

A REFERENCE DATA MODEL FOR NEESGRID SHAKE TABLE EXPERIMENTS

J. Peng and K.H. Law

Department of Civil and Environmental Engineering, Stanford University
Stanford, CA, USA

ABSTRACT

NEES (the George E. Brown, Jr. Network for Earthquake Engineering Simulation) is a national, networked resource for next-generation experimental earthquake engineering research and education in the United States. The goal of NEES is to support open access to and use of NEES's facilities and data by the earthquake engineering community. The system infrastructure to support the NEES activities is the NEESgrid which is intended as a distributed virtual "collaboratory" cyberinfrastructure for earthquake experimentation and simulation. Building on grid technologies, this collaboratory will allow researchers to gain remote, shared access to experimental equipment and data. Tools are being developed to support the sharing, access, and utilization of the NEESgrid data repository, and thus enhance efficient communication among the researchers in earthquake engineering. One of the key ingredients required to bridge between the data repository and the experimental users is the development of data models that provide the "semantic" information describing the project and experimental data. This paper describes the current effort in developing a reference data model for NEESgrid shake table experiments.

INTRODUCTION

In order to facilitate collaboration within the NEES framework, one of the key services that NEESgrid needs to support is with respect to the data and metadata for earthquake engineering simulations. It is well known that engineering design and manufacturing activities generally involve a large set of independent but interrelated data items (Law et al. 1987). Traditional hierarchical, network, and relational database models, which are designed for highly structured commercial applications, do not adequately support technical engineering problem domains. To support the expressive concepts and the semantic content of engineering data, object oriented data models are often employed. Influenced by the field of artificial intelligence, semantic relationships such as classification, association, aggregation and generalization can be used for organizing and structuring engineering data (Law 1988). Besides the mechanism needed to represent and manipulate data, data model development also requires some knowledge on the intended use of the data (Law et al. 1991). For this purpose sample data sets are being developed that demonstrate the features of the data models along with scenarios for the use of the data and models. Specifications for the tools necessary to support entering, importing, storing, searching, and extracting data from the repository are being proposed and developed. This paper briefly describes the current effort in developing a reference NEESgrid data model, focusing on shake table experiments.

A data model is in essence a representation of the data and their interrelationship and provides a conceptual or implementation view of the data. A **data model** can be viewed as the “grammar”, “vocabulary” and “content” that represent the types of “information” stored in a “system”. The **grammar** defines the relationships among the data **elements** in the system; the **vocabulary** defines the terminology used to describe these elements; the **content** defines what is to be included in the system. Within the scope of the NEESgrid data/metadata effort, **data** is defined as *all* of the project related information and encompasses **observational** (or *acquired*) data recorded prior to the experiments and during the experiments by means of sensors, cameras and the like; **computational** (or *generated*) data for and as a result of simulations, post-processing, etc.; **literature** in the form of reports, journal papers, drawings, etc. Associated descriptive and related data, i.e. **metadata** (or data about the data), are defined and published in a prescribed (by NEESgrid) format and language.

DATA MODELING TOOL AND APPROACH

There are many existing data modeling techniques and tools that are available to help design a database and to structure the data. A brief review of relevant techniques and approaches has been reported earlier (Peng and Law 2004a). This section briefly discusses the tool selected for developing the NEESgrid data model and the basic concepts of object-oriented data modeling.

Protégé – An Ontology Development Tool

To facilitate the design of the NEESgrid data model, we selected Protégé (<http://protege.stanford.edu>), which is an open-source software package designed to help developing ontology for knowledge-based systems (Gennari et al. 2002). Ontology represents explicit formal specifications of the terms in the domain and the relations among them (Noy and McGuinness 2002). As open source software, Protégé has attracted a wide variety of plug-ins from around the world to enhance its capabilities. Some of these software plug-ins allow a model developed in Protégé to be exported in many standard formats, including UML (Unified Modeling Language (Arlow and Neustadt 2001)), XML Schema (<http://www.w3.org/XML/Schema>), RDF (Resource Description Framework, <http://www.w3.org/RDF/>), OWL (Web Ontology Language, <http://www.w3.org/2001/sw/WebOnt/>), and others.

In Protégé, a graphical user interface (GUI) is provided to facilitate ontology development. The interface enables the modeling of an ontology of classes to describe a particular subject with a set of concepts and their relationships. The interface also allows direct entering of specific instances of data and the creation of a knowledge base. Figure 1 shows an example of the GUI, with the view of classes shown in the left window, and the view of detailed attributes of a class (e.g. Project class) shown in the lower right window.

