A Proteomics Laboratory System and Digital Mentor

Anthony Maher[1], Michael Lake[1], Tim Langtry[1] and Cameron Hill[2]

[1] Computational Research Support Unit

[2] Proteomics Technology Centre of Expertise

University of Technology, Sydney

Abstract

The typical small biological laboratory is composed of electronic scientific equipment and wet laboratory equipment where staff record results into paper workbooks. There is often a lack of expert staff to guide new users, and hard won experience is locked away with individual staff and their workbooks. The Proteomics Laboratory System and Digital Mentor aims to: facilitate the capture, analysis and sharing of results; facilitate the creation and documentation of standardised work flows and experimental procedures; and mentor inexperienced users in best practice when engaged in particular procedures in the laboratory. In this paper we discuss the design of the system and report on the initial production release. This is based on a relational database and web-server framework. A novel aspect of the system is the exploitation of custom extensions to a wiki-style interface that add functionality for non-expert end-users.

Keywords: LIMS, Laboratory Information Management Systems [0]

Responsible Author: Anthony Maher, Computational Research Support Unit,

University of Technology, Sydney, PO Box 123, Broadway NSW 2007, Australia.

Introduction

The Proteomics Technology Centre of Expertise (PTCE) at the University of Technology (UTS) is a small group of researchers and technical staff which offers services and training in proteomics discovery technologies to Australian and international researchers from academia and industry. The group has particular expertise in experimental design, custom method development, sample preparation, complex mixture fractionation and protein separations. However, in addition to servicing established researchers, the group is responsible for meeting the equipment and training needs of a relatively transient and inexperienced student population. As a result, there are a number of knowledge and information management issues to be addressed in the laboratory's operations.

- There are insufficient specialised bioinfomatics resources to provide on-call assistance in guiding research work, staff with the knowledge required for a particular experimental protocol are not always available.
- Electronic laboratory equipment is only partially networked.
- There is a continuing stream of new untrained users.
- Users need to move around the laboratory when performing experiments – they require a ubiquitous assistant.
- Working notes and experimental results that have traditionally been stored in users' paper workbooks are not easily accessible to other researchers.

These issues arise in many traditional laboratory environments, and have given rise to the development of a range of LIMS and LIS (for example [1], [2]). Indeed there are now a number of commercial products offering aspects of the required functionality in different application areas [3]. Nevertheless, there remains a need for the development of customised systems to address the specific type of requirements, be these in the context of of a particular reporting regimes [3], or, as in this case, of a higher education environment that combines both research and training activities.

In particular, the development of a custom laboratory management system for PTCE was influenced the cost and functionality of commercial systems, and the rapidly evolving nature of the application field (proteomics). In addition, laboratory staff requested the development of a digital mentoring system that would extend the functionality of a conventional laboratory information system. Three major areas of functionality were to be addressed:

- support for the design of new experiments and associated procedures, and their inclusion in an electronic knowledge base;
- capture, storage and retrieval of experimental data; and
- support for management of the knowledge base.

A hard-coded demonstration system was developed during the second half of 2006 as a proof of concept and an initial production version, developed in early 2007, has been released to users. The production version is subject to ongoing development. Experience gained to date has informed development of the production system in a number of ways.

Objectives of the Proteomics Laboratory Information and Digital Mentoring System

Desired attributes of the initial system were to include the following.

- A domain specific (in this case proteomics) knowledge database containing local and/or distilled knowledge combined with links to external references.
- A web-based interface that will work efficiently via wireless PDAs in the laboratory, allowing users to access the knowledge database and/or enter experimental results directly in an electronic form. The web interface must be able to be rendered on portable devices such as PDAs, so that data can be entered in the laboratory at the time the experiment is conducted.
- A work flow system (or experiment planner) which provides a step by step guide to

performing an experiment, allowing new users to follow standard procedures with minimal guidance from laboratory staff.

- A centralised repository of all experimental data and information. This repository should be readily accessible and make it easy to share data and information between users, while providing adequate security. It should facilitate, where necessary, additional post-processing of experimental data.
- Networking of equipment, or other methods of allowing data to be easily stored in the central repository.
- The ability to upload digital images. Often in the laboratory users will take a picture or movie to explain or demonstrate what is happening at a particular point in an experiment. This facility was identified by proteomics users as being extremely valuable, particularly in reference to the mentoring aspect of the system.
- The ability for end-users to customize and add to the system by defining new procedures and new experiments.

