. If no job is received from the server, the client will wait for a set period before polling again. If the server does not respond to the request, the client will retry a set number of times before pausing in 'standby' mode for an hour. The client code has been written in such a way that multiple servers may be polled by the same script; this is to facilitate reusability, it being a straightforward task to add other servers to the system and so providing a valuable service to other Combechem projects. Any errors which occur during the optimisation process are

communicated back to the server through the same type of web request transfer as a normal result. This is illustrated in Figure 4.

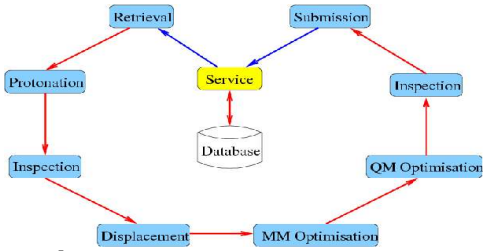

Fig 4. The QM optimisation process

On completion of a successful download, the first processing step is protonation; an open source program called 'OpenBabel' [OB2005] is used to do this. After the molecule has been protonated, we would ideally like to move straight on to the QM calculation to determine the accurate geometry. Unfortunately we cannot do this because the QM software packages available to us cannot optimise a structure from the simple topology data created by our editor application. To fix this problem, a program was written to do a rough geometric optimisation using a simple molecular mechanics force field; a reasonable molecular structure is needed for QM optimisation. This program is called 'molopt'. Molopt is a C program written for the e-Malaria project to solve the optimisation problem described above. It also performs a variety of error checks and calculations on the initial molecule configuration—verifying the chemical 'sanity' of the user's molecule.

Before the optimisation is performed, the coordinates of the molecule are displaced by a small random distance to remove degeneracy problems for molecules whose atoms are all at the origin (null coordinates). This allows the optimiser to generate structures for molecules for which the user only has topology information, such as an INCHI code. Although not needed for the purposes of e-Malaria, this may make the code useful for other projects within the Combechem programme.

A more accurate geometry opimisation using a QM package is then performed. During the early development of the software, the program Gaussian 98 was used, for which the university has a site licence. This licence explicitly forbids providing any sort of service that makes use of their software, though, and so another program had to be found before rollout. After much searching and trial and error, a version of MOPAC was obtained which compiles correctly under Linux. Several other MOPAC versions had been tried, but these met problems with modern FORTRAN compilers. Early versions of MOPAC were in the public domain, but the software was eventually acquired by Fujitsu corporation. Although they still own the legal rights to use it, their development work on it has ceased and apparently no-one in the company is currently responsible for, or contactable about, its licensing. In the light of this we were obliged to obtain an updated, but working copy of the last public domain version of the software. Our modular approach to design means that if we were to create further systems based on e-Malaria for internal use, we would be able to substitute Gaussian seamlessly where licensing permits.

After MOPAC has finished, the topology of the structure it has produced is compared against that of the original structure, to check for the molecule breaking apart.

4. Docking Jobs

Docking calculations have been implemented using the GOLD software package generously provided for the project by the Cambridge Crystallographic Data center [CCDC2005]. In order to obtain a license for this software, we needed to keep it secure from theft. To do this we used the United Devices (UD) platform.

4.1 United Devices

UD [UnitedDevices2005] produce software for managing large scale distributed computation systems. The company has its origins in the pharmaceutical software industry, and their platform is particularly well suited for drug docking calculations.

Their platform follows a client server model. Although it is possible to build a server system comprised of multiple machines, communicating with each other through web services, this was deemed unnecessary for our relatively small scale operation. Instead, the UD software, along with the IBM DB2 database and Apache server that come with it, were installed on the dual Xeon server, 'Purple'. The system is fairly resource hungry, and little else could profitably be run on Purple.

The client part of UD can be obtained for both Linux and Windows systems. Earlier in the project we experimented with the Linux client, but eventually dropped it in favour of a purely Windows based system. A standard Windows installer executable with the IP address of the server built into it was

produced; after installation this client repeatedly polls the server requesting jobs. All transactions are done using standard HTTPS (web services). The first time a docking job is sent to the client, a copy of the GOLD executable tailored for execution in the UD environment is sent along with it. Subsequent jobs use the cached copy. To maximise the security of the system, the executables we are using are cryptographically signed by the UD system to verify their authenticity.

