Eventually Perfect Data:
The Minimum Requirements of Scientific Data Management Software

Authors: Sean Cleveland, Pol Llovet, Robbie Lamb, Ryan Heimbuch, Ivan Judson, Jade Robbins and Gwen A. Jacobs.
Center for Computational Biology, Montana State University, Bozeman, MT 59717

Abstract

This gets written last….

Introduction: An explosion of data

The quantity of scientific data generated by labs around the world is growing every year. Not only are investigators responsible for managing this deluge of data, but they are also tasked with sharing this data with their colleagues in the scientific community (for a recent review see [1] [2]). Funding agencies are now encouraging (in some cases mandating) that scientists share their data after publishing [2] and funded the development of data and tool repositories [3]. While the availability of these tools and resources does improve the situation for the investigator, this new trend still presents a number of challenges for scientists who want to manage and share their data. Effective data sharing requires a data management strategy to ensure the organization, curation, and annotation of research data to provide useful context for their colleagues. When publishing research data, an investigator can contribute to a community repository (after being annotated to community standards[4]) or create their own web accessible database (which must be programmatically exposed for discovery). Once published, the proliferation of these resources creates challenges locating relevant or useful research data. Finding specific data repositories, individual data-sets, and analytical tools is a difficult process[5] . Managing collaboration across distributed teams multiplies these challenges. Remote collaboration is common, with each group contributing to the research data. For both data management anc collaboration, current solutions are either expensive (and often complicated) or custom (and often crude), leaving no clear solution for the investigator.

Existing software tools that have been developed to fill this need can be divided roughly into two categories: Electronic Lab Notebooks (ELN) and Laboratory Information Management Systems (LIMS). Over 30 companies offer commercial versions of these tools [31,32,33,34], but only a few of these commercial products are appropriate for the individual investigator due to cost (licensing and training). Also, many of these tools are designed specifically to support the discovery process in bio-tech and pharmaceutical companies and do not fit the needs of individual investigators [32]. E-Cat [31] and iLabber (ilabber.com) are examples of software aimed at the individual investigator market and offer relatively light-weight and hosted solutions for data management. While these products are a step in the right direction, they do not represent a complete solution to the data management problem (data management, curation, collaboration and publishing). As a result, many labs resort to creating their own data management solutions.

Regardless of the data management solution, the process of data curation itself is time consuming and often requires ontology expertise by the investigator. Hence, it is impractical to expect every project to be completely curated. Instead to promote best practices of data curation, the data management solution should allow for levels of data curation that can scale with the needs of the project: from very basic curation (tags or column maps) to curation in RDF with ontological terms from community based ontologies such as NIFSTD [27], OBO Foundry [42], Gene ontology (Ashburner, 2002), NCBO, etc. The options for data curation are so varied, that attempts have been made to standardize the data annotation process: neuroimaging community standards (NIfTI) [11,12]; the micro-array analysis standards (MIAME) [13]; standards for neural models [14,15,16] and neurophysiology [17] . Once the data is curated, it can be contributed to a community database, such as those for model organisms: ZFIN (zebrafish) [7]; Flybase (drosophila) [8]; Wormbase (C. elegans) [9]; TAIR (arabidopsis) [10]. The reward for this effort is that the value of data increases with the level of curation; data-sets with the most information about how the data was collected or analyzed are those that are the most useful to the scientific community [6].

The neuroscience community has responded to the data sharing challenge with an explosion of resources including disease-based repositories like the Alzheimers Disease Neuroimaging Initiative (ADNI) [18]; Parkinson's Disease [19]; neural system based repositories, such as Senselab [20], NeuronBank [21], CoCoMac (Kotter, 2004); repositories of specific data types such as neuronal morphologies (Neuromorpho.org) [22]; cellular and subcellular data (Cell Centered Database) [23]; synaptic morphological data (SynapseWeb) [24]; neuroimaging data (fMRIDC) (Van Horn, 2004); atlas-based repositories (Allen Brain Atlas); surface-based atlases (http://brainmap.wustl.edu), and time series data [25] (Gardner, neurodatabase.org, 2004). These repositories must be discoverable, so additional steps have been made to create portals which knit together the disparate community repositories. The Neuroscience Information Framework (NIF) [26] is a comprehensive effort to provide a portal to these many resources, by curating each resource according to a standard set of descriptors using a standardized ontology [27]. The NIF now lists over 2000, web accessible information or knowledge resources with useful data. However, getting access to the data sets is still difficult. Each resource must be described sufficiently to be discoverable, and the process of fully curating a resource is time consuming and requires a professional ontologist [28]. Many valuable resources have not been annotated or curated with sufficient detail for the data to be discoverable with data mining techniques [29,30].