Object-Oriented Data Modeling

Object-oriented data modeling approach is employed in the development of NEESgrid data model. In an object-oriented data model, information is modeled as **objects**, which can be any sorts of entities (Rumbaugh et al. 1990). The general representation of certain type of objects is called a **class**, which represents explicit description of concepts in a domain. The creation of an object of a certain class is called **instantiation**. The relationship between an object and a class can be viewed analogically in a procedural language in that a variable being a particular instance of a pre-defined type such as an integer. For example, **Project** is modeled as a class, and a Mini-MOST experiment (Nakata et al. 2004) is an object instance of the **Project** class.

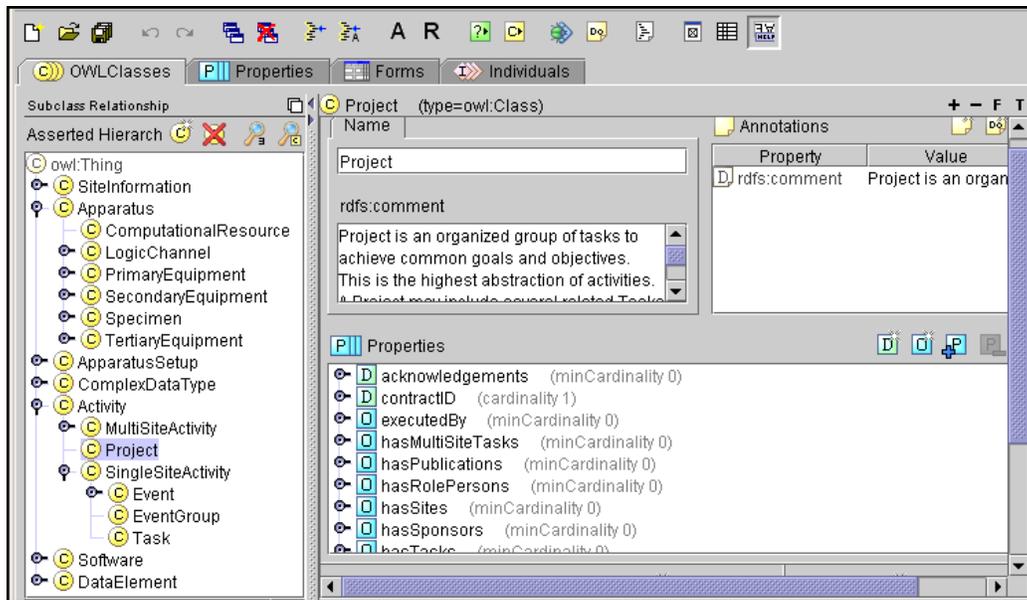


Figure 1 – Protégé Interface with OWL Plug-in

An object encapsulates certain related data as **slots**, which are also called **attributes** or **properties**. Slots can have different **facets** describing the value type, allowed values, the number of values (**cardinality**), and other features of the values the slot can take. A value-type facet describes what types of values can fill in the slot. Some common value types are string, number, Boolean, enumerator, and object instance. Allowed objects and the variety of values (e.g. minimum number, maximum number) of a slot are often referred to as a **range** of a slot. Slot cardinality defines how many values a slot can have, such as single (at most one value) or multiple (more than one value). For example, “minute” is a slot of class **Time**, with data type defined as integer, cardinality one and range from 0 to 59.

One important feature of object-oriented modeling is the concept of class hierarchies, with slots of a **superclass** being inherited by its **subclasses**. This **inheritance** feature allows us to define the common slots used by several classes at the highest possible level in the hierarchy, which avoids the duplication of slots at the lower levels. A class can have subclasses that represent concepts that are more specific than the superclass.

A REFERENCE DATA MODEL FOR NEESGRID

As depicted in Figure 2, the NEESgrid data/metadata effort is working towards producing end-to-end solutions that integrate site specifications database, project level model, domain specific data models, and common elements. To capture all these data, the reference data model is designed to include six base classes, namely SiteInformation, Activity, Apparatus, ApparatusSetup, DataElement, and ComplexDataType. The high-level class diagram of the reference data model is presented in Figure 3, which shows the association relationship among classes. (The ComplexDataType class, which is employed to support other base classes, is not shown in the figure.) The association relationship exists between classes when an object of one class *knows/contains* an object of another class. For example, a Project object knows about its Tasks objects, a Project also contains Organizations, Sites, and RolePersons. RolePerson is in turn defined as the combination of a Person and his/her role in a Project. The arrow in Figure 3 denotes the direction of the relationship *contains*; i.e., $A \rightarrow B$ indicates that class A *knows/contains* class B.