4.2 Polling Scripts

A pair of scripts were written to obtain jobs from Green and submit them to the UD system running on Purple. The first script is quite similar to the optimiser client in its functioning (and was indeed based on it); its purpose is to poll Green for jobs and submit them to the UD system using the tool which comes with the UD/GOLD package. The second script periodically polls the UD system for the results of any jobs that have been done. When results are ready, they are extracted using the UD/GOLD submission/retrieval tool and examined.

UD allows for jobs to have names, and the name given to each job reflects the ID of the molecule in Green's database; this is how the two docking scripts keep track of the jobs. When a job is retrieved, it is extracted into an appropriately named directory and the retrieval script checks for any errors that have occurred. If the error is chemical in nature, the error code sent back to the user will reflect this, asking them to check for any unusual or nonstandard functional groups; otherwise, if the problem encountered was a systems error, the error message returned to the user suggests that the system may be under heavy load (a likely cause of such errors) and suggests that they resubmit the job at another time.

If the GOLD job has run without problem, the file is returned to the server.

5. Lessons Learned

There now follows a discussion of some of the lessons we have learned in creating the e-Malaria system.

5.1 Shift to Client Side Processing

When first creating the site it was decided, without much critical thought, to construct the pages using very similar techniques to most other web sites. The page headers and footers containing information about the user would be constructed on the fly by a CGI script, while the sidebar and document content would be added by the server itself, as directed by Server Side Include (SSI) directives issued by the CGI. Complex content requiring information from the database would be constructed by CGI modules included from the master script.

Following this approach the navigation bar content for each page of the site, as seen by each user type, was originally generated as a large set of partial HTML files to be included by the server. A script was written to build these files automatically and place them into a directory structure calculable by the CGI. It was found after a while that this constructor script required considerable maintenance and was prone to error. To fix this, a client side constructor script was put into place, containing all the data on user permissions and page titles to construct a navigation sidebar at page view time. This had several interesting features:

1. The script did not change between pages, and so could be cached on the remote browser, reducing load on the server.
2. It provided a single point of maintenance and reduced the complexity of the server scripts.
3. It created no security disadvantage: even though the script could be downloaded and the names of pages to which the user does not have access rights to determined, a constructed request for such pages would still be turned away by the server through the cookie based authentication / authorisation system.

These advantages were noted. When it came time to create the account administration system, it was decided that its user interface would be implemented in Javascript on the client. A request for an administration page from the server returns a web page containing a series of script include statements, a Javascript data structure containing a verbatim copy of the relevant information from the database, and finally a call to a function in one of the included scripts to render the page. No HTML beyond the standard header and footer bars is created by the server.

The user interface to the administration tools is a complex application with buttons for creating, modifying and deleting accounts, checking username and password fields, displaying the changes to be made in an intuitive fashion, and relaying the necessary instructions to the server once the user actions the changes. Multiple changes can be made to tables of users or groups in the

system; the totality of these changes can be seen by the user before they action them in a single transaction.

It would have been a long-winded programming task to construct such a system were server contact to be required after every user action, and the maintenance of state at each stage would make it very complicated. Also, the need for server contact would make the system much slower, and unwieldy to work with from the user's point of view.

Detailed testing and evaluation of the site was done some time after the account administration was completed. Many recommendations were made from this, with much of the criticism being directed at the 'molecule table' page. This latter was, at the time, a table containing text, graphics, hyperlinks and tooltips created entirely on the server. The improvements requested included amongst other things the following:

- Help texts, beyond the tooltips, should be available on the same page, removing the need for students to flip between pages. Ideally these should be context sensitive.
- Help texts should be illustrated where necessary.
- It should be possible to control how many elements are displayed in the table to make the page quicker to navigate around.
- The tooltips did not work consistently across all target browsers.
- The tooltip display within the browser was outside of the designer's control, and the tiny fonts used by default would cause problems for users with visual problems. New tooltips would have to be implemented independently of the browser's built-in mechanism.
- Confirmation should be sought before deleting molecule entries.
- On copying a molecule, a new name should be requested for the copy.
- The page should automatically update itself every few seconds when a job was running, but not otherwise to limit load on the server.
- A workaround was required for the poor handling of PNG images in one commonly used web browser.
- A workaround was required for the poor handling of CSS in the same browser.