Development and Design Aspects

Detailed design and development of this system is being undertaken by the Computational Research Support Unit (CRSU) at UTS. This is a very small unit with a reasonably broad range of responsibilities, so the system development environment must be simple and highly productive. In addition to meeting the immediate needs of the researcher in the laboratory, the system is intended to be sufficiently flexible that it can take advantage of current developments in grid technologies. Desired attributes of the development environment therefore include:

- a simple but powerful development framework;
- a relational database for all data storage (no data should be stored in the file system);

- use of grid technologies for post-processing of data, including data transfer, authentication and authorisation; and
- automated capture of meta-data (time of data entry, by whom, recording of changes to data, history of data from input/capture through computational processing to final presentation).

Based on these criteria we chose the Python language using the Django web framework with a SQLite database back-end for development.

Python was chosen as the programming language as we have existing expertise in this language, it provides ease of programming, clarity of code and in-built documentation functionality. It has been used in a variety of project setting, including laboratory automation systems [4]. Additionally it provides bindings for the Globus toolkit and has a number of development frame works to choose from.

The python-based web framework being used is Django [5]. This provides an excellent HTML templating system, a relational database back-end with database abstraction through an object relational mapper, and in-built administrative tools that can be extended. The Django framework also provides a ready-made user and groups system for controlling access.

As the Django framework mandates the use of a relational database back-end, it avoids the need to develop a custom ad-hoc storage and query system required by a filesystem based approach. The relational database system can also be accessed from outside of the Django framework, making the integration of additional facilities simpler. While Django allows for a number of different database back-end, SQLite [6] was chosen based on its simplicity and how it mitigates the problem of providing developers with an up-to-date private copy of the current database (it can be simply copied from production). This can easily be changed later to MySQL or PostgreSQL (via a simple configuration change) if required for heavy duty production use.

The web front-end relies heavily on the use of graphical representations of work flows. The images

used to render these are produced by using graphviz software [7] on lists of nodes representing procedures that are stored in the database.

The System – Current Implementation

An initial implementation of the experiment planner has been completed. Development of the knowledge base, however, is still in progress. When completed, it will consist of experiment protocols, electronic copies of literature, and links to related information on other websites. This section will not be modified by users, but will be updated by an administrator to ensure the information remains current. The information will be accessed in the same way that stored data is accessed, namely via experiment "trees" as described in the following sections. Each experiment tree that is created and saved will include links at the nodes to the specific knowledge base information relating to the corresponding step in the experiment.

The base system currently consists of two parts:

- the relational database which contains all the experimental data and procedures; and
- the web-server framework which provides access to the database and the work flows.

Users can access or upload data via the web server and all data resides in the relational database.

The system is currently running under the Apache web-server using mod_python. Since it is a web-based system, the authorisation and authentication can be done using Shibboleth. In the initial stages a local LDAP server is being used.

The Django web framework (Figure 1) consists of an object relational mapper (ORM) which maps tables in the database to Python objects, a request handler that passes incoming requests for web pages to the appropriate code to handle that request in the business logic of the application, and an HTML templating system. The templating system neatly separates the application's look-and-feel from the Python code which makes the code more maintainable and

makes it easier to modify the appearance without touching the main code.

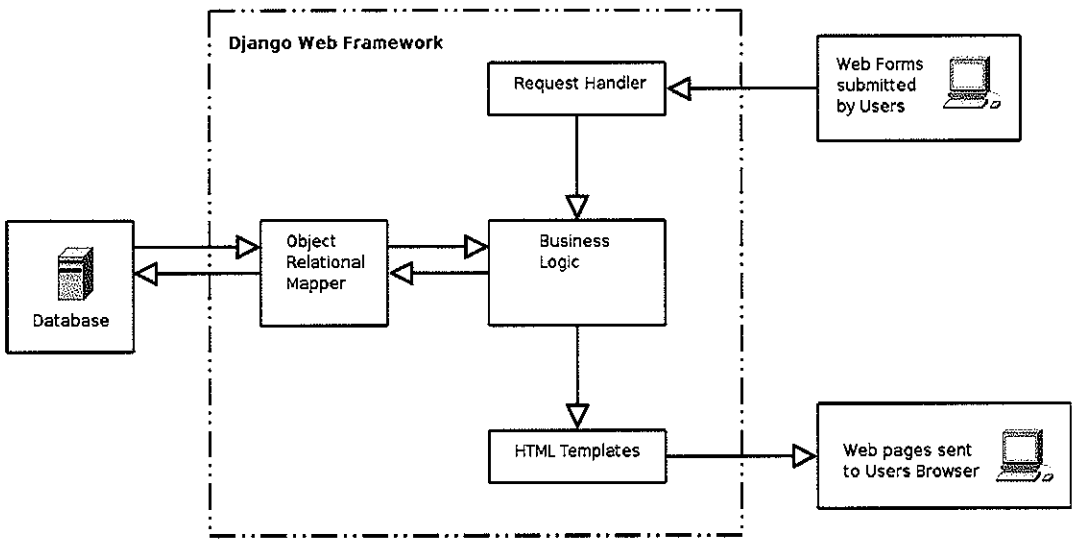


Figure 1: Simplified diagram of the system architecture.