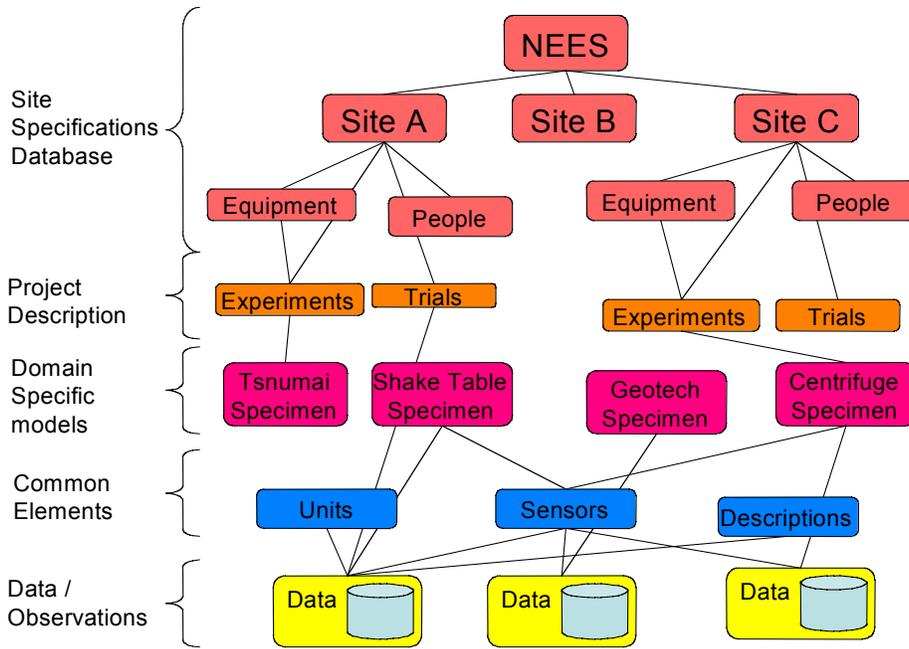


Figure 2 – Overall Data Model for NEESgrid (Courtesy of Chuck Severance)

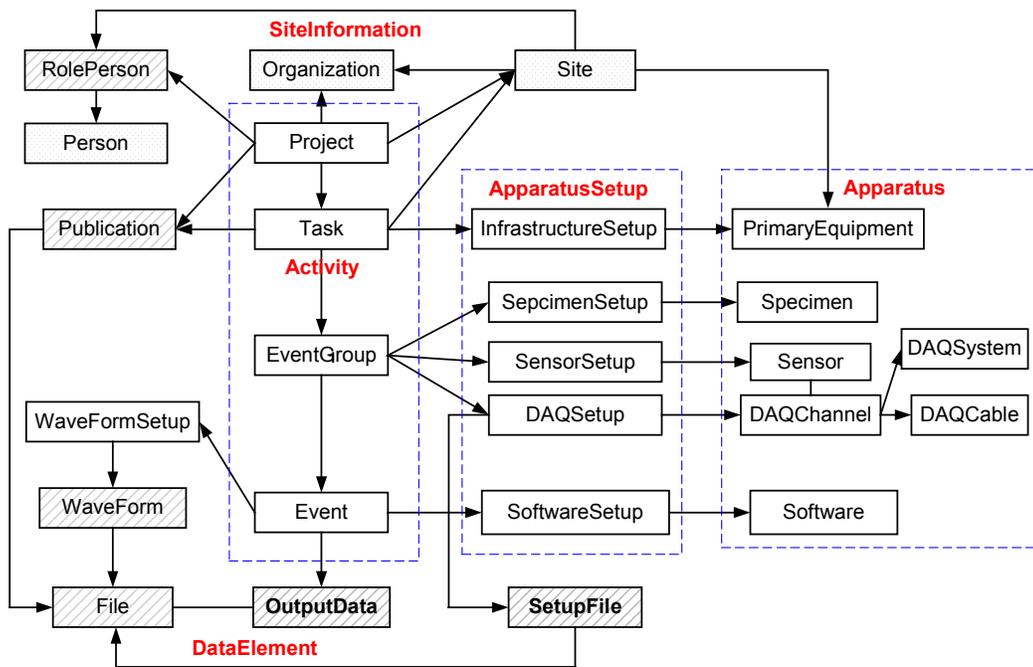


Figure 3 – High-level Class Diagram of the Reference Data Model

Overview of the Reference Data Model

Although the NEESgrid reference data model is intended to focus on the data requirements for shake table experiments, large portion of the model is of sufficient generality to be used for other types of experiments, such as centrifuge, pseudo-dynamic tests, etc.. The following gives a brief description about the six groups of base classes defined in the reference data model.

