ARTICLE

OBC
www.rsc.org/obc

The semantic smart laboratory: a system for supporting the chemical *e*Scientist†

Gareth Hughes,*[a] Hugo Mills,[a] David De Roure,[a] Jeremy G. Frey,[b] Luc Moreau,[a] m. c. schraefel,[a] Graham Smith[a] and Ed Zaluska[a]

[a] *Electronics and Computer Science, University of Southampton, Southampton, UK SO17 1BJ. E-mail: gvh@ecs.soton.ac.uk; hrm@ecs.soton.ac.uk; dder@ecs.soton.ac.uk; mc@ecs.soton.ac.uk; l.moreau@ecs.soton.ac.uk; gms@ecs.soton.ac.uk; ejz@ecs.soton.ac.uk*

[b] *School of Chemistry, University of Southampton, Southampton, UK SO17 1BJ. E-mail: J.G.Frey@soton.ac.uk*

Received 5th July 2004, Accepted 16th September 2004
First published as an Advance Article on the web 18th October 2004

One goal of *e*Science is to enable the end-to-end publication of experiments and results. In the Comb*e*chem project we have developed an innovative human-centred system which captures the process of a chemistry experiment from plan to execution. The system comprises an electronic lab book replacement, which has been successfully trialled in a synthetic organic chemistry laboratory, and a flexible back-end storage system. Working closely with the users, we found that a light touch and a high degree of flexibility was required in the user interface. In this paper, we concentrate on the representation and storage of human-scale experiment metadata, introducing an ontology to describe the record of an experiment, and a storage system for the data from our lab book software. Just as the interfaces need to be flexible to cope with whatever a chemist wishes to record, so the back end solutions need to be similarly flexible to store any metadata that may be created. The storage system is based on Semantic Web technologies, such as RDF, and Web Services. It gives a much higher degree of flexibility to the type of metadata it can store, compared to the use of rigid relational databases.

Introduction

The UK's *e*Science programme[1] has so far concentrated on the issues of using Grid computing to support large-scale compute-intensive science. The relatively small-scale needs of everyday laboratory users, such as recording experiments, have largely been left out of the *e*Science picture. While the flexible and robust paper-based lab book has been the *de facto* experimental recording tool literally for centuries, it has significant problems in the world of digital science. When scientists wish to share data or move towards electronic publication of complete and detailed results, as in the Publish@Source vision,[2] the paper lab book becomes an obstacle: data captured in the lab book are invisible to those who cannot access it physically. Our goal, in concert with the wider *e*Science program, has been to develop a digital lab book system with three core components: (1) a lab book-like experimental capture system, (2) a middleware architecture to support storing and sharing those data, and (3) an application both to plan the experiment in accordance with safety requirements and to represent the results.

The paper lab book is flexible, portable and robust. It can be easily transported; it readily captures both writing and drawing; it is highly resistant to damage. As a legal document, it is used to situate and date intellectual property claims. As a communications tool, it is used to discuss work in progress and reflect on best practice. One of the key challenges of this project, therefore, has been to design a system that can compete with paper with the least perceived cost and the least change in practice to the scientist. As such, our goal has been to develop a useful, digital system that captures the best attributes of paper while adding the benefit of digital capture. In order to ensure that our system would meet these requirements and be useful to real chemists carrying out real chemistry, our design approach throughout has been chemist-centered, from interface to architecture.

In this paper we review our human-centered design approach to develop the lab book application. We then describe the ontology we have developed for human-scale experimental metadata. We also present the associated flexible storage technology we developed to support the lab book applications. We discuss the advantages we have found in particular in using RDF for experiment metadata storage over more traditional technologies such as relational database tables. In addition we show an experiment planning application and how we have integrated our existing digital lab book deployment with this new technology.

A review of other relevant work is included towards the end of this paper. This is to allow us to introduce the technologies and concepts we use in our work which this readership might not be familiar with, and to place the review in the context of those technologies.

Understanding the process of recording an experiment

In human computer interaction (HCI), a common approach for designing new software systems is called "user-centered design." In this approach, designers have a variety of techniques available that they can use to engage with the potential users of the system in order to understand those users' requirements for the new system. These requirements are then translated into a set of prototypes that are again tested with the user community. Once the prototype design is settled, the functional requirements for the working system can be derived. From this map, the actual system can be developed.

In the case of developing a digital experimental capture system, we quickly learned that extant HCI methods were not appropriate. Most of these methods assume that there is either common knowledge between the designer and the expert about the artefacts being used, or where there is not, that the process being systematised can be readily explained (see ref. 3 for a review of these approaches). There is also an assumption that the complete process being systematised can be observed in its entirety. These assumptions did not hold true in the lab: chemistry experiments, we learned, are highly expert, loosely

† This is one of a number of contributions on the theme of molecular informatics, published to coincide with the RSC Symposium "New Horizons in Molecular Informatics", December 7th 2004, Cambridge UK.

DOI: 10.1039/b410075a

structured, and potentially of long duration. The bottom line was that without being chemists ourselves, we could not effectively model the attributes of the task that we were trying to capture in the new system.

We needed to bridge the gap between the world of chemists and the world of computer scientists. To do so we developed a new design elicitation approach, which we called "making tea" or "design by analogy." In this case, we made tea as an analogue for a chemistry experiment. Making tea as an experiment let us focus on the process of the experiment and what gets recorded in an experiment without getting bogged down at the start with the particulars of an actual experiment. With tea, both designers and chemists could understand and use the analogy to communicate about the details of what happens in the lab. Tea could be used to describe both how chemistry experiments are like making tea, and, equally critically, where making tea is not like an actual experiment.

By making tea as an experiment we were able to investigate the chemist's recording process both physically and in the abstract. Physically, we could look at what activities take place where in the laboratory that require recording, from measuring chemicals to mixing compounds; in the abstract, we could see specifically what is recorded about an event, and how it is recorded during an experiment (drawing, notes, references). Perhaps most effectively, making tea let us interrogate why one thing was recorded rather than others. For instance, with tea we could ask, "Why record the amount of milk, but not which was added to the cup first, the milk or the tea?" Another advantage of tea is that the tea experiment could be carried out in a short period of time, letting us observe a complete experiment from start to finish. Indeed, we ran the experiment multiple times, and with a variety of chemists, moving from common kitchen utensils in one iteration, to full chemistry kit in another. Perhaps ironically, the result of these repeated tea sessions was, rather than the expected increase in functional requirements, we progressively reduced the complexity of the designs we prototyped.

The result of developing a method to help designers interrogate these highly expert, loosely structured, potentially long-duration tasks has been, so far, the development of a well-received prototype of a usable digital lab book system.[4] Based on the lessons learned from that work we have been developing the ontologies, architecture and experiment planning application required to support the chemist in the lab. In the following sections of the paper, we discuss each of these components in further detail. In order to contextualize the rationale for our ontology and architecture designs, we first review the interfaces for the lab book application: the user requirements for the human interaction have very much informed the design requirements for these other system components.