It thus became clear that a major overhaul of the molecule table rendering was required, and that the new engine would need to be much more complicated than what was then in place. The success we had had building the complex administration tools with client-side scripting led us to attempt the molecule table rebuild using the same methods.

The script had to build the interactive molecule table screen and implement context-sensitive help, CSS-defined tooltips, popup alerts and queries, automatic intelligent updating and workarounds for browser foibles. From the outset this was a much more ambitious project than the administration screens had been.

In the event, with the experience gained from the administration scripts, the implementation went quickly and smoothly, apart from the fixing of the mentioned browser problems which took considerable time. The results exceeded the expectations of the designers.

Looking at this in the round, then, a clear lesson can be drawn:

Complex web applications are easier to design, maintain, test, secure and deploy if they are constructed from small secure server side scripts which simply pull the relevant, authorised data out of a database and ship it to the client, along with a Javascript program which takes care of drawing the page and handling the user interface entirely in the browser.

5.2 File Formats and Dispatch: One Point of Contact

Over the course of development, e-Malaria has had many different formats for storing and transmitting molecule information. As the system developed at the start, file conversions and storage into/retrieval from the database were performed in many different places, and it started to become difficult to keep track of. In particular, it started to become difficult to track down errors caused by information loss, as illustrated in Figure 5.

Eventually the decision was taken to move most of these disparate conversions and database interactions to a single master service. This would provide one point of contact for information retrieval and submission, and would handle all necessary file conversions itself. All interaction with this program would be through standard HTTP methods (GET/POST requests). By doing it this way, we reduced the complexity of the system and broadened its scope; the

potential for extending and reusing this element of the code is fairly high. Some changes would make it better still, though.

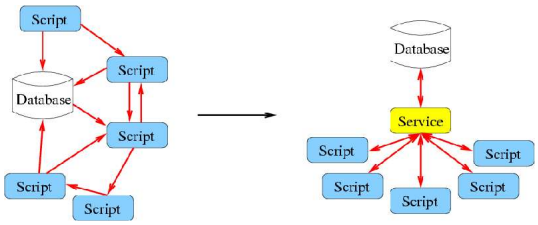

Fig 5. The system has changed from having many scripts (blue squares) interacting with the database (white cylinder) to having them speak to a master service (yellow square) which speaks to it alone.

In particular, the internal file format used was developed on the fly, and took the form of a simple space-separated column file containing all the information we needed to store without any self-describing properties. While developing the molecule viewing application it became clear that CML was going to be necessary to communicate partial charges to the Jmol applet. Our initial expectations were that the use of this XML-based file format would take a very large amount of complex programming work. As it turned out, however, the conversion from the internal molecule format to CML could be accomplished using just four (admittedly complex) Perl regular expressions, operating on the file as a whole. This was far less painful than had been expected. Although reading and interpreting a completely general CML format file would be an involved job, reading and writing files which used just a narrowly defined subset of the format (used consistently within the scope of an application suite) would require very little effort. While writing less than general code for CML would lose some compatibility, most of the benefits of the XML/CML approach would be retained, such as self-description and readability by truly standards-compliant applications, and also extensibility and future-proofing.

If the project were started again, just such a narrowly defined subset of XML/CML would be used internally for data storage. The space efficiency of the format can be raised to quite a high value using appropriate types of tags, and if space were still a problem, compression is a simple and effective solution.

5.3 Cascading Stylesheets, Javascript and the DOM

Cascading style sheets (CSS) were used to obtain a high degree of presentational consistency across the site, and also to gain the ability to apply multiple styles to the site if necessary – for the purposes of making the site usable by students with dyslexia or other disabilities. Ideally, the flexibility of this feature of modern web browsers should also enable the designer to produce significant amounts of interactivity with a very low code content, i.e. very little presentation-related Javascript. Some Javascript was always going to be necessary to create the site and its interactive pages if server load was to be minimised, but at the start of the project, as described above, it was assumed that there would be relatively little of it. Where Javascript was necessary, it was to be written in a consistently standards compliant fashion, such that it would work on any standards compliant browser.