Despite many successful efforts aimed at community data sharing and standardization, the emerging data infrastructure is fragile and disjointed, yet becomes increasingly critical to scientific progress. The lack of software to fill this critical need is due to a lack of understanding of the requirements of a complete scientific

data management software solution. To determine these requirements, three data management prototypes were built, each implementing very different features in order to better understand the minimum requirements of modern scientific data management software.

User Driven Data Management Prototypes

User-driven Prototypes

Our approach to validating the set of requirements for data management software involved designing and building three different prototype applications based on three separate data management use cases. The development of each prototype application was done in collaboration with scientists who were fully engaged in the specification of features, testing the application, and use in a production environment. They worked with the team throughout the development process, driving the features and design. Our goal was to build functional applications and distill from them the essential requirements for data management software. The application use-cases ranged from simple annotation and publishing, to collaborative data management, to protocol-driven experimental design and management.

We chose the Agile software engineering process to develop each prototype. The Agile [45], or Spiral [46] methodology, represents a shift from the older Waterfall [47] methodology of software engineering. The main principle of the Agile process is to first develop the minimum increment of useful functionality (as defined by the user) and then iterate the next development steps based on user feedback. This method helps create useful applications without accumulating unnecessary features or diverging from the original focus of the project. The users drive feature development and the application evolves through use. In this way we ensure that we build software *is useful* to scientists, not software that we *think* is useful (an important distinction).

Prototype Overviews

The Cricket Cercal Sensory Cell Database (CercalDB)[REF] was developed to publish and share an existing dataset of over 400 3D anatomical reconstructions of both primary sensory neurons and primary sensory interneurons from the cricket cercal sensory system. Dr. Jacobs and her colleagues have published many papers using this data over the last 10 years [52,53,54,55,56,57,58,59,60]. The goal was to create a publicly accessible database to search, view and download this data. The data would be exposed in the form of web pages for manual browsing, but would also be published as XML for use by other applications or web services. The focus of the application was on the data sharing features of browsing, searching, comparing and API implementation.

CercalDB represented a simple data sharing use-case (one which is common in data management) and is the minimally sufficient example of scientific data sharing [2]. The first challenge was the import of the legacy data, stored in the first iteration of our data management software, Neurosys. The data model for the data was simple, so much of the work was spent on the user interface and programmatic API. Easy, user-friendly navigation, comparison and download interfaces and dashboards were built in collaboration with Dr. Jacobs. The programmatic access API was a simple REST interface to the data, served as XML[61].

The Voltage Probe Database (VPD)[REF] was developed to support an NIH funded project to identify candidate voltage-activated fluorescent probes for real time imaging of neuronal activity. The project included 5 teams of collaborators who performed imaging and voltage clamp experiments on each candidate probe. This project required a database to house the experimental results and a collaborative platform to allow each group to post their experimental results and view the work of the other teams simultaneously. Interfaces were built that showed the activity of the collaborative teams in order to motivate teams and show progress. These features were valuable not just for the team, but for the NIH program officer of the grant. The development requirements of the VPD software highlighted the needs of real time collaboration and the critical importance of a flexible database system.

The Crux Experimental Management System (Crux) was designed as a proof of concept application for the Michael J. Fox Foundation in collaboration with Dr. Gully Burns at the University of Southern California Information Sciences Institute, and Dr. Alan Ruttenburg at Science Commons. The purpose of Crux was to improve the process of managing data produced by funded investigators in order to track the progress of research and outcomes of the experiments. Research funding organizations face difficult data management challenges, many that stem from the difficulty of getting a “big picture” view of the research work being done. Crux explored making this problem more tractable by using a formalized syntax for graphically describing experimental protocols, producing a data repository from that protocol, and annotating the data directly with ontological terms to allow for data analysis of both the resulting research data and the protocols that were used to generate it. Major requirements for Crux included data modeling, curation with ontologies and import and export of data in spreadsheet format.

Identification of Data Management Requirements

Data Modeling and Schema Development

The initial task in the creation of a data management solution for research is the development of the data model. From this, the schema of the data storage is built. This is the foundation upon which the data management software is built. Data models were built for both CercalDB and the VPD using the conventional data modeling method. In each case, the developers worked closely with the investigator as they described the experimental work that they were doing and the resulting data that needed to be stored. The subsequent data models were then translated into database schema to store the data. For Crux, a novel solution was used: a user-driven process where the scientist describes and annotates the experimental protocol which is then converted into a data model and schema by the software during runtime.