The experiment cycle and the application interface

The experiment cycle starts with some form of planning. In the UK the chemist has to produce a plan of the experiment as a list of the reagents to be used, and any associated hazards, as part of the COSHH (control of substances hazardous to health)[5] assessment. Often the experiment planned is a variation on a previous procedure with a variation in reagents. The plan will be authorized by a supervisor once the amounts of reagents have been calculated and the relevant safety information researched and noted for COSHH. At this point, work generally moves into the laboratory. As previously described[4] the laboratory is a cramped and hostile place for delicate equipment such as computers, requiring careful thought in the design of electronic lab book replacements. The experiment is carried out in the laboratory and in other locations, generally those providing specialist services such as the mass spectroscopy laboratory. The chemists will then analyse and write up their experiments back at their desks.

Our method of supporting the experimenter throughout the experiment cycle is to provide an experiment planning application for use at a desk and a Tablet PC based experiment capture suite of applications for use in the laboratory. The planning application will expand in time to form a viewer of the experiment results.

We have thus developed three primary applications: a planning tool, which is used to set up the plan and ingredients for the experiment; a weigh-station/liquid-measure application, used for recording the quantities of ingredients actually used, as an example of a measurement device; and a "bench" application, used for making notes and annotations on the plan while performing the experiment. The latter two applications we have implemented on a Tablet PC, to be carried around in the laboratory. The current prototype planner application is implemented as a set of dynamic, form-based web pages. The "smart lab" system is modular. For instance, other measurement devices, such as a digital camera for recording TLC plates, or a formatter for adding mass spectrograph recordings, can also be added to the system in the same way as the weigh-station application. We describe the planning application first, followed by the Tablet applications. These were developed first since they replace the lab book, and are the most critical components of the system.

The planner

As noted, chemists have to fill in a COSHH form for safety requirements (see Fig. 1). We have leveraged this necessity into a virtue for our planning approach: we modify the COSHH form to include extended descriptions of the steps for the experiment process listed on the form. We emulate elements of the COSHH form itself in the Web form we provide for chemists to develop their plan. This extended plan represents a change in the degree of detail chemists currently need to provide in order to seek approval for their experiement. In our interviews with chemists who tested this approach, it was clear that the perceived benefit of this extra effort was worth the perceived cost. Reasons ranged from preplanning meaning that less energy is spent during the experiment remembering what to do next, to having persistently legible results available.

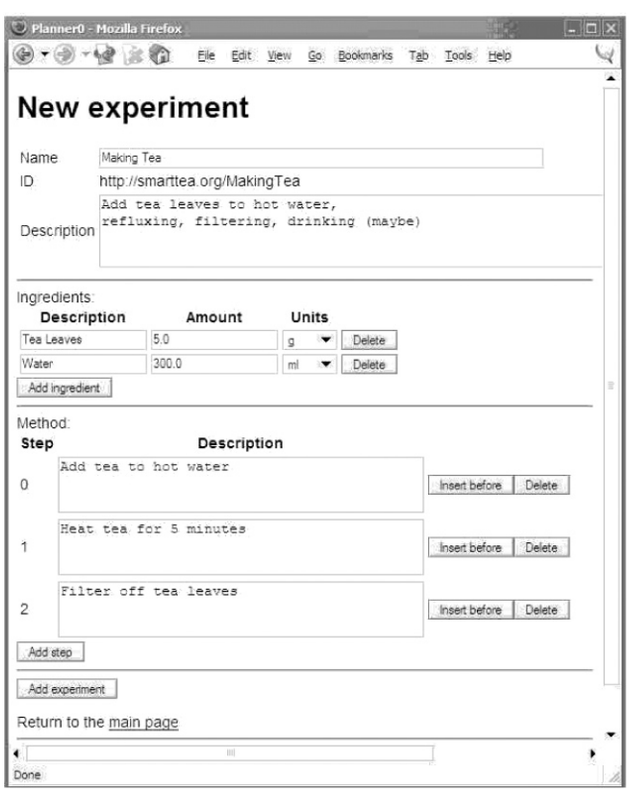

Fig. 1 The prototype experiment planner. At the top is the experiment metadata. In the middle, the list of ingredients for the experiment. At the bottom, the descriptions of the steps to be performed.

This web-based planner, therefore, allows the basic structure of an experiment to be generated from scratch or from copying an existing experiment and refining it. The main metadata for the experiment, such as the experimenter's name and a high-level description of the experiment, can be entered or updated, and more detailed data about the experiment, such as the lists of ingredients and steps can be created or modified.

The current Planner interface does not allow the creation of a non-linear experiment plan (*i.e.* where more than one step can be done in parallel), but this is not a serious problem for our test environment of a synthetic organic research laboratory, where the individual experiments rarely have parallel execution paths.

In-lab experiment capture

From planning, the chemist generally moves into the lab, with lab book, to carry out and record the experiment. The lab book takes up real space in the lab and is used to record observations about physically occurring chemical interactions. Likewise our experiment capture system, deployed on a Tablet PC, has both a similar footprint to a regular lab book, and a similar portability. This is not the only physical design that could be used. Rather than one device to be carried around the lab, a set of contextually specific devices could be deployed: a screen at the weigh scales, at the TLC plate viewer, at the fume cupboard. The Tablet prototype let us focus our evaluation on the usability of the software interaction. In future work, we will be looking further at the hardware deployment. That said, in terms of the system's support for the abstractions of recording observations, the system integrates a set of modularized services which could easily be distributed to specific locations. These services currently integrated into the single Tablet are a dry measures recorder, a wet measures recorder and what we call "the bench" for recording observations at each step of the process.

Each component of the recording system is pre-populated with data from the experiment plan. The dry and wet measure components list the names of the chemicals to be measured and their planned quantities, requiring the chemist only to enter their actual values. To facilitate rapid data entry, a numeric keypad is provided on the screen for quick tapping in of the appropriate values (Fig. 2). Likewise, the Bench component shows the list of steps produced with the planner. Each step can be annotated easily: clicking on a step opens an area for hand-written notes or sketches (Fig. 3). Again, working with chemists made it clear that hand-written input needed to be supported, hence our annotations on the plan support pen-based gestures.

All of the entries from each component are immediately written to the server so that the data are backed up on entry. This writing to the server feature was the one most commented on by chemists in our user trials: the key value added of the system for them was the sense that their data were safe: if something happened to the device in the lab, the data would still be protected.

In our initial field tests, we used a simple recording process to capture the data since our main emphasis was on testing the interaction. With the success of that evaluation, our next phase has been to develop a more robust and sophisticated architecture to store the record of the experiment, its annotations, and appropriate associated data to support review of those data. We describe these components in the following sections.

Experiment data produced by the interface applications are stored in a persistent Jena[6] RDF store using an ontology we have developed. Many of the concepts used by the system may be new to this audience so before we describe the system we have included a tutorial section.

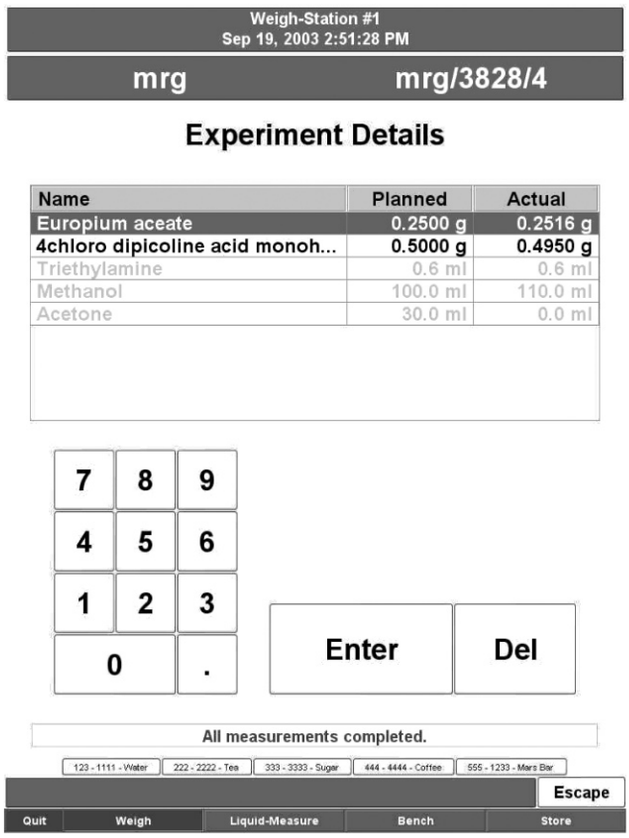

Fig. 2 The weigh-station Tablet application, showing the list of reagents, with planned and actual amounts.

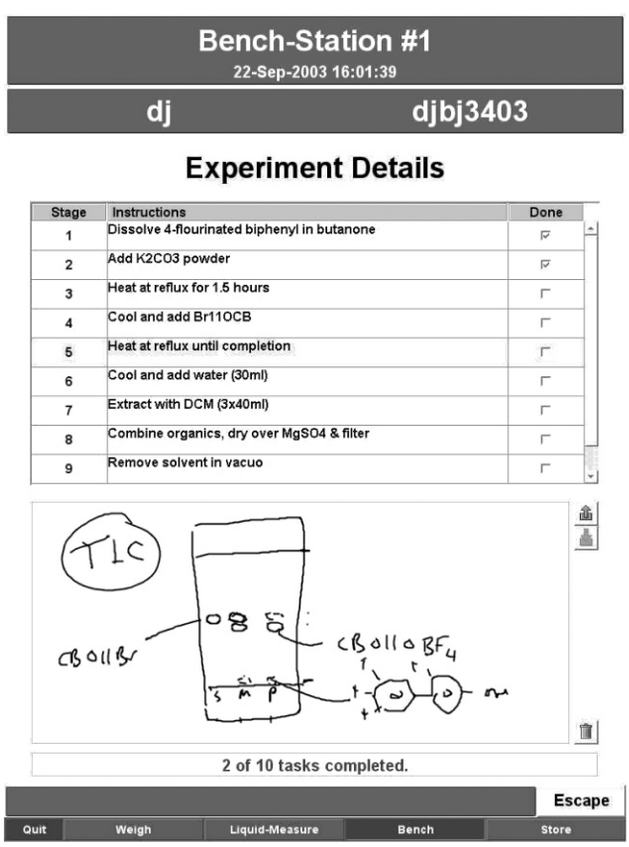

Fig. 3 The Bench Tablet application. The top half of the display shows the planned steps to be performed. The lower half shows hand-written annotations for the selected step.

A brief introduction to the world of semantics

In the record of an experiment, there are three important parts: the data gathered during the experiment, what those data mean, and *the relationships between those pieces of data.* The traditional lab book stores only the first two parts explicitly, and the third part generally only by context (*e.g.* if it is on the same page, it is part of the same experiment). When storing

the record of an experiment on a computer, it is vital to ensure that all three parts are stored, and done so explicitly, so that the maximum amount of information can be extracted from it. We use languages called RDF and RDFS to give a structure to the data which we store.

RDF (Resource Description Framework)[7] is a form of data storage which deals with the relationships between individual objects. Every piece of data in RDF is represented by a *URI* (uniform resource locator: a URL for the web is a specific type of URI), which may refer to some actual piece of data or simply refer to some non-electronic item or concept. An RDF structure consists of a set of relationships, called *triples*. Each triple consists of three URIs: the *subject* and the *object*, which refer to two entities, and the *predicate*, which is a URI with a commonly agreed-on meaning, representing the relationship between the subject and the object. So, for example, the triple:

```
[http://www.ecs.soton.ac.uk/info/person-00389/
 http://www.aktors.org/ontology/portal#has-web-address/
 http://www.ecs.soton.ac.uk/~gvh/]
```

indicates that the person represented by the "person-00389" URI has a particular web address (see Fig. 4). The URIs do not necessarily have to display anything useful in a web browser, indeed, they do not even have to resolve.

Fig. 4 An example triple.

It is possible to represent any relationship between two objects simply by adding a new triple into the RDF file or RDF storage system. It is possible to add greater structure and meaning to a relationship by using a predefined set of definitions or *ontology*. In the example above, the relationship (predicate) is

```
http://www.aktors.org/ontology/portal#has-web-address/,
```

which is part of a set of definitions published by the AKT project, and which has been given a specific meaning. Ontologies are a shared understanding of a concept which allow both humans and software to know precisely the relationships between entities and thus to be able to compare objects from disparate sources.

With RDF, basic ontologies can be written using *RDFS* (RDF Schema),[8] which is used to lay out classes of object for use in RDF. For example, in our ontology, we define

```
http://www.combechem.org/ontology/process/0.1#Reflux/
```

as a refluxing process. When we define a URI to represent such a process, we can store a relationship in our RDF store, indicating that the process in question has the type

```
http://www.combechem.org/ontology/process/0.1#Reflux/
```

(see Fig. 5 for an illustration of this).

Fig. 5 Defining an instance of a class Reflux.

RDFS can also be used to give certain types of relationships between types of entity—for example, with *subclasses*, where all things of one type are also things of another type. As an example of this, consider the process of filtering with a Buchner funnel. Every example of filtering with a Buchner funnel is also an example of filtering, and every example of filtering is also an example of a process. There are therefore relationships between "FiltrationWithBuchnerFunnel", "Filtration", and "Process" (see Fig. 6). RDFS can be used to represent such hierarchies.

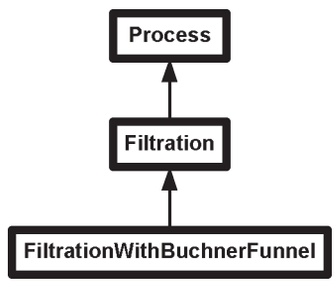

Fig. 6 Example hierarchy of process types.

An alternative language which can be used in place of RDFS is *OWL* (the Web Ontology Language),[9] which can describe the structures that RDFS describes, and also many other forms of relationship (*e.g.* that two things or concepts are the same; that two relationships are the inverse of each other; that certain classes of thing are mutually exclusive). OWL is considerably more expressive than RDFS, but is also more complex. We have started developing our ontology in RDFS, but plan to move to OWL in future work.

A body of RDF relationships (sometimes referred to as a *triple store*) does not necessarily have to have an ontology to give it the extra structure, but it is useful to have one. Conversely, if it does have an ontology, it does not necessarily have to have only one. Different relationships in the store can refer to different ontologies. This allows for ontologies to be written to deal with specific and limited areas of knowledge, and to be placed together to cover larger areas. In practise, we use the AKT ontology[10] to refer to basic information about people (identity, name, contact details, *etc.*), since the AKT project has specialised in representing such information, and use our own ontology for representing information about processes and experiments, which the AKT ontology knows nothing about.

Why should we choose to use RDF for our data storage instead of XML or a traditional relational database? The main reason is that XML and relational databases require fixed schemas in order to make sense of the data. It is, in the terminology of knowledge management, *data* rather than *information*. RDF captures information rather than data. Due to the way that ontologies work, they can be extended piecemeal and at will without breaking existing applications. In addition, the structure of RDF and suitable ontologies allows the use of software which can perform *reasoning* based on automated logical inferences over a body of information to find new connections. This is not easy to do (if it is possible at all) in a relational database structure or in XML, but the fact that RDF is based on relationships rather than fixed structures makes it much easier.

Experiment ontology

We have developed an ontology in RDFS to encompass the major phases of an experiment: planning the ingredients, planning the procedural steps, and recording the experiment. The ontology uses a high level representation of work flow: Processes have inputs and outputs. A process can be an activity performed by a person, such as refluxing, or it can be the running of a software application. In the former case the output is a substance in the reaction chamber, in the latter case it will be a file or a collection of files, and the RDF will record the URI to those output files. These can be stored in other, more appropriate, database

systems in a distributed manner. In the case of the experimental procedure the URI to the result of a process is a more abstract concept of little direct use to the chemist but is a unique reference to the substance at that particular stage of the experiment.

From observation to ontology

Information recording using the ontology is a representation of the human-scale activities of scientists performing experiments. The requirements have been gathered from our work with chemists in the laboratory. We have also incorporated requirements from the chemistry aspects of the Comb*e*chem project so that concepts such as a process can mean either an act performed by a human or the running of a piece of software. The initial design was inspired by analysing the experiments performed by chemists during the Tablet interface trials as well as some simple procedures such as the preparation of aspirin. We quickly noted that there was a spine of activity running through the record in which a process led to a product which in turn formed the input to the next process step. In the laboratory this maps to the adding of ingredients to a reaction vessel and the actions performed on it to result finally in an end product.

This process-product pairing is important, both in terms of the physical reality of an experiment, and also in software where each computational process results in output files. Recording these intermediate outputs allows us to link to each outcome from the final report and provide far greater opportunities for other scientists to reproduce experiments. Previously all that remained from an experiment would typically be the analysis of the final product. Our ontology is designed to make it easier for systems such as Ebank[11] to retrieve all of the intermediate data and results vital to reproducing a procedure.

Major concepts of the ontology

Our ontology separates out a plan from the record of an execution of that plan. There is no reason, from the point of view of the ontology, why a plan cannot have more than one execution trace, although we do not currently allow for that in our software. The separation between plan and record is an important one, since the two may not match exactly. An experimenter may perform additional steps over and above those in the plan, for example if something unexpected or unplanned happens (say, an interim result is a solid, rather than a liquid as expected, and must be dissolved to continue). The two main components which the ontology models are Materials and Processes. A Material may be either a data set, or a physical sample of a chemical (possibly some anonymous and indeterminate mixture partway through a reaction scheme). A Process may be a purely *in silico* process (a computation); a purely *in vitro* process (a chemical reaction); or a hybrid process such as the measurement of a spectrum, which takes a substance as input, and produces a piece of data.

The ontology has a hierarchy of different process types; for example, it separates *in vitro* processes into broad classes of "Separate", "Mix" and "React", with each of these being subdivided into different types: Separate has subclasses including Filter, LiquidLiquidExtract and ColumnChromatography (see Fig. 7). These may be further subdivided to give more detailed descriptions of the processes being used, as required. Extending the ontology to include additional types of process is a job for the domain experts (in this case, chemists), who are much better able to identify what classifications are useful. Extending the ontology in this way does not require a high degree of expertise in ontology creation, as it is primarily a matter of classification of processes. There are tools available which can aid the domain expert in modifying an ontology.[12]

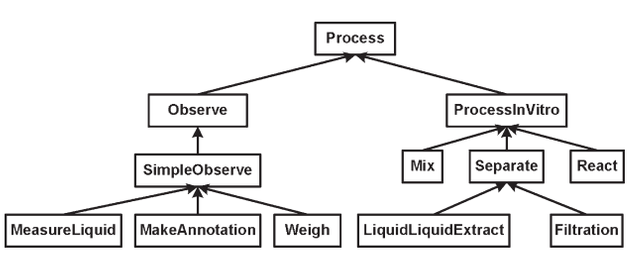

Fig. 7 The top-level elements of the process section of our ontology.

To use the ontology's Materials and Processes, observe that every Material may be the input to or the result of a Process. Some Materials will be used as input to several processes (if they are data, or if they are a large sample of substance split up into smaller samples). Some Processes may have many Materials as inputs (or as outputs—consider a filtration or fractional distillation process). Thus, the main part of the experiment consists of a network of nodes, always alternating between Materials and Processes. A measurement, which may be of some property of a substance, or of the state of a process, or even an arbitrary annotation made by the experimenter, has three parts to it: a measurement process, which is the act of making the measurement, a measurement material, which is the URI representing the result, and optionally a type/value pair, representing the data of the measurement in the case that it is a simple (one-value) observation with a unit, such as a weight or a temperature.

Ontology examples

We have developed a diagram style during this project to aid us in visualising the ontology and any example experiment models. These diagrams aid us in writing code to navigate along paths in the graphs.

The URIs in the sample experiments were created manually with descriptive names to aid debugging. In the live system the URIs are allocated by a simple Web Service to generate unique URIs. Fragments of such diagrams are included here to illustrate the experiment model structures. In the diagrams the shaded circles depict processes, hollow circles depict substances, triangles represent the making of observations and squares represent literal values. Each circle, triangle or square is a node in the RDF with its shortened URI represented in bold, black text. The class name of each node is given in italics and arrows represent relationships.

The process-product pairing

There is a spine running through an experiment in the form of a series of process-result pairs. Fig. 8 shows the nodes from one of the experiments carried out during evaluation of the Tablet PC interface software. In the first process step, butanone is added to a sample of 4-fluorinated biphenyl resulting in a mixture in the reaction vessel. Potassium carbonate is added before the vessel is heated. Fig. 8 does not show annotations and observations made by the chemist. Examples of these are described below.

Annotations

The experimenter made an annotation using the Tablet PC during the reflux, and the graph fragment is shown in Fig. 9. From the `first_step` node there is a `process_observed_by` property to a MakingAnnotation node. This represents the actual act of making the annotation and results in an observation node `add_notes` which has a literal string containing the Tablet's sketch data (see Fig. 3 for an illustration of this type of annotation being entered).

Observations

Some methods of making observations are more complex than others. The ontology accounts for this with SimpleObservation and ComplexObservation classes. Weighing solids, measuring liquids and making annotations are all subclasses of SimpleObservation. Fig. 10 shows the graph for a simple observation. Examples of ComplexObservations include mass spectroscopy or the making of a thin layer chromatography plate (TLC). In

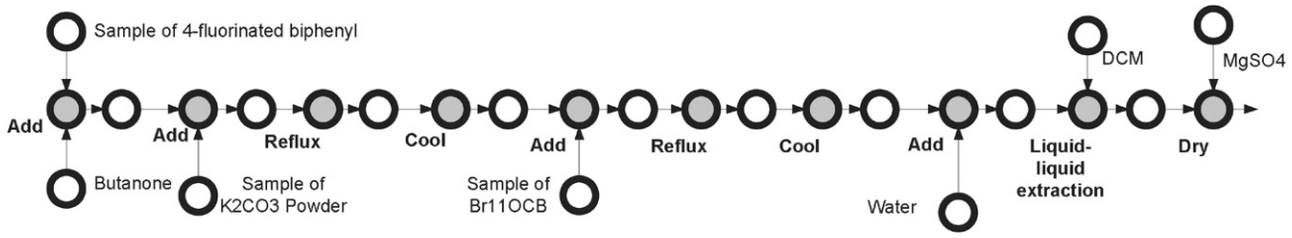

Fig. 8 Process-product spine of events in an experiment performed during the Tablet evaluations. It shows the main ingredients being added (far left), and the processes performed on that mixture. Note the alternation of processes (grey) with products (white) along the experiment path. In this diagram observations and annotations have been removed for clarity.

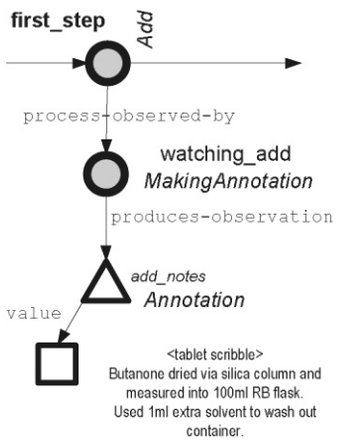

Fig. 9 RDF graph of the annotation of a process (first_step), showing the annotation process, the annotation URI itself (triangle), and the data of the annotation (squares).

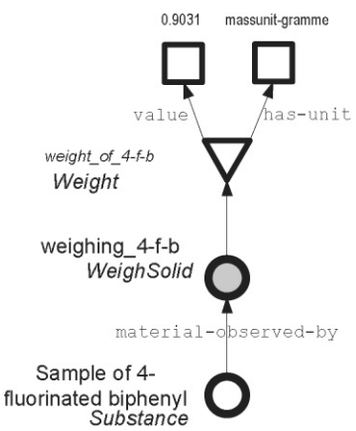

Fig. 10 RDF graph of the observation of a process, in this case the weighing 4-fluorinated biphenyl.

these processes there are many more parameters to record about the process of making the observation. The ontology does not specify the details of these but our metadata storage system will store and retrieve all statements attached to any type of Observation so there is no need to specify carefully the details in the ontology.

The metadata storage system

Experiment metadata, the connections between the results and processes in an experiment, are stored in a persistent Jena RDF store[6] backed by MySQL. We have written a system to manage this storage as a Web Service (using SOAP) with an Application Programming Interface (API, the set of functions that can be used in a client application) that serves and manipulates sub-graphs of an experiment. We call this the ModelServer. The API allows clients such as the Planner and Tablet to retrieve logical components of an experiment such as the textual steps of a plan or the nodes depicting the core record of the experiment process.

Querying the ModelServer

The ModelServer API closely matches the ontology and has evolved to meet the needs of the front end applications. For example the call getExperimentMetaData() will return the top level experiment node along with all of its properties such as the experiment name, the name of the experimenter and the first planned process node and final planned product node. From this information a client application can use subsequent API calls to navigate the experiment graph.

The method getRecord() requires the ModelServer to walk down the graph from the top level Experiment node to the first process record node and along the process-product pairs. It cannot assume that any of the nodes exist as the record of the experiment may be in any state.

The API methods return triples as a complete RDF graph, over Web Services. This requires the clients of the Web Service to be able to manipulate RDF. In the case of the Tablet it is a Java application that utilises Jena to load the RDF returned by the Web Service and work with the triples.

Searching with RDQL

Each call to the ModelServer will return a sub-graph of a full experiment. A client application can make multiple calls and add each answer to a locally cached model until it has all of the triples that it needs. For instance, the Tablet must make a large number of calls to populate the weigh-station interface, as the data come from both the plan and record sections of the experiment. The ModelServer also has a performRDQL() method in which a client can compose a search query using *RDQL* (a search language designed for finding triples in an RDF store) and the ModelServer will return the triples found by the statement. This allows clients flexibility to circumvent the standard API as required. Some of the API calls of the ModelServer are internally implemented as RDQL statements.

Generating unique URIs

To support the creation of entries in the RDF graph, we have a URI generator service. This Web Service creates unique URIs in a particular namespace on demand, and is used by many components of the system when new resources are created. This is not to say that all resources used in the RDF model must be generated by the URI generator service, or even that they must all come from the same namespace. The URIs used in the sample diagrams, for example, have been generated by hand.

Making changes to the model

As the experimenter works, the record of the experiment needs to be updated by the client applications. For example, recording a new weight for an ingredient, or adding a new note to an annotation. The method for adding or updating triples in the store is simple yet powerful: a client application supplies two RDF models containing the triples to be removed and the triples to be added to the experiment model, and calls a function in the API to make the required changes to the triple store.

A simple example is for the Tablet to update a text annotation from the bench drawing area. The sketch data is a single long

string stored in a single literal. The Tablet client must supply a model containing the triple of the old sketch and a model containing a triple for the new sketch. This may seem like a lot of work for one simple update, but good software engineering simplifies the process. The real power of this technique arises when the clients manipulate and store whole sub-graphs of an experiment. The planner does precisely this, building and manipulating whole RDF graphs as the user creates and edits the plan of the experiment. When the user presses the Save Plan button the planner updates large sub-graphs of the experiment model in one transaction. The Tablet PC is also able to manipulate whole sub-graphs of the model as required. As the user performs the experiment and ticks off steps from the bench the Tablet will create the record structure as well as annotation structures for the sketch pad.

It is important to note that the triple store will not recognise that a triple already exists and needs to be replaced. For instance if the node Experiment already has a property `experiment-description` of "making tea" which the user wants to change to "making coffee" asserting a new triple

```
[Experiment,
experiment-description,
"making_coffee"]
```

will add a new triple. A call to getExperimentMetaData() will then return two triples with the property of `experiment-description`. This is the correct behaviour for a triple store. Enforcing ontology rules and experiment structure has to be performed at a higher level by the ModelServer and the client applications.

Discussion

The ModelServer is built to walk along the spine of an experiment record or plan but cannot assume that the structures exist or have been correctly created by a client. The sub-graphs shown in Fig. 8, 9 and 10 are samples from a single diagram depicting the whole graph for a three stage experiment. This diagram, which can be downloaded from our project web site `http://www.smarttea.org`, was created to help the developers correctly write the software and has been an invaluable tool in the development cycle.

Using RDF provides a considerable advantage. If a new application needs to store new data about an experiment then it simply needs to add the triples to the store. For instance a new piece of metadata could be attached to the top level Experiment node. The ModelServer has been written always to return all triples for all nodes as it retrieves sub-graphs. Therefore the new triple is guaranteed to be returned. Clients that are not written to use the new property simply ignore triples they do not expect. This contrasts with the disruption caused by the adding of a new field to a relational database table.

Chemistry experiments vary considerably as procedures change and the Comb*e*chem project has had first-hand experiences of the difficulty of using relational databases to store and manage metadata for complex chemical procedures. A relational database intended to store data for a physical chemistry experiment had serious practical difficulties, due to the fact that the experimenters change the type and format of their experiments at regular intervals, and the database needs to be constantly maintained, which has proven to be impractical.

The design of our ontology does not provide a rigid structure for an experiment; it provides an agreed vocabulary to describe the concepts involved. An RDF API tool such as Jena does not provide constraints on a model even if an ontology is loaded. It is still possible to add any triple into the model whether it conforms to the ontology design or not. Therefore our ontology design and structure of an experiment is created and maintained by the software. There is no underlying rigid structure enforcement as there would be by a relational database.

There are two issues in the design of the client-side code. The first is that the client-side code handles RDF, and it is RDF that is passed over the network. The second is that, in our design, the RDF is stored in a "disconnected" manner. Each object in the client-side API is stored as a separate object, containing a small RDF model describing that object only. This has the advantage of separation of the objects into clearly-delineated components, making the use of them in the client-side code simpler. The main disadvantage is that the models describing these objects must be combined into a larger model in order to do inferencing over them.

Inferencing

When storing information using semantic technologies such as RDF and OWL, it is possible to use the ground-rules laid down in the ontology along with the information being stored to determine additional information automatically. As a simple example, a process marked as being of type "FilterWithBuchnerFunnel" can be automatically inferred also to be of types "Filter" and "Process", because of the relationship between the process types (see Fig. 6). This process can be automated, and is known as *inferencing*.

The current triplestore system uses Jena's relational database API to store experiment metadata permanently. Our ontology is an RDFS file stored separately. We are in the process of experimenting with permutations of using the RDFS file to provide inferencing capabilities. We have used it on the ModelServer within Jena by merging the ontology with the data model into an inference model at run time. As our ontology is written in RDFS and we have used the Jena RDFS inferencing rule set, the inferred relationships mainly consist of those given by the ontology hierarchy.

For a typical experiment the inference model contains 10 times as many triples as the original model. The inferencing model can only infer relationships by querying the database-backed model with a severe performance penalty. The major problem is that changes to the raw metadata will cause the inferencing system to start again. Such a system is not going to provide scalable performance at this present time. Loading the whole model into memory would give us the performance we need but is not compatible with needing to store data permanently. The client software can also make use of the ontology to provide inferencing. The Tablet system uses Jena for its RDF manipulation so it could also be set up to do inferencing over the sub-graphs it uses. As ontologies become more complex, and written in more expressive languages (*e.g.* OWL as opposed to RDFS), it becomes possible to infer more information in this way. This is one of the driving factors for our planned change from RDFS to OWL.

Workflow

Since there are no general restrictions on the number of inputs or outputs to a process, it is possible to create not just linear plans and records, but arbitrary networks of processes. Such networks, when created with the Comb*e*chem ontology, represent the logical parallelism of tasks, but are not as flexible in specifying the execution semantics of the network as, say, BPEL[13] or OWL-S.[14] As an example, optional branches cannot be indicated, nor can resource-limitation constraints (where two parallel tasks depend on the same piece of equipment, but not on each other). This is not a serious problem, given our goals, as we are not developing a scheduling system, or a prescriptive workflow language.

Background

Having described our work in detail as well as the concepts and technologies behind it, we are now in a position to place our work in context. In this section we briefly discuss relevant work in the areas of lab book replacements and efforts to create ontologies for areas of science such as biology and chemistry.

Lab book replacements

There have been many systems developed that attempt to replicate the features of the lab book in different ways. These include both commercial systems and academic research projects. They range from the electronic storage and indexing of paper-based notes[15] to being fully integrated into the lab environment,[16] and from duplicating the free-form style of the lab book through to providing detailed structure for all parts of the experimental process. Below, we describe briefly a range of existing lab book systems, both commercial and academic.

Commercial systems similar to SCRIP-SAFE and PatentPad[15] are largely deployed to strengthen intellectual property (IP) claims. They use specialised stationery containing user-specific metadata, which are scanned into the system after use. The pages can then be searched by metadata, but not by their content, which is retained as a bitmap only.

a-Book[17] provides devices which literally augment paper-based lab books. These augmentations, such as attaching a lab book to a Tablet, or providing a PDA to act as an annotation lens over a lab book page, attempt to allow the scientist to continue to use the familiar lab book while adding additional devices to this book to enable digitization of new input. Drawbacks to this type of approach are that the system is cumbersome to use (manipulating a lab book and the "viewer" device and a pen at the same time), and that the resulting data are still not searchable by content.

The Labscape Lab Assistant[16] supports experiment planning and data entry. The scientist creates a plan for an experiment by arranging icons representing lab processes into a graph of the experiment. Throughout the lab are identical stations which display this graph. The scientist selects the appropriate part of the graph, which provides a dialogue box for data entry. This system does not claim to replace the lab book. Indeed, published papers show that the lab book is still in use, and that printouts of the graphs are taped into books where hand written entries are visible.

ELN,[18] OpenELN,[19] ChemOffice[20] and NotebookMaker[21] concentrate for the most part on providing form-based entry of experiment information at a PC workstation. This mode of operation effectively assumes that either the experimenter has a PC with a keyboard at their disposal in the lab, or that they are writing notes in a traditional lab book and transcribing to the computer after the experiment.

The main criticisms of the existing lab book replacement systems are that they either attempt to replicate the form of the lab too well or that they are geared to a specific form of a laboratory, where almost all experiments use exactly the same equipment in the same ways, and only the type and quantity of the materials being altered. As examples of the two extremes, a-Book[17] keeps the paper book and enhances it electronically (using a PDA as an overlay) for annotation purposes; at the other extreme, the LabScape[16] system is used in biology labs, and is geared more towards annotating multiple parallel samples being processed through a small number of process types, allowing the software to be greatly constrained in the types of input it needs to be able to record.

In some respects, the approaches of SCRIP-SAFE, PatentPad and a-Book are similar to ours, in that we allow free-form data entry. Where our approach is more advanced than the others is in the placing of the individual annotations within a structure representing the processes performed in the laboratory. In our system, the addition of high-level structures for the experiment record in the back-end storage systems gives useful metadata which can be used to find relevant parts of the experimental record more easily.

Ontologies for eScience

There are numerous efforts underway to create ontologies that represent concepts in science. Biology lends itself well to such work with some highly established projects to standardise concepts in areas such as genetics. The umbrella for many of these is the Open Biological Ontologies Project[22] which currently contains over 40 ontologies for various domains. This effort includes the Gene Ontology project which is creating structured vocabularies of gene product attributes.

The Microarray Gene Expression Data Society[23] is concerned with describing samples and processes used in microarray gene experiments. Their first achievement was a format for the minimal annotation of an experiment. From there they have created an object model to describe experiments. This is accompanied by a toolset to aid developers to convert outputs from systems into their formats enabling data exchange. The formats are now becoming the *de facto* way to publish data in the field.

Noy and Hafner[24] developed an ontology for describing the procedural descriptions of biology experiments. The ontology models (in our terminology) processes and materials, including complex processes, and assemblies of materials and objects (such as "a pot of tea", being a pot and some tea). They tested the ontology by attempting to represent the information contained in sample paragraphs from molecular biology papers, to see if the ontology could cover all of the necessary concepts described. It was not developed based on an experience of users in laboratories but from experimental descriptions. Evaluation was done by domain experts examining the ontology and samples of it in use. This "top-down" approach has the advantage of leading to highly expressive and powerful ontologies, but misses the strongly user-centric outlook which we have brought to our system design. An interesting concept of the ontology is the notion of object histories. This was a mechanism for coping with the change of the form of a substance and hence the class of objects to which it belonged. For instance, batter becomes cake after cooking and the batter no longer exists. Our ontology does not explicitly make use of such an object history but is able to express the same process history.

The experiments that we have concerned ourselves with happen mainly in the laboratory, but many parts of experiments are run using computers only. Managing the results of such experiments is of increasing importance in *e*Science and Grid computing. The Earth System Grid project[25] is a major collaboration to create an ontology that describes the processes and data found in such large scale distributed software systems. The major high level concepts include *pedigree*, the line of ancestry of data from creation through various transformations; *scientific_use*, how the scientist used the data including parameters and configuration; other concepts include datasets, services and details of access to services. In many respects, the Earth System Grid work covers the same ground as our work, but this is not a disadvantage. One of the great advantages of ontologies written in languages such as OWL is that they are designed to extend and to reference other ontologies as required. It is our belief that there is no single ontology capable of expressing all of the concepts of chemistry experiments and therefore it is natural to make use of other work as appropriate.

XML markup

In chemistry, the most visible project relating to processing chemistry information on computers is the work creating CML.[26,27] The Chemical Markup Language is primarily an XML specification for describing chemical molecules. It has extensions to cover computational chemistry and a number of other areas. It has achieved considerable success as a common format to describe molecular data and so is used as a data intermediary between differing software systems and in many online publications. Our ontology is designed to use CML when appropriate for describing molecules or other chemistry-specific data structures.

CMCS and related systems

The collaboratory for multi-scale chemical science (CMCS)[28] is a very large project building a toolkit of resources to aid in multi-scale combustion research. The portal-based project encompasses a vast amount of work in individual systems to share resources and results in the community. CMCS is a powerful approach to managing diverse data formats produced from different analysis systems and tools such as electronic notebooks.

One of the major components of the project is SAM (Scientific Annotation Middleware).[29] SAM is a set of middleware components and services designed to aid storage of data or metadata in its native form and provide tools to map metadata from one form to another. Their approach is that rather than attempt to define a single standard format for analysis data, a middleware layer provides translators to map the metadata formats as required. Mappings are written in XML so that an arbitrary combination of analysis systems, underlying data stores and electronic notebooks can be used together through the one portal. Researchers can add new definitions of how metadata should be translated for both input into and output from the system. The system is based upon a WebDav implementation adding metadata management and notebook services layers. The project was also responsible for ELN and so has released newer versions of the ELN lab book client/server system that integrates with SAM. Earlier ELN versions used CGI but version 5 now uses the SAM system and WebDav.

The CMCS and SAM projects are by far the most advanced work in electronic notebooks and the distributed storage of diverse experimental data. Where their approach differs from the work described here is in our use of an ontology to describe the nature of an experiment and that our client/server API is designed to build a structured RDF graph of an experiment plan and record. In the ELN system the structure is tree-based, with users creating virtual pages in which they embed objects such as text, drawings, graphs. Our ontology and client/server API employ higher level concepts of an experiment such as plan and record rather than page and sub-page.

Conclusion

We have designed and developed a novel system for recording laboratory notes, using experiences and analysis of users in a synthetic organic laboratory with inputs from users in other areas of chemistry. We have produced innovations in not only the design methodology, but also in the lightweight style of the user interface, and in the human-scale representation of the information being recorded, leading to a system which supports the experimental process end-to-end, from planning of the experiment to publication at source of the process and results. User-acceptance testing of the system in a live wet-laboratory environment was extremely positive, particularly in the light of the severe adverse reactions of chemists to existing systems.

Current and future work will concentrate on the presentation of the information for an experiment in an easily-read and easily-navigated form for Publish@Source dissemination. Other areas which are in need of additional work are to improve the planner interface to allow for more complex process plans; the development of service wrappers to record *in silico* processes and results in the same way as we currently record *in vitro* processes; and developing or using a suitable ontology to describe the inputs and outputs of different process classes.

Future work

This work is part of the Combechem project and, as such, is committed to open source principles. It is planned that the technologies described here will be lodged with the OMII repository (http://www.omii.ac.uk/). The making tea design method and the tablet software from will also be evaluated in myTea project (http://mytea.ecs.soton.ac.uk/) which will take the lessons we have learned and apply them within the MyGrid project framework with bioinformaticians. It is also our intention to pursue future collaborations with other research groups in this area such as those working on CML and the CMCS project.

Acknowledgements

This work is supported by the Combechem project: (Structure-Property Mapping: Combinatorial Chemistry and the Grid), grant GR/R67729/01 from the UK Engineering and Physical Sciences Research Council. We wish to thank the chemists from the synthesis lab of Dr Martin Grossel for their time in helping us develop and test the system.

References

1 J. Frey, D. De Roure, m. c. schraefel, H. Mills, H. Fu, S. Peppe, G. Hughes, G. Smith, and T. R. Payne, Context Slicing the Chemical Aether, *Proceedings of First International Workshop on Hypermedia and the Semantic Web*, ed. M. David, University of Southampton EPrints Service Southampton, UK, 2003.
2 J. G. Frey, D. De Roure and L. Carr, Publication at Source: Scientific Communication from a Publication Web to a Data Grid, *Proceedings of Euroweb 2002*, British Computer Society, Swindon, 2002, pp. 88–90.
3 m. c. schraefel, G. Hughes, H. Mills, G. Smith and J. Frey, Making Tea: Iterative Design through Analogy, *Proceedings of Designing Interactive Systems*, ACM Press, New York NY, 2002, pp. 49–58.
4 m. c. schraefel, G. Hughes, H. Mills, G. Smith, T. Payne and J. Frey, Breaking the Book: Translating the Chemistry Lab Book into a Pervasive Computing Lab Environment, *Proceedings of CHI 2004*, ACM Press, New York NY, pp. 25–32.
5 COSHH Essentials. UK Health and Safety Executive. http://www.coshh-essentials.org.uk/.
6 Jena, A Semantic Web Framework for Java. http://jena.sourceforge.net/.
7 *Resource Description Framework (RDF): Concepts and Abstract Syntax*, ed. G. Klyne and J. J. Carroll, W3C, Geneva, 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
8 *RDF Vocabulary Description Language 1.0: RDF Schema*, ed. D. Brickley and R.V Guha, W3C, Geneva, 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
9 *OWL Web Ontology Language Semantics and Abstract Syntax*, W3C Recommendation, ed. P. F. Patel-Schneider, P. Hayes and I Horrocks, 10 February 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.
10 AKT project reference ontology. http://www.aktors.org/publications/ontology/.
11 R. Heery, M. Duke, M. Day, L. Lyon, S. Coles, J. Frey, M. Hursthouse, L. Carr, and C. Gutteridge, Integrating research data into the publication workflow: eBank experience, *PV-2004: Ensuring the Long-Term Preservation and Adding Value to the Scientific and Technical Data*, held 5–7 October 2004, Frascati, Italy.
12 Protégé Project. http://protege.stanford.edu/.
13 BEA, IBM, Microsoft, SAP AG and Siebel Systems, *Business Process Execution Language for Web Services Version 1.1*, May 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.
14 DAML Services Coalition, OWL-S specification. http://www.daml.org/services/owl-s/1.0/.
15 SCRIP-SAFE, PatentPad. http://www.scrip-safe.com/laboratory_notebooks.htm/.
16 L. Arnstein, G. Borriello, S. Consolvo, C. Hung and J. Su, Labscape: A Smart Environment for the Cell Biology Laboratory, *IEEE Pervas. Comput. Mag.*, 2002, **1**(3), 13–21.
17 K. Bøegh, C. Letondal, W. E. Mackay, G. Pothier and H. E. Sørensen, The Missing Link: Augmenting Biology Laboratory Notebooks, *Proc. UIST*, ACM Press, New York NY, 2002, pp. 41–50.
18 J. D. Myers, Collaborative Electronic Notebooks as Electronic Records: Design Issues for the Secure Electronic Laboratory Notebook (ELN), *Proc. CTS'03*, Society for Modeling and Simulation (SCS), http://www.scs.org. Proceedings: http://www.scs.org/scsarchive/search.cfm?presearch=db&dbrec=29. Paper: http://www.scs.org/scsarchive/getDoc.cfm?id=2051. Metadata: http://www.scs.org/scsarchive/docInfo.cfm?get=2051.
19 Amphora Research Ltd., http://www.amphora-research.com/.

20 ChemOffice, CambridgeSoft Corporation. http://www.camsoft.com/.
21 NoteBookMaker, NoteBookMaker ltd. http://www.notebook-maker.com/.
22 Open Biological Ontologies. http://obo.sourceforge.net/.
23 C. J. Stoeckert Jr, H. C. Causton and H. A. Ball, Microarray databases: standards and ontologies, *Nat. Genet.*, 2002, **32**, 469–473.
24 C. D. Hafner, N. Fridman Noy, Ontological Foundations for Biology Knowledge Models, *4th International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, St. Louis MO, 1996, 78–87.
25 L. Pouchard, L. Cinquini, B. Drach, *et al.*, Exploring Ontologies in ESG, *Symposium on Cluster Computing and the Grid (CCGrid 2003)*, IEEE Comp. Soc., Los Alamitos, pp. 626–632.
26 Chemical Markup Language project. http://cml.sourceforge.net/.
27 P. Murray-Rust and H. S. Rzepa, Chemical markup, XML and the Worldwide Web, Part 4: CML Schema, *J. Chem. Inf. Comput. Sci.*, 2003, **43**(4), 757–772.
28 C. M. Pancerella, J. D. Myers, *et al.*, Metadata in the Collaboratory for Multi-Scale Chemical Science, *Proceedings of the 2003 Dublin Core Conference: Supporting Communities of Discourse and Practice-Metadata Research and Applications (DC 2003)*. Conference proceedings: http://www.siderean.com/dc2003/search.jsp. Paper: http://www.siderean.com/dc2003/401_Paper67.pdf.
29 J. D. Myers, A. Chappell, M. Elder, A. Geist and J. Schwidder, Re-integrating the research record, *IEEE Comput. Sci. Eng.*, 2003, **5**(3), 44–50.