The object relational mapper maps tables in the database to Python objects. This provides the Python programmer with a higher level view of the database and a simpler API to work with. It also handles any database specific SQL syntax so one can easily replace the database back-end.

As an example the experiments table in the database consists of several columns such as experiment name, description, experiment structure, etc. Normally one would query the database with a SQL query such as:

```
SELECT name, description, structure FROM experiments \
    WHERE experiment_id ='3'
```

However when the experiment table is mapped to an experiment object in Python one can then create an experiment object (e) like this: e = Experiment.objects.get(experiment_id='3') and then access the data as e.name, e.description, e.structure etc. As the queries become more complex the object-oriented approach is greatly appreciated.

The Django web framework also has an inbuilt web-based administration interface, an authorisation scheme with users and groups and handles smart caching of requests.

The Users' Perspective

The system is designed to be entered from the PTCE website, giving access to the experiment planner function, the knowledge base, and stored data.

The access system is flexible, allowing users access only to their own experiments and data, and those experiments and data that other users have made public. All the data stored by a specific user is stored in the form of saved experiment trees. When a user logs into the system they have access to a list of all the experiments that user has created, and hence to all the data they have generated to that point.

The user sees a web-based interface in a style with which they are familiar. As the system is designed to be usable on the relatively small screen of a PDA, the web pages are designed to be as simple as possible with the minimum amount of information. Use is made of techniques such as having links to hide/display information depending on what the user wishes to see.

Three main functionalities are currently provided.

- Managing experiments – including creation of new experiments, cloning of existing experiments (without the data), deletion of experiments (but only if there is no experimental data).
- Running experiments – a view of the entire experiment tree is shown; the user can select an individual step or procedure and get an input form for data along with relevant background information.
- Viewing data from previous experiments – similar to running experiments but without the ability to change data.

An experiment is composed of three parts:

1. the name of the experimental run;
2. a tree of procedures; and

3. data associated with each procedure (step) in the experiment.

Each experiment plan is a decision tree, or directed acyclic graph, where at each step the user may follow a different path (or multiple paths) depending on their current results (see Figure 2). The user-selectable node corresponding to a given procedure contains the relevant instructions and data input forms (for example procedure see Figure 9). The following figure shows a typical experiment tree of procedures.

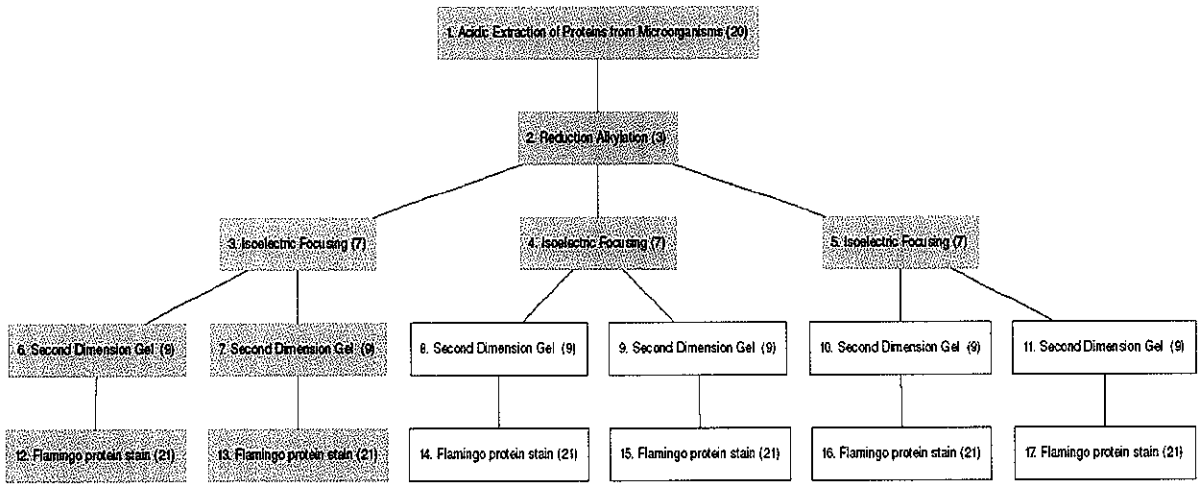


Figure 2: A typical experiment tree; in this case the experiment "Acidic extraction of encapsulated fungus". Procedures for which data has been collected are shown coloured - non-coloured nodes are procedures that have not yet been "run".