Conventional Data Modeling

The CercalDB was the simplest use case since it represented a simple data set that was the result of completed research, it was not research that was in-progress. The developers were able to see the complete data-set that needed to be published. The data model was simple (see Fig 1). The primary data were 3D reconstructions of neurons, each annotated with three attributes corresponding to its physiological properties: directional tuning to stimuli, receptor hair length, and cell type (based on location of the receptor cell in the epithelium). Three types of derived data are also represented within the data model, an image of the 3D reconstruction, the number of number of synaptic varicosities of the neuron and the total varicosity surface area. These data were derived from measurements taken from the anatomical reconstruction file. The CercalDB represents the ideal case for conventional data modeling: a static data-set with a simple data model.

The data model for the Voltage Probe Database (VPD) represents a more challenging (and common) use case for conventional data modeling: one in which the data model is designed based on an initial assessment of the requirements for the system with no preexisting data. The initial data model for the VPD is shown in Figure 2 and is quite simple as data models go. The Clone is a Gene Construction Kit (GCK) [62] file that describes the clone, along with some metadata. The Experiment represents the results of experimental testing: a set of one or more images with notes. The User object was included to allow collaborators to share the data but to disallow public browsing, viewing and downloading of the data. The Feedback object provided a way for users to provide text-based comments on any Clone or Experiment. This feature was explicitly requested to facilitate discussion and communication among the distributed groups on the project.

As development of VPD progressed it was discovered that the initial data model was inadequate for describing the project data. Specifically, two new additions to the data model were required, the Well and Plate objects, which are organizational objects that keep track of the production and distribution of clones. This change necessitated an update to the data model and the underlying application schema (the resulting final data model is shown in Fig 2B). While in this case the change was minor, this kind of update to the data model of a data

management application can require an enormous amount of time and effort to implement, during which time the data collection process is halted or at least impaired.

Protocol-driven Data Modeling

Most scientists have little, if any, experience in creating data models that can be used to develop a data management system for their research programs. This presents a difficult challenge to developers who want give investigators the ability to describe their data model using their data management application. One thing that the investigators do have is experience designing experimental protocols. Crux explores this idea by combining a graphical experimental design editor with a set of programmatic operations that converts the experimental design into a data model which updates the database schema (see Fig. 3A). This operation can happen Crux was designed specifically to permit the evolution of the data model over time. The experimental design editor was a component written by Dr. Gully Burns and his team at ISI and it described the experimental protocol using a formalism called Knowledge Engineering for Experimental Design (KEfED) [63] which organizes the protocol into variables, objects and actions. One of the test cases of Crux was a re-implementation of the CercalDB (see Fig. 3B). After the user is finished describing the protocol, the data model and schema is built from the KEfED editor description.

The resulting user interface creates a spreadsheet representing the underlying database tables of independent and dependent variables. This interface can then be used to add, view and update data in the system. In this case, the user of the system (the scientist) takes on the role of the database administrator, by building a complete database to store the data, simply by creating an experimental design that represents the experimental protocol. If the Experimental Design is updated, the data repository is subsequently updated to accommodate the changes in the data model. Future versions of the software will use versioning to allow users to roll back to previous versions of the experimental design so that data is never inadvertently lost.

These examples provide useful information regarding the problem of data modeling in data management applications. From the CercalDB and VPD prototype, we learn that the convential data modeling approach is well suited only to situations where the data has already been collected and just needs to be organized or published. Crux, on the other hand, allowed the user to modify the data model through an interface that was readily understandable. While this proved generally successful, the general conclusion is that if a scientifc data management system is to be successful, it must allow the user to an understandble interface through which they are able to modify the data model as the project evolves, so as not to disrupt the collection and analysis of data. Also, though none of our prototypes specifically supported it, the barrier to evolving data

models would be further removed by versioning the data within the system. This would allow the investigator to roll back to previous versions of the data should a mistake occur.

Data Curation

So that scientific data is useful when published, or so that it can be contributed to community data repository, it must be curated[25,67,68]. The curation makes the data semantically understandable so that it can be reviewed and reused by other investigators. The CercalDB used a very minimal curation methodology, a simple controlled vocabulary was used for metadata, but there was no interface for further data curation . The VPD used free-form tags and comments as a kind of curation in order to allow the collaborators to find relevant experiments. Crux represents the most ambitious prototype for data curation, as it used both controlled vocabularies and term links to semantic ontologies for full data curation.

Basic Curation

The CercalDB use case blurred the line between data and meta-data, in that most of the terms used to describe the data were meta-data about the process used to create the reconstruction. The reconstruction file is the primary data in this use-case, the meta-data provided information about the reconstruction (cell type, hair length). Three types of derived data were also stored: images, number of varicosities and total varicosity surface area. The terms used to describe the different instances of the cell reconstructions, conformed to this controlled vocabulary, although none of the terms used were curated against an external ontology. The controlled vocabulary used for the variables made it easy to recognize and search for data (Fig 5).