- **SiteInformation:** SiteInformation is modeled to represent a typical experiment site and its associated personnel, facilities, equipments and other information. In the reference data model, Site is modeled as the aggregation of other component classes, such as RolePerson, Organization, PrimaryEquipment, and SecondaryEquipment. This group of classes is intended to be associated with the site specifications database (Kutter et al. 2004) that is currently under development by the NEES community.
- **Activity:** Activity is designed to support project level modeling. Common properties of Activity classes include objective, description, start time and end time. The duration of an Activity can be calculated by using start time together with end time. The start time of an Activity can also be used to identify which Activity happens first, i.e. to sort out the sequence of several Activities. There are four basic types of Activities, including Project, Task, EventGroup and Event. A **Project** is a collection (organized group) of tasks designed to achieve specific goals and objectives of a project. A **Task** belongs to a particular Project and contains one or more EventGroups. Each Task typically serves a specific role in a Project. An **EventGroup** is defined as a collection of Events. Any change to the data acquisition setup, sensor setup or the specimen setup would initiate a new EventGroup. An **Event**, which is the atomic level of Activity, refers to each single run of an experiment or a simulation.
- **Apparatus:** Apparatus is defined as any equipment, specimen, or computational resource that may be used in an Activity. In the current version of the reference data model, the direct subclasses of Apparatus include Specimen, PrimaryEquipment, SecondaryEquipment and TertiaryEquipment. Explicit modeling of Specimen is not considered in the reference model (Peng and Law 2004a). Only the most basic modeling is provided (as a collection of descriptive files, drawings, and/or photos). This design reflects current approach used to describe specimen in earthquake engineering experiments. However, if so desired, the Specimen class can be extended to support other, more detailed, models.
- **ApparatusSetup:** ApparatusSetup models the arrangement and setup of apparatus for all experiments. Universal modeling of ApparatusSetup is very difficult. Not only are there different types of experiments (such as shake table, pseudo-dynamic tests, centrifuge, and tsunami) and different materials (such as concrete, steel, wood, etc.), but also the geometry of specimen, the arrangement of sensors, and the configuration of PrimaryEquipment may be too complicated and cumbersome to model. For example, the “as-built” locations of sensors may be different from the “design” locations, and the exact physical locations (i.e., the coordinates x, y, z values) of sensors may be very hard to be recorded. Therefore, current design for the ApparatusSetup model focuses on tools and methodologies that can capture and organize CAD drawings, sketched drawings and notes, photos, narrative descriptions, and electronic notes, etc.
- **DataElement:** DataElement represents all types of data that may be generated or processed during an Activity. The DataElement normally serves as Input/Output to an Activity. Types of DataElement include text document, publication, earthquake record, photo, CAD drawing, movie, etc. In NEESgrid data/metadata effort, it is assumed that the data is saved in or translated into computer-readable format. Therefore, a DataElement object is represented in the format (such as a file) that can be saved in computer memory, on disks, or in some kind of data storage repository.
- **ComplexDataType:** ComplexDataType is defined in the reference data model to represent any data type that is not a simple data type such as integer, float, Boolean, or character string. Some of the example subclasses of ComplexDataType include Folder, RolePerson, Unit, DateTime, Location, Measurement, Angle, etc..

Features of the Reference Data Model

Since the reference NEESgrid data model is developed specifically to support collaborative earthquake engineering experiments and simulations, it has several unique features. Details of these

features have been discussed by Peng and Law (2004c). The following briefly describes two important but unique features of the developed reference NEESgrid data model.

The reference data model explicitly models certain Activities that are carried out at multiple Sites. Figure 4 shows an example project that has a single site Task (e.g. Task1) and a MultiSiteTask (e.g. M_Task1). The MultiSiteTask M_Task1 has Tasks that are undertaken at both Site1 and Site2. The MultiSiteEvent M_E1 has an Event E2 at Site1 and an Event E4 at Site2, and the MultiSiteEvent M_E2 has an Event E3 at Site1 and an Event E5 at Site2. As shown in Figure 4, although Project does not directly *contain* Task2 that takes place at Site2, Task2 can still be accessed from the Project since M_Task1 *contains* Task2. This design enables the support of the types of experiments that are carried out either simultaneously or independently at several Sites.