Users can develop their own experiment plans and share them (and optionally their data) with other researchers. Each time an experiment is created, a new and unique experiment tree is created. The primary interface for both input and retrieval of data is through these experiment trees. A given tree then serves as a link to all the data generated in that particular experiment, via links for each step of the experiment that access the specific data generated in that particular step.

Building experiments – the basics

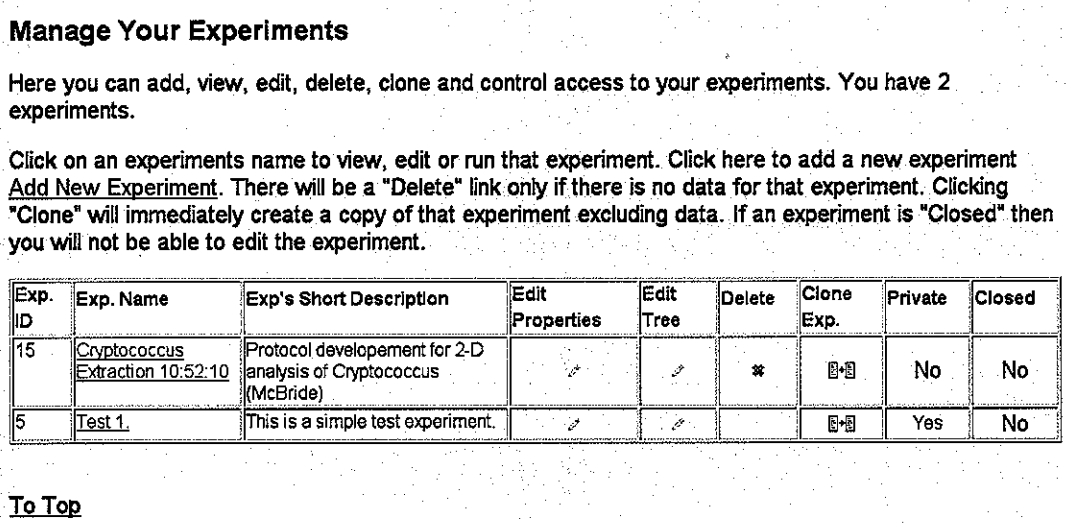
The user has a number of options when building an experiment. The system provides a 'null' experiment template from which the user can construct an experiment from scratch. It also provides a number of standardised procedures which have been written by the proteomics system administrators

who have the domain expert knowledge. Each procedure forms a single step in an experiment. A user can clone any of their own or any of the publicly available experiments. Complex experiments can be designed or modified by clicking on links to add or remove nodes in the experiment tree.

Basic data curation can be performed at data entry by supplying expected ranges for variables and imposing range checking. Once data is entered it can only be removed by an administrator. Any changes are recorded together with when and by whom the change was made plus a non-null comment explaining the reason for the change. This is to allow detection of fraud.

Creating and editing experiment trees

Experiments can be easily created and edited by users. User can create experiments by clicking on the "Add New Experiment" link or cloning an existing experiment link as shown in Figure 3.



Manage Your Experiments

Here you can add, view, edit, delete, clone and control access to your experiments. You have 2 experiments.

Click on an experiments name to view, edit or run that experiment. Click here to add a new experiment Add New Experiment. There will be a "Delete" link only if there is no data for that experiment. Clicking "Clone" will immediately create a copy of that experiment excluding data. If an experiment is "Closed" then you will not be able to edit the experiment.

Exp. ID	Exp. Name	Exp's Short Description	Edit Properties	Edit Tree	Delete	Clone Exp.	Private	Closed
15	Cryptococcus Extraction 10:52:10	Protocol developement for 2-D analysis of Cryptococcus (McBride)					No	No
5	Test 1.	This is a simple test experiment.					Yes	No

To Top

Figure 3: Manage Your Experiments page allows for creation and cloning of experiments as well as adjusting or displaying some meta-properties of the experiments.