In addition to using controlled vocabularies where possible, VPD required more collaborative annotation tools, such as the ability for a collaborator to make comments on their data, make comments on other collaborators' data, and tag data with terms (Fig 6). This data comments and tags allowed the collaborators to easily find relevant data, and it also made it easier for the program officer to evaluate the activity and results of the research. While this functionality was critical for collaboration, the tags and comments would hold less utility if the data were to be published, since the vernacular used was internal to the project. This emphasized that data curation is not just important for data publication, it is critical for effective collaboration also.

Advanced Curation

The Crux application was designed to support data curation with terms linked to external community ontologies. The simplest type of data curation available was done by choosing the type and units of a variable. Variables could also be restricted to "allowable values" (Fig 7), a controlled vocabulary that the data values

must adhere to. Each variable, object and action in the experimental design was could be associated with one or more ontological terms. The terms were linked to the ontological source that they referenced (NCBO BioPortal [65], or OBI).

An important feature of this method of curation is that the curation process can take place *in parallel* with the other work performed in the system. The vernacular terms chosen by the user who constructed the initial experimental design need not be discarded; they could continue to be used if they are meaningful to those involved in the project. This was an important feature for the users, since ontological curation has a reputation for creating time bottlenecks in the research data management process. Note that curation of data using Crux is an optional process; it was not required for the basic functioning of the system. The data could be curated at a later time, without loss of data, in order to clarify the data for analysis or prepare it for sharing or publication.

Collaboration and Data Sharing

It is quite common for data management applications to involve the collaboration of many different investigative individuals or groups. This problem is related to data sharing in that publicly published data could be thought of as a collaboration with the anonymous public. There are two basic methods for sharing of research data: via a website or portal, or programmatically, through an API. Depending on the level of desired access to the data, a resource can be fully open access, as in the CercalDB or limited to a collaborative group such as the Voltage Probe Database or fully closed, as in the Crux application.

Role Based Access Controls (RBAC)

The basis for collaborative activity in data management is allowing multiple users to use the systtem simultaneously. While Crux and CercalDB were not collaborative applications, and as such had no user management system, this is unusual in data management applications. Generally an application will need to isolate actions and data of one user from another. The VPD was designed as a collaboration tool and required a more complex degree of data sharing involving collaborative groups working on data simultaneously, which required the implementation of role-based access controls (RBAC). The data was still available via the simple data sharing mechanisms listed above, but access controls were implemented to restrict a user's permissions. For example, users were able to post new experiment images, but only administrators were able to perform deletions of data. The authority to assign users and roles was delegated away from the programming team to the administrators of the application through the RBAC administrative interface. The tracking of user actions was done through an audit log, so that all of the activity on the site was aggregated into a live data feed that could be easily reviewed by all of the collaborators.

Web Publishing and Web Application Programming Interface (API)

We implemented the simplest form of data sharing in the CercalDB by exposing the data to be viewed and downloaded through a web-accessible database. The data could be accessed in two ways. First, the data could be browsed or searched, selected and then downloaded as a compressed archive (zip file). Second, the data could be accessed programmatically via the REST interface for the application and downloaded as XML files. The REST API was architected primarily for access of the data files by another software application, the Model Interaction Environment for Neuroscience (MIEN)[ref] neural modeling application which provides tools for development, searching, editing, execution, and visualization of biophysical models, abstract mathematical models, and experimental protocols used in neuroscience research. An interface was then built in MIEN by the developer to use the REST API of the CercalDB application to download, render and upload the neuron reconstruction images (Fig 9). This is a prime example of programmatic data-access, through a well defined API, that should be a part of any data management software to address use cases such as this one.

Collaboration and Sharing are basic requirements for data management applications. The RBAC features implemented for the VPD proved to be very valuable to the investigators and also to the program officer responsible for managing the project. All of the team members could quickly see which experiments were and were not yielding results, and the program officer could easily see the relative contributions of the different teams and allowed the program officer to track the progress of the project in real time. A fully featured data management software system should be able to share data with the public (via an anonymous public user), and with third party applications or community databases via a programmatic API.

Discussion

The techniques used to organize and store data span the gamut of complexity; from flat file systems and spreadsheets with user defined naming conventions to highly complex custom relational database systems [36]. Increasingly, as investigators want to share their data, it is becoming essential to establish provenance for the datasets, which must include the methods used to obtain the data, essential file formats and conversions and the sequence of analysis routines used to analyze or convert the data to meet standards of exchange [37]. In addition, the terms used to annotate and curate the data must be consistent across experiments, requiring a disciplined approach, shared by all members of the laboratory. Lastly, data must be both accessible and discoverable by others through web-based interfaces and data must be available via programmatic access.