This strategy proved to be somewhat idealistic. While CSS works extremely well in most browsers and can greatly simplify the creation of complex web-based user interfaces such as the ones in this project, there is a major problem with using it in real-world systems: the most common web browser on Windows-based PC machines has an inconsistent implementation of the standard, and workarounds for its foibles were required in several places. Unfortunately we have no choice but to support this system and make allowances for it, and this somewhat limited what it was worthwhile attempting in terms of web design.

To work around some of these limitations, a greater use of Javascript was required than originally planned. Although this is an excellent language, use of some of the more advanced features of the browser in a standards-compliant fashion (through the Document Object Model) proved too complicated to be worth the effort.

5.4 Dyslexic Friendly Design

One of the original project aims was to create a website that would be usable by children with dyslexia. We consulted with the special needs teaching unit at the University, who provided us with a long list of design guidelines for the site. The graphical design of the site was implemented using CSS in a flexible manner. The original plan was to create *two* stylesheets for the site, one for users with dyslexia and one for those without. This was working under the tacit

assumption that the dyslexic friendly styling would, while usable for those with this condition, be visually unattractive.

As it turned out that the dyslexic design, although unusual, is both usable by all users and visually appealing. We decided to make it the default style for all users.

In general, the design considerations we were given could very easily be applied to all website designs. The results would improve presentation and structure for all users, regardless of whether or not they have dyslexia. This really comes down to making full use of the new design possibilities electronic hypertext media has to offer.

6. Conclusion

e-Malaria has established that it is possible for a small team of designers, programmers and teaching specialists to construct a complicated learning environment in a relatively short time frame. Modern, scriptable, plugin enabled web browsers, and the standards which unite them (for the most part) make available a rich environment for composing an interactive multimedia user interface. Furthermore, the tools they make available to content designers allow for the tailoring of a site for users who have special needs with relatively little effort.

From the outset it was clear that the e-Malaria system was going to be a very, very complicated undertaking. This complexity derives from the requirements of uniting software from many disparate sources, working on many different computers. Furthermore, some of the software used was or is still in a pre-release development state. In this paper we have described some of our methods for dealing with this complexity using modern web-based design methods.

Sitting behind the web-based user interface, the tools made available by webservers have extended their usefulness far beyond their original application sphere. To paraphrase Larry Wall, designer of the 'Perl' language, communications media differ not so much in what they make possible, but in what they make easy. In this project we have found that web based communications methods remove most of the back-breaking 'detail work' of intermachine communication, such as connection initiation / tracking, multithreading, security, etc. leaving the programmer to get straight into the problem they are actually trying to solve. By consistent use of web interfaces, reusability is also facilitated.

As the designers of this project, we feel that this sort of combination of advanced browser technology and advanced server technology is what the early visionaries and pioneers of the World Wide Web had in mind. Though today there is far less attention paid to developments in this sphere by the conventional media, and the wild-eyed evangelism of the late twentieth century has evaporated, development work has continued at breakneck pace. Now, finally, all this hard work in designing standards for uniting computers and people throught complex multimedia applications is starting to pay off.

Finally, it must be mentioned that this project would not have been possible without the availability of good quality, free and open source software. We would like to acknowledge Professor Tim Clark at the University of Erlangen for his assistance with MOPAC, CCDC for kindly allowing us to use their GOLD molecular docking software, and to all of the developers who have worked on the free and open source software we have made use of in this project.

References

[Southampton2005] http://emalaria.soton.ac.uk

[Essex2005] http://www.soton.ac.uk/~chemphys/jessex

[Tai2004] Kaihsu Tai, *et al.* (2004) BioSimGrid: towards a worldwide repository for biomolecular simulations. *Org. Biomol. Chemistry* 2:3219-3221, http://www.biosimgrid.org

[ACD2005] http://www.acdlabs.com

[Sourceforge2005] http://jmol.sourceforge.net

[Southampton2004] http://www.soton.ac.uk/Press/PressReleases/Name,4929,en.php

http://www.sciencelearningcentres.org.uk

[UnitedDevices2005] http://www.ud.com

[CCDC2005] http://www.ccdc.cam.ac.uk

[OB2005] http://openbabel.sourceforge.net