Metadata and Annotations for Multi-scale Electrophysiological Data

Mark R. Bower, Ph.D., Matt Stead, M.D., Ph.D., Benjamin H. Brinkmann, Ph.D., Kevin Dufendach, Gregory A. Worrell, M.D., Ph.D

Abstract— **The increasing use of high-frequency (kHz), long-duration (days) intra-cranial monitoring from multiple electrodes during pre-surgical evaluation for epilepsy produces large amounts of data that are challenging to store and maintain. Descriptive metadata and clinical annotations of these large data sets also pose challenges to simple, often manual, methods of data analysis. The problems of reliable communication of metadata and annotations between programs, the maintenance of the meanings within that information over long time periods, and the flexibility to re-sort data for analysis place differing demands on data structures and algorithms. Solutions to these individual problem domains (communication, storage and analysis) can be configured to provide easy translation and clarity across the domains. The Multi-scale Annotation Format (MAF) provides an integrated metadata and annotation environment that maximizes code reuse, minimizes error probability and encourages future changes by reducing the tendency to over-fit information technology solutions to current problems. A description of a graphical utility for generating and evaluating metadata and annotations for "big data" files is presented.**

I. Introduction

Multi-scale electrophysiology requires collection and analysis of data over a wide range of spatial and temporal scales, and clearly falls under the heading of "big data" [1] . Multi-scale data creates a range of difficult technical issues including efficient data formats and storage solutions. Several of these needs specific to the acquisition, storage, and analysis of data for systems electrophysiology spurred the creation of a new file format, Multi-scale Electrophysiology Format (MEF) [2]. In addition to the raw data, another set of problems arises concerning the handling of the data that describe or are attached to the raw data; i.e., metadata and annotations. While some aspects of metadata and annotations from multi-scale recordings are similar to those for all types of data acquisition (e.g., subject identification, dates, filenames), several other aspects are unique to multi-scale acquisition from humans, such as patient privacy issues and scalability. The difficulties surrounding human multi-scale data acquisition (e.g., cost, ethics, supremacy of patient needs over scientific questions) emphasize another issue regarding the handling of scientific data: analyses commonly involve data that were collected over multiple years on an array of recording systems by different people asking a range of questions, and often located at different institutions. This creates a range of technical problems, including the problem of communicating the conditions under which data were acquired, variations in data acquisition parameters (e.g., sampling frequency) and the sheer numbers of observations inherent in such large data files. These challenges can be grouped into three problem domains: communication (e.g., between users, between applications), storage (e.g., across time, across institutions), and analysis (e.g., dynamic data manipulation, statistics). Software within each domain are optimized to solve the problems that dominate that particular domain, but a new problem arises when research questions bridge multiple domains. Smaller, more focused data sets often avoid dealing with these problems by a range of simpler solutions that may not scale to multi-terabyte data files.

For example, consider computing the spectral power within a fixed bandwidth during each second for data recorded from 50 patients, where data from 300 channels/ patient were obtained for several days. Furthermore, imagine that some patients were recorded at one sampling frequency, others at a second frequency and others at a third. Finally, imagine that the results are to be sorted based on the anatomical location of each electrode (e.g., frontal lobe, hippocampus). If the scope of the project involved fewer channels, patients or shorter recordings, many of the difficulties in this scenario could be handled manually (e.g., a lab notebook or a spreadsheet). Several solutions to different parts of this problem would seem reasonable: each file could be filtered for its specific sampling frequency, each file could be filtered and down-sampled to the lowest sampling frequency, the files or the results could be moved to folders based on recording location anatomy, or file-specific information could be stored in and extracted from the header of each file. This particular analysis, however, involves 15,000 files, many of which could have sizes exceeding one terabyte. Who would do all of this standardizing and moving? How would they know it had all been done correctly? How long would this take, and would it all have to be done again for the next analysis?

A more reasonable solution might be to store the information in a database, but storing the results into a database would require a table with (15,000 files) * (3600 sec/hr) * (24 hr/day) * (7 days), which is approximately 10 billion rows! The next analysis would add another 10 billion rows. In short, manipulating the data, metadata or annotations to best suit one particular problem often creates more work to solve the next problem. Because we cannot predict what questions we will ask for all future times, a more reasonable alternative might be to search for general solutions within the individual problem domains and then determine whether those solutions can work together, specifically in regards to the problems that would be expected for "big data" arising from systems electrophysiology. Fortunately, general solutions have been found in the areas of data communication, storage and analysis, which offer significant advantages to system electrophysiologists.