The process of scientific inquiry does not work well with existing database technologies. The biggest and most complex challenge faced by scientists (or programmers) when building a scientific data management application is the task of data modeling. Data modeling requires the creation of a full, detailed description of the data (a schema) that will be used throughout the research process. Existing data modeling technologies include entity-relationship diagrams[38], UML [39], and database specific tools to create database schema from models. This data model will be used to build the database schema and store the research data. It is the foundation upon which the entire data management system is built. This paradigm could be described as *initially perfect data*, since the data must be completely described prior to the research process. Existing data storage technologies usually include a set of tools for modeling the database schema and building form-based user-interfaces, but where they provide high-performance solutions the implementation tightly couples the user-interfaces to the data models, again making the assumption the data models are initially perfect[40,41].

Scientific discovery is not static, experimental designs change, new technologies emerge and both the structure and content of a scientists' data changes as the investigation progresses; each time this happens it requires revising the data model and changing the schema. And once the data is stored, it almost always requires curation which involves yet another revision of the data model and schema. Revising the data model is not an easy task, and so to capture the true nature of research data, a evolvably data modeling system is required. This would enable ***eventually perfect data***, data that may initially be entered into the data management system flawed, but can be revised in place until it is eventually perfect.

Our analysis of creating prototype applications for a variety of scientific data-management challenges has illustrated the significance of the need to support the *eventually perfect data* process. We recognized during the development of both the Voltage Probe Database and the Crux system that the ability to evolve the schema was critical to the act of data management. This supports our previous ideas that scientific data evolves through the process of scientific discovery and scientific data is iterative. A data management solution that does not support this eventually perfect process is forced to go through the process of schema and data migrations (if it is to fully capture the data), however this process is inelegant, inefficient and often incomplete. Additionally, such scientific data is also difficult to curate (since annotations and metadata represent a similar schema evolution), and as such, the data curation is often either expensive or unfinished. This uncurated data has much less utility for the rest of the scientific community, an investigator who wants to publish their data or contribute it to a community data repository will have to manually update their data schema.

Eventually Perfect Data Application Framework

The ideal solution is a data management “framework” that directly supports the eventually perfect data process through evolvable data models. The framework must allow users to modify the schema without losing data or requiring database expertise. These modifications will impact the entire system (Fig 3), so the software must be designed to accommodate these changes gracefully. In order to maximize usability of the software, the user interface components of the software must be decoupled from the data. In this way, the user interface can be customized to the specific task instead of being tied directly to the complexity of the data (which will evolve). To preserve data integrity, the data and schema should be versioned so that every modification is recorded and the system can revert to any previous version. Ideally, each version should be annotated with a revision comment. This allows scientists to implicitly retain the provenance of their research with minimal effort and as previous studies have shown provenance is important to identifying data quality, integrity and interpretation of results[37,66,67].

The evolvable data models of the framework will allow the user to curate and share their data more easily. For the data to be analyzed by outside researchers, any shared data must be minimally curated . The more robust the annotations the more useful the data is for meta-analysis and comparison with similar data, however this annotation requires work on the part of the group sharing the information. The evolvable data models would allow the scientist to apply the eventually perfect process to easily add initial annotations of information in a lightweight manner that could become more robust as needs arise. The simplest annotations would be tags (such as units) and comments that can be applied to any data item, as used in the VPD (Fig 6). Controlled vocabularies are an example of more sophisticated curation where data entry is constrained to a certain set of predefined values, as has been shown in Crux (Fig 7). The most comprehensive form of curation would be incorporating existing community ontologies into the data by having a curator attach terms to data and schema values, as illustrated in Crux (Fig 7). The curation process should take place subsequent to or in parallel with the activities of data gathering to prevent curation from inhibiting the experimental research workflow. This would enable the project to acquire the services of an ontologist later on in the process or have an ontologist on-staff that is tasked with curating the data without becoming an impediment. The software would allow investigators to use whatever vernacular terminology they choose, and the ontologist would link the vernacular terms to ontologically precise definitions to increase the utility of the published data. Evolvable data models enable all of these curation activities in order to easily and efficiently allow for the end result of eventually perfect data.

Collaboration with other researchers (such as an ontologist) requires that the software provide RBAC for user management. When used in conjunction with the concept of a project (or experiment), users can have various roles within the scope of the project. This authorization can be used to limit users abilities to create, modify or

delete schema and data. Using these roles the framework would allow or deny users access to schema and the data they represent (as was shown in the VPD). Further, the existence of authorization directly allows for public and private data-sets and thus enables various levels of data sharing as well as data security[69].

Data managed by the framework application must be publishable, published data is analogous to data accessible to an anonymous user, so this collaborative functionality covers the use case of public data web publishing. The data itself should be exposed via an API, since the evolving database will make direct database connections problematic. API keys issued through the RBAC should be used for access to sensitive or premature data that is not ready for public exposure. If the data management framework is itself web accessible, the REST standard API should be used to expose common data formats (CSV, XML, and JSON) (Fig 10). Application developers can implement integrated data access with other services via this API.