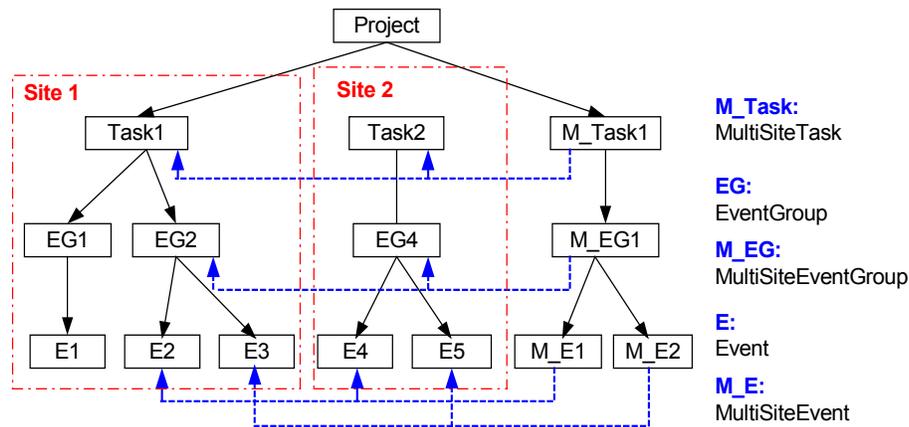


Figure 4 – Layout of an Example Project

The reference data model includes a sensor model that is designed specifically to support earthquake engineering experiments. The data acquisition equipment is modeled as a collection of classes, including Sensor, DAQCable, DAQChannel, and DAQSystem. Figure 5 shows the relationships and the slots of these classes. Typically a data acquisition system involves at least three main components: (1) the sensors which respond to a physical stimulus and generate analog voltage signals; (2) a DAQchannel which receives the signal and uses predefined filter, gain, offset, excitation, sensitivity information for Analog-to-Digital and Engineering Unit conversions; and (3) a PC unit which uses some communications link to retrieve the data.

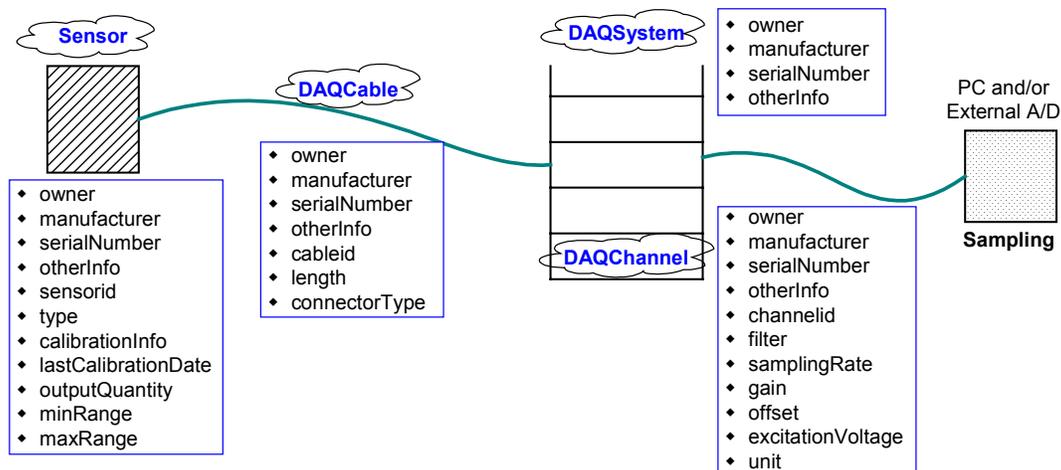


Figure 5 – Setup and Modeling of DAQ Devices

VALIDATION AND USABILITY TEST

The usability of the reference data model has been tested with legacy experimental data (Peng and Law 2004b). For the validation tests, Protégé (Gennari et al. 2002) was employed as the interface to input experimental data and local file system was used as the storage medium. Legacy data from several earthquake experiments have been used. For illustration purpose, this paper focuses on the data sets obtained from a Mini-MOST experiment (Nakata et al. 2004).