When creating an experiment from scratch the system just requires a name for the new experiment. Once this is entered, the system creates an experiment tree of just two 'Unknown' nodes. The user then edits each of the two nodes to assign a procedure to each node. Figure 4 shows the initial experiment

that has had the procedures assigned to each node. The mechanism for assigning procedures and adding new nodes is illustrated in Figures 5 to 7. Nodes can also be deleted if they are leaf nodes by simply clicking on the '[X]' link in the node.

The tree view consists of a PNG image and an imagemap to allow actions to be associated with parts of the image. The image and image map are generated using the graphviz software. The same basic tree is shown for all three access modes; View, Edit and Run. Just the links inserted into the imagemap differ depending on the mode.

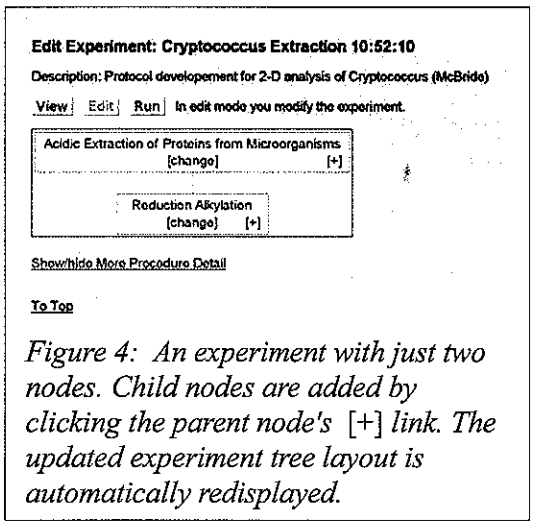


Figure 4: An experiment with just two nodes. Child nodes are added by clicking the parent node's [+] link. The updated experiment tree layout is automatically redisplayed.

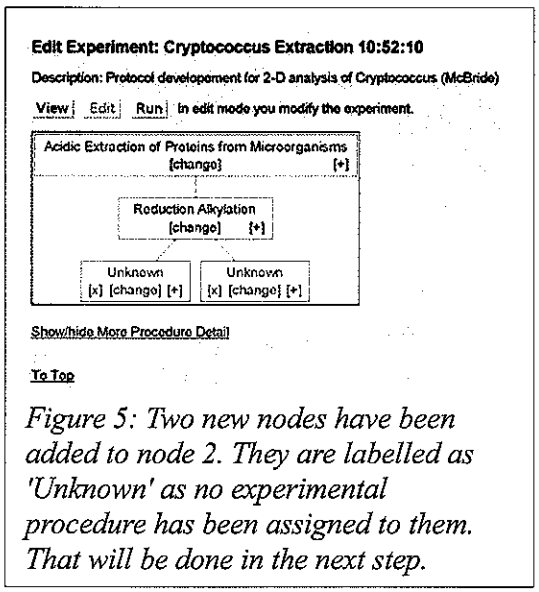


Figure 5: Two new nodes have been added to node 2. They are labelled as 'Unknown' as no experimental procedure has been assigned to them. That will be done in the next step.

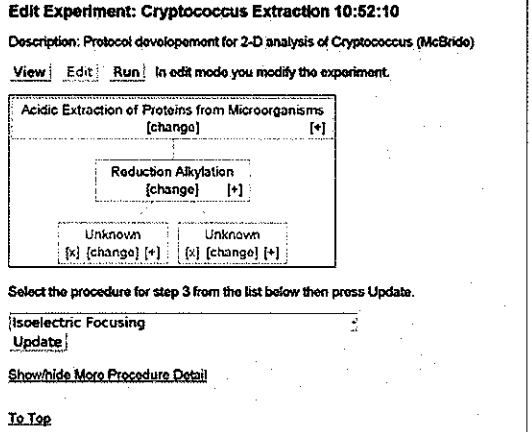


Figure 6: For each unknown node the user clicks the [change] *link for that node and a drop-down list box will appear that lists the available procedures. The user selects a procedure and presses the 'Update' button.*

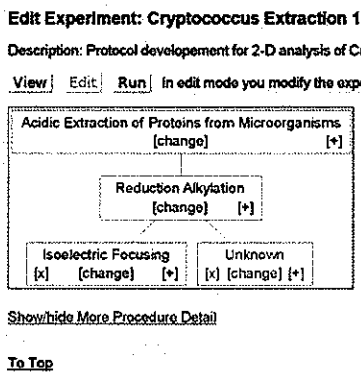


Figure 7: Node 3 (which was previously an 'Unknown' procedure) is now 'Isoelectric Focussing'. The same process would be followed for node 4 and any additional nodes.