Lessons from Neurosys

This is not the first time we have tackled the problem of scientific data management software. Our first attempt was the development of an ELN-type system called Neurosys[35]. Neurosys featured user-designed databases that combined the data, schema, and visual layout into a single, versioned XML structure. The client application provided users the ability to design and construct user interfaces for data entry, query, and retrieval. User-defined, controlled vocabularies were integrated into the system so that the vocabulary could be curated separately from the data. Although Neurosys was in some ways successful, it fell short of a robust and scalable solution. We discovered that tightly coupling the data, schema and layout as a single XML asset made new feature development and scalability difficult. As an ELN-type system, it was not designed to support data sharing with the community or export of data files tagged with appropriate metadata for use by community data repositories. Due to these shortcomings, it was not a sufficient solution, but we were able to take the lessons from the development of Neurosys and apply them to future development.

Conclusion

There does not currently exist a piece of software or a software framework that fulfills the requirements identified here for scientific data management. Due to the fact that different scientific domains have vastly different use cases, a single software solution is impractical. Thus a software framework must be developed to support the construction of scientific data management software. This framework must encapsulate the requirements of “Eventually Perfect Data & Schema”(so that …). Fully functinoal “Curation” and tools (NIF etc

becomes possible). RBAC allowing “Colloboration”. Robust APIs allow and “Sharing” and publishing. This framework needs to exist to allow for the re-use of tools and establish a consistency for better interoperabiliuty of data-repositories. This framework needs to be open-source with a permissive license that allow commercial and open source development. This environment encourages the participation of a larger community of developers and scientists that can share and re-use tools.

The development of such a framework has been engaged by Dr. Gwen Jacobs at Montana State University. In the companion papers both a high level description of the Yogo Framework and a detailed description of the implementation of novel schema evolution and transformation functionality which suppport the *eventually perfect data* paradigm. The open source implementation of the Yogo Framework (REF) and the supporting resources can be found at http://yogo.msu.montana.edu.

References Cited

1. Science (2011) Dealing with Data. Science.
2. NIH (2003) NIH Data Sharing Policy. http://grants.nih.gov/grants/policy/data_sharing/data_sharing_guidance.htm.
3. Luo X-z, Kennedy D, Cohen Z (2009) Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC) Resource Announcement. Neuroinformatics 7: 55-56.
4. Gardner D, Toga A, Ascoli G, Beatty J, Brinkley J, et al. (2003) Towards effective and rewarding data sharing. Neuroinformatics 1: 289-295.
5. Gupta A, Bug W, Marenco L, Qian X, Condit C, et al. (2008) Federated Access to Heterogeneous Information Resources in the Neuroscience Information Framework (NIF). Neuroinform 6: 205-217.
6. Van Horn J, Ball C (2008) Domain-Specific Data Sharing in Neuroscience: What Do We Have to Learn from Each Other? Neuroinformatics 6: 117-121.
7. Sprague J, Doerry E, Douglas S, Westerfield M (2001) The Zebrafish Information Network (ZFIN): a resource for genetic, genomic and developmental research. Nucleic Acids Res 29: 87-90.
8. Tweedie S, Ashburner M, Falls K, Leyland P, McQuilton P, et al. (2009) FlyBase: enhancing Drosophila Gene Ontology annotations. Nucleic Acids Res 37: D555-559.
9. Harris TW, Antoshechkin I, Bieri T, Blasiar D, Chan J, et al. (2010) WormBase: a comprehensive resource for nematode research. Nucleic Acids Res 38: D463-467.
10. Swarbreck D, Wilks C, Lamesch P, Berardini TZ, Garcia-Hernandez M, et al. (2008) The Arabidopsis Information Resource (TAIR): gene structure and function annotation. Nucleic Acids Res 36: D1009-1014.
11. NIfTI (2005) Neuroimaging Informatics Technology Initiative.
12. Patel V, Dinov ID, Van Horn JD, Thompson PM, Toga AW (2010) LONI MiND: metadata in NIfTI for DWI. Neuroimage 51: 665-676.
13. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, et al. (2001) Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. Nat