Solutions in each of these problem domains has evolved in response to different goals, which has often led to

This work was supported by the National Institutes of Health (Grant K23 NS47495) , State of Minnesota Partnership Grant, andEpilepsy Therapy Development Project grant from the Epilepsy Foundation of America.

M. R. Bower, M. Stead, B. H. Brinkmann, K. Dufendach and G. A. Worrell are with the Mayo Systems Electrophysiology Lab, Rochester, MN 55905 USA (303-555-5555; fax: 507-284-4795; e-mail: bower.mark@ mayo.edu).

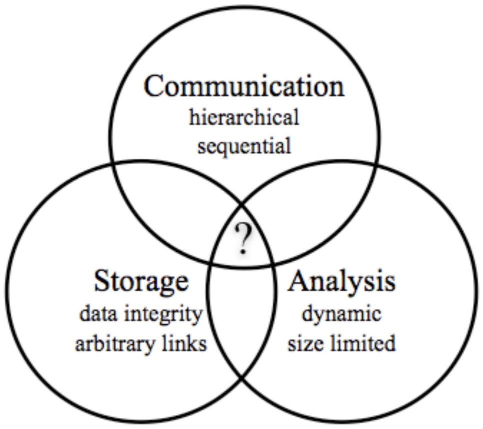

Figure 1. Problem domains involved in handling multi-scale electrophysiological data. An integrated solution would take advantage of the strengths of existing, tested solutions at the intersection of all three; i.e., would allow easy translations between the domains.

somewhat divergent ways of looking at data in general (Figure 1). Arranging information in a context-free manner for communication between computers or between human readers has often focused on methods to serialize data; that is, methods by which the meaning within data is maintained even when the data must be broken into a series of values for transmission along a communication channel. Perhaps the most surprising result of this work is just how generally applicable and powerful such representations can be. We are all certainly aware of the descriptive power that can be contained in messages passed across a network of multiple computers, as shown by the example of the internet. Equally surprising is that the vast majority of computer communications are encoded in just a handful of protocols and formats. TCP/IP is the standard for network traffic; XML/HTML is the standard for text-based information and the internet; PDF is the standard for image-quality documents; JPEG and TIFF handle most of the digital storage of images. Communicating metadata and annotations between programs or users according to one of these protocols would give users access to a range of tested, powerful tools that convey information flexibly and dynamically.

Storing information in an efficiently searchable format that also reduces the incidence of errors also has a long history that has centered on the concept of databases. We would expect that a hospital, internet website or university administration would store their patient, visitor or student data in a database, as do geneticists and molecular biologists (e.g., GenBank) [3]. Again, perhaps surprisingly, virtually all databases use a handful of tested tools with a long track record. The SQL programming language lies at the heart of virtually all database software whether the vendor is Oracle, Microsoft or an open-source provider. Open Database Connectivity (ODBC) provides a widely accepted set of programming standards that provide easy access to database data across a range of programming languages. Storing metadata and annotations in a database provides users with increased access to their data in terms of searching and sorting, while reducing the chances of error or data corruption.

The situation regarding the preferred programming language for data analysis, however, is not so clear. A long list of programming languages have been used in systems electrophysiology (e.g., C, C++, R, Java, Fortran, Matlab, Python) and that diversity is compounded by a range of operating systems (e.g., Windows, Mac OS X, Linux). The advent of web-based programming languages (e.g., PHP, JavaScript) adds even more options. Several programming concepts, however, have attracted a great deal of attention in the software community (e.g., object-oriented programming, software "design patterns", agile programming) that have led to increased portability and reusability of software components. Computing with metadata and annotations stored as modular data objects package into publishable Application Programming Interfaces, or APIs, could reduce the complexity of analysis software, increase cross-platform transportability, and promote increased code reuse by multiple researchers and labs.

In each problem domain, general solutions to a vast range of problems have been optimized through decades of research and products. Many of these solutions are freely available in the form of open-source software, which could provide the added benefit of freeing data from being tied to the particular software vendor that supplied the recording equipment; i.e., "vendor lock-in". In this paper, we describe ongoing work to develop MAF, the Multi-scale Annotation Format, which attempts to find a set of software solutions within each problem domain that can interact with one another to provide an integrated efficient, fast, error-minimizing, reusable, open-source and scalable suite of software tools tailored to handling "big data" metadata and annotations.