Mini-MOST Experiment

The main purpose of the Mini-MOST experiment is to show the capability of the various NEESgrid service components using a small-scale physical experimental setup (Nakata et al. 2004). The Mini-MOST experimental hardware, as implied by its name, is small in size and can be easily packed and shipped to experimental sites. The Mini-MOST experiment provides a platform for researchers to become familiar with the NEESgrid software and to gain first-hand experience in using the NEESgrid services. For the validation test of the reference data model, the data sets generated from a particular Mini-MOST test on February 28, 2004 at University of Illinois at Urbana-Champaign (UIUC) are employed.

Inputting Experimental Data

For validation purpose, data generated from the Mini-MOST experiment was ingested using Protégé (Gennari et al. 2002) and saved as files in a local file system. As we mentioned earlier, besides defined classes and their properties, Protégé can also be used to handle instances of data. Data related to Mini-MOST experiment can be inputted using the slots (properties) as defined in the reference data model. If a slot is defined as primitive type, such as Integer, Real Number, Time, or String, etc., we can simply type in the value. If a slot is defined as Objects, then we can either choose a previously created object or create a new one. If a slot is defined as of type “URI” (Universal Resource Identification, which would normally refers to a file), we can save the file in a particular location and then enter the URI for the file location. Other types of objects, such as Task, EventGroups, Event, SensorSetup, InfrastructureSetup, Sensor, Specimen, and etc., can also be created and inputted through Protégé interface. Therefore, all the objects related to Mini-MOST experiment can be created and saved. The metadata, i.e. information about the data, are saved as an OWL (<http://www.w3.org/2001/sw/WebOnt/>) file. Other experimental data, such as specimen photos and sensor readings, can be stored in a file on a web server with its URI saved in the OWL file.

Browsing Experimental Data

For validation purpose, we implemented a project viewer to retrieve the saved data and to view the data on a web browser according to the data model. The program is implemented using Java Servlet technology (<http://java.sun.com/products/servlet/>); and the parsing of the OWL file is handled by using Jena (McBride et al. 2004).

Since the data is organized according to the project it belongs to, the project viewer is started with a webpage that contains a list of projects. When we click on a particular project, say miniMOST-1, the details of the project will be shown on the browser. The details include the values for all the slots defined for Project class. If a slot is defined to have data with primitive data type, the value is shown directly in the webpage. If a slot is defined to be a reference to other object, then a link is shown in the webpage. We can then follow these links to browse the details of other objects. For example, we can navigate and access all the Tasks that belong to the Project by following specific links. We can further follow links to access EventGroups, as each Task in a project may contain one or more

EventGroups. Since an EventGroup is defined as a collection of Events, each Event can be accessed from the EventGroup object. The details of an Event can be accessed and shown in a browser (see Figure 6 for an example named miniMOST_test_0228). An Event, which is an entity at the atomic (or lowest) level of an Activity, refers to a single run of an experiment or a simulation. Experimental results, such as SensorReading, can be accessed from an Event object, as shown in Figure 7.

The details of other objects, such as InfrastructureSetup, can also be retrieved by navigating the project viewer. An InfrastructureSetup object essentially is a collection of texts, documents (in the format of Word, PDF, Excel, etc.), figures and drawings stored as files. Files are saved in a web server and their URIs are saved as metadata. The files can be dynamically downloaded and shown on a web browser, as illustrated in Figure 8.