Genet 29: 365-371.
14. Crook S, Gleeson P, Howell F, Svitak J, Silver RA (2007) MorphML: level 1 of the NeuroML standards for neuronal morphology data and model specification. Neuroinform 5: 96-104.
15. Crook SM, Howell FW (2007) XML for data representation and model specification in neuroscience. Methods Mol Biol 401: 53-66.
16. Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, et al. (2010) NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS Comput Biol 6: e1000815.
17. Frank G, Paul GO, Tom VS, Simon RS, Stephen JE, et al. (2008) Minimum Information about a Neuroscience Investigation (MINI) Electrophysiology.
18. Weiner MW, Aisen PS, Jack CR, Jagust WJ, Trojanowski JQ, et al. (2010) The Alzheimer's disease neuroimaging initiative: progress report and future plans. Alzheimers Dement 6: 202-211.e207.
19. Foundation MJF (2010) pdOnline.
20. Crasto CJ, Marenco LN, Liu N, Morse TM, Cheung KH, et al. (2007) SenseLab: new developments in disseminating neuroscience information. Brief Bioinform 8: 150-162.
21. Katz P, Calin-Jageman R, Dhawan A, Frederick C, Guo S, et al. (2010) NeuronBank: A Tool for Cataloging Neuronal Circuitry. Front Syst Neurosci 4: 9.
22. Halavi M, Polavaram S, Donohue DE, Hamilton G, Hoyt J, et al. (2008) NeuroMorpho.Org implementation of digital neuroscience: dense coverage and integration with the NIF. Neuroinform 6: 241-252.
23. Martone ME, Tran J, Wong WW, Sargis J, Fong L, et al. (2008) The cell centered database project: an update on building community resources for managing and sharing 3D imaging data. J Struct Biol 161: 220-231.
24. Harris K (1999) Synapse Web. http://synapses.clm.utexas.edu.
25. Teeters J, Harris K, Millman K, Olshausen B, Sommer F (2008) Data Sharing for Computational Neuroscience. Neuroinformatics 6: 47-55.
26. Gardner D, Akil H, Ascoli G, Bowden D, Bug W, et al. (2008) The Neuroscience Information Framework: A Data and Knowledge Environment for Neuroscience. Neuroinformatics 6: 149-160.
27. Bug W, Ascoli G, Grethe J, Gupta A, Fennema-Notestine C, et al. (2008) The NIFSTD and BIRNLex Vocabularies: Building Comprehensive Ontologies for Neuroscience. Neuroinformatics 6: 175-194.
28. Gupta A, Bug W, Marenco L, Qian X, Condit C, et al. (2008) Federated Access to Heterogeneous Information Resources in the Neuroscience Information Framework (NIF). Neuroinform 6: 205-217.
29. Larson SD, Martone ME (2009) Ontologies for Neuroscience: What are they and What are they Good for? Front Neurosci 3: 60-67.
30. Larson SD (2009) Ontologies for neuroscience: What are they and what are they good for? Front Neurosci 3: 1-8.
31. Goddard NH, Macneil R, Ritchie J (2009) eCAT: Online electronic lab notebook for scientific research. Autom Exp 1: 4.
32. Rubacha M, Rattan… A (2010) A Review of Electronic Laboratory Notebooks Available in the Market Today. Journal of the Association for Laboratory Automation.
33. Goddard NH, Cannon RC, Howell FW (2003) Axiope tools for data management and data sharing. Neuroinform 1: 271-284.

34. Li H, Gennari JH, Brinkley JF (2006) Model driven laboratory information management systems. AMIA Annu Symp Proc: 484-488.
35. Pittendrigh S, Jacobs G (2003) NeuroSys: a semistructured laboratory database. Neuroinform 1: 167-176.
36. Bug W, Nissanov J (2003) A guide to building image-centric databases. Neuroinformatics 1: 359-377.
37. Dinov I, Lozev K, Petrosyan P, Liu Z, Eggert P, et al. (2010) Neuroimaging Study Designs, Computational Analyses and Data Provenance Using the LONI Pipeline. PLoS One 5: e13070.
38. Bachman C (1969) Data structure diagrams. ACM Sigmis Database.
39. Booch G, Rumbaugh… J (1997) The Unified Modeling Language For Object-Oriented Development, Documentation Set Version 1.0. Rational Corporation.
40. Bellahsene Z (1998) View adaptation in data warehousing systems. Database and Expert Systems Applications.
41. Papastefanatos G, Vassiliadis P, Simitsis… A (2008) Design metrics for data warehouse evolution. … Modeling-ER 2008.
42. Smith B, Ashburner M, Rosse C, Bard J, Bug W, et al. (2007) The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotechnol 25: 1251-1255.
43. Wallis J, Milojevic S, Borgman C, William S (2007) The special case of scientific data sharing with education. Proceedings of the American Society for Information Science and Technology 43.
44. Fielding R (2000) Architectural Styles and the Design of Network-based Software Architectures, chapter 5" Representational State Transfer (REST)". citeulikeorg.
45. Fowler… M (2001) The agile manifesto. Software Development.
46. Boehm B (1988) A spiral model of software development and enhancement. Computer.
47. Royce W (1970) Managing the development of large software systems. proceedings of IEEE WESCON.
48. Django-Software-Foundation (2005-2011) Django. http://www.djangoproject.com/.
49. SpringSource (2009) Grails. http://www.grails.org/.
50. Hansson DH (2003) Ruby on Rails. http://rubyonrails.org.
51. Sam Smoot DK (2007-2011) Datamapper. http://datamaper.org.
52. Jacobs… G (1991) Anatomical relationships between sensory afferent arborizations in the cricket cercal system. The Anatomical Record.
53. Jacobs… G (1996) Functional organization of a neural map in the cricket cercal sensory system. Journal of Neuroscience.
54. Jacobs… G (2000) Extraction of sensory parameters from a neural map by primary sensory interneurons. Journal of Neuroscience.
55. Jacobs… G (2002) Predicting emergent properties of neuronal ensembles using a database of individual neurons. … neuroanatomy: principles and ….
56. Doan C, Jacobs G (1999) Neural mapping of direction and frequency in the cricket cercal sensory system. Journal of Neuroscience.
57. Levin J, Jacobs G (1994) Construction and analysis of a database representing a neural map. Microscopy research and ….
58. Miller J, Jacobs G (1991) Anatomy and physiology of identified wind-sensitive local interneurons in the cricket cercal sensory system. Journal of Comparative Physiology A: ….
59. Miller J, Jacobs G (2002) Modeling frequency encoding in the cricket cercal sensory system. Neurocomputing.
60. Ogawa H, Cummins G, Jacobs… G (2008) Dendritic design implements algorithm for synaptic