Running an experiment

Referring back to Figure 2 it can be seen that some nodes are coloured. This is a partially completed experiment tree. The coloured nodes indicate that data exists for that step of the experiment. The user can click on any node, which will take them to the corresponding procedure step as shown in Figures 8 and 9. Where the protocol gives a detailed written description, it can be hidden (as shown in Figure 8) allowing for easier data input when using the PDA, or fully displayed (as shown in Figure 9) when the user needs additional assistance. With the advent of high resolution PDAs (1024x600) and their specialised screen access methods, they have been found to work well in the laboratory.

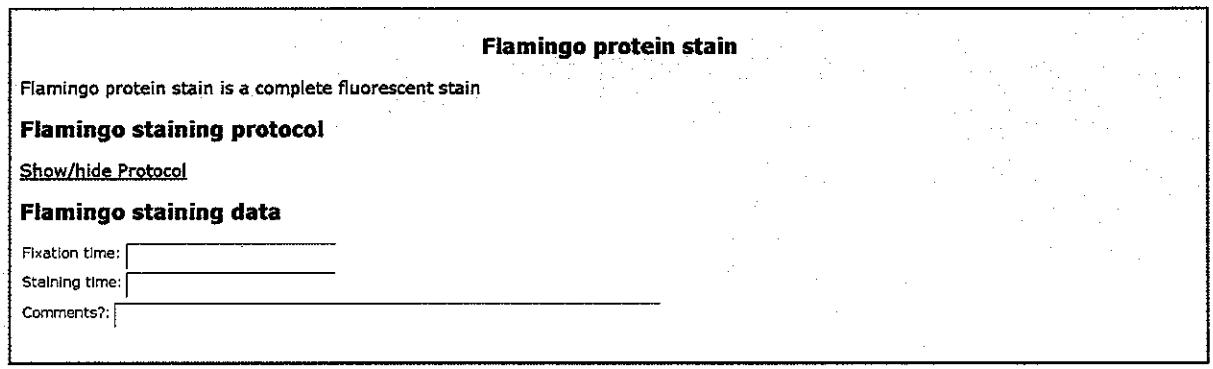


Figure 8: Displayed version of wiki protocol definition with details hidden.