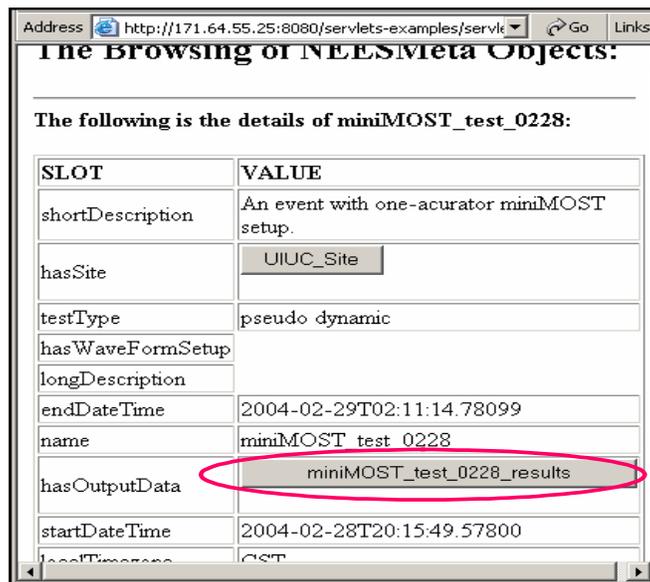


Figure 6 – Detailed Display of the Event miniMOST_test_0228

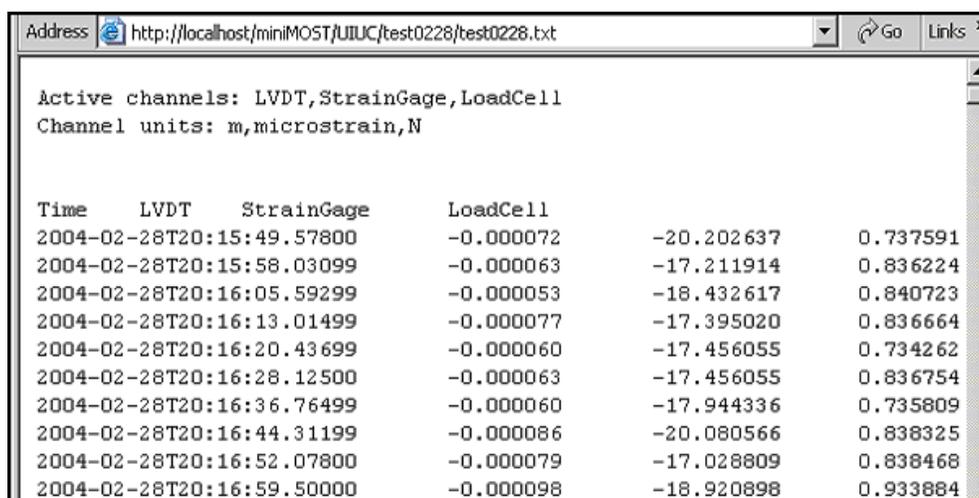


Figure 7 – Access of SensorReading for the Event miniMOST_test_0228

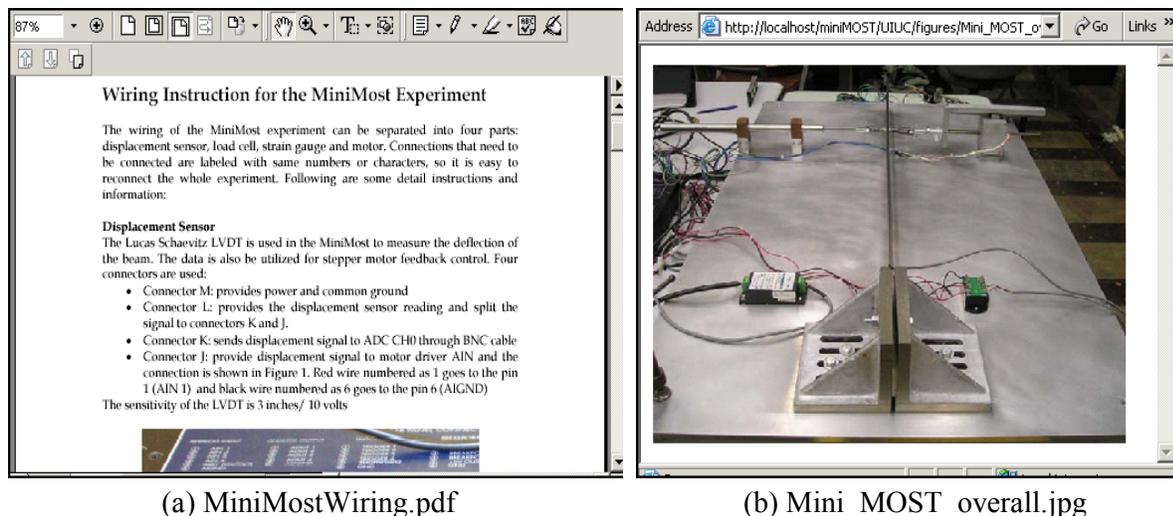


Figure 8 – Access of Files Representing the Infrastructure Setup

SUMMARY AND DISCUSSIONS

This paper has discussed the approach for the development of a reference NEESgrid data model. Brief description of the developed data model and some of its features are also presented. Six base classes are presented and the relationships among these classes are defined. These classes represent the essential elements to support the end-to-end solution of NEESgrid data efforts. We believe the proposed reference data model is sufficiently flexible and extendible: (1) new classes can easily be introduced; (2) the slots of a particular class can be added, deleted, or modified; (3) and the relationships among the classes can be altered. Other models, such as specimen model, unit model, geometry/location model, and Site model, can be appended to (or even replace certain parts of) the reference data model.