extraction of sensory information. Journal of Neuroscience.
61. Bray T, Paoli… J (1997) Extensible markup language (XML). W3C.
62. Soderberg P, Stewart J, Calley… J (1994) Genetics construction kit: A tool for open-ended investigation in transmission genetics. Journal of Computing in ….
63. Burns G, Hovy E, Ingulfsen T (2008) Biomedcal knowledge engineering approaches driven by processing the primary experimental literature. Frontiers in Neuroinformatics Conference Abstract: Neuroinformatics 2008.
64. Adobe (2010) Adobe Flex. http://www.adobe.com/devnet/flex.html.
65. NCBO (2010) NCBO BioPortal. http://bioportal.bioontology.org/.
66. Zhao J, Goble C, Stevens… R (2008) Mining Taverna's semantic web of provenance. … and Computation: Practice ….
67. Myers J, Allison T, Bittner S, Didier… B (2005) A collaborative informatics infrastructure for multi-scale science. Cluster Computing.
68. Kennedy D (2004) Barriers to the socialization of information. Neuroinformatics.
69. Ferraiolo… D (2009) Role-based access controls. Arxiv preprint arXiv:09032171.

Figure 1: figure1.png
Cercal DB data model; Cell attributes are shown, the reconstruction is the data file, the three other types of derived data were obtained from the binary reconstruction file

Figure2: clover-data-model.png (A and B)
A) The initial VPD data model. The focus of the database is the Clone object (input data to experiments) and the Experiment object (resulting data from experiments), supported by the user, team, role and feedback objects. B) The final voltage probe database abstract data model, expanded to include the Audit Log (tracking of all activity by all users) and reorganized to create an Asset class, which is used to store both GCK files and experimental images.

Figure 3: Figure 3 shows the KEfED editor interface with the experimental design for the Cricket CercalDB. The independent variables are shown on the left as blue boxes, the objects, actions and workflow are shown as blue ovals and rectangles. The dependent variable, Reconstruction, is shown as a checkerboard boxes on the right of the diagram. Caption: The Cricket Cercal Database user interface showing the KE-f-ED diagram of the experimental design and the resulting columns of the database tables derived from the independent and dependent variables of the experimental design.

Figure 4: sends the description of the KEfED model to the application (via a JSON REST API) which then builds a data repository from the diagram shown in Figure 4.

Figure 5: When developing the application it became apparent the distinction between data, meta-data, and annotation was irrelevant to the design of the data storage layer. This distinction was critical at the user interface where users needed to be able to use the meta-data to quickly understand (and search for) what was presented about the data (Figure 5).

Figure 6: VPD comments and tags The Voltage Probe Database required more collaborative annotation tools, such as the ability for a collaborator to make comments on their data, make comments on other collaborators' data, and tag data with terms (Fig 6). Caption: Voltage Probe Database's annotation tools that allow comments to be applied to a probe's results.

Figure 7: Crux Annotation ability Caption: Crux’s annotation tools that use a controlled vocabulary.

Figure 8: VPD RBAC and Audit Log
Figure X3.4.2: screenshot of the VPD user homepage showing updates of work. Caption: The VPD dashboard.

Figure 9: NEuron reconstruction upload

Figure 10: MEIN API?