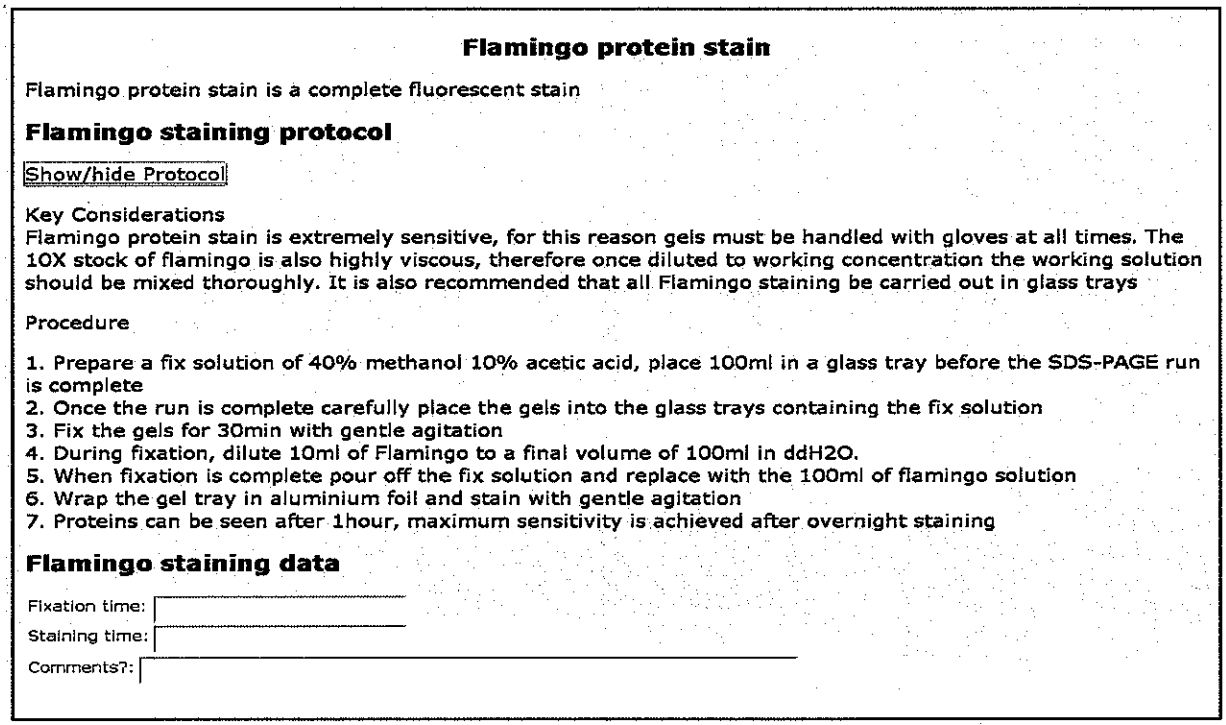


Figure 9: Wiki protocol definition with details displayed.

Defining a procedure – the wiki form extension

The proteomic laboratory users are not computer experts, and this has driven the interface design. In order to provide the flexibility required to allow users, or more likely the proteomics systems administrators (who have the domain expert knowledge), to customise the system, a framework based on a wiki style interface is used for producing input forms and for adding to the knowledge database. A novel feature of the system is that the wiki interface has been extended to allow the production of

forms for input and not just documentation. As well as giving each data entry box a name, attributes such as type, units, maximum, minimum and default values can be specified. These can then be used in rendering the HTML forms for data input.

Each procedure is entered via the wiki style interface using a wiki format called "textile". The wiki format data is stored directly in the database. The wiki mark-up language was extended to allow for the specification of form input elements. These are delimited by '[[' and ']]' and are referred to as the form element tags. This syntax was chosen so as not to conflict with other elements and also because it looks like an input element to the user editing the procedure. Within the form element tags, the user must specify, as a minimum, the name of the data field, e.g. [[name=Fixation_time]].

It is also possible to specify the standard HTML [8] input attributes:

name	name of the control to send back to server
type	type of control to create
value	initial value of the control
size	initial width of the control (in characters)
maxlength	maximum number of characters a user may enter

Additionally the following extensions are planned:

min	minumum value that can be entered
max	maximum value that can be entered
comment	comment that could form part of a mouseover tooltip
units	a list of allowed units
dbtype	the type for the database storage.

For example, the following string specifies an input form element called *Fixation_time,* with

appropriate range limits and comment:

```
[[name=Fixation_time type=text value=2 size=2 maxlength=2
dbtype=integer min=0 max=60  comment="The fixation time has maximum
value of 60s as the chemical reaction is complete after this time."
units=[s|ms] ]]
```

```
h1. Flamingo protein stain

Flamingo protein stain is a complete fluorescent stain

h2. Flamingo staining protocol

<div id="protocol">

Key Considerations
Flamingo protein stain is extremely sensitive, for this reason gels must be handled with gloves at all times. The
10X stock of flamingo is also highly viscous, therefore once diluted to working concentration the working solution
should be mixed thoroughly. It is also recommended that all Flamingo staining be carried out in glass trays

Procedure

1.      Prepare a fix solution of 40% methanol 10% acetic acid, place 100ml in a glass tray before the SDS-PAGE run
is complete
2.      Once the run is complete carefully place the gels into the glass trays containing the fix solution
3.      Fix the gels for 30min with gentle agitation
4.      During fixation, dilute 10ml of Flamingo to a final volume of 100ml in ddH2O.
5.      When fixation is complete pour off the fix solution and replace with the 100ml of flamingo solution
6.      Wrap the gel tray in aluminium foil and stain with gentle agitation
7.      Proteins can be seen after 1hour, maximum sensitivity is achieved after overnight staining


</div>

h2. Flamingo staining data

|Fixation time: | [[ name=Fixation_time ]]  |
|Staining time: | [[ name=Staining_time  ]] |

|Comments?:   |\3. [[ name=my_comments size=56 ]]                                    |
```

Figure 10: Wiki version of definition of a procedure showing form input elements.

Figure 10 provides a more complete example – the wiki form definition page corresponding to the users views as seen in Figure 8 (collapsed form) and Figure 9 (full form). The wiki platform offers the advantages of being browser-based and of requiring little additional training for users to be able to add components to the system.

When a page is rendered, the wiki format data is taken from the database and passed though a number of filters to be finally translated to XHTML. The framework allows for any number of filters to be used.

Currently two filters (*wikiform* and *textile*) are called from within an HTML template such as

```
<p>{{ p.description|wikiform|textile }}</p>
```

The two filters are:

- Textile: a wiki markup language to XHTML transformation filter (available from [9]) and
- Wikiform: a filter developed in-house that takes the entire wiki description of a form page and converts only the elements enclosed in the '[[' and ']]' delimiters into XHTML form input elements.

A simplified wikiform filter (which just handles the base W3C attributes) can be easily coded using regular expressions in Python as:

```
def wikiform(content):
    '''content is the raw wiki markup language'''
    badchars   = '"=|'
    stripchars = ' \t"' # we put our own quotes, so strip user supplied.
    inputbox_re  = re.compile(r'\[\[.*?\]\]') # non-greedy match
    attribute_re = re.compile(r'\w+\s*=\s*(?:\w+|"[^%s]+")' % badchars)
    for input_box in inputbox_re.findall(content):
        data = ''
        for attribute in attribute_re.findall(input_box):
            (key, value) = attribute.split('=')
            data += '%s="%s" ' % (key.strip(), value.strip(stripchars))
        replacement = '<input %s/>' % data
        content = inputbox_re.sub(replacement, content, 1)
    return content
```

The simplicity of the filtering approach should facilitate future extensions of the forms with more advanced functionality. As an example, the wikiform filter has been expanded so that rather than the user having to specify complex HTML to handle the hiding or displaying of protocols (as displayed in Figures 8 and 9), they can use simpler HTML tags `<div id="protocol">` and `</div>` to mark the protocol.

Current State

The base system is currently being trialled in the Proteomics Technology Centre of Expertise (PTCE). The development of the system is following a "ground up" approach. One of the most important requirements is to start the on-line collection of experimental data as soon as possible. To this end, the focus of work so far has been the user interface used in the laboratory to collect data.

Future Development

Future development will be guided by the requirements of the Proteomics Technology Centre of Expertise. Planned enhancements include:

- the history mechanism to record all changes and who made them;
- additional meta-data such as the experimental instrument, type or class of data and other parameters consistent with data handling standards;
- ability to store voice (or sound) clips by the researcher (similar to image uploading);
- the knowledge database and expert system. Initially the knowledge database will be a manually constructed set of HTML pages with dynamic links to the experiment planner. Later a deductive expert system will be investigated;
- the Django framework provides a basic search interface as part of its administration interface but will develop a end-user targeted simple but configurable search page;
- post-processing interfaces for grid-enabled computational work and sharing of data within a virtual organisation.

Since the system uses standard tools (Apache, python, Django, SQLite) that can be installed on Windows as well as Unix-like systems, it should be possible to run the system entirely self-contained on the PDAs. This standalone mode will allow for disconnected field work. This will bring in added

complications such as database synchronisation but the addition of history means that the source of data is recorded and any overwrites will be recorded. Also data duplication is unlikely since experimental steps are usually done by a single researcher.

Conclusion

The system is currently in an early stage of development. However even at this stage users are finding it a useful tool for recording laboratory results.

The important features are:

- a consistent interface from either desktop or PDA;
- ability to access data easily;
- a simple framework that allows the end user to produce customised web-form via a simple extension to a standard wiki system;
- a simple but powerful and efficient development environment.

The system has been designed in a way that recognises a number of key principles:

- all data and meta data is being recorded in a format that makes post-processing easier;
- a complete history of the data can be will be recorded, including any post-processing systems built onto the base system; and
- simplicity for the non-expert end-user.

While the current system is geared around work done in a proteomics laboratory, the ideas and components can be used to replace the workbook in many areas of scientific research.

References

[0] Laboratory Information Management Systems.

http://en.wikipedia.org/wiki/Laboratory_information_management_system

[1] Charles T. Lohrke, Herman Dolezal, Sherri L. Reynolds. Analytical laboratory: world class distinction with world-wide connection; from managing instrumentation to managing knowledge. In *Laboratory Automation and Information Management. **34*** (1) 1999. pp. 41-49.

[2] Scientific Computing LIMS Guide

http://www.scientificcomputing.com/LIMS.aspx?SECTYPE=LIMS

[3] C. Karlsson, S. Holgersson. ForumDNA, A custom designed Laboratory Information Management System. In *International Congress Series 1239.* 2003. pp. 783-786.

[4] Matthew H. Cahn, Mark F. Russo. Python and Automated Laboratory System Control. In *The Journal of the Association for Laboratory Automation.* February 2007. pp. 46-55.

[5] Django. http://www.djangoproject.com/

[6] SQLite. http://www.sqlite.org/

[7] Graphviz. http://www.graphviz.org/

[8] World Wide Web Consortium (W3C). http://www.w3c.org/

[9] Textile Markup Language. http://textism.com/tools/textile/