To validate the reference data model, we have populated the model with the Mini-MOST experimental data provided by UIUC. This validation process helps evaluate the completeness, flexibility and usability of the data model. The usability test has demonstrated that the data model is sufficiently comprehensive to save and organize all the Mini-MOST data. In addition, as the experimental data are organized according to the data model, browsing and accessing them are fairly intuitive and straightforward. Efforts will continue to validate, evaluate and refine the reference data model using other experimental projects and data. Furthermore, a general project browser is currently under development.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the active collaboration and contributions of the NEESgrid's Data/Metadata task committee members: Bill Spencer (University of Illinois, Urbana-Champaign), Chuck Severance and Jim Eng (University of Michigan), Kincho H. Law and Jun Peng (Stanford University), Jean-Pierre Bardet and Jennifer Swift (University of Southern California), Andrei Reinhorn (State University of New York, Buffalo), Gokhan Pekcan (University of Nevada, Reno), Hank Ratzesberger (University of California, Santa Barbara), Ken Ferschweiler (Oregon State University), and Lelli Van Den Einde (University of California, San Diego). The authors would also like to thank Prof. Gokhan Peckan and Dr. Patrick Laplace of University of Nevada, Reno; Prof. Steve

Mahin, Bozidar Stojadinovic, Greg Fenves, and Dr. Frank McKenna of University of California, Berkeley; Professor Andrei Reinhorn of SUNY-Buffalo; and Prof. Jerome Hajjar and Catherine French of University of Minnesota for their time to discuss the data and metadata issues related to earthquake engineering experiments and simulations. Any opinions, findings, and conclusions or recommendations expressed in this material are, however, those of the authors and do not necessarily reflect the views of others and the National Science Foundation.

REFERENCES

Arlow, J. and Neustadt, I. (2001). UML and the Unified Process: Practical Object-Oriented Analysis and Design, Addison-Wesley Pub Co., Boston, MA.

Gennari, J., Musen, M. A., Ferguson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F. and Tu, S. W. (2002). The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. Technical Report SMI-2002-0943, Stanford Medical Informatics, Stanford University, http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html.

Kutter, B. L., Wilson, D. W., Pancake, C. and Haerer, S. (2004). Introduction to the Site Specifications Database. NEES, <http://nees.orst.edu/IT/site.specs.db/cohorts/Introduction.pdf>.

Law, K. H., Jouaneh, M. K. and Spooner, D. L. (1987). Abstraction Database Concept for Engineering Modeling. *Engineering with Computers* **2**, 79-94.

Law, K. H. (1988). Conceptual Database Design for Engineering Modeling, in *Managing Engineering Data: Emerging Issues. Proceedings of the ASME International Computers in Engineering Conference and Exhibition*, San Francisco, CA.

Law, K. H., Wiederhold, G., Siambela, N., Sujansky, W., Zingmond, D., Singh, H. and Barsalou, T. (1991). Architecture for Managing Design Objects in a Sharable Relational Framework. *International Journal of Systems Automation: Research and Applications (SARA)* **1**, 47-65.

McBride, B., Boothby, D. and Dollin, C. (2004). An Introduction to RDF and the Jena RDF API. http://jena.sourceforge.net/tutorial/RDF_API/index.html.

Nakata, N., Yang, G. and Spencer, B. F. (2004). System Requirements for Mini-MOST Experiment. NEESgrid Report, http://www.neesgrid.org/mini-most/Mini_MOST_requirements_revised3.pdf.

Noy, N. F. and McGuinness, D. L. (2002). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University, Stanford, CA. http://protege.stanford.edu/publications/ontology_development/ontology101.html.

Peng, J. and Law, K. H. (2004a). A Brief Review of Data Models for NEESgrid. Technical Report NEESgrid-2004-01, http://www.neesgrid.org/documents/TR_2004_01.pdf.

Peng, J. and Law, K. H. (2004b). Validity and Usability of the NEESgrid Reference Data Model. Technical Report NEESgrid-2004-44, http://www.neesgrid.org/documents/TR_2004_44.pdf.

Peng, J. and Law, K. H. (2004c). Reference NEESgrid Data Model. NEESgrid Technical Report.

Rumbaugh, J. R., Blaha, M. R., Lorensen, W., Eddy, F. and Premerlani, W. (1990). Object-Oriented Modeling and Design, Prentice Hall.