II. Methods and Results

In describing various software packages and alternatives, the profusion of acronyms can begin to look like alphabet soup! To avoid some confusion, this paper will not attempt to list all possible alternatives for a given problem domain, but this is not meant to suggest that the solutions mentioned were the only solutions considered, nor is this meant to suggest that the suite of solutions presented here is necessarily the best. As will be addressed in the Discussion, the software world repeatedly shows that there are many ways to solve any particular programming problem. The considerations that drove the choices described in this paper included: ease of use, applicability to the problem within a given domain, breadth of current use and ability to interact with other technologies.

The first choice faced by the MAF project was whether to favor a single programming language. Practical development issues forced the use of at least one language with which to test ideas. But, the choice of programming language is highly personal, and debates regarding which language is "best" can rival those of politics! That said, languages that supported object-oriented programming offered several useful advantages including increased code reusability and the ability of objects to read, write and otherwise maintain themselves [4]. This is particularly valuable when long periods of time may pass between the creation and use of data, or when the software required for the reading of those

```
public class Subject implements Tag, Iterable<String> {
   private TreeMap<String,String> map;
   String nameString;
   String[] columns;
   public Subject() {
      map = new TreeMap<String,String>();
      columns = new String[]
         {"name_first","name_last","mcn"};
   }
   public void add(String a, String b) {
      map.put(a, b);
   }
   public Iterator<String> iterator() {
      return map.keySet().iterator();
   }
...
}
```

Figure 2. Example Java class for containing data related to experimental subjects. Object-oriented programming uses classes to encapsulate data, increase modularity and utilize data abstraction (i.e., reduce the responsibility placed on the user to maintain data integrity). The use of class-based objects increases the clarity of subsequent analyses by allowing programs to mimic language-based logic; i.e., in terms of relations between "things", rather than manipulations of data structures.

data can be lost or modified (Figure 2). In addition, the tenets of Design Pattern programming found a prominent place in programming paradigm, if only in the form of asking future programmers to "program to an interface" [5,6] in the form of an API (Application Programming Interface). An API can be thought of as a contract between programming language creators and users regarding the names of functions within a software package and the number and types of variables that those functions expect. One analogy is the three-pronged electrical outlet: designers of consumer electronics (that is, "users") need only to know the physical separations of the prongs and the voltage, current and frequency of the electricity, without worrying about how the electricity is generated by the power company (in our case, the software writers). Likewise, the power company aims only to supply power to the plug that conforms to the standard (the API), without worrying about what gets plugged in. The goal of MAF can be stated more concisely as an attempt to find APIs that provide solutions within problem domains that interact flexibly and easily with the APIs chosen in the other problem domains. The language chosen for the development of MAF was Java written in the Eclipse IDE (Integrated Development Environment), because Java objects can be loaded easily into several different interactive environments, including Matlab (http://www.mathworks.com) and Jython (http://www.jython.org).

Several factors influenced the choice of the communication format. Clarity of information transmission is vital, particularly when months or years can separate recording and analysis. Such delays can lead to confusion regarding terminology or conventions, or to changes in the overall design of the annotation processing stream. If the communication protocol was not flexible, then it would have to be redesigned on a regular basis. Such considerations led to the development of the eXtensible Markup Language (XML), a flexible specification for sharing structured data (http://www.w3.org/XML). In the context of neuroimaging studies, the XML-based Clinical Data Exchange (XCEDE) schema was developed by the Biomedical Informatics

```
<?xml version="1.0" encoding="utf-8"?>
<MAF>
   <subject id="1" dbID="11" mcn="0-000-000" name_first="John"
               name_last="Doe" />
   <task id="1" dbID="21" subjectID="1" author="unknown" method="visual"
               date="1149260098374351" units="uUTC"/>
   <source id="1" dbID="31" subjectID="1" name="Macro_01.mef" />
   <event name="seizure" type="onset" id="1" dbID="41" taskID="1"
               sourceID="1" time="1149260078374351" />
   <event name="seizure" type="offset" id="2" dbID="42" taskID="1"
               sourceID="1" time="1149260098374351" />
   <event name="HFO"  type="point" id="3" dbID="43" taskID="1"
               sourceID="1" time="1149260099404351" />
</MAF>
```

Figure 3. Example of XML for metadata and annotations of multi-scale data (MAF). Each element has an "id" attribute, denoting an identifying string that is unique for all elements of that type in the document. Elements retrieved from the database also contain a "dbID" attribute, which is unique across the entire database. Each element-type has a matching class, and the attributes within the tag are matched by variables within the class and columns in the database.

Research Network (BIRN, http://www.nbirn.net) to document research and clinical studies. The schema easily represents "many-to-many" relationships (e.g., between data files and analyses, where one analysis can use multiple files, and one data file can be used in multiple analyses), which can pose a problem for hierarchical schema. XCEDE proposes assigning reference IDs to each element that can subsequently be used flexibly (Figure 3). This schema integrated easily with the other problem domains, particularly with that of data storage, because it allows for representations of arbitrary relations between data elements. XML tags extend naturally to object-oriented classes representing subjects, recording episodes, analysis tasks, EEG events, etc. In particular, we chose the JDOM package of XML parsers and document objects (http://www.jdom.org), because it presents an integrated set of objects that can be loaded easily into any Java-capable environment.

Data storage technologies include a broad range of options: files within hierarchical folders, XML, spreadsheets

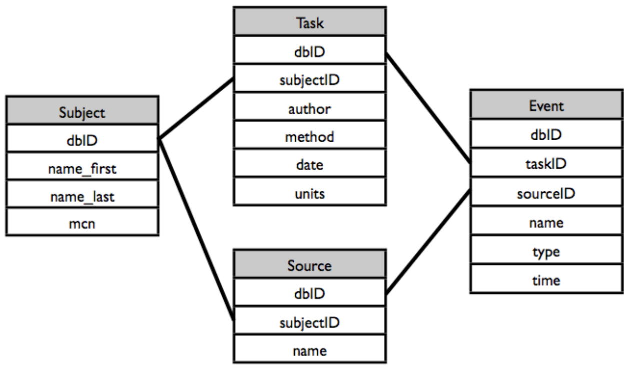

Figure 4. Database schema example showing tables for the classes and tags described previously in Java and XML. Links between reference IDs in the tables help enforce data integrity rules, and allow users to retrieve data in multiple ways. Creating a different table for each object-oriented class/ XML element-type seems to be a reasonable, intuitive first step in arranging the database.

and databases each offer advantages. Storing data within a system of hierarchical folders distributes information across the folder tree; the source of data in a file is a function of the chain of parent folders. This tends, however, to enforce, fixed relationships (e.g., patient/session/channel), does not prevent errors (e.g., if a file is misplaced or dropped) and is difficult to search. Spreadsheets, too, bias storage towards a fixed relationship, holding all information in a single table, and attach meaning to location within a table. As with hierarchical files, it is difficult to catch errors due to misplaced or dropped data. Databases were designed specifically to address these issues. Relational databases, in particular, minimize errors by enforcing data integrity rules, requiring all related information to be entered in a state that is consistent rules and preventing the deletion of data that are required by an existing element. Relational databases link related information via unique identification numbers, allowing arbitrary relations to be extracted after data have been entered and preventing accidental deletion of necessary data. Another advantage of database storage is that information related to each entry is stored in only one location; if an error is identified in an entry, it only needs to be corrected in one place. Data can be entered and retrieved by software through drivers that conform to the Open DataBase Connectivity (ODBC) interface. Although relational databases are the most commonly used type, several other database paradigms could offer useful features in the future. In particular, two technologies appear promising: object-relational map databases utilize an intermediate description of classes to automate the storage and retrieval of data within objects into a relational database (e.g., http://www.hibernate.org and http://www.datanucleus.org); object-oriented databases store the objects themselves into a modified relational database (http://www.db4o.org); distributed databases allow storage of very large data objects (http://www.project-voldemort.org), which could be particularly useful for multi-scale electrophysiology. Although not required, creating a different database table for each object-oriented class seems a reasonable first step. This creates a class-tag-table organization based on commonly used clinical and research components (e.g., subjects, EEG events). It also provides a solution to the problem posed in the introduction, namely the persistence of large vectors of annotations (e.g., timestamps of action potential from individual neurons). These data can be stored efficiently within data objects that are then stored to disk, while references to the file are stored in the database. Currently, MySQL, a popular relational database, was chosen as the initial database underlying MAF, because it is freely available, is easy to install and use, and publishes drivers for most ODBC standards.

The first project to be implemented using these guidelines was an interactive, Matlab-based data viewer for adding and visualizing neurological annotations to patient EEG, which is called "eeg_view." Annotations are label-time stamp pairs associated with neurological events. Events consist of a type-task pair, can contain one or more annotations, may contain or be contained by other events, may be identified visually by an expert or automatically by a program, and are generally associated with a specific analysis task (e.g., to identify all of the seizure onsets in a file). Annotations (particularly those generated by software) may be inspected visually, allow users to accept or reject existing annotations and add events that were missed. MAF XML records users rejections, as well as additions, so that false positive detections can be counted.

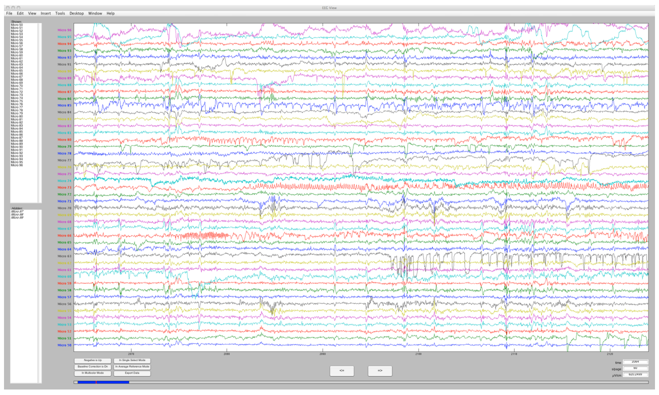

Figure 5: Screen capture from Multiscale IEEG viewer. The ms-IEEG viewer is an interactive tool for marking and visualization of microwire and macroelectrode activity. The annotations are passed to a database via XML

The user can load or generate events for multiple EEG files. Annotations are saved to an XML file corresponding to the MAF format (Figure 3). The XML file can be transferred directly to another program or stored to the database. Associating events with tasks in the database allows users to select different sets of events based on the method by which they were generated, combine events generated by different sources, or keep only those events that have been designated as the "gold standard". This kind of flexibility in relation to annotations for multi-scale data files is the goal of the MAF project.

III. Discussion

It should be stressed that the technologies selected and combined in this paper represent only one possible solution to the problem of managing metadata and annotations in big data. Many more solutions are possible with current technology, and future developments promise to expand these opportunities. For example, the current choice of technologies described in this paper underutilizes the power of browser-based technologies. One of the leading contenders under consideration to replace our current, Matlab-based "eeg_view" program is Google Web Toolkit, which allows existing Java code to run within a browser without modifications (http://code.google.com/webtoolkit/). One driving consideration is that the suite of technologies chosen should be useful to system electrophysiologists; those technologies should offer real-world assistance to challenges in ways that bridge the problem domains. Utilizing a given technology because it is new, clever or popular does little good for the practicing researcher or clinician if usage of that technology creates more problems outside its domain of specialty. Another consideration is that solutions should provide for their own future replacement; technology lock-in can be just as insidious a problem as vendor lock-in. Solutions that rely on a feature specific to one particular technology should be avoided. The hope is that a community will develop around a suite of software tools where the modular nature of individual tools and techniques will updates and changes in one domain with little or no impact on the other domains.

IV. CONCLUSION

Multi-scale electrophysiology poses challenges both in terms of the amount of data and the complexity of analyses. Attempting to solve these challenges by focusing either on a single problem or on particular technology often creates new problems in transferring information from one task to the next. General purpose solutions exist to the problems of communication, storage and analysis of metadata and annotations that can alleviate the problems of complexity and lack of portability, but doing so requires integration of software tools tailored to the clinical and research needs. The Multi-scale Annotation Format (MAF) integrates robust solutions within each of the problem domains of communication, storage and analysis in the forms of XML, relational databases and object-oriented programming by promoting a class-element-table approach to data representation. These proven technologies not only offer solutions to current problems, but modularize the problem domains, providing the opportunity for future improvements without impacting existing information experimental information. The goal of this work is an integrated annotation environment that will be freely available under GNU open-source licensing.

ACKNOWLEDGMENTS

REFERENCES

[1] Howe D, Costanzo M, Fey P, Gojobori T, Hannick L, Hide W, Hill DP, Kania R, Schaeffer M, St Pierre S, Twigger S, White O, Yon Rhee S. (2008) Big data: The future of biocuration. Nature. 2008 Sep 4;455 (7209):47-50.

[2] Brinkmann BH, Bower MR, Stengel KA, Worrell GA, Stead M (2009) Large-scale Electrophysiology: Acquisition, Compression, Encryption, and Storage of Big Data. J. Neuroscience Methods (in press).

[3] DA Benson, MS Boguski, DJ Lipman, J Ostell, BF Ouellette, BA Rapp and DL Wheeler. GenBank. Nucleic Acids Research 1999. 27 (1):12-17.

[4] Eckel, B. (2006) Thinking in Java. Prentice Hall.

[5] Gamma E., Helm, R, Johnson R, Vlissides JM (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

[6] Kuchana P (2004) Software Architecture Design Patterns in Java. Auerbach